Open Source Dataset Generator for Power Quality Disturbances with Deep-Learning Reference Classifiers

R. Machlev, A. Chachkes, J. Belikov, Y. Beck, Y. Levron

***Abstract*—In recent years power quality monitoring tools are becoming a necessity, and many studies focus on detection and classification of Power Quality Disturbances (PQD)s. However, presently a core obstacle that prevents the direct comparison of such classification techniques is the lack of a standard database that can be used as a benchmark. In this light, we propose here an open-source software which enables the creation of synthetic power quality disturbances, and is designed specifically for comparison of PQD classifiers. The software produces several types of standard disturbances from the literature, with varying repetitions and random parameters of the labeled disturbances, and includes two reference classifiers that are based on deep-learning techniques. Due to the good performance of these classifiers, we suggest that they can be used by the community as benchmarks for the development of new and better PQD classification algorithms. The developed code is available online, and is free to use.**

***Keywords*—power quality, harmonic distortion, PQD, public dataset, classification, classifier, deep-learning.**

I. Introduction

POWER quality is a measure of the degree to which voltage and current waveforms comply with established specifications [1]. Several power quality measures are harmonic distortions, variations in peak or RMS voltage, spikes and sags in voltages and currents, and variations in frequency [2]. In addition, many other different measures are mentioned in the recent literature, and used in practice [3].

In the last two decades, Power Quality (PQ) monitoring tools are becoming a necessity [4]. One reason for the popularity of such tools is the continuing integration of nonlinear generators and loads in power grids, most of them based on power electronics technology, which may inject high-order voltage and current harmonics into the grid [5], [6]. In this light, many recent studies focus on detection and classification of Power Quality Disturbances (PQD)s. Most of the proposed algorithms merge feature extraction methods that are based on signal processing techniques with classification methods which stem from the theory of machine learning. One familiar approach is to use the Wavelet transform with a support-vector machine classifier [7], [8]. Another well-known technique is to use the S-transform [9], which can be combined with different classifiers such as those based on artificial neural network [10], and parallel stacked sparse auto-encoders [11]. Other signal processing techniques that are being used are the fast Fourier transform [12] and sparse signal decomposition methods [13]. Moreover, in the last few years, with the evolution of Deep Learning (DL) techniques, better classifiers are now available in multiple fields and applications [14]. As part of this development, several works focusing on PQD classification such as [15], [16], [17], [18] have implemented deep learning techniques, which in certain cases outperform the traditional classification algorithms. A comprehensive review of recent techniques may be found in [19] and [20]. Survey [19] focuses on traditional algorithms of signal processing, feature extraction, artificial intelligence and optimization methods, while in [20] the focus is on classification of PQD in utility grids with high renewable energy penetration.

Presently a core obstacle that prevents the direct comparison of PQD classification techniques is the lack of a standard database that can be used as a benchmark [21]. While few public databases do exist, they are still in limited use since they were not specifically tailored for PQD classification. For instance, [22] is a data-set that shows only sag events, and [23] is a data-set that presents only transient events. As mentioned in [21], the available datasets are non-standard, and differ in their disturbance types, number of samples, data labels, and access methods.

Considering this challenge, we propose in this work an open-source software which enables the creation of synthetic power quality disturbances, and is designed specifically for comparison of PQD classification algorithms. The software engine may produce many types of standard disturbances from the literature, with varying repetitions and random parameters of the labeled disturbances. The software package also includes two reference classifiers that are based on deep-learning techniques: a convolutional neural-network classifier and a bidirectional long short-term memory classifier. The code of this software is available online, and is free to use. Download instructions appear at the end of this manuscript.

The paper continues as follows: Section II explains the main features of the proposed software package and the two reference classifiers. Section III provides few representative examples, and Section IV concludes the paper.

