






Evaluating Autoencoders for Parametric and Invertible Multidimensional Projections

Frederik L. Dennig¹ , Nina Geyer¹ , Daniela Blumberg¹ , Yannick Metz¹ , and Daniel A. Keim¹ 

¹University of Konstanz, Germany

Abstract

Recently, neural networks have gained attention for creating parametric and invertible multidimensional data projections. Parametric projections allow for embedding previously unseen data without recomputing the projection as a whole, while invertible projections enable the generation of new data points. However, these properties have never been explored simultaneously for arbitrary projection methods. We evaluate three autoencoder (AE) architectures for creating parametric and invertible projections. Based on a given projection, we train AEs to learn a mapping into 2D space and an inverse mapping into the original space. We perform a quantitative and qualitative comparison on four datasets of varying dimensionality and pattern complexity using *t*-SNE. Our results indicate that AEs with a customized loss function can create smoother parametric and inverse projections than feed-forward neural networks while giving users control over the strength of the smoothing effect.

CCS Concepts

• **Human-centered computing** → Visualization; • **Computing methodologies** → Machine learning;

1. Introduction

Multidimensional projections, also known as dimensionality reduction (DR) methods, are well-established and frequently used to analyze high-dimensional data visually [Jol86]. 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 [NA19]. A shortcoming of these approaches is that the result can be hard to interpret since the relationships are projected non-linearly. To address this, researchers proposed to enhance the visualization through layout enrichments, showing the distortions and gradients between projected points [NA19; EAS*21; DMKE24]. However, we argue that these methods benefit from projections that are *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 [SMG21]. (2) Invertible methods have a smooth mapping from the projection space back to the original high-dimensional space, allowing us to generate new data from any arbitrary position of the projection [vWvO03], e.g., for interactive counterfactual generation [SRK24]. However, DR methods, such as *t*-SNE [vd-Maa09], are generally not parametric or invertible. In recent years, neural networks (NNs) have been used to learn mappings from high-dimensional space to the projection space [EHT20], and vice versa [EAS*21; HHKS23]. However, while these approaches can be applied to arbitrary projections, they only consider each mapping di-

rection individually. In this work, we explore and evaluate different autoencoders (AEs) [BKG23] to support parametric mapping and inversion jointly through their encoder-decoder architecture. More specifically, we compare the individual *feed-forward NNs* proposed by Espadoto et al. [EAS*21] and Hinterreiter et al. [AEC*22] 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 inverse project data points and compare them qualitatively by visually assessing their outputs. Additionally, we analyze the smoothness of the projection through gradient maps. Overall, we contribute the following:

- (1) We propose three *AE-based NN architectures* for creating parametric and invertible projections.
- (2) We perform an *evaluation* comparing the three architectures *quantitatively* and *qualitatively* on four datasets using *t*-SNE.
- (3) For *reproducibility*, we provide the analysis, results and source code at <https://osf.io/r2yqcd>.

With this work, we aim to enhance high-dimensional data analysis by improving the interpretation and explainability of DR methods.

2. Related Work on Multidimensional Projections

Let $D = \{x_i\}_{1 \leq i \leq n}$ be a high-dimensional dataset with d dimension and n samples $x_i \in \mathbb{R}^d$. A *projection* method P maps D to $P(D) = \{P(x_i) | x_i \in D\} = \{y_i\}_{1 \leq i \leq n}$, where $P(D) \subset \mathbb{R}^q$ with $q \ll d$. For our case, $q = 2$; thus, $P(D)$ can be visualized in a 2-dimensional scatterplot. Projection methods have been extensively studied and

assessed in several surveys [Yin07; CG15; EMK*19; NA19]. They can be classified into *linear* and *non-linear* methods [LMW*17; EMK*19]. Additionally, they either preserve *global* structure or *local* neighborhoods. Linear projection methods like PCA [Jol86] can be computed very efficiently and preserve the global structure of the data. MDS [Kru78] 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 [vdMH08] or UMAP [MHM18]. Autoencoder-based approaches generally also fall into this category [WYZ16].

Parametric Projections Methods: Standard non-linear DR methods [Kru78; vdMH08; MHM18] are non-parametric, needing complete recalculation when projecting new data points [HHKS23]. Parametric approaches [BBH12] address this limitation, e.g., by using NNs to learn mapping functions into lower-dimensional space [HS06; vdMaa09]. Van der Maaten [vdMaa09] used a feed-forward NN to build *parametric t-SNE*. Similarly, *parametric UMAP* [SMG21] uses NNs, including autoencoders, designed to replace the non-parametric embedding step. By training NNs to infer 2D coordinates for input domain data points, Espadoto et al. [EHT20] showed that NNs with sufficient size can effectively approximate many existing non-parametric methods. *HyperNP* [AEC*22] uses NNs to approximate projection techniques across hyperparameters (e.g., perplexity for t-SNE), allowing for their interactive exploration. *ParaDime* [HHKS23] streamlines the creation of NN-based DR approaches by proposing a grammar specific to parametric DR.

