Enhancing Power Flow Simulations Using Function Mapping

Michael Bardwell*, Petr Musilek*†
*Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada
†Department of Cybernetics, Faculty of Science, University of Hradec Králové, Czech Republic

***Abstract*—The main contribution of this paper is demonstrating the approximation of the power system load flow (PSLF) function. The topological correspondence of radial, electrical power grids is captured with an Artificial Neural Network. A hyperparameter grid search was used to confirm the optimal training conditions. It was discovered that even for large radial networks, the identity linear regression outperforms non-linear regression under reasonable training circumstances. This indicates PSLF is a relatively linear function even when the feature space is large. The findings were used to develop a Python-based framework to autonomously handle simulation function mapping, which allows for model re-use making the simulation process less computationally expensive and less time-consuming on average.**

I. Introduction

Modern simulation techniques are expensive. High resolution, complex simulations can take on the order of hours or days to come to completion and require advanced, costly computers. When these simulations are completed, it is likely the researcher or practitioner will run another, restarting the whole process. The institutional solution is to purchase faster, more expensive computers. While the outputs of the simulation are the desired results, no conclusive research has been done on capturing the simulation model for re-use. It is viewed as a black box, often deleted post-simulation only for the same model to be rerun later on a different input sequence.

Today's simulation models are handled in engines like MATLAB Simulink. The engine handles highly multivariate, nonlinear equations, masked by a graphical user interface and solved using numerical methods. Advanced regression techniques and computing power has made it possible to capture extremely high-dimensional, nonlinear equations. This paper explores the regression techniques successful in function approximation of the topological correspondence in electric power grids.

$$f(x) = \sum_{i=0}^{z} w_i^T \cdot \phi_i(x) \tag{1}$$

The general regression formula is shown in Eq. (1). The function is linear with respect to the weights w, and the basis functions $\phi(x)$. One example of an estimator consistent with this formula are one-layer Artificial Neural Networks (ANN) and will be discussed later in this paper.

The electric power grid is governed by the power system load flow (PSLF) equation in Eq. (2). PSLF defines the relationship between the input power vector S_i, and the output voltage vector $V_{i/k}$. The user creates a Y-bus admittance matrix $Y_{ii/ik}$, by selecting the bus connection topology like the radial topology in Fig. 1, where each circle represents a node in the power system network. Our focus is to capture the unique correspondence between houses, embedded in Y, using function mapping.

$$V_i = \frac{1}{Y_{ii}} \cdot \left(\frac{S_i^*}{V_i^*} - \sum_{k=1, k \neq i}^{n} Y_{ik} \cdot V_k \right) \tag{2}$$

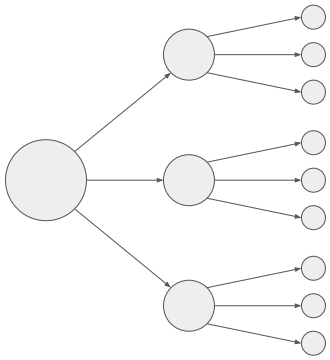

Fig. 1. A sample radial network

There are many function approximation techniques, starting in statistics with methods like Nonlinear Least Squares and Maximum Likelihood Estimates which belong to a broader class called M-estimators [1]. Radial basis functions are used in [2] to model nonlinear voiced speech sounds. In [3], Chen develops a backpropagation technique for multi-layer perceptron function approximation. Concepts of extrema equivalence are developed to estimate the complexity of a function in [4], thereby providing an empirical method to select ANN network size and topology. A statistical perspective on many of the aforementioned techniques is provided by Cheng in [5]; for example, assume there is a constant V, such that $f(x)/V$ is inside the convex hull of a trained one-layer ANN represented as f_M, amongst a few other assumptions. We can say the approximation error is bounded as:

$$\|f - f_M\| \leqslant \frac{V}{\sqrt{M}}, \tag{3}$$

where M is the number of neurons in the hidden layer.

Artificial neural networks are a good choice for function approximation due to their adjustable basis function nature [6]. This is excellent for our application as there is uncertainty in the nature of the PSLF-based function to be approximated. Before training, hyperparameter adjustments prepare the underlying basis structure and training tunes the basis functions.

Once a function has been approximated, it is important to make it as usable as possible. A Python program was developed to autonomously handle the PSLF function approximation process. The program is summarized in Fig. 3.