The work of Y. Levron was partly supported by Israel Science Foundation, grant No. 1227/18. R. Machlev, A. Chachkes and Y. Levron are with the Andrew and Erna Viterbi Faculty of Electrical Engineering, Technion—Israel Institute of Technology, Haifa, 3200003, Israel (ramm@campus.technion.ac.il, chachkes247@gmail.com, yoashl@ee.technion.ac.il). J. Belikov is with the Department of Software Science, Tallinn University of Technology, Akadeemia tee 15a, 12618 Tallinn (juri.belikov@taltech.ee). Y. Beck is with the Physical Electronics Department, School of Electrical Engineering, Tel Aviv University, Tel Aviv 69978, Israel (beck@tauex.tau.ac.il).

Paper submitted to the International Conference on Power Systems Transients (IPST2021) in Belo Horizonte, Brazil June 6-10, 2021.

II. The Open-Source Software Package

The synthetic dataset generator is designed to work with sixteen types of PQDs, all of them well known in the literature. Table I details the type of disturbances available, as well as their characteristic mathematical model and typical parameters. The parameters are in accordance with the IEEE-1159 standard [24] and the proposed characteristic equations are used in various works, such as [5], [10], [18]. The sampling rate for all signals is 3.2 kHz, the nominal frequency can be set to either 50 or 60 Hz, and the default duration is ten cycles. The basic code was created using MATLAB 2016b, and the Graphical User Interface (GUI) was constructed using the same software.

The software supports two main GUIs. The first GUI is a signal generator and analyzer that synthesizes the disturbance signals based on the fundamental models shown in Table I. One example is shown in Fig. 1, which shows a sag disturbance signal without noise. The number of cycles is 10, and the nominal frequency is 50 Hz. Note that the right side of the GUI presents the disturbances. The second GUI is a database creator that assembles different PQD vectors into a single ".mat" file. This function has three main options:

- Generate a specific disturbance signal with a predefined number of identical duplications, as demonstrated in Fig. 2.
- Generate a specific disturbance signal with a predefined number of duplicates that consist of random parameters and additive white Gaussian noise. This option allows to create a dataset of chosen disturbances by concatenating them one to another. An example is shown in Fig. 3. In this example 100 duplicates of a sag disturbance signal with random noise and a nominal frequency of 50 Hz with 10 cycles per signal are presented.
- Generate a comprehensive database from all sixteen disturbance signals. Each disturbance has a predefined number of copies with random parameters and additive white Gaussian noise, thus an infinite number of combinations can be generated. For example, in Fig. 4 1600 different disturbance signals are generated.

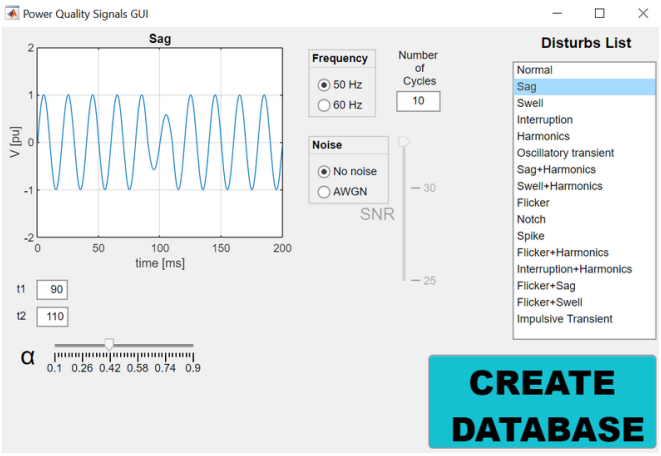

Fig. 1: GUI example: a sag disturbance signal without noise. In this example $\alpha = 0.42$.

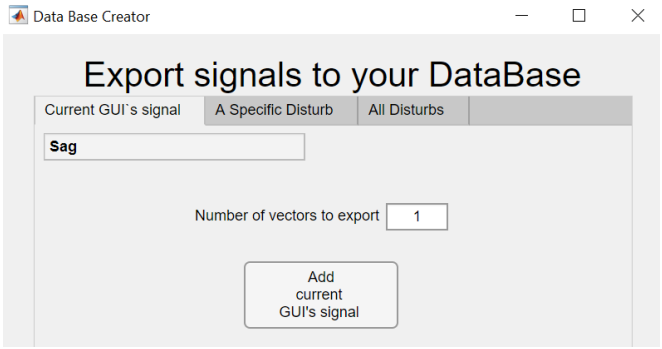

Fig. 2: GUI example: generation of a disturbance signal with a single sag.

