Using Transfer Learning to Distinguish between Natural and Forced Oscillations

Zhe Yu[1], Di Shi[1], Jin Li[2], Yishen Wang[1], Xiaoying Zhao[1], Zhiwei Wang[1], and Jie Li[3]

[1]GEIRI North America, San Jose, CA, 95134, USA
[2]NARI Group, Nanjing, Jiangsu, 211000, China
[3]State Grid US Representative Office, New York, NY, 10119, USA
Email: zhe.yu@geirina.net

***Abstract*—In power systems, there are two different types of oscillation mechanisms–natural and forced oscillations. Natural oscillations are caused by system disturbance when the damping of the system is not sufficient. Forced oscillations are typically caused by external input driving the system into a sustained oscillation. Distinguishing low-frequency oscillations is a prerequisite to dealing with oscillation events in power systems. This paper attempts to distinguish between natural oscillations and forced oscillations using machine learning technologies. Decision tree, support vector machine, neural network, and convolutional neural network algorithms are evaluated. Transfer learning is applied to overcome the lack of training data.**

***Index Terms*—Oscillation Mechanism Classification, Neural Networks, Convolutional Neural Networks, Support Vector Machine, Decision Tree, Transfer Learning**

I. Introduction

With the growth in size of interconnected power systems and the participation of unsynchronized distributed energy resources, the phenomenon of oscillation has become common and widespread [1], [2]. Insufficient damped oscillations reduce the system margin and increase the risk of instability and cascading failure. Thus timely and precise control response is crucial.

Oscillations are typically classified as either natural or forced, based on their initial causes. Natural oscillation is caused by a lack of system damping and is triggered by disturbance. Forced oscillation is due to periodic energy injection into the system and can occur even when system damping is sufficient. The most common control strategy for natural oscillations is to adjust the power system stabilizer. The most effective control for forced oscillations is to locate the disturbance source. Thus distinguishing the two types of oscillations is a prerequisite for the effective damping of oscillations.

Oscillation classifications have been attracting more attention in the past decade. Envelope based approaches have been proposed in [3] and [4], in which an increase in amplitude is used to distinguish natural oscillations from forced ones. However, the accuracy of the classification depends on the size of the envelope, since the algorithm is found failing when the oscillation is lightly damped [5], [6]. The performance of the spectral method proposed in [5] is shown to degrade when the forced oscillation has a frequency close to a system mode frequency. In [7], a power spectral density and kurtosis based approach is proposed, which is simple and accurate when there is a long time period of data. However, the long time data requirement limits the method as an off-line application.

As described here, the state of the art in oscillation classification methods typically tends to extract some features of different mechanisms and then summarize them to a given index. This is followed by application of simple (linear) logic rules for the classification of oscillation events. This approach usually is complicated and considerable oscillation event information is lost in the process. Moreover, the rules are typically linear and over-simplified. In some studies, machine learning approaches have been applied in the area of oscillation detection, analysis, and damping control [4], [8]–[10] that demonstrate the advantages of dealing with highly nonlinear systems. In this paper, we apply machine learning techniques to identify oscillation mechanisms that are capable of keeping in tact as much information as possible of the system while simultaneously addressing the common problem of lack of data in the system.

A. Summary of results

The three major contributions of this work are as follows: first, mainstream machine learning approaches are applied to distinguish natural and forced oscillations using simple features that keep as much information of the system as possible. Second, to overcome the impact of detection of starting points of oscillations, a time augmentation approach is proposed. Third, a transfer learning approach is applied to transfer models between different systems, which helps to resolve the problem of lack of training data.

II. Review of Algorithms

In this work, making the distinction between natural and forced oscillations is formulated as a supervised learning problem, by which oscillation data is collected. Features are extracted, and oscillation types are labeled in terms by domain experts. Features and labels are fed to supervised learning algorithms to train a classifier model. The trained classifier can

This work was funded by SGCC Science and Technology Program under project "AI based oscillation detection and control" and contract number SGJS0000DKJS1801231.

be used online to distinguish oscillation mechanisms. The key points during this process are feature extraction and classifier model selection. Accurate extraction is needed to ensure that all information is available to train classifier models and to remove noise information. Another requirement of feature extraction is that it must reduce the volume of data, *i.e.,* the size of the features should be as small as possible.

