www.czasopisma.pan.pl PAN www.journals.pan.pl

INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2020, VOL. 66, NO. 4, PP. 655-664
Manuscript received November 14, 2019; revised October, 2020. DOI: 10.24425/ijet.2020.134024

A High-Accuracy of Transmission Line Faults (TLFs) Classification Based on Convolutional Neural Network

S. Fuada, H.A. Shiddieqy, and T. Adiono

***Abstract*—To improve power system reliability, a protection mechanism is highly needed. Early detection can be used to prevent failures in the power transmission line (TL). A classification system method is widely used to protect against false detection as well as assist the decision analysis. Each TL signal has a continuous pattern in which it can be detected and classified by the conventional methods, i.e., wavelet feature extraction and artificial neural network (ANN). However, the accuracy resulting from these mentioned models is relatively low. To overcome this issue, we propose a machine learning-based on Convolutional Neural Network (CNN) for the transmission line faults (TLFs) application. CNN is more suitable for pattern recognition compared to conventional ANN and ANN with Discrete Wavelet Transform (DWT) feature extraction. In this work, we first simulate our proposed model by using Simulink® and Matlab®. This simulation generates a fault signal dataset, which is divided into 45.738 data training and 4.752 data tests. Later, we design the number of machine learning classifiers. Each model classifier is trained by exposing it to the same dataset. The CNN design, with raw input, is determined as an optimal output model from the training process with 100% accuracy.**

***Keywords*—fault detection, fault classification, transmission lines, convolutional neural network, machine learning**

I. Introduction

THE power system transmission consists of three main parts, i.e., a transmission line, distribution line, and load. This system delivers electrical power from power generator to various loads including homes and industries. Power system transmission becomes more complex since new power resources have been implemented massively worldwide as nowadays, which is renewable energy generators. Many small renewable energy generators have been utilized instead of conventional power generator. The system transmission employing the renewable energy generators can be managed under "smart grid" scheme.

System failure in the power system transmission makes major blackouts. Therefore, the unstable power system must be anticipated. The power system is a nonlinear system that operates in a continually changing environment; loads, generator connect-disconnected and operating parameters that evolve [1]. A rapid protection scheme is highly needed to ensure the stability and sustainability of the power system. Commonly, the protection scheme uses fault detection and classification fault methods. These methods are essential things to avoid power system failure. When the fault occurs in the power system, the area containing an error (fault) will be isolated from the entire system.

The general protection method uses a Bus bar with a specific calculation. When the detected fault exceeds the determined fault rate, it will disconnect the faulted area. To detect as well as cut-off (disconnect) the faulted zone, we need a particular algorithm embedded on a microprocessor-based relay or other devices that can measure three-phase voltages and currents [2]. In previous work [3], we have proposed a protective system based on automatic relay integrated with fault detection algorithm. It has been used to perform a disconnect fault area from a whole system. This FPGA-based digital protective relay can directly detect a fault area without automatically, it means, the proposed system does not necessary to send measurement data to the control center. Using this onsite protection scheme, the overall system response becomes faster, cost-efficient, low-cost maintenance budget. However, in [3] can only detect the fault. To create a robust, accurate, and efficient TL system, a "fault classification" method must be applied. Thus, the TLFs can be mapped as well. Related studies, the protection scheme employs artificial intelligence (AI) to detect as well as classify the fault [4].

Machine learning can be applied to analysis the occurred fault in the system. It has the ability to learn from training data so the power transmission system can find the fault location and classify the fault correctly. Moreover, machine learning can locate the fault direction [5]. We must consider many parameters of the fault classification. The machine learning capabilities can adapt these considered parameters by re-training the new data (pre-trained model). Machine learning-based fault classification is commonly used Artificial Neural Network (ANN) jointly with various extraction features, such as Discrete Fourier Transform (DFT) [6], Fast Fourier Transform (FFT), and Discrete Wavelet Transform (DWT) [7-

This work was supported by Program Penelitian, Pengabdian kepada Masyarakat Inovasi (P3MI) in 2017.

First Author is with program Studi sistem telekomunikasi, Universitas Pendidikan Indonesia, Indonesia (e-mail: syifaulfuada@upi.edu).

Second and third Author is with the University Center of Excellence on Microelectronics, Instiut Teknologi Bandung, Indonesia (e-mail: syifaulfuada@pme.itb.ac.id, tadiono@stei.itb.ac.id).

10]. The obtained data from extraction features are used to find fault by comparing the determined threshold. Later, the filtered data is used to train the machine learning model [11]. The difference point between CNN and ANN is the neuron connection (neuron network) on the system.

Many of the practical applications of machine learning today use linear classifiers over hand-engineered features [12]. However, it is difficult to choose parameters and algorithms to obtain the best result. The machine learning algorithm consists of many hyperparameters that have different functions. The main purpose of determining hyperparameter manually is to find the model complexity that is useful for completing a task. By using the appropriate configuration, the training cost will be minimized. In addition, time to change the complex function to simpler functions (without reducing model performance) can be less. A practical way to find a suitable model is to increase the model capacity for certain training data continuously, so that an adequate final result can be obtained. This method can make the values not converge because of difficulty in optimization (failed training process). This problem will rarely occur if the model or algorithm is chosen correctly [13].

The convolutional neural network (CNN)-based algorithm can recognize a pattern accurately. Various researchers used it for pattern classification purposes. CNN uses signal convolution with filter/kernel [12]. This method requires minimal engineering by hand but needs a large amount of dataset and computing power.

In this paper, we propose a method to detect and classify the fault without feature extraction and human tuning. We use the large dataset for training purpose to get a model that can represent the data. In this work, high accuracy classification in validation data reaches 100%.

II. METHODS

Fig. 1 shows the proposed method, it can be divided into two main steps: we first simulate the model to create datasets and then design the classifier for machine learning model.