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 inversely defined, e.g., UMAP [MHM18]. An inverse mapping is not explicitly available for most other projection methods, requiring alternative methods, such as NNs [EAS*21]. One of the early methods introduced a global interpolation-based technique to generate new data points [vWvO03]. Later, *iLAMP* [dSBI*12], which builds on LAMP [JCC*11], addresses inverse projection by focusing on preserving local relationships. Amorim et al. [ABM*15] later refined this by using a radial basis function kernel for interpolation. Blumberg et al. [BWT*24] 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. [EAS*21] learn the mapping from the projection space back to the high-dimensional space.

3. Training Autoencoders for Multidimensional Projections

We train AEs to *jointly* create a parametric projection P and inverse projection P^{-1} . AEs are a class of NNs designed for unsupervised learning of efficient data representations [BKG23]. AEs were already well-established in the context of DR [HS06]. 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, e.g., SSNP [EHT21] and ShaRP [MTB24] integrate pseudo-labels from clustering into their loss functions. However, these approaches do not allow for inverting arbitrary user-defined projections.

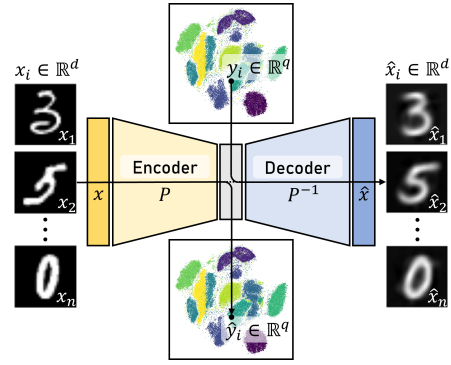


Figure 1: The encoder of the AE learns the parametric projection P mapping new data record x_i into 2D space (as \hat{y}_i). The decoder learns the inverse projection P^{-1} generating a high-dimensional sample \hat{x}_i from any 2D point y_i .

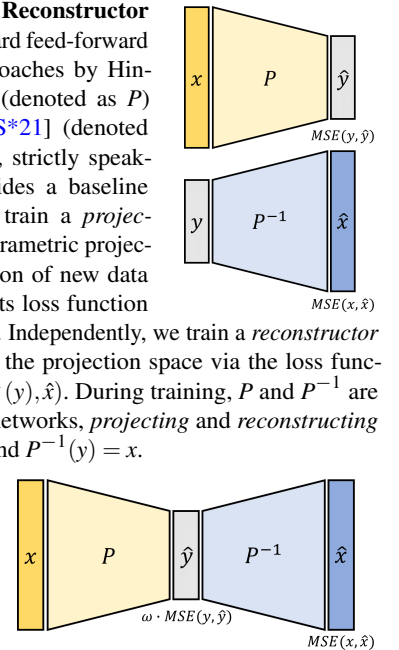
An AE consists of an *encoder* and a *decoder* (Fig. 1). The encoder $\text{Enc} : \mathbb{R}^d \rightarrow \mathbb{R}^q$ learns a mapping from the high-dimensional input space into a lower-dimensional latent space $\hat{Y} = \{\hat{y}_i\}_{i=1}^N$, with $\hat{y}_i = \text{Enc}(x_i) \in \mathbb{R}^q$ and N data points. In contrast, the decoder $\text{Dec} : \mathbb{R}^q \rightarrow \mathbb{R}^d$ aims to map these latent representations back into the original input space $\hat{x}_i = \text{Dec}(y_i) \in \mathbb{R}^d$. 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, i.e., *end-to-end*. A common choice for this loss is the mean squared error (MSE) $\mathcal{L}_{\text{rec}}(x_i, \hat{x}_i) = \text{MSE}(x_i, \hat{x}_i) := \|x_i - \hat{x}_i\|^2$. This error is *backpropagated* through the AE, and its parameters are updated accordingly. We use modern training methods for *deep* AEs, including batches of data and stochastic gradient descent [KB15]. We use standard loss functions, like the mean squared error (MSE) and the Kullback-Leibler divergence (D_{KL}), a measure of the difference of two probability distributions A and B . In the following paragraph, we describe our proposed NN architectures.

Individual Projector and Reconstructor (P&R):

We train two standard feed-forward NNs to combine the approaches by Hinterreiter et al. [AEC*22] (denoted as P) and Espadoto et al. [EAS*21] (denoted as P^{-1}). This structure is, strictly speaking, not an AE but provides a baseline for our experiments. We train a *projector* network learning the parametric projection P , enabling the addition of new data to the projection through its loss function $\mathcal{L}_{\text{pro}}(x, \hat{y}) = \text{MSE}(P(x), \hat{y})$. Independently, we train a *reconstructor* network learning to invert the projection space via the loss function $\mathcal{L}_{\text{rec}}(y, \hat{x}) = \text{MSE}(P^{-1}(y), \hat{x})$. During training, P and P^{-1} are learned by the individual networks, *projecting* and *reconstructing* the dataset, i.e., $P(x) = y$ and $P^{-1}(y) = x$.

