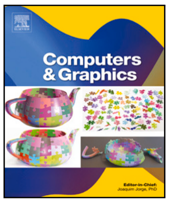
Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag



Special Section on VSI: EuroVA 2025 SI

Autoencoder-based regularization methods for parametric and inverse projections[☆]

Frederik L. Dennig [a,*], Daniela Blumberg [a], Nina Geyer [a], Yannick Metz [a,b]

[a] Department of Computer and Information Science, University of Konstanz, Konstanz, 78464, Germany
[b] AI Center, ETH Zürich, Zürich, 8092, Switzerland

ARTICLE INFO

Keywords:
Inverse projections
Parametric projections
Dimensionality reduction
Data visualization
Autoencoders
Machine learning

ABSTRACT

Neural networks are used to create parametric and invertible multidimensional data projections. In this context, parametric projections enable the embedding of previously unseen data points without requiring a complete recomputation of the projection, while invertible projections allow for the reconstruction or generation of data in the original space. In this paper, we investigate the use of autoencoder (AE) architectures for simultaneously learning parametric and inverse mappings independent of the underlying dimensionality reduction method. We introduce and compare three regularization methods for autoencoder architectures designed to learn a forward mapping into two-dimensional space induced by the projection as well as inverse mappings back into the original feature space. To evaluate their performance, we conduct a systematic study on six datasets of varying dimensionality and structural complexity, using the established projection techniques t-SNE and UMAP as training targets. Our evaluation combines both quantitative metrics and qualitative assessments. The results demonstrate that AEs, particularly when trained with Kullback–Leibler divergence regularization, can achieve high-quality reconstructions while providing users with control over the degree of smoothing in the projection. Compared to disjoint neural networks, AE architectures yield superior generative capabilities for out-of-distribution samples, while still providing comparable reconstruction quality and parametric projection accuracy. This highlights their potential for interactive data generation in use cases such as classifier evaluation and counterfactual creation.

1. Introduction

Dimensionality reduction (DR) methods are commonly used to create low-dimensional projections for visual analysis of high-dimensional data [1]. DR methods reduce high-dimensional data to a lower-dimensional projection, usually to 2D or 3D, while trying to preserve relationships, i.e., distances and neighborhoods of data samples [2]. Low-dimensional projections are a ubiquitous tool in visual data analysis, e.g., for representing patterns such as clusters. However, a shortcoming of these approaches is that the result can be hard to interpret since the relationships are often projected non-linearly, which is the case for many well-established methods. To address this, researchers proposed to enhance the visualization through layout enrichments, showing the distortions and gradients between projected points [2–4]. This helps analysts to perceive and understand the non-linearity inherent in the projections. Yet, most techniques do not supply the means with which such a representation can be created.

Ideally, such methods should be parametric and invertible. (1) Parametric methods use a model with learnable parameters to control the mapping between high- and low-dimensional spaces. Doing so allows the projection of new data points, real and synthetic, without recalculating all pairwise relationships. This is faster and keeps the projection stable [5]. (2) Invertible methods have a smooth mapping from the projection space back to the original high-dimensional space, allowing users to generate new data from any arbitrary position of the projection [6], e.g., for interactive counterfactual generation [7]. However, DR methods, such as t-SNE [8] and UMAP [9], are generally not parametric or invertible. In this paper, we report that through the use of neural networks (NNs) and the choice of appropriate regularization methods, we can learn projections, effectively making them *parametric* and *invertible*.

In recent years, neural networks have been used to learn mappings from high-dimensional space to the projection space [10,11],

☆ This article is part of a Special issue entitled: 'EuroVA 2025 SI' published in Computers & Graphics.
* Corresponding author.
E-mail addresses: frederik.dennig@uni-konstanz.de (F.L. Dennig), daniela.blumberg@uni-konstanz.de (D. Blumberg), nina.geyer@uni-konstanz.de (N. Geyer), yannick.metz@uni-konstanz.de (Y. Metz).

https://doi.org/10.1016/j.cag.2026.104552
Received 15 October 2025; Received in revised form 16 February 2026; Accepted 17 February 2026
Available online 17 February 2026

and vice versa [3]. However, while these approaches can be applied to arbitrary projections, they only consider each mapping direction individually. We explore and evaluate different regularization methods for autoencoders (AEs) [12] to support parametric mapping and inversion jointly through their encoder–decoder architecture, continuing our earlier work [13]. More specifically, we compare the individual *feed-forward NNs* proposed by Espadoto et al. [3], Appleby et al. [14], and Hinterreiter et al. [11] with *standard* AEs and *variational* AEs with custom loss functions for representing the projection space in the latent space. We measure their ability to parametrically project and inversely project data points and compare them qualitatively. Additionally, we visually assess their outputs for out-of-distribution samples. To evaluate the effect of different regularization methods on the inverse projection, we analyze the smoothness of the projection through a quantitative and visual assessment of the respective gradient maps. Since inverse projections are commonly used for the creation of counterfactuals [7] and the evaluation of classification models in general, we also compare the different decision maps [3] created by each approach. Overall, in this paper, we contribute the following:

(1) Three *AE-based NN architectures* leveraging different regularization methods for creating parametric and invertible projections.
(2) An *evaluation* comparing the three architectures *quantitatively* and *qualitatively* on six datasets of varying dimensionality using t-SNE and UMAP.
(3) A *discussion* of the properties of the individual architectures and the effect of hyperparameters, along with *recommendations* for generating parametric and inverse projections.
(4) Our *analysis, results, and source code* at OSF and GitHub for *reproducibility*.

With this work, we aim to improve interpretation and explainability of DR methods [15] while also exploring their use as visual interactive generative models.

2. Related work on multidimensional projections

First, we introduce some notation. Let $D = \{x_i\}_{1\leq i\leq n}$ be a high-dimensional dataset with d dimensions and n samples $x_i \in \mathbb{R}^d$. A *projection* method P maps each item in D to a lower-dimensional representation, i.e., $P : D \rightarrow \mathbb{R}^q$. In general, $P(D) = \{P(x_i)|x_i \in D\} = \{y_i\}_{1\leq i\leq n}$, where $P(D) \subset \mathbb{R}^q$ with $q \ll d$ (in our case, $q = 2$). Thus, $P(D)$ can be visualized as a 2-dimensional scatterplot.

Multidimensional projections aim to map high-dimensional data into a lower-dimensional space while retaining its most relevant structures, such as clusters. Projection methods have been extensively studied and assessed in several surveys [2,16–18]. Various types of projection methods exist, differing in how they capture relationships and structures in the data. They can be classified into *linear* and *non-linear* methods [18,19]. Additionally, they either preserve *global* structure or *local* neighborhoods. Linear projection methods like PCA [1] can be computed very efficiently and preserve the global structure of the data. MDS [20] is an example of a global, non-linear projection method. There exist many *non-linear* methods that put stronger weight on preserving *local* neighborhood structures at the expense of global structure fidelity. Such methods include t-SNE [8] or UMAP [9]. Generally, autoencoder-based approaches also fall into the category of non-linear projection methods, since they learn non-linear mappings through stacked neural network layers [21].

Regularization is essential because unconstrained latent spaces can become *discontinuous*, causing nearby inputs to map to distant latent codes, which reduces both interpolation and reconstruction capabilities. Enforcing *smoothness* ensures that small latent perturbations yield predictable changes in output, improving invertibility. The latent spaces of regular autoencoders are *discontinuous* [22], and thus, these may yield samples that are less interpretable and inverse projections with high local gradients. In this paper, we address these shortcomings.

2.1. Parametric projections

Standard non-linear DR methods [8,9,20] are non-parametric, needing complete recalculation when projecting new data points. This makes them impractical for dynamic datasets, out-of-sample projections, and large-scale applications due to their high computational cost. Parametric approaches [23] address these limitations, e.g., by using NNs to learn mapping functions into lower-dimensional space [22,24]. Van der Maaten used a feed-forward NN to build *parametric t-SNE* [24]. Similarly, *parametric UMAP* [5] uses NNs designed to replace the non-parametric embedding step. These ideas have been extended by *HyperNP* [14], which uses NNs to approximate projection techniques across hyperparameters (e.g., perplexity for t-SNE), allowing for their interactive exploration. *ParaDime* [11] streamlines the creation of NN-based DR approaches by proposing a grammar specific to parametric DR, and thus, extending the use of NNs to create parametric projections beyond t-SNE and UMAP.

By training NNs to infer 2D coordinates for input domain data points, Espadoto et al. [10] showed that NNs with sufficient size can effectively approximate many existing non-parametric methods. This is feasible for large datasets, since batch-wise training enhances scalability, while enabling efficient embedding of unseen data, and supports additional loss functions to balance global and local structure preservation. However, directly optimizing projection objectives in NNs without pre-training can lead to poor local minima [24]. Yet, Lai et al. [25] demonstrated that combining the Adam optimizer [26] with the *LeakyReLU* [27] activation function can mitigate this issue. An alternative strategy is to train NNs to mimic a given projection by using paired training data, i.e., high-dimensional samples together with their 2D projections obtained from a user-given method [10].

Reichmann et al. [28] use parametric out-of-sample extensions for a variety of DR methods to improve the projection of large datasets. In this context, they report that the quality of t-SNE and UMAP projections varies most across reference set sizes, likely due to overfitting on small sets caused by their focus on preserving local neighborhoods. In contrast to the previous work, we address issues with regard to overfitting and low out-of-sample performance by proposing the use of autoencoders with latent space regularization methods.

2.2. Inverse projections

Inverse projections are functions $P^{-1} : \mathbb{R}^q \rightarrow \mathbb{R}^d$ which are smooth and minimize the cost $\sum_{x\in D} \|P^{-1}(P(x)) - x\|$ with projection P and $\|\cdot\|$ denoting the L_2 norm. Only a handful of projections are inherently invertible, e.g., UMAP [29]. An inverse mapping is not explicitly available for most other projection methods, requiring alternative methods, such as NNs [3]. One of the early inverse projection methods introduced a global interpolation-based technique to generate new data points [6]. Later, iLAMP [30], which builds on the projection technique LAMP [31], addresses inverse projection by focusing on preserving local relationships. Amorim et al. [32] later refined this by using a radial basis function kernel for interpolation. Blumberg et al. [33] invert MDS projections by leveraging geometrical relationships between data points through multilateration. Recently, deep learning was employed for creating inverse projections. The feed-forward NNs proposed by Espadoto et al. [3] learn the mapping from the projection space back to the high-dimensional space. However, such a network is then capable of learning either the parametric or the inverse projection but not both end-to-end.

The inverse capability is valuable for automatically and interactively generating high-dimensional data samples from low-dimensional representations in a variety of applications. For example, users can interactively explore the projected space by generating high-dimensional data from any 2D position. Such functionality is applicable to synthetic data generation [3], exploring projection artifacts [2], and creating counterfactual explanations [7].

