Rank Reduction Autoencoders - Enhancing interpolation on nonlinear manifolds.

Anonymous Author(s)
Affiliation
Address
email

Abstract

The efficiency of classical Autoencoders (AEs) is limited in many practical situations. When the latent space is reduced through autoencoders, feature extraction becomes possible. However, overfitting is a common issue, leading to "holes" in AEs' interpolation capabilities. On the other hand, increasing the latent dimension results in a better approximation with fewer non-linearly coupled features (e.g., Koopman theory or kPCA), but it doesn't necessarily lead to dimensionality reduction, which makes feature extraction problematic. As a result, interpolating using Autoencoders gets harder. In this work, we introduce the Rank Reduction Autoencoder (RRAE), an autoencoder with an enlarged latent space, which is constrained to have few dominant singular values (i.e., low-rank). The latent space of RRAEs is large enough to enable accurate predictions while enabling efficient feature extraction. As a result, the proposed autoencoder features a minimal rank linear latent space. To achieve what's proposed, two formulations are presented, a strong and a weak one, that build a reduced basis accurately representing the latent space. The first formulation consists of a truncated SVD in the latent space, while the second one adds a penalty term to the loss function. We show the efficiency of our formulations by using both of them for interpolation tasks and comparing the results to state-of-the-art autoencoders on both synthetic data and MNIST.

1 Introduction

Interpolation of vector functions over a parametric space is an active research topic since accurate interpolation allows the reconstruction of a physical solution in an entire parametric space from a set of pre-computed samples. Multiple techniques have been proposed to perform interpolation, to mention a few, the Proper Orthogonal Decomposition with Interpolation (PODI) [24, 17, 19], or the sparse-PGD (sPGD) [7]. Most of these techniques are based on Model Order Reductions, such as the Proper Orthogonal Decomposition (POD) [12], the Proper Generalized Decomposition (PGD) [21, 6], and the Principal Component Analysis (PCA) [9]. These techniques stack the vector functions in what is called the solution matrix, and their efficiency is inversely proportional to the rank of this matrix. If the solution matrix only has a few dominant singular values (i.e., low-rank), it is easier for the aforementioned techniques to reduce the problem and interpolate. However, when this assumption does not apply, they fail to define an efficient surrogate for the correct prediction of physical phenomena. A high-rank solution matrix reduces the efficiency of techniques based on different formulations such as those based on Grassmann manifolds [1], the Optimal Transport (OT) [25], or every low-rank technique (e.g. [22, 14]).

On the other hand, the nonlinearity in autoencoders with Neural Networks as their encoding and decoding functions makes them appealing to be used as interpolators [10]. Their modified architectures have been used for different purposes in many applications such as speech recognition [4], biophysics [13], medicine [18], and others [2]. Yet, the efficiency of Vanilla Autoencoders is limited. On one hand, Autoencoders with reduced latent spaces (or Diabolo Autoencoders (DAEs)), can easily overfit the data and are known for having "holes" in their latent spaces. On the other hand, even though enlarged latent spaces lead to better approximations with less non-linear behavior (as stated in multiple theories like the Koopman theory or the kPCA), the representations learned are usually of a large dimension which limits both interpolation and feature extraction. This has led to multiple enhancements such as Variational AEs [8, 26], Sparse AEs [20], and denoising AEs [27] which improved Autoencoders overall but did not definitively solve the interpolation issues.

Neural Networks are increasingly being used for nonlinear reduction techniques [3, 5]. Recently, the Implicit Rank-Minimizing Autoencoder (IRMAE) [11], and the Low-Rank Autoencoder (LoRAE) [15] showcased how increasing the latent space dimension while encouraging a low-rank achieves better results, for both approximation and interpolation. If the latent space is of low rank, the efficiency of all presented interpolation techniques (including basic linear interpolation) in the latent space is enhanced. The resulting Autoencoder would benefit from the large data dimensionality of the latent space to find better approximations while allowing feature extraction because of its low rank. The architecture of IRMAEs consists of adding linear layers between the encoder and the latent space, while LoRAE only adds one linear layer as well as its nuclear norm as a penalty term in the loss. While both papers show how their resulting latent spaces may exhibit a lower rank compared to Vanilla and Variational Autoencoders, their work has some limitations. First, while both architectures may find a low-rank latent space with singular values that are sharply decreasing, they do not enforce the small singular values to go to zero. Accordingly, the decoder always has some noise from the small singular values, even though ideally, we would like to remove their effect entirely. In addition, the computational time of both architectures highly depends on the latent space dimension L. Since a long latent space usually helps in achieving better results, both IRMAEs and LoRAEs can be computationally expensive. Finally, both architectures do not provide explicit control over the rank of the latent space. While they include some tuning parameters, we show later in the paper that the optimal parameters proposed in the papers can not reach a satisfying low-rank space.

In this paper, we present the Rank Reduction Autoencoder (RRAE), which has a large latent space restricted to have a low rank. By enforcing the latent space to accept a linear reduction (hence a lower rank), we show that our model resolves the issues previously mentioned. Our architecture includes two proposed formulations: (i) a strong and (ii) a weak one. Throughout the paper, we show that the strong formulation finds orthogonal basis vectors through a principal component analysis of the latent space, while the weak formulation is allowed to find non-orthogonal ones. Further, our results illustrate that both proposed formulations can interpolate efficiently whether between high-rank synthetic solutions, or between MNIST pictures while achieving both a lower latent space rank than the IRMAE and the LoRAE, and a lower computational overhead.

The present paper is structured as follows: Section 2 presents the architecture and both proposed formulations. Section 3 explains the insights behind long latent spaces with a low rank on two synthetic examples. Then, we compare the interpolation capabilities of RRAEs with IRMAEs, and LoRAEs on a variety of problems in section 4. We explain the limitations of the proposed formulations in section 5, before finally summarizing the main original contributions of the present paper in section 6.

2 Rank Reduction Autoencoders (RRAEs)

To define the architecture of RRAEs, we begin by defining autoencoder notations. Let $\{X_i\}_{i\in[1,D]} \in \mathbb{R}^T$ be a set of D series of observations, each having T degrees of freedom. We define our input $X \in \mathbb{R}^{T\times D}$ with X_i as its ith column. Let, $Y \in \mathbb{R}^{L\times D}$, with L, the chosen dimension[1] of the latent space. We also define the encoding map $e : \mathbb{R}^{T\times D} \to \mathbb{R}^{L\times D}$ and the decoding map $d : \mathbb{R}^{L\times D} \to \mathbb{R}^{T\times D}$. The Vanilla autoencoder can be written as the following two operations,

$$Y = e(X), \qquad \tilde{X} = d(Y). \tag{1}$$

In practice, we usually enforce that the output of the autoencoder gives us back the original data, hence the loss $\mathcal{L}$ usually reads,

$$\mathcal{L}(X, \tilde{X}) = \|X - \tilde{X}\|_2, \quad \text{where,} \quad \|\cdot\|_2 \text{ is the } L2\text{-norm}. \tag{2}$$

[1] See Appendix B for details on the choice of L.

