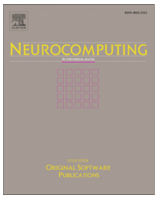
Contents lists available at ScienceDirect

Neurocomputing

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



Additive autoencoder for dimension estimation

Tommi Kärkkäinen [*], Jan Hänninen

Faculty of Information Technology, University of Jyväskylä, Finland

ARTICLE INFO

Article history:
Received 10 February 2023
Revised 15 May 2023
Accepted 30 June 2023
Available online 6 July 2023
Communicated by Zidong Wang

Keywords:
Autoencoder
Dimension reduction
Intrinsic dimension
Deep learning

ABSTRACT

Dimension reduction is one of the key data transformation techniques in machine learning and knowledge discovery. It can be realized by using linear and nonlinear transformation techniques. An additive autoencoder for dimension reduction, which is composed of a serially performed bias estimation, linear trend estimation, and nonlinear residual estimation, is proposed and analyzed. Compared to the classical model, adding an explicit linear operator to the overall transformation and considering the nonlinear residual estimation in the original data dimension significantly improves the data reproduction capabilities of the proposed model. The computational experiments confirm that an autoencoder of this form, with only a shallow network to encapsulate the nonlinear behavior, is able to identify an intrinsic dimension of a dataset with low autoencoding error. This observation leads to an investigation in which shallow and deep network structures, and how they are trained, are compared. We conclude that the deeper network structures obtain lower autoencoding errors during the identification of the intrinsic dimension. However, the detected dimension does not change compared to a shallow network. As far as we know, this is the first experimental result concluding no benefit from a deep architecture compared to its shallow counterpart.

1. Introduction

Dimension reduction is one of the main tasks in unsupervised learning [1]. Both linear and nonlinear techniques can be used to transform a set of features into a smaller dimension [2]. A specific and highly popular set of nonlinear methods are provided with autoencoders, AE [3], which by using the original inputs as targets integrate unsupervised and supervised learning for dimension reduction. The main purpose of this paper is to propose and thoroughly test a new autoencoding model, which comprises an additive combination of linear and nonlinear dimension reduction techniques through serially performed bias estimation, linear trend estimation, and nonlinear residual estimation. Preliminary, limited investigations of such a model structure have been reported in [4,5].

With the proposed autoencoding model, we consider its ability to estimate the intrinsic dimensionality of data [6–8]. According to [9], the intrinsic dimension can be defined as the size of the lower-dimension manifold where data lies without information loss. For the most popular linear dimension reduction technique, the principal component analysis (PCA), the information loss can be measured with the explained variance which is provided by the eigenvalues of the covariance matrix [10]. Indeed, the use of an autoencoder to estimate the intrinsic dimension can be considered a nonlinear extension of the projection method based on PCA [11]. However, in the nonlinear case measures for characterizing the essential information and how this is used to reduce the dimensionality vary [12,13].

With the intrinsic dimension, for instance the data reduction step in the KDD process [14,15] would not loose information. In [16] it was concluded that a shallow autoencoder shows the best performance when the size of the squeezing dimension is approximately around the intrinsic dimension. In this direction, techniques that are closely related to our work were proposed in [17], where the intrinsic dimension was estimated using an autoencoder with sparsity-favoring l_1 penalty and singular value proxies of the squeezing layer's encoding. In the experiments, the superiority of the autoencoder compared to PCA was concluded. This and the preliminary work by [18] applied *a priori fixed architectures of the autoencoder and different autoencoding error measure compared to our work. Here, multiple feedforward models are used and compared, with a simple thresholding technique to detect the intrinsic dimension based on the data reconstruction error.*

[*] Corresponding author.
E-mail address: tommi.karkkainen@jyu.fi (T. Kärkkäinen).

https://doi.org/10.1016/j.neucom.2023.126520
0925-2312/© 2023 The Author(s). Published by Elsevier B.V.

1.1. Motivation of the autoencoding model

The proposed model is formally derived in Section 3.1, but the basic idea to apply sequentially linear and nonlinear transformation models for dimension reduction and estimation could be realized in many ways. However, in order to reduce the reconstruction error efficiently– and to reveal the instrinsic dimension– there is one pivotal selection which is illustrated in Fig. 1: *After the linear transformation, residual for the nonlinear model's encoding should be treated in the original dimension!*.

Interestingly, our experiments reveal that the intrinsic dimension can be identified by using only a shallow feedforward network as the nonlinear residual operator in the additive autoencoding model. This results exactly from the two essential choices: *i*) to include the linear operator in the overall transformation and *ii*) to consider the unexplained residual in the original data dimension. This does not happen with the classical autoencoder (without the linear term) or if the nonlinear autoencoding would be applied in the reduced dimension after the linear transformation. These phenomena, with the models and techniques fully specified in the subsequent sections, are illustrated in Fig. 1. Therefore, in addition to exploring the capabilities of revealing the intrinsic dimension we assess the advantages of applying deeper networks as nonlinear operators.

1.2. Contributions and contents

The main contribution of the paper is the derivation and evaluation of the additive autoencoding model. We provide an experimental confirmation of the new autoencoder's ability to reveal the intrinsic dimension and study the effect of model depth. With minor role, mostly covered in the Supplementary Material (SM), we also discuss and provide an experimental illustration of the difficulties of currently popular deep learning techniques in realizing the potential of deep network models. Apparently, our results diversify views on the general superiority of deeper network architectures. They suggest that current and upcoming applications in deep learning and in the use of autoencoders could be improved by using an explicit separation of the linear and nonlinear aspects of the data-driven model.

The remainder of the paper is organized in the following manner: In Section 2, we provide a brief summary of the forms and applications of autoencoders and provide a preliminary discussions on certain aspects of deep learning. In Section 3, we discuss the formalization of the proposed method as a whole. In Section 4, we describe the computational experiments and summarize the main findings. In Section 5, we provide the overall conclusions and discussion. In the SI, we provide more background and preliminary material and, especially, report the computational experiments as

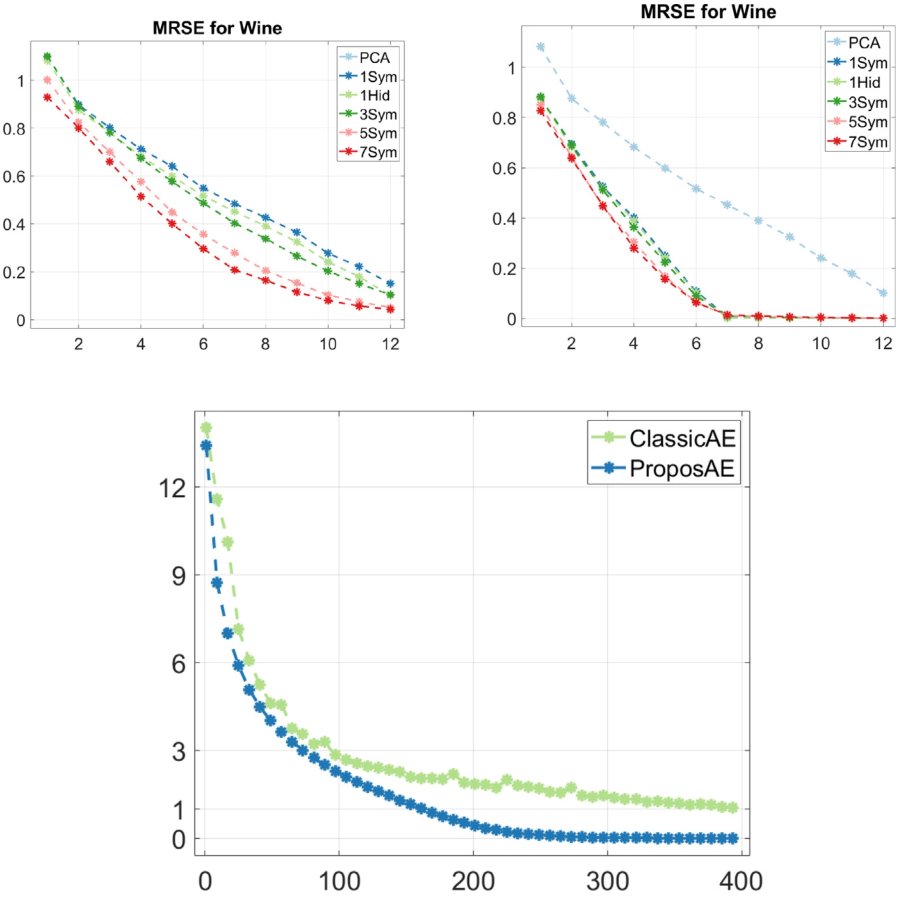


Fig. 1. Top row: Residual errors with Wine dataset for the usual autoencoders with different number of hidden layers (left) and for the proposed, additive autoencoders (right). **Bottom row:** Comparison of classical and proposed autoencoder with five-hidden-layers for the MNIST dataset. Here *x*-axis contains the squeezing dimension and *y*-axis the autoencoding (reconstruction) error. Deeper models provide lower autoencoding errors on the top left, but, only with the additional linear operator as proposed here, the intrinsic dimension is revealed on the top right and bottom figures. Improvement due to the depth of the model is significant on the top left but only moderate on the top right, where all models are stricly better than the linear PCA alone. On the bottom, better capability of the proposed autoencoder to encapsulate the variability of MNIST compared to the classical approach is clearly visible.

a whole. Main findings solely covered on SI confirm the quality of the implementation of the methods and especially indicate that different autoencoder models, as depicted in the previous section, could be used for the nonlinear residual estimation.

2. Related work and preliminaries

In this section, we introduce the different variants of autoencoders, summarize their wide range of applications, and provide some preliminaries on deep learning techniques.

2.1. On Autoencoders

Feedforward autoencoders have a versatile history, beginning from [19,20]. Their development as part of the evolution from shallow network models into deep learning techniques with various network architectures has been depicted in numerous large and comprehensive reviews [3,21,22]. Therefore, we will not go into details in what follows.

Deep feedforward autoencoding was highly influenced by the seminal work of Hinton and Salakhutdinov [23]. In what follows, we refer this approach, where the encoder was composed of multiple feedforward layers until the squeezing dimension and estimation of the network weights was based on the least-squares fidelity, possibly with a Tykhonov type of squared regularization (penalization) term [24], as *the classical autoencoder*. The work in [23] emphasized the importance of pretraining and the usability of stacking (i.e., layer-by-layer construction of the deeper architecture). Such techniques, by directing the determination of weights to a potential region of the search space, particularly alleviate the vanishing gradient problem (see Section 5.9 [3] and references therein).

As summarized– for example, by [25], many architectures and training variants for deep autoencoders (AE) have been proposed over the years. Summary of the main AE approaches is given in Table 1, where 'Architecture' refers to the AE's model structure, 'Learning principle' summarizes the essential basis of learning, 'Evaluation' contains the given information on the assessment of quality, efficiency and complexity, and 'Applications' points to the basic use cases within the cited papers.

In addition to the approaches given in Table 1, many other combinations and hybrid forms of AEs exist. *Special AEs* typically integrate concepts and techniques from relevant areas, for instance, autoencoder bottlenecks (AE-BN) that are based on Deep Belief Networks [26] and rough autoencoders where the rough set based neurons are used in the layers [27]. In *Methodological combinations with AEs*, the base case is to use AE for dimension reduction before supervised learning. Interesting unsupervised hybrids are provided by clustering techniques that incorporate AEs for feature transformation [28,29] and unsupervised AE-based hashing methods that can be used for large-scale information retrieval [30]. More involved combinations include use of AE with unsupervised clustering and generative models [31,32] and for representation learning of various learning tasks, e.g., low-dimensional visualization, semantic compression, and data generation [33].