Autoencoder with Latent Loss Term (AEL):

To avoid potential projection and reconstruction artifacts caused by training individual networks, we propose using AEs. In AEs, the encoder and decoder are trained jointly as one NN. The encoder of the AE learns to project the data



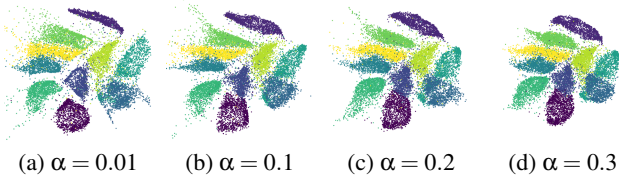


Figure 2: The effect of α on the parametric projection of VAEL using a t-SNE projection of MNIST. β is set to 1.0 in all examples.

points, while the decoder learns the inverse projection. To impose a structure on the latent space, we define the loss:

$$\mathcal{L}_{AEL}(x, \hat{x}, \hat{y}) = \text{MSE}(x, \hat{x}) + \omega \cdot \text{MSE}(y, \hat{y}) \quad (1)$$

Here, the $\text{MSE}(x, \hat{x})$ denotes the reconstruction loss. To enable the AE to learn a given projection, we modify its loss function by adding a component $\omega \cdot \text{MSE}(y, \hat{y})$ to force its latent space to conform. The weight $\omega \in \mathbb{R}^+$ determines the strength of the latent space to conform to the projection, i.e., between the projection and reconstruction quality. We performed a parameter scan (see supplementary material) and determined that $\omega = 0.5$ archives a generally low MSE for test data. We discuss the effect of ω in Sec. 5.

Variational Autoencoder with Latent Loss Term (VAEL):

Instead of directly predicting the latent variable \hat{y} , the encoder predicts the parameters of the normal distribution μ and σ^2 . In training, \hat{y} is sampled from a 2D normal distribution, i.e., $\hat{y} \sim \mathcal{N}(\mu, \sigma^2)$ with $\hat{y}, \mu, \sigma^2 \in \mathbb{R}^q$. The VAE is still trained end-to-end, using the *reparameterization trick*, to backpropagate through the sampling operation [KW14]. For our VAEL, we incorporate the *evidence lower bound loss (ELBO)* used for VAE training as:

$$\mathcal{L}_{VAEL}(x, \hat{x}, y, \mu, \sigma^2) = \text{MSE}(x, \hat{x}) + \alpha \cdot \text{MSE}(y, \hat{y} \sim \mathcal{N}(\mu, \sigma^2)) + \beta \cdot D_{\text{KL}}(\mathcal{N}(\mu, \sigma^2) \parallel \mathcal{N}(0, 1)) \quad (2)$$

We follow the framework of the β -VAE [HMP*17], which adds an additional β -parameter to balance conformity to the prior normal distribution and reconstruction quality. We sampled various combinations and found that $\alpha = 1.0$ and $\beta = 0.1$ achieve generally good quality in terms of MSE on test data. The effect of varying α is shown in Fig. 2. We discuss the selection of α and β in Sec. 5. Similar to Chen et al. [CG24], we use μ as a 2D coordinate to create parametric projections, i.e., for inference only, since it is the mean and mode of the normal distribution.

3.1. Data Preprocessing and Training

We derive the topologies of our AEs from the configuration of the NNs proposed by Espadoto et al., and Appleby et al. [EAS*21; AEC*22]. For P&R, we use these topologies directly; for AEL and VAEL, we combine the NNs to create a bottleneck according to our descriptions. Using the *Adam optimizer* [KB15], we observed that the training of our AEs converges similarly fast. Thus, we set the number of training epochs to 50, allowing all to converge. The learning rate is set to 0.001, and the batch size is set to 32, which are

Dataset	d	ρ_d	n	γ_n	Type
Rings [BWT*24]	3	3	180	0.0%	Synthetic
HAR [AGO*12]	561	120	735	0.0%	Sensor Data
MNIST [LCB98]	784	330	70000	82.9%	Images
Fashion MNIST [XRV17]	784	187	3000	50.2%	Images

Table 1: Evaluation datasets with dimensionality d , intrinsic dimensionality ρ_d , dataset size n , and sparsity γ_n [EMK*19].

