DATA-EFFICIENT TRAINING OF AUTOENCODERS FOR MILDLY NON-LINEAR PROBLEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Principal Component Analysis (PCA) provides reliable dimensionality reduction (DR) when data possesses linear properties even for small datasets. However, faced with data that exhibits non-linear behaviour, PCA cannot perform optimally as compared to non-linear DR methods such as AutoEncoders. By contrast, AutoEncoders typically require much larger datasets for training than PCA. This data requirement is a critical impediment in applications where samples are scarce and expensive to come by. One such area is nanophotonics component design where generating a single data point might involve running optimization methods that use computationally demanding solvers.

We propose Guided AutoEncoders (G-AE) of nearly arbitrary architecture which are standard AutoEncoders initialized using a numerically stable procedure to replicate PCA behaviour before training. Our results show this approach yields a marked reduction in the data size requirements for training the network along with gains in capturing non-linearity during dimensionality reduction and thus performing better than PCA alone.

1 INTRODUCTION

Principal Component Analysis (PCA) is a classical approach for dimensionality reduction [Pearson (1901)]. It provides excellent performance for linearly dependent data, however, its performance is compromised when the data exhibits non-linearity. PCA has been successfully used in numerous domains and provides respectable performance for dimensionality reduction. However, in many of these domains, the data exhibits some degree of non-linearity and therefore non-linear methods are, in principle, expected to yield superior results.

One such domain is a nanophotonic component design that requires computationally expensive Maxwell equation solvers to obtain data or optimize device performance. In [Melati et al. (2019)], it was shown that PCA can be effectively used to reveal a lower dimensional design subspace of well-performing designs of a vertical grating coupler. As a result, it was possible to investigate only the interesting part of the design space very efficiently and reveal meaningful patterns and trade-offs. However, it was noticed that the original design space was slightly curved, meaning that linear dimensionality reduction might be suboptimal, leaving room for improvement using non-linear methods.

AutoEncoders are standard neural networks with the distinct feature that their inputs and outputs are of the same dimension n, while in an intermediary layer a smaller number of neurons, d, is placed ($d < n$). The result is that the layers preceding the bottleneck act as an *encoder* and the layers proceeding it are the *decoder* for the reduced representation in the bottleneck. As such, the size of the bottleneck controls the degree of dimensionality reduction achieved. While Autoencoders are well-suited for non-linear dimensionality reduction, their typical data requirements far exceed those of PCA to be competitive in problems with scarce data.

Typically, AutoEncoders are trained with random initialization of weights. Yet, it is well known that the behaviour of PCA with p components can be modeled using an AutoEncoder with a bottleneck of p neurons and linear activation functions throughout the network [Baldi & Hornik (1989)]. In this paper, we explore the use of PCA to initialize the weights of an AutoEncoder in a numerically stable way and to enable the AutoEncoder to benefit from the linearly optimized solution. Specifically,

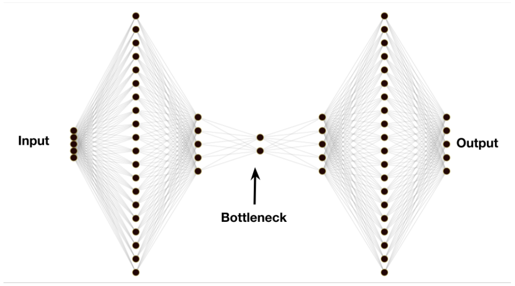

Figure 1: AutoEncoder architecture used in experiments

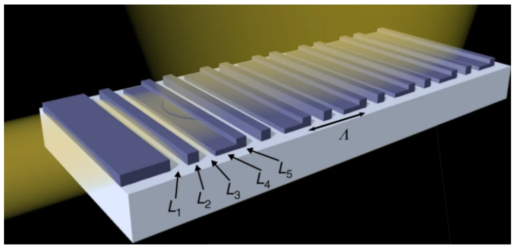

Figure 2: Schematic representation of grating coupler structure (reproduced with permission from [Melati et al. (2019)]).

we propose Guided AutoEncoders (G-AE) of nearly-arbitrary architecture, which, given a PCA dimensionality reduction to d dimensions, are initialized with PCA weights with linear activation functions and a bottleneck of size d thus replicating PCA behaviour. The activation function is then changed to a non-linear one and the G-AE is trained in the usual way, hence enabling the network to capture non-linear behavior.

