Enhancing Autoencoder Performance: Leveraging Gain and Relative Gain Array (RGA) Through Innovative Functions for Training Optimization and Pruning

Rafael Henrique Martello[a], Marcelo Farenzena[a], Jorge Otávio Trierweiler[a]

[a]*Group of Intensification, Modeling, Simulation, Control, and Optimization of Process, GIMSCOP, Federal University of Rio Grande do Sul, Chemical Engineering Department, St. Engenheiro Luiz Englert - 12.204, Porto Alegre, 90040-040, RS, Brazil*

Preprint submitted to Computers in Industry *February 20, 2024*

Abstract

High-dimensional data poses significant challenges in modern data science, necessitating innovative approaches for effective processing and extracting meaningful insights. Autoencoders, a type of neural network, offer a promising solution by leveraging the latent space representation to extract valuable information from complex datasets. However, optimizing autoencoders remains challenging due to the high dimensionality, nonlinearity, and potential for overfitting or underfitting. To address these issues, this study introduces two novel objective functions based on the concepts of the Gain Matrix (G) and Relative Gain Array (RGA), along with pruning techniques, to enhance autoencoder training and performance. A case study of a simulated propene production plant is used to demonstrate the effectiveness of the proposed approach. The results reveal that autoencoders with novel objective functions outperform traditional methods, such as PCA and conventional autoencoders with mean square error (MSE) objective functions, leading to gains of up to 40% in the explained variance when compared to a PCA and up to 20% when compared to usual autoencoders. The findings underscore the potential of autoencoders as powerful tools for high-dimensional data analysis. By incorporating multivariable control concepts and refining objective functions, autoencoders offer new opportunities to address complex data challenges and advance data science in diverse real-world scenarios.

Keywords: Unsupervised Learning, Multivariable Control, Data Compression, Soft-Sensors

1. Introduction

High-dimensional data is a pervasive challenge across various domains of modern data science. The curse of dimensionality looms over the analysis of such data, requiring innovative approaches for effective processing and extraction of meaningful insights [1]. In this context, autoencoders emerge as a promising solution, offering a powerful framework to tackle the complexities of high-dimensional data. By leveraging the latent space representation generated by autoencoders, valuable information can be extracted, enabling improved predictions, classifications, and data-driven decision-making [2].

Autoencoders are a type of neural network that learns in an unsupervised way, and they were first introduced as a model that is trained to reconstruct its input instead of relying on labeled data. It relies on the relationship between the input variables [3, 4]. They have many applications, including assisting in classifications, regressions, and performing unsupervised tasks in high-dimensional spaces, such as data compression, denoising, dimensionality reduction, and feature extraction [5, 6]. As shown in [3, 7], there are numerous applications of AEs and their variants, focused on sound [8, 9] to image and video data [10, 11].

In the realm of data-driven decision-making, soft-sensors have emerged as valuable tools for capturing real-time information from complex systems. These virtual sensors, derived from machine learning techniques, provide a means to extract meaningful and actionable insights from raw data streams [12, 13]. Previous studies have explored soft-sensors based on neural networks [14] using a PCA followed by a supervised adaptive network-based fuzzy inference system (ANFIS) for the prediction of carbon and nitrogen removal in wastewater treatment, [15] a supervised feed-forward artificial neural network to provide estimates of the polyethylene terephthalate (PET) viscosity in production and [16] used self-organizing map (SOM) and supervised feed-forward backpropagation neural networks to estimate parameters of fermentation processes with recombinant *E coli*. However, what sets autoencoders (AEs) apart is their potential to integrate predictions with the specific structural characteristics of a neural network, offering unique advantages.

Autoencoders offer many advantages, making them an ideal choice for implementing soft-sensors in various applications. Firstly, their ability to perform unsupervised learning allows soft sensors to extract valuable insights from unlabeled data streams, eliminating the need for costly and time-consuming data annotation [17]. Secondly, autoencoders excel at dimensionality reduction, enabling soft sensors to handle high-dimensional data efficiently and effectively. This reduction enhances computational efficiency and simplifies data visualization and interpretation [18, 19]. Thirdly, by learning compressed and robust representations of the input data, autoencoders facilitate extracting relevant features, making soft sensors more accurate and resilient to noise and variations [17].

Moreover, the flexibility of autoencoders in handling different types of data, such as images, time series, and text, extends soft-sensors applicability to diverse industries, including manufacturing, healthcare, and finance [20]. In the context of industrial processes, the autoencoders capacity to handle

vast amounts of data is particularly appealing. In complex manufacturing settings, autoencoders process a plethora of sensor data efficiently, enabling real-time monitoring and control of various processes [21].