	Rings	HAR	MNIST	Fashion MNIST
Average MSE of the Parametric Projection (lower is better)				
R&P	.119 (.021)	.037 (.003)	.039 (.001)	.027 (.001)
AEL	.013 (.005)	.041 (.002)	.074 (.004)	.046 (.003)
VAEL	.030 (.010)	.047 (.004)	.053 (.005)	.045 (.003)
Average MSE of the Inverse Projection (lower is better)				
R&P	.228 (.028)	.398 (.005)	.720 (.010)	.459 (.008)
AEL	.033 (.012)	.357 (.006)	.744 (.052)	.525 (.082)
VAEL	.118 (.037)	.420 (.005)	.847 (.048)	.556 (.019)

Table 2: Aggregated MSE and standard deviation (in braces) of the parametric and inverse projections on test data for 10 runs each.

the typical values. We use batch normalization after each layer and dropout regularization with a probability of 0.25. For all details, refer to the source code. In our experiments, we standardize the input data per dimension, i.e., the dataset and the projection. When projecting or reconstructing data using the encoder or decoder, we apply the inverse of the standardization to the output, enabling us to recover the representation in the original space and the projection space, respectively. This is possible since standardization is an invertible linear transformation due to $\sigma^2 \in \mathbb{R}^+$.

4. Evaluation

We evaluate the approaches quantitatively using the *mean squared error (MSE)*, the *average gradient* of the inverse projection, and *runtime measurements*. Additionally, we qualitatively compare the results of the NN architectures by visually comparing parametric and inverse projections.

Since there is no ground truth for many points $\hat{y} \in \mathbb{R}^2 \setminus P(D)$, we use *gradient maps* [EAS*21] to evaluate the inverse projection for those points. For each \hat{y} , we compute a pseudo-derivative of P^{-1} measuring the difference between the horizontal and vertical neighbors of a pixel defined as $G(q) = \sqrt{\|P^{-1}(\hat{y}_{left}) - P^{-1}(\hat{y}_{right})\|^2 + \|P^{-1}(\hat{y}_{up}) - P^{-1}(\hat{y}_{down})\|^2}$. This allows us to visualize the rate of change in the high-dimensional space and quantitatively assess the inverse projection's gradients. We compare the results across four datasets (Tab. 1) using t-SNE with standard parameters [vdMH08] as the ground-truth projection to be learned.

4.1. Quantitative Comparison

We evaluate the quality of the parametric and inverse projections by measuring the average MSE of test samples for each inverse method using an 80/20 train-test split. 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. The aggregated MSEs and their standard deviations are shown in Tab. 2.