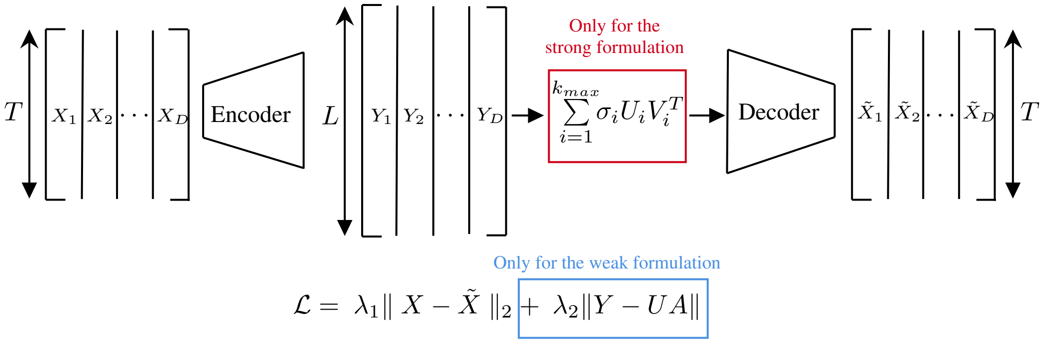
T
X_1 X_2 $\cdots$ X_D
Encoder
L
Y_1 Y_2 $\cdots$ Y_D
Only for the strong formulation
$\sum_{i=1}^{k_{max}} \sigma_i U_i V_i^T$
Decoder
$\tilde{X}_1$ $\tilde{X}_2$ $\cdots$ $\tilde{X}_D$
T
Only for the weak formulation
$\mathcal{L} = \lambda_1 \| X - \tilde{X} \|_2 + \lambda_2 \|Y - UA\|$

Figure 1: Schematic showing the autoencoder in use as well as both formulations. There are two terms in the loss function for the Weak formulation. On the other hand, there's an additional step before the decoder for the Strong formulation.

The idea behind RRAEs is to enforce the latent matrix to have a low rank while finding a reduced basis. In other words, if Y has a rank r, let $Y = U\Sigma V^T$ be the Singular Value Decomposition (SVD) [23, 16] of Y, with $U \in \mathbb{R}^{L\times r}$, $\Sigma \in \mathbb{R}^{r\times r}$, and $V^T \in \mathbb{R}^{r\times D}$. Let $\{\sigma_i\}_{i\in[1,r]}$ be the sorted diagonal values of Σ. Thus, by considering the k most significant modes (choice of k discussed in Section 4.2 and Appendix B), it results,

$$Y = \sum_{i=1}^{r} \sigma_i U_i V_i^T \qquad \Rightarrow \qquad Y \approx \sum_{i=1}^{k} \sigma_i U_i V_i^T, \qquad k \ll r, \tag{3}$$

where U_i is the ith column of U and V_i^T is the ith row of V^T. In other words, we can write Y_d, the dth column of Y as,

$$Y_d \approx \sum_{i=1}^{k} \left(\sigma_i U_i V_i^T\right)_d = \sum_{i=1}^{k} \sigma_i U_i V_{i,d}^T = \sum_{i=1}^{k} \alpha_{i,d} U_i, \qquad \forall d \in [1, D], \tag{4}$$

with $V_{i,d}^T$ being entry d of vector V_i^T.

Accordingly, for k modes, each column of Y is defined by k coefficients and k vectors. Further, vectors U_i form a basis for the latent space. We can write (4) in matrix form as follows,

$$Y \approx UA, \qquad \text{with: } A_{i,j} = \alpha_{i,j}, \quad U \in \mathbb{R}^{L\times k}. \quad A \in \mathbb{R}^{k\times D}, \tag{5}$$

Based on (4), and (5), we propose two formulations that enforce the low rank of the latent space and find its reduced basis. The architecture is sketched in Figure 1.

1. The Weak formulation: After choosing the maximum allowed number of modes k_{max}, we generate two trainable matrices $A \in \mathbb{R}^{k_{max}\times D}$, and $U \in \mathbb{R}^{L\times k_{max}}$. Afterward, we add a term to the loss as seen in blue in Figure 1. By doing so, minimizing the loss means that the latent space would have at most a rank of k_{max}. After convergence, the columns of our trainable matrix U form the reduced basis of the latent space. Additionally, the coefficients found in matrix A describe how to reconstruct each column Y_d as a linear combination of the basis vectors. We will refer to this method as the Weak formulation since throughout training, the Network minimizes a sum of both terms and not each term individually. Accordingly, predictions $\tilde{X}$ could be less accurate, and we might end up with more modes than the specified value of k_{max}.

   Remark: The two trainable matrices can be computed from a one-rank greedy procedure, as PGD performs.

2. The Strong formulation: Unlike the weak formulation, this architecture enforces, in a strong manner, the maximum dimension of the reduced basis of the latent space. Similarly to the weak formulation, we begin by choosing the maximum rank k_{max} of the latent space. Then, as seen in red in Figure 1, a truncated SVD (of order k_{max}) of the latent space is given to the decoder, instead of the latent space itself. Accordingly, the input of the decoder (or our new

latent space) will have at most k_{max} dominant singular values. We refer to this method as the Strong formulation since we strictly enforce the latent space to have a rank that's lower or equal to k_{max}. In this case, the basis vectors and coefficients are simply the ones found by the truncated SVD.

In both formulations, k_{max} is a hyperparameter to be chosen. We propose a strategy to choose this hyperparameter and discuss its effect in Section 4.2 and in Appendix B.

When using the strong formulation, we compute a POD basis, where the vectors are by construction orthogonal. The orthogonality of the basis vectors, as well as refraining from adding terms in the loss, can enhance both the training and interpolation results. On the other hand, backpropagation through the singular value decomposition is not common in practice. All the work presented in this paper was performed using `equinox` in `JAX`, where gradients of the singular value decomposition are implemented and accessible.

Both formulations reduce the limitations of IRMAE and LoRAE. We sum up our contributions as follows:

1. RRAEs with a strong formulation lead to low-rank latent spaces that have many singular values exactly equal to zero. In other words, the decoder will get a sum of exactly k_{max} rank-one updates. As will be shown later in the paper, this gives the strong formulation an advantage for training and interpolation.
2. The computational overhead of RRAEs is reduced compared to other architectures, especially for large latent spaces. For the strong formulation, when batches are used, the SVD is only performed on a matrix of size $L \times bs$, bs being the batch size. Similarly, for the weak formulation, the added computational cost is minimal since the trainable matrices are of shape $L \times k_{max}$ and $k_{max} \times D$ with $k_{max} \ll L$. On the other hand, the IRMAE or the LoRAE either performs a gradient descent or finds the nuclear norm of an $L \times L$ matrix. Since a large latent space dimension L usually helps in achieving better results, both IRMAEs and LoRAEs can be computationally challenging.
3. Both formulations give us explicit control over the rank of the latent space. As shown next in the paper, we can enforce the latent space to have a lower rank than IRMAE and LoRAE, which leads to better interpolation and could help for feature extraction in future applications.