Fig. 2 depicts the conventional model of ANN for TLFs application, it has three parts: Input, Hidden layer, and Output. The input consists of three-phase voltages (`Va, Vb, Vc`) and currents (`Ia, Ib, Ic`). Then Hidden layer, and the last stage is output layer. There are 11 fault types, i.e., AG, BG, CG, AB, AC, BC, ABG, ACG, BCG, ABCG, and NON_Fault. The operation of ANN is not allowing the network to be deeper. It happens because of each neuron in the ANN architecture is connected to every other neuron. Then CNN was introduced CNN is similar to most neural network architecture, which is made of neurons that can be trained to gain "weight" and "bias," according to the specification and application. The neurons of CNN are fully-connected to each other. CNN architecture mainly consists of three layers: (1) Convolutional layer, (2) Pooling layer, and (3) Fully-connected layer (neural network) [12]. CNN may contain one or more layers of these three layers (convolutional, pooling, or fully-connected) with regards to the needs. The CNN technique is suitable for applications with intrinsic structures data.

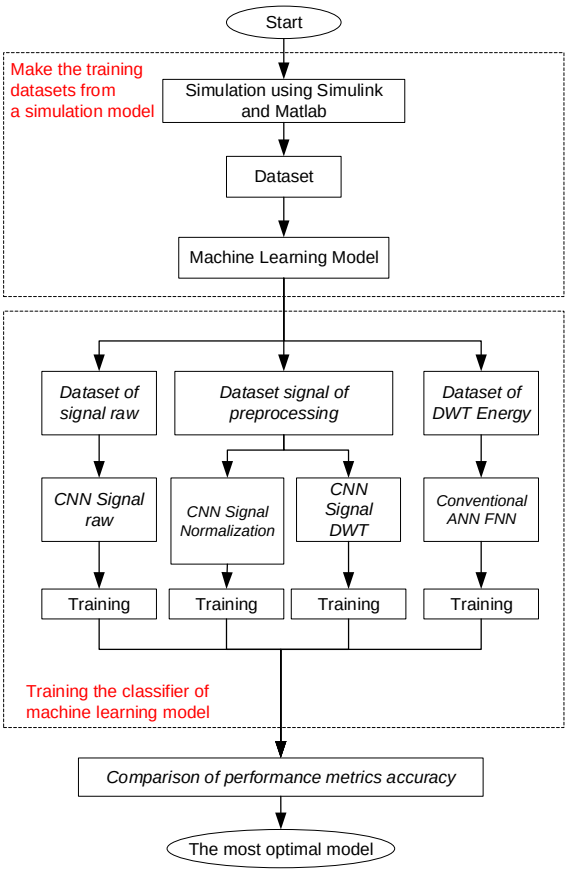

Fig.1. Flowchart of the proposed method

.

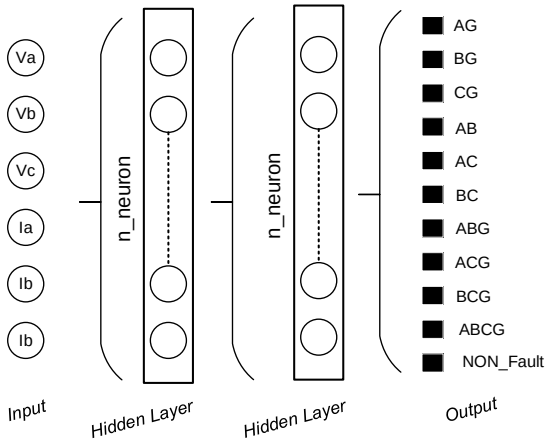

Fig.2. Conventional model of Wavelet Energy ANN

In this work, we used Simulink simulation and the Matlab approach. Through this fault simulation modeling, we obtained datasets that can represent faults on the TL. By using the classifier in machine learning system, we can detect as well as classify the fault types on transmission lines. Finally, we obtained the best performance of the machine learning model. The proposed can classify the test data with >95% accuracy.

CNN is designed to process the entering data in the form of arrays. CNN can be classified into three types: (a) 1-D type for signals and sequential data; (b) 2-D type for audio images and

spectrograms, and (c) 3-D type for video or volumetric images. There are four main things to consider in creating a model from CNN, i.e., local connections, joint weights, collections, and number of layers [13].

In this paper, practical development for machine learning are used, the process is as follows: 1) select the metrics; 2) select the initial models; 3) development step; 4) select the hyperparameter, 5) select the needed data set, and 6) debug the model. The metric expected from our modeling can reach 95% accuracy level in the test data. Hopefully, the results of this training process will be better than the ANN classification model. We modified the model as in Fig. 2 to obtain the most optimal model of machine learning for TFLs application.

B. Power System Model

In this paper, we simulate the model suggested by IEEE Std C37.114-2004, as depicted in Fig. 3. With this power system model, we can determine the fault types as well as fault location. The technique used is one-ended impedance-based measurement techniques. The faults calculation can be done by observing the apparent impedance at the last point of the line transmission. All faults types detection must be done by measuring the voltage and current in each phase of the single line diagram.

The power system model, as in Fig 3, is widely used for generating fault datasets for machine learning system [15] or ANN [16]. However, the power system is susceptible to a fault because of the fault resistance and load current change. Load current variations occur because of the dynamic load factor in several locations between two power generator sources. In this work, we used a power model using Matlab. It can compute the variations changes in load current over time.

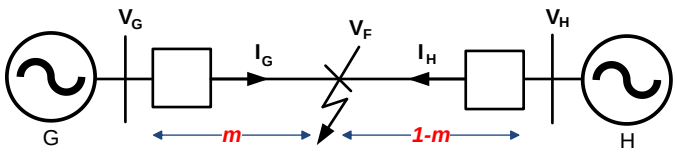

Fig.3. Power system model suggested by IEEE std C37.114-2004 [2]

A single line diagram, as depicted in Fig 3, is then simulated by using the Simulink software to get a fault signal. Table I shows the parameters used for power system simulation, including the value of Generator I and II, Transformer I and II, Load I to IV, Multimeter, Ts, and frequency standard used in Indonesia (50 Hertz). Later, we vary the input of the power system based on Table II, including fault location, resistance, and inception angle. Moreover, pre-fault angle and fault types have been varied. The faults on the output voltage and three-phase currents are then stored in the dataset, and it will be named according to the fault label. This process is repeated continuously until all possible combinations of parameters have been carried out (Fig. 4).