While autoencoders are helpful tools, optimizing neural network training for autoencoder-based unsupervised learning can be challenging due to the high dimensionality and complexity of the input data, the nonlinear and nonconvex nature of the optimization problem, and the potential for overfitting or underfitting [4]. A significant challenge of unsupervised models like autoencoders is the absence of a clear, straightforward objective function to cluster data points based on their semanticity [22]. Additionally, certain objective functions can be overly simplistic, and when associated with the network training with a high number of parameters, it can lead to issues like the optimization of abnormal points and the generation of chaotic latent spaces [23]

To overcome these limitations, we need a way to gain more control over the training and evolution of this unsupervised method aiming to limit the chaos in latent spaces. The latent space can be input for constructing models that predict data characteristics. In this paper, we propose novel objective functions using concepts explored by Bristol [24] over multi-input, multi-output problems in industrial control of gain matrix (G) and relative gain array (RGA) and pruning functions using these matrices for each layer to improve the training of neural networks for autoencoder-based unsupervised learning. The AE's compressed latent space will be used as input for the calibration of linear predictive models as soft sensors for a case study of a simulation of a propene production plant [25] consisting of three distillation columns and aims to distillate a load of liquefied petroleum gas.

2. Methodology

2.1. Autoencoder

Neural networks draw inspiration from the human brains neural structure [26]. It consists of interconnected nodes, called neurons, organized into layers. Each neuron is an essential processing unit that receives input data and performs a simple computation. It multiplies each input by a corresponding weight, sums them up with a bias term, and then applies an activation function to determine its output. This output is then passed on to other neurons in the network. Through a process called training, neural networks adjust their weights and biases to learn patterns and relationships within the data, enabling them to perform tasks such as image recognition, natural

language processing, and decision-making with remarkable accuracy. During training, neural networks utilize objective functions, also known as loss functions, to measure the difference between predicted outputs and the actual targets. The neural network fine-tunes its parameters by minimizing the value of the objective function through optimization algorithms like gradient descent. It improves its ability to make accurate predictions on unseen data [27].

As a class of artificial neural networks, autoencoders play a crucial role in unsupervised learning and dimensionality reduction tasks [28]. The architecture of an autoencoder comprises an input layer and one or more hidden layers constituting the encoder and decoder segments, as illustrated in Figure 1. These hidden layers generate an encoded representation of the input data, subsequently reconstructed back to its original form in the output layer.

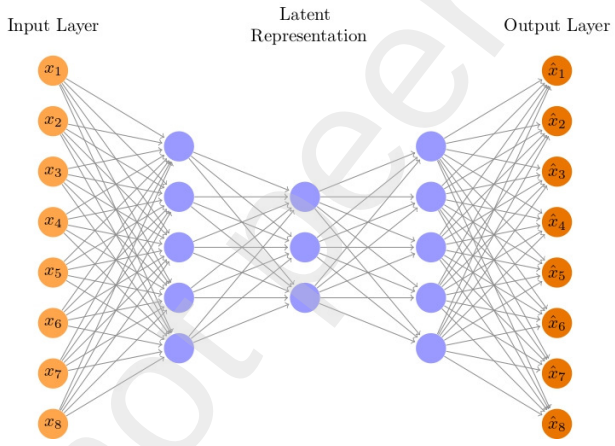

Figure 1: Autoencoder Structure

Autoencoders, a type of neural network, possess a captivating structure where neurons interact to minimize reconstruction errors between input and output during training [20]. This enables them to recreate input data from encoded representations, facilitating efficient data compression and representation of critical features [29]. Autoencoders find extensive applications in data compression, anomaly detection, image denoising, and computer vision tasks [30]. In industrial processes, they efficiently handle large datasets for real-time monitoring and control [21].

Despite their advantages, autoencoders face challenges in generating organized structures in the latent space due to non-specific objective functions and a lack of good regularizations [23]

Nevertheless, despite these challenges, autoencoders remain powerful tools with vast potential across various applications in advancing machine learning and artificial intelligence. Ongoing research continues to address and improve upon these limitations. As this kind of model continues to be explored in various fields, there is a growing interest in developing methods of optimizing the training of these neural networks.