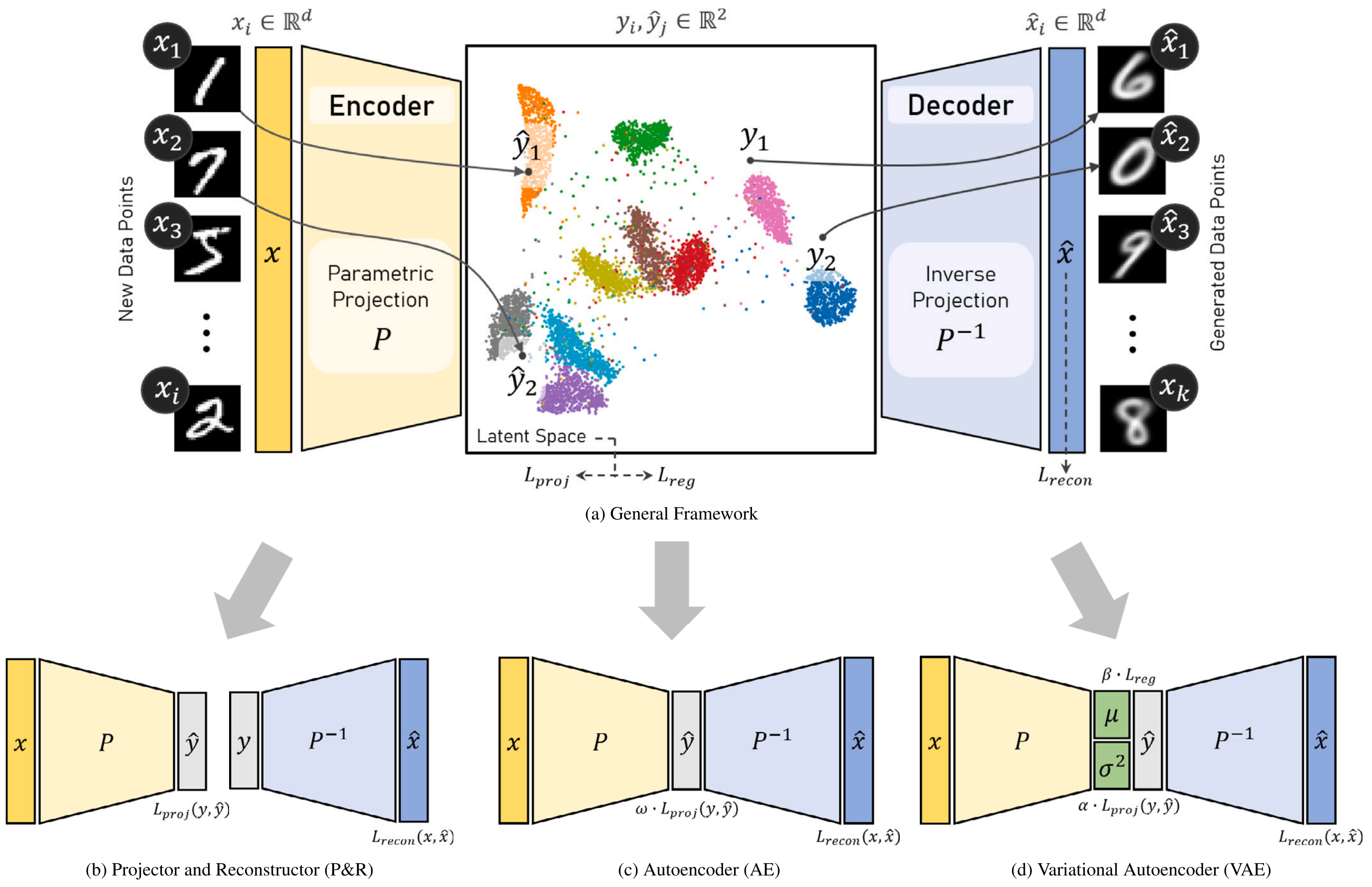


Fig. 1. In our framework (a), the *encoder* network learns a parametric projection P mapping new data record x_i into 2D space (as $\hat{y}_i$). The *decoder* learns the inverse projection P^{-1} generating a high-dimensional sample $\hat{x}_i$ from any 2D point y_i. Our approaches can utilize the losses: $\mathcal{L}_{\text{recon}}$, ensuring correct reconstruction; $\mathcal{L}_{\text{proj}}$, aligning $\hat{y}$ with points of the projection; and $\mathcal{L}_{\text{reg}}$, regularizing the latent space to avoid discontinuity. We evaluate three architectures: (b) Two feed-forward NNs, i.e., *Projector and Reconstructor* (P&R), are trained separately to learn a forward and inverse mapping between data and projection spaces without end-to-end optimization or latent space regularization. (c) An *Autoencoder* (AE) is trained end-to-end, with a joint loss that combines reconstruction accuracy and a latent-space alignment term to match a target projection. (d) A *Variational Autoencoder* (VAE) extends the AE by introducing stochastic latent representations, enforcing structure through a KL divergence term and aligning the latent space to a target projection using either sampled or mean-based loss.

Additionally, invertible approaches are required to create *decision maps* [34,35] that enable visually inspecting the decision boundaries of a classification model. Thereby, each pixel of a 2D map is transformed into a high-dimensional representation by the inverse projection, classified by the trained model, and then colored according to the assigned class label. Same-colored regions show the decision zones, and neighboring pixels of different colors reveal the decision boundary of the classifier. Comparing decision maps generated with different inverse projections also allows for the visual assessment of their reconstruction capabilities.

In addition to the visual assessments of inverse projections such as decision maps, the quality can be measured quantitatively either by the mean-squared error (MSE) or binary cross-entropy (BCE) between original and reconstructed data samples. To evaluate out-of-distribution capabilities, the gradient map technique [3] can be employed. It determines the gradients of the inverse projection by measuring the rate of change (or distance) in the low-dimensional space in relation to the rate of change implied by the inverse projection through the reconstruction of samples.

2.3. Autoencoders in dimensionality reduction

AEs are a class of NNs designed for unsupervised learning of efficient data representations [12]. Autoencoders are already well-established in the context of DR [22]. However, a standard AE will learn a latent space representation optimized for data compression and feature extraction. Thus, modified loss functions were proposed to impose a structure on the latent space of an AE. For example, *SSNP* [36] and *ShaRP* [37] integrate pseudo-labels derived from clustering into their loss functions. These approaches do not allow for inverting arbitrary user-defined projections. In this paper, we address this limitation by proposing AEs that can approximate a chosen predefined projection.

3. Training autoencoders for multidimensional projections

We evaluate three neural network architectures for learning parametric projections and reconstructions (see Fig. 1). First, the *Projector and Reconstructor (P&R)* consists of two independently trained feed-forward networks that learn forward and inverse mappings without end-to-end optimization or latent-space regularization. Second, an *Autoencoder (AE)* jointly learns projection and reconstruction in an *end-to-end* manner, combining reconstruction accuracy with a projection loss to match a target projection. Third, a *Variational Autoencoder (VAE)* extends the AE by introducing a probabilistic latent space regularized via Kullback–Leibler divergence, while additionally aligning the latent distribution to a target projection using either sampled or mean-based supervision.

All methods share a reconstruction loss term $\mathcal{L}_{\text{recon}}$. A common choice for this loss is the mean squared error (MSE), formally defined as

$$\text{MSE}(x_i, \hat{x}_i) = \frac{1}{d}\|x_i - \hat{x}_i\|^2, \quad (1)$$

where d is the data dimensionality. As a common alternative, binary cross-entropy (BCE) is used when the data values are in the interval $[0,1]$, which is the case for image datasets in Table 2. The BCE is formally defined as

$$\mathrm{BCE}(x_i,\hat{x}_i) = -\frac{1}{d}\sum_{j=1}^{d}\left[\, x_{i,j}\log \hat{x}_{i,j} + (1-x_{i,j})\log(1-\hat{x}_{i,j})\,\right]. \tag{2}$$

When BCE is used as reconstruction loss, the decoder output layer applies a sigmoid activation to ensure outputs lie in $[0,1]$.

3.1. Projector and reconstructor (P&R)

We train two standard feed-forward NNs (see Fig. 1(b)) to combine the approaches by Appleby et al. [14] (denoted as P) and Espadoto et al. [3] (denoted as P^{-1}). This structure is, strictly speaking, not an AE, since it is not trained *end-to-end*. However, this setup provides a baseline for our experiments for comparing projection and reconstruction quality without any implicit or explicit regularization. We train a *projector* network learning the parametric projection P, enabling the addition of new data to the projection through its loss function:

$$\mathcal{L}_{proj}(y,\hat{y}) = \mathrm{MSE}(y,\hat{y}) \tag{3}$$

In all cases, we use the mean squared error $\mathrm{MSE}(P(x),\hat{y})$ as projection loss. Independently, we train a *reconstructor* network learning to invert the projection space via the loss function $\mathcal{L}_{recon}(x,\hat{x}) = \mathrm{MSE}(P^{-1}(y),\hat{x})$ or, dependent on the dataset, use binary cross-entropy $\mathrm{BCE}(x,\hat{x})$ as formulations of the reconstruction loss. During training, P and P^{-1} are learned by the individual networks, *projecting* and *reconstructing* the dataset, i.e., $P(x) = y$ and $P^{-1}(y) = x$. The neural network architecture can be adapted to the data type, i.e., convolutional layers for images and fully connected for vector data.

3.2. Autoencoder (AE)

Training individual networks can lead to projection and reconstruction artifacts, including noise amplification and false structures. These artifacts are prevalent for out-of-distribution samples [13]; we propose using AEs (see Fig. 1(c)).

We train AEs to *jointly* create a parametric projection P and inverse projection P^{-1}. An AE consists of an *encoder* and a *decoder*. The encoder $\mathrm{Enc}: \mathbb{R}^d \to \mathbb{R}^q$ learns a mapping from the high-dimensional input space into a lower-dimensional latent space $\hat{Y} = \{\hat{y}_i\}_{i=1}^n$, with $\hat{y}_i = \mathrm{Enc}(x_i) \in \mathbb{R}^q$. In contrast, the decoder $\mathrm{Dec}: \mathbb{R}^q \to \mathbb{R}^d$ aims to map these latent representations back into the original input space $\hat{x}_i = \mathrm{Dec}(y_i) \in \mathbb{R}^d$. A standard AE aims to learn a compressed latent encoding y_i while reconstructing the input x_i as accurately as possible. During training, we feed the input x_i forward through the encoder and decoder to obtain a reconstruction $\hat{x}_i$, then compute a reconstruction loss $\mathcal{L}_{recon}(x_i,\hat{x}_i)$. This error is *backpropagated* through the AE, and its parameters are updated accordingly.

In AEs, the encoder and decoder are trained *end-to-end* as one NN. The encoder of the AE learns to project the data points, while the decoder learns the inverse projection. To impose a structure on the latent space, we define the loss as

$$\mathcal{L}_{AE}(x,\hat{x},y,\hat{y}) = \mathcal{L}_{recon}(x,\hat{x}) + \omega\cdot\mathcal{L}_{proj}(y,\hat{y}), \tag{4}$$

where $\mathcal{L}_{recon}(x,\hat{x})$ denotes the reconstruction loss. Similarly to the P&R we use $\mathrm{MSE}(x,\hat{x})$ or $\mathrm{BCE}(x,\hat{x})$ for $\mathcal{L}_{recon}(x,\hat{x})$ and $\mathrm{MSE}(y,\hat{y})$ for $\mathcal{L}_{proj}(y,\hat{y})$. To enable the AE to learn a given projection, we modify its loss function by adding a component $\omega\cdot\mathcal{L}_{proj}(y,\hat{y})$ to force its latent space to conform with $\mathrm{MSE}(y,\hat{y})$. The trade-off parameter between projection and reconstruction loss $\omega \in \mathbb{R}^+$ determines the degree to which the latent space conforms to the projection, i.e., balances between the projection and reconstruction quality. We discuss the selection of ω in Section 4.2.

3.3. Variational autoencoder (VAE)

VAEs [38] extend standard AEs by introducing a probabilistic latent space. Instead of mapping inputs to fixed latent representations, the encoder produces parameters of a distribution, from which latent variables are sampled. The decoder then reconstructs inputs from these samples. Training balances reconstruction quality with latent space regularization by incorporating the Kullback–Leibler divergence (D_{KL}), a measure of the difference between the approximate posterior and the prior probability distribution, typically a standard Gaussian. We use modern training methods for *deep* AEs, including mini-batches and an optimizer based on stochastic gradient descent [26]. We use standard loss functions, like the MSE, BCE, and Kullback–Leibler divergence.