In Section II the nuances of function approximating are discussed. In Section III the experimental method is outlined and the results are tabulated in Section IV. Section V summarizes the contributions of this paper. The research is wrapped in an open-source Python package available through *pip install simhandler* or GitHub[1].

II. Background

A mapping is a rule of correspondence between input and output vectors. To function map a simulation model, two sets of vectors $a_i \in R^p$, $b_i \in R^q$ are required: a finite input data set $\{a_0, a_1, \ldots, a_m\}$ and a finite output data set $\{b_0, b_1, \ldots, b_n\}$; the output set is calculated by passing the input set through a well-defined simulation model. m, n are the number of input/output features in the simulation model respectively and the length of each vector in either set is defined as the number of samples and is orthogonal to m, n. For example, a Modified National Institute of Standards and Technology (MNIST)[2] identifier might have a 1000 pixels per image and 10 possible outcomes (digits 0-9), giving them m = 1000 and n = 10 respectively. The length of vector $a_0 = 100$, for example, which could be the top left pixel from 100 pictures; b_0 would be 100 respective classifications.

The more non-linear the simulation model, the more samples required for function mapping to meet high accuracy goals. For example, imagine the simulation model is a simple, two-dimensional linear function. From Eq. (1), assuming $\phi(x) = \{1, x\}$

$$f(x) = w_0 + w_1 x. \tag{4}$$

To determine the bias w_0 and weight w_1, you only require two, linearly independent, m-dimensional data points and you have a perfect function mapping of the model. Now let us imagine a more likely scenario for a simulation model, something like the noisy function in Fig. 2.

[1]https://github.com/mbardwell/intelligent-simulation-handler

[2]From http://yann.lecun.com/exdb/mnist/ accessed 20/03/2019

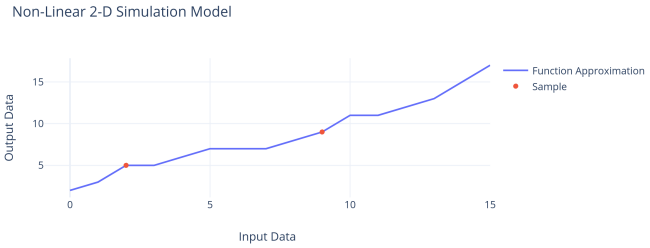

Fig. 2. Sample simulation data

It is obvious by inspection that any line, like one drawn between the two sample dots in Fig. 2, would be an underfitted approximation of the correspondence in this simulation. To get a better fit that encompasses the valley at $x = 7$ or the peak at $x = 10$, for example, more samples from the simulation model are required. This type of underfitting is exponentially amplified in higher dimensions, due to the curse of dimensionality [7], so with larger features spaces, even more data is required.

To map the PSLF equation, a feed-forward ANN is fed with electric power system data. The network is trained using simple backpropagation. A Monte Carlo approach is taken to fill S_i using a distribution with support $x \in [0, 1)$; the stochastic approach optimizes the efficiency of mapping non-linear functions, much like random search optimizes efficiency for hyperparameter exploration [8]; possibly, this is because the Monte Carlo approach nicely balances the feature vectors [9]. The input and output vectors are defined in per unit values and thus are naturally scaled. This should help avoid premature saturation of sigmoidal units [10]. The activation functions logistic sigmoid and ReLU will be used as activation functions. Since ReLU [11], defined as $max(0, x)$, zeroes any non-positive input, it is more likely to create sparse machines. This lowers capacity and is expected to be less desirable for regression.

There are many variations in power system topology, see Table I. The objective in this paper is to provide the experimental procedure for function mapping a radial feeder network with resistive-only lines. The methodology should naturally extend to other variations as well. For this paper, an accuracy of 10^{-2} was arbitrarily selected as sufficient. For example, if the base voltage for a distribution line is 35 kV, the upper bound on the estimation error would be 350 V. This factor can be used to tell the network when to stop training.

Mean-square error (MSE), also known as the ℓ_2 or Euclidean norm is the accuracy metric used for regression evaluation. It is optimal for function approximation as it is more sensitive to outliers than mean absolute error. Root-MSE (RMSE) will be used for analysis, considering the fact that it is in the same units as the feature space.