2.2. Gain Matrix (G) and Relative Gain Array(RGA)

To improve the training of AE models, multivariable gain matrix (G) and relative gain matrix (RGA) concepts were proposed here to formulate two modified goal functions for AE training. The first concept here introduced is the Gain matrix (G) given by:

$$G = \begin{bmatrix} \frac{\partial \hat{x}_1}{\partial x_1} & \cdots & \frac{\partial \hat{x}_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{x}_m}{\partial x_1} & \cdots & \frac{\partial \hat{x}_m}{\partial x_n} \end{bmatrix} \tag{1}$$

where $\hat{x}_m$ is the output m and $\mathrm{x_n}$ is the input n.

In industrial control, the gain matrix refers to a mathematical representation of the relationship between the inputs and outputs of a control system. It is commonly used in multivariable control systems with multiple inputs and outputs. Each element of the gain matrix represents the gain or sensitivity of the corresponding output to a unit change in a particular input [31].

The second concept is the relative gain array (RGA) proposed by Bristol (1966), which corresponds to a matrix used in multivariable control to analyze and understand the interactions between different inputs and outputs. It provides insights into the relative importance of each input to each output regarding the systems overall performance. It quantifies the impact of a change in input m on output n while holding all others. It can be calculated using (2) for a square matrix or be generalized to a l x m non-square matrix using (3).

$$RGA = G \odot (G^{-1})^T \tag{2}$$

$$RGA = G \odot (G^{\dagger})^T \tag{3}$$

where $\odot$ is the elementwise Hadamard product of the matrices of gain (G), and † is the Moore–Penrose inverse.

2.3. Objective functions and Pruning techniques

2.3.1. Objective functions

Generative models, such as autoencoders, often require an objective function to govern the relationship between the input data and its reconstructed counterpart. The commonly employed objective function is MSE (mean squared error). However, these constraints can be overly simplistic. Although MSE is straightforward to solve, it tends to prioritize outliers when they exist in the data. This phenomenon occurs because MSE assigns greater weight to outliers, diverting the models optimization efforts towards these atypical points [23]

The mean square error (equation 4) was used as a base equation for the new objective functions.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i + \hat{x}_i)^2 \tag{4}$$

where x_i is the input value of the variables and $\hat{x}_i$ is the reconstructed output n is the number of features or variables of the model.

The methodology of using the gain matrix and RGA as part of the objective function can potentially improve the quality of unsupervised models. The first objective function created uses the gain matrix (G) to incorporate the interaction between each input-output into the cost function as a second term, together with mean square error (equation 5).

$$f_{gain} = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{n}\sum_{i=1}^{n}(x_i + \hat{x}_i)^2 + \lambda(\parallel G - I \parallel_2)^2\right) \tag{5}$$

where I is an identity matrix with the same dimensions as G, which is obtained by the *autograd* function in PyTorch, and m is the batch size or number of observations processed at a time.

Based on widely studied regularization functions such as Lasso (L1) and Ridge (L2) described in [32], the function was developed using a lambda (λ) to define the relative importance of the gain matrix, with the regularization value being the norm of the matrix subtracting an identity matrix of equivalent size, which has the function of correcting the value to an ideal system (no interaction between the channels), plus the widely used error function MSE.

Likewise, for the RGA, equation 6 was developed:

$$f_{RGA} = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{n}\sum_{i=1}^{n}(x_i + \hat{x}_i)^2 + \lambda(\parallel RGA - I \parallel_2)^2\right) \tag{6}$$

The RGA was used in the objective function, resembling equation 5 using a lambda value (λ) and the squared Euclidean norm of the difference between RGA and an identity matrix. where I is an identity matrix with the same dimensions as RGA, and m is the batch size or number of observations processed at a time.

The selection of the Euclidean norm to penalize differences between matrices in an autoencoder is rooted in the objective of highlighting and amplifying noteworthy individual errors. This decision is motivated by the intention to assign greater significance to substantial deviations between the real and ideal coupling values, thereby possibly promoting a selective approach to penalization. By choosing the Euclidean norm, the attention is redirected to a more localized evaluation of errors, potentially facilitating a more precise strategy for addressing specific coupling patterns within the autoencoder architecture.