Instead of directly predicting the latent variable $\hat{y}$, the encoder of the VAE predicts the parameters of the normal distribution μ and σ^2 (see Fig. 1(d)). In training, $\hat{y}$ is sampled from a 2D normal distribution, i.e., $\hat{y} \sim \mathcal{N}(\mu,\sigma^2)$ with $\hat{y},\ \mu \in \mathbb{R}^q$ and $\sigma^2 \in \mathbb{R}^q_+$. The VAE is still trained end-to-end, using the *reparameterization trick*, to backpropagate through the sampling operation [38]. So, instead of sampling $\hat{y} \sim \mathcal{N}(\mu,\sigma^2)$ directly, we rewrite the sample as a deterministic function of the network outputs and a fixed noise variable $\hat{y} = \mu + \sigma \odot \epsilon$, with $\epsilon \sim \mathcal{N}(0,I)$. We incorporate the *evidence lower bound loss (ELBO)* used for VAEs consisting of the reconstruction term $\mathcal{L}_{recon}$ and the KL-divergence regularizer $\mathcal{L}_{reg}$, while $\mathcal{L}_{proj}$ is an additional supervision term beyond the standard ELBO. The overall loss is defined as

$$\begin{aligned}\mathcal{L}_{VAE}(x,\hat{x},y,\hat{y},\mu,\sigma^2) = \mathcal{L}_{recon}(x,\hat{x})&\\ +\ \alpha\cdot\mathcal{L}_{proj}(y,\mu,\sigma^2)&\\ +\ \beta\cdot\mathcal{L}_{reg}(\mu,\sigma^2).&\end{aligned} \tag{5}$$

The reconstruction loss $\mathcal{L}_{recon}$ is defined analogous to the P&R and AE variants. For VAEs, we explore two alternative latent loss formulations $\mathcal{L}_{proj}$. First, the sample-based loss $\mathcal{L}^{(\hat{y})}_{proj}$ is computed between the reference projection y and a random sample $\hat{y} \sim \mathcal{N}(\mu,\sigma^2)$, which is formally defined as

$$\mathcal{L}^{(\hat{y})}_{proj}(y,\mu,\sigma^2) = \mathrm{MSE}(y,\hat{y}) \text{ with } \hat{y} \sim \mathcal{N}(\mu,\sigma^2). \tag{6}$$

All results pertaining to this configuration are reported using the abbreviation VAE-$\hat{y}$. The second variant, the mean-based loss $\mathcal{L}^{(\mu)}_{proj}$ is computed between y and the mean μ of the latent distribution as

$$\mathcal{L}^{(\mu)}_{proj}(y,\mu) = \mathrm{MSE}(y,\mu). \tag{7}$$

The results of this configuration are presented with the abbreviation VAE-μ. Note that, during training, we sample from the diagonal Gaussian posterior distribution, which is typical for a VAE. The Kullback–Leibler divergence (D_{KL}) is a dissimilarity measure between probability distributions; its minimization induces a regularization commonly used to avoid discontinuous latent spaces [38].

To regularize the latent space representing the projection, we use the D_{KL} in $\mathcal{L}_{reg}$, formally defined as

$$\mathcal{L}_{reg}(\mu,\sigma^2) = D_{\mathrm{KL}}(\mathcal{N}(\mu,\sigma^2)\ \|\ \mathcal{N}(0,1)). \tag{8}$$

We follow the framework of the β-VAE [39], which adds an additional β-parameter to balance conformity to the prior normal distribution and reconstruction quality. We discuss the selection of the trade-off regularization parameters α and β in Section 4.2. Similar to Chen et al. [40], we use μ as $\hat{y}$ to create parametric projections, i.e., for inference only, since it is the mean and mode of the normal distribution.

4. Hyperparameter analysis

In the following, we perform a hyperparameter analysis to determine useful settings for all NN architectures, including the choice of activation function and suitable values of ω for AE, as well as α and β for VAE-$\hat{y}$ and VAE-μ.

Table 1
Common activation functions used in neural networks. For each function, we state the formal definitions with x representing the input signal.

Name	Formula	Range
ReLU [41]	$\max(0, x)$	$[0, \infty)$
LeakyReLU [27]	$\max(0, x) + 0.01 \cdot \min(0, x)$	$(-\infty, \infty)$
Tanh [42]	$tanh(x)$	$(-1, 1)$
HardTanh [43]	$\max(-1, \min(1, x))$	$[-1, 1]$
Sigmoid [44]	$1/(1 + \exp(-x))$	$(0, 1)$
SiLU [45]	$x \cdot (1/(1 + e^{-x}))$	$(-\infty, \infty)$

4.1. Choice of activation function

Choosing an activation function in a NN is a key design decision, and the rationale depends on several factors, including the dataset properties and computational efficiency [46]. We tested six common activation functions (see Table 1).

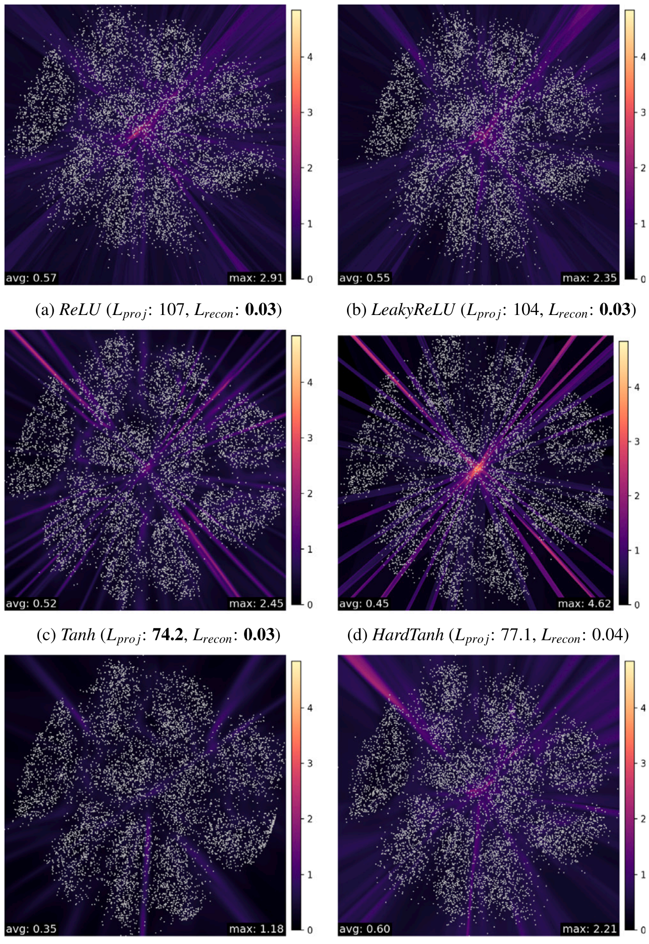
The activation function can have a large impact on the properties of the inverse projection (see Fig. 2). However, this has not been systematically explored in existing work yet. We demonstrate the impact of various activation functions on the gradients of inverse projections, as exemplified using the MNIST (see Table 2) dataset projected with t-SNE.

To evaluate and visualize the effect of the activation function choice, we use gradient maps. For each pixel p, we compute a pseudo-derivative of P^{-1} measuring the Euclidean distance between the inverse projections of neighboring pixels of p along the horizontal and vertical directions as

$$G(p) = \sqrt{\|P^{-1}(p_{\text{left}}) - P^{-1}(p_{\text{right}})\|^2 + \|P^{-1}(p_{\text{up}}) - P^{-1}(p_{\text{down}})\|^2}. \quad (9)$$

Because the distance between neighboring pixels is fixed, $G(p)$ serves as an approximation of the gradients of P^{-1} at pixel p. This enables meaningful visualization and comparison of local gradients across the entire projection, despite potentially large variations in the corresponding distances in the high-dimensional space. Alongside, we report the projection loss $\mathcal{L}_{proj}$ and reconstruction loss $\mathcal{L}_{recon}$. The color scales are uniform to display the different properties, i.e., smoothness, high local gradients, and sharp and sudden changes in the gradient.

The model used is a P&R model with fully connected encoder–decoder networks operating on inputs of dimension 784. The encoder consists of a sequence of linear layers with decreasing widths ($784 \to 256 \to 128 \to 64 \to 32 \overset{\text{id}}{\to} 2$). All arrows indicate a linear transformation followed by the specified activation function, except where annotated with id, which denotes the identity mapping. Although consecutive linear layers are functionally equivalent to a single affine transformation, omitting the activation in the penultimate layer improves gradient flow near the low-dimensional bottleneck. The final layer outputs a 2-dimensional latent representation. The decoder mirrors the encoder with increasing widths ($2 \overset{\text{id}}{\to} 32 \to 64 \to 128 \to 256 \to 784$), also using the specified activation in all hidden layers and no activation in the output layer (as MSE is used for reconstruction), yielding the reconstructed input. The use of fewer neurons and the absence of regularization make these patterns more pronounced. In general, all activation functions were able to reconstruct items with reasonable accuracy, indicated by the overall low and comparable reconstruction loss. The *Sigmoid* activation function yields the lowest average and maximum gradient overall, with high values mainly appearing in gaps between clusters and in sparse regions of the projection (see Fig. 2(e)). In contrast, all other activation functions produce a more striking star pattern of high gradients originating from the projection center. They primarily differ, however, in how abruptly the gradient value increases. *HardTanh* has relatively low average gradients, but the highest maximum value among all activation functions (see Fig. 2(d)). These appear as very thin, sharp, and straight lines of high gradients. *Tanh* resembles the same pattern but with smoother transitions between the lines (see Fig. 2(c)). Regardless of the choice of activation function, certain areas consistently display high gradients, e.g., the upper-left and upper-right corners, as well as the center of the projection. Similarly, the cluster on the very left is characterized by low gradients for all activation functions.



(a) *ReLU* (L_{proj}: 107, L_{recon}: **0.03**) (b) *LeakyReLU* (L_{proj}: 104, L_{recon}: **0.03**)
(c) *Tanh* (L_{proj}: **74.2**, L_{recon}: **0.03**) (d) *HardTanh* (L_{proj}: 77.1, L_{recon}: 0.04)
(e) *Sigmoid* (L_{proj}: 87.1, L_{recon}: 0.04) (f) *SiLU* (L_{proj}: 91.1, L_{recon}: **0.03**)

Fig. 2. Impact of activation functions on inverse-projection gradients for MNIST with t-SNE. We also report projection loss L_{proj} and reconstruction loss L_{recon} (lower is better). Scales are uniform to highlight differences in smoothness and gradient behavior. Images use one-fourth of the P&R model neurons and no regularization to emphasize these effects.

As the *Sigmoid* activation function showed the lowest gradients with smooth transitions, with otherwise equal training and NN configurations, we use it in the following evaluations. Because the sigmoid activation requires evaluating exponentials and divisions at each neuron, its per-layer computational cost scales linearly with the number of activations, making it comparatively more expensive than piecewise-linear functions such as ReLU. However, ReLU was among the activation functions that created high average gradients (see Fig. 2(a)).

4.2. Trade-off regularization parameter analysis

The trade-off regularization parameters ω and α, for AE and VAE respectively, balance reconstruction and projection (or latent) conformity. A higher ω in the AE and a higher α in the VAE force the latent space to align more closely with the target projection. The effect of varying ω in the AE and α and β in the VAE on the projection is shown in Fig. 3 and Fig. 5. We performed a parameter scan and determined that for our experiments $\omega = 10$ achieves generally low reconstruction

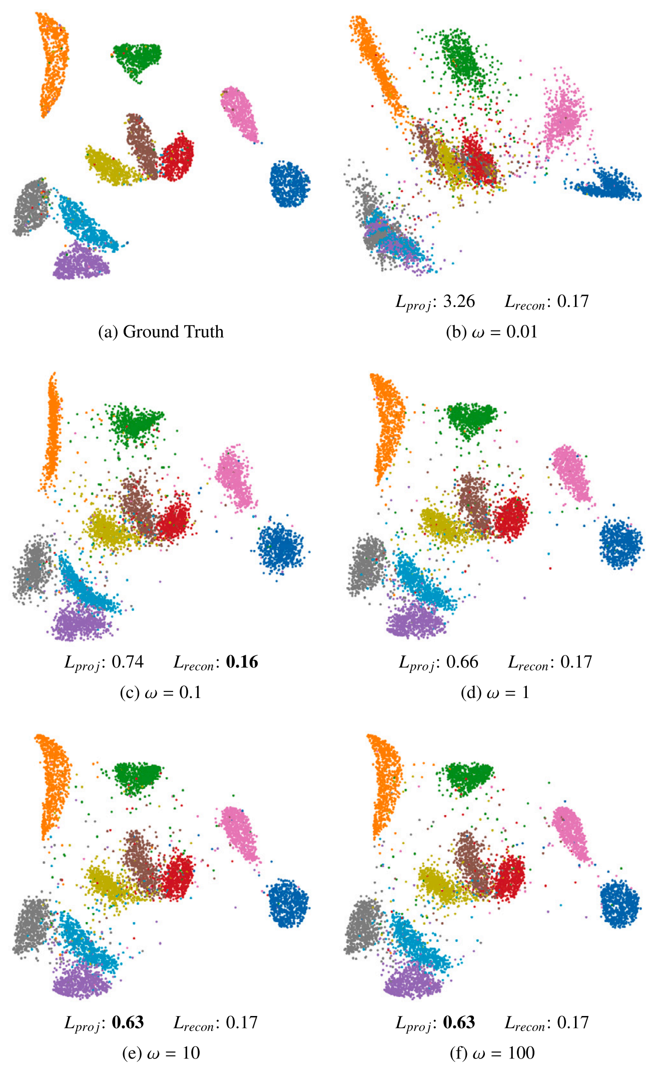


Fig. 3. The effect of ω on the parametric projection of the AE for the MNIST dataset projected with UMAP. We report the projection and reconstruction loss for each projection. The projection loss decreases for higher ω, while keeping the same reconstruction loss, yielding more faithful projections.