To demonstrate the accuracy capabilities of nonlinear function mapping, the results are compared to an identity linear regression (ILR) with m-dimensions, which is derived by setting $\phi_i(x) = x_i$ in Eq. (1)

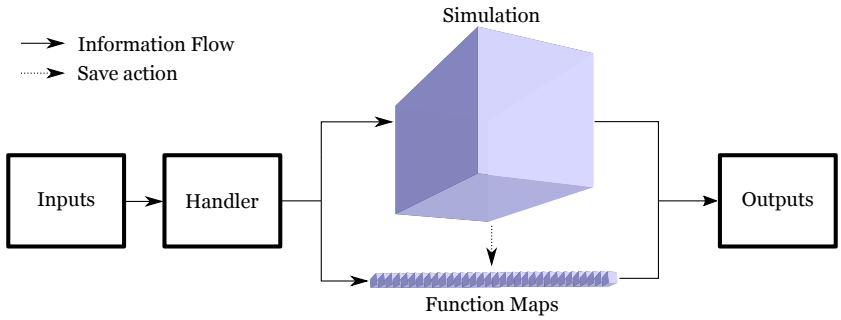

Fig. 3. Flow of information in Simulation Handler Python program. The user provides a simulation model and input samples and the handler automates the PSLF process. The output is a set of node voltages concurrent with the samples seen at the input

TABLE I
POWER SYSTEM NETWORK PARAMETERS

PS Parameter	Examples
Bus Topology	Main-Tie-Main, Ring, Primary Loop
Feeder Network	Radial, Parallel, Ring, Meshed
Line Characteristics	Resistance, Reactance, Length, Maximum Power
Line Support	Shunt/Series Compensation
Storage Parameters	BESS's, Flywheels, Compressed Air
Generator Availability	Minimum/Maximum Real and Reactive Power

$$f(x) = w_0 + \sum_{i=1}^{m} w_i^T \cdot x_i. \tag{5}$$

For smaller-featured systems, ILR may out-perform the ANN. This is because PSLF with very small systems tends to have little noise; also ILR is fitted to the data, which unlike gradient descent used in ANN training is an exact process. ILR is fitted using the normal equation.

$$\hat{\theta} = (\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot X^T \cdot \mathbf{y} \tag{6}$$

One non-obvious trade off between ILR and ANN is the expensive nature of normal equation calculations; for ILR a matrix inversion is required, which in Big-O notation is between $O(m^{2.4}) - O(m^3)$. The processing cost of ANN training scales linearly, or $O(m)$. This is demonstrated by the timing comparison in Fig. 4.

III. Experimental Method

The generic approach to function approximating PSLF equations is outlined in Fig. 5. To achieve results in swift fashion, a hyperparameter search will be automated using an open-source Python library, Talos.

According to [3], [12], three-layer ANN's are considered universal approximators. Naturally, this means one hidden layer would be a good starting point. Preliminary tuning with ILR, which is similar to a one-layer ANN has shown reasonable results. For variety, the number of hidden layers will be tuned between 1-3.

Sample size is also important. The theoretical optimum S_i would be an infinite set of samples, which would then be used

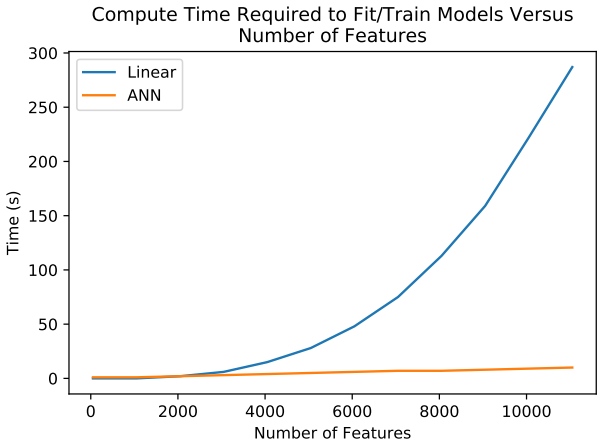

Fig. 4. Compute time required to fit an ILR (*Linear* in legend) using the normal equation versus train an ANN using backpropagation