A. Classification Tree

A classification tree is used to divide samples into distinct and non-overlapping regions. Specifically, in this instance the classification tree divides the predictor space into high-dimensional rectangles or boxes. Any new sample is classified as the most commonly occurring class of training observations in the region to which it belongs. The classification tree is intuitive and easy to explain. However, its predictive accuracy is usually not at the same level of other approaches.

B. Support Vector Classifier

Given a feature vector $\boldsymbol{x} \in \mathbb{R}^P$, a support vector classifier seeks to find a hyper plane $w_0 + w_1 x_1 + \cdots + w_P x_P = 0$ to divide samples into the corresponding hyperplane. A typical support vector classifier (SVC) model is formulated as follows:

$$\begin{array}{l} \max_{w_p, \epsilon_i, M} \quad M \\ \text{subject to} \quad \sum_{p=1}^{P} w_p^2 = 1, \\ y_i(w_0 + w_1 x_{i,1} + \cdots + w_P x_{i,P}) \geq M(1 - \epsilon_i), \\ \epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C. \end{array} \tag{1}$$

where $\boldsymbol{w} \triangleq (w_0, \ldots, w_P)^\intercal$ is the parameter that defines the hyper-plane, $x_{i,k}$ the kth element of the ith sample, $y_i \in \{0, 1\}$ the label of the ith sample, M the width of the margin between two regions, ϵ_i are slack variables that allow observations to be on the wrong side of the hyper-plane, n the number of training samples, and C is the non-negative tuning parameter defining the "budget" that allows the margin that can be violated. After solving Equation (1), the set of parameter $\boldsymbol{w}$ defines a classifier. For a new sample $\boldsymbol{x}$, the sign of $f(\boldsymbol{x}) = w_0 + \sum_{i=1}^{P} w_i x_i$ indicates its category.

It can be shown that the linear support vector classifier $\boldsymbol{w}$, which is the optimum solution of Equation (1), can be represented as $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i$. α_i's are the optimum solution of the dual problem of Equation (1) and the classifier can be stated as $f(\boldsymbol{x}) = w_0 + \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i^\intercal \boldsymbol{x}$.

The support vector machine (SVM) is an extension of the support vector classifier [11], [12]. For an SVC, the classifier is represented by the inner product of features, $< \boldsymbol{x}_i, \boldsymbol{x}_j >= \boldsymbol{x}_i^\intercal \boldsymbol{x}_j$, which defines a linear boundary between classes. For an SVM, the inner product is replaced by a kernel function, $K(X_i, X_j)$, which quantifies the similarity of two observations. Commonly used kernel functions include polynomial kernels and radial kernels.

C. Feedforward Neural Networks

A feedforward neural network (FNN) is typically composed of three components, *e.g.,* input, hidden, and output layers. Each layer is composed by processing units, so called "nodes" or "neurons". Figure 1 shows an example of an FNN with a 3 dimension input and a 2 dimension output. The input layer usually does not have any special form other than simply input features. As shown in Figure 1, the input feature vector $\boldsymbol{x} \in \mathbb{R}^3$ is passed to each neuron in the hidden layer.

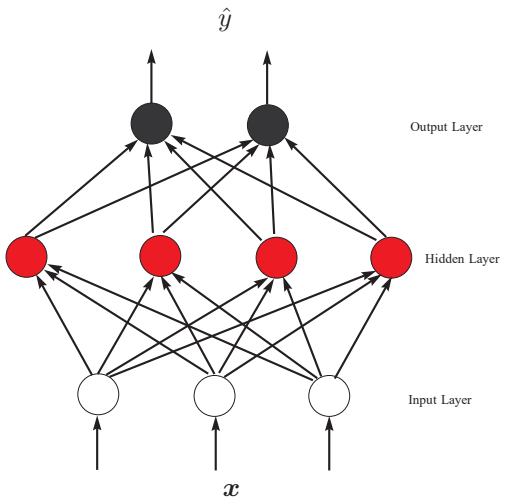


Fig. 1. An example of the structure of feedforward neural network.

A neuron in a hidden layer takes the result vector from the previous layer, either the input layer or another hidden layer, and applies a nonlinear transform to the vector. It then passes the processed results to the next layer. A simple example of the nonlinear transform can be stated as follows.

$$\begin{aligned} h(\boldsymbol{x}) &= g(f(\boldsymbol{x})) \\ f(\boldsymbol{x}) &= W^\intercal \boldsymbol{x} + \boldsymbol{b} \\ g(x) &= \max\{0, x\} \end{aligned}$$

where $f(\boldsymbol{x})$ is a linear transform with weight W and bias $\boldsymbol{b}$, and $g(x)$ is an activation function or rectified linear unit (ReLU). Other activation functions include softplus, radial basis function, and hard tanh.