3 Insights behind Long latent spaces with low rank

An enlarged latent space can exhibit a linear behavior (as explored for instance in the Koopman theory, or the kPCA). Furthermore, a latent space with a reduced basis allows easier interpolation and feature extraction. The Diabolo Autoencoder on the other hand has "holes" in its interpolation [11], since it does not find a basis, but only a set of coefficients that are helpful for the decoder to retrieve the solution. Since the decoder is highly nonlinear, these coefficients can be anything, which leads to overfitting.

To illustrate the aforementioned arguments, we test DEAs and our Strong formulation on two examples characterized by one parameter. The first curves we propose are shifted sine curves since these have a simple nonlinearity, but they are hard to separate (nonmonotonic and cross each other multiple times). For our second example, we chose curves with stair-like behavior. In that case, we create highly nonlinear curves (different supports, different numbers of jumps of different magnitudes), but we define them to be monotonic and only cross each other occasionally (i.e. easier to separate). The equations used to define the columns of our input matrix X in each case are as follows,

$$\begin{cases} X_d(t_v, p_d) = f_{shift}(t_v, p_d) = \sin(t_v - p_d\pi), & p_d \in [0, 1.7], \\ X_d(t_v, p_d) = f_{stair}(t_v, p_d, \text{args}) & p_d \in [1, 5], \end{cases}$$

where $t_v \in \mathbb{R}^T$ is the time discretization vector, and f_{stair} takes some arguments "args" as detailed in the algorithm in Appendix A. The training is performed over 17, and 40 equidistant values of p_d for the shifted sine curves and the stair-like curves respectively. Later on, we interpolate the resulting curves in the latent space of the autoencoders on 80 and 300 random values of p_d, respectively, chosen

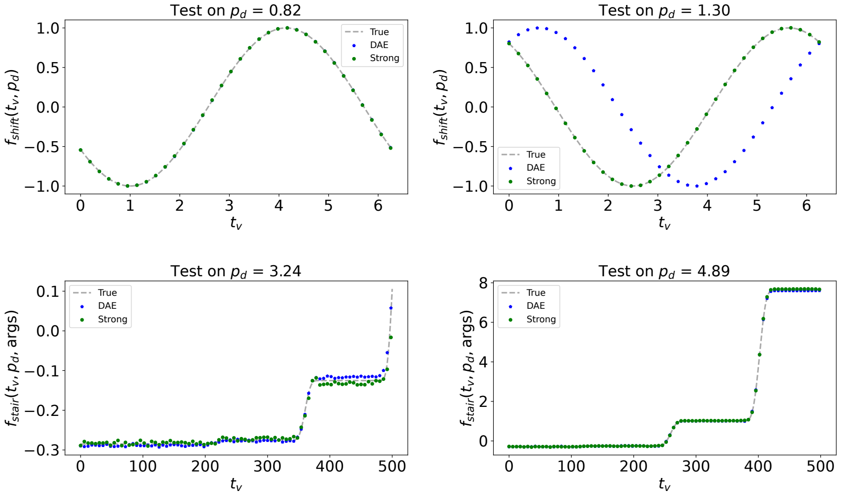

Figure 2: Predictions of DAEs and RRAEs with both formulations over two particular values of p_d for the shifted sine (above) and the stair-like examples (below).

inside the training domain. The large number of tests guarantees that the models are learning the dynamics and not just the training curves and some tests nearby. Since the solution curves depend on one parameter, we use a DAE with a single scalar latent space and an RRAE with a longer latent space of rank one. We then linearly interpolate in the latent space to predict the test set. The training parameters, including the dimension of the latent space, can be found in Appendix B. The relative error over all p_d values for both the train and test sets is summarized in Table 1. Further, for each example, the predictions over some selected test cases are plotted in Figure 2.

Table 1: Relative error (in %) for all three architectures on both the train and test sets for both the examples of shifted sin curves and stair-like ones.

	Shifted sine		Stair-like	
Model	Train Error	Test error	Train Error	Test error
DAE	2.12	32.42	2.97	3.74
RRAE (strong)	**1.73**	**1.90**	**1.87**	**3.2**

The results show that when curves are hard to separate, RRAEs are better interpolators than DAEs. On the other hand, the effect of longer latent spaces is reduced for simple curves that can be highly nonlinear, but characterized by one parameter, and easily separable.

To further investigate the results, we plot the coefficients to be interpolated in the latent space as a function of the corresponding parameter p_d in Figure 3. It is important to note that the coefficients are defined differently between the RRAE and the DAE. For RRAEs, when $k_{max} = 1$, the coefficients are simply the entries of $A \in \mathbb{R}^{1\times D}$ in equation (5). On the other hand, for a Diabolo Autoencoder with a scalar latent space, the values in the latent space themselves are the coefficients.

The main problem with the coefficients found by the DAE for the shifted sine curves (the blue crosses and dots in Figure 3 (left)) is that the resulting curve from linearly interpolating the coefficients is not an injection, over two significant parts of the domain. Specifically, for any value of p_d in approximately $[0, 0.3]$ and $[1.3, 1.5]$ (between the dotted lines), there exists another value with the same coefficient α, leading to the same decoded curve. Accordingly, the decoder will find the same curve for two different parameters, which is wrong since p_d defines a shift. This explains why the DAE can interpolate well in the top left subplot in Figure 2, but not in the top right one. These results also show what is meant by “holes” in the latent space for DAEs.

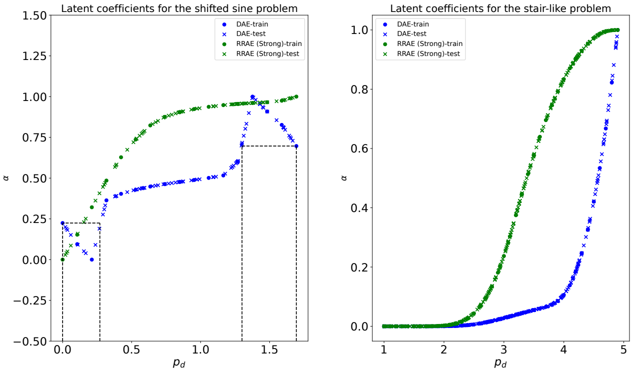

Figure 3: Normalized latent coefficients to be interpolated (dots) for DAE and RRAE with a strong formulation, and the interpolated values for the test set (crosses) for the shifted sine curves (left) and the stair-like curves (right).

On the other hand, as proposed earlier, a longer latent space allows us to find better features. This is clearly shown by the coefficients of the strong method in Figure 3 (left), which have a monotonic behavior.

Finally, the right part of Figure 3 depicts that when the curves are simple to separate and are characterized by only one parameter, both architectures can find monotonic coefficients that fit both the train and test sets.

4 Testing on Numerical Data

The solutions interpolated in the previous section were only characterized by one parameter. In this section, we test RRAEs and compare them to IRMAEs, and LoRAEs on two examples with a parametric space of dimension two, as well as on the MNIST dataset. Throughout the paper, we don't compare RRAEs with different variations of VAEs. Our future work will include a Variational version of RRAEs and its comparison to different VAE architectures.

