A neural network regularization method to address variance inflation in autoencoders

Boeun Kim * **Kyung Hwan Ryu** ** **Seongmin Heo** ***

* *Andlinger Center for Energy and the Environment, Princeton University, Princeton, NJ 08544, USA (e-mail: bk3460@princeton.edu)*
** *Department of Chemical Engineering, Sunchon National University, 225 Jungang-ro, Suncheon, Jeollanam-do 57922, Republic of Korea (e-mail: khryu@scnu.ac.kr)*
*** *Department of Chemical Engineering, Dankook University, Yongin 16890, Republic of Korea (e-mail: smheo@dankook.ac.kr)*

Abstract: There exist various machine learning techniques which can be used to reduce the dimensionality of original data while minimizing the information loss. Principal component analysis (PCA) is one of the most well known such techniques, which transforms the original correlated variables into uncorrelated variables called principal components. Although PCA is known to preserve the total variance of the original data during the transformation, there are some cases with a potential of variance inflation, where the total variance of principal components becomes much larger than that of original variables. It is important to prevent variance inflation, as it can negatively affect the performance of other application systems (e.g. process monitoring systems) which are designed on the basis of principal component with inflated variances. Variance inflation also has a high potential to occur during the training of autoencoder, a special type of neural network performing nonlinear version of PCA. Although there are several neural network regularization methods available to alleviate the problem of variance inflation, none of them is tailored to do such task. To this end, in this work, an alternative neural network regularization method is proposed, which can strongly regulate the total variance in the feature space. Using the Tennessee Eastman process as an illustrative example, the proposed regularization method is compared with the existing ones in terms of neural network overfitting, variance inflation, and training time.

Keywords: principal component analysis, autoencoder, feature extraction, feature variance, neural network regularization

1. INTRODUCTION

Data dimensionality reduction is one of the most important steps in machine learning applications, as it can expedite such applications by removing the redundancy in the original data (Pyatykh et al., 2012; Heo and Lee, 2018; Jiang and Yan, 2018; Ryu et al., 2018). Dimensionality reduction is also important for data visualization which can help understand the underlying characteristics of original data (Roweis and Saul, 2000; Hadsell et al., 2006). Principal component analysis (PCA) is a traditional yet very effective dimensionality reduction technique which performs orthogonal transformation to obtain new set of variables called principal components (Jolliffe and Cadima, 2016). Principal components can be naturally arranged in a descending order in terms of variance, and dimensionality reduction can be achieved by removing a few last components. In this way, PCA is able to reduce the dimensionality of data while preserving a significant amount of information encoded in the original data.

During the orthogonal transformation, PCA is known to preserve the total variance of the original data. However, in some cases, PCA causes the total variance of principal components to exceed that of original variables, which is called *variance inflation* (Kjems et al., 2001). Variance inflation is typically observed in the cases where the dimensionality of original data is larger than the number of training samples. In this case, the projection of the original data onto the principal component space is expected to be overfitted, leading to low generalization power. When the principal components with inflated variances are used for the design of other application systems, such as process monitoring systems, the performance of such systems is expected to be low due to weak generalization power (García-Moreno et al., 2012).

Variance inflation is also reported to occur for kernel principal component analysis (Abrahamsen and Hansen, 2011), which may suggest that any type of PCA can suffer from variance inflation under certain circumstances. Autoencoder is generally viewed as a nonlinear extension of PCA, which extracts nonlinear principal components from the original data through a self-reconstruction process (Kramer, 1991). These nonlinear principal components may have inflated variances, especially when the size of autoencoder becomes large as we try to increase

its explanatory power. It is also highly likely that variance inflation would lead to neural network overfitting, resulting in large test reconstruction errors, and thus, we can apply neural network regularization methods available in the literature during the network training to suppress variance inflation.

Among various neural network regularization methods, most widely used ones are L1 and L2 regularization methods. These methods essentially regulate the absolute values of weight parameters so that the variance in each layer does not become much larger than that of previous layer, thus maintaining the hypervolume in each feature space spanned by the training data at an acceptable value. Another type of regularization methods tries to reduce the number of effective parameters to prevent network overfitting, and some examples include dropout (Srivastava et al., 2014) and network pruning (Karnin, 1990; Han et al., 2015). Although the regularization methods mentioned above can prevent network overfitting, they cannot effectively handle the variance inflation since they are not specifically designed to do so (Heo and Lee, 2019b). To this end, in this work, we propose an alternative neural network regularization method, which directly regulates the total variance of the features in the bottleneck layer of autoencoders. The Tennessee Eastman process is used as an illustrative example, and the normal operation data are used to train autoencoders of varying sizes with different regularization methods. First, we analyze the total variance of the bottleneck layer using the existing regularization methods to demonstrate that variance inflation indeed occurs during the network training (even when the network overfitting is not observed), and when it becomes more significant. Then, we examine the effectiveness of the proposed regularization method by comparing it with the existing ones in terms of i) ability to prevent network overfitting, ii) ability to suppress variance inflation, and iii) neural network training time.