A wide variety of tasks and domains has been and can be addressed with autoencoders. AEs are typically used in numerous application domains in scenarios where transfer learning can be utilized– for example, in speech processing [55], time series prediction [56], fault diagnostics in condition monitoring [57–59], and machine vision and imaging tasks [60,61,33,62,63,44,64–68]. Further, AEs have been used for the estimation of data distribution [69]. Use of a variational AE for joint estimation of a normal latent data distribution and the corresponding contributing dimensions has been addressed in [70]. Yet another use case of autoencoders is outlier or anomaly detection [71,33,72–74]. Also data imputation has been realized using a shallow autoencoder in [75], with deeper models in [76–78], and especially for spatio-temporal data in [79,76,80–83]. More exotic use cases include (but are not limited to) use of AE to transfer domain knowledge in recommendation systems [84] and to construct efficient surrogates in astronomy [85].

2.2. Preliminaries on deep learning

Deep learning has been an extremely active field of research and development in the twenty-first century [86]. These techniques reveal repeatedly promising results in new application areas [87]. On the other hand, deep networks might be overly complex and equal performance could be reestablished after significant pruning [88]. Deep neural networks are sensitive to small variations in training and architectural design parameters, thereby requiring careful calibration [89] and meticulousness in analyzing and comparing different models [90]. These factors have caused, e.g., the emergence of automated neural architecture search techniques [91]. Therefore, it is important to improve our basic understanding of both the empirical and theoretical bases of deep learning [92]. Systematic methods and studies to analyze the behavior of deep neural networks are needed [93].

Indeed, there exist some fundamental aspects of deep learning in which theory and practice are not fully aligned. The first issue is the universal approximation capability, whose main shallow results are reviewed, for example, by Pinkus [94] and whose general importance was excellently summarized in [95, p. 363]: "We have thus established that such [feedforward] 'mapping' networks are universal approximators. This implies that any lack of success in applications must arise from inadequate learning, insufficient numbers of hidden units or the lack of a deterministic relationship between input and target." To put it succinctly, a sufficiently large width of one or at most two hidden layers is, according to the approximation theory, enough for an accurate approximation of a nonlinear function, which is implicitly represented by a discrete set of examples in the training data.

Another aspect is the dominant approach to training a deep network structure– that is, estimating unknown weights in different layers. This is realized by applying a certain form of the steepest gradient descent method with a rough approximation of the true gradient, using one observation (online or stochastic gradient descent) or a small subset of observations (minibatch). In classical optimization, even the batch gradient descent using the complete data is considered a slow algorithm but still a convergent one provided that the gradient of the minimized function is Lipschitz continuous and the line search– that is, the determination of the step size (the learning rate in neural networks terminology) when moving to the search direction– satisfies specific decrease conditions as given in Theorems 6.3.2 and 6.3.3 in [96] and Theorem 3.2 in [97]. In deep learning, step size may be fixed to a small positive constant [98] or be based on monitoring the first- and second-order moments of the gradient during the search with direct updates [99]. Often restricted to a fixed number of iterations and not assuring stopping criteria measuring fulfillment of the optimality conditions (see Section 7.2 in [96]), this implies that the actual optimization problem for determining the weights of a DNN might be solved inaccurately [100].

In a genuine supervised learning for a regression model or a classifier, inexact optimization can be tolerated when seeking the best generalizing network. Then, the search of weights that provide better minimizers for the cost/loss function is stopped prematurely when the test or validation error of the model begins increasing as an indication of overlearning. In such a case, we are not seeking the most accurately optimized network but the best generalizing

Table 1
Summary of main forms of autoencoders.

Technique	Architecture	Learning principle	Evaluation	Applications

Denoising AE (DAE) [34–39]	Classical	Add noise to inputs [34] or pretraining [34] or weight updates [35]	Linear complexity for Marginalized Stacked Linear DAE [36];
			Higher imputation accuracy [38]
				Better higher-order representations [34];
				Time series classification [37];
				Generative models for genetic operators [39];
				Data imputation [38]

Sparse AE (SAE) [40] (see also Section 5.6.4 [3])	Classical	Introduce sparsity favoring regularization and special solvers	Reduced models with less active weights	High-dimensional regression and classification problems, e.g., in Chemometrics
Separable AE (SAE) [41] (cf. Siamese neural networks that do opposite and use shared weights)	Classical	Pretrain multilayer AE for speech and on-the-fly multi-layer AE for residual noise from Spectrogram with nonnegativity constraint	Quality of enhanced speech, analysis of noise distribution	Separate speech and noise
Graph AE (GAE) [42–47] Variational GAE [48] (see VAE next)	Schrinking and enlarging forward–backward transformation of data graph (GNN); Use AE to encode-decode Graph layer Convolution Network [43]; Masked GAE [46]; Encode-decode both topology and node attributes [47]; GNN encoder + separate feature&label decoders [48]	Reconstruction of adjacency matrix in least-squares sense [42] or cross-entropy sense [43]; Apply weight-least-squares and graph Laplacian regularization [44,48]; Regularize both topology and node attribute proximity [47]	Use latent representation to test preservation of topological features or node or graph structure; Classification and clustering quality; Linear training scalability [48]	Suppressed graph encapsulation of linked data representing citations, emails, flights, species, social networks, blockchains, collaborations, functional magnetic resonance imaging (fMRI); Estimation of missing feature information on nodes for link prediction [48]
Variational AE (VAE) [49–52]	Structurally classical, input and/or latent distributions, generative use emphasized [49,50]	Stochastic fitting to samples and regularization of latent space, e.g., with Kulback-Leibler divergence, non-convex optimization problems [49,50]	Recovery of generated latent space [49], size of latent space [49], quality of generated topics [50]; imbalanced classification quality [51,52]; Quality of density estimation and number of active units [52]	Generation of topics [50], Generation of samples for minority class [51]
Regularized AE (RAE) [53]	Any	Combination of classical reconstruction error and contractive regularizer (encoder-decoder derivative with respect to input)	Function approximation, recovery of manifolds	Recovery of data-generation density
Multi-modal AE [54]	Set autoencoder to process different data modalities, recurrent units	Input–output order matching using sum of reconstruction error and cross-entropy loss between set memberships	Quality of set assignments and matching, quality of kernel density estimation	Genes-to-proteins mapping

model based on another error criterion at the meta-optimization level. Theoretically, however, generalization and optimality can be linked in certain respects: An outer-layer locally optimal network, independently on the level of optimality of the hidden layers, provides an unbiased estimate of the prediction error over the training data in the sense of mean, median, or spatial median [101].

In a typical use case, the goal of a dimension reduction through autoencoding is to obtain a reduced, compact representation that encapsulates variation of data. Similar to linear dimension reduction techniques, when attempting to represent the given data accurately in lower dimensions, we aim for the best possible reconstruction accuracy. Then, the cost function that measures the autoencoding error (such as the least-squares error function) must be solved with sufficiently high optimization accuracy because of the direct correlation: The smaller the cost function, the better the autoencoder.

3. Methods

In this section, we describe the methods used here as part of the autoencoding approach. In the following account, we assume that a training set of N observations $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^n$, is given.

3.1. The autoencoding model

In mathematical modelling, linear and nonlinear models are typically treated separately [102]. Following [5], according to Taylor's formula, in the neighborhood of a point $\mathbf{x}_0 \in \mathbb{R}^n$, the value of a sufficiently smooth real-valued function $f(\mathbf{x})$ can be approximated as

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^T(\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T \nabla^2 f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) + \dots,$$

where $\nabla f(\mathbf{x}_0)$ denotes the gradient vector and $\nabla^2 f(\mathbf{x}_0)$ the Hessian matrix at $\mathbf{x}_0$. According to Lemma 4.1.5 [96], there exists $\mathbf{z} \in ls(\mathbf{x}, \mathbf{x}_0), \mathbf{z} \in \mathbb{R}^n$, where ls refers to the line segment connecting the two points $\mathbf{x}$ and $\mathbf{x}_0$, such that

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^T(\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T \nabla^2 f(\mathbf{z})(\mathbf{x} - \mathbf{x}_0). \tag{1}$$

This formula yields the sufficient condition of $\mathbf{x} \in \mathbb{R}^n$ to be the local minimizer of a convex f (whose Hessian is positive semidefinite) Theorem 2.2 [97]: $\nabla f(x) = 0$. However, another interpretation of the formula above is that we can locally approximate the value of a smooth function as a sum of its bias (i.e., constant level), a linear term, and a nonlinear higher-order residual operator. This observation is the starting point for proposing an autoencoder that has exactly such an additive structure.

The bias estimation simply involves the elimination of its effect through normalization by subtracting the data mean and scaling each feature separately into the same range $[-1, 1]$ with the scaling factor $\frac{2}{\max(x) - \min(x)}$. Thus, we combine the mean component from z-scoring and the scaling component from min–max scaling. The reason for this is that the unit value of the standard deviation in z-scoring does not guarantee equal ranges, and min–max scaling into $[-1, 1]$ does not preserve the zero mean.

In the second phase, we estimate the linear behavior of the normalized data by using the classical principal component transformation [103]. For a zero-mean vector $\mathbf{x} \in \mathbb{R}^n$, the transformation to a smaller-dimensional space $m < n$ spanned by m principal components (PCs) is given by $\mathbf{y} = \mathbf{U}^T\mathbf{x}$, where $\mathbf{U} \in \mathbb{R}^{n \times m}$ consists of the m most significant (as measured by the sizes of the corresponding eigenvalues) eigenvectors of the covariance matrix. Thus, because of the orthonormality of $\mathbf{U}$, the unexplained residual variance of the PC coordinates (i.e., the linear trend in $\mathbb{R}^m$) *in the original space* (see SI, Section 7) can be estimated in the following manner:

$$\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{U}\mathbf{y} = \mathbf{x} - \mathbf{U}\mathbf{U}^T\mathbf{x} = \left(\mathbf{I} - \mathbf{U}\mathbf{U}^T\right)\mathbf{x}. \tag{2}$$

This transformation is referred to as PCA. With erroneous data or data with missing values, mean and classical PCA can be replaced with their statistically robust counterparts [104].

In the third, nonlinear phase, we apply the classical fully connected feedforward autoencoder to the residual vectors in (2). We note that structurally, in comparison with the classical autoencoding model [23] (see Section 2.1), the difference according to (2) is the inclusion of the linear dimension reduction operator acting on the original dimension. To use the same data dimension in the linear approximation is suggested by the Taylorian analogy according to Eq. (1). Furthermore, one needs to return to the original dimension in the linear approximation, because otherwise (i.e., if we would consider the residual in the reduced dimension m) the overall reconstruction error would be constrained by the accuracy of the linear part. Here, instead, when the residual data is represented in the original dimension, both the linear and nonlinear parts of the additive autoencoder contribute to the reduction of the autoencoding error in the original data dimension.

As anticipated by the scaling, the tanh activation function $f(x) = \frac{2}{1+\exp(-2x)} - 1$ is used. This ensures the smoothness of the entire transformation and the optimization problem of determining the weights. The currently popular rectified linear units are nondifferentiable [101] and, therefore, are not theoretically compatible with the gradient-based methods Section 6.3.1 [98]. The importance of differentiability was also noted in [105].

The formalism introduced by [24] is used for the compact derivation of the optimality conditions. By representing the layer-wise activation using *diagonal function-matrix* $\mathcal{F} = \mathcal{F}(\cdot) = \mathbb{D}iag\{f_i(\cdot)\}_{i=1}^m$, where $f_i \equiv f$, the output of a feedforward network with L layers and linear activation on the final layer reads as

$$\mathbf{o} = \mathbf{o}^L = \mathcal{N}(\tilde{\mathbf{x}}) = \mathbf{W}^L\mathbf{o}^{(L-1)}, \tag{3}$$

where $\mathbf{o}^0 = \tilde{\mathbf{x}}$ for an input vector $\tilde{\mathbf{x}} \in \mathbb{R}^{n_0}$, and $\mathbf{o}^l = \mathcal{F}\left(\mathbf{W}^l\mathbf{o}^{(l-1)}\right)forl = 1, \dots, L-1$. To allow the formal adjoint transformation to be used as the decoder, we assume that L is even and that the bias nodes are not included in (3). The dimensions of the weight matrices are then given by $\dim\left(\mathbf{W}^l\right) = n_l \times n_{l-1},\ l = 1, \dots, L$. In the autoencoding context, $n_L = n_0$ and $n_l, 0 < l < L$, define the sizes (the number of neurons) of the hidden layers with the squeezing dimension $n_{L/2} < n_0$.