There have been efforts to use PCA initialization of AutoEncoders such as [Seuret et al. (2017)] however, these studies differ from our study in two important ways: (i) they focus predominantly on images which present the AutoEncoder much larger vectors than we deal with here, and (ii) their chief concern is not with drastically reducing the size of the training set.

Evaluated on the nanophotonic component design problem, our results show a marked improvement in the data size requirements for training the network along with clear gains in capturing the non-linearity of the data during dimensional reduction.

2 METHODOLOGY

We assume that the architecture of the G-AE is vase-shaped as in Figure [1], where first the data is expanded in its dimensionality and then reduced back to the original dimension n before being reduced to the bottleneck dimension d. Although not necessary, it makes the weights initialization process easier while allowing a significant degree of randomness. The n-D to n-D expand-contract parts can be effectively generated by a series of random projection matrices (full rank matrices with all singular values being 1) such that the entire expand-contract part becomes a full rank projection matrix itself. The weights of the bottleneck are then calculated from PCA obtained from data. This allows a numerically stable randomized initialization of G-AE of arbitrary depth that mimicks PCA.

The design of the experiment is intended to replicate a low-data regime in practice. In particular, in such a setting, one is severely constrained in the data to be set aside for testing and cross validation. Having good initial weights of the AutoEncoder plays a crucial role in giving it a significant head-start since this initialization potentially covers the action of many epochs of training that would otherwise be needed to get to that point—epochs that in a low-data regime run a serious risk of overfitting. Because there is an element of randomness in the choice of initial weights, we try different initializations of the AutoEncoder and select the one that appears to produce the best result.

In our experiments we split the data into 80% for training 10% for validation and 10% for testing, where the validation set is used to identify when to stop the training process and the test set is used to compare performance across all models including PCA. The total data amounts considered to be available is only 50 samples. In our case this resulted in 40 data samples used for training, 5 for validation and 5 for testing. As well, to corroborate our entire experimental setup that can be sensitive to data splits due to the low data regime, we set aside 100 designs as an oracle test set. The experiments are further repeated 100 times creating different data splits to measure accurate statistics of the results.

The complete set of data we have for the vertical grating coupler consists of 540 good designs. These were obtained from computationally expensive simulations-based optimizations and selected from a set consisting of more than 30,000 candidate designs based on an optical performance criterion.

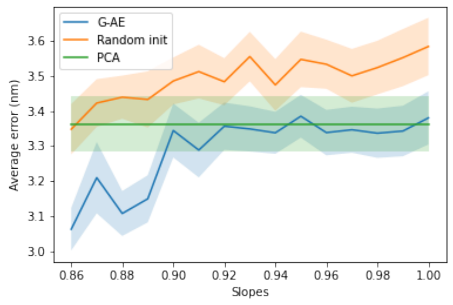

Figure 3: Error for PCA, Random Initialization and PCA Initilization on Test Set

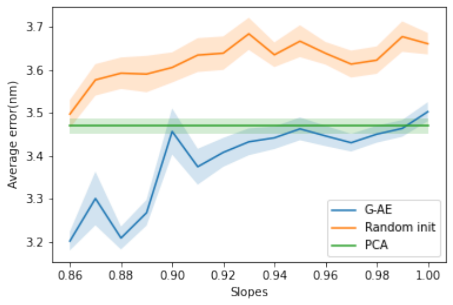

Figure 4: Error for PCA, Random Initialization and PCA Initilization on Oracle Set

The designs are characterized by five segment values (L_1 to L_5) which are the parameters to the design problem (see Figure 2). Each of these segment values is a real number that forms an element of the input vector. In [Melati et al. (2019)], it was shown that 2 principal components were enough to capture most of the good design subspace.

Using the Adam optimizer, and the LeakyReLU activation function for all the layers, we set the negative slope progressively from 1 (linear) to 0.86 (mildly non-linear) with a gradation of 0.01 and trained the network for each slope, once with PCA initialization and once without. We instituted early stopping in the training based on the validation set to avoid overfitting. The Euclidean distance between the input and output vectors is used as the loss function, which is an implicit objective of PCA as well.