2. PRINCIPAL COMPONENT ANALYSIS (PCA)

2.1 Linear PCA

Let us consider a data sample matrix X whose dimensions are n by m, where m and n represent the number of variables and data samples, respectively. The principal components of X can be computed by the following equation:

$$T = XP \tag{1}$$

where T and P denote the score matrix and loading matrix, respectively. T is the matrix of principal component values, and P is the matrix with orthogonal column vectors each of which can be used to transform the original variables into uncorrelated variables. PCA is closely related to the singular value decomposition of X which can be written as:

$$X = U\Sigma V^T \tag{2}$$

where U, Σ and V are the matrices of left-singular vectors, singular values and right-singular vectors, respectively. Then, by comparing Eqs.(1) and (2), T and P can be simply set to be $U\Sigma$ and V, respectively. P can be partitioned into to submatrices as follows:

$$P = [P_{PC},\, P_R] \tag{3}$$

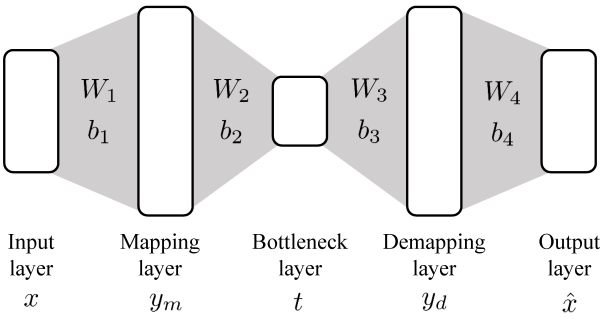

Fig. 1. Schematic representation of autoencoder

where P_{PC} is the matrix containing the first f column vectors of P, and P_R is the matrix of the remaining column vectors.

If the singular value decomposition is used to compute P, the singular values in Σ is arranged in a descending order, and the j-th column vector of P corresponds to the principal component with j-th largest variance. Thus, only P_{PC} can be applied to X to project the original data samples onto a feature space of reduced dimensionality.

2.2 Nonlinear PCA (NLPCA) by autoencoder

Figure 1 shows a schematic representation of a typical autoencoder, which is a special type of neural network used to learn non-trivial identity mappings through a self-reconstruction process. A mathematical model for autoencoders can be simply written in the following form:

$$\begin{aligned} y_m &= a(xW_1 + b_1) \\ t &= y_m W_2 + b_2 \\ y_d &= a(tW_3 + b_3) \\ \hat{x} &= y_d W_4 + b_4 \end{aligned} \tag{4}$$

where x and $\hat{x}$ denote the data sample and the reconstructed data sample. y_m and y_d represent the vectors of hidden layer nodes, and t is the vectors of bottleneck layer nodes. W's and b's are the weight matrices and the bias vectors. a represents an activation function such as sigmoid function and rectified linear unit (ReLU). To obtain an autoencoder with more hidden layers, the first and third equations in Eq.(4) can be used multiple times. Typical training objective for autoencoder is the reconstruction error which is defined as:

$$E = \frac{1}{2}\sum_i \sum_j (x_{ij} - \hat{x}_{ij})^2 \tag{5}$$

We can enforce an autoencoder to learn a non-trivial identity mapping with feature space of reduced dimensionality by restricting the size of bottleneck layer to be smaller than that of input layer.

3. NEURAL NETWORK REGULARIZATION METHODS

In this section, we provide a brief description of four neural network regularization methods: L1 and L2 regularizations, dropout and network pruning.