To determine the weights in (3), we minimize the regularized mean least-squares cost function of the form

$$\mathcal{J}\left(\left\{\mathbf{W}^l\right\}_{l=1}^L\right) = \frac{1}{2N}\sum_{i=1}^N \left\|\mathbf{W}^L\mathbf{o}_i^{(L-1)} - \tilde{\mathbf{x}}_i\right\|^2 + \frac{\alpha}{2\sqrt{\sum_{l=1}^L \#\left(\mathbf{W}_1^l\right)}} \sum_{l=1}^L \left\|\mathbf{W}^l - \mathbf{W}_0^l\right\|^2, \tag{4}$$

where $\|\cdot\|$ denotes the Frobenius norm and $\#\left(\mathbf{W}_1^l\right)$ the number of rows of $\mathbf{W}^l$. Let α be fixed to 1e-6 throughout; to simplify the notations, we define $\beta = \alpha/\sqrt{\sum_{l=1}^L |\mathbf{W}_1^l|}$. The underlying idea in (4) is to average in both terms: in the first, the data fidelity (least-squares error, LSE) term, and in the second, the regularization term. Averaging the first term with $\frac{1}{N}$ implies that the term scales automatically by the size of the data subset, for instance, in minibatching, thereby providing an approximation of the entire LSE on a comparable scale. In the second term, because α is fixed, the inverse scaling constant $1/\sqrt{\sum_{l=1}^{L}|\mathbf{W}_1^l|}$ balances the effect of the regularization compared to the data fidelity for networks with a different number of layers of different sizes. Because (4) will be minimized with local optimizers, we simply use the initial guesses $\left\{\mathbf{W}_0^l\right\}$ of the weight values in the second term to improve the local coercivity of (4) and to restrict the magnitude of the weights, thereby attempting to improve generalization [106]. Because of the residual approximation, the random initialization of the weight matrices is generated from the uniform distribution $\mathcal{U}([-0.1, 0.1])$. s

The gradient matrices $\nabla_{\mathbf{W}^l}\mathcal{J}\left(\left\{\mathbf{W}^l\right\}_{l=1}^{L}\right)$, $l = L, \ldots, 1$, for (4) are of the following form (see [24]):

$$\nabla_{\mathbf{W}^l}\mathcal{J}\left(\left\{\mathbf{W}^l\right\}\right) = \frac{1}{N}\sum_{i=1}^{N}\mathbf{d}_i^l\left(\mathbf{o}_i^{(l-1)}\right)^T + \beta\left(\mathbf{W}^l - \mathbf{W}_0^l\right), \tag{5}$$

where the layerwise error backpropagation reads as

$$\mathbf{d}_i^L = \mathbf{e}_i = \mathbf{W}^L\mathbf{o}_i^{(L-1)} - \tilde{\mathbf{x}}_i, \tag{6}$$

$$\mathbf{d}_i^l = \mathbb{D}iag\left\{(\mathcal{F})'\left(\mathbf{W}^l\mathbf{o}_i^{(l-1)}\right)\right\}\left(\mathbf{W}^{(l+1)}\right)^T\mathbf{d}_i^{(l+1)}. \tag{7}$$

The use of different weights in the encoding– that is, in the transformation until layer $L/2$– and decoding, from layer $L/2$ to L, implies more flexibility in the residual autoencoder but also rougly doubles the amount of weights to be determined. Therefore, it is common to use the tight weights, which means that the formal adjoint $\left(\mathbf{W}^1\right)^T\mathcal{F}\left(\left(\mathbf{W}^2\right)^T\mathcal{F}\left(\ldots\left(\mathbf{W}^{L/2}\right)^T\right)\right)$ of the encoder is used as the decoder. Then, it is easy to see that the layerwise optimality conditions for $l = 1, \ldots, L/2$ read as

$$\nabla_{\mathbf{W}^l}\mathcal{J} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{d}_i^l\left(\mathbf{o}_i^{(l-1)}\right)^T + \mathbf{o}_i^{(\tilde{l}-1)}\left(\mathbf{d}_i^{\tilde{l}}\right)^T) + \beta\left(\mathbf{W}^l - \mathbf{W}_0^l\right), \tag{8}$$

where $\tilde{l} = L - (l-1)$. For convenience, we define $\widetilde{L} = L/2$– that is, the number of layers to be optimized with the symmetric models.

We note that when the layerwise formulae above are used with vector-based optimizers, we always need to reshape operations to toggle between the weight matrices and a column vector of all weights.

Remark 1. Let us briefly summarize the use of the additive autoencoder for an unseen dataset after it has been estimated (and the corresponding data structures have been stored) for the training data through the three phases. First, data is normalized through mean subtraction and feature scaling into the same range $[-1, 1]$. Then, residuals according to formula (2) are computed and this residual data is fed to the feedforward autoencoder. Again due to (2), the reduced, m-dimensional representation of new data is obtained as a sum of its PC projection and the output of the autoencoder's squeezing layer. Formula (2) shows that the explicit formation of the residual data between linear and nonlinear representations can be replaced by setting $\widetilde{\mathbf{W}}^1 = \mathbf{W}^1\left(\mathbf{I} - \mathbf{U}\mathbf{U}^T\right)$ and using this as the first transformation layer of the autoencoder for the normalized, unseen data.

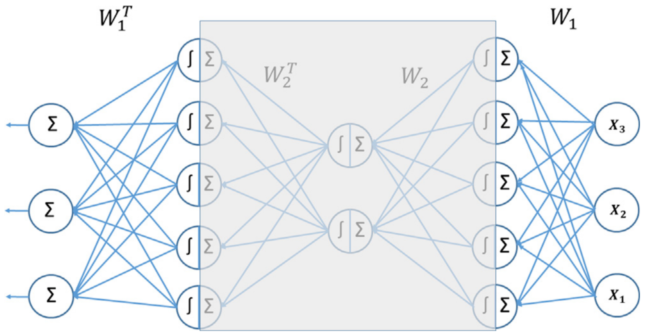


Fig. 2. Layerwise pretraining from heads to inner layers. The most outer layer is trained first and its residual is then fed as training data for the next hidden layer until all layers have been sequentially pretrained.

3.2. Layerwise pretraining

We apply the classical stacking procedure depicted, e.g., in Section 6.2 [107]. A similar idea appears with the deep residual networks (ResNets) in [108], where consecutive residuals are stacked together using layer skips– for example, over two or three layers with batch normalization. However, in ResNets, the layer skips can introduce additional weight matrices whereas the layer-by-layer pretraining follows the originally chosen network architecture.

The stacking procedure is illustrated in Fig. 2. For three hidden layers with two unknown weight matrices, $\mathbf{W}^1$ and $\mathbf{W}^2$, we first estimate $\mathbf{W}^1$ with the given data $\{\tilde{\mathbf{x}}_i\}$. Then, the output data of the estimated layer $\left\{\mathbf{W}^1\tilde{\mathbf{x}}_i\right\}$ are used as the training data (the input and the desired output) for the second layer $\mathbf{W}^2$. Thereafter, the entire network is fine-tuned by optimizing over both weight matrices. The process from the heads to the inner layers is naturally enlarged for a larger number of hidden layers. We could then also apply partial fine-tuning– for example, to fine-tune the three hidden layers during the process of constructing a five-hidden-layer network. However, according to our tests and similar to [23], the layerwise pretraining suffices before fine-tuning the entire network. A special case of utilizing a simpler structure is the one-hidden-layer case: The symmetric model 1Sym with one weight matrix is first optimizer to obtain $\mathbf{W}^1$ and then used in the form $\left(\left(\mathbf{W}^1\right)^T, \mathbf{W}^1\right)$ as the initial guess for optimizing the nonsymmetric model 1Hid with two weight matrices. Again such an approach could be generalized to multiple hidden-layer case for the nonsymmetric, deep autoencoding model.

Remark 2. As stated in Section 2.1, stacking attempts to mitigate the vanishing gradient problem, which may prevent the adaptation of the weights in deeper layers. We assessed the possibility of such a phenomenon by studying the relative changes in the weight matrix norms $\left(|\|\mathbf{W}_0^l\| - \|\mathbf{W}_*^l\||/\|\mathbf{W}_0^l\|\right)$ while fine-tuning the symmetric autoencoders with 3–7 layers (3Sym, 5Sym, and 7Sym; see Section 4). The subscripts '0' and '$*$' refer to the initial and final weight matrix values, respectively. This study revealed that the relative changes in the weights in the deeper layers were not on a smaller numerical scale compared to the other layers. Apparently, the double role of the layers in the symmetric models as part of the encoder and the decoder, with the corresponding effect on the gradient as seen in formula (8), is also helpful in avoiding a vanishing gradient.

3.3. Determination of intrinsic dimension

The basic procedure to determine the intrinsic dimension is to gradually increase the size of the squeezing layer and to seek a small value of the reconstruction error measuring autoencoding error, with a knee point [109] indicating that the level of nondeter-

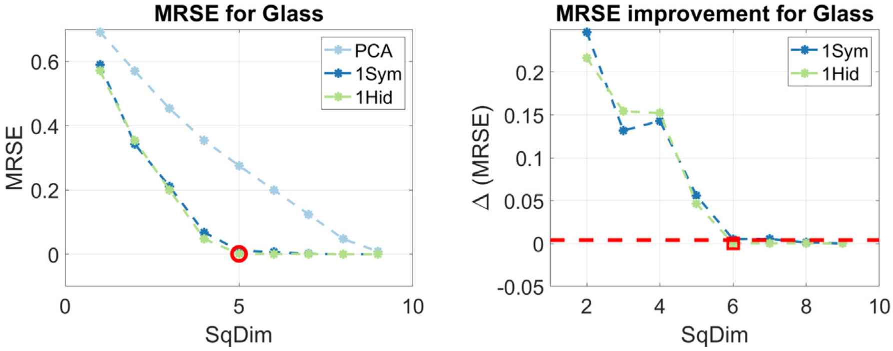


Fig. 3. Identification of the intrinsic dimension for the Glass dataset. The hidden dimension (plus one) on the left is captured by the sufficiently small error improvement on the right.

ministic residual noise has been reached in autoencoding. We apply the mean root squared error (MRSE) to compute the autoencoding error:

$$e = \frac{1}{N}\sqrt{\sum_{i=1}^{N}|\mathbf{x}_i - \mathcal{N}(\mathbf{x}_i)|^2}, \tag{9}$$

where $\{\mathbf{x}_i\}$ is assumed to be normalized and $\mathcal{N}$ denotes the application of the autoencoder. This choice was made because in [110], MRSE correlated better with the independent validation error than the usual root mean squared error RMSE, In practice, the difference between the RMSE and the MRSE is only the scaling factor, $1/\sqrt{N}$ vs. $1/N$. After the linear PC trend estimation, the MRSE is obtained by using (9) for the residual data defined in formula (2). Note that the reconstruction error is a strict error measure and its use requires higher accuracy from autoencoding compared to other measures: with the Wine dataset in Fig. 1, the linear PCA needs all dimensions of the rotated coordinate axis for the reconstruction whereas already 10 principal components out of 13 would explain over 96% of the data variance.

An example of determining the intrinsic dimension of the Glass dataset (see the next section) is presented in Fig. 3. In the figure, the x-axis "SqDim" presents the squeezing dimension and the y-axis on the left the "MRSE" and on the right its change "Δ(MRSE)" (i.e., backward difference) for the symmetric model with one hidden dimension (1Sym) and the corresponding nonsymmetric model 1Hid. The intrinsic dimension of data is detected by first locating a sufficiently small change in the autoencoding error (the right plot). For this purpose, a user-defined threshold τ = 4e-3 is applied. The detected dimension 5 on the left, marked with a circle, is the dimension below the threshold on the right minus one. The intrinsic dimension 5 is also characterized by a clear knee point in the MRSE behavior.