The neurons in the output layer of a two-class classification problem is usually formulated as sigmoid units. Given the result vector $\boldsymbol{h}$ from the previous hidden layer, a sigmoid unit is defined as follows.

$$\hat{y} = \sigma(\boldsymbol{z}) = 1/(1 + \exp(-\boldsymbol{w}^\intercal \boldsymbol{h} - c))$$

where $\sigma(z)$ is the logistic sigmoid function, and $z = \boldsymbol{w}^\intercal \boldsymbol{h} + c$ is a linear transform of the outputs $\boldsymbol{h}$ from previous layers.

Given a network structure like the one in Figure 1, a feedforward neural network looks for an optimal set of parameters to maximize the likelihood with training samples $\boldsymbol{x}$, or an equivalent, to minimize the loss function defined as the negative log-likelihood. The log-likelihood for such a two-class classification problem is usually formulated as a linear function of label y's and the output of the final layer z's, *i.e.,* $\log P(y) \propto yz$. Then the loss function for maximum likelihood learning of a Bernoulli distribution is stated as

$$-\log P_{\boldsymbol{\theta}}(y|\boldsymbol{x}) = -\log \sigma((2y - 1)z)$$

where $\boldsymbol{\theta} = [W, \boldsymbol{b}, \boldsymbol{w}, c]$ are the parameters. Thus the training process of the feedforward neural network can be formulated as follows.

$$\min_{\boldsymbol{\theta}} \quad J(\boldsymbol{\theta}) = -\mathbb{E} \log P_{\boldsymbol{\theta}}(y|\boldsymbol{x}) \tag{2}$$

Unfortunately, solving Equation (2) is not trivial due to the nonlinearity. Gradient based algorithms are usually applied to look for local optimums.

D. Convolutional Neural Networks

The convolutional neural network (CNN) is a popular approach in image processing. Its applications include face recognition, item detection, and video processing. A typical CNN model can be found in Figure 2. It takes in an image, represented by a sum of multiple matrices, as the input. Usually, there are three matrices indicating three channels of RGB colors, and the image can be viewed as the sum of these three matrices. However, there can be more channels of signals that do not change the fundamentals. The signal is passed through an input layer similar to that in neural networks. The signal then goes through several convolution layers and pooling layers, which is the most important architecture of CNN.

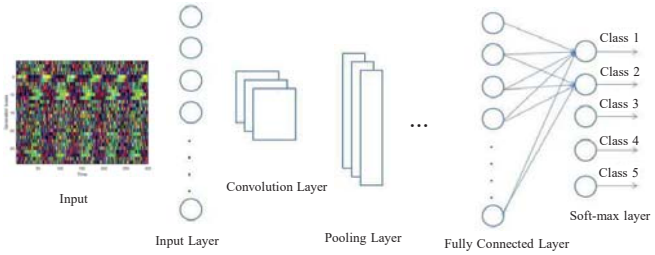
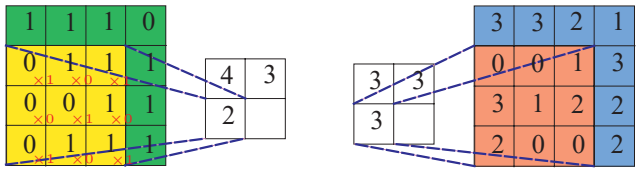


Fig. 2. A typical CNN model

As shown in Figure 3(a), a convolution layer defines a mask/filter (the orange one) and convolutes it with each input matrix. This process will result in a feature matrix smaller or equal to the original matrix. The purpose of this process is to extract the feature in the signal. The size of this filter is a critical design of the CNN model. A default choice would be to choose a mask with an odd number of pixels in each dimension. The value of the elements in the filter is adjusted during the learning process.

After the convolution layer, a pooling layer is constructed to reduce the dimension. Typical pooling includes maximum pooling and mean pooling. As shown in Figure 3(b), the maximum pooling moves a mask through a matrix and calculates the maximum within the mask. This process is intended to reduce the computational cost and de-noise the signal.

After several convolution and pooling layers, the result is passed to a fully connected layer and a classification layer similar to those in other neural networks.