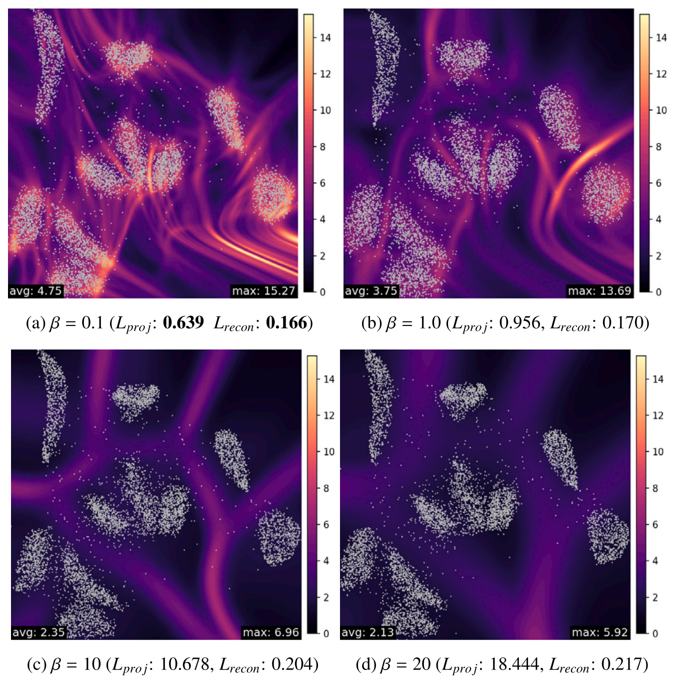


Fig. 4. The β parameter of VAE-$\hat{y}$ and VAE-μ controls the smoothness of the inverse projection. For MNIST projected with UMAP, we show that increasing β leads to progressively smoother gradients. This is evidenced by the attenuation of local patterns and by decreases in both the average and maximum gradient magnitudes. The α parameter is set to 10.0 in all cases.

and projection losses for test data. For lower values, the projection loses structure and deviates from the ground-truth projection. For $\omega = 0$ the model behaves like a standard autoencoder, meaning that there is no incentive for the latent representation to match the reference projection. Selecting a low ω can lead to a decrease in parametric projection quality (see Fig. 3(c)). Generally, choosing an excessively high ω can lead to a decrease in the inverse projection quality. Thus, we recommend starting the parameter scan with a low ω and increasing it stepwise until the inverse projection quality degrades.

The additional Kullback–Leibler divergence trade-off regularization parameter β in the VAE controls the regularization of the latent space (or projection), with a larger β encouraging each learned latent distribution to match the standard normal prior more closely. By penalizing a small standard deviation of the normal distribution centered at μ, β will force the decoder to reconstruct data points from coordinates further from the associated projection point. However, excessively increasing β collapses the latent means toward zero for all data points. Thus, a careful selection of β is necessary. We sampled various combinations (see Fig. 5) and found that $\alpha = 10.0$ and $\beta = 1.0$ achieve generally good quality in terms of reconstruction and projection error on test data. As β increases from 0.1 to 100, it encourages stronger regularization toward the prior distribution. For very large β, the projection scale is altered (see Fig. 5 with $\beta = 10$), increasing projection error while largely preserving the overall structure. For larger α, the clusters in the latent space become more compact and better separated.

Increasing α forces the projection to conform more strongly to the user-chosen one, yielding a clearer separation of structure and emergence of clusters independent of β. β is the more sensitive parameter directly influencing the smoothness of the inverse projection as measured by the gradient. We recommend evaluating the β parameter on a case-by-case basis. In general, we found that VAE-$\hat{y}$ and VAE-μ behave similarly for given α and β values (see supplementary material).

Finally, we directly analyze the effect of β on the smoothness of the inverse projections learned by VAE-$\hat{y}$ and VAE-μ. For this experiment, we tested four values $\beta \in \{0.1, 1, 10, 20\}$ with the VAE-μ model and a UMAP projection of the MNIST dataset. The results are shown in Fig. 4. Overall, we observe that the gradient maps exhibit decreasing gradients with increasing β. Local gradients become increasingly smoothed for larger β values, as evidenced by the disappearance of detailed gradient patterns in Fig. 4(a), which become smoother in Fig. 4(b) and largely vanish in Fig. 4(c) and Fig. 4(d). This behavior is confirmed by the maximum gradient measure. At a global level, gradients are also smoothed, as confirmed by the average gradient. However, this smoothing effect comes at a cost, namely reduced projection and reconstruction accuracy. Both measures show decreasing performance with increasing β values. For the subsequent experiments, we use the *Sigmoid* activation function, $\omega = 10$, $\alpha = 10.0$ and $\beta = 1.0$.

5. Evaluation

We evaluate our proposed approaches quantitatively using the projection loss and reconstruction loss, i.e., *mean-squared error (MSE)* (see Eq. (1)) or *binary cross-entropy (BCE)* (see Eq. (2)) for measuring the out-of-sample performance on test data. Alongside, we report

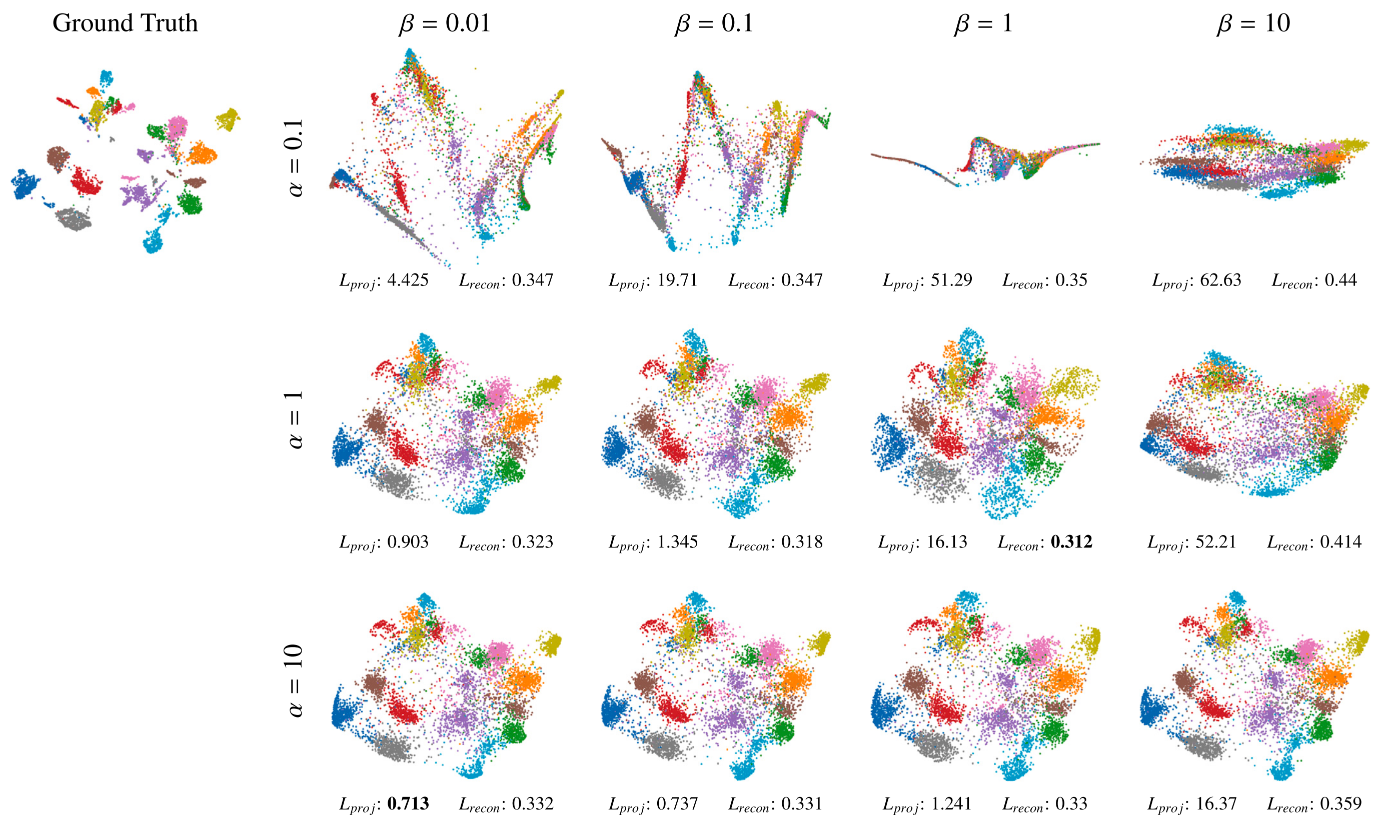


Fig. 5. The effect of α and β on the parametric projection of the VAE-μ model for a UMAP projection of *KMNIST* (see supplementary material for an example for VAE-$\hat{y}$). We provide the projection loss and reconstruction loss for each projection. Generally, higher α values produce projections closer to the ground-truth. For sufficiently high β, the projection loss increases, since the projection is scaled toward a normal distribution by the regularization effect.

Table 2
All datasets used during our evaluation. We report the ambient dimensionality $\mathbf{d}$, intrinsic dimensionality $\rho_{\mathbf{d}}$ (i.e., number of dimensions needed to capture 95% of the variance in the data), dataset size $\mathbf{n}$, and sparsity $\gamma_{\mathbf{n}}$ (i.e., ratio of data values that are zero or empty) [18].

Dataset	d	ρ_d	n	γ_n	Type
Blobs [33]	10	4	1000	0.0%	Synthetic
HAR [47]	561	120	10 299	0.0%	Sensor Data
MNIST [48]	784	330	70 000	82.9%	Images
Fashion-MNIST [49]	784	187	70 000	50.2%	Images
KMNIST [50]	784	238	70 000	66.7%	Images
COIL-20 [51]	4096	130	1440	35.6%	Images

runtime measurements. To evaluate the out-of-distribution properties and capabilities, we measure the *average* and *maximum gradient* of the inverse projection. Additionally, we qualitatively compare the results of the NN architectures by visually comparing parametric and inverse projections using projection plots, gradient maps, decision maps and reconstructions of image datasets [3].

5.1. Data and training

We compare the results across six datasets (see Table 2) using t-SNE [8] and UMAP [9] with standard parameters (t-SNE: `perplexity=30.0`, `early_exaggeration=12.0`; UMAP: `n_neighbors=15`, `min_dist=0.1`) as the ground-truth projections to be learned by the models. We did not standardize the input data per dimension, neither the dataset nor the projection, since in previous experiments, it did not improve the results [52].