4. Results

The main focus of the computational experiments, which are fully reported in the Supplementary Material (SM), was to investigate the ability of the proposed additive autoencoder model to represent a dataset in a lower-dimensional space. Therefore, we confined ourselves to the use of Matlab as the platform (mimicking the experiments in [23]) to have full control over the realization of the methods in order to study the effects of different parameters and configurations. Reference implementation of the proposed method and its basic testing is available in GitHub[1].

We apply and compare the following set of techniques to approximate the nonlinear residual of the autoencoder, after normalization and identification of the linear trend: 1Hid (model with one hidden layer and separate weight matrices for the encoder and the decoder), 1Sym (symmetric model with one hidden layer and a shared weight matrix), 3Sym (three-hidden-layer symmetric model with two shared weight matrices), 5Sym, and 7Sym. To systematically increase the flexibility and the approximation capability of the deeper models, the sizes of the layers for $l = \widetilde{L}, \ldots, 1$ are given below, where $n_{\widetilde{L}}$ is the size of the squeezing layer:

3Sym:$n_{\widetilde{L}} - 2n_{\widetilde{L}} - n$,
5Sym:$n_{\widetilde{L}} - 2n_{\widetilde{L}} - 4n_{\widetilde{L}} - n$,
7Sym:$n_{\widetilde{L}} - 2n_{\widetilde{L}} - 3n_{\widetilde{L}} - 4n_{\widetilde{L}} - n$.

Note that for $n_{\widetilde{L}} > n/2$ the size of the second layer and, therefore, the dimension of the first intermediate representation, is larger than the input dimension for all these models.

4.1. Identification of the intrinsic dimension

The first purpose of the experiments was to search for the intrinsic dimension of a dataset via autoencoding. This was done using the shallow models 1Sym and 1Hid. The optimization settings and visualization of all results are given in the online SM.

The experiments were carried out for two groups of datasets, one with small-dimension data (less than 100 features) and the other with large-dimension data (up to 1024 features). The datasets were obtained from the UCI repository [111], except the FashMNIST, which was downloaded from GitHub[2]. For most of the datasets, only the training data was used; however, with Madelon, the given training, testing, and validation datasets were combined. The datasets do not contain missing values. The constant features were identified and eliminated according to whether the difference between the maximum and minimum values of a feature was less than $\sqrt{MEps}$, where $MEps$ denotes machine epsilon (this is classically used numerical proxy of zero, see [96, p. 12]). Because of this preprocessing, the number of features n in Tables 2 and 3 is not necessarily the same as that in the UCI repository.

During the search, the squeezing dimension for the small-dimension datasets began from one and was incremented one by one up to $n - 1$. For the large-dimension cases, we began from 10 and used increments of 10 until the maximum squeezing dimension $\lfloor 0.6 \times n \rfloor$ was reached (cf. the "Red" values in Tables 2 and 3). The experiments were run with Matlab on a Laptop with 2.3 GHz Intel i7 processor and 64 GB RAM and on a server with a Xeon E5-2690 v4 CPU and 384 GB of memory.

[1] https://github.com/TommiKark/AdditiveAutoencoder

[2] https://github.com/zalandoresearch/fashion-mnist

Table 2
Results of the identification of the intrinsic dimension for small-dimension datasets. The intrinsic dimensions were identified with the reduction rates varying between 0.41–0.54. The SteelPlates and COIL2000 (with the most discrete feature profile) have the best reduction rate. The residual errors are between 1.1e-2–4.3e-4.

Dataset	*N*	*n*	ID	Red	MRSE	FeatProf (%)
Glass	214	10	5	0.50	1.3e-3	10–40–50–0
Wine	178	13	7	0.54	1.2e-3	0–46–54–0
Letter	20 000	16	8	0.50	9.4e-4	0–100–0–0
SML2010	2 763	17	9	0.53	5.4e-4	0–12–18–71
FrogMFCC	7 195	22	11	0.50	1.1e-3	0–0–5–95
SteelPlates	1 941	27	11	0.41	4.3e-3	11–11–56–22
BreastCancerW	569	30	14	0.47	6.9e-3	0–0–100–0
Ionosphere	351	33	17	0.52	1.9e-3	3–0–97–0
SatImage	6 435	36	18	0.50	4.3e-4	0–75–25–0
SuperCond	21 263	82	37	0.45	1.1e-2	2–1–12–84
COIL2000	5 822	85	35	0.41	2.8e-2	99–1–0–0

Table 3
Results of the identification of the intrinsic dimension for large-dimension datasets. The intrinsic dimensions were identified with the reduction rates varying between 0.39–0.55. The HumActRec dataset with a continuous feature profile has the best reduction rate. The residual errors are between 8.0e-2–2.9e-3.

Dataset	*N*	*n*	ID	Red	MRSE	FeatProf (%)
USPS	9 298	256	130	0.51	2.9e-3	0–0–0–100
BlogPosts	52 397	277	130	0.47	3.9e-3	79–8–12–1
CTSlices	53 500	379	180	0.47	6.3e-2	8–4–10–78
UJIIndoor	19 937	473	200	0.42	7.0e-2	26–73–1–0
Madelon	4 400	500	250	0.50	7.9e-2	2–31–67–0
HumActRec	7 351	561	220	0.39	8.0e-2	0–2–0–98
Isolet	7 797	617	310	0.50	9.4e-3	1–6–14–80
MNIST	60 000	717	350	0.49	9.3e-3	9–14–77–0
FashMNIST	60 000	784	380	0.48	5.0e-2	0–2–98–0
COIL100	7 200	1 024	560	0.55	6.4e-3	0–11–89–0

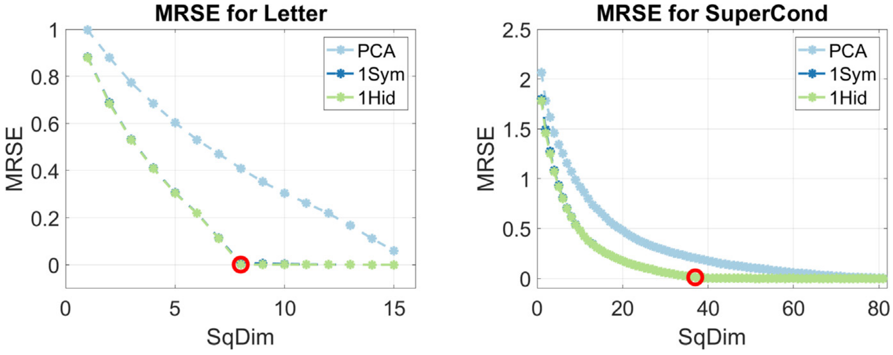


Fig. 4. Identification of the intrinsic dimension for the Letter dataset and SuperCond dataset. **Left**: Clearly identified knee-point in ID = 8 for Letter. **Right**: Gradual decrease of MRSE with ID = 37 for SuperCond.

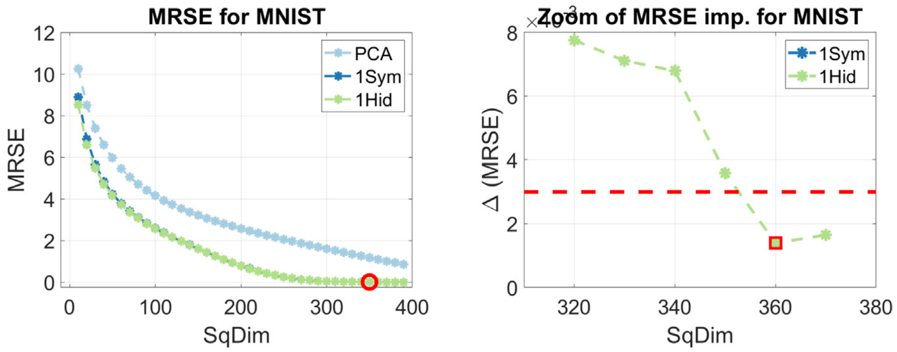


Fig. 5. Identification of the intrinsic dimension for the MNIST dataset. The hidden dimension (plus one) on the left is captured by the sufficiently small error improvement on the right. **Left**: Gradual decrease of MRSE with ID = 350 for MNIST. **Right**: Zoom of MRSE improvement with the threshold τ = 3e-3 confirms the detection.

In Tables 2 and 3, we present the name of the dataset, the number of observations N, the number of features n, and the detected intrinsic dimension ID. The autoencoding error trajectories and thresholdings are illustrated for all datasets in the online SM. The detection threshold for small-dimension datasets was fixed to τ = 4e-3. The reduction rate ID/n for the intrinsic data dimension is

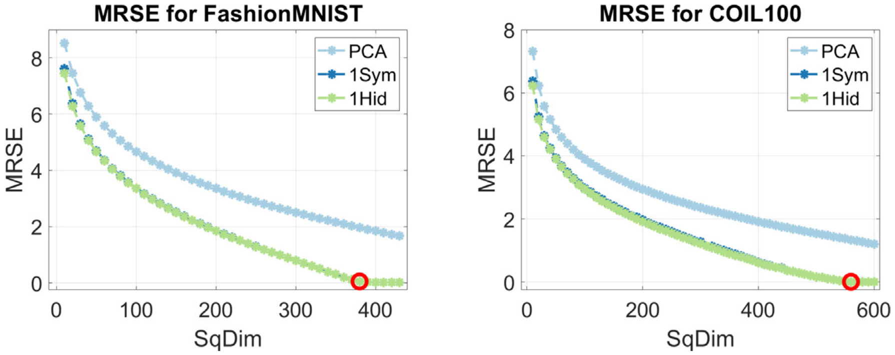


Fig. 6. Identification of the intrinsic dimension for the FashMNIST dataset and COIL100 dataset. **Left**: Clear knee-point of MRSE with ID = 380 for FashMNIST. **Right**: More gradual decrease of MRSE with ID = 560 for COIL100.

reported in the Red column, and the autoencoding error of 1Hid for ID according to (9) is included in the MRSE column. There is no averaging over the data dimension n in (9), so that for higher-dimension datasets this error is expected to remain larger. This was probably one of the reasons why, for large-dimension datasets, we needed to use two values of the threshold τ (based on visual inspection; see the zoomed illustrations in the SM): 3e-3 for USPS, BlogPosts, HumActRec, MNIST, and COIL100, and 3e-2 for the remaining five datasets.

For the analysis, we also included a depiction of how discrete or continuous the set of features for a dataset is. We categorized the features into four groups based on the number of unique values (UV) each feature has: C1 = {UV $\leqslant$10}, C2 = {10 $<$ UV $\leqslant$100}, C3 = {100 $<$ UV $\leqslant$1000}, and C4 = {1000 $<$ UV}. The FeatProf column in Tables 2 and 3 presents the proportions of C1–C4 in percentages.

4.1.1. Conclusions

Examples of the ID detection are given in Figs. 3 (Glass), 4 (Letter on the left, SuperCond on the right), 5 (MNIST), and 6 (FashMNIST on the left, COIL100 on the right). Identifications in the first two cases and for FashMNIST are characterized by clear knee-points in IDs. For SuperCond, with gradual decrease of the MRSE, determination of ID is based on the mutual threshold value τ = 4e-3 of small-dimension datasets. Also MNIST has such a behavior and the zoom in Fig. 5 (right) illustrates the detection decision with τ = 3e-3.

Overall, the intrinsic dimensions were successfully identified for all tested datasets. The use of a feedforward network to approximate the nonlinear residual notably decreased the autoencoding error of the linear PCA. The overall transformation summarizing the essential behavior of data roughly halved the original dimension: The mean reduction rate over the 21 datasets was 0.48.

The reduction rate was independent of the form of the features–that is, the best reduction rates for small-dimension datasets were obtained with the very categorical COIL2000 and primarily continuous SteelPlates datasets. This suggests further experiments on how to treat different types and forms of features, for instance, to address whether different loss functions should be applied [76]. The best reduction rate, 0.39, was anyway obtained for HumActRec, which is characterized by a continuous feature profile. This indicates that we may obtain smaller reduction rates with more continuous sets of features with the proposed approach.