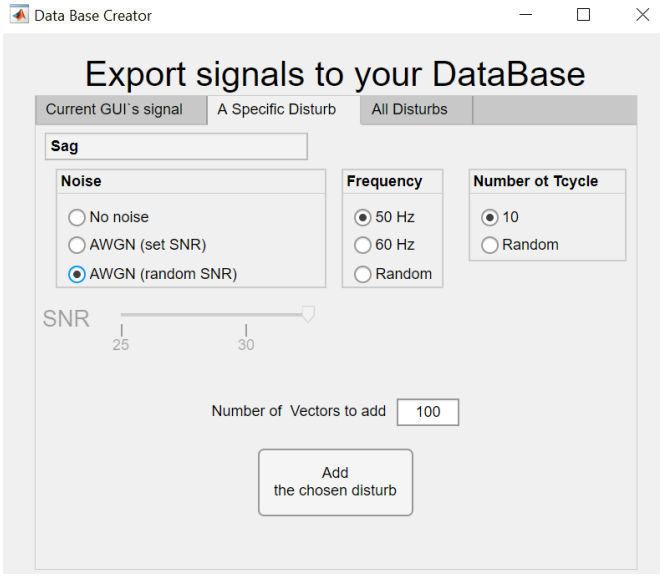

Fig. 3: GUI example: generation of 100 different sag disturbance signals with random noise.

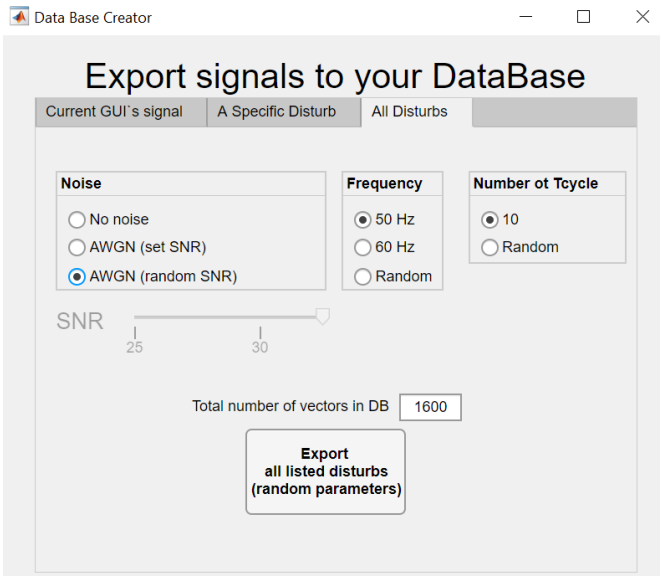

Fig. 4: GUI example: generation of 1600 disturbance signals, 100 signals for each disturbance, with random noise.