We derive the topologies of our AEs from the configuration of the NNs proposed by Espadoto et al. [3], and Appleby et al. [14]. For P&R, we use these topologies directly; for AE and VAE, we combine the NNs to create a bottleneck according to our descriptions (see Fig. 1). During training, we quantify the model quality after each epoch using a validation set, which contains data items distinct from the test and train sets. In our experiments, we used an 80/10/10 train-validation-test split. Using the *Adam optimizer* [26], we observed that the training time of our AEs can vary. To end the training, we stop after 20 epochs without improvement of the model on the validation set. We set the maximum number of training epochs to 5000 for *COIL-20* and 1000 for all other datasets; however, this number was never reached. The learning rate is set to 0.0001, and the batch size is set to 128, which are typical values. However, these values may need to be chosen differently for other datasets. For all datasets except the *COIL-20* dataset, we use fully connected layers for the encoder and decoder [52]. More concretely, a flat vector is encoded by a sequence of fully connected layers with decreasing dimensionality ($1024 \rightarrow 512 \rightarrow 256 \rightarrow 128 \rightarrow 64$), yielding a compact intermediate representation. This representation is mapped to a 2-dimensional latent code dependent on the model as described in Section 3. The latent representation is then decoded by a symmetric sequence of fully connected layers ($64 \rightarrow 128 \rightarrow 256 \rightarrow 512 \rightarrow 1024 \rightarrow 784$), reconstructing the original 28×28 image. Since the *COIL-20* dataset is a large image dataset, we use convolutional layers in the encoder and transposed convolutional layers (for deconvolution) in the decoder. For *COIL-20*, a $1 \times 64 \times 64$ grayscale input is encoded by four 3×3 convolutional layers with stride 2, increasing the number of feature maps from 8 to 64, yielding a $64 \times 4 \times 4$ representation. This representation is flattened and mapped by fully connected layers to a 2-dimensional latent code dependent on the respective model variant. The latent representation is subsequently decoded by symmetric fully connected expansion and transposed convolutional layers, reconstructing an output of size $1 \times 64 \times 64$.

Table 3
Average MSE for quantifying the quality of the parametric projection, test loss of the inverse projections, trustworthiness, and continuity measures as well as the number of training epochs and the training time for 10 runs of each model, along with the standard deviation of each measurement.

Projection	Dataset	P&R	AE	VAE-$\hat{y}$	VAE-μ
Average Accuracy (MSE) of the Parametric Projection (lower is better)					
UMAP	Blobs	**0.073** ± **0.014**	0.117 ± 0.027	0.866 ± 0.095	0.863 ± 0.061
UMAP	MNIST	0.662 ± 0.08	**0.646** ± **0.07**	0.966 ± 0.071	0.969 ± 0.076
UMAP	KMNIST	0.747 ± 0.044	**0.745** ± **0.049**	1.214 ± 0.067	1.239 ± 0.068
t-SNE	HAR	**31.35** ± **3.894**	31.36 ± 4.228	44.43 ± 5.895	43.77 ± 3.83
t-SNE	Fashion-MNIST	30.65 ± 2.202	30.77 ± 2.128	**30.29** ± **1.398**	30.46 ± 1.534
t-SNE	COIL-20	**3.462** ± **3.475**	3.6 ± 3.651	9.642 ± 3.43	9.302 ± 3.302
Average Reconstruction Error of the Inverse Projection (lower is better)					
UMAP	Blobs	**1.005** ± **0.044**	1.942 ± 2.105	2.248 ± 2.582	1.647 ± 1.953
UMAP	MNIST	**0.163** ± **0.001**	0.17 ± 0.002	0.169 ± 0.002	0.171 ± 0.002
UMAP	KMNIST	**0.31** ± **0.005**	0.336 ± 0.003	0.325 ± 0.003	0.327 ± 0.003
t-SNE	HAR	**0.022** ± **0.001**	0.024 ± 0.001	0.024 ± 0.001	0.024 ± 0.001
t-SNE	Fashion-MNIST	**0.309** ± **0.001**	0.32 ± 0.002	0.32 ± 0.002	0.32 ± 0.002
t-SNE	COIL-20	**0.381** ± **0.017**	0.386 ± 0.018	0.391 ± 0.017	0.387 ± 0.017
Average Trustworthiness $T(k)$ with $k \in \{2,4,8,\dots,n/2\}$ (higher is better)					
UMAP	Blobs	**0.8770** ± **0.1508**	0.8757 ± 0.1498	0.8764 ± 0.1494	0.8769 ± 0.1498
UMAP	MNIST	0.8615 ± 0.1169	0.8606 ± 0.1153	0.8675 ± 0.1185	**0.8680** ± **0.1186**
UMAP	KMNIST	0.8171 ± 0.0828	0.8182 ± 0.0832	**0.8312** ± **0.0886**	0.8296 ± 0.0879
t-SNE	HAR	**0.8761** ± **0.1352**	0.8760 ± 0.1357	0.8759 ± 0.1351	**0.8761** ± **0.1350**
t-SNE	Fashion-MNIST	0.9261 ± 0.0624	0.9260 ± 0.0624	**0.9262** ± **0.0628**	0.9259 ± 0.0626
t-SNE	COIL-20	0.8478 ± 0.1586	**0.8479** ± **0.1589**	**0.8479** ± **0.1584**	0.8475 ± 0.1586
Average Continuity $C(k)$ with $k \in \{2,4,8,\dots,n/2\}$ (higher is better)					
UMAP	Blobs	**0.8994** ± **0.1159**	0.8972 ± 0.1148	0.8975 ± 0.1144	0.8975 ± 0.1154
UMAP	MNIST	0.8634 ± 0.1344	0.8614 ± 0.1325	0.8646 ± 0.1342	**0.8648** ± **0.1342**
UMAP	KMNIST	0.8686 ± 0.1101	0.8688 ± 0.1100	**0.8712** ± **0.1100**	0.8711 ± 0.1099
t-SNE	HAR	0.8813 ± 0.1446	0.8810 ± 0.1451	0.8813 ± 0.1444	**0.8815** ± **0.1442**
t-SNE	Fashion-MNIST	0.9435 ± 0.0611	**0.9436** ± **0.0611**	0.9433 ± 0.0614	0.9434 ± 0.0613
t-SNE	COIL-20	0.8533 ± 0.1527	0.8533 ± 0.1535	**0.8534** ± **0.1527**	**0.8534** ± **0.1527**
Average Number of Training Epochs (lower is better)					
UMAP	Blobs	708.3 ± 63.281	572.4 ± 128.0	**542.7** ± **134.08**	579.0 ± 87.514
UMAP	MNIST	**232.5** ± **36.482**	268.0 ± 41.721	280.3 ± 57.380	300.8 ± 51.155
UMAP	KMNIST	227.3 ± 124.67	**199.7** ± **80.793**	326.8 ± 107.03	307.2 ± 78.509
t-SNE	HAR	581.1 ± 35.228	599.3 ± 63.931	589.5 ± 95.061	**574.5** ± **108.96**
t-SNE	Fashion-MNIST	240.6 ± 41.796	**231.4** ± **37.851**	253.1 ± 29.915	240.6 ± 33.965
t-SNE	COIL-20	972.5 ± 271.12	**968.9** ± **287.31**	835.8 ± 233.35	1034.0 ± 382.27
Average Training Time in Seconds (lower is better)					
UMAP	Blobs	17.840 ± 1.712	**12.014** ± **2.506**	12.908 ± 2.799	13.684 ± 1.959
UMAP	MNIST	274.283 ± 42.991	**249.815** ± **40.137**	293.248 ± 58.932	314.746 ± 53.561
UMAP	KMNIST	270.868 ± 148.01	**186.208** ± **74.794**	341.128 ± 111.79	320.657 ± 81.749
t-SNE	HAR	112.59 ± 6.933	**91.943** ± **9.950**	101.571 ± 16.046	100.105 ± 18.454
t-SNE	Fashion-MNIST	286.086 ± 49.863	**216.647** ± **35.383**	264.735 ± 31.336	251.691 ± 35.121
t-SNE	COIL-20	66.988 ± 35.5	61.472 ± 35.05	**52.032** ± **24.28**	67.320 ± 45.51

Across all evaluated datasets and projection methods, we use a consistent set of trade-off parameters. For the AE variants, the reconstruction loss $\mathcal{L}_{recon}$ is chosen according to the data modality (MSE for *Blobs* and *HAR*, BCE for image datasets), with the weighting parameter fixed to $\omega = 10$. For both VAE-$\hat{y}$ and VAE-μ, the projection loss and KL regularization are weighted uniformly with $\alpha = 10$ and $\beta = 1$, respectively, across all experiments.

5.2. Quantitative comparison

We evaluate the quality of the parametric and inverse projections by measuring the average projection and reconstruction loss of test samples for each inverse method. To minimize the effect of outliers, we train 10 NNs for each method using different random assignments of items to the training and test sets. To evaluate neighborhood preservation, we use the standard Trustworthiness (T) and Continuity (C) measures [2,53]. For a neighborhood size k, T quantifies false neighbors as

$$T(k) = 1 - \frac{1}{T_{\max}(k)} \sum_i \sum_{j \in F_i(k)} (\rho_{ij} - k), \tag{10}$$

and C quantifies missed neighbors as

$$C(k) = 1 - \frac{1}{C_{\max}(k)} \sum_i \sum_{j \in M_i(k)} (r_{ij} - k). \tag{11}$$

ρ_{ij} and r_{ij} are the ranks of point j in the data and embedding spaces, and $F_i(k)$ and $M_i(k)$ denote the false and missed neighbors of i, respectively. The normalizations $T_{\max}(k) = C_{\max}(k)$ ensure values in $[0,1]$ and equal

$$T_{\max}(k) = C_{\max}(k) = \begin{cases} \dfrac{k\,n\,(2n - 3k - 1)}{2}, & k < \dfrac{n}{2}, \\ \dfrac{n\,(n-k)\,(n-k-1)}{2}, & k > \dfrac{n}{2}. \end{cases} \tag{12}$$

The aggregated results and their standard deviations are shown in Table 3.

The *average MSE on the parametric projection* (see first section of Table 3) shows how well the different methods project high-dimensional data points for a given projection method P, i.e., UMAP or t-SNE. Here, no method clearly outperforms the others with errors of similarly low ranges. Projection errors tend, however, to be lower for NNs without D_{KL} regularization (i.e., P&R and AE), demonstrating their ability to match a target projection more closely due to their simpler loss

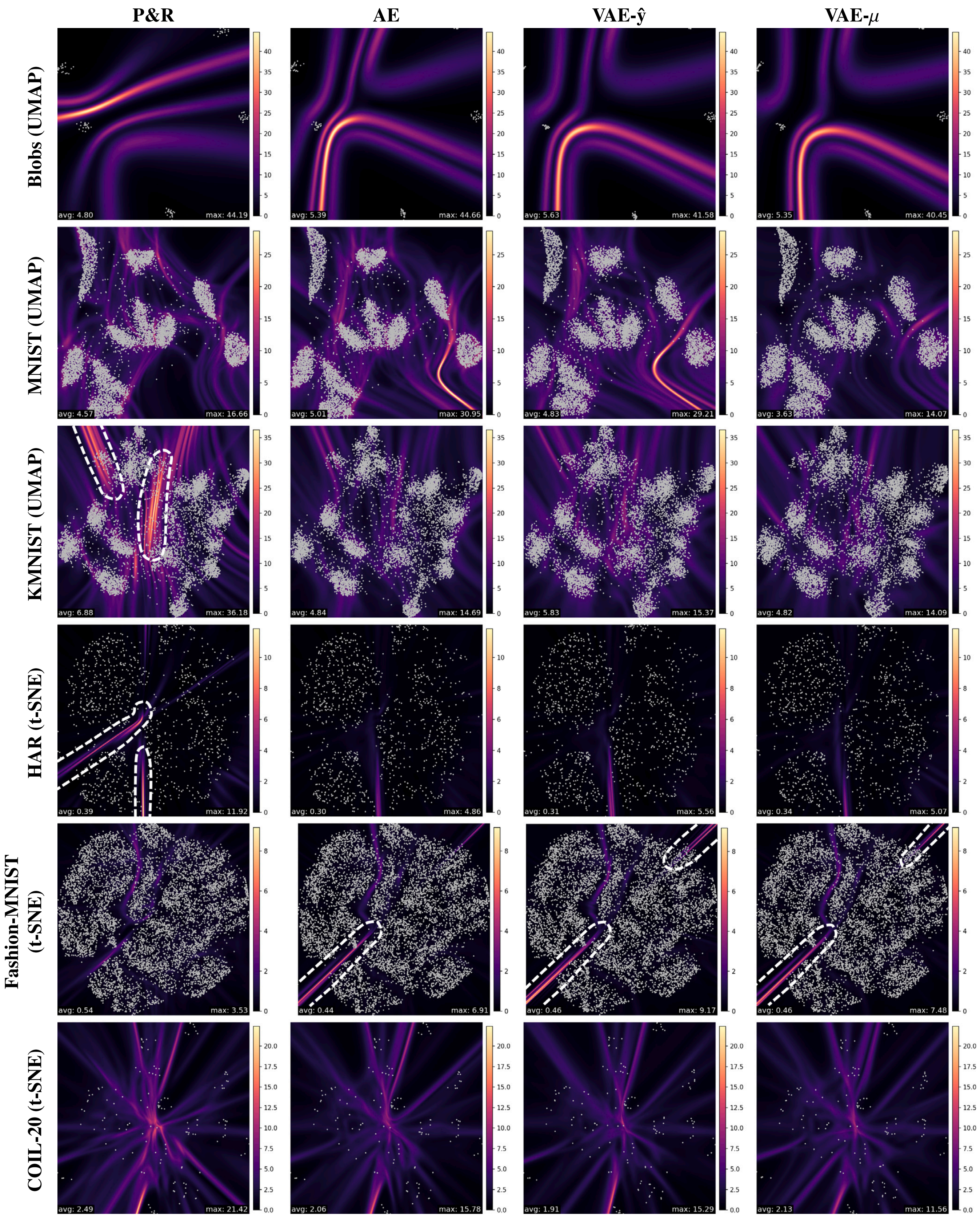


Fig. 6. Gradient maps of the four inverse projection methods for six datasets. The projected points correspond to the test data and approximate the reference projection. Darker colors indicate a low rate of change, and lighter areas indicate a high rate of change. The numbers show the average gradient (bottom left) and maximum gradient (bottom right).