4.1 Examples with two parameters

We generated two challenging synthetic tests for interpolation. First, we propose the sum of two sine curves with different frequencies, as well as two Gaussian bumps in two different locations. We show how in such examples both our formulations result in latent spaces with a lower rank and better results than IRMAEs and LoRAEs for the hyperparameters chosen (again, training details can be found in Appendix B). We define the columns of our input matrix $X_d(t_v,\, p_d) = f_{prob}$ for each problem as follows,

$$\begin{cases} f_{freqs}(t_v,\, \mathbf{p}_d) = \sin(p_d^1 \pi t_v) + \sin(p_d^2 \pi t_v), & p_d^1 \in [0.3, 0.5], \quad p_d^2 \in [0.8, 1], \\ f_{gauss}(t_v,\, \mathbf{p}_d) = 1.3e^{-\frac{(t_v - p_d^1)^2}{0.08}} + 1.3e^{-\frac{(t_v - p_d^2)^2}{0.08}}, & p_d^1 \in [1, 3], \qquad p_d^2 \in [4, 6]. \end{cases}$$

We distinguish between the **bold** notation for vectors and non-bold ones for scalars. In both expressions, our parametric space is of dimension 2 and so $\mathbf{p}_d = [p_d^1,\, p_d^2]^T$. For each example and each architecture, we present some interpolated predictions in Figure 4, and the error over all the training/testing sets, as well as the average training time for 100 batches, are summarized in Table 2.

As can be seen in Table 2, RRAEs with the Strong formulation are the most efficient in interpolation. Additionally, we note that increasing the parameter l for the IRMAE leads to divergence of the gradient descent (hence the N/A). Note that we only used the parameters specified in both papers for IRMAE and LoRAE. A fine-tuning of the parameters may potentially lead to better results for these architectures, but this venue is not investigated in this work. On the other hand, the table shows that RRAEs are faster than both IRMAEs and LoRAEs for the latent space dimension chosen. Our formulations are fast since we only add small matrices to the loss in the weak formulation, and we compute an SVD of an $L \times bs$ matrix, bs being the batch size, in the Strong formulation. On the

Table 2: Relative error (in %) for all architectures on both the train and test set for the two examples presented, and the average time (in s) for 100 batches (size 20).

	Two sines (freqs)		Two Gaussians		
Model	Train Error	Test error	Train Error	Test error	Average time
RRAE (strong)	6.33	**12.83**	4.46	**8.75**	1.61
RRAE (weak)	10.33	15.09	8.50	10.69	**0.52**
IRMAE (l=2)	6.95	17.35	4.68	13.93	3.6
IRMAE (l=4)	N/A	N/A	8.41	14.78	7.50
LoRAE	**5.40**	13.83	**3.03**	9.39	420.4

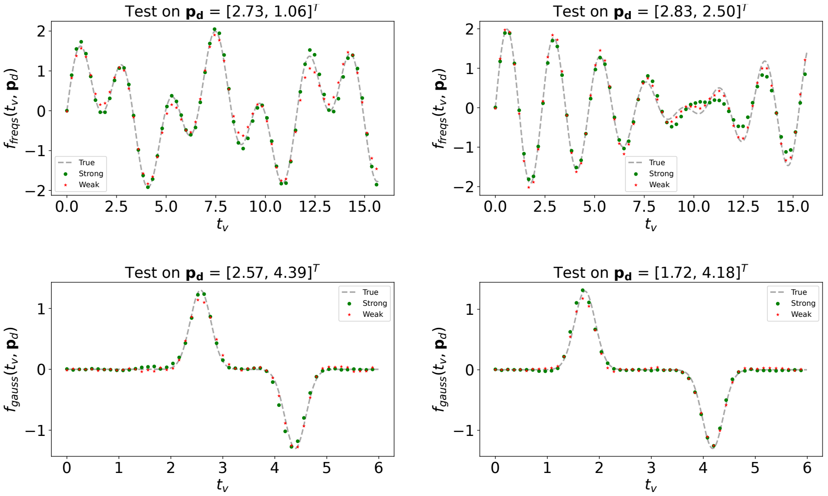

Figure 4: Interpolated results of RRAEs with both formulations on both examples presented with bilinear interpolation in the latent space.

other hand, IRMAEs and LoRAEs find the gradient/the nuclear norm of an $L \times L$ matrix respectively, with L relatively large.

Additionally, we draw some of the normalized singular values of the latent space for the problem with the two Gaussians in Figure 5. The figure illustrates that adding the number of linear layers for the IRMAE (i.e. increasing l) indeed reduces the rank of the latent space. However, both of our formulations can be restricted to finding latent spaces with lower ranks. In addition, it is important

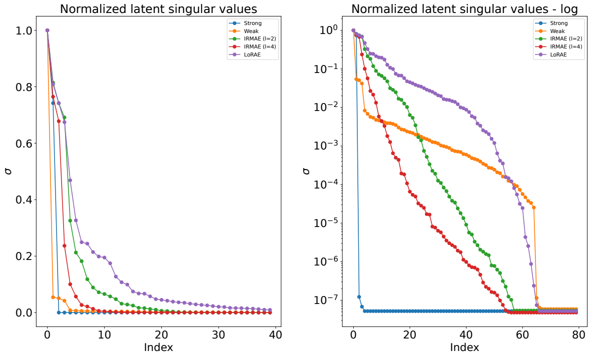

Figure 5: Normalized singular values of the latent space Y when trained over the two Gaussians. The first 40 singular values are shown to the left, and the first 80 singular values are shown on a log scale to the right.

to note that even though the LoRAE has a low error overall, the latent space rank is still relatively high compared to the other techniques (as can be seen from the slowly decreasing singular values in violet in Figure 5 (left)). Hence, the trained model can not efficiently be used for other tasks, such as feature extraction. From the log-scaled graph illustrated in Figure 5 (right), we can understand why the strong formulation achieves the best results. As previously mentioned, the IRMAE, the LoRAE and our weak formulation don't enforce the singular values to fall to zero. So even though many singular values are small, they still have a noise effect over the decoder, which reduces their efficiency in interpolation.