4.2. Comparison of shallow and deep models

The second aim of the experiments was to examine whether deeper network structures and deep learning techniques (the network structure and optimization of the weights) can improve the identification of the intrinsic dimension and the data restoration ability of the additive autoencoder. This aim is pursued as follows: Here, we compare shallow and deep networks in cases where fine-tuning is performed using a classical optimization approach, i.e., using the L-BFGS optimizer with the complete dataset. In the SM, we report the results of using different minibatch-based approaches. Also detailed depictions of the parameter choices and visualization of the results for all datasets are included there.

In addition to visual assessment, we performed a quantitative comparison between the deep and shallow models. First, the MRSE values of all models were divided with the corresponding value of the 1Hid model's error. This was done for the squeezing dimensions from the first until next to last of ID, to cover the essential search phase of the intrinsic dimension. To exemplify relative performance, if the MRSE value of a model divided by the 1Hid's value for a particular squeezing dimension would be 0.5, then such a model would have half the error level and, conversely, twice the efficiency compared to 1Hid. Therefore, the model's efficiency is defined as the reciprocal of relative performance.

The relative performances are illustrated in Figs. 7 and 8. Descriptive statistics of the efficiencies of symmetric models are given in Tables 4 and 5. In each cell there, both the mean efficiency and the maximal efficiency are provided. The latter includes, in parentheses, the squeezing dimension where it was encountered.

4.2.1. Conclusions

During the early phases of searching the intrinsic dimension, deep networks provide smaller autoecoding errors compared to the shallow models. However, as exemplified in Fig. 7 (left) and is evident from all illustrations in the SI, MRSE in ID is not better for deeper models compared to 1Hid. Therefore, the use of a deeper model would not change the ID values and, in fact, fluctuation of the error for the deeper models compared to 1Hid may hinder the detection of a knee-point and negatively affect the simple thresholding.

Usually, the mean efficiency of the two deepest models, 5Sym and 7Sym, is better than that of 1Sym or 3Sym, but for many datasets, there is only a slight improvement. Overall, the plots for the relative efficiencies between different symmetric models and the quantitative trends in the rows of Tables 4 and 5 include varying patterns. The mean efficiency is highest for UJIIndoor, Ionosphere, Madelon, and COIL100, where the last three datasets are characterized by high data dimension/number of observations, n/N, ratio (0.09, 0.11, and 0.08, respectively). The following are the grand means of the mean efficiencies over all 21 datasets for the symmetric models: 1Sym 0.97, 3Sym 1.19, 5Sym 1.38, and 7Sym 1.44. This concludes that deeper models improve the reduction of the autoencoding error during the search of ID. However, the speed of improvement decreases as a function of the number of layers.

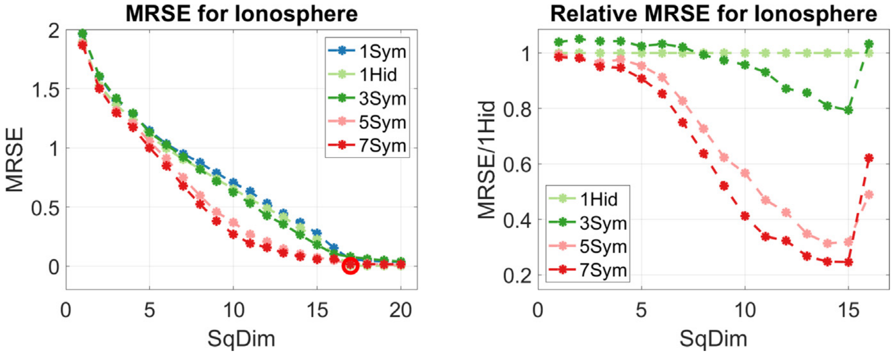


Fig. 7. Left: Behavior of the MRSE with all residual models for Ionosphere. **Right**: Relative performance of the models with respect to 1Hid. During the search of the ID the deeper models show clear improvement but the detected ID is the same for all models.

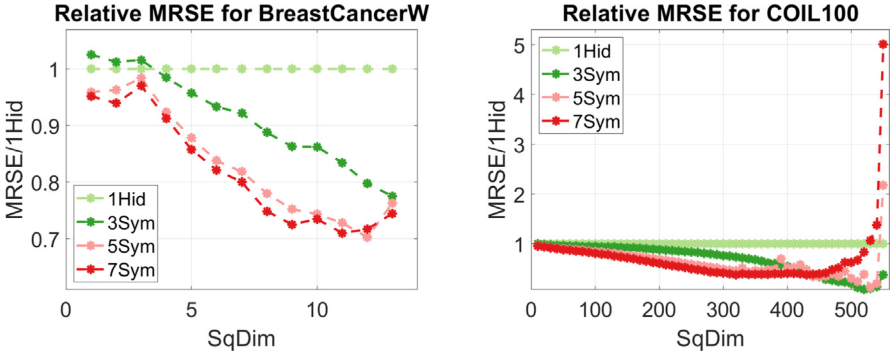


Fig. 8. Left: Relative performance of the models for BreastCancerW. **Right**: Relative performance of the models for COIL100. The deeper models show clear improvement over the shallow ones but the detected ID stays the same and near ID the improved efficiency may be completely lost.

Table 4
Efficiencies of symmetric models for small-dimension datasets.

	1Sym		3Sym		5Sym		7Sym	
Dataset	mean	max (dim)	mean	max (dim)	mean	max (dim)	mean	max (dim)
Glass	0.91	1.03 (2)	0.91	1.10 (3)	1.18	1.31 (3)	1.15	1.40 (3)
Wine	0.97	0.99 (1)	1.07	1.22 (6)	1.25	1.59 (6)	1.36	1.75 (6)
Letter	1.00	1.00 (1)	1.03	1.06 (6)	1.10	1.14 (6)	1.11	1.15 (6)
SML2010	0.94	0.99 (1)	0.95	1.07 (5)	1.12	1.23 (3)	1.15	1.32 (3)
FrogMFCCs	0.99	1.00 (7)	1.04	1.17 (8)	1.10	1.23 (8)	1.11	1.23 (8)
SteelPlates	0.94	0.99 (4)	0.94	1.09 (5)	1.04	1.22 (5)	1.04	1.27 (5)
BreastCancerW	0.98	0.99 (11)	1.10	1.29 (13)	1.22	1.42 (12)	1.24	1.41 (11)
Ionosphere	0.91	0.97 (3)	1.04	1.26 (15)	1.74	3.19 (14)	2.07	4.06 (15)
Satimage	0.99	1.00 (10)	1.02	1.07 (17)	1.05	1.09 (17)	1.06	1.12 (1)
SuperCond	1.00	1.00 (30)	1.10	1.18 (33)	1.20	1.30 (29)	1.23	1.34 (26)
COIL2000	0.99	1.02 (16)	1.24	1.85 (32)	1.49	2.89 (29)	1.48	2.62 (30)

Table 5
Efficiencies of symmetric models for large-dimension datasets.

	1Sym		3Sym		5Sym		7Sym	
Dataset	mean	max (dim)	mean	max (dim)	mean	max (dim)	mean	max (dim)
USPS	0.99	1.00 (90)	1.08	1.16 (90)	1.13	1.21 (70)	1.14	1.23 (60)
BlogPosts	0.95	1.00 (110)	1.18	1.39 (70)	1.28	1.56 (80)	1.23	1.53 (70)
CTSlices	0.99	1.00 (170)	1.17	1.51 (150)	1.30	1.76 (150)	1.32	1.74 (150)
UJIIndoor	0.99	0.99 (60)	1.69	2.46 (150)	2.15	3.11 (130)	2.24	3.51 (130)
Madelon	0.97	1.00 (30)	1.38	3.74 (240)	2.29	5.77 (220)	3.01	8.02 (220)
HumActRec	0.99	1.00 (160)	1.15	1.31 (160)	1.22	1.40 (130)	1.24	1.40 (130)
Isolet	0.99	1.00 (290)	1.22	2.16 (290)	1.44	2.89 (270)	1.55	2.65 (270)
MNIST	0.99	1.00 (200)	1.32	1.99 (260)	1.42	2.20 (250)	1.41	2.25 (240)
FashMNIST	0.99	1.00 (320)	1.15	1.63 (370)	1.22	1.59 (350)	1.23	1.53 (350)
COIL100	0.98	1.00 (310)	2.18	11.19 (520)	1.96	8.51 (530)	1.79	2.63 (430)
COIL100-Min	0.98	1.00 (310)	1.75	8.05 (510)	1.79	4.22 (510)	1.87	2.63 (430)

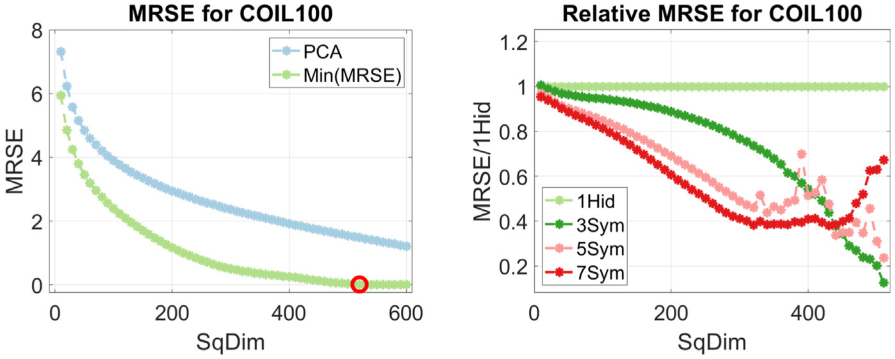


Fig. 9. Left: Minimum autoencoding error of the all models for COIL100. **Right**: Reduced relative performance of the models for COIL100. For COIL100, use of minimum autoencoding error to identify ID yielded more reasonable results.

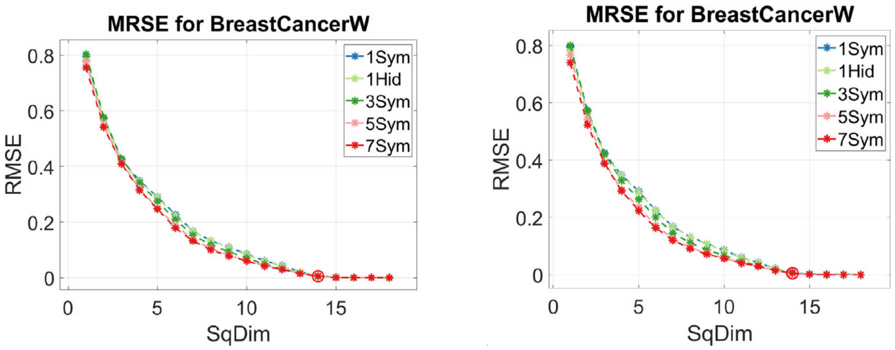


Fig. 10. Left: RMSEs for BreastCancerW with the original 2–3–4 pattern for the hidden layers. **Right**: RMSEs for BreastCancerW with 3–5–7 pattern for the hidden layers. Slighly smaller errors were encountered during the early search phase of the larger model on the right but the detected ID and the overall behavior remained the same.

Actually, close to the intrinsic dimension, the benefits of deeper models may be completely lost. This is illustrated in Fig. 8 (right) and in Table 5 for COIL100 (see also, for example, plots of BlogPosts and MNIST in the SI). The reason for such a behavior with COIL100 is the value of ID, 560, which was obtained with the smaller threshold τ = 3e-3 for large-dimension datasets. Therefore, with COIL100, we also tested an alternative approach to the identification of ID, where we apply the same thresholding technique (and the same τ) to the minimum autoencoding error of the all models. This error plot, the identified ID = 520 (for which the reduction rate would be 0.51), and the corresponding reduced set of relative efficiencies are illustrated in Fig. 9. The summary of the efficiencies for this modified way to identify ID are given in the last line "COIL100-Min" in Table 5. It can be concluded that for the largest dimensional dataset COIL100, the use of the minimum autoencoding error of the models yielded more reasonable results.