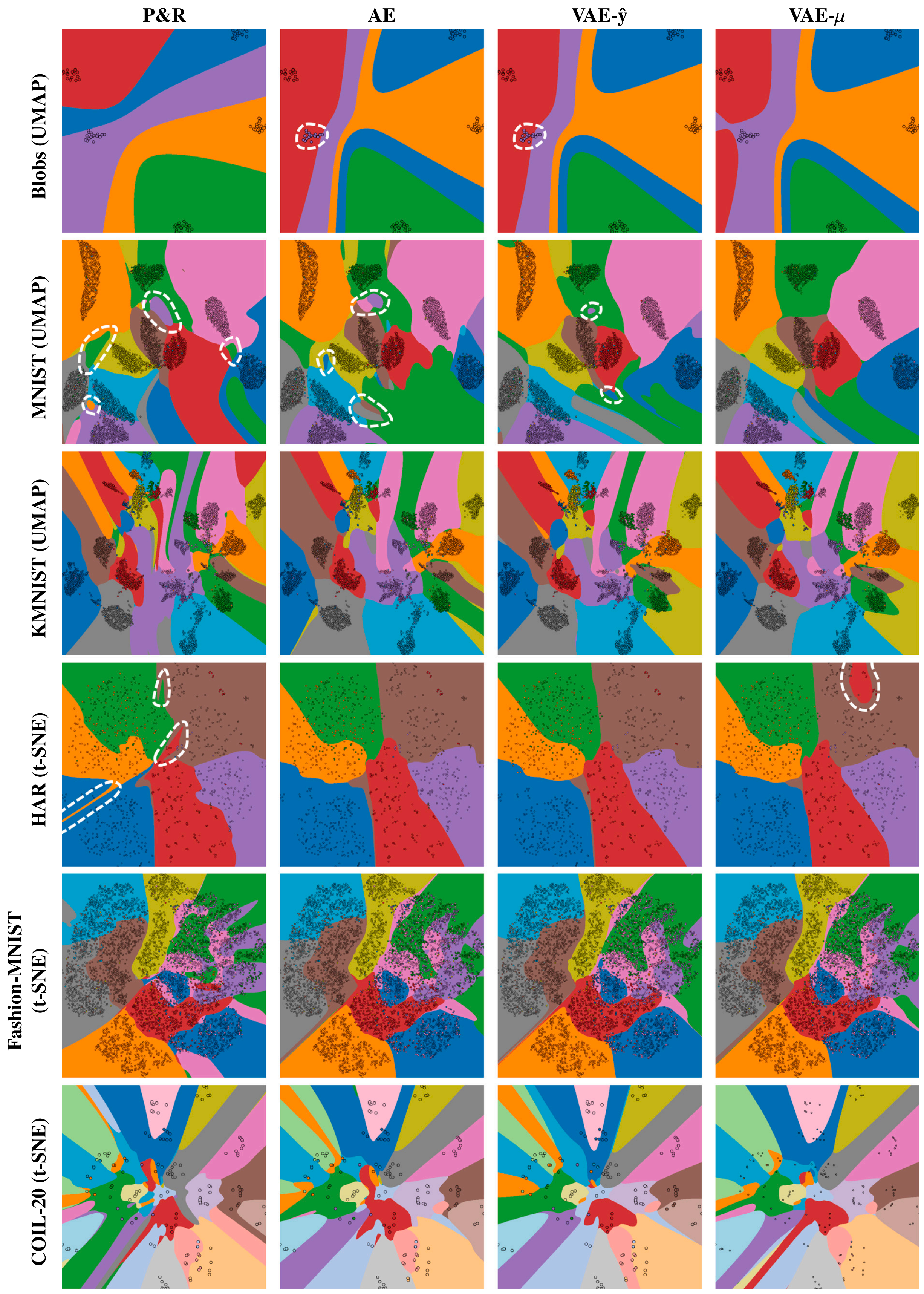


Fig. 7. Decision maps of the four inverse projection methods for six datasets. The images show the decision on the associated high-dimensional data item of a logistic regression classifier. The projected points correspond to the test data and approximate the reference projection.

formulation than the VAEs. The *average MSE of the inverse projection* (see second section of Table 3) shows how well the approaches can create a high-dimensional data point from a 2D point in the projection. P&R achieves the highest reconstruction quality overall, since it is explicitly trained for the inverse mapping only, without the added constraint of maintaining a consistent forward projection. Yet, the errors of the autoencoder-based architectures remain close to those of P&R, indicating that all inverse methods achieve comparably high reconstruction fidelity. During training, the standard AE is the most efficient in terms of training time across all datasets (last two sections of Table 3), though this may come at the cost of lower (inverse) projection quality. Training times generally increase with the intrinsic dimensionality and size of the dataset.

Smooth gradients are desirable as they promote local stability and ensure that nearby points in the projection space correspond to similar reconstructions. In contrast, extremely steep gradients indicate high sensitivity to small perturbations, which can reduce robustness and interpretability of the inverse projection. The *average gradient* (see Fig. 6 bottom left) measures the average rate of change of the high-dimensional inverse point when moving to a neighboring pixel in the 2D projection. The average gradient tends to increase with the intrinsic dimensionality of the dataset. Across methods, minor differences in average gradient magnitude are observed, with P&R showing slightly higher average gradients for the higher-dimensional datasets. The maximum gradient (see Fig. 6 bottom right) in relation to the mean gradient can be a first indicator of how suddenly high gradients appear, where a large gap between the maximum and average gradient suggests the presence of regions with steep transitions, i.e., high gradients. The difference is most pronounced for P&R in *HAR* (t-SNE), *KMNIST* (UMAP), and *COIL-20* (t-SNE). Among the AE-based methods, VAE-μ shows generally smaller differences between maximum and average gradients, indicating smoother transitions.

Trustworthiness and Continuity: We calculate the aggregated Trustworthiness (see Eq. (10)) and Continuity (see Eq. (11)) by averaging over powers of two, which provides balanced, logarithmic coverage of neighborhood scales while avoiding redundancy and bias toward densely sampled small values.

Across all datasets and projection methods, the observed *Trustworthiness* values are consistently high and exhibit only minor differences between P&R, AE, and VAE-based models. This indicates that all architectures are similarly capable of preserving local neighborhood relationships and none of the models degrades projection quality in a systematic manner. P&R and AE architectures perform on par with their VAE counterparts in most settings, suggesting that explicit probabilistic regularization is not strictly necessary to maintain trustworthy projections. Nevertheless, VAE-based models, particularly VAE-$\hat{y}$ and VAE-μ, achieve slightly higher average trustworthiness on several datasets (e.g., *MNIST* and *KMNIST*), which can be attributed to the regularizing effect of the latent prior that encourages regularized and more structured embeddings. However, these gains fall within the reported standard deviations. Overall, the results show that the choice between P&R, AE, and VAE architectures has only a limited impact on trustworthiness, and that differences in model design primarily influence other aspects such as stability and smoothness of the inverse mapping, rather than projection accuracy as measured by $T(k)$.

Table 3 also reports the aggregated *Continuity* $C(k)$ across datasets, projections, and model variants. Overall, the differences between P&R, AE, and VAE-based approaches are small, indicating that all models are capable of preserving neighborhood continuity. From a model perspective, P&R and AE variants exhibit similar behavior across all datasets, suggesting that explicitly coupling projection and reconstruction does not negatively impact continuity. The VAE-based models achieve continuity values that are on par with, and in some cases marginally higher than, their deterministic counterparts. In particular, VAE-$\hat{y}$ and VAE-μ tend to slightly outperform P&R and AE on datasets with more complex

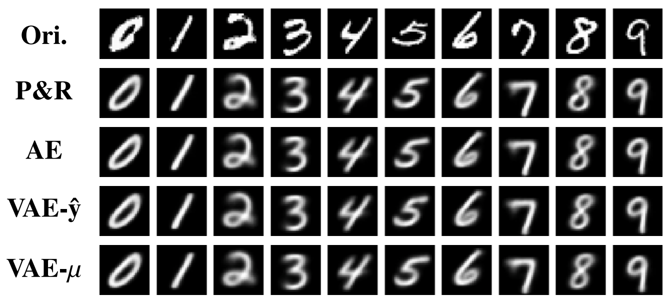


Fig. 8. Reconstructions on MNIST. All models produce visually similar reconstructions, indicating comparable reconstruction quality across methods.

structure (e.g., KMNIST and COIL-20), which may be attributed to the regularizing effect of the latent distribution. The differences between VAE-$\hat{y}$ and VAE-μ are negligible, indicating that sampling from the latent space does not substantially affect neighborhood preservation. Overall, these results suggest that incorporating probabilistic latent representations does not degrade continuity.

In general, the small magnitude of the observed differences across most measurements is insufficient to distinguish between inverse projection models, motivating the complementary qualitative analyses presented in the following section.

5.3. Qualitative comparison

In the following, we qualitatively assess and compare the parametric projection quality and the smoothness of the inverse projection through gradient and decision maps. Additionally, we compare their image reconstruction and generation capabilities visually.

Projection Quality: We projected all samples in the test set using the learned parametric projections of the proposed models. For example, the resulting projection for *MNIST* with UMAP of the standard AE is shown in Fig. 3 ($\omega = 10$) and for *KMNIST* with UMAP of the VAE in Fig. 5 ($\alpha = 10$ and $\beta = 1.0$). All parametric projections of our final evaluation exhibit the general patterns of the underlying projection to be learned. These observations confirm the quantitative evaluation with respect to low parametric projection errors.

Gradient Maps: Since there is no ground truth for many points $\hat{y} \in \mathbb{R}^2 \setminus P(D)$, we use *gradient maps* [3] to evaluate the inverse projection for those points. This allows us to visualize the rate of change in the high-dimensional space and quantitatively assess local stability in the inverse projections. Fig. 6 shows the gradient maps for the different datasets and NN architectures. Overall, all architectures tend to exhibit recognizable, similar structures in the gradient maps. For example, *COIL-20* (t-SNE) shows star-like structures, i.e., lines of high gradients, radiating from the center, and *Fashion-MNIST* (t-SNE) has high gradients in empty regions separating projected clusters (see areas surrounded by a dashed line). Especially among the three AE architectures, patterns are largely consistent. All methods struggle most with the UMAP projection of the lower-dimensional *Blobs* dataset, where gradient magnitudes are highest. This reflects instability in reconstructing from simple but sparsely structured embeddings, leaving large empty spaces where the models need to reconstruct far from projected samples. This dataset is also the only one in which the P&R gradient patterns strongly deviate from those of the AE-based methods. P&R also produces the strongest and most abruptly appearing high gradients in *HAR* (t-SNE) and *KMNIST* (UMAP) (see areas surrounded by a dashed line), indicating less smooth inverse mappings. Additionally, P&R in *COIL-20* (t-SNE) shows relatively narrow stripes of high gradients. In contrast, VAE-μ generally yields smoother gradient distributions with lower average and maximum gradient magnitudes, suggesting more stable and continuous mappings in the high-dimensional space.