4.2 Testing on MNIST

In this section, we compare our architecture to the IRMAE and the LoRAE on the MNIST dataset. Previously on our synthetic data, we showed that interpolating between two sine curves (for instance), in the latent space leads to another decoded sine function (and not a random one). Similarly, autoencoders should be able to interpolate between MNIST pictures of numbers to find other pictures of numbers in between. To illustrate this, we begin by training an autoencoder with each architecture on all $60,000$ pictures from the training set (training parameters are available in Appendix B). Then, we use each model to create interpolated pictures and form an "interpolated set". Interpolation is done by randomly choosing two pictures from the training set (e.g., leftmost and rightmost in Figure 6) and linearly interpolating their latent variables to find five new pictures in between. Interpolated pictures (in the red rectangle from left to right in Figure 6) are expected to transition from the leftmost picture to the rightmost one in an equidistant manner. For instance, the pictures in the second column of subplots in Figure 6 take $5/6$ of the first picture (i.e. number 7) and $1/6$ of the last picture (i.e. number 3). Similarly, pictures in the third column would have proportions of $4/6$ and $2/6$ respectively. This procedure is then repeated 2000 times to create, for each architecture, an interpolated set of size $10,000$. To quantify the quality of the generated images, we train a classifier on the original training set and test it on the interpolated set generated by each architecture. Our classifier is a multilayer perceptron with a `softmax` final activation function (details in Appendix B). It takes as input an MNIST picture and outputs a probability for every possible class (i.e., in $[0,1]^{10}$). Since the labels of the interpolated images are unknown, we measure the success of classification by the certainty of the classifier, which we quantify by the entropy of the probability distribution written as follows,

$$H = \frac{-1}{10000} \sum_{i=1}^{10000} \sum_{j=1}^{10} p_j^i \log(p_j^i), \tag{6}$$

with p_j^i being the probability of class j predicted by the softmax activation function for sample i. More details about the entropy can be found in Appendix E. The lower the entropy, the more the classifier is certain about the class prediction of the interpolated image. Hence a low entropy means that the interpolated images resemble numbers between 0 and 9, just like the original dataset. We

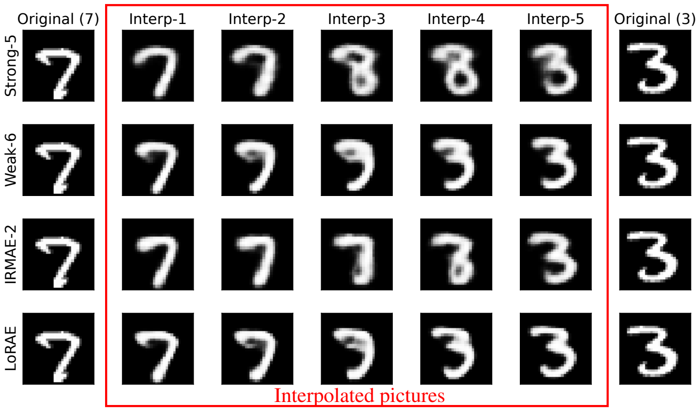

Figure 6: Five-step interpolation between a number 7 (leftmost) and a number 3 (rightmost) on MNIST using RRAEs with both formulations, IRMAEs, and LoAREs. Numbers next to method names represent k_{max} for the RRAEs and the number of linear layers l for the IRMAE.

perform training using our Strong formulation with three choices of k_{max} to illustrate how the choice of this hyperparameter affects the model. We also generate five different interpolated sets for each architecture using different training images for interpolation. The mean and standard deviation of the entropy over the five sets are presented in Table 3.

Table 3: Mean entropy, standard deviation (over five different interpolation sets) and the rank of the latent space for the Strong formulation with three values of k_{max}, the weak formulation with $k_{max} = 6$, the IRMAE with $l = 2$, and the LoRAE, on their corresponding MNIST interpolated sets.

Model	Strong-5	Strong-8	Strong-12	Weak-6	IRMAE-2	LoRAE
H	0.50 ± 6e-3	**0.45 ± 4e-3**	0.49 ± 4e-3	0.51 ± 5e-3	0.50 ± 6e-3	0.46 ± 5e-3
Rank	**5**	8	12	6	8	30

The table illustrates how the choice of k_{max} can be made. We propose to start with a small value and increase it until the error stagnates or increases. In this case, $k_{max} = 8$ is the best choice, but the Strong formulation can interpolate the MNIST pictures even when restricted to 5 features. Both Table 3 and Figure 6 illustrate how RRAEs can interpolate well between MNIST pictures while using a lower rank. The strong formulation, for instance, recognizes that it is hard to go from 7 to 3 and goes through 8 (first line interp-3,4,5 in figure 6), while the others find unclear figures along the way. Further, while the LoRAE has a low entropy, its latent space has 30 dominant singular values, which doesn't allow any feature extraction. On the other hand, compared to IRMAEs in Table 3, the noise from the smaller singular values as well as the explicit control over the rank allows us to get either a smaller entropy for the same rank (Strong-8) or almost the same entropy with fewer features to extract (Strong-5 and Weak-6).

5 Limitations

As illustrated in the paper's results, RRAEs with both formulations can interpolate well while using a latent space with a lower rank. However, our proposed model has some limitations:

1. Even though both of our formulations allow explicit control over the rank of the latent space, k_{max} is a hyperparameter to be tuned. In practice, starting with a small value of k_{max} and increasing it until error convergence is a good strategy. In general, other techniques (e.g., PCA) could be used to approximate the intrinsic dimension of the latent space "à priori".
2. The weak formulation adds regularisation constants to the loss, which can be hard to tune. In practice, we had to repeat training multiple times to tune the parameters, which isn't ideal, especially for a larger dataset such as the MNIST.
3. For high dimensional problems with very long latent spaces, the strong formulation can be computationally expensive. Even though the SVD is only performed on a matrix of size $L \times bs$, bs being the batch size and L the length of the latent space, the cost of computing an SVD and backpropagating through it when L is excessively large can be high.
4. The effect of a long latent space is reduced when the solution is simple and separable. In such cases, an increased dimension of the latent space, hence RRAEs, may not be necessary.

6 Summary and Conclusions

In this article, we presented Rank Reduction Autoencoders (RRAEs), Autoencoders with latent spaces that accept linear reduction. We proposed two formulations, a weak and a strong one to find the latent space while building its reduced basis. Even though the basis vectors in the strong formulation are orthogonal, and they need not be in the weak formulation, we showed that both formulations can interpolate correctly between curves. Overall, our results show that the Strong formulation has a superior capability of interpolation since it doesn't have any noise from small nonzero singular values in the latent space. We also showed that both the Strong and the Weak formulations can achieve lower ranks in the latent space while being able to efficiently interpolate vector functions. Finally, both formulations are fast to train, with the weak formulation being the fastest. While the Strong formulation leads to better predictions, the Weak formulation is much simpler to implement since it only adds a penalty term to the loss.

References

[1] David Amsallem and Charbel Farhat. Interpolation method for adapting reduced-order models and application to aeroelasticity. *Aiaa Journal - AIAA J*, 46:1803–1813, 07 2008. doi: 10.2514/1.35374.

[2] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *Machine learning for data science handbook: data mining and knowledge discovery handbook*, pages 353–374, 2023.

[3] Joshua L Barnett, Charbel Farhat, and Yvon Maday. Neural-network-augmented projection-based model order reduction for mitigating the kolmogorov barrier to reducibility of cfd models, 2022.

[4] Sarath Chandar A P, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. An autoencoder approach to learning bilingual word representations. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper_files/paper/2014/file/2bcab9d935d219641434683dd9d18a03-Paper.pdf`.