We used a fixed pattern for the sizes of the hidden layers in deeper models: 2–3–4 times the squeezing dimension for 7Sym, the first and last of these coefficients for 5Sym, and the first one for 3Sym. As reported in Section 4.1, the mean reduction rate over the 21 datasets was close to 0.5. Therefore, one may wonder whether this behavior is due to the fact that from this case onwards all the hidden dimensions are larger than the number of features so that a kind of nonlinear kernel trick occurs. In other words, would a different pattern of the hidden dimensions change the results and conclusions here? This consideration was tested by considering a 3–5-7 pattern providing much more flexibility for the nonlinear operator compared to the used pattern. These tests are not reported as a whole, because the clearly identified trend of the results is readily exemplified in Fig. 10: Increase of the sizes of the hidden layers slightly improve the reduction rate during early phase of the search but does not change the value of the ID.

4.3. Comparison of classical and additive autoencoder

As depicted in Section 3, the basic structural difference between the classical and the additive autoencoder is the inclusion of the PCA based linear dimension reduction operator, and transformation back to the original dimension, in the latter model. One recovers the classical autoencoding model from the additive one by simply setting $\mathbf{U} = 0$ in (2). Next, we study how this affects the autoencoding error and the training time. In order to solely compare the two model structures, we performed experiments for all datasets using the 5Sym architecture and exactly the same training (i.e., optimization) settings (see SM) either with $\mathbf{U} = 0$ ('ClasAE') or in the proposed form ('PropAE'). As before, the tests were carried out over an increasing set of dimensions of the squeezing layer n_L, which started from one and were incremented one by one for the small-dimension datasets. For the large-dimension datasets, a data-specific increment of the order 10–40 starting from the squeezing layer's size 10–40 were used, so that altogether 10–15 squeezing layers were tested in all of these cases (exact specifications of the tested dimensions are given in the SM). The intrinsic dimension ID identified and reported in Tables 2 and 3 was used as the size of the last squeezing layer for all datasets.

Results of the comparison are summarized in Table 6 and illustrated, for the two large-dimension dataset, in Fig. 11 (all figures comparing RMSEs are given in the SM). In Table 6, for the compared models, the CPU time is reported in minutes and it includes the whole training time over all tested levels (for the proposed AE, also time to compute $\mathbf{U}$ using PCA is included). For the latter CPU value in the fifth column, the CPU ratio 'rat' of column five to column two is reported in parenthesis. For both models, the column 'RMSE' provides the final reconstruction error and the column '$\mathcal{J}_0$' includes the first value of the cost function (4) when training

Table 6
Comparison of classical and additive autoencoder.

	Classic AE			Proposed AE		
Data	CPU	$\mathcal{J}_0$	RMSE	CPU (rat)	$\mathcal{J}_0$	RMSE
Glass	4.6e-1	1.1e-1	1.62e-1	5.0e-1 (1.09)	9.0e-4	6.69e-3
Wine	7.1e-1	1.5e-1	2.77e-1	6.8e-1 (0.96)	2.7e-4	1.13e-2
Letter	9.7e0	1.1e-1	3.33e-1	9.7e0 (1.0)	2.5e-4	1.39e-2
SML2010	2.3e0	1.2e-1	1.67e-1	2.1e0 (0.91)	2.1e-4	9.65e-3
FrogMFCC	5.1e0	3.5e-2	1.70e-1	5.0e+0 (0.98)	1.3e-4	6.71e-3
SteelPlates	2.9e0	1.8e-1	2.25e-1	3.0e0 (1.03)	1.4e-3	8.78e-3
BreastCancerW	2.7e+0	7.8e-2	1.65e-1	2.7e+0 (1.0)	6.0e-5	5.71e-3
Ionosphere	2.9e+0	9.2e-1	2.98e-1	2.9e+0 (1.0)	1.5e-2	1.94e-2
SatImage	1.7e+1	1.1e-1	2.25e-1	1.7e+1 (1.0)	7.6e-7	7.21e-4
SuperCond	2.0e+2	1.6e-1	1.85e-1	1.9e+2 (0.95)	1.0e-4	9.92e-3
COIL2000	5.6e+1	3.2e-1	4.47e-1	5.7e+1 (1.02)	4.8e-3	1.90e-2
USPS	1.7e+2	1,4e0	6.27e-1	1.6e+2 (0.94)	9.6e-6	2.01e-4
BlogPosts	8.8e+2	9.0e-1	2.99e-1	8.0e+2 (0.91)	1.4e-4	4.82e-3
CTSlices	1.3e+3	6.8e0	1.51e+0	1.0e+3 (0.77)	7.5e-3	7.42e-2
UJIIndoor	5.7e+2	1.5e0	5.81e-1	5.2e+2 (0.91)	8.1e-3	7.50e-2
Madelon	1.8e+2	6.3e0	1.06e+0	1.7e+2 (0.94)	1.2e-2	3.88e-2
HumActRec	3.2e+2	7.2e-1	4.50e-1	2.9e+2 (0.91)	4.0e-5	5.58e-3
Isolet	7.0e+2	2.9e0	1.59e+0	5.7e+2 (0.81)	6.2e-4	8.34e-3
MNIST	3.1e+3	4.2e0	1.17e+0	2.4e+3 (0.77)	1.1e-3	1.27e-2
FashMNIST	3.7e+3	8.2e0	2.30e+0	2.9e+3 (0.78)	1.0e-2	5.35e-2
COIL100	1.4e+3	2.8e0	1.43e+0	1.1e+3 (0.79)	3.8e-3	9.06e-3

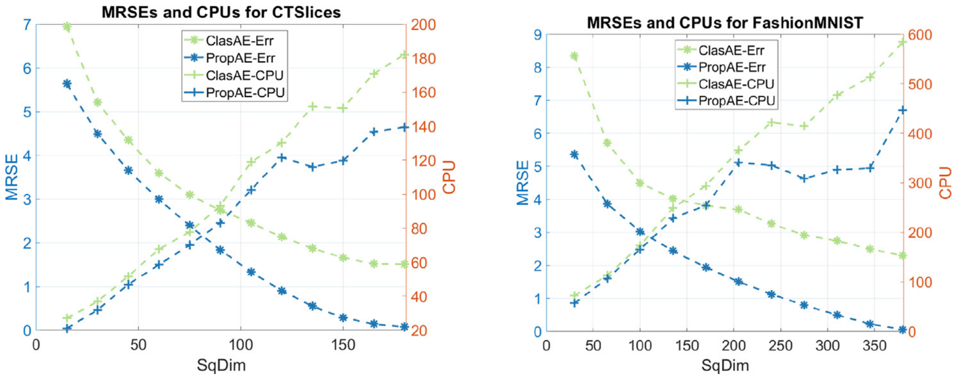


Fig. 11. 5Sym for CTSlices (left) and FashMNIST (righ): behavior of autoencoding error for the classical and additive autoencoder (left y-axis) and the training time (right y-axis) over a set of squeezing layer dimensions. Both reconstruction errors and CPU times are strictly smaller for the additive autoencoder compared to the classical one.

the models in the last tested dimension ID. The latter allows one to compare the effect of the layerwise pretraining through stacking as depicted in Section 3.2.

This comparison summarizes the additional benefits, in addition to the possibility to identify the intrinsic dimension, of having the explicit linear operator in the autoencoder. The autoencoding error is always smaller, for all tested dimensions, for the proposed, additive model. It reaches much smaller reconstruction error in ID and, for the large-dimension datasets with more computational efforts needed, decreases the training time (the larger the overall CPU time the larger the benefit as shown, e.g., by 'rat' in the last three datasets). As exemplified in Fig. 11 (right y-axis) and confirmed by comparing the columns three and six of $\mathcal{J}_0$s in Table 6, both the better quality and the reduced CPU of the additive AE are especially due to improvements in the quality of stacking during the shallow pretraining phase. This underlines the usefulness of the original idea to construct a serial approximation of data in the reduced dimension using an explicit separation of linear and nonlinear operators acting both on the original dimension, as motivated in the beginning of Section 3.1.

Finally, in Section 4.2 of the SM, results from a second comparison that studied further the effect of the autoencoding model's structure and assessed our reference implementation are given. More precisely, with the nonsymmetric one-hidden-layer 1Hid model and utilizing Matlab's own 'trainAutoencoder'-method that uses a sparsity regularizer based on the Kullback–Leibler divergence (cf. Table 1), we tested three different autoencoding pipelines: 1) Apply Matlab's own AE routine in the reduced dimension after PCA and compute the reconstruction error with the inverse PCA; 2) Apply Matlab's own AE according to our proposition to data obtained after the linear trend estimation in the original data dimension; 3) Use our own implementation of the suggested method. These tests were concluded as follows: In cases 2) and 3), both Matlab's AE and our AE worked similarly and retrieved the intrinsic dimension. This indicates that different autoencoding models, as depicted in Section 3.1, could be used for the nonlinear residual estimation in the additive model. As expected, large reconstruction errors without recovery of the intrinsic dimension were obtained if the nonlinear autoencoding model was operating directly on the reduced dimension after the linear transformation (case 1). Our implementation (which can be applied with any number and size of layers differently from the Matlab routine) was typically many times faster than the Matlab's AE and completely stable, which was not the case with the proprietary routine.

4.4. Generalization of the autoencoder

In the last experiments, we demonstrate and evaluate the generalization of the additive 5Sym autoencoder. Search over squeezing dimensions is performed in a similar manner as that done in

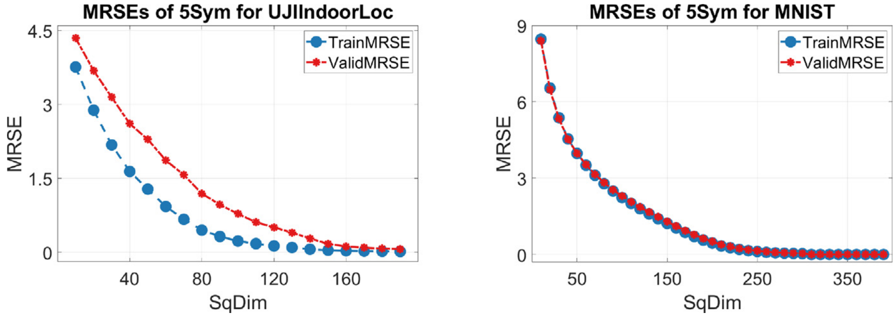


Fig. 12. Agreements of training and validation set MRSE values for UJIIndoor (left) and MNIST (right). Large deviation between training-validation errors on the left but perfect match on the right. In ID, similar autoencoding error level is reached with both datasets.

Section 4.2. We apply a small sample of datasets, for which a separate validation data was given in the UCI repository. More precisely, we use Letter (size of training data $N = 16000$, size of validation data $N_v = 4000$, i.e., 80%–20% portions with respect to the entire data; number of nonconstant featureas $n = 16$), UJIIndoor ($N = 19937,\ N_v = 1111$, 95%–5% portions; $n = 473$), HumActRecog ($N = 7351,\ N_v = 2946$, 71%–29% portions; $n = 561$), and MNIST ($N = 60000,\ N_v = 10000$, 86%–14% portions; $n = 666$). Note that because all data are used as is, we have no information or guarantees on how well the data distributions in the training and validation sets actually match each other.

As anticipated, both training-validation portions and the data dimension affected the generalization results. For Letter, with 80%-20% division between training-validation sizes and small number of features, we witnessed a perfect match between the training and validation MRSE values. The same held true for MNIST, which is illustrated in Fig. 12 (right). The largest discrepancy between the training and validation errors, depicted in Fig. 12 (left), was obtained for UJIIndoor, which has the most deviating 95%-5% portions with almost 500 features. This dataset also had one of the largest efficiencies (i.e., reduction potential) in Table 5. HumActRec was somewhere in the middle in its behavior, with clearly visible deviation. Because of the data portions (∼70%-30%), the difference raises doubts regarding the quality of the validation set. Note that these considerations provide examples of the possibilities of autoencoders to assess the quality of data.