TABLE I: Mathematical Models of PQDs Which are Used by the Synthetic Generator

#	Disturbance	Characteristic equation	Parameters
1	Normal	$[1 \pm \alpha(u(t-t_1)-u(t-t_2))]\sin(\omega t)$	$\alpha < 0.04,\ T \leq (t_2-t_1) \leq 9T$
2	Sag	$[1-\alpha(u(t-t_1)-u(t-t_2))]\sin(\omega t)$	$0.1 \leq \alpha < 0.9,\ T \leq (t_2-t_1) \leq 9T$
3	Swell	$[1+\alpha(u(t-t_1)-u(t-t_2))]\sin(\omega t)$	$0.1 \leq \alpha \leq 0.8,\ T \leq (t_2-t_1) \leq 9T$
4	Interruption	$[1-\alpha(u(t-t_1)-u(t-t_2))]\sin(\omega t)]$	$0.9 \leq \alpha \leq 1,\ T \leq (t_2-t_1) \leq 9T$
5	Harmonics	$\alpha_1\sin(\omega t)+\alpha_3\sin(3\omega t)+\alpha_5\sin(5\omega t)+\alpha_7\sin(7\omega t)$	$0.05 \leq \alpha_3,\alpha_5,\alpha_7, \leq 0.15,\ \sum(\alpha_i^2)=1$
6	Flicker	$[1+\alpha_f\sin(\beta\omega t)]\sin(\omega t)$	$0.1 \leq \alpha_f \leq 0.2,\ 5 \leq \beta \leq 20Hz$
7	Oscillatory transient	$\sin(\omega t)+\alpha^{-(t-t_1)/\tau}\sin(\omega_n(t-t_1))(u(t_2)-u(t_1))$	$0.1< \alpha \leq 0.8,\ 0.5T \leq (t_2-t_1) \leq 3T,$ $8\leq \tau \leq 40,\ 300 \leq 2\pi\omega_n \leq 900$
8	Impulsive transient	$[1-\alpha(u(t-t_1)-u(t-t_2))]\sin(\omega t)]$	$0.1 \leq \alpha \leq 0.414,\ T/20 \leq (t_2-t_1) \leq T/10$
9	Notch (periodic)	$\sin(\omega t)-\text{sign}(\sin(\omega t))\times$ $\sum_{n=0}^{9}k[u(t-(t_1-0.02n))-u(t-(t_2-0.02n))]$	$0\leq t_1,t_2 \leq 0.5T,\ 0.1 \leq K \leq 0.4,$ $0.01T\leq t_2-t_1 \leq 0.05T$
10	Spike	$\sin(\omega t)+\text{sign}(\sin(\omega t))\times$ $\sum_{n=0}^{9}k[u(t-(t_1-0.02n))-u(t-(t_2-0.02n))]$	$0\leq t_1,t_2 \leq 0.5T,\ 0.1 \leq K \leq 0.4,$ $0.01T\leq t_2-t_1 \leq 0.05T$
11	Sag with harmonics	$[1-\alpha(u(t-t_1)-u(t-t_2))]\times$ $[\alpha_1\sin(\omega t)+\alpha_3\sin(3\omega t)+\alpha_5\sin(5\omega t)+\alpha_7\sin(7\omega t)]$	$0.1\leq \alpha < 0.9,\ T \leq (t_2-t_1) \leq 9T,$ $0.05\leq \alpha_3,\alpha_5,\alpha_7 \leq 0.15,\ \sum(\alpha_i^2)=1$
12	Swell with harmonics	$[1+\alpha(u(t-t_1)-u(t-t_2))]\times$ $[\alpha_1\sin(\omega t)+\alpha_3\sin(3\omega t)+\alpha_5\sin(5\omega t)+\alpha_7\sin(7\omega t)]$	$0.1\leq \alpha < 0.8,\ T \leq (t_2-t_1) \leq 9T,$ $0.05\leq \alpha_3,\alpha_5,\alpha_7 \leq 0.15,\ \sum(\alpha_i^2)=1$
13	Interruption with harmonics	$[1-\alpha(u(t-t_1)-u(t-t_2))]\times$ $[\alpha_1\sin(\omega t)+\alpha_3\sin(3\omega t)+\alpha_5\sin(5\omega t)+\alpha_7\sin(7\omega t)]$	$0.9\leq \alpha \leq 1,\ T \leq (t_2-t_1) \leq 9T$ $0.05\leq \alpha_3,\alpha_5,\alpha_7 \leq 0.15,\ \sum(\alpha_i^2)=1$
14	Flicker with harmonics	$[1+\alpha_f\sin(\beta\omega t)]\times$ $[\alpha_1\sin(\omega t)+\alpha_3\sin(3\omega t)+\alpha_5\sin(5\omega t)+\alpha_7\sin(7\omega t)]$	$0.1\leq \alpha_f \leq 0.2,\ 5 \leq \beta \leq 20$ $0.05\leq \alpha_3,\alpha_5,\alpha_7 \leq 0.15,\ \sum(\alpha_i^2)=1$
15	Flicker with sag	$[1+\alpha_f\sin(\beta\omega t)][1-\alpha(u(t-t_1)-u(t-t_2))]\sin(\omega t)$	$0.1\leq \alpha_f \leq 0.2,\ 5 \leq \beta \leq 20$ $0.1\leq \alpha \leq 0.9,\ T \leq (t_2-t_1) \leq 9T$
16	Flicker with swell	$[1+\alpha_f\sin(\beta\omega t)][1+\alpha(u(t-t_1)-u(t-t_2))]\sin(\omega t)$	$0.1\leq \alpha_f \leq 0.2,\ 5 \leq \beta \leq 20$ $0.1\leq \alpha \leq 0.8,\ T \leq (t_2-t_1) \leq 9T$