[5] Yingyi Chen, Qinghua Tao, Francesco Tonin, and Johan Suykens. Primal-attention: Self-attention through asymmetric kernel svd in primal representation. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 65088–65101. Curran Associates, Inc., 2023. URL `https://proceedings.neurips.cc/paper_files/paper/2023/file/cd687a58a13b673eea3fc1b2e4944cf7-Paper-Conference.pdf`.

[6] Francisco Chinesta and Elias Cueto. *PGD-based modeling of materials, structures and processes*. Springer, Switzerland, 2014.

[7] Francisco Chinesta, Pierre Ladeveze, and Elias Cueto. A short review on model order reduction based on proper generalized decomposition. *Archives of Computational Methods in Engineering*, 18(4):395–404, 2011.

[8] Eizaburo Doi and Michael Lewicki. Sparse coding of natural images using an overcomplete set of limited capacity units. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2004. URL `https://proceedings.neurips.cc/paper_files/paper/2004/file/309a8e73b2cdb95fc1affa8845504e87-Paper.pdf`.

[9] David González, José Vicente Aguado, E Cueto, E Abisset-Chavanne, and F Chinesta. kpca-based parametric solutions within the pgd framework. *Archives of Computational Methods in Engineering*, 25:69–86, 2018.

[10] Guillaume Huguet, Daniel Sumner Magruder, Alexander Tong, Oluwadamilola Fasina, Manik Kuchroo, Guy Wolf, and Smita Krishnaswamy. Manifold interpolating optimal-transport flows for trajectory inference. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 29705–29718. Curran Associates, Inc., 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/file/bfc03f077688d8885c0a9389d77616d0-Paper-Conference.pdf`.

[11] Li Jing, Jure Zbontar, and yann lecun. Implicit rank-minimizing autoencoder. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 14736–14746. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/a9078e8653368c9c291ae2f8b74012e7-Paper.pdf`.

[12] Gaetan Kerschen, Jean-claude Golinval, Alexander F Vakakis, and Lawrence A Bergman. The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: an overview. *Nonlinear dynamics*, 41:147–169, 2005.

[13] Tianxiao Li, Hongyu Guo, Filippo Grazioli, Mark Gerstein, and Martin Renqiang Min. Disentangled wasserstein autoencoder for t-cell receptor engineering. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 73604–73632. Curran Associates, Inc., 2023. URL `https://proceedings.neurips.cc/paper_files/paper/2023/file/e95da8078ec8389533c802e368da5298-Paper-Conference.pdf`.

[14] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):171–184, 2012.

[15] Alokendu Mazumder, Tirthajit Baruah, Bhartendu Kumar, Rishab Sharma, Vishwajeet Pattanaik, and Punit Rathore. Learning low-rank latent spaces with simple deterministic autoencoder: Theoretical and empirical insights, 2023.

[16] Yuji Nakatsukasa. Accuracy of singular vectors obtained by projection-based svd methods. *BIT Numerical Mathematics*, 57(4):1137–1152, 2017.

[17] Minh-Nhan Nguyen and Hyun-Gyu Kim. An efficient podi method for real-time simulation of indenter contact problems using rbf interpolation and contact domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 388:114215, 2022.

[18] Daehyung Park, Yuuna Hoshi, and Charles C Kemp. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *IEEE Robotics and Automation Letters*, 3(3):1544–1551, 2018.

[19] RR Rama and S Skatulla. Towards real-time modelling of passive and active behaviour of the human heart using podi-based model reduction. *Computers & Structures*, 232:105897, 2020.

[20] Marc' aurelio Ranzato, Y-lan Boureau, and Yann Cun. Sparse feature learning for deep belief networks. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper_files/paper/2007/file/c60d060b946d6dd6145dcbad5c4ccf6f-Paper.pdf.

[21] S Rodriguez, David Néron, P-E Charbonnel, Pierre Ladevèze, and G Nahas. Non incremental latin-pgd solver for non-linear vibratoric dynamics problems. In *14ème Colloque National en Calcul des Structures, CSMA 2019*, 2019.

[22] Nathan Srebro and Tommi Jaakkola. Weighted low-rank approximations. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 720–727, 2003.

[23] Gilbert W Stewart. On the early history of the singular value decomposition. *SIAM review*, 35 (4):551–566, 1993.

[24] Marco Tezzele, Nicola Demo, and Gianluigi Rozza. Shape optimization through proper orthogonal decomposition with interpolation and dynamic mode decomposition enhanced by active subspaces, 2019.

[25] Sergio Torregrosa, Victor Champaney, Amine Ammar, Vincent Herbert, and Francisco Chinesta. Hybrid twins based on optimal transport. *Computers & Mathematics with Applications*, 127: 12–24, 2022. ISSN 0898-1221. doi: https://doi.org/10.1016/j.camwa.2022.09.026. URL https://www.sciencedirect.com/science/article/pii/S0898122122004060.

[26] Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural discrete representation learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/7a98af17e63a0ac09ce2e96d03992fbc-Paper.pdf.

[27] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We include mainly three claims (zero singular values, computational time, and explicit control over the rank), which are all tackled later in the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 basically discusses the main limitations of our proposed model. We also discussed how Diabolo Autoencoders can achieve similar results to RRAEs in certain training conditions in Section 3.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We include all the necessary details for creating the synthetic datasets throughout the article. We also detail all the hyperparameters used for training, as well as other details (such as seeds used for creating randomness) in Appendix B.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: While we can not share the code used, we provide all the necessary details for generating the data and performing training throughout the article and in the appendices.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide the training parameters for each example in the corresponding section. All the parameters, including `JAX` random keys to exactly reproduce the test sets can be found in Appendix B.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: On the MNIST dataset, we repeated experiments five times with different interpolation sets and we documented both the mean and the standard deviation of the results in Table 3 (with an explanation of how they were computed).

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All of the details related to the PC specifications were mentioned in Appendix B, we also provide the time of 100 batches for training over the synthetic problems in Table 2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The code of ethics has been thoroughly read by the corresponding author who made sure that the research conducted in the paper conforms to it. The format of the paper is anonymous (following the guidelines of the template provided).

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: Since we are working on foundational research with no particular applications. We believe our topic is too broad to have any specific societal impacts.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We are the owners of the model presented and we credited the libraries and languages that were used for the code.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The model we propose with both formulations is detailed as a concept in Section 2, our appendices include more details on how to use the model with both our proposed formulations.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

Appendix A Algorithm for the stair-like function

```
**Algorithm 1:** Algorithm to find $f_{stair}$ for a parameter $p_d$.
**Input:** $p_d \in \mathbb{R}, t_v \in \mathbb{R}^T, (\text{Ph}_0, \text{Amp}_0, \kappa, y_0, w) \in \mathbb{R}$
$\text{Amp}_{p_d} = p_d$
$\text{Ph}_{p_d} = \text{Ph}_0 + \kappa(\text{Amp}_{p_d} - \text{Amp}_0)$
$g_{p_d}(t_v) = \text{Amp}_{p_d}\sqrt{t_v}\sin(w(t_v - \text{Ph}_{p_d})) - y_0$
$h_{p_d}(t) = \left(\dfrac{|g_{p_d}(t)| + g_{p_d}(t)}{2}\right)^5$
$X_d(t_v,\, p_d) = \text{cumsum}(h_{p_d}(t_v))$
**Output:** $X_d(t_v,\, p_d)$ for each parameter $p_d$.
```

In this paper, we choose the initial parameters of the stair function to be,

$$\begin{cases} \text{Ph}_0 = 0.875, \qquad \text{Amp}_0 = 1 \\ \kappa = 2.286, \qquad y_0 = 2.3, \qquad w = 2\pi. \end{cases}$$

Appendix B Training details

The main purpose of this appendix is to allow readers to reproduce the results, and share the parameters we used to train the models.