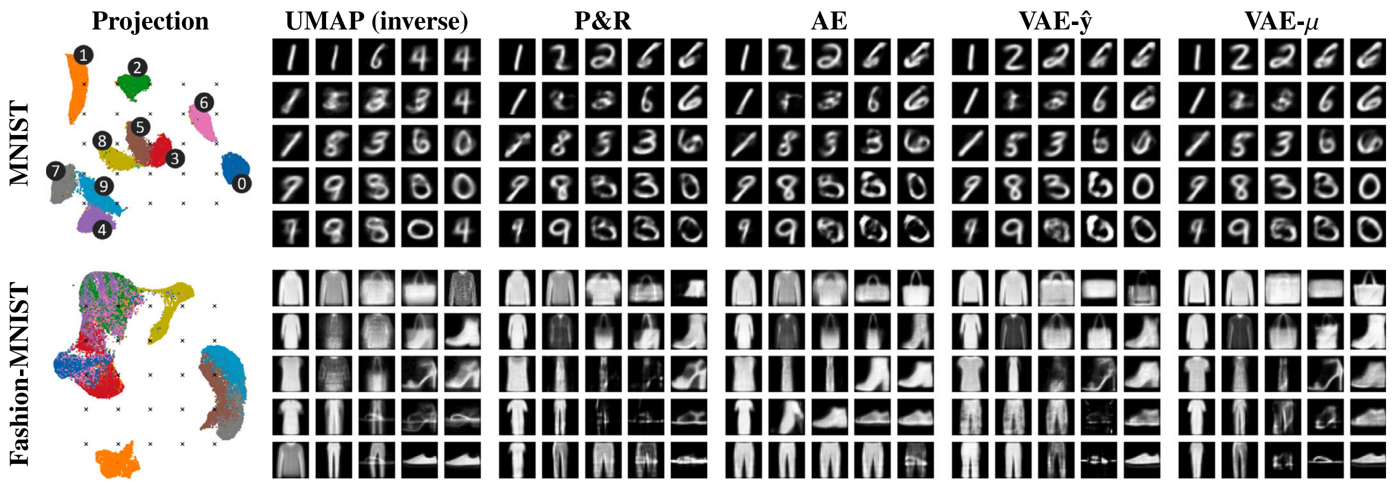


Fig. 9. We compare the reconstructed images using the different inverse projections of our models for the *MNIST* and *Fashion-MNIST* datasets with the UMAP built-in inverse function [29]. The locations of each inverse sample are shown in the projection on the left. VAE-μ generates the most coherent and diverse images, especially in between-class regions, outperforming other methods.

Decision Maps: We use *decision maps* [3] to partition the projected space into regions of one distinct classification. For visualization purposes only, we train a logistic regression classifier on the test samples and use it to classify the reconstructed samples of each pixel of the 2D map. The resulting classification is visualized as distinct zones, allowing us to visually assess the differences in the stability of inverse projections. The decision maps of the inverse projections shown in Fig. 7 match the patterns of the gradient maps. Generally, the strong correspondence between gradient and decision maps shows that smoother inverse projections yield more stable and coherent classification regions, with VAE-μ providing the least fragmented and most semantically faithful decision zones, especially in sparse regions of the projections. More specifically, regions of high gradient magnitude often correspond to misclassified areas or zones of classification uncertainty in the decision maps, e.g., *Blobs* (UMAP) and *HAR* (t-SNE) (see areas surrounded by a dashed line). This effect is visible specifically for P&R on *HAR* (t-SNE) showing spikes and thin lines reaching into areas associated with a different class. Also for MNIST (UMAP), disjoint patches are visible for all models except VAE-μ (see areas surrounded by a dashed line). Also, general patterns like the prominent star-pattern in the gradient maps of *COIL-20* (t-SNE), resemble the pattern of the decision zones. Except for *Blobs* (UMAP), all architectures produce broadly similar decision maps, consistent with their analogous gradients. However, VAE-μ consistently yields the most accurate and least fragmented decision zones, closely matching the true data classes of the projected samples. Notably, for *HAR* (t-SNE), VAE-μ is the only method where the small red cluster of data points within the brown region in the upper right corner (see areas surrounded by a dashed line) is embedded within a small decision zone of its own class. Furthermore, in sparse areas far from projected samples, VAE-μ produces less fragmented transitions between data classes, indicating better extrapolation behavior, e.g., *MNIST* (UMAP).

Image Reconstruction and Generation: For the image datasets, we can also directly inspect generated data samples of the models to qualitatively assess their inverse projection performance. Specifically, we compare the inverse projection results for data records in the test set to directly compare the reconstructions with ground truth records. The results for *MNIST* can be observed in Fig. 8 and for all other datasets in the supplementary material. In line with the low average reconstruction error in Table 3, all methods produce visually similar and accurate reconstructions, closely matching the ground truth images. Reconstructions in general show a high fidelity with generated images that mostly correspond to the same class as the target sample. Yet, we observe a loss of detail and increased fuzziness compared to the original images.

To qualitatively examine image generation capabilities for all regions of the projection, including *out-of-distribution* points, we reconstruct 25 points sampled from an evenly spaced 5 × 5 grid on the projection space. This allows us to visually examine how each method interpolates between known data regions and how well it handles sparse or unrepresented areas of the projected space. Fig. 9 shows the respective results for the UMAP projection of the *MNIST* and *Fashion-MNIST* datasets.

For *MNIST*, the proposed architectures generate images of comparable quality, with mostly plausible reconstructed digits. They all outperform the direct UMAP inverse, which produces blurrier numbers, particularly in regions between clusters. In contrast to the UMAP inverse, our architectures generate numbers matching the nearest cluster in the projection.

The *Fashion-MNIST* results reveal clearer distinctions between the proposed approaches. Overall, the VAE-μ produces the most coherent and interpretable reconstructions. This becomes particularly evident in regions between classes, where data generation is most challenging. Here, VAE-μ generates plausible intermediate samples (e.g., flip-flop and high-heel). In contrast, the P&R baseline and the direct inverse of UMAP often yield mixed-class images, i.e., overlaying features of multiple classes (e.g., trousers and shoes). This indicates instability in areas between classes. Among the AE-based approaches, the standard AE performs worst with respect to exhibiting low variability in the generated samples. For instance, many of the generated samples in the last row of *Fashion-MNIST* look similar and the last sample contains blended features from multiple classes, similar to P&R and the direct inverse of UMAP. VAE-$\hat{y}$ and VAE-μ produce fewer mixed-class images with generally sharper appearance and exhibit stronger out-of-distribution reconstruction capabilities, generating meaningful images even in sparsely populated regions of the embedding. However, the differences between the two models are negligible. Overall, VAE-μ consistently outperforms both its AE counterparts and the baseline P&R method.

6. Discussion and future work

The results for P&R indicate that a feed-forward neural network generally provides the highest accuracy for learning a parametric projection when accuracy is the primary objective. In this setting, P&R also slightly outperformed both the AE and VAE in generating inverse projections (see Section 5.2). To ensure a fair comparison, the architectures of the AE and VAE were kept identical to P&R, with the exception of their bottleneck structures.

Overall, the findings suggest that *joint* training of the projection P and its inverse P^{-1} using AEs tends to produce *smoother* inverse projections (see Section 5.3) reducing gradients and thus mitigating abrupt changes when generating data items. Among the methods tested, the VAE yielded the smoothest results, characterized by low average gradients and few regions of high gradient magnitude (see Fig. 6). These properties are also reflected in the most coherent decision zones, which closely align with the underlying labels (see Fig. 7). This suggests that the inclusion of the D_{KL} regularization term contributes to smoothing, enabling out-of-distribution reconstructions of high quality, especially in areas between multiple classes. Nevertheless, this smoothing effect is accompanied by a slight reduction in projection and inverse projection accuracy (see loss values reported in the subcaptions of Fig. 4). Importantly, the extent of this trade-off can be controlled via the VAE's hyperparameters α and β.

We propose the following *recommendations*. In applications where projection accuracy is critical, a standard feed-forward neural network, like P&R, is typically sufficient and yields the most accurate results. However, for tasks involving inverse projection, we recommend training VAEs. They improve the structure of the latent space through conditioning, enabled by the Kullback–Leibler divergence regularization term. α controls the projection supervision strength, while β controls the trade-off between latent space regularization and reconstruction accuracy. The strength of this regularization can be effectively tuned by performing a parameter sweep over multiple β values. We do not recommend the use of standard AEs, as they do not provide significant benefits over feed-forward neural networks in this context.

Furthermore, for VAEs, we suggest training the projection on the mean vector μ rather than the sampled output $\hat{y}$. While quantitative differences between the two variants are negligible (see Table 3), training on μ avoids injecting sampling noise into the projection loss, providing a more stable training signal. In some cases, projection rescaling (see Fig. 5 with $\beta = 100$) due to regularization may be acceptable. However, in such cases, similarity to the ground-truth projection should be evaluated using Procrustes analysis [54].

Future Work: So far, we have only compared Autoencoders (AE) and Variational Autoencoders (VAE) to disjoint networks (P&R). However, we did not compare them to non-neural-network-based inverse projection methods (see Section 2). Additionally, instead of relying on pre-calculated projections, we plan to directly optimize for a projection loss function, such as the t-SNE loss function, in the training process. Given that we employed common benchmark datasets and an 80/10/10 train–validation–test split, we aim to further evaluate each approach regarding its stability. In our method, the D_{KL} term in the VAE acts as an implicit regularization of the latent space; however, a comparable effect could be obtained in a standard AE through explicit regularization, such as applying L2-regularization to the Jacobian of the latent space. While our network topologies were adapted from prior studies, we intend to conduct ablation experiments to quantify performance degradation in the proposed architectures. Additionally, alternative NN architectures, such as generative adversarial networks (GANs) [55,56] or invertible neural networks [57], could be explored in future work. Finally, we plan to extend our method to support supervised dimensionality reduction by taking class annotations into account [58].

7. Conclusion

We evaluated three different AE-based regularization methods for generating *parametric* and *invertible* multidimensional data projections. Our qualitative and quantitative results for t-SNE and UMAP showed the differences in projection and reconstruction capabilities between the tested models. Overall, we observed that feed-forward NNs used for data projection and reconstruction generally outperform autoencoder-based methods in terms of accuracy. Nevertheless, AEs can achieve comparable performance while typically yielding smoother parametric and inverse projections. In particular, VAEs equipped with joint loss functions and Kullback–Leibler regularization show the highest potential for generating smooth inverse projections with superior out-of-distribution performance. Their flexible parameterization enables fine-grained, case-specific trade-offs between reconstruction accuracy and projection smoothness.

CRediT authorship contribution statement

Frederik L. Dennig: Writing – original draft, Visualization, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Daniela Blumberg:** Writing – original draft, Methodology, Conceptualization. **Nina Geyer:** Writing – review & editing, Visualization, Software. **Yannick Metz:** Writing – review & editing, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

During the preparation of this work, the author(s) used ChatGPT (OpenAI) for language refinement, i.e., grammatical and spelling checks. After using this tool, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the published article.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cag.2026.104552.

Data availability

A link to the accompanying data and code is included in the paper.

References

[1] Jolliffe IT. Principal component analysis. Springer; 1986, http://dx.doi.org/10.1007/978-1-4757-1904-8.

[2] Nonato LG, Aupetit M. Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. IEEE Trans Vis Comput Graph 2019;25(8):2650–73. http://dx.doi.org/10.1109/TVCG.2018.2846735.

[3] Espadoto M, Appleby G, Suh A, Cashman D, Li M, Scheidegger C, Anderson EW, Chang R, Telea AC. UnProjection: Leveraging inverse-projections for visual analytics of high-dimensional data. IEEE Trans Vis Comput Graph 2021;29(2):1559–72. http://dx.doi.org/10.1109/TVCG.2021.3125576.

[4] Dennig FL, Miller M, Keim DA, El-Assady M. FS/DS: a theoretical framework for the dual analysis of feature space and data space. IEEE Trans Vis Comput Graph 2024;30(8):5165–82. http://dx.doi.org/10.1109/TVCG.2023.3288356.