The fault type parameters as shown in Table III, are used as an input variable to the simulation models. In this work, we only used only one full-wave signal stored in the datasheet, even though there are more than one wave signals in the simulation. This work correlates with our previous work[17]. However, in [17], we focus on the effect of sampling variation in accuracy for TLs classification. We recommend the efficient sampling rate in the power measurement is 16700 Hz (334 samplings/signal).

TABLE I
PARAMETERS OF POWER SYSTEM SIMULATION

Power model	Value
Generator 1 & Generator 2	13.8 kV & 735 kV
Transformer 1	13.8/735 kV
Transformer 2	735/230kV
Load 1 & Load 4	100 MW
Load 2 & Load 3	330 Mvar (Reactive)
Multimeter	V (pu), I (pu/100MVA)
Ts	5e-05 s
Freq	50 Hz

TABLE II
PARAMETER VARIATION IN SIMULATION MODEL

Model parameter	Variation	
	Train data	Test data
Fault Location (km)	20, 40, 60, 80, 100, 120, 140, 160, 180, 200, 220, 240, 260, 280	50, 70, 90, 110, 130, 150
Fault Resistance	5, 10, 15, 20, 25, 30, 40, 50, 60	5, 15, 30, 50
Fault Inception angle (degrees)	0, 30, 60, 90, 120, 150, 180, 210, 240, 270, 300, 330	30, 60, 120, 180, 240, 300
Pre-fault angle (degrees)	10, 20, 30	10, 20, 30
Fault Type (IEEE C37.114-2004)	AG, BG, CG, AB, AC, BC, ABG, ACG, BCG, ABCG, NON_Fault	AG, BG, CG, AB, AC, BC, ABG, ACG, BCG, ABCG, NON_Fault

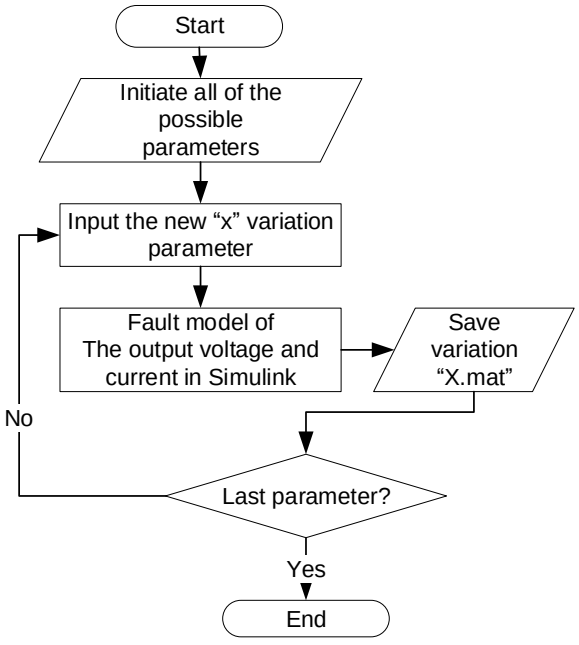

Fig.4. Flowchart of the dataset generation

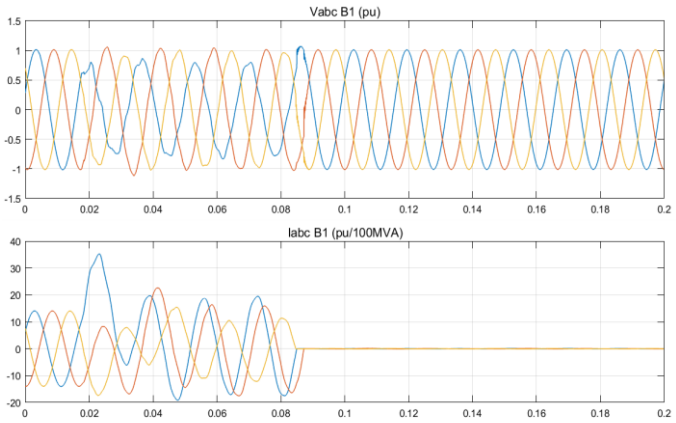

Fig.5. Dataset for "AG" fault type (captured on Simulink)

Fig. 5 illustrates a sample of the obtained dataset. The dataset is then reprocessed and stored in the Numpy array (`*.npy`) format by removing the fault parameter label. Numpy arrays are composed of the array one signal data with six channels sequential of voltage and current, the fault sequence is as follows: [`Va Vb Vc Ia Ib Ic`]. Then followed by array two labels fault data with binary format, as an example, no-fault type format is as follows [`0 0 0 0 0 0 0 0 0 0 1`].

A binary label with an array length of 11 is then composed. The number of fault types have generated of the sequence format, i.e., AG, BG, CG, AB, AC, BC, ABG, ACG, BCG, ABCG, and NON_Fault as in Table II. In total, there are eleven labels represents 10 types of faults:

- Line-to-Ground (LG) category, i.e., AG, BG, CG
- Line-to-Line (LL) category, i.e., AB, BC, AC
- Line-to-Line-to-Ground (LLG) category, i.e., ABG, ACG, BCG
- Line-to-Line-to-Line-to-Ground (LLLG), i.e., ABCG
- NON_Fault category.

Fig. 5 shows the power model (single line diagram) used in this work refers to Fig. 3, while Fig. 6 visualizes the system block under Simulink simulation. The value of each block refers to Table II.