3.1 Regulating the value of weight parameters

The first type of neural network regularization methods is to control the average absolute values of weight parameters (i.e. elements of W's). L1 and L2 regularizations fall into this type, whose definitions are given by the following equations:

$$E_{reg} = E + \alpha L1 \tag{6}$$

$$E_{reg} = E + \alpha L2 \tag{7}$$

with

$$L1 = \sum |w_1| + \sum |w_2| + \sum |w_3| + \sum |w_4| \tag{8}$$

$$L2 = \sum w_1^2 + \sum w_2^2 + \sum w_3^2 + \sum w_4^2 \tag{9}$$

where w_k represents the elements of W_k. α is a weighting factor, which determines the level of regularization. If α is too small, autoencoder would have a high potential of overfitting, while if it is too large, autoencoder would learn a poor identity mapping, i.e. reconstruction error would be large. These two methods act as a neural network regularizer by limiting the variance amplification caused by each weight matrix.

3.2 Regulating the effective number of weight parameters

Another type of neural network regularization methods tries to reduce the effective number of weight parameters. A representative method is dropout, whose objective is to prevent co-adaptation of different neurons by deactivating random neurons (along with their connections to other neurons) during the training phase (Srivastava et al., 2014). Although dropout does not add an additional term to the training objective function, the level of regularization can be controlled by adjusting the portion of neurons to be deactivated in each iteration (or equivalently, epoch) of network training.

In network pruning, unimportant connections among neurons are permanently removed to reduce the number of parameters to be optimized. One criterion to determine relative importance of weight parameters is to calculate the sensitivity of reconstruction error with respect to the change in each parameter (Karnin, 1990). Then, the parameters with small sensitivities are removed. Another criterion is to set a threshold for the absolute value of weight parameters (Han et al., 2015). In this case, parameters whose absolute values are smaller than the threshold are permanently removed.

A systematic pruning procedure has been also proposed for autoencoders by Heo and Lee (2019a), whose schematic representation is shown in Figure 2. In this procedure, all the layers excluding the input and output layers are decomposed into f decoupled parts, where f is the number of neurons in the bottleneck layer. It has been reported that, through such decomposition, less correlated features can be extracted (compared to plain autoencoder), and the number of parameters can be reduced by a factor of up to 10 for large size autoencoders.

3.3 Alternative regularization method for autoencoders: variance regularization

Let us propose an alternative regularization method for autoencoder training, which we call variance regularization.

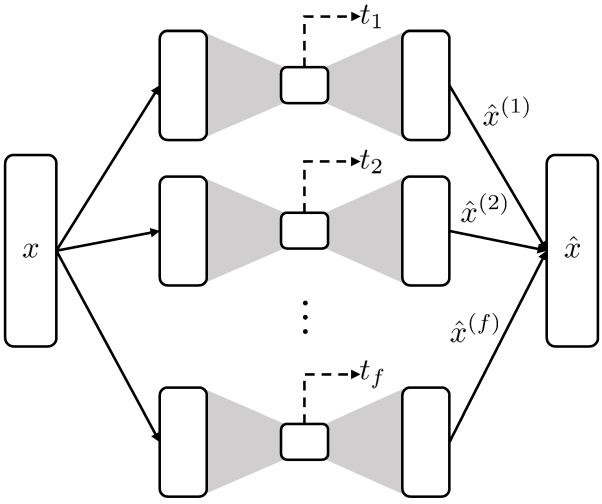

Fig. 2. Schematic representation of parallel autoencoders for uncorrelated feature extraction

The mathematical definition of this regularization method is given below:

$$E_{reg} = E + \alpha V \tag{10}$$

$$V = \sum_l t_l^2 \tag{11}$$

where t_l is the l-th element of t.

Variance regularization has a similar form as L1 and L2 regularizations, and is expected to act as a regularizer in a similar way. Since the total variance of bottleneck features depends only on the weight parameters of the mapping layers, variance regularization can be viewed as a modification of L2 regularization with a specific focus on addressing the problem of variance inflation. Although variance regularization poses no constraint on the weight parameters of demapping layers, they are expected to be automatically regulated by the original training objective (i.e. reconstruction error).

4. CASE STUDY: THE TENNESSEE EASTMAN PROCESS

In this section, we evaluate the effectiveness of the proposed regularization method from various aspects using the Tennessee Eastman process. First, we briefly describe the main features of the process, and the data used to train autoencoders. Then, we analyze the total variance of each layer (with a specific focus on the bottleneck layer) in autoencoders of different sizes trained using the existing regularization methods to illustrate the occurrence of variance inflation. Finally, we compare the proposed regularization method with the existing ones to illustrate its effectiveness.