(a) A typical convolution process (b) A typical pooling process

Fig. 3. Convolution and pooling layers

E. Transfer Learning

A critical uncertainty in application of machine learning approaches in power system studies is that there is not enough labeled data. To address this point, this work employs transfer learning techniques across different system data to evaluate the performance. The basic idea is to take a pre-trained neural network and use samples from other systems or scenarios to retrain (part of) the network and complete other tasks.

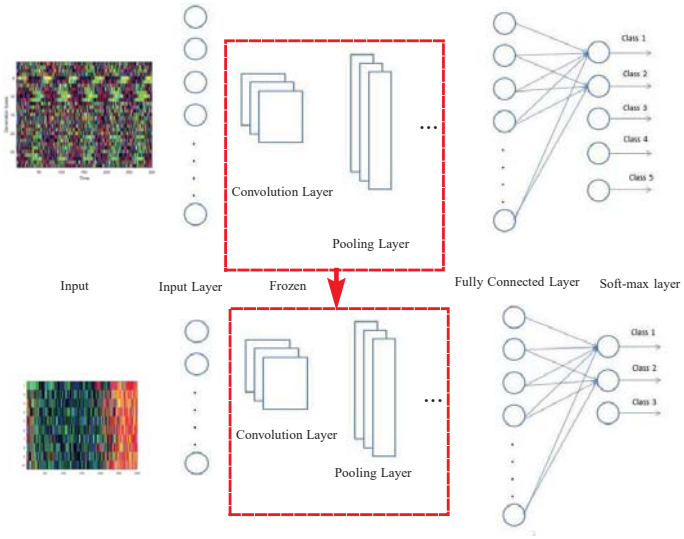


Fig. 4. An example of transfer learning

As shown in Figure 4, one CNN model is trained using data from the WECC 179-bus system [13]. The pre-trained convolutional layers and pooling layers are taken out to test in a 2-Area-4-Machine system [14]. An input layer, a fully connected layer, and a classification layer are added to the front and back of the pre-trained networks to adjust the input and output dimensions properly. Then a small number of samples from the 2-Area-4Machine system is fed into the newly constructed network to retrain. During the retrain process, the inherited part of the network is kept frozen, while the number of samples are far less than usual cases. In this way, the information of the WECC 179-bus system is utilized and helps to develop a model that performs well in the 2-Area-4-Machine system.

III. Numerical Results

A. Sample Generation

In this work, the Kundur 2-Area-4-Machine (2A4M) and WECC 179-Bus (179Bus) test systems are simulated using the Transient Security Assessment Tool (TSAT) [15]. For natural oscillation cases, the damping factor of each generator is set to a random value uniformly distributed among $[0, 4]$. Further, loads at each bus are multiplied by factors uniformly distributed among $[.9, 1.1]$ to mimic the randomness in operation conditions. A three-phase fault is added to a random bus and cleared after 0.5 seconds to trigger oscillations. Other parameters are kept unchanged.

For forced oscillation cases, a sinusoid signal is added to the exciter of a randomly picked generator, and the damping factor of the chosen generator is set to 0 to mimic the injected oscillation source. Loads at different buses are multiplied by factors uniformly distributed among $[.9, 1.1]$. Other parameters are kept unchanged.

Four hundred natural oscillations and 400 forced oscillations are generated for the 2A4M system, and 900 natural oscillations and 1400 forced oscillations are generated for the 179Bus system. After the generation of raw data, a Gaussian distributed factor is multiplied to each measurement to simulate the measurement noise.

B. Feature Selection

In this work, we select the nonlinear phase of oscillations as the input signal, *i.e.,* the beginning period of oscillations. Considering that it is hard to detect precisely the beginning point of oscillations, a sliding window with a 5 second width is applied to samples. In this way, multiple sample clips with different beginning points are generated using one piece of data. Furthermore, each clip of sample is normalized to its z-score[1] to eliminate the impact of different magnitude of signals.

For a CNN model, the feature extraction process is mainly dealt by the convolution process, which simplifies the procedure. Three time variant matrices are constructed using generator angle, voltage, and speed matrices.

$$X_{\text{ang}} \triangleq \begin{bmatrix} X_{\text{ang},1}[1] & X_{\text{ang},1}[2] & \dots & X_{\text{ang},1}[T] \\ X_{\text{ang},2}[1] & X_{\text{ang},2}[2] & \dots & X_{\text{ang},2}[T] \\ \dots & & & \\ X_{\text{ang},N}[1] & X_{\text{ang},N}[2] & \dots & X_{\text{ang},N}[T] \end{bmatrix} \tag{3}$$

In Equation (3), a matrix of generator angle is constructed, where N is the number of generators and T is the number of time instances. The same process is carried out to generate generator voltage and speed matrices.