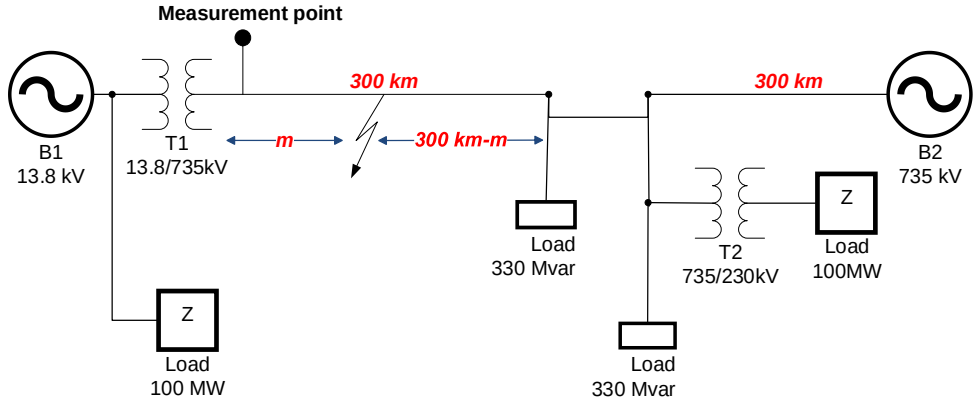

Fig.5. Single line diagram proposed power system model

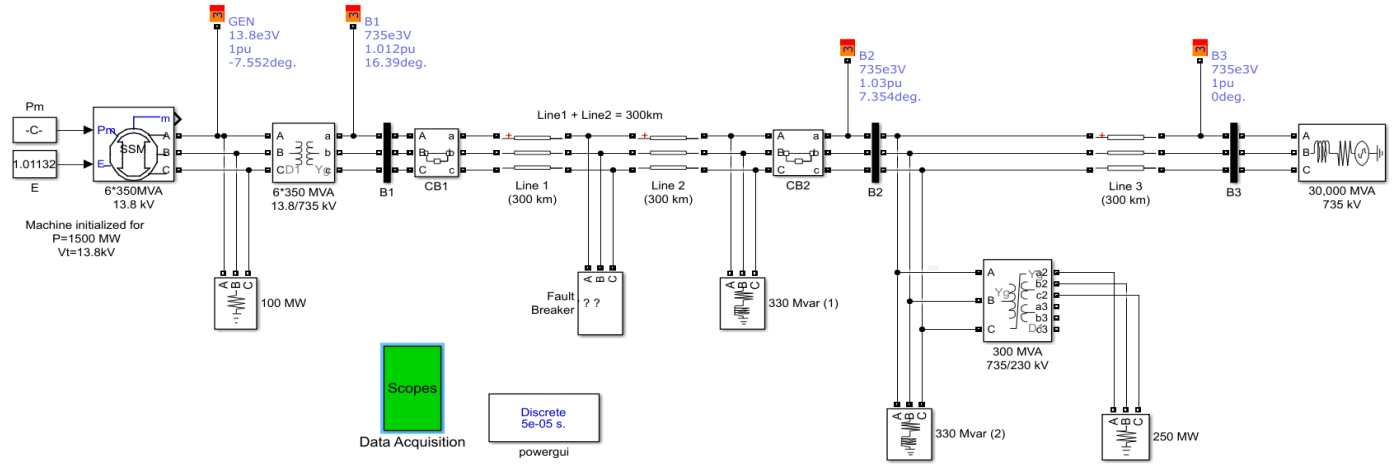

Fig.5. Three-phase series fault network in Simulink. The measurement is carried out in three nodes, i.e., Generator1 (B1), middle (B2), and (B3). However, in this work we used single terminal, that is G1 voltage and current and it will be stored as training data. We get "AG" fault data.

C. *Preprocessing*

Preprocessing data is made so that the dataset feature becomes simpler. By reducing the less essential data, the machine learning model can be more straightforward. The data input from preprocessing can eliminate the feature data that is less important for the classification process. Hence, the classification model can be made simpler.

The normalization of datasets is commonly used in many machine learning. The process is by updating the normalized new data with the normalize x algorithm as Eq.1. The Matlab library used for

normalization process is based on arrays or vectors. Preprocessing normal results have the same size as the raw inputs.

Wavelet is a process that converts data from the time domain to the frequency domain without erasing all data in the time domain [14]. Furthermore, the use of wavelets with multi-resolution analysis can be used to divide signals at high frequencies because high-frequency signals tend to have more patterns. The signal is separated utilizing a pass filter with the configuration in Table III.

$$\frac{(x - \text{np.mean}(x))}{(\text{np.max}(x) - \text{np.min}(x))} \quad (1)$$

Note:
`np`: number of data
`np.mean`: the average of all data on the 0 axes
`np.max`: the maximum values for each channel
`np.min`: the minimum values for each channel

The multi-resolution analysis makes data in the time domain still exist at each frequency. The results of Multi-resolution DWT become the input of various types of algorithms used to classify faults. The `cA4` value can be an ANN input to get the fault type [10], or by using wavelet energy from `cD1`/`cD4` signals. Energy wavelets are used as an input of the ANN classifier model to classify faults.

The Daubechies 4 (`db4`) can be selected as Mother wavelet because of provides better accuracy than other mother wavelets. Besides, it was commonly used for transient analysis in power system application [18]. The results of `db4` are accurate enough to get fault signals [11].

TABLE III
MULTI-RESOLUTION ANALYSIS OF DWT LEVEL 4

Level	Frequency	Sample
4	0 to fn/16	27 cA4
	fn/16 to fn/8	27 cD4
3	fn/8 to fn/4	54 cD3
2	fn/4 to fn/2	88 cD2
1	fn/2 to fn	170 cD1

Fn = Sampling / windows (frequency)
cA4 = Approximation coefficient 4
cD4 = Coefficient Detail 4
cD1 = Coefficient Detail 1

D. CNN Model

As described in Fig. 1, after the dataset have been generated, we design machine learning. In this work, we first used the default model of CNN as depicts in Fig. 6. The design step starts from the simplest model and then develops into a larger network. The training process and its implementation of classification are using *Python 3.5* programming with the help of the *Tensorflow™* platform.

Tensorflow™ is a machine learning library made by Google used for numerical operations based on graphs. After the graph has been created, the session is then developed and executed. Later, the results are distributed on the CPU and GPU. The main components in *Tensorflow™*: 1) *Variable*: the session value used for weight and bias; 2) *Nodes*: arithmetic operations; 3) *Tensor*: signal passing from a node or signal passing to a node; 4) *Placeholder*: used to send data between our program and *Tensorflow™* graphs; and 5) Session: the point where the graph is run.

Preprocessing is done so that the dataset features are simpler and can be completed with a simpler CNN class-file model. The preprocessing method is presented in Section II.C.