The visual inspection was augmented by computing the correlation coefficient between the MRSE values in the training and validation sets. The following values confirmed the conclusions of the visual inspection: Letter 1.0000, UJIIndoor 0.9766, HumActRecog 0.9939, and MNIST 0.9999. Finally, an important observation from Fig. 12 is that when the squeezing dimension is increased up to the intrinsic dimension, then the validation error tends to the same error level than the training error. Therefore, the additive autoencoder determined using the training data was always able to explain the variability of the validation data with a compatible accuracy.

5. Conclusions

This study illustrated a case where all main concerns with feedforward mappings, as quoted in Section 2.2 from [95, p. 363], were solved: learning was successful, the size and the number of hidden layers were identified, and the deterministic relationship within a dataset was found. Similar to [23], stacking was found to be an essential building block for estimating the weights of deep autoencoders. Learning and dimension estimation were based on a simply weighted, automatically scalable cost function with compact layerwise weight calculus and a straighforward heuristic for determining the intrinsic data dimension. Intrinsic dimensions for all tested datasets with a low autoencoding error were revealed. A similar autoencoding error, and the corresponding intrinsic dimension, was obtained independently on the depth of the network. This was not obtained with the classical autoencoding model, without the linear operator, or if the residual after the linear dimension reduction was processed further in the reduced dimensional space. However, the experiments clearly indicated that other autoencoding techniques could be used for the nonlinear residual estimation in the additive form.

One clear advantage of the proposed methodology is the lack of meta-level parameters (e.g., number and form of layers, selection of activation function, detection of the learning rate) that are usually tuned or grid-searched when DNNs are applied. The only parameter that may need adjustment based on visual assessment is τ– that is, the threshold for identifying the hidden dimension. Moreover, because of the observed smoothly decreasing behavior of the autoencoding error, the intrinsic dimension could be searched for more efficiently than just incrementally: One could attempt to utilize one-dimensional optimization techniques like a golden-section search and/or polynomial and spline interpolation to more quickly identify the beginning of the error plateau.

These results challenge the common beliefs and currently popular traditions with deep learning techniques. The experiments summarized here and given in the SM suggest that many existing deep learning results could be improved by using a clear separation of linear and nonlinear data-driven modelling. Also use of more accurate optimization techniques to determine the weights of such models may be advantageous.

We can use the additive transformation to the intrinsic dimension as a pretrained part for transfer learning with any prediction or classification model [105]. It would be interesting to test in the future whether one should use this as is or would a transformation into a smaller squeezing dimension than the intrinsic one generalize better in prediction and classification tasks? Another detectable dimension of the squeezing layer worth investigating, as illustrated in the relative MRSE plots (see also the SM) and in Tables 4 and 5, could be the one with the largest nonlinear gain– that is, with the maximum difference between the PCA error and the autoencoder error or between the shallow and deep results. Moreover, we used global techniques in every part of the autoencoder. The technique might benefit from encoding locally estimated behavior– for example, using convolutional layers for local-level estimation [112]. Similarly, other linear transformation techniques and modifications of PCA might provide better performance [2,113,114], although in the proposed form, we also need the inverse of the linear mapping to be able to estimate the residual error in the original vector space.

Data availability

Data and codes are available in public repositories.

Declaration of Competing Interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Tommi Kärkkäinen reports financial support from Academy of Finland. Jan Hänninen reports financial support from Jenny and Antti Wihuri Foundation.

Acknowledgments

This work was supported by the Academy of Finland from the project 351579 (MLNovCat).

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, athttps://doi.org/10.1016/j.neucom.2023.126520.

References

[1] M.A. Carreira-Perpinán, A review of dimension reduction techniques, Department of Computer Science. University of Sheffield. Tech. Rep. CS-96-09 9 (1997) 1–69.

[2] C.J. Burges et al., Dimension reduction: A guided tour, Foundations and Trends, Machine Learning 2 (4) (2010) 275–365.

[3] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Networks 61 (2015) 85–117.

[4] T. Kärkkäinen, J. Rasku, Application of a knowledge discovery process to study instances of capacitated vehicle routing problems, in: Computation and Big Data for Transport, Chapter 6, Computational Methods in Applied Sciences, Springer-Verlag, 2020, pp. 1–25.

[5] T. Kärkkäinen, On the role of Taylor's formula in machine learning, in: Impact of scientific computing on science and society, Springer Nature, 2022, (18 pages, to appear).

[6] K. Fukunaga, D.R. Olsen, An algorithm for finding intrinsic dimensionality of data, IEEE Transactions on Computers 100 (2) (1971) 176–183.

[7] F. Camastra, Data dimensionality estimation methods: a survey, Pattern recognition 36 (12) (2003) 2945–2954.

[8] J.A. Lee, M. Verleysen, Nonlinear dimensionality reduction, Springer Science & Business Media, 2007.

[9] K. Fukunaga, Intrinsic dimensionality extraction, Handbook of statistics 2 (1982) 347–360.

[10] I. Jolliffe, Principal Component Analysis, 2nd Edition., Springer Verlag, 2002.

[11] E. Facco, M. d'Errico, A. Rodriguez, A. Laio, Estimating the intrinsic dimension of datasets by a minimal neighborhood information, Scientific reports 7 (1) (2017) 1–8.

[12] F. Camastra, A. Staiano, Intrinsic dimension estimation: Advances and open problems, Information Sciences 328 (2016) 26–41.

[13] G. Navarro, R. Paredes, N. Reyes, C. Bustos, An empirical evaluation of intrinsic dimension estimators, Information Systems 64 (2017) 206–218.

[14] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, The kdd process for extracting useful knowledge from volumes of data, Communications of the ACM 39 (11) (1996) 27–34.

[15] A. Rotondo, F. Quilligan, Evolution paths for knowledge discovery and data mining process models, SN Computer Science 1 (2) (2020) 1–19.

[16] Y. Wang, H. Yao, S. Zhao, Auto-encoder based dimensionality reduction, Neurocomputing 184 (2016) 232–242.

[17] N. Bahadur, R. Paffenroth, Dimension estimation using autoencoders with applications to financial market analysis, in: 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2020, pp. 527–534.

[18] N. Bahadur, R. Paffenroth, Dimension estimation using autoencoders, arXiv preprint arXiv:1909.10702.

[19] G.W. Cottrell, Learning internal representations from gray-scale images: An example of extensional programming, in: Proceedings Ninth Annual Conference of the Cognitive Science Society, Irvine, CA, 1985, pp. 462–473.

[20] H. Bourlard, Y. Kamp, Auto-association by multilayer perceptrons and singular value decomposition, Biological cybernetics 59 (4) (1988) 291–294.

[21] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[22] M.I. Jordan, T.M. Mitchell, Machine learning: Trends, perspectives, and prospects, Science 349 (6245) (2015) 255–260.

[23] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.

[24] T. Kärkkäinen, MLP in layer-wise form with applications to weight decay, Neural Computation 14 (6) (2002) 1451–1480.

[25] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F.E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing 234 (2017) 11–26.

[26] T.N. Sainath, B. Kingsbury, B. Ramabhadran, Auto-encoder bottleneck features using deep belief networks, in: 2012 IEEE international conference on acoustics, speech and signal processing (ICASSP), IEEE, 2012, pp. 4153–4156.

[27] M. Khodayar, O. Kaynak, M.E. Khodayar, Rough deep neural architecture for short-term wind speed forecasting, IEEE Transactions on Industrial Informatics 13 (6) (2017) 2770–2779.

[28] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, J. Long, A survey of clustering with deep learning: From the perspective of network architecture, IEEE Access 6 (2018) 39501–39514.

[29] R. McConville, R. Santos-Rodriguez, R.J. Piechocki, I. Craddock, N2d:(not too) deep clustering via clustering the local manifold of an autoencoded embedding, in: 2020 25th International Conference on Pattern Recognition (ICPR), IEEE, 2021, pp. 5145–5152.

[30] B. Zhang, J. Qian, Autoencoder-based unsupervised clustering and hashing, Applied Intelligence 51 (1) (2021) 493–505.

[31] B. Diallo, J. Hu, T. Li, G.A. Khan, X. Liang, Y. Zhao, Deep embedding clustering based on contractive autoencoder, Neurocomputing 433 (2021) 96–107.

[32] C. Ling, G. Cao, W. Cao, H. Wang, H. Ren, Iae-clustergan: A new inverse autoencoder for generative adversarial attention clustering network, Neurocomputing 465 (2021) 406–416.

[33] D. Charte, F. Charte, M.J. del Jesus, F. Herrera, An analysis on the use of autoencoders for representation learning: Fundamentals, learning task case studies, explainability and challenges, Neurocomputing 404 (2020) 93–107.

[34] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, L. Bottou, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion., Journal of machine learning research 11 (12).

[35] K. Ho, C.-S. Leung, J. Sum, Objective functions of online weight noise injection training algorithms for MLPs, IEEE transactions on neural networks 22 (2) (2010) 317–323.

[36] M. Chen, K.Q. Weinberger, Z. Xu, F. Sha, Marginalizing stacked linear denoising autoencoders, The Journal of Machine Learning Research 16 (1) (2015) 3849–3875.

[37] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Deep learning for time series classification: a review, Data mining and knowledge discovery 33 (4) (2019) 917–963.

[38] Q. Ma, W.-C. Lee, T.-Y. Fu, Y. Gu, G. Yu, Midia: exploring denoising autoencoders for missing data imputation, Data Mining and Knowledge Discovery 34 (6) (2020) 1859–1897.

[39] M. Probst, F. Rothlauf, Harmless overfitting: Using denoising autoencoders in estimation of distribution algorithms, Journal of Machine Learning Research 21 (78) (2020) 1–31.

[40] P. Filzmoser, M. Gschwandtner, V. Todorov, Review of sparse methods in regression and classification with application to chemometrics, Journal of Chemometrics 26 (3–4) (2012) 42–51.

[41] M. Sun, X. Zhang, T.F. Zheng, et al., Unseen noise estimation using separable deep auto encoder for speech enhancement, IEEE/ACM Transactions on Audio, Speech, and Language Processing 24 (1) (2015) 93–104.

[42] M. Haddad, M. Bouguessa, Exploring the representational power of graph autoencoder, Neurocomputing 457 (2021) 225–241.

[43] M. Ma, S. Na, H. Wang, Aegcn: An autoencoder-constrained graph convolutional network, Neurocomputing 432 (2021) 21–31.

[44] C. Qiao, X.-Y. Hu, L. Xiao, V.D. Calhoun, Y.-P. Wang, A deep autoencoder with sparse and graph laplacian regularization for characterizing dynamic functional connectivity during brain development, Neurocomputing 456 (2021) 97–108.

[45] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE Transactions on Neural Networks and Learning Systems 32 (1) (2021) 4–24.

[46] Z. Hou, X. Liu, Y. Dong, C. Wang, J. Tang, et al., Graphmae: Self-supervised masked graph autoencoders, in: Proceedings o the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 594–604.

[47] Y. Pan, J. Zou, J. Qiu, S. Wang, G. Hu, Z. Pan, Joint network embedding of network structure and node attributes via deep autoencoder, Neurocomputing 468 (2022) 198–210.

[48] J. Yoo, H. Jeon, J. Jung, U. Kang, Accurate node feature estimation with structured variational graph autoencoder, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 2336–2346.

[49] B. Dai, Y. Wang, J. Aston, G. Hua, D. Wipf, Connections with robust PCA and the role of emergent sparsity in variational autoencoder models, The Journal of Machine Learning Research 19 (1) (2018) 1573–1614.

[50] S. Burkhardt, S. Kramer, Decoupling sparsity and smoothness in the dirichlet variational autoencoder topic model, Journal of Machine Learning Research 20 (131) (2019) 1–27.

[51] Y. Zhao, K. Hao, X.-S. Tang, L. Chen, B. Wei, A conditional variational autoencoder based self-transferred algorithm for imbalanced classification, Knowledge-Based Systems 218 (106756) (2021) 1–10.

[52] H. Takahashi, T. Iwata, A. Kumagai, S. Kanai, M. Yamada, Y. Yamanaka, H. Kashima, Learning optimal priors for task-invariant representations in variational autoencoders, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 1739–1748.