The AutoEncoder we experimented with had 8 layers including the input and output layers and, in sequence from the input layer to the output layer, they had the following number of neurons: 5-20-5-2-5-20-5 (see Figure 1). Note that the bottleneck for our experiment is 2 which allows us to compare with PCA with 2 principal components as in [Melati et al. (2019)]. The choice of this architecture is somewhat arbitrary and was not optimized for the problem.

Once the training is complete, and based on its performance on the test set, we choose the best model from the various initializations we deployed. As mentioned earlier, we also confirm that this procedure is statistically meaningful by measuring the performance of all model types (AE, G-AE, PCA) on a much larger oracle test set.

As well, we experimented with the Parametric ReLU (PRELU) activation function. The key difference with our initial experiment is that in our experiments with LeakyReLU we probed negative slopes between 0.86 and 1 and for each training cycle every node had exactly the same slope. In the case of PRELU, each node is allowed to vary its slope independently as a trainable parameter. To reflect the PCA-based initialization, all initial slopes of PRELU were initialized to 1.

3 Results

Figure 3 shows the performance of the model on the test set which is used to assess the quality of the model. We notice that G-AE outperforms randomly-initialized AutoEncoders and PCA for a variety of slope values. Yet, randomly initialized AEs perform comparably or worse than PCA. As expected, when we look at the performance on the oracle set (Figure 4) we see the same trends as observed in the Figure 3 while having tighter error bounds.

Table 1 shows the results of our experiments with PReLU. Although randomly initialized AE with PReLU activation function performs comparably to PCA, the G-AE outperfoms both.

Table 1: PReLU

Model	Oracle Set Error	Test Set Error
PCA-initialized PReLU (G-AE)	3.27 ± 0.02	3.21 ± 0.06
PCA	3.48 ± 0.01	3.62 ± 0.07
Randomly Initialized PReLU AE	3.42 ± 0.1	3.35 ± 0.09

4 Conclusion and Future Work

In this work we demonstrated that AutoEncoders, with proper initialization, can offer a viable solution for dimensionality reduction even in a regime of limited data. Our results show that, on small but sufficient datasets, the use of PCA to initialize a LeakyReLU and PReLU AutoEncoders in a numerically stable way yields results that are superior to randomly initialized AutoEncoders, and even PCA alone. These results are encouraging in domains where only PCA has been used to reduce the dimensionality due to very limited datasets available.

In our experiments the range of slopes was limited and potentially better models can be achieved by expanding on that range. On the other hand, drastic change in slope from 1 (linear) can also be detrimental and the resulting model will become highly non-linear and might loose the benefits of PCA-based initialization. Studying the trade-offs of those initializations and suggesting smooth learning schemes that follow continuation methods is one avenue for future work.

While in this work we evaluated the proposed method directly on real data, it would be instructive to construct a synthetic study to compare the performance of PCA versus AutoEncoders on datasets of different size, level of non-linearity and noise. This can be, for instance, data generated from a paraboloid function and other shapes.

Finally, initialization of the network weights in the proposed fashion might offer a degree of stability and robustness similar to the proposals that address adversarial example issues in neural networks and enforce general smoothness [Cisse et al. (2017); Anil et al. (2019)].

References

Cem Anil, James Lucas, and Roger Grosse. Sorting out lipschitz function approximation. In *International Conference on Machine Learning*, pp. 291–301. PMLR, 2019.

Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning*, pp. 854–863. PMLR, 2017.

Daniele Melati, Yuri Grinberg, Mohsen Kamandar Dezfouli, Siegfried Janz, Pavel Cheben, Jens H Schmid, Alejandro Sánchez-Postigo, and Dan-Xia Xu. Mapping the global design space of nanophotonic components using machine learning pattern recognition. *Nature communications*, 10(1):1–9, 2019.

Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

M. Seuret, M. Alberti, M. Liwicki, and R. Ingold. Pca-initialized deep neural networks applied to document image analysis. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pp. 877–882, 2017. doi: 10.1109/ICDAR.2017.148.