As mention above, a central challenge in the development cycle of PQD classifiers is the lack of comparative implementations. Therefore, the open-source code also includes two deep-learning networks for PQD classification, which can be used as a benchmark. Both networks are based on [18], which proposes a novel full closed-loop approach for detection and classification of power quality disturbances, based on deep neural networks.

The first network included in the package is a Convolutional Neural Network (CNN), which is typically used for image analysis [25]. The proposed network uses a 1-D convolution with a Rectified Linear Unit (ReLU), maxpooling layers, and batch-normalization layers, which are designed to capture multi-scale features and to reduce overfitting. The architecture of the network is shown in Table II. For all convolutional and maxpooling layers the kernel size is 3, and the stride value is 1. The implementation of the CNN network is done by two different platforms using the "Keras" and "TensorFlow" frameworks in Python and using the Deep-Learning toolbox in MATLAB. The user can choose to work with any software.

The second network is a recurrent neural network which, unlike the conventional feed-forward neural network, processes sequential inputs by an internal state that depends on all the previous inputs. When the sequential data is long the training process may be difficult, due to the problem of exploding and vanishing gradients. In order to overcome this issue, more sophisticated units such as Long Short-Term Memory (LSTM) units are typically used [26]. In the proposed reference classifier the network consists of bidirectional Long Short-Term Memory (BiLSTM) units which are typically used for speech recognition [27]. While in work [18] unidirectional LSTM units are proposed, we chose to use BiLSTM in this work since it considered more accurate for classification of sequences [28]. The architecture of the BiLSTM network is shown in Table III. Each BiLSTM layer contains 32 hidden units, and uses $\tanh(\cdot)$ as an activation function. The

TABLE II: Convolutional Neural Network (CNN) Architecture (Kernel=3 and Stride=1)

#	Layer	Parameters	Activation
1	Convolution	Filter size = 32	Relu
2	Convolution	Filter size = 32	Relu
3	MaxPooling		
4	Batch normalization		
5	Convolution	Filter size = 64	Relu
6	Convolution	Filter size = 64	Relu
7	MaxPooling		
8	Batch normalization		
9	Convolution	Filter size = 128	Relu
10	Convolution	Filter size = 128	Relu
11	Global MaxPooling		
12	Batch normalization		
13	Fully connected	Size = 256	Relu
14	Fully connected	Size = 128	Relu
15	Batch normalization		
16	Fully connected	Size = 16	Softmax

implementation of the BiLSTM network is done using the Deep-Learning toolbox in MATLAB.

TABLE III: BiLSTM Architecture

#	Layer	Parameters	Activation
1	BiLSTM	Number of hidden units is 32	tanh
2	BiLSTM	Number of hidden units is 32	tanh
3	BiLSTM	Number of hidden units is 32	tanh
4	Fully connected	Size is 16	Softmax

Both networks are configured based on the setups shown in Table IV. The size of each mini-batch is set to 64 signals, which are selected randomly for the back-propagation stage in the training process. The number of epochs, which indicates the number of times the network run on the entire training dataset, is 43 for the CNN, and 63 for the BiLSTM. The learning rate is initiated at 0.01, and is dynamically reduced by a factor of 1/2 after each 10 epochs, in case the loss function does not decrease. The selected loss function is based on cross-entropy, and the optimizer is "Nadam" for the CNN, and "Adam" for the BiLSTM [29].