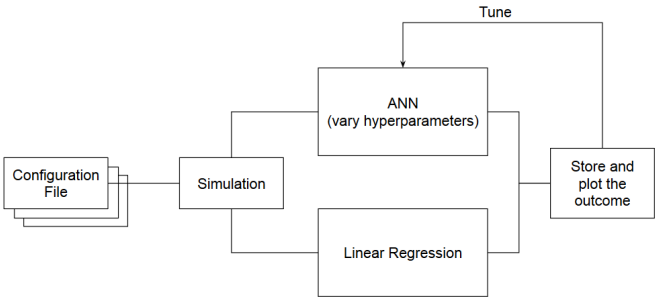

Fig. 5. General approach to function mapping PSLF equations

to fit/train the approximation. However, since this is practically unrealistic, a preliminary study was done to determine the time it takes to calculate the output set $V_{i/k}$ of various sizes. The relationship between processing time and set size can then be used to select a reasonable sample size. Note that the intention of this study is to observe the high extrema of accuracy, so a high sample/small feature space approach was taken.

The results from Fig. 6 show the run time increases around $0.034\ s$ per sample for a very small network. Another study showed an increase of $0.016\ s$ per feature. This gives us the

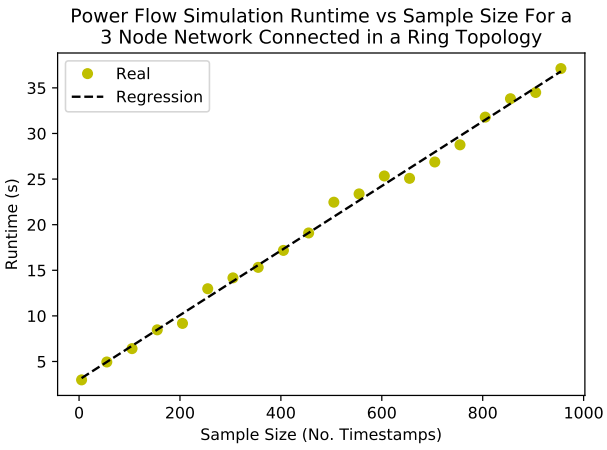

Fig. 6. Power system load flow analysis time requirements as a function of sample size. Run on Intel Core i5 and 16 GB of RAM

following run time equation: $t(s) = 0.034 \cdot N + 0.016 \cdot m$, with N being the number of samples. The study will be capped at $m = 200$ and $N = 5000$ which puts our maximum run time around 3 minutes, this is very reasonable as it allows us to collect many data points. In short, N will be bounded by $[m_{min}, 25m]$. The optimization algorithm will be ADAM [13], a very popular variant of momentum-based optimizers and the initial learning rate will be set to 10^{-4}.

IV. Results

Using 100 samples, a grid search was performed on a simple 3 node, radial network. The validation loss in Fig. 7 shows that sigmoid performs about 4 times better on average than ReLu. Layer density appeared to have the next biggest effect, followed by number of layers. Initial learning rate had an unnoticeable effect.

Fig. 8 shows the results of function approximating with ILR. It is clear that when the sample-to-feature ratio approaches two, the error exponentially ramps upwards. It can also be observed that for sample-to-feature ratios above five, the results are very similar. This suggests that there is a statistically significant sample-to-feature roof for ILR around five, which permits an empirical selection of samples when given the number of features by the user.

The results in Fig. 9 show that the nonlinear ANN solution produces more error than the ILR fit. This would suggest that the PSLF has extremely little noise, even when the feature space is as large as 200 dimensions. Further hyperparameter tuning should result in better ANN results but may require a potentially unrealistic amount of time. One interesting observation from the figure is the approximation results are much more sporadic when the samples-to-feature ratio is below five. It suggests that the nonlinear solution requires more data to be trained, than the ILR to be fitted, in order to produce results with a low RMSE, which is to be expected [14]. The underlying trends of both Fig. 8 and Fig. 9 are interesting and something that will be considered in future work.