Fig. 8 shows a default CNN, *n_filter* denotes the number of filters, *CNN Layer* is layers of the convolutional, pooling layer, and neural network, *Conv_1D* denotes a layer convolutional 1D, *n_neuron* denotes the number of hidden units in fully-connected layer.

The CNN model is then variated into four categories depending on its input, i.e.,

- Model with raw input (using architecture as in Fig. 8),
- Preprocessing normalization,
- Preprocessing DWT, and
- Energy DWT with ANN.

Fig. 9(a) is a CNN model with preprocessing normalization while Fig. 9(b) depicts a CNN model with preprocessing DWT. The Energy DWT with ANN used architecture as in Fig. 2. The performance of four models are then compared. Table IV shows the requirement of mentioned models.

TABLE IV
MODEL VARIATION OF MACHINE LEARNING

Input layer	CNN network variation	fc network
Raw CNN	1-5 layers	16-1024 fc
Pre – Normalization CNN	1-5 layers	16-1024 fc
Pre – MRA DWT CNN	1 layer	12-32 fc
Energy DWT – ANN	-	2- 3 layer

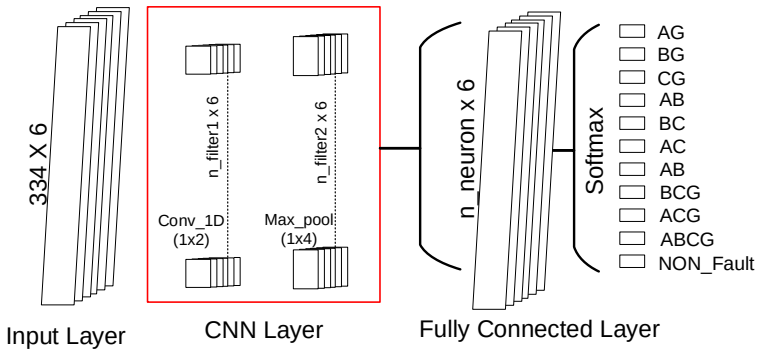

Fig 8. The default of the CNN model, A collection of these three layers (convolutional, pooling, and fully-connected) can form an architecture called CNN

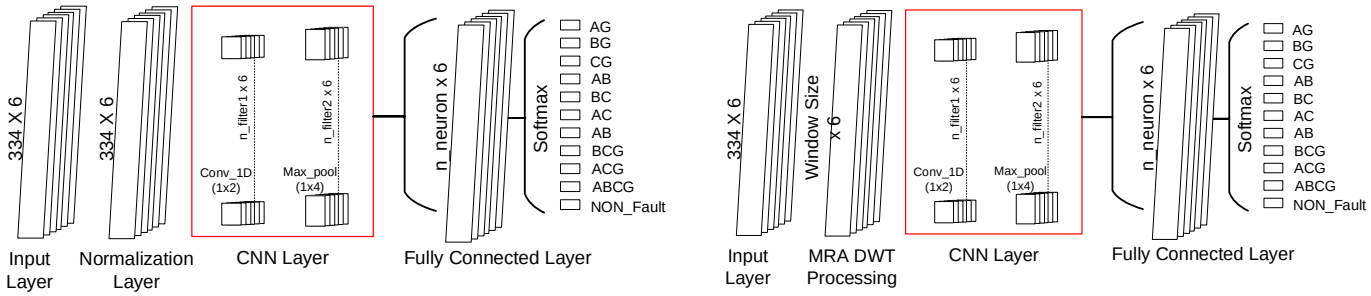

Fig 8. The modified CNN model: (a) CNN with preprocessing normalization dataset; (b) CNN with preprocessing MRA DWT dataset

TABLE V
HYPERPARAMETERS OF CNN

Layer	Input	Filter size	Stride	Num Filter	Activation function	Out
Convolution	334 × 6	2	1 same padding	n_filter*	Relu	334 × n_filter
Maximum Pooling (max_pol)	334 × n_filter	2	None padding same	Same as Convolution	-	167 × n_filter
fc1	Last max_pool	N/A			Relu	fc**
fc2	fc**				Softmax	11 class

334 × 6 means 334 samplings/signal with 6 signal channels. These signals represent multidimensional arrays 334 × 6 in size, and structure of [`Va Vb Vc Ia Ib Ic`]

The hyperparameter of the CNN model variation is set as following: input size `334 × 6`, maximum pool unit same as number of convolutional kernel unit, `2 × 1` filter size, and 1 same padding. Each convolution unit using active Rectified linear unit **(**Relu) function, and in the fully-connected (fc) layer uses activation function Softmax with 11 classes output. The Hyperparameter is listed in Table V.

The CNN model with Preprocessing DWT used MRA DWT D1, D4, and A4. While ANN model used Energy DWT D1 and D4. However, we used a narrow ANN model with fewer inputs and layers, hence the ANN can classify fault types with low-computation. This ANN model is then compared to CNN-based machine learning.

III. RESULTS

A. *Dataset generation*

Table VI shows the detailed of the RAW dataset; it produces 11 types of faults. Table VII shows the preprocessing normalization result. The db4 is selected as Mother Wavelet according to Ref. [11] recommendation. Multilevel 1-D is used because of this application for signal analysis. We select frequency sampling with sample `170 cD1` as in Table III.

TABLE VI
RESULT OF RAW DATASET

Parameters	Description
Sampling frequency	16,700 Hz
Number of samples in one signal	334 sampling
Voltage signal	3 phases
Current Signals	3 phases
Number of Label faults	11 types

TABLE VII
RESULT OF PREPROCESSING NORMALIZATION

Parameters	Description
Mother Wavelet	db4
Frequency sampling	fn/2 to fn
Multiresolution analysis	Multilevel 1D level 4
Frequency band	0 – fn/16
Number of sampling per window	334

The dataset is created from all combinations of existing training parameters, and it produces dataset with 45,738 training data and 4,752 test data. The simulation results are in the form of an extension (`*.mat`) file with file names according to each parameter. Data is converted into several preprocessing and raw data files for training and testing, with labels and randomized sequences.