TABLE IV: Training Options

Parameter	Value
Max epochs	43 (CNN) 63 (BiLSTM)
Mini batch size	64
Initial learning rate (m_0)	0.01
Learning rate	$m_0 \times 0.5^{\lfloor \frac{1+\text{epoch}}{10} \rfloor}$
Loss function	Cross-entropy
Optimizer	Nadam (CNN) Adam (BiLSTM)

Furthermore, the BiLSTM network size is only 215 Kbyte while the CNN network size is 520 Kbyte. Also, in case of CPU usage the computational time for classification is faster for the BiLSTM while in case of GPU usage the CNN is faster. Therefore, by means of resources the BiLSTM network should be used in case of general hardware with size limitation while in case of DNN processor the CNN network should be used.

III. Representative Examples

To further explain the proposed tool functions and capabilities, we conducted four simulations that demonstrate the use of the database generator and classifiers. The first and second simulations use the CNN classifier, while the third and forth simulations use the BiLSTM classifier. Two datasets were created using the synthetic dataset generator from Section II, one dataset without noise and the second with random additive white Gaussian noise having an SNR between 20-50 dB. Each dataset contains 76800 signals, which are 4800 signals for each disturbance type. The first and third simulations use the noiseless dataset, and the second and forth simulations use the noisy dataset. The training set includes 90% of the samples in the complete dataset, and the testing set includes the remaining 10%. The accuracy is defined as

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}. \quad (1)$$

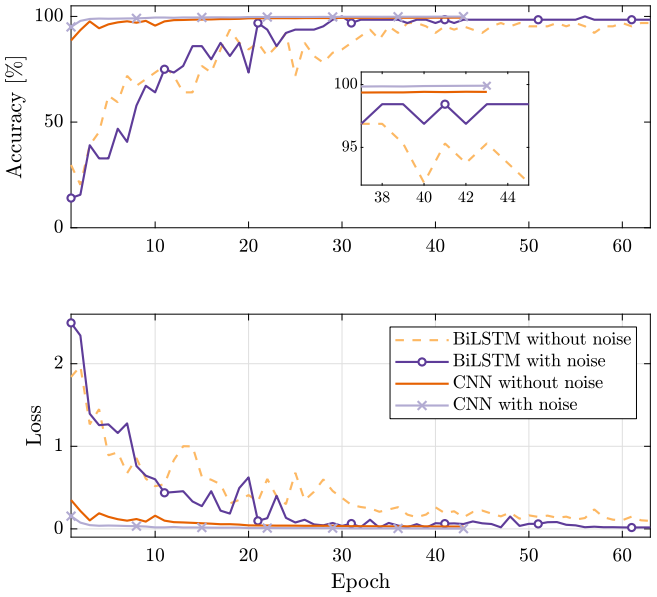

Fig. 5: Representative examples: training progress.

The accuracy and loss function values of each epoch in the training process is presented in Fig. 5 for all simulations. It can be seen that the CNN network training process has better performance and converges faster in comparison to the BiLSTM network.

The average classification accuracy of the training and testing sets is presented in Table V and Fig. 6, and the accuracy of the testing set per disturbance is presented in Table X. The CNN outperforms the BiLSTM classifier by means of average accuracy. Also, when the performance for a specific disturbance is compared the accuracy for most of the disturbances in the CNN outperforms the BiLSTM classifier.

TABLE V: Simulation Average Accuracy Results

	Training	Testing
CNN without noise	99.410%	99.280%
CNN with noise	99.900%	99.750%
BiLSTM without noise	96.875%	96.289%
BiLSTM with noise	98.437%	98.138%

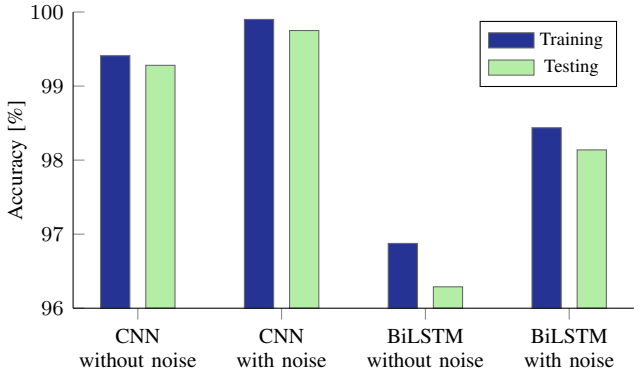

Fig. 6: Representative examples: simulation average accuracy results.