4.1 Process and data description

Figure 3 shows the process flow diagram of the Tennessee Eastman (TE) process, which is widely used as a test bed for various process systems engineering applications (Downs and Vogel, 1993). This process involves five major unit operations (reactor, condenser, compressor, separator

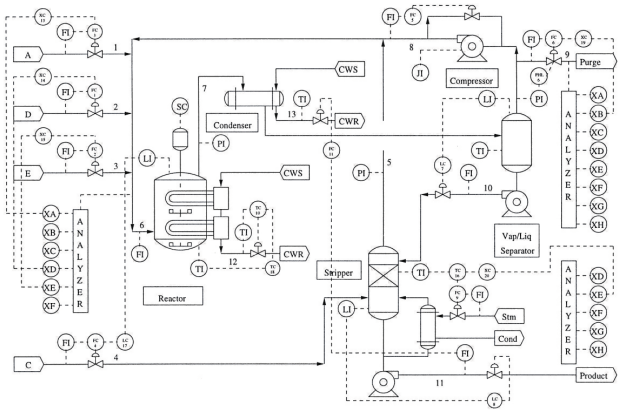

Fig. 3. Process flow diagram of the Tennessee Eastman process

Table 1. Different autoencoder structures used in this study

Network type	Network structure
Type 1	52-100-f-100-52
Type 2	52-100f-f-100f-52
Type 3	52-100-50-f-50-100-52
Type 4	52-100f-50f-f-50f-100f-52

Table 2. Neural network training settings

Attribute	Value	Reference
Training epochs	1000	
Learning rate	0.001	
Activation function	ReLU	Glorot et al. (2011)
Parameter initializer	Xavier	Glorot and Bengio (2010)
Optimizer	ADAM	Kingma and Ba (2014)

and stripper) and eight chemical compounds. Data samples from the TE process are of 52 dimensions, and 21 types of data samples are available (one normal operation data type and 20 faulty operation data types). In this study, only the normal operation data samples are used, which come from the large data set provided by Rieth et al. (2017). This dataset includes the normal data samples from 500 simulation runs (each of which consists of 500 samples) for the training, and from another 500 simulation runs (each of which consists of 960 samples) for the testing.

4.2 Neural network training settings

Two different autoencoder architectures shown in Figures 1 and 2 are used, which will be referred to as sm-NLPCA (simultaneous NLPCA) and p-NLPCA (parallel NLPCA), respectively. Table 1 tabulates four different neural network types used in this work, which have different size characteristics. Numbers in network structure column represent the number of nodes in each layer starting from the input layer, and f denotes the number of nodes in the bottleneck layer. Four different values are used for f: 5, 10, 15 and 20. Specific settings used for the neural network training are summarized in Table 2.

4.3 Variance analysis of the bottleneck layer

In the first case study, we evaluate the total variance of bottleneck layers in autoencoders trained without any regularization method, whose results are summarized in Table 3. In this work, the case with variance inflation is defined as a case where the total variance of bottleneck layer is larger than that of input layer. In Table 3 (and in similar tables), the cases with variance inflation are highlighted in bold. For a comparison purpose, the results from linear PCA are also included in this table. We can observe that the total variance of bottleneck layer generally increases with the increasing number of features as expected. Also, in some cases, the variance inflation was observed even though no overfitting was occurred during the network training. In the case of sm-NLPCA, the variance inflation became significant as the size of autoencoders increased, and p-NLPCA slightly alleviated this problem. However, p-NLPCA suffered from variance inflation when small size autoencoders were used.

Table 3. Total variance of the bottleneck features without neural network regularization

	Number of bottleneck features			
	5	10	15	20
Linear PCA				
	20.13	28.10	33.42	38.29
sm-NLPCA				
Type 1	7.42	12.64	19.53	23.31
Type 2	10.78	31.68	42.20	**54.18**
Type 3	3.65	6.67	10.24	16.13
Type 4	14.29	**56.54**	**86.89**	**108.74**
p-NLPCA				
Type 1	49.12	**85.78**	**142.54**	**130.74**
Type 2	15.36	46.18	**97.61**	**103.44**
Type 3	8.32	18.51	13.84	15.19
Type 4	22.33	30.47	**56.66**	**85.57**

Table 4. Total variance of the bottleneck features with the existing neural network regularization methods