[5] Sainburg T, McInnes L, Gentner TQ. Parametric UMAP embeddings for representation and semisupervised learning. Neural Comput 2021;33(11):2881–907. http://dx.doi.org/10.1162/neco_a_01434.

[6] van Wijk JJ, van Overveld CWAM. Preset based interaction with high dimensional parameter spaces. In: Data visualization: the state of the art. Kluwer; 2003, p. 391–406.

[7] Schlegel U, Rauscher J, Keim DA. Interactive counterfactual generation for univariate time series. In: 6th ECML pKDD international workshop on eXplainable knowledge discovery in data mining. 2024.

[8] van der Maaten L, Hinton G. Visualizing data using t-SNE. J Mach Learn Res 2008;9(86):2579–605.

[9] McInnes L, Healy J, Saul N, Grossberger L. UMAP: Uniform Manifold Approximation and Projection. J Open Source Softw 2018;3(29):861. http://dx.doi.org/10.21105/joss.00861.

[10] Espadoto M, Hirata NST, Telea AC. Deep learning multidimensional projections. Inf Vis 2020;19(3):247–69. http://dx.doi.org/10.1177/1473871620909485.

[11] Hinterreiter AP, Humer C, Kainz B, Streit M. ParaDime: A framework for parametric dimensionality reduction. Comput Graph Forum 2023;42(3):337–48. http://dx.doi.org/10.1111/cgf.14834.

[12] Bank D, Koenigstein N, Giryes R. Autoencoders. In: Machine learning for data science handbook: data mining and knowledge discovery handbook. Springer; 2023, p. 353–74. http://dx.doi.org/10.1007/978-3-031-24628-9_16.

[13] Dennig FL, Geyer N, Blumberg D, Metz Y, Keim DA. Evaluating autoencoders for parametric and invertible multidimensional projections. In: 16th international euroVis workshop on visual analytics. The Eurographics Association; 2025, http://dx.doi.org/10.2312/eurova.20251099.

[14] Appleby G, Espadoto M, Chen R, Goree S, Telea AC, Anderson EW, Chang R. HyperNP: Interactive visual exploration of multidimensional projection hyperparameters. Comput Graph Forum 2022;41(3):169–81. http://dx.doi.org/10.1111/cgf.14531.

[15] Ngo QQ, Dennig FL, Keim DA, Sedlmair M. Machine learning meets visualization – experiences and lessons learned. It - Inf Technol 2022;64(4–5):169–80. http://dx.doi.org/10.1515/itit-2022-0034.

[16] Yin H. Nonlinear dimensionality reduction and data visualization: A review. Int J Autom Comput 2007;4(3):294–303. http://dx.doi.org/10.1007/s11633-007-0294-y.

[17] Cunningham JP, Ghahramani Z. Linear dimensionality reduction: survey, insights, and generalizations. J Mach Learn Res 2015;16:2859–900, URL http://jmlr.org/papers/v16/cunningham15a.html.

[18] Espadoto M, Martins RM, Kerren A, Hirata NST, Telea AC. Toward a quantitative survey of dimension reduction techniques. IEEE Trans Vis Comput Graph 2019;27(3):2153–73. http://dx.doi.org/10.1109/TVCG.2019.2944182.

[19] Liu S, Maljovec D, Wang B, Bremer P, Pascucci V. Visualizing high-dimensional data: Advances in the past decade. IEEE Trans Vis Comput Graph 2017;23(3):1249–68. http://dx.doi.org/10.1109/TVCG.2016.2640960.

[20] Kruskal JB, Wish M. Multidimensional scaling. Quantitative applications in the social sciences, Sage Publications; 1978, http://dx.doi.org/10.4135/9781412985130.

[21] Wang Y, Yao H, Zhao S. Auto-encoder based dimensionality reduction. Neurocomputing 2016;184:232–42.

[22] Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science 2006;313(5786):504–7. http://dx.doi.org/10.1126/science.1127647.

[23] Bunte K, Biehl M, Hammer B. A general framework for dimensionality-reducing data visualization mapping. Neural Comput 2012;24(3):771–804. http://dx.doi.org/10.1162/neco_a_00250.

[24] van der Maaten L. Learning a parametric embedding by preserving local structure. In: 12th international conference on artificial intelligence and statistics. vol. 5, 2009, p. 384–91.

[25] Lai C, Kuo MF, Lien Y, Su K, Wang Y. Parametric dimension reduction by preserving local structure. In: 2022 IEEE visualization and visual analytics. IEEE; 2022, p. 75–9. http://dx.doi.org/10.1109/VIS54862.2022.00024.

[26] Kingma DP, Ba J. Adam: A method for stochastic optimization. In: 3rd international conference on learning representations. 2015.

[27] Maas AL, Hannun AY, Ng AY. Rectifier nonlinearities improve neural network acoustic models. In: ICML workshop on deep learning for audio, speech, and language processing. 2013, URL https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf.

[28] Reichmann L, Hägele D, Weiskopf D. Out-of-core dimensionality reduction for large data via out-of-sample extensions. In: 14th IEEE symposium on large data analysis and visualization. IEEE; 2024, p. 43–53. http://dx.doi.org/10.1109/LDAV64567.2024.00008.

[29] McInnes L. Inverse transforms – UMAP 0.5.8 documentation. 2018, Online, URL https://umap-learn.readthedocs.io/en/latest/inverse_transform.html, Last [Accessed 09 December 2025].

[30] dos Santos Amorim EP, Brazil EV, II JD, Joia P, Nonato LG, Sousa MC. iLAMP: Exploring high-dimensional spacing through backward multidimensional projection. In: 7th IEEE conference on visual analytics science and technology. 2012, p. 53–62. http://dx.doi.org/10.1109/VAST.2012.6400489.

[31] Joia P, Coimbra DB, Cuminato JA, Paulovich FV, Nonato LG. Local affine multidimensional projection. IEEE Trans Vis Comput Graph 2011;17(12):2563–71. http://dx.doi.org/10.1109/TVCG.2011.220.

[32] Amorim E, Brazil EV, Mena-Chalco J, Velho L, Nonato LG, Samavati F, Sousa MC. Facing the high-dimensions: Inverse projection with radial basis functions. Comput Graph 2015.

[33] Blumberg D, Wang Y, Telea A, Keim DA, Dennig FL. MultiInv: Inverting multidimensional scaling projections and computing decision maps by multilateration. Comput Graph 2025.

[34] Oliveira AAM, Espadoto M, Jr. RH, Telea AC. SDBM: Supervised Decision Boundary Maps for Machine Learning Classifiers. In: 17th international joint conference on computer vision, imaging and computer graphics theory and applications. SCITEPRESS; 2022, p. 77–87. http://dx.doi.org/10.5220/0010896200003124.

[35] Rodrigues FCM, Espadoto M, Hirata R, Telea AC. Constructing and visualizing high-quality classifier decision boundary maps. Information 2019;10(9):280. http://dx.doi.org/10.3390/info10090280.

[36] Espadoto M, Hirata NST, Telea AC. Self-supervised dimensionality reduction with neural networks and pseudo-labeling. In: 16th international joint conference on computer vision, imaging and computer graphics theory and applications. 2021, p. 27–37. http://dx.doi.org/10.5220/0010184800270037.

[37] Machado A, Telea AC, Behrisch M. Controlling the scatterplot shapes of 2D and 3D multidimensional projections. Comput Graph 2024.

[38] Kingma DP, Welling M. Auto-Encoding Variational Bayes. In: 2nd international conference on learning representations. 2014, arXiv:http://arxiv.org/abs/1312.6114v10.

[39] Higgins I, Matthey L, Pal A, Burgess CP, Glorot X, Botvinick MM, Mohamed S, Lerchner A. Beta-VAE: Learning basic visual concepts with a constrained variational framework. In: 5th international conference on learning representations. 2017.

[40] Chen F, Gärtner T. Scalable interactive data visualization. In: Mach. learn. knowl. discov. databases. vol. 14948, Springer; 2024, p. 429–33. http://dx.doi.org/10.1007/978-3-031-70371-3_34.

[41] Nair V, Hinton GE. Rectified linear units improve restricted Boltzmann machines. In: 27th international conference on machine learning. Omni Press; 2010, p. 807–14, URL https://icml.cc/Conferences/2010/papers/432.pdf.

[42] LeCun Y, Bottou L, Orr GB, Müller K. Efficient BackProp. In: Neural networks: tricks of the trade - second edition. Lecture notes in computer science, vol. 7700, Springer; 2012, p. 9–48. http://dx.doi.org/10.1007/978-3-642-35289-8_3.

[43] Collobert R, Weston J. A unified architecture for natural language processing: deep neural networks with multitask learning. In: 25th international conference on machine learning. ACM international conference proceeding series, vol. 307, ACM; 2008, p. 160–7. http://dx.doi.org/10.1145/1390156.1390177.

[44] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature 1986;323(6088):533–6. http://dx.doi.org/10.1038/323533a0.

[45] Elfwing S, Uchibe E, Doya K. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. Neural Netw 2018;107:3–11. http://dx.doi.org/10.1016/j.neunet.2017.12.012.

[46] Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press; 2016, http://www.deeplearningbook.org.

[47] Anguita D, Ghio A, Oneto L, Parra X, Reyes-Ortiz JL. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In: International workshop on ambient assisted living. Springer; 2012, p. 216–23.

[48] LeCun Y, Cortes C, Burges C. MNIST handwritten digit database. 1998.

[49] Xiao H, Rasul K, Vollgraf R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. 2017, http://dx.doi.org/10.48550/arXiv.1708.07747, CoRR (Preprint), arXiv:1708.07747.

[50] Clanuwat T, Bober-Irizar M, Kitamoto A, Lamb A, Yamamoto K, Ha D. Deep learning for classical Japanese literature. 2018, http://dx.doi.org/10.48550/arXiv.1812.01718, CoRR (Preprint), arXiv:1812.01718.

[51] Nene SA, Nayar SK, Murase H. Columbia object image library (COIL-20). Tech. Rep. CUCS-005-96, Department of Computer Science, Columbia University; 1996.

[52] Dennig FL, Keim DA. DE-VAE: Revealing Uncertainty in Parametric and Inverse Projections with Variational Autoencoders using Differential Entropy. In: IEEE workshop on uncertainty visualization: unraveling relationships of uncertainty, AI, and decision-making. IEEE; 2025, p. 33–7. http://dx.doi.org/10.1109/UncertaintyVisualization68947.2025.00009.

[53] Venna J, Kaski S. Neighborhood preservation in nonlinear projection methods: An experimental study. In: 30th international conference on artificial neural networks ICANN. vol. 2130, Springer; 2001, p. 485–91. http://dx.doi.org/10.1007/3-540-44668-0_68.

[54] Gower JC. Generalized Procrustes analysis. Psychometrika 1975;40(1):33–51. http://dx.doi.org/10.1007/BF02291478.

[55] Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville AC, Bengio Y. Generative adversarial nets. In: Advances in neural information processing systems. vol. 27, Curran Associates, Inc.; 2014, p. 2672–80, URL https://proceedings.neurips.cc/paper_files/paper/2014/file/f033ed80deb0234979a61f95710dbe25-Paper.pdf.

[56] Durall R, Ho K, Pfreundt F, Keuper J. Latent space conditioning on generative adversarial networks. In: 16th international joint conference on computer vision, imaging and computer graphics theory and applications. vol. 4, SCITEPRESS; 2021, p. 24–34. http://dx.doi.org/10.5220/0010178800240034.

[57] Ardizzone L, Kruse J, Rother C, Köthe U. Analyzing inverse problems with invertible neural networks. In: 7th international conference on learning representations. OpenReview.net; 2019, URL https://openreview.net/forum?id=rJed6j0cKX.

[58] Colange B, Peltonen J, Aupetit M, Dutykh D, Lespinats S. Steering distortions to preserve classes and neighbors in supervised dimensionality reduction. In: Advances in neural information processing systems. vol. 33, Curran Associates, Inc.; 2020, p. 13214–25, URL https://proceedings.neurips.cc/paper_files/paper/2020/file/99607461cdb9c26e2bd5f31b12dcf27a-Paper.pdf.