In this work, we used two preprocessing units, first is Normalization as in Table VII, the second one is Multi-resolution Analysis Discrete Wavelet Transform (MRA DWT) with parameters as in Table VIII. The fault signals are generated using algorithm as follow [19]:

Algorithm for fault generation

```
Fault length parameter = {}
Fault resistance = {}
Fault inception = {}
  for i = 1 to fault length parameter
    for i = 1 to fault resistance parameter
      for i = 1 to fault inception parameter
        input (combination of parameter)
        run Simulink® power simulation
        save (pre-fault + fault signal
measure)
repeat for every combination
```

TABLE VIII
RESULT OF PREPROCESSING DWT

Parameters	Description
Sampling frequency	16,700 Hz, according to Ref. [17]
Number of samples in one signal	334 sampling
Signal channel	6 Channels
Angle normalization	0 between channel
Normalization method	Zero mean and unit variance

Fig. 8 visualizes a sample of data comparison between signal raw, preprocessing MRA DWT for dataset. In this sample, we used only two fault types due to limited space, that is BC type and ABG type. The method of dataset generation detail is elaborated in other paper [19].

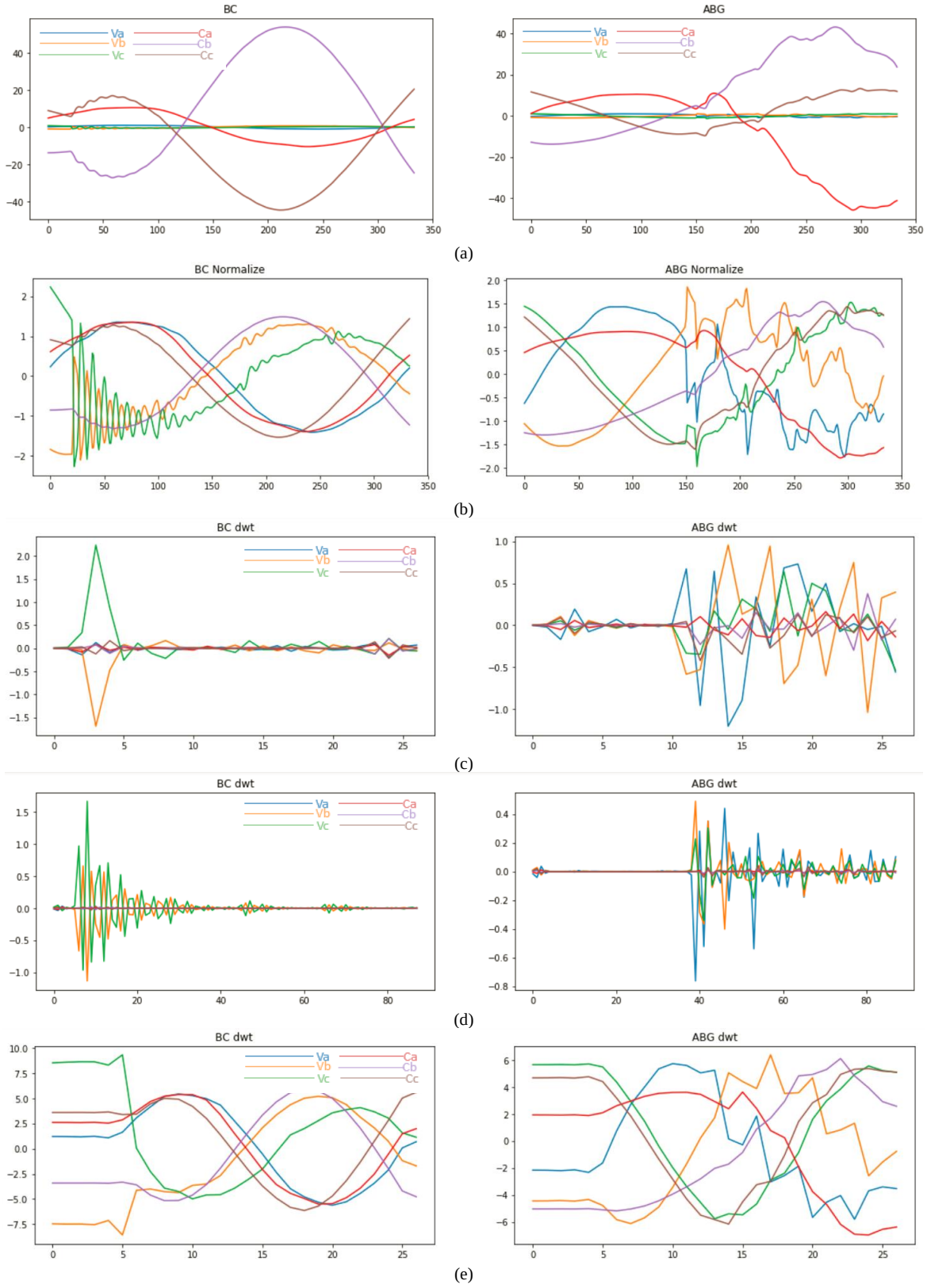

Fig 8. Generated dataset of BC and ABG fault types for various input signal: (a) RAW data; (b) Normalization; (c) MRA DWT D4; (d) MRA DWT D1; and (e) MRA DWT A4

MRA DWT produces a signal, which is a coefficient of each frequency distribution. The coefficient signal used is cD4, cD1, and Approximation 4 (A4). The width of each window is 170 and 27. On the results of MRA DWT D4 signal, the frequency and time data do not disappear completely. But the output signal can be used to represent a pattern with 27 sample results for each window. This technique is one feature extraction to form the new signals that represent frequency and time domains.

Reduction of input sampling from 334 to 27 makes the CNN classifier model is more straightforward. We can also reduce the number of layers or other techniques to reduce its capacity. This technique makes the training process, and classification is faster. As the ANN model, extracted feature Energy DWT D1 is used. DWT D1 signal is discrete data, each window has 170 samplings with 6 channels. All samples will be summed up by Eq. 2,

$$E = \sum_{n1}^{n1+170} |x(n)|^2 \quad (2)$$

Note:
E: Energy wavelet
n: Sampling

B. *Training Process*

Each model will have a trainable parameter that represents the capacity of the model. The higher the model capacity, the greater the ability to represent datasets, but models with large sizes require longer preprocessing times. Moreover, it is easier to overfit in the training data. Overfitting is a situation where the classification model is able to have high-accuracy on work, each machine learning model is named according to its training data but fails or low accuracy on new data. In this element architecture. The format of model is expressed as Eq.3,

```
Name model = {Machine learning classifier
model}_{input of preprocessing}_{architectural
parameters}
```
(3)

For example, name of RAW input is `CNN_raw_nfilter8-16-drop0.8-fc.model`, means a "CNN" model with "RAW" signal input and the CNN layer architecture consists of two layers with "8" filters on the 1st slayer, then "16" filters on the 2nd layer with dropout of "0.8", and 32 hidden unit in the fully-connected layer.