2.4. Pruning functions

For the pruning functions, gain matrices and RGAs were calculated layer by layer according to Figure 2:

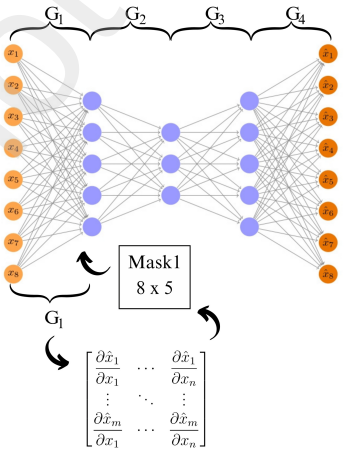

Figure 2: Diagram of Gain Pruning

The calculated matrices were used in a pruning function that iterates over the list of weights of the layers and creates a mask that sets values lower

than a pre-established threshold of 0.01 to 0 throughout the training, based on this and to avoid very sudden cuts, we applied these masks on the layers every 25 epochs in networks that are trained for a total of 1000 epochs.

2.5. Soft-Sensor methodology

The soft-sensor development methodology employed in this study embraces a comprehensive data-driven approach, integrating advanced process control and machine learning techniques. Data preparation and analysis are conducted using Python v.3.9.16, with specific libraries tailored for diverse tasks to initiate the process. Pandas v.2.0.1 and NumPy v.1.21.5 are harnessed for data manipulation and feature engineering. PyTorch v.2.0.0 and Scikit-learn v.1.2.2 are employed to create and train predictive models. These libraries leverage their capabilities for modeling and optimizing predictions based on autoencoders.

2.6. Data processing

Given the temporal nature of the data organized in time series format, the dataset was partitioned in a 60:20:20 ratio using a methodology of modified systematic sampling that was guaranteed to select the samples with minimum and maximum values of the output variable for the training subset, using the *k-rank* method [33]. This was crucial as all samples must be within the range for the black-box models so that no extrapolation would be needed [34].

With the split data, a normalization was applied using a Standard Scaler method. This method rescaled the distribution of values, ensuring that the mean of observed values was centered at 0, and the standard deviation was set to 1.

2.7. Soft-sensor Structure

The autoencoder hyperparameters were defined through a Grid Search of hyperparameters (Table 1) to obtain the best model and capture the behavior of data for each soft-sensor.

We aim to optimize the hyperparameters of our model effectively using grid search, with a batch size fixed in 512. Our focus lies in achieving the best possible settings for the autoencoder + linear model (Figure 3) by evaluating its explained variance. To accomplish this, the autoencoder must first learn to reconstruct the data accurately and, subsequently, utilize the encoded space to make precise estimations. Through this process, we can determine the optimal hyperparameters that enhance our models performance and robustness.

Table 1: Hyperparameter search space

Parameter	Search Space
Number of layers	3 or 5 layers
Activation function	Sigmoid and ReLU
Size of hidden layers	5, 6, 7 and 8 neurons
Size of latent space	4, 5, 6, 7 and 8 neurons
Learning rate	$1x10^{-4}$, $1x10^{-3}$ and $1x10^{-2}$
Lambda (λ)	1, 0.1 and $1x10^{-2}$

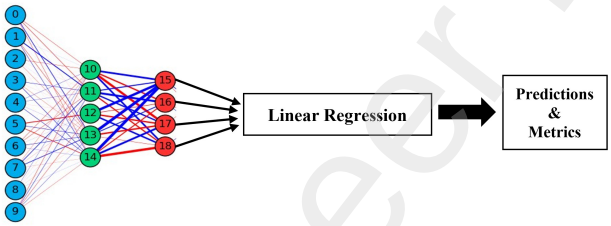

Figure 3: Structure of Soft-sensor

The grid search not only allowed us to find the best hyperparameters for our soft-sensor model but also provided valuable insights into the behavior of the autoencoder under different configurations. By systematically evaluating the models performance with various hyperparameter combinations, we better understood how each parameter impacts the models ability to reconstruct data and make accurate predictions. This knowledge proved invaluable in fine-tuning the soft sensor, ultimately leading to a highly efficient and reliable system for detecting the characteristics of our case study.

subsectionCase study

This study utilizes a simulated dataset from a propene production plant [25]. The process consists of a series of three distillation columns that will be respectively called T-01, T-02, and T-03 and is designed to separate liquefied petroleum gas into a stream containing 99.6% propene as the final product.

In the first column, it is important to pass on as little impurities as possible to the second column, which consists of removing the heavies (longer chain hydrocarbons). Therefore, the soft-sensor target of T-01 corresponds to the amount of impurities in the top stream. The inference of this variable is of great importance for the control of the process, since this impurity must be

kept to a minimum, considering the thermodynamic and economic conditions of the plant. For column T-02, the variable of most significant interest to infer is the propylene output at the top of the column, representing its waste. In T-03, it is important to control the specification of the propylene concentration at the top of the column. For this, it is necessary to infer the concentration of propane, a component that makes the propylene stream impure. An Aspen Plus® representation of the unit can be seen in Figure 4.