	Regularization method			
f	None	L1	L2	Dropout
sm-NLPCA (Type 2)				
10	31.68	42.47	23.79	-
15	42.20	28.68	20.80	-
20	**54.18**	30.42	23.88	-
sm-NLPCA (Type 4)				
10	**56.54**	13.33	16.46	16.37
15	**86.89**	18.35	19.80	23.55
20	**108.74**	23.06	48.10	31.82
p-NLPCA (Type 2)				
10	46.18	22.95	10.83	-
15	**97.61**	47.48	23.23	-
20	**103.44**	**60.96**	24.69	-
p-NLPCA (Type 4)				
10	30.47	25.35	12.24	30.70
15	**56.66**	33.81	17.17	**63.89**
20	**85.57**	44.26	18.10	**70.86**

In the second case study, the existing neural network regularization methods are applied to the network training. Table 4 shows the results of this study, and the values for Type 2 with dropout are not reported, since dropout showed almost no regularization effect. We can see that, among three methods, L2 regularization showed the best performance on average in terms of variance inflation, since no variance inflation was observed when it is applied to the network training. Although variance inflation was observed for a few cases, L1 regularization and dropout were also able to reduce the total variance of bottleneck

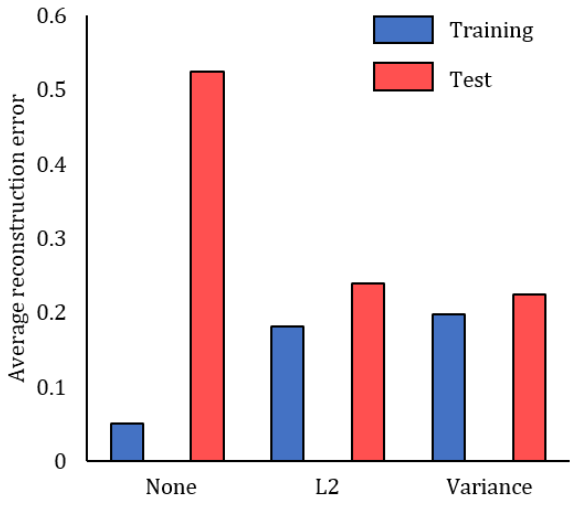

Fig. 4. Training and test errors with different regularization methods

layer for the most cases compared to the network without regularization.

4.4 Comparison 1: neural network overfitting

Let us now compare the variance regularization method with the other methods. Variance regularization is first compared with L2 regularization in terms of overfitting regularization power, and the results are shown in Figure 4. Type 4 sm-NLPCA autoencoder was used with 20 neurons in the bottleneck layer to obtain these results. From this figure, it can be seen that the autoencoder without any regularization is quite overfitted, showing very large test error. Both L2 and variance regularizations were able to handle the problem of network overfitting, and they showed similar regularization powers.

4.5 Comparison 2: variance inflation

In the next analysis, variance regularization is compared with L1 and L2 regularizations in terms of variance inflation inhibition power. Type 4 autoencoders with 20 bottleneck neurons were also used in this analysis, and the results are summarized in Table 5. Note that the results shown in this analysis were obtained by selecting different values of α for each regularization method such that the reconstruction errors for test data samples have similar values.

From this table, we can observe that the variance regularization indeed address the problem of variance inflation better than L1 and L2 regularizations. In the case of sm-NLPCA, the total variance with variance regularization is different from that with L1 or L2 regularizations by three orders of magnitude (an order of magnitude for p-NLPCA). Table 6 provides more detailed results by showing the total variance of each layer for L2 and variance regularizations. It can be seen that, for all the layers, variance regularization resulted in smaller total variances compared to L2 regularization. Also, as mentioned above, the demapping layers were not overfitted even though no constraint was put on them, showing similar total variance values as the mapping layers.

Table 5. Total variance of the bottleneck layer: comparison between L1, L2 and variance regularizations

	Regularization method			
	None	L1	L2	Variance
sm-NLPCA	108.74	23.06	48.10	0.01
p-NLPCA	85.57	44.26	18.10	1.28

Table 6. Total variance of each layer with L2 and variance regularizations

	Regularization method	
	L2	Variance
Input	52.28	52.28
Hidden 1	109.83	40.80
(activation)	39.07	9.65
Hidden 2	79.91	40.40
(activation)	24.97	0.26
Hidden 3	50.75	0.01
Hidden 4	143.40	0.52
(activation)	49.24	0.33
Hidden 5	74.29	41.75
(activation)	36.74	17.05
Output	41.20	41.84

Table 7. Neural network training time with different regularization methods (in seconds)

	Regularization method			
	None	L1	L2	Variance
sm-NLPCA	341.02	347.67	363.78	401.32
p-NLPCA	467.92	521.23	571.61	602.07

4.6 Comparison 3: neural network training time

Finally, let us compare the time required for the network training with different regularization methods. Table 7 tabulates the training time for both sm-NLPCA and p-NLPCA. More time was required to train p-NLPCA autoencoders than sm-NLPCA autoencoders, which is consistent with the results reported in Heo and Lee (2019a). Although it took the longest time to train autoencoders with variance regularization for both NLPCA methods, it is a computationally tractable regularization method, showing reasonably small differences with other methods. The main reason why it takes more time to train autoencoders with variance regularization is that it requires the evaluation the total variance, whose calculation increases with the sample data size, while L1 and L2 regularizations do not require such evaluation.