Preprocessing is placed between the input of the machine learning system and the CNN layer. The difference of the preprocess classifier model with the RAW classifier model is the addition of the normalization (DWT layer). It is involved between the input and the CNN layer.

Afterward, performance comparisons were made for each classifier of the machine learning model. The most optimal model is a model with the fewest number of parameters, small capacity, but it still represents training data with the highest accuracy in the test data.

Table IX lists the training data result from various models, that is: the CNN model with the RAW dataset, preprocessing normalization, MRA DWT preprocessing and ANN model by extracting Energy DWT features.

TABLE IX
COMPARISON RESULTS FOR EVERY EXPERIMENTS

Preprocessing (input size)	Number of filters in the CNN network	Number of Neuron in the fully-connected layer	Total Parameter	Accuracy in training data	Accuracy in test data	Loss data during test validation
Raw	4	16	10,934	0.9139	0.9276	0.22366
Raw	17	144	410,776	0.9231	0.9386	0.19126
Raw	17	1024	2,919,656	0.9349	0.9697	0.17727
Raw	32	1024	5,484,971	0.932	0.969	0.18859
Raw	96	1024	16,430,315	0.9376	0.939	0.2088
Raw	4-8	16	11,079	0.9375	0.9346	0.21784
Raw	**17-32**	**32**	**87,752**	**0.9141**	**1**	**0.19306**
Raw	17-32	144	390,152	0.9487	0.9697	0.17512
Raw	17-32	1024	1,475,387	0.9264	0.9697	0.18419
Raw	4-8-17-32	32	23,432	0.9304	0.9695	0.21113
Raw	8-17-32-72	144	225,660	0.9487	0.9697	0.18826
Raw	17-32-72-144	144	464,096	0.9508	1	0.18831
Raw	17-32-72-144	1024	3,135,776	0.9475	0.9697	0.17398
Normalization	4	16	10,934	0.9139	0.9276	0.22366
Normalization	17	144	410,776	0.9453	0.9394	0.18919
Normalization	17	1024	2,919,656	0.9503	0.9697	0.18055
Normalization	32	1024	5,484,971	0.9341	0.9697	0.18415
Normalization	96	1024	16,430,315	0.9403	0.9697	0.17939
Normalization	4-8	16	11,079	0.9069	0.9318	0.23383
Normalization	**17-32**	**32**	**87,752**	**0.9356**	**1**	**0.18661**
Normalization	17-32	144	390,152	0.9373	0.9697	0.18165
Normalization	17-32	1024	2,766,152	0.9408	0.9697	0.18058
Normalization	4-8-17-32	32	23,432	0.9268	0.9394	0.19107
Normalization	8-17-32-72	144	225,660	0.9302	0.9394	0.19626
Normalization	17-32-72-144	144	464,096	0.9365	0.9695	0.17436
Normalization	17-32-72-144	1024	3,135,776	0.9402	0.9689	0.1893

CONTD. TABLE IX
COMPARISON RESULTS FOR EVERY EXPERIMENTS

Preprocessing (input size)	Number of filters in the CNN network	Number of Neuron in the fully-connected layer	Total Parameter	Accuracy in training data	Accuracy in test data	Loss data during test validation
MRA DWT D1 (170)	7	16	9,814	0.7707	0.7498	0.74988
MRA DWT D1 (170)	17	32	46,856	0.9086	0.8664	0.8664
MRA DWT D1 (170)	17	144	210,040	0.9224	0.8887	0.38701
MRA DWT D1 (170)	49	144	602,136	0.9367	0.8611	0.44501
MRA DWT D1 (170)	17-32	32	45,768	0.907	0.943	0.32229
MRA DWT D1 (170)	17-32	144	201,224	0.9274	0.8843	0.30325
MRA DWT D4 (27)	2	12	517	0.3925	0.4114	1.61717
MRA DWT D4 (27)	4	16	1,151	0.7689	0.8011	0.54734
MRA DWT D4 (27)	4	24	1,695	0.7878	0.8401	0.43306
MRA DWT D4 (27)	16	32	7,771	0.9209	0.9501	0.24681
MRA DWT **D4 (27)**	**16**	**144**	**34,203**	**0.9344**	**0.9611**	**0.20475**
MRA DWT A4 (27)	2	12	517	0.8329	0.8537	0.33223
MRA DWT A4 (27)	4	16	1,151	0.8632	0.888	0.27673
MRA DWT A4 (27)	4	24	1,695	0.9133	0.8929	0.24619
MRA DWT A4 (27)	16	32	7,771	0.9299	0.9346	0.2104
MRA DWT A4 (27)	16	144	34,203	0.9277	0.9367	0.1085
Energy DWT D1	24		443	0.7215	0.7548	0.80313
Energy DWT D1	6-6		161	0.6456	0.6199	0.6456
Energy DWT D1	12-12		383	0.6374	0.7186	0.86201
Energy DWT D1	24-24		1,043	0.8001	0.7639	0.80014
Energy DWT D1	6-12-6		281	0.4854	0.4493	1.49282
Energy DWT D1	12-24-12		839	0.5832	0.5972	1.29485
Energy DWT D1	24-24-24		1,643	0.6764	0.653	1.02303
Energy DWT D4	24		443	0.6658	0.7553	0.73453
Energy DWT D4	6-6		161	0.6833	0.733	0.84699
Energy DWT D4	12-12		383	0.6711	0.7068	0.7637
Energy DWT D4	24-24		1,043	0.6648	0.7466	0.74744
Energy DWT D4	6-12-6		281	0.6332	0.7315	0.82771
Energy DWT D4	**12-24-12**		**839**	**0.6648**	**0.8076**	**0.7555**
Energy DWT D4	24-24-24		1,643	0.6635	0.7622	0.70488