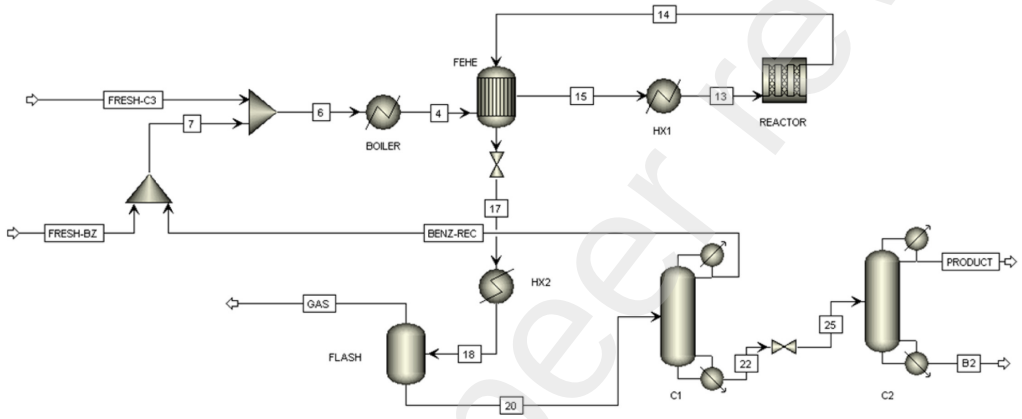

Figure 4: Aspen Plus® model of the unit adapted from [25]

Data is segregated by column, where column T-01 has 871 process operation data points with available measurements of 10 different instruments, column T-02 has 407038 entries with 11 instruments, and column T-03 has 1764 data points with 19 instruments.

3. Results

A technique called Grid Search was used to find the ideal hyperparameters for the neural networks used in this study. Grid Search is a thorough search method that allows exploring various possible combinations of hyperparameters to determine the optimal configuration of the model. In Table 2, presented below, the best hyperparameters for each column are presented as obtained in the optimization.

After adjusting the autoencoders using the optimized hyperparameters, they were used in soft-sensors with linear regression to detect the characteristics addressed in the case study. The adjusted autoencoders made it possible to

Table 2: Selected hyperparameters

Hyperparameter	Columns		
	T-01	T-02	T-03
Layers	5	5	5
Hidden Neurons	5	7	10
Encoded size	4	3	4
Activation	Sigmoid	Sigmoid	Sigmoid
Learning rate	$1x10^{-2}$	$1x10^{-2}$	0.1
Lambda	0.1	0.1	$1x10^{-2}$

compress the original data information into lower-dimensional representations, preserving the most relevant characteristics. To evaluate the performance of the adjusted autoencoders, Tables 3 was generated, which present the explained variances by the different models in comparison. The results of the models developed in this study were compared with a PCA (Principal Component Analysis) and a conventional autoencoder that uses the same configuration with an MSE objective function.

After analyzing the results presented in Table 3, we further validated the significance of the differences between the soft-sensor models using the Fisher Least Significant Difference (LSD) method. The Fisher LSD method is a statistical test commonly employed in multiple comparisons to determine if there are significant differences between the means of different groups. Our study applied it to verify the differences in explained variances obtained by the various soft-sensor models across columns T-01, T-02, and T-03.

The Fisher LSD test revealed statistically significant differences ($p < 0.05$) between several pairs of soft-sensor models, providing additional evidence that the proposed novel objective functions, which integrate the gain matrix and relative gain array, substantially impact the models performance.

Comparing the results, it is observed that the autoencoders with novel functions obtained a higher explained variance concerning the reference methods. This indicates that the soft sensors developed in this study could capture the relevant information in the input data more accurately. Works by authors such as [35, 36, 37] show that autoencoders, in many applications, tend to outperform linear methods such as PCA, being able to detect subtle anomalies in data. This more remarkable ability to explain variances indicates

Table 3: Explained variance for each soft sensor in columns T-01, T-02 and T-03.