mae	mse	val_mae	val_mse	val_rmse	first_neuron	lr	hidden_layers	Activation
3.96E-03	8.12E-05	4.46E-03	1.17E-04	1.08E-02	16	1.00E-05	1	relu
4.41E-03	3.21E-05	6.22E-03	6.76E-05	8.22E-03	64	0.001	1	relu
5.22E-03	6.80E-05	7.72E-03	1.49E-04	1.22E-02	32	1.00E-05	1	relu
2.80E-03	1.83E-05	4.50E-03	7.97E-05	8.93E-03	16	0.001	1	relu
3.52E-03	2.02E-05	6.17E-03	7.00E-05	8.37E-03	64	1.00E-05	1	relu
9.06E-03	1.32E-04	1.22E-02	2.32E-04	1.52E-02	32	0.001	1	relu
2.90E-03	1.64E-05	4.64E-03	5.12E-05	7.16E-03	64	0.001	3	relu
4.80E-03	6.00E-05	6.39E-03	1.30E-04	1.14E-02	16	0.001	3	relu
6.45E-03	7.79E-05	7.37E-03	9.60E-05	9.80E-03	32	1.00E-05	3	relu
1.14E-02	2.05E-04	1.16E-02	2.14E-04	1.46E-02	16	1.00E-05	3	relu
4.33E-03	2.82E-05	7.90E-03	1.23E-04	1.11E-02	64	1.00E-05	3	relu
6.83E-03	8.79E-05	6.95E-03	8.18E-05	9.04E-03	32	0.001	3	relu
				1.06E-02				
1.04E-03	1.83E-06	9.77E-04	2.03E-06	1.43E-03	64	0.001	1	sigmoid
5.13E-03	6.55E-05	4.28E-03	3.78E-05	6.15E-03	16	1.00E-05	1	sigmoid
6.66E-04	1.09E-06	7.30E-04	1.20E-06	1.09E-03	64	1.00E-05	1	sigmoid
3.35E-03	2.44E-05	3.42E-03	2.25E-05	4.74E-03	16	0.001	1	sigmoid
1.51E-03	3.97E-06	1.53E-03	4.27E-06	2.07E-03	32	0.001	1	sigmoid
1.25E-03	2.87E-06	1.32E-03	3.34E-06	1.83E-03	32	1.00E-05	1	sigmoid
2.48E-03	1.38E-05	2.31E-03	1.09E-05	3.30E-03	16	1.00E-05	3	sigmoid
2.98E-03	1.48E-05	2.71E-03	1.20E-05	3.46E-03	16	0.001	3	sigmoid
1.49E-03	4.58E-06	1.63E-03	5.76E-06	2.40E-03	32	0.001	3	sigmoid
1.11E-03	2.18E-06	1.18E-03	2.16E-06	1.47E-03	32	1.00E-05	3	sigmoid
6.02E-04	6.80E-07	5.59E-04	5.42E-07	7.36E-04	64	0.001	3	sigmoid
5.94E-04	6.28E-07	5.38E-04	5.49E-07	7.41E-04	64	1.00E-05	3	sigmoid
				2.45E-03				

Fig. 7. Grid search. Manipulated Hyperparameters include activation function, number of layers, density of layers, learning rate. Activation function had the largest effect as expected

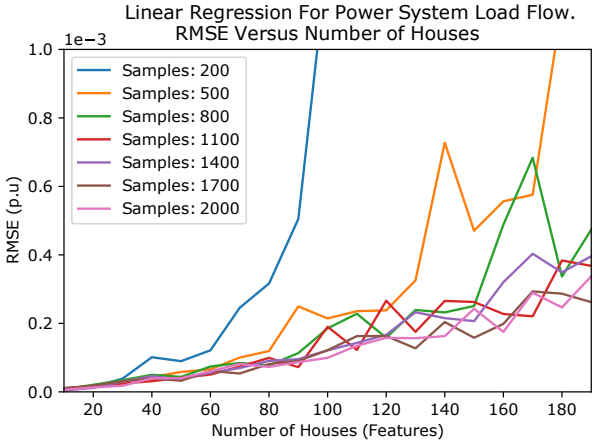

Fig. 8. ILR for PSLF equations with various input features spaces and sample

Fig. 10 visualises the accuracy of function approximating with both ILR and ANN mappings for one node in a 500 house radial power system network . The ILR and ANN RMSE were 0.0021 and 0.0019 respectively.