TABLE X
COMPARISON RESULTS FOR THE BEST DATA FROM EVERY EXPERIMENTS

Model Name	Preprocessing	Number of filters in the CNN network	Number of Neuron in the fully-connected layer	Total Parameter	Accuracy in test validation
ANN	Energy DWT D4 (6)	N/A	12-24-12	839	80.764%
CNN	MRA DWT D4 (27)	16	144	34,203	96.11%
CNN	Raw (334)	17-32	32	87,752	99.99%
CNN	Normalization (334)	17-32	32	87,752	100%

From each model, we choose the best one and then select the most optimum model. The chosen model is `CNN_norm_nfilter17-32-fc32.model`, in line with the highest accuracy and the least capacity parameter. Even though CNN with preprocessing RAW also has a model that reaches almost 100% accuracy (99.99%), but the capacity is almost similar to the raw input. The model that has the same raw input can provide a more responsive system or faster classification. Hence for this reason, we select CNN with processing normalization. The least trainable parameters are in ANN; the accuracy is only 80% less than the other CNN models, which on average, produces accuracy above 90%. The preprocessing method requires computational resources, resulting in increasing the response time when the system is implemented. That is why the CNN model with MRA DWT is less accuracy than CNN with RAW data as well as CNN with normalization.

CONCLUSION

In the transmission line, the short current circuit must be quickly solved to avoid the fail of generation synchronism and power system blackouts. Machine learning technology has been widely applied in various consumer products today. This technology is able to form a classifier model without hardcode. But through a training process, making it suitable for fault classification applications that have varied fault patterns. In this paper, we generate a fault dataset in the power system using Simulink and Matlab. Faults are categorized into 11 types. Based on the test, the CNN-based machine learning model capable of detecting and classifying faults on transmission lines with 100% accuracy in data validation. A combination of the preprocessing blocks can improve the CNN performance with lower complexity, then it can be used to classify data tests. The most optimal model with the least number of parameters and with the best accuracy will be implemented in hardware.

ACKNOWLEDGEMENT

We would like to thank our institution for supporting this work through MP3I scheme. We also thank Dr. Permata Nur Miftahur Rizki and Dr. Farkhad Ihsan Hariadi for their support in reviewing this text.

REFERENCES

[1] P. Kundur, et al., "Definition and classification of power system stability IEEE/CIGRE join task force on stability terms and definitions" 19(3), pp. 1387–1401, 2004.

[2] IEEE Guide for Determining Fault Location on AC Transmission and Distribution Lines, Vol. 2014. [Online], Available at: https://doi.org/10.1109/IEEESTD.2015.7024095.

[3] H.A. Shiddieqy, et al., "Implementations Fault Detection Phasor Measurement Unit Using Zybo SoC," *Proc. of ISESD*, October 2017.

[4] S. Upadhyay, et al., "Fault classification and detection in transmission lines using ANN," *Proc. of ICIRCA*, 2018.

[5] A. Yadav and Y. Dash, "An Overview of Transmission Line Protection by Artificial Neural Network: Fault Detection, Fault Classification, Fault Location, and Fault Direction Discrimination" *Advances in Artificial Neural Systems*, 2014.

[6] K. Chen, et al., "Detection and Classification of Transmission Line Faults based on Unsupervised Feature Learning and Convolutional Sparse Autoencoder" *IEEE Transactions on Smart Grid*, Vol.9(3), pp. 1748–58, 2016.

[7] A. Abdullah, "Ultrafast Transmission Line Fault Detection using a DWT-based ANN." *IEEE Transactions on Industry Applications*, Vol. 54(2), pp. 1182–93, 2015.

[8] R.C. Mishra, et al., "Wavelet Based Transmission Line Fault Classification and Location," *Proc. of ISEG*, 2014.

[9] X.G. Magagula, et al., "Fault Detection and Classification Method Using DWT And SVM in A Power Distribution Network," *Proc. of IEEE PES PowerAffrica*, pp. 1–6, 2017.

[10] A.G. Saik and R.R.V. Pulipaka, "A New Wavelet Based Fault Detection, Classification and Location in Transmission Lines." *Int. J. of Electrical Power and Energy Systems*, Vol. 63, pp. 35–40, 2015.

[11] K.M. Silva, et al., "Fault Detection and Classification in Transmission Lines based on Wavelet Transform and ANN," Vol. 21(4), pp. 2058–63, 2006.

[12] Y. Lecun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, 521, pp. 436-444, May 2015.

[13] Standford, "CS231n Convolutional Neural Networks for Visual Recognition," [Online]. Available at http://cs231n.stanford.edu/.

[14] L.C. Lin, "A Tutorial of the Wavelet Transform," [Online]. Available at http://disp.ee.ntu.edu.tw/tutorial/WaveletTutorial.pdf

[15] P. Ray, and D.P. Mishra, "Support Vector Machine Based Fault Classification and Location of a Long Transmission Line" *Engineering Science and Technology, an Int. J.*, Vol. 19(3), pp. 1368–80, 2016.

[16] M. Patel and R.N. Patel, "Fault Detection and Classification on a Transmission Line Using Wavelet Multi Resolution Analysis and Neural Network." *Int. J. of Computer Applications*, 47(22), Vol. 27–33, 2012.

[17] H.A. Shiddieqy, F.I. Hariadi, and T. Adiono, "Effect of Sampling Variation in Accuracy for Fault Transmission Line Classification Application Based on Convolutional Neural Network," *Proc. of ISESD*, 2018.

[18] J. Klomjit, et al., "Comparison of Mother Wavelet for Classification Fault On Hybrid Transmission Line Systems," *Proc. of iCAST*, pp. 527–32, 2018.

[19] H.A. Shiddieqy, F.I. Hariadi, and T. Adiono, "Power Line Transmission Fault Modeling and Dataset Generation for AI based Automatic Detection," *Proc. of Int. Symp. On Electronics and Smart Devices* (ISESD), 2019.