Model	Columns		
	T-01	T-02	T-03
PCA	0.66 ± 0.0^{a}	0.03 ± 0.0^{ab}	0.46 ± 0.0^{a}
AE	0.82 ± 0.0^{bc}	0.34 ± 0.21^{cd}	0.65 ± 0.02^{bc}
GAE	0.89 ± 0.08^{bc}	0.55 ± 0.04^{de}	0.64 ± 0.09^{bc}
RGAE	0.79 ± 0.02^{ab}	0.33 ± 0.18^{cd}	0.57 ± 0.08^{ab}
AE PG	0.79 ± 0.07^{ab}	0.59 ± 0.02^{e}	0.79 ± 0.07^{cd}
AE PRGA	0.90 ± 0.06^{bc}	0.02 ± 0.0^{a}	0.59 ± 0.04^{ab}
GAE PG	0.81 ± 0.03^{bc}	0.46 ± 0.14^{cde}	0.67 ± 0.08^{bc}
GAE PRGA	0.81 ± 0.12^{bc}	0.26 ± 0.04^{bc}	0.63 ± 0.14^{b}
RGAE PG	0.86 ± 0.07^{bc}	0.46 ± 0.14^{cde}	0.88 ± 0.04^{d}
RGAE PRGA	0.93 ± 0.04^{c}	0.32 ± 0.01^{c}	0.63 ± 0.08^{b}

[a] first group; [b] second group; [c] third group; [d] fourth group; [e] fifth group in Fisher's Test.

that the adjusted autoencoders perform better in the data compression and reconstruction task than the PCA and the conventional autoencoder.

Moreover, by observing the behavior of the network weights during training and at the end of it (Figure 5) we can verify simpler relationships between neurons, showing a potential to reduce the interaction between channels and consequently reduce the number of variables adjusted to each training epoch.

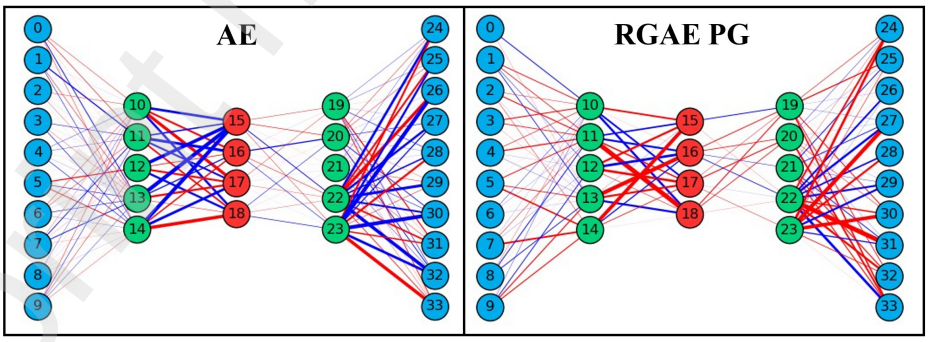

Figure 5: Visual comparison of T-01 column weights for a normal autoencoder (AE) and an autoencoder that uses the functions developed in the article (RGAE PG) after training

These results reinforce the importance of the specific functions developed in this work to adjust the autoencoders more appropriately to the problem in question, thus improving the quality of the latent representations obtained and, consequently, the ability to explain data variances.

4. Conclusions

In conclusion, this study demonstrates the effectiveness of autoencoders in addressing the challenges of high-dimensional data analysis. By leveraging the latent space representation generated by autoencoders, valuable insights can be extracted from complex datasets, leading to improved predictions and data-driven decision-making. Incorporating novel objective functions based on Gain Matrix (G) and Relative Gain Array (RGA) concepts, along with pruning techniques, enhances the training and performance of autoencoder models.

The case study results, focused on a simulated propene production plant, highlight the superiority of autoencoders with novel objective functions compared to traditional methods such as PCA and conventional autoencoders with the aim of MSE functions. These autoencoders achieve higher explained variance and accurately capture the relevant information in the input data, surpassing linear methods and enabling the detection of subtle anomalies. The observed behavior of network weights during training further indicates a potential reduction in channel interactions, enhancing the models efficiency and simplifying relationships between neurons.

Overall, this research underscores the potential of autoencoders as powerful tools for high-dimensional data analysis and soft-sensors. By refining the objective functions and incorporating multivariable control concepts, autoencoders can provide better control over the training process and improve the quality of latent representations. Future work may explore diverse datasets and extend the application of autoencoders to various real-world scenarios, offering new opportunities for addressing complex data challenges and advancing the field of data science.

5. Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author(s) used ChatGPT in order to improve readability. After using this tool/service, the author(s) reviewed

and edited the content as needed and take(s) full responsibility for the content of the publication.