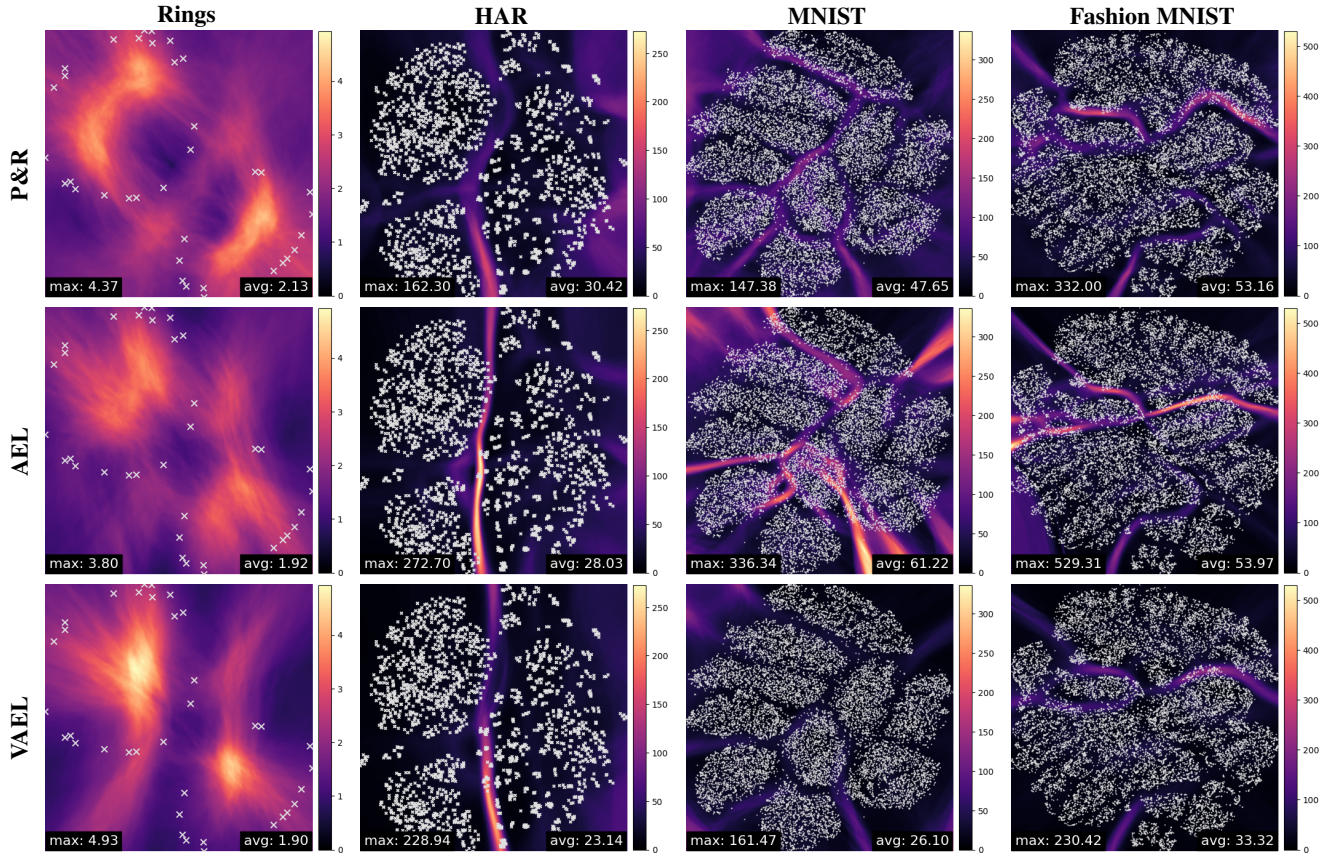


Figure 3: Gradient maps of the three inverse projection methods for four datasets. Darker colors indicate a low rate of change, and lighter areas indicate a high rate of change. The numbers show the maximum gradient (bottom left) and average gradient (bottom right).

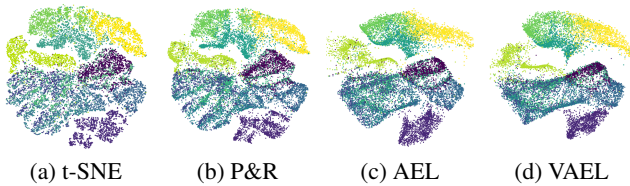


Figure 4: Ground truth t-SNE projection of Fashion MNIST (a) and the parametric projections of test data for the three methods (b–d).

The average MSE of the parametric projection (Tab. 2 top) shows how well the different methods project high-dimensional data points for a given projection method P , i.e., t-SNE. The P&R applied to the *Rings* dataset results in a high average MSE, suggesting that the projection performs the worst comparatively to the others, while AEL outperformed the other two. For high-dimensional datasets (i.e., *HAR*, *MNIST*, and *Fashion MNIST*), AEL and VAEL cannot outperform P&R. VAEL outperformed AEL in all benchmarks except *HAR*. The average MSE of the inverse projection (Tab. 2 bottom) shows how well the approaches can create a high-dimensional data point from a 2D point in the projection. AEL outperformed the other approaches for the *Rings* and *HAR* datasets. For *Fashion MNIST*, it performed the worst. Comparing P&R with VAEL shows that R&P generally outperformed VAEL except for the low-dimensional *Rings* dataset. The average gradient (Fig. 3 bottom right) 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 increases with the intrinsic dimensionality of the dataset and generally decreases from R&P to VAEL, suggesting that the VAEL generally produces smoother gradient maps. Training times generally increase with the intrinsic dimensionality and size of the dataset (see supplementary material). R&P requires the longest time to train, mainly because it involves training two separate networks. Additionally, the VAEL takes longer to train than the AEL. Overall, inference times are similar and generally low for all the datasets, staying between 0.01s and 0.04s for all architectures.

4.2. Qualitative Comparison

We projected all samples in the test set using the learned parametric projections for the *Fashion MNIST* dataset (Fig. 4). All parametric projections exhibit the general patterns of the t-SNE projection. P&R (b) looks similar to the ground truth. The AEL projection (c) looks more clumpy and stringy than the P&R projection. VAEL (d) results in slightly fuzzier classes but misses the stringy artifacts in (c). The observations confirm the quantitative evaluation. We compared the inverse projection result for data records in the test set of the image datasets, i.e., *MNIST* and *Fashion MNIST*. The results can be observed in the supplementary material. P&R and AEL produce output that can be assigned to one of the classes in the *MNIST* and *Fashion MNIST* datasets. P&R is more accurate than AEL since it produces

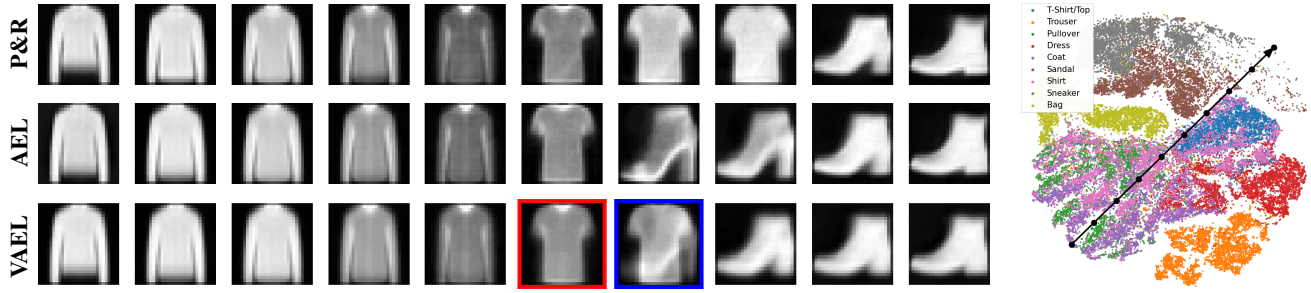


Figure 5: For each of the three NN architectures, we inverse project 10 samples from an interpolation line of the projection space. Samples are drawn at equal intervals from an interpolation line in the t-SNE projection of the Fashion MNIST dataset (right).