V. Conclusions

In this paper, multiple function approximation techniques are applied to a PSLF equation. The ILR outperformed the ANN-based nonlinear regression which was contrary to the original hypothesis. More hyperparameter tuning would likely deliver better ANN results, but it is unclear how long it would take to decrease RMSE by at least two orders of magnitude, which would make it outperform the ILR solution according to the results in Fig. 8 and Fig. 9. A hyperparameter search suggests using sigmoid activation functions and increasing layer density would be the best approach. However future

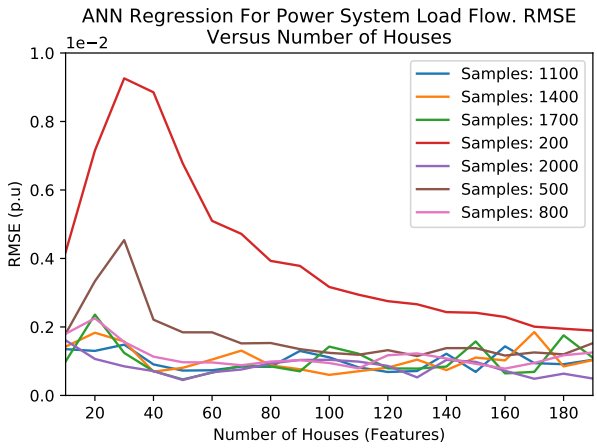

Fig. 9. ANN for PSLF equations with various input features spaces and samples

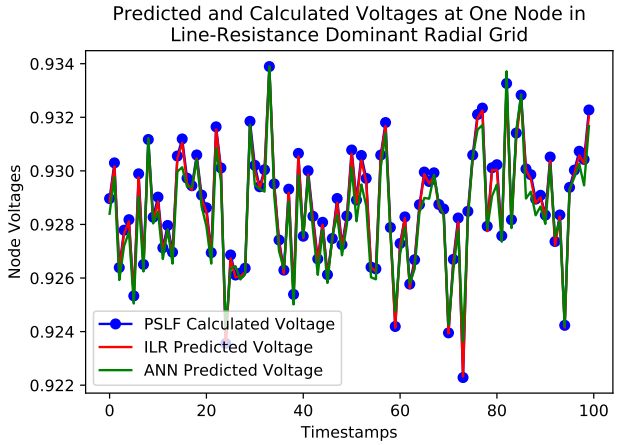

Fig. 10. Comparison of voltage profiles: PSLF calculated, ILR approximated and ANN approximated. Radial network included 500 houses

work should also consider deeper networks with ReLU [15]. Consideration of the extrema in the feature space could lead to an analytical selection of ANN hyperparameters. Further studies should include other features common to power system topologies found in Table. I.

ACKNOWLEDGMENT

Support from Future Energy Systems under the Canada First Research Excellence Fund (CFREF) at the University of Alberta is humbly appreciated.

REFERENCES

[1] F. Hayashi, *Econometrics*. Princeton University Press, 2000.

[2] I. Mann, "An Investigation of Nonlinear Speech Synthesis and Pitch Modification Techniques," Ph.D. dissertation, The University of Edinburgh, 1999.

[3] D. S. Chen and R. C. Jain, "A robust backpropagation learning algorithm for function approximation," *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 467–479, May 1994.

[4] X. M. Zhang, Y. Q. Chen, N. Ansari, and Y. Q. Shi, "Mini-max initialization for function approximation," *Neurocomputing*, vol. 57, pp. 389 – 409, 2004, new Aspects in Neurocomputing: 10th European Symposium on Artificial Neural Networks 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092523120300522

[5] B. Cheng and D. M. Titterington, "Neural Networks: A Review from Statistical Perspective," *Statistical Science*, vol. 9, no. 1, pp. 2–30, Feb. 1994.

[6] A. Barron, "[Neural Networks: A Review from Statistical Perspective]: Comment," *Statistical Science*, vol. 9, no. 1, pp. 2–30, Feb. 1994.

[7] G. Geenens *et al.*, "Curse of dimensionality and related issues in nonparametric functional regression," *Statistics Surveys*, vol. 5, pp. 30–43, 2011.

[8] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[9] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.

[10] S. Khomfoi and L. M. Tolbert, "Fault diagnostic system for a multilevel inverter using a neural network," *IEEE Transactions on Power Electronics*, vol. 22, no. 3, pp. 1062–1069, May 2007.

[11] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[12] K. Du and M. Swamy, *Neural Networks and Statistical Learning*. Springer, London, 2014.

[13] D. Kingma, P. and J. Ba, Lie, "Adam: A Method for Stochastic Optimization," in *ICLR 2015*, 2015.

[14] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," vol. 4, pp. 1–58, 1992.

[15] J. Schmidt-Hieber, "Nonparametric regression using deep neural networks with relu activation function," 08 2017.