TABLE VI: Confusion Matrix for the Testing Set on the CNN Network: Without Noise

TRUE \ PREDICTED	Flicker	Flicker+harmonics	Flicker+Sag	Flicker+Swell	Harmonics	Impulsive transient	Interruption	Interruption+harmonics	Normal	Notch	Oscillatory transient	Sag	Sag+Harmonics	Spike	Swell	Swell+harmonics
Flicker	483	0	0	0	0	0	0	0	0	0	0	0	0	0	1	6
Flicker+harmonics	0	510	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Flicker+Sag	0	0	419	0	0	12	1	0	0	0	0	0	0	0	0	0
Flicker+Swell	0	0	0	474	0	0	0	0	0	0	0	0	0	0	0	0
Harmonics	0	0	0	0	457	1	0	0	0	0	0	0	0	0	0	0
Impulsive transient	0	0	0	0	0	468	0	0	0	0	0	0	0	0	0	0
Interruption	0	0	13	0	0	0	484	0	0	0	0	0	0	0	0	1
Interruption+harmonics	0	0	1	0	0	0	0	452	0	0	0	0	0	0	0	0
Normal	0	0	0	0	0	5	0	2	462	0	0	0	0	0	0	0
Notch	0	0	1	1	0	0	0	0	0	463	0	0	0	3	0	0
Oscillatory transient	0	0	0	0	0	0	0	0	0	0	478	0	1	0	0	0
Sag	0	0	0	0	0	1	0	3	0	0	0	485	0	0	0	0
Sag+Harmonics	0	0	0	0	0	0	0	0	0	0	0	0	488	0	0	0
Spike	0	0	0	0	0	0	0	0	0	0	0	0	0	501	0	0
Swell	0	0	0	0	0	0	0	0	0	0	0	0	0	0	512	1
Swell+harmonics	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	489

TABLE VII: Confusion Matrix for the Testing Set on the CNN Network: With Noise

TRUE \ PREDICTED	Flicker	Flicker+harmonics	Flicker+Sag	Flicker+Swell	Harmonics	Impulsive transient	Interruption	Interruption+harmonics	Normal	Notch	Oscillatory transient	Sag	Sag+Harmonics	Spike	Swell	Swell+harmonics
Flicker	486	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0
Flicker+harmonics	0	489	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Flicker+Sag	0	0	457	0	0	0	0	0	0	0	0	0	0	0	0	0
Flicker+Swell	0	0	0	495	0	0	0	0	0	0	0	0	0	0	0	0
Harmonics	0	0	0	0	480	0	0	0	0	0	0	0	8	0	0	0
Impulsive transient	0	0	0	0	0	485	0	0	0	0	0	0	0	0	0	0
Interruption	0	0	0	0	0	0	489	0	0	0	0	0	0	0	0	0
Interruption+harmonics	0	0	4	0	0	0	0	439	0	0	0	0	0	0	0	0
Normal	0	0	2	0	0	0	0	0	471	0	0	0	0	0	0	0
Notch	0	0	1	0	0	0	0	0	0	444	0	0	0	0	0	0
Oscillatory transient	0	0	0	0	0	0	0	0	0	0	502	0	0	0	0	0
Sag	0	0	0	0	0	0	0	0	0	0	0	514	0	0	0	0
Sag+Harmonics	0	0	0	0	1	0	0	0	0	0	0	0	501	0	0	0
Spike	0	0	0	0	0	0	0	0	0	0	0	0	0	461	0	0
Swell	1	0	0	0	0	0	0	0	0	0	0	0	0	0	475	0
Swell+harmonics	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	473

TABLE VIII: Confusion Matrix for the Testing Set on the BiLSTM Network: Without Noise