more images matching the ground truth. VAE will produce fuzzier-looking images compared to P&R and AEL. To show how the NNs handle transitions between multiple classes, we also visualized the inverse projections of samples of low-dimensional points along an interpolation line (Fig. 5). The samples generated by P&R and AEL exhibit a generally high fidelity and can be assigned to a class of the *Fashion MNIST* dataset. The sample from VAE marked red is less sharp, and the sample marked blue shows a shirt and a shadow of a boot. Thus, VAE produces smoother gradients while generating less accurate inverse projections. Finally, we analyze the *gradient maps* for the different datasets and NN architectures (Fig. 3). All approaches create differing gradient maps for the *Rings* dataset, with AEL creating the smoothest map. For the other datasets, the gradient maps show similar patterns, with higher gradients between clusters and generally lower gradients within clusters. *HAR* shows higher gradients on the right edge, while *MNIST* exhibits a similar pattern on the left. For the *HAR*, a high gradient separates the two left clusters from the right in all maps. For *MNIST* and *Fashion MNIST*, the maps show different gradients separating clusters. P&R creates smoother gradient maps for the higher-dimensional datasets than AEL with lower maximum gradients. When comparing the gradient maps of VAE for *HAR*, *MNIST*, and *Fashion MNIST* with the others, we can observe that the approach creates gradients in similar locations. However, their average gradients are the lowest. Thus, the inverse projections of VAE are generally *smoother*.

5. Discussion and Future Work

The results for P&R show that a feed-forward NN generally works best for creating a parametric projection. Similarly, P&R outperformed AEL and VAE for generating inverse projections of *MNIST* and *Fashion MNIST*. The architectures of AEL and VAE are directly derived from P&R. However, AEL and VAE may benefit from additional and larger layers, requiring a validation set to determine their hyperparameters. Our results suggest that the *joint* training of P and P^{-1} using AEs generally yields *smoother* inverse projections. In terms of smoothness, VAE gives the best results with low average gradients and smaller areas of high gradients (Fig. 3), showing that incorporating the D_{KL} loss term has a smoothing effect. Our evaluation suggests that this effect comes at the cost of the parametric and inverse projection accuracy. However, the smoothing strength is controllable through the parameters of VAE.

Loss Weights ω , α , and β : We determined the values for ω , α , and β experimentally by calculating the MSE on reconstruction and

latent space for different levels of ω , α , and β (see supplementary material). For AEL, choosing a lower ω leads to a lower MSE on inverse projection and a higher MSE on the parametric projection. We recommend selecting an ω of 0.5 as the values between 0.4 and 5.0 yield similarly low MSEs. For the VAE, the smaller the β , the better the weighted sum of MSE on parametric and inverse projection. In contrast, the weighted sum decreases with larger α . We chose a β of 0.1 and an α of 1.0 since these weights yield generally low MSE losses. α can be increased to 5.0 (with $\beta = 0.1$), without negatively affecting the MSE. β is the more sensitive parameter directly influencing the smoothness of the inverse projection. We recommend evaluating the β parameter on a case-by-case basis.

Future Work: Our work focused on t-SNE; however, we plan to extend the evaluation to other projection methods, like UMAP. We also plan to compare our AE-based architectures to existing parametric and inverse projection methods. Since we used common datasets and an 80/20 train-test split, we would like to further test the stability of each approach w.r.t. amount of training data and intrinsic dimensionality. In our approach, D_{KL} of the VAE serves as an implicit regularization of the latent space. However, a similar effect could be achieved with a standard AE using explicit regularization, such as an L2-regularization of the Jacobian of the latent layer. We adapted network topologies from previous applications, but we would like to perform ablation studies to assess the degradation of the proposed architectures. Finally, other NN types could be tested, namely generative adversarial networks (GANs) or invertible NNs.

6. Conclusion

We evaluated three different AEs for generating *parametric* and *invertible* multidimensional data projections. Our qualitative and quantitative results for t-SNE showed the differences in projection and reconstruction capabilities between the tested models. In general, we found that feed-forward NNs for the projection and reconstruction of data points generally outperform AE-based approaches in terms of accuracy. However, AEs can produce comparable results while generally producing smoother parametric and inverse projections. In particular, we found that VAEs with a customized loss function have the greatest potential for producing smooth inverse projections. Their parameterization allows for case-by-case fine-tuning between overall accuracy and smoothness.

Acknowledgments – This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 251654672 – TRR 161 (Project A03).