B.1 Hyperparameters for synthetic data and PC properties

In general, the main parameters for RRAEs are the dimension of the latent space, k_{max}, the encoder/decoder architectures, learning rates, epochs, and batch sizes. For each problem, we fix the common parameters between all architectures. We try to only change necessary parameters between different examples, to show that training RRAEs, especially with the strong formulation, doesn't require too much hyperparameters tuning. For all problems (except MNIST) and all formulations, we use an encoder of depth 1 and width 64, and a decoder of depth 6 and width 64. Additionally, we use batches of size 20, the `softplus` as activation function between all layers, and the `adabeleif` optimizer. Furthermore, we use multiple learning rates, starting with $1\text{e}{-3}$ and dividing by 10 until reaching $1\text{e}{-5}$ (3 steps). In each step, we train for 2000 batches. however, we impose stagnation criteria which usually stop training earlier. Further, we normalize the data by subtracting the mean and dividing it by the standard deviation. We found that normalization was necessary, especially for the stair-like functions.

Throughout the paper, the only two parameters that we vary for RRAEs are the length of the latent space L and a coefficient κ_w that changes the learning rates for the trainable matrices of the weak method. In practice, we found that changing the learning rate of the trainable matrix A for the weak formulation is easier than changing the weights in the loss. Accordingly, while we use the same learning rate strategy proposed before for the encoder/decoder, we propose to multiply the learning rate by a constant κ_w before applying it to the trainable matrix A. By doing so, we find that there is no need to tune the loss parameters (i.e. both are equal to one). It is important to note that the vectors in the trainable matrix U are normalized at every training step so A captures the coefficients. In Table 4, we illustrate the latent space dimension L and the constant κ_w used for all the illustrated examples in this work (N/A means the weak method was not used for this example).

Next, we detail how the choice of k_{max} was made for each example. In general, an approximation of k_{max} can be found using multiple techniques (e.g. PCA). However, in this paper, we chose a simpler approach. If this hyperparameter is too small, the model will not converge, but if it is too large, the Neural Network will simply learn a latent space of a higher rank. Accordingly, we started with a

Table 4: Different values of the latent space length L and the constant κ_w that are used for all the examples in this work (except MNIST).

Param.	Shifts	Stair-like	Mult. Freqs.	Mult. Gauss.
L	4500	4500	2800	2800
κ_w	N/A	N/A	0.66	0.13

small value of k_{max} and increased it until the error converged. The values chosen for each example are detailed in Table 5.

Table 5: Different values of k_{max} that are used for all the examples in this work (except MNIST).

Param.	Shifts	Stair-like	Mult. Freqs.	Mult. Gauss.
k_{max}	1	1	12	2

As can be seen in the table, while we were able to choose exactly the dimension of the parametric space for most of the examples, for the sine curves with different frequencies, the method needed a higher rank in the latent space to converge to the low errors presented. This is mainly because the latent space was not long enough for the problem. However, we tried to fix the parameter L, so we had to change k_{max} accordingly. On the other hand, we chose the parameters that were shown to give the best results for the IRMAE and the LoRAE. We tried to have two and four linear layers for the IRMAE (i.e. $l = 2$ and $l = 4$), and we used a weight of 0.001 in the loss for the LoRAE (the optimal value specified in the presenting paper). Other parameter values for LoRAE and IRMAE were tested with little improvements overall. However, a fine-tuned choice of parameters could lead to better results than the ones presented in the paper. To ensure a fair comparison, every other parameter of these models was chosen to be the same as RRAEs (the ones listed before).

Since we provide average computational times in Section 4.1, we give some details about the machine used to generate these results. The PC used is an MSI Stealth 17Studio A13VH. The processor is Intel (13th generation), Core i9, 2600 MHz, 14 CPUs. The PC also has 64 GB of RAM. The library used was `equinox`, in `JAX` for the training. However, the code was only run on CPUs.

B.2 Hyperparameters for MNIST

The architecture for the MNIST dataset was different since convolutional Neural Networks were used. We fixed the kernel size to 4, the padding to 1, and the strid to 2 for each convolusion/convolusion transpose. For the encoder, we used convolutions with output 32, 64, 128, and 256 respectively, with relu activation functions in between. These were followed by a flattening layer, and a Multilayer Perceptron (MLP) with softplus activation functions, of depth 2, and width 64. The output of the MLP was fixed to be 128, the dimension of the latent space. On the other hand, the decoder included an MLP with depth 2, width 64, and softplus activation functions with an output dimension of 1568. The vector was then reshaped into a tensor of shape $(32, 7, 7)$, which was then followed by two transposed convolutions with output shapes 8, and 1 respectively. For training, we used a learning rate of 0.0001, the optimizer `adabeleif`, and a total of 50 epochs.

Even though the architecture and training are probably not the best ones, our purpose was to show that for a fixed architecture, RRAEs can outperform other existing methods and not achieve SOTA results over the MNIST since this was done for many architectures before.

While the choice of the hyperparameter k_{max} and l (for IRMAE) has been mentioned in Section 4.2, we used again the optimal weight in the loss proposed in the paper of LoRAE as $\lambda = 0.001$. We also used a factor $\kappa_w = 0.8$ for the weak formulation of RRAEs.

Now, we detail how our interpolation sets were created. As previously mentioned in the paper, we choose two random figures from the training set and generate five pictures by interpolating the latent space. This procedure is done 2000 times to generate the interpolation set. The entire thing is done 5 times to provide statistically significant results (i.e. with a mean and a standard deviation). For readers who would like to reproduce our results, we used the seeds 0, 10, 100, 1000, and 10000 to generate a `jax.random` key respectively. The key was then split into 2000 other keys (by using `jax.random.split`). Finally, we fed these keys to create 2000 permutations of the indices of the training figures (i.e. 0 until $60,0000$) and only took the first two numbers as our choice of the figures to be interpolated. By using these seeds, readers should be able to exactly generate the interpolation sets used in the paper. In addition, the example presented in Figure 6 is the interpolation between pictures of indices 52 and 12.

B.3 Choice of parameters for synthetic data

Throughout the paper, we mentioned the range in which the values of $\mathbf{p}_d$ were chosen for each example. In this subsection, we provide some details on the chosen values of $\mathbf{p}_d$, mainly to show that the test covers most of the parametric space. Throughout the paper, we presented curves with parametric spaces of one and two. The following figures show the plot of the second parameter against the first one when the space is of dimension two (Figure 8). On the other hand, when the curve is only characterized by one parameter, we plot the vector of the parameter against itself (Figure 7). Hence, we plot dots on a diagonal line to show where the test values lie compared to the train values. Our test set was chosen randomly but using a `JAX` seed to ensure reproducibility. As can be seen in the figures, we carefully chose the seeds and the number of tests to represent most of the parametric space.