References

[1] M. Verleysen, D. François, The curse of dimensionality in data mining and time series prediction, in: J. Cabestany, A. Prieto, F. Sandoval (Eds.), Computational Intelligence and Bioinspired Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 758–770.

[2] W. Wang, Y. Huang, Y. Wang, L. Wang, Generalized autoencoder: A neural network framework for dimensionality reduction, IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (2014) 496–503doi:10.1109/CVPRW.2014.79.

[3] D. Charte, F. Charte, S. Garcia, M. J. del Jesus, F. Herrera, A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines, Information Fusion 44 (2018) 78–96. doi:10.1016/J.INFFUS.2017.12.007.

[4] D. Bank, N. Koenigstein, R. Giryes, Autoencoders, arXiv e-prints 2 (2020). doi:10.48550/arXiv.2003.05991.

[5] X. Wu, Q. Cheng, Fractal autoencoders for feature selection, 35th AAAI Conference on Artificial Intelligence, AAAI 2021 12A (2020) 10370–10378. doi:10.48550/arxiv.2010.09430.

[6] R. Clarke, H. W. Ressom, A. Wang, J. Xuan, M. C. Liu, E. A. Gehan, Y. Wang, The properties of high-dimensional data spaces: implications for exploring gene and protein expression data, Nature Reviews Cancer 2008 8 (2008) 37–49. doi:10.1038/nrc2294.

[7] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (2006) 504–507. doi:10.1126/science.1127647.

[8] J. Deng, Z. Zhang, F. Eyben, B. Schuller, Autoencoder-based unsupervised domain adaptation for speech emotion recognition, IEEE Signal Processing Letters 21 (2014) 1068–1072. doi:10.1109/LSP.2014.2324759.

[9] I. P. Yamshchikov, A. Tikhonov, Music generation with variational recurrent autoencoder supported by history, SN Applied Sciences 2 (2017). doi:10.1007/s42452-020-03715-w.

[10] C. Doersch, Tutorial on variational autoencoders, ArXiv e-prints stat.ML (2016). doi:10.48550/arXiv.1606.05908.

[11] J. Almotiri, K. Elleithy, A. Elleithy, Comparison of autoencoder and principal component analysis followed by neural network for e-learning using handwritten recognition, 2017 IEEE Long Island Systems, Applications and Technology Conference (2017). doi:10.1109/LISAT.2017.8001963.

[12] C. Shang, F. Yang, D. Huang, W. Lyu, Data-driven soft sensor development based on deep learning technique, Journal of Process Control 24 (2014) 223–233. doi:10.1016/J.JPROCONT.2014.01.012.

[13] X. Yuan, Y. Wang, C. Yang, W. Gui, Stacked isomorphic autoencoder based soft analyzer and its application to sulfur recovery unit, Information Sciences 534 (2020) 72–84. doi:10.1016/J.INS.2020.03.018.

[14] G. Civelekoglu, A. Perendeci, N. O. Yigit, M. Kitis, Modeling carbon and nitrogen removal in an industrial wastewater treatment plant using an adaptive network-based fuzzy inference system, CLEAN - Soil, Air, Water 35 (2007) 617–625. doi:10.1002/CLEN.200700076.

[15] J. Gonzaga, L. Meleiro, C. Kiang, R. M. Filho, Ann-based soft-sensor for real-time process monitoring and control of an industrial polymerization process, Computers & Chemical Engineering 33 (2009) 43–49. doi:10.1016/j.compchemeng.2008.05.019.

[16] K.-I. Lee, Y.-S. Yim, S.-W. Chung, J. Wei, J. I. Rhee, Application of artificial neural networks to the analysis of two-dimensional fluorescence spectra in recombinant e coli fermentation processes, Journal of Chemical Technology & Biotechnology 80 (2005) 1036–1045. doi:10.1002/JCTB.1281.

[17] S. M. Miraftabzadeh, M. Longo, M. Brenna, Knowledge extraction from pv power generation with deep learning autoencoder and clustering-based algorithms, IEEE Access (2023). doi:10.1109/ACCESS.2023.3292516.

[18] O. Kuchaiev, B. Ginsburg, Training deep autoencoders for collaborative filtering, ArXiv e-prints 3 (2017). doi:10.48550/arXiv.1708.01715.

[19] J. Bharadiya, J. P. Bharadiya, A tutorial on principal component analysis for dimensionality reduction in machine learning, Article in International Journal of Innovative Research in Science Engineering and Technology 8 (2023). doi:10.5281/zenodo.8002436.