5. CONCLUSION

In this work, a regularization method for autoencoder training was proposed, which is called variance regularization. It was motivated by the phenomena known as variance inflation, which has been reported to be observed for linear PCA and kernel PCA. We showed that variance inflation also occurs in the nonlinear PCA performed by autoencoders, even when the autoencoders are trained without overfitting. We also demonstrated some cases where the existing regularization methods, such as L1 regularization and dropout, failed to prevent variance inflation.

Variance regularization was compared with L1 and L2 regularizations on the basis of three criteria: network

overfitting regularization, variance inflation inhibition and training time. It was shown that variance regularization has a comparable power to regulate network overfitting as the other regularizations, while showing much better ability to suppress variance inflation. However, it required slightly longer training time than the other regularization methods, as it involves the evaluation of feature values of all the test samples, which may be time consuming.

Although we demonstrated the effectiveness of variance regularization, there is still more to be explored. The most important issue that requires an extensive investigation is: can variance regularization be beneficial to other applications which build upon the results obtained from autoencoders? One example is the design of autoencoder-based process monitoring systems, which is our ongoing research task.

ACKNOWLEDGEMENT

The authors declare no financial support for this work.

REFERENCES

Abrahamsen, T.J. and Hansen, L.K. (2011). A cure for variance inflation in high dimensional kernel principal component analysis. *Journal of Machine Learning Research*, 12(6).

Downs, J.J. and Vogel, E.F. (1993). A plant-wide industrial process control problem. *Computers & Chemical Engineering*, 17(3), 245–255.

García-Moreno, P., Artés-Rodríguez, A., and Hansen, L.K. (2012). A hold-out method to correct pca variance inflation. In *2012 3rd International Workshop on Cognitive Information Processing (CIP)*, 1–6. IEEE.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 249–256. JMLR Workshop and Conference Proceedings.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 315–323. JMLR Workshop and Conference Proceedings.

Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, 1735–1742. IEEE.

Han, S., Pool, J., Tran, J., and Dally, W.J. (2015). Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*.

Heo, S. and Lee, J.H. (2018). Fault detection and classification using artificial neural networks. *IFAC-PapersOnLine*, 51(18), 470–475.

Heo, S. and Lee, J.H. (2019a). Parallel neural networks for improved nonlinear principal component analysis. *Computers & Chemical Engineering*, 127, 1–10.

Heo, S. and Lee, J.H. (2019b). Statistical process monitoring of the tennessee eastman process using parallel autoassociative neural networks and a large dataset. *Processes*, 7(7), 411.

Jiang, Q. and Yan, X. (2018). Parallel pca–kpca for nonlinear process monitoring. *Control Engineering Practice*, 80, 17–25.

Jolliffe, I.T. and Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065), 20150202.

Karnin, E.D. (1990). A simple procedure for pruning back-propagation trained neural networks. *IEEE transactions on neural networks*, 1(2), 239–242.

Kingma, D.P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kjems, U., Hansen, L.K., and Strother, S.C. (2001). Generalizable singular value decomposition for ill-posed datasets. In *Advances in neural information processing systems*, 549–555. Citeseer.

Kramer, M.A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2), 233–243.

Pyatykh, S., Hesser, J., and Zheng, L. (2012). Image noise level estimation by principal component analysis. *IEEE transactions on image processing*, 22(2), 687–699.

Rieth, C., Amsel, B., Tran, R., and Cook, M. (2017). Additional tennessee eastman process simulation data for anomaly detection evaluation. *Harvard Dataverse*, 1.

Roweis, S.T. and Saul, L.K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500), 2323–2326.

Ryu, J.Y., Kim, H.U., and Lee, S.Y. (2018). Deep learning improves prediction of drug–drug and drug–food interactions. *Proceedings of the National Academy of Sciences*, 115(18), E4304–E4311.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1), 1929–1958.