References

- [ABM*15] AMORIM, ELISA, BRAZIL, EMILIO VITAL, MENA-CHALCO, JESÚS, et al. “Facing the high-dimensions: Inverse projection with radial basis functions”. *Comput. Graph.* 48 (2015), 35–47. DOI: [10.1016/J.CAG.2015.02.009](#) 2.
- [AEC*22] APPLEBY, GABRIEL, ESPADOTO, MATEUS, CHEN, RUI, et al. “HyperNP: Interactive Visual Exploration of Multidimensional Projection Hyperparameters”. *Comput. Graph. Forum* 41.3 (2022), 169–181. DOI: [10.1111/CGF.14531](#) 1–3.
- [AGO*12] ANGUITA, DAVIDE, GHIO, ALESSANDRO, ONETO, LUCA, et al. “Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine”. *Ambient Assist. Living Home Care*. Springer, 2012, 216–223 3.
- [BBH12] BUNTE, KERSTIN, BIEHL, MICHAEL, and HAMMER, BARBARA. “A General Framework for Dimensionality-Reducing Data Visualization Mapping”. *Neural Comput.* 24.3 (2012), 771–804. DOI: [10.1162/NECO_A_00250](#) 2.
- [BKG23] BANK, DOR, KOENIGSTEIN, NOAM, and GIRYES, RAJA. “Autoencoders”. *Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook*. Springer, 2023, 353–374. DOI: [10.1007/978-3-031-24628-9_16](#) 1, 2.
- [BWT*24] BLUMBERG, DANIELA, WANG, YU, TELEA, ALEXANDRU, et al. “Inverting Multidimensional Scaling Projections Using Data Point Multilateration”. *15th Int. EuroVis Workshop Vis. Anal.* 2024. DOI: [10.2312/eurova.20241112](#) 2, 3.
- [CG15] CUNNINGHAM, JOHN P. and GHAHRAMANI, ZOUBIN. “Linear dimensionality reduction: survey, insights, and generalizations”. *J. Mach. Learn. Res.* 16 (2015), 2859–2900. DOI: [10.5555/2789272.2912091](#) 2.
- [CG24] CHEN, FLORIAN and GÄRTNER, THOMAS. “Scalable Interactive Data Visualization”. *Mach. Learn. Knowl. Discov. Databases*. Vol. 14948. 2024, 429–433. DOI: [10.1007/978-3-031-70371-3_34](#) 3.
- [DMKE24] DENNIG, FREDERIK L., MILLER, MATTHIAS, KEIM, DANIEL A., and EL-ASSADY, MENNATALLAH. “FS/DS: A Theoretical Framework for the Dual Analysis of Feature Space and Data Space”. *IEEE Trans. Vis. Comput. Graph.* 30.8 (2024), 5165–5182. DOI: [10.1109/TVCG.2023.3288356](#) 1.
- [dSBI*12] DOS SANTOS AMORIM, ELISA PORTES, BRAZIL, EMILIO VITAL, II, JOEL DANIELS, et al. “iLAMP: Exploring high-dimensional spacing through backward multidimensional projection”. *7th IEEE Conf. Vis. Anal. Sci. Technol.* 2012, 53–62. DOI: [10.1109/VAST.2012.6400489](#) 2.
- [EAS*21] ESPADOTO, MATEUS, APPLEBY, GABRIEL, SUH, ASHLEY, et al. “UnProjection: Leveraging Inverse-Projections for Visual Analytics of High-Dimensional Data”. *IEEE Trans. Vis. Comput. Graph.* 29.2 (2021), 1559–1572. DOI: [10.1109/TVCG.2021.3125576](#) 1–3.
- [EHT20] ESPADOTO, MATEUS, HIRATA, NINA SUMIKO TOMITA, and TELEA, ALEXANDRU C. “Deep learning multidimensional projections”. *Inf. Vis.* 19.3 (2020), 247–269. DOI: [10.1177/1473871620909485](#) 1, 2.
- [EHT21] ESPADOTO, MATEUS, HIRATA, NINA S. T., and TELEA, ALEXANDRU C. “Self-supervised Dimensionality Reduction with Neural Networks and Pseudo-labeling”. *16th Int. Jt. Conf. Comput. Vis. Imaging Comput. Graph. Theory Appl.* 2021, 27–37. DOI: [10.5220/0010184800270037](#) 2.
- [EMK*19] ESPADOTO, MATEUS, MARTINS, RAFAEL MESSIAS, KERREN, ANDREAS, et al. “Toward a Quantitative Survey of Dimension Reduction Techniques”. *IEEE Trans. Vis. Comput. Graph.* 27.3 (2019), 2153–2173. DOI: [10.1109/TVCG.2019.2944182](#) 2, 3.
- [HHKS23] HINTERREITER, ANDREAS P., HUMER, CHRISTINA, KAINZ, BERNHARD, and STREIT, MARC. “ParaDime: A Framework for Parametric Dimensionality Reduction”. *Comput. Graph. Forum* 42.3 (2023), 337–348. DOI: [10.1111/CGF.14834](#) 1, 2.