Of the decision tree, SVC, and FNN models, the features selection is performed manually. In this work, the distribution of generator voltages and the standard statistic features including kurtosis and skewness are chosen as the features after experiments [16].

[1]Z-score is defined as follows.

$$z[t] = (x[t] - \mu(x))/\sigma(x)$$

, where $\mu(x)$ and $\sigma(x)$ are the mean and standard deviation of time series x

C. Classification Results

Monte Carlo simulations are carried out to validate the performance of different approaches. In each Monte Carlo run, the labeled data set is separated to a training set and a testing set randomly with a ratio of $0.8/0.2$. Furthermore, for each clip of training data, 10 samples are generated by sliding a window with width of 5 seconds and a step size of 0.2 second, *i.e.,* the 10th sample is 1.8 seconds later than the first one. To generate a clip of test data, a starting point uniformly distributed among $[0, 2]$ is first generated. Then a clip of data with the randomly generated starting point and window width of 5 seconds is sampled from the simulation data.

Various models are trained using the training set and tested on the test set. A kurtosis-based method is adopted as a benchmark [7], which uses a threshold of data kurtosis to distinguish oscillation classes. The threshold of kurtosis is set to -0.5. The accuracy is averaged over all Monte Carlo simulations and shown in Table I. All machine learning models perform well, which indicates the efficiency of the features in identification of the oscillation types. On the other hand, the kurtosis method is not the desirable method in this case due to the short period of data.

TABLE I
Average accuracy of models over test set

System	Decision Tree	SVM	FNN	CNN	Kurtosis
2A4M	99.97%	99.31%	100%	93.04%	56.06%
179Bus	99.60%	95.43%	100%	100%	68.60%

Next, the trained model using one system is directly applied to the other system to test the transfer performance without retraining. The result is shown in Table II. For example, a decision tree model is trained using all samples of the 2A4M system and tested directly using the samples of the 179Bus system and listed in the first cell of the table. Note that the input dimension of the CNN model is different for two test systems, thus the CNN model is not tested here. It is illustrated that the performance is not desirable if the model is directly applied to a different system.

TABLE II
Accuracy of models tested using samples from the other system

System	Decision Tree	SVM	FNN
2A4M to 179Bus	72.54%	39.13%	39.35%
179Bus to 2A4M	78.85%	96.16%	55.2%

D. Transfer Learning Results

In this subsection, the FNN model is first trained using all labeled data from one system. Then it is retrained using 1% data, and tested using the remaining data from the second system. In the retraining process, the learning rate is limited

to 0.01 and the number of epochs is limited to 5 so that the retrained model is kept frozen as much as possible. The result of the retrained model is summarized in Table III and IV. In Table III, an FNN model is trained using data from the 2A4M system and tested in the 179Bus system. After retraining using 1% data from 179Bus, the accuracy of the FNN model increases from 39.57% to 90.2% using 179Bus samples. Then the retrained model is tested again using samples from the 2A4M system, and the accuracy decreases from 100% to 84.28%. These results suggest that the retrained model performs well in the test (179Bus) system even with small number of retraining data. Meanwhile, the accuracy of the retrained model remains high for the original training (2A4M) system, which suggests that most structures of the model are inherited. The reverse process is carried out, and a similar result is obtained and shown in Table IV.

TABLE III
ACCURACY OF TRANSFER LEARNING OF FNN MODELS: TRANSFER FROM 2A4M TO 179BUS SYSTEM

System	Before retraining	After retraining
2A4M	100%	84.28%
179Bus	39.57%	90.2%

TABLE IV
ACCURACY OF TRANSFER LEARNING OF FNN MODELS: TRANSFER FROM 179BUS TO 2A4M SYSTEM

System	Before retraining	After retraining
179Bus	100%	94.64%
2A4M	55.2%	81.51%

A similar transfer learning process is applied to the CNN models. One difference of CNN models is that the input dimension is different for two simulation systems. Thus, the input layers need to be replaced and retrained, and the retrained CNN model cannot be applied directly back to the original training system. During the retraining process, the learning rate of the inherited network is set to 0.001 and the maximum number of epochs is set to 5 so that the inherited network is frozen. The learning rate of other parts is set 20 times larger.