[53] G. Alain, Y. Bengio, What regularized auto-encoders learn from the data-generating distribution, The Journal of Machine Learning Research 15 (1) (2014) 3563–3593.

[54] N. Janakarajan, J. Born, M. Manica, A fully differentiable set autoencoder, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 3061–3071.
[55] J. Deng, S. Frühholz, Z. Zhang, B. Schuller, Recognizing emotions from whispered speech based on acoustic feature transfer learning, IEEE Access 5 (2017) 5235–5246.
[56] C. Sun, M. Ma, Z. Zhao, S. Tian, R. Yan, X. Chen, Deep transfer learning based on sparse autoencoder for remaining useful life prediction of tool in manufacturing, IEEE transactions on industrial informatics 15 (4) (2018) 2416–2425.
[57] M. Sun, H. Wang, P. Liu, S. Huang, P. Fan, A sparse stacked denoising autoencoder with optimized transfer learning applied to the fault diagnosis of rolling bearings, Measurement 146 (2019) 305–314.
[58] M.A. Chao, B.T. Adey, O. Fink, Implicit supervision for fault detection and segmentation of emerging fault types with deep variational autoencoders, Neurocomputing 454 (2021) 324–338.
[59] N. Amini, Q. Zhu, Fault detection and diagnosis with a novel source-aware autoencoder and deep residual neural network, Neurocomputing 488 (2022) 618–633.
[60] S. Kim, Y.-K. Noh, F.C. Park, Efficient neural network compression via transfer learning for machine vision inspection, Neurocomputing 413 (2020) 294–304.
[61] W. Sheng, X. Li, Siamese denoising autoencoders for joints trajectories reconstruction and robust gait recognition, Neurocomputing 395 (2020) 86–94.
[62] Z. Cao, X. Li, Y. Feng, S. Chen, C. Xia, L. Zhao, Contrastnet: Unsupervised feature learning by autoencoder and prototypical contrastive learning for hyperspectral imagery classification, Neurocomputing 460 (2021) 71–83.
[63] G. Lin, C. Fan, W. Chen, Y. Chen, F. Zhao, Class label autoencoder with structure refinement for zero-shot learning, Neurocomputing 428 (2021) 54–64.
[64] J. Song, G. Shi, X. Xie, Q. Wu, M. Zhang, Domain-aware stacked autoencoders for zero-shot learning, Neurocomputing 429 (2021) 118–131.
[65] D. Sun, W. Xie, Z. Ding, J. Tang, Silp-autoencoder for face de-occlusion, Neurocomputing 485 (2022) 47–56.
[66] X. Yue, J. Li, J. Wu, J. Chang, J. Wan, J. Ma, Multi-task adversarial autoencoder network for face alignment in the wild, Neurocomputing 437 (2021) 261–273.
[67] W. Yin, L. Li, F.-X. Wu, A semi-supervised autoencoder for autism disease diagnosis, Neurocomputing 483 (2022) 140–147.
[68] Q. Zhou, B. Li, P. Tao, Z. Xu, C. Zhou, Y. Wu, H. Hu, Residual-recursive autoencoder for accelerated evolution in savonius wind turbines optimization, Neurocomputing 500 (2022) 909–920.
[69] A. Khajenezhad, H. Madani, H. Beigy, Masked autoencoder for distribution estimation on small structured data sets, IEEE Transactions on Neural Networks and Learning SystemsEarly Access, to appear.
[70] Y. Ikeda, K. Tajiri, Y. Nakano, K. Watanabe, K. Ishibashi, Estimation of dimensions contributing to detected anomalies with variational autoencoders, arXiv preprint arXiv:1811.04576.
[71] Y. Gao, B. Shi, B. Dong, Y. Chen, L. Mi, Z. Huang, Y. Shi, RVAE-ABFA: robust anomaly detection for highdimensional data using variational autoencoder, in: 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), IEEE, 2020, pp. 334–339.
[72] Y. Liu, Y. Lin, Q. Xiao, G. Hu, J. Wang, Self-adversarial variational autoencoder with spectral residual for time series anomaly detection, Neurocomputing 458 (2021) 349–363.
[73] Q. Yu, M. Kavitha, T. Kurita, Autoencoder framework based on orthogonal projection constraints improves anomalies detection, Neurocomputing 450 (2021) 372–388.
[74] N. Li, F. Chang, C. Liu, Human-related anomalous event detection via spatial-temporal graph convolutional autoencoder with embedded long short-term memory network, Neurocomputing 490 (2022) 482–494.
[75] S. Narayanan, R. Marks, J.L. Vian, J. Choi, M. El-Sharkawi, B.B. Thompson, Set constraint discovery: missing sensor data restoration using autoassociative regression machines, in: Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02, Vol. 3, IEEE, 2002, pp. 2872–2877.
[76] N. Abiri, B. Linse, P. Edén, M. Ohlsson, Establishing strong imputation performance of a denoising autoencoder in a wide range of missing data problems, Neurocomputing 365 (2019) 137–146.
[77] X. Lai, X. Wu, L. Zhang, W. Lu, C. Zhong, Imputations of missing values using a tracking-removed autoencoder trained with incomplete data, Neurocomputing 366 (2019) 54–65.
[78] Y. Zhou, Z. Ding, X. Liu, C. Shen, L. Tong, X. Guan, Infer-avae: An attribute inference model based on adversarial variational autoencoder, Neurocomputing 483 (2022) 105–115.
[79] L. Tran, X. Liu, J. Zhou, R. Jin, Missing modalities imputation via cascaded residual autoencoder, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1405–1414.
[80] J. Zhao, Y. Nie, S. Ni, X. Sun, Traffic data imputation and prediction: An efficient realization of deep learning, IEEE Access 8 (2020) 46713–46722.
[81] L. Li, M. Franklin, M. Girguis, F. Lurmann, J. Wu, N. Pavlovic, C. Breton, F. Gilliland, R. Habre, Spatiotemporal imputation of MAIAC AOD using deep learning with downscaling, Remote sensing of environment 237 (2020).
[82] M. Sangeetha, M.S. Kumaran, Deep learning-based data imputation on time-variant data using recurrent neural network, Soft Computing 24 (17) (2020) 13369–13380.
[83] S. Ryu, M. Kim, H. Kim, Denoising autoencoder-based missing value imputation for smart meters, IEEE Access 8 (2020) 40656–40666.
[84] A. Ahmed, K. Saleem, O. Khalid, J. Gao, U. Rashid, Trust-aware denoising autoencoder with spatial-temporal activity for cross-domain personalized recommendations, Neurocomputing 511 (2022) 477–494.
[85] P. Nousi, S.-C. Fragkouli, N. Passalis, P. Iosif, T. Apostolatos, G. Pappas, N. Stergioulas, A. Tefas, Autoencoder-driven spiral representation learning for gravitational wave surrogate modelling, Neurocomputing 491 (2022) 67–77.
[86] L. Alzubaidi, J. Zhang, A.J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M.A. Fadhel, M. Al-Amidie, L. Farhan, Review of deep learning: concepts, CNN architectures, challenges, applications, future directions, Journal of big Data 8 (1) (2021) 1–74.
[87] V. Carletti, A. Greco, G. Percannella, M. Vento, Age from faces in the deep learning revolution, IEEE transactions on pattern analysis and machine intelligence 42 (9) (2020) 2113–2132.
[88] S. Chen, Q. Zhao, Shallowing deep networks: Layer-wise pruning based on feature representations, IEEE transactions on pattern analysis and machine intelligence 41 (12) (2018) 3048–3056.
[89] C. Guo, G. Pleiss, Y. Sun, K.Q. Weinberger, On calibration of modern neural networks, arXiv preprint arXiv:1706.04599 (2017).
[90] S. Lathuilière, P. Mesejo, X. Alameda-Pineda, R. Horaud, A comprehensive analysis of deep regression, IEEE transactions on pattern analysis and machine intelligence 42 (9) (2020) 2065–2081.
[91] T. Elsken, J.H. Metzen, F. Hutter, Neural architecture search: A survey, The, Journal of Machine Learning Research 20 (1) (2019) 1997–2017.
[92] T.J. Sejnowski, The unreasonable effectiveness of deep learning in artificial intelligence, Proceedings of the National Academy of SciencesWww.pnas.org/cgi/doi/10.1073/pnas.1907373117.
[93] S. Yu, J.C. Principe, Understanding autoencoders with information theoretic concepts, Neural Networks 117 (2019) 104–123.
[94] A. Pinkus, Approximation theory of the mlp model in neural networks, Acta Numerica 8 (1999) 143–195.
[95] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359–366.
[96] J.E. Dennis Jr., R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Vol. 16, Siam, 1996.
[97] J. Nocedal, S. Wright, Numerical Optimization, Springer Science & Business Media, 2006.
[98] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[99] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proceedings of International Conference on Learning Representations, 2015, arXiv: 1412.6980.
[100] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, T. Goldstein, Training neural networks without gradients: A scalable ADMM approach, in: International Conference on Machine Learning (ICML), PMLR, 2016, pp. 2722–2731.
[101] T. Kärkkäinen, E. Heikkola, Robust formulations for training multilayer perceptrons, Neural Computation 16 (4) (2004) 837–862.
[102] N. Bellomo, L. Preziosi, Modelling mathematical methods and scientific computation, Vol. 1, CRC Press, 1994.
[103] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.
[104] T. Kärkkäinen, M. Saarela, Robust principal component analysis of data with missing values, in: International Workshop on Machine Learning and Data Mining in Pattern Recognition, Springer, 2015, pp. 140–154.
[105] A. Ghods, D.J. Cook, A survey of deep network techniques all classifiers can adopt, Data mining and knowledge discovery 35 (1) (2021) 46–87.
[106] H. Gouk, E. Frank, B. Pfahringer, M.J. Cree, Regularisation of neural networks by enforcing lipschitz continuity, Machine Learning 110 (2) (2021) 393–416.
[107] Y.O. Bengio, Learning deep architectures for ai, Foundations and trends, Machine Learning 2 (1) (2009) 1–127.
[108] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[109] R.L. Thorndike, Who belongs in the family?, Psychometrika 18 (4) (1953) 267–276
[110] T. Kärkkäinen, On cross-validation for MLP model evaluation, in: Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), Springer, 2014, pp. 291–300.
[111] D. Dua, C. Graff, UCI Machine Learning Repository (2017). URL: http://archive.ics.uci.edu/ml.
[112] Y. LeCun, B.E. Boser, J.S. Denker, D. Henderson, R.E. Howard, W.E. Hubbard, L. D. Jackel, Handwritten digit recognition with a back-propagation network, in: Advances in Neural Information Processing Systems, 1990, pp. 396–404.
[113] L. Song, H. Ma, M. Wu, Z. Zhou, M. Fu, A brief survey of dimension reduction, in: International Conference on Intelligent Science and Big Data Engineering, Springer, 2018, pp. 189–200.
[114] J.T. Vogelstein, E.W. Bridgeford, M. Tang, D. Zheng, C. Douville, R. Burns, M. Maggioni, Supervised dimensionality reduction for big data, Nature communications 12 (1) (2021) 1–9.

Tommi Kärkkäinen (TK) received the Ph.D. degree in Mathematical Information Technology from the University of Jyväskylä (JYU), in 1995. Since 2002 he has been serving as a full professor of Mathematical Information Technology at the Faculty of Information Technology (FIT), JYU. TK has led 50 different R&D projects and has been supervising 60 PhD students. He has published over 200 peer-reviewed articles. TK received the Innovation Prize of JYU in 2010. He has served in many administrative positions at FIT and JYU, leading currently a Research Division and a Research Group on Human and Machine based Intelligence in Learning. The main research interests include data mining, machine learning, learning analytics, and nanotechnology. He is a senior member of the IEEE.

Jan Hänninen received the BSc and MSc degrees in Mathematical Information Technology from the University of Jyväskylä, Department of Mathematical Information Technology. He is working towards the PhD degree in Mathematical Information Technology at the Faculty of Information Technology, University of Jyväskylä. His research interests include neural networks, nonlinear optimization, and computational efficiency.