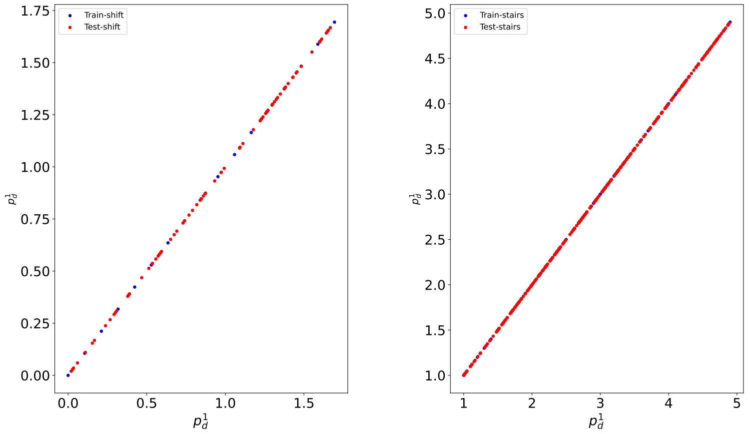

Figure 7: Train and test parameter values for the example with two shifted sine curves (left), and stair-like curves (right).

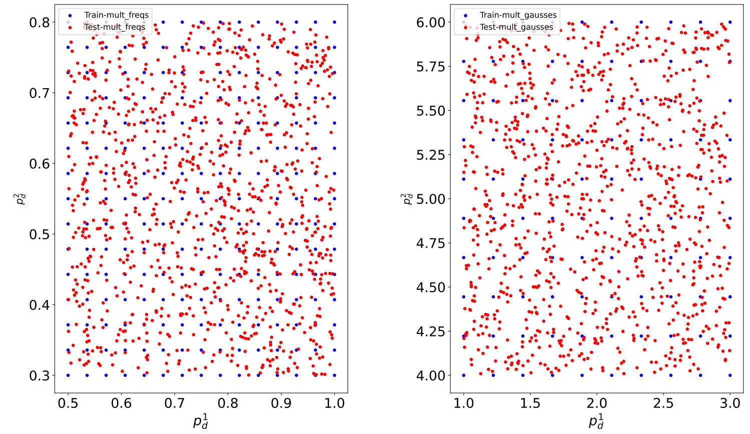

Figure 8: Train and test parameter values for the example with two accelerated sine curves (left), and two Gaussians (right).

We now explain how to exactly reproduce the test set we had. Each random parameter was created using `jax.random.uniform` which takes a `jax.random.key(seed)` as its first parameter. By choosing the same seed, the random numbers are guaranteed to be the same. Accordingly, we provide the seeds used to generate the test sets for each synthetic example in the paper. These can be found in Table 6.

Table 6: Chosen `jax.random` seeds for generating test parameters.

Problem	Shifted sine	Stairs	Two sines (freq)		Two Gaussians	
Parameter	p_d	p_d	p_d^1	p_d^2	p_d^1	p_d^2
Seed	0	0	140	8	1000	50

Appendix C Comparing RRAEs to AEs with long latent dimensions

Throughout the paper, we only compared our proposed models with others that allow feature extraction, since it is of interest to us. However, Vanilla Autoencoders with long latent spaces can interpolate very well! On both examples with the sines with two frequencies and the two Gaussians, the comparison between our Strong formulation and a Vanilla Autoencoder with a latent space of dimension 2800 (same length) is shown in table 7.

Table 7: Error (in %) on our two synthetic problems for our Strong formulation and an AE with the same latent dimension with no rank restriction.

	Mult. Freqs		Mult Gausses	
Model	Train Error	Test error	Train Error	Test error
AE (long)	7.96	14.08	**3.37**	10.56
RRAE (strong)	**6.33**	**12.83**	4.46	**8.75**

As can be seen in the table, reducing the rank not only allows feature extraction, it can also help in training to achieve better results and interpolation.

Appendix D Batching

In this section, we detail how batching was performed for both the Weak and the Strong formulations.

The weak formulation: We remind the reader that the weak formulation had the norm of $Y - UA$ in the loss, with $Y \in \mathbb{R}^{L\times D}$, $U \in \mathbb{R}^{L\times k_{max}}$, and $A \in \mathbb{R}^{k_{max}\times D}$. However, when training over batches of size bs, we have a batched latent space $Y^b \in \mathbb{R}^{L\times bs}$ and so the shape of A needs to be different. Accordingly, for each batch, we keep the indices of the vector functions used and take the column of the same indices from A to form $A^b \in \mathbb{R}^{k_{max}\times bs}$. Accordingly, for each forward/backward pass, we train different columns of matrix A.

The strong formulation: For the strong formulation, nothing changes. The truncated SVD is performed over the batched latent space $Y^b = U\Sigma V^T$. It is important to note though that since the columns change depending on the batch, the values of the right singular vector (i.e. V^T) fluctuate a lot in training. However, since the same vector U is used for all the batches, the RRAE converges towards the right basis vectors in U. After training, to be sure that training is performed over the whole dataset, we perform the truncated SVD over the entire latent space (i.e. without batching) to get the corresponding reduced basis and coefficients that are used for interpolation (i.e. the ones found in training are disregarded). Our findings are that the RRAE converges towards a unique U, which is why interpolation is so successful throughout the paper.

Appendix E Entropy as a measure of uncertainty

In section 4.2, we used the entropy to measure the uncertainty of the model. Other measures such as the Fréchet inception distance (FID) and the inception score (IS) are conventionally used/trained for colored pictures. Accordingly, we chose to use a similar concept to evaluate the generation of uncolored pictures (MNIST numbers). Using the entropy was based on IS. In general, IS is a scalar that gives an idea of how good the generated pictures are by evaluating the diversity and the quality of the generated pictures. For the IS, the entropy is a measure of the quality of the generated pictures. It is computed using the pre-trained v3 model (on colored pictures). Accordingly, we created our own version of a classifier by using a traditional CNN trained on the training set (model and training parameters found here but for 937 batches of size 64 (or one epoch), and a `softmax` instead of a `logsoftmax` with the corresponding functions modified accordingly), which predicted, from the input images, the probability of being in each class (an output in $[0, 1]^{10}$) for each architecture. By the equation of the entropy which multiplies the probability of each class by the logarithm of that probability, an ideal prediction would simply be 1 for a class and 0 for every other class (hence 100% certainty). The further the predicted values of our classifier are from 0 and 1, the harder it is to classify the pictures in the interpolated test, which means that the generated pictures don't necessarily resemble the original training set. However, we don't evaluate the diversity of the generated pictures, since our interpolation process is between two pre-specified pictures (contrary to generative models where they could generate anything).