- [HMP*17] HIGGINS, IRINA, MATTHEY, LOIC, PAL, ARKA, et al. “beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework”. *5th Int. Conf. Learn. Represent.* 2017 3.
- [HS06] HINTON, G. E. and SALAKHUTDINOV, R. R. “Reducing the Dimensionality of Data with Neural Networks”. *Science* 313.5786 (2006), 504–507. DOI: [10.1126/science.1127647](#) 2.
- [JCC*11] JOIA, PAULO, COIMBRA, DANILO BARBOSA, CUMINATO, JOSÉ ALBERTO, et al. “Local Affine Multidimensional Projection”. *IEEE Trans. Vis. Comput. Graph.* 17.12 (2011), 2563–2571. DOI: [10.1109/TVCG.2011.220](#) 2.
- [Jol86] JOLLIFFE, IAN T. *Principal Component Analysis*. Springer, 1986. DOI: [10.1007/978-1-4757-1904-8](#) 1, 2.
- [KB15] KINGMA, DIEDERIK P. and BA, JIMMY. “Adam: A Method for Stochastic Optimization”. *3rd Int. Conf. Learn. Represent.* 2015 2, 3.
- [Kru78] KRUSKAL, JOSEPH B. “Multidimensional scaling”. *Murry Hill* (1978) 2.
- [KW14] KINGMA, DIEDERIK P. and WELLING, MAX. “Auto-Encoding Variational Bayes”. *2nd Int. Conf. Learn. Represent.* 2014 3.
- [LCB98] LECUN, YANN, CORTES, CORINNA, and BURGES, CHRIS. *MNIST handwritten digit database*. 1998 3.
- [LMW*17] LIU, SHUSEN, MALJOVEC, DAN, WANG, BEI, et al. “Visualizing High-Dimensional Data: Advances in the Past Decade”. *IEEE Trans. Vis. Comput. Graph.* 23.3 (2017), 1249–1268. DOI: [10.1109/TVCG.2016.2640960](#) 2.
- [MHM18] MCINNES, LELAND, HEALY, JOHN, and MELVILLE, JAMES. “UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction”. *CoRR* abs/1802.03426 (2018) 2.
- [MTB24] MACHADO, ALISTER, TELEA, ALEXANDRU C., and BEHRISCH, MICHAEL. “Controlling the scatterplot shapes of 2D and 3D multi-dimensional projections”. *Comput. Graph.* 124 (2024), 104093. DOI: [10.1016/J.CAG.2024.104093](#) 2.
- [NA19] NONATO, LUIS GUSTAVO and AUPETIT, MICHAËL. “Multidimensional Projection for Visual Analytics: Linking Techniques with Distortions, Tasks, and Layout Enrichment”. *IEEE Trans. Vis. Comput. Graph.* 25.8 (2019). DOI: [10.1109/TVCG.2018.2846735](#) 1, 2.
- [SMG21] SAINBURG, TIM, MCINNES, LELAND, and GENTNER, TIMOTHY Q. “Parametric UMAP Embeddings for Representation and Semisupervised Learning”. *Neural Comput.* 33.11 (2021), 2881–2907. DOI: [10.1162/NECO_A_01434](#) 1, 2.
- [SRK24] SCHLEGEL, UDO, RAUSCHER, JULIUS, and KEIM, DANIEL A. “Interactive Counterfactual Generation for Univariate Time Series”. *6th Int. Workshop Explain. Knowl. Discov. Data Min.* 2024 1.
- [vdMaa09] Van der MAATEN, LAURENS. “Learning a Parametric Embedding by Preserving Local Structure”. *12th Int. Conf. Artif. Intell. Stat.* Vol. 5. 2009, 384–391 1, 2.
- [vdMH08] Van der MAATEN, LAURENS and HINTON, GEOFFREY. “Visualizing Data using t-SNE”. *J. Mach. Learn. Res.* 9.86 (2008), 2579–2605 2, 3.
- [vWvO03] Van WIJK, JARKE J. and van OVERVELD, CORNELIUS W. A. M. “Preset based interaction with high dimensional parameter spaces”. *Data Visualization: The State of the Art*. 2003, 391–406 1, 2.
- [WYZ16] WANG, YASI, YAO, HONGXUN, and ZHAO, SICHENG. “Auto-encoder based dimensionality reduction”. *Neurocomputing* 184 (2016), 232–242 2.
- [XRV17] XIAO, HAN, RASUL, KASHIF, and VOLLGRAF, ROLAND. “Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms”. *CoRR* abs/1708.07747 (2017) 3.
- [Yin07] YIN, HUIJUN. “Nonlinear dimensionality reduction and data visualization: A review”. *Int. J. Autom. Comput.* 4.3 (2007), 294–303. DOI: [10.1007/s11633-007-0294-y](#) 2.