[20] S. Chen, W. Guo, Auto-encoders in deep learning - a review with new perspectives, Mathematics 11 (2023) 1777. doi:10.3390/MATH11081777.

[21] A. L. Alfeo, M. G. Cimino, G. Manco, E. Ritacco, G. Vaglini, Using an autoencoder in the design of an anomaly detector for smart manufacturing, Pattern Recognition Letters 136 (2020) 272–278. doi:10.1016/J.PATREC.2020.06.008.

[22] N. Mrabah, N. M. Khan, R. Ksantini, Deep clustering with a dynamic autoencoder smart vision view project landmine detection view project deep clustering with a dynamic autoencoder, ArXiv e-prints 5 (2019). doi:10.48550/arXiv.1901.07752.

[23] Q. Zhu, H. Wang, R. Zhang, Wavelet loss function for auto-encoder, IEEE Access 9 (2021) 27101–27108. doi:10.1109/ACCESS.2021.3058604.

[24] E. H. Bristol, On a new measure of interaction for multivariable process control, IEEE Transactions on Automatic Control (1966). doi:10.1109/TAC.1966.1098266.

[25] E. S. Schultz, J. O. Trierweiler, M. Farenzena, The importance of nominal operating point selection in self-optimizing control, Industrial and Engineering Chemistry Research 55 (2016) 7381–7393. doi:10.1021/acs.iecr.5b02044.

[26] R. B. Gharbi, G. A. Mansoori, An introduction to artificial intelligence applications in petroleum exploration and production, Journal of Petroleum Science and Engineering 49 (2005) 93–96. doi:10.1016/J.PETROL.2005.09.001.

[27] J. Zou, Y. Han, S. S. So, Overview of Artificial Neural Networks, Humana Press, Totowa, NJ, 2009, pp. 14–22. doi:10.1007/978-1-60327-101-1_2.

[28] M. Sewak, S. K. Sahay, H. Rathore, An overview of deep learning architecture of deep neural networks and autoencoders, Journal of Computational and Theoretical Nanoscience 17 (2020) 182–188. doi:10.1166/JCTN.2020.8648.

[29] E. Trunz, M. Weinmann, S. Merzbach, R. Klein, Efficient structuring of the latent space for controllable data reconstruction and compression, Graphics and Visual Computing 7 (2022) 200059. doi:10.1016/J.GVC.2022.200059.

[30] P. Li, Y. Pei, J. Li, A comprehensive survey on design and application of autoencoder in deep learning, Applied Soft Computing 138 (2023) 110176. doi:10.1016/J.ASOC.2023.110176.

[31] S. Skogestad, I. Postlethwaite, Multivariable Feedback Control Second Edition: Analysis and Design, n.2 Edition, Wiley & Sons, Incorporated, John, 2005.

[32] L. E. Melkumova, S. Y. Shatskikh, Comparing ridge and lasso estimators for data analysis, Procedia Engineering 201 (2017) 746–755. doi:10.1016/J.PROENG.2017.09.615.

[33] P. V. Santos, L. Ranzan, M. Farenzena, J. O. Trierweiler, K-rank: An evolution of y-rank for multiple solutions problem, Brazilian Journal of Chemical Engineering 36 (2019) 409–419. doi:10.1590/0104-6632.20190361S20170455.

[34] P. Kadlec, B. Gabrys, S. Strandt, Data-driven soft sensors in the process industry, Computers & Chemical Engineering 33 (2009) 795–814. doi:10.1016/J.COMPCHEMENG.2008.12.012.

[35] M. A. Helal, S. Eldawlatly, M. Taher, Using autoencoders for feature enhancement in motor imagery brain-computer interfaces, 2017 13th IASTED International Conference on Biomedical Engineering (BioMed) (2017) 89–93doi:10.2316/P.2017.852-052.

[36] A. M. Tăutan, A. C. Rossi, R. D. Francisco, B. Ionescu, Dimensionality reduction for eeg-based sleep stage detection: comparison of autoencoders, principal component analysis and factor analysis, Biomedical Engineering / Biomedizinische Technik 66 (2020) 125–136. doi:10.1515/BMT-2020-0139.

[37] P. Korkos, J. Kleemola, M. Linjama, A. Lehtovaara, Representation learning for detecting the faults in a wind turbine hydraulic pitch system using deep learning, Energies 15 (2022). doi:10.3390/EN15249279.