TABLE V
ACCURACY OF TRANSFER LEARNING OF CNN MODELS

Training System	Retraining System	Accuracy
2A4M	179Bus	99.87%
179Bus	2A4M	98.57%

The result of the CNN models is summarized in Table V. The high accuracy demonstrates the outstanding performance of retrained CNN models.

IV. CONCLUSION

Machine learning techniques are applied to the area of oscillation classification, including decision tree, support vector classification, feedforward neural networks, and convolutional neural networks. Augmentation is adopted to deal with the problem of identification of the beginning point of oscillation events. To overcome the problem of lack of data, a transfer learning approach is proposed to reduce the need for samples.

Future research includes to test the proposed approaches in real systems and extend them to other applications, including forced oscillation source locating.

ACKNOWLEDGMENT

This work was funded by SGCC Science and Technology Program under project "AI based oscillation detection and control" and contract number SGJS0000DKJS1801231.

REFERENCES

[1] Pacific Northwest National Lab, "Power System Oscillatory Behaviors: Sources, Characteristics, & Analyses," Tech. Rep. PNNL-26375, PNNL, 2017.

[2] D. Bian, Z. Yu, D. Shi, R. Diao, and Z. Wang, "A real-time robust low-frequency oscillation detection and analysis (lfoda) system with innovative ensemble filtering," *Accept by CSEE, available at arXiv preprint arXiv:1812.11266*, 2018.

[3] Y. Ma, W. Liu, and S. Zhao, "On-line identification of low-frequency oscillation properties based on envelope fitting," *Automation of Electric Power Systems*, vol. 38, no. 23, pp. 46–53, 2014.

[4] J. Liu, W. Yao, J. Wen, H. He, and X. Zheng, "Active power oscillation property classification of electric power systems based on svm," *Journal of Applied Mathematics*, vol. 2014, 2014.

[5] R. Xie and D. Trudnowski, "Distinguishing features of natural and forced oscillations," in *2015 IEEE Power & Energy Society General Meeting*, pp. 1–5, IEEE, 2015.

[6] M. Ghorbaniparvar, N. Zhou, X. Li, D. J. Trudnowski, and R. Xie, "A forecasting-residual spectrum analysis method for distinguishing forced and natural oscillations," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 493–502, 2017.

[7] X. Wang and K. Turitsyn, "Data-driven diagnostics of mechanism and source of sustained oscillations," *IEEE Transactions on Power Systems*, vol. 31, no. 5, pp. 4036–4046, 2015.

[8] Y. Meng, Z. Yu, D. Shi, D. Bian, and Z. Wang, "Forced oscillation source location via multivariate time series classification," in *2018 IEEE/PES Transmission and Distribution Conference and Exposition (T&D)*, pp. 1–5, IEEE, 2018.

[9] C. Jiang, J. Liu, Y. Liu, J. Gou, C. Chen, R. Chen, and M. Bazargan, "Online forced oscillation detection and identification based on wide area measurement system and cell theory," *Electric Power Automation Equipment*, vol. 35, no. 2, pp. 125–132, 2015.

[10] J. Duan, H. Xu, and W. Liu, "Q-learning-based damping control of wide-area power systems under cyber uncertainties," *IEEE Transactions on Smart Grid*, vol. 9, no. 6, pp. 6408–6418, 2017.

[11] Y. Cui, Z. Yi, J. Duan, D. Shi, and Z. Wang, "A Rprop-Neural-Network-Based PV Maximum Power Point Tracking Algorithm with Short-Circuit Current Limitation," in *2019 IEEE Power Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pp. 1–5, Feb 2019.

[12] Z. Yi and A. H. Etemadi, "Line-to-line fault detection for photovoltaic arrays based on multiresolution signal decomposition and two-stage support vector machine," *IEEE Transactions on Industrial Electronics*, vol. 64, pp. 8546–8556, Nov 2017.

[13] S. Maslennikov, B. Wang, Q. Zhang, E. Litvinov, *et al.*, "A test cases library for methods locating the sources of sustained oscillations," in *2016 IEEE Power and Energy Society General Meeting (PESGM)*, pp. 1–5, IEEE, 2016.

[14] P. Kundur, N. J. Balu, and M. G. Lauby, *Power System Stability and Control*, vol. 7. McGraw-Hill New York, 1994.

[15] Powertech, "DSATools," 2019. Available from: https://www.dsatools.com/tsat/.

[16] A. Tsimpiris and D. Kugiumtzis, "Clustering of oscillating dynamical systems from time-series data bases," in *Workshop Knowledge Extraction and Modeling*, pp. 4–6, 2006.