TRUE \ PREDICTED	Flicker	Flicker+harmonics	Flicker+sag	Flicker+swell	Harmonics	Impulsive transient	Interruption	Interruption+harmonics	Normal	Notch	Oscillatory transient	Sag	Sag+harmonics	Spike	Swell	Swell+harmonics
Flicker	480	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Flicker+harmonics	0	480	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Flicker+sag	3	0	473	0	0	0	1	0	0	0	0	3	0	0	0	0
Flicker+swell	4	0	0	474	0	0	0	0	0	0	0	0	0	0	2	0
Harmonics	0	0	0	0	480	0	0	0	0	0	0	0	0	0	0	0
Impulsive transient	0	0	0	0	0	457	0	0	10	0	1	3	0	9	0	0
Interruption	0	0	1	0	0	0	476	0	2	0	0	1	0	0	0	0
Interruption+harmonics	0	0	0	0	3	0	0	475	0	1	0	0	1	0	0	0
Normal	0	0	0	0	0	0	0	0	480	0	0	0	0	0	0	0
Notch	0	0	1	0	0	0	0	0	16	460	0	2	0	1	0	0
Oscillatory transient	0	0	0	1	0	5	0	0	1	4	464	0	0	0	5	0
Sag	0	0	7	0	0	5	97	0	1	3	1	366	0	0	0	0
Sag+harmonics	0	0	0	0	2	0	1	54	0	0	0	0	423	0	0	0
Spike	0	0	0	0	0	2	0	0	6	8	1	0	0	463	0	0
Swell	0	0	0	5	0	0	0	0	4	0	0	0	0	2	469	0
Swell+harmonics	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	475

TABLE IX: Confusion Matrix for the Testing Set on the BiLSTM Network: With Noise

TRUE \ PREDICTED	Flicker	Flicker+harmonics	Flicker+sag	Flicker+swell	Harmonics	Impulsive transient	Interruption	Interruption+harmonics	Normal	Notch	Oscillatory transient	Sag	Sag+harmonics	Spike	Swell	Swell+harmonics
Flicker	480	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Flicker+harmonics	0	455	0	0	25	0	0	0	0	0	0	0	0	0	0	0
Flicker+sag	5	0	467	1	0	0	1	0	0	0	0	5	1	0	0	0
Flicker+swell	1	0	0	478	0	0	0	0	0	0	0	0	0	0	1	0
Harmonics	0	2	0	0	478	0	0	0	0	0	0	0	0	0	0	0
Impulsive transient	0	0	0	0	0	454	0	0	0	22	3	1	0	0	0	0
Interruption	0	0	0	0	0	0	477	0	1	0	1	1	0	0	0	0
Interruption+harmonics	0	0	0	0	1	0	0	477	0	0	0	0	1	1	0	0
Normal	0	0	0	0	0	0	0	0	479	0	0	0	0	1	0	0
Notch	0	0	0	0	0	18	0	0	0	461	0	0	0	1	0	0
Oscillatory transient	3	0	1	0	0	5	0	0	4	1	465	0	0	0	1	0
Sag	0	0	0	0	0	4	1	0	0	1	0	474	0	0	0	0
Sag+harmonics	0	0	0	0	8	0	0	8	0	0	1	0	462	1	0	0
Spike	0	0	0	0	0	0	0	0	1	0	0	0	0	479	0	0
Swell	0	0	0	2	0	3	0	0	0	0	0	0	0	0	475	0
Swell+harmonics	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	476

In addition, to demonstrate the performance of the algorithms for each disturbance, "confusion" matrices were applied to the testing sets of the CNN and BiLSTM networks. These matrices are presented in Tables VI, VII, VIII, and IX. The results show that the average accuracy of the CNN network is superior to that of the BiLSTM network. In addition, for both networks the performance of the dataset with noise is better than the dataset without noise. One possible explanation for this phenomenon is that the dataset is rather small, and such small size may lead to over-fitting which can be reduced by adding noise [30]. Moreover, from the confusion matrix in Table VIII it can be seen that the trained network classify several of the sag disturbances in the testing set as interruption disturbances, when the dataset without noise is used. Such errors are not common with the noisy dataset. Due to the good performance of these classifiers, we suggest that they can be used by the community as benchmarks for the development of new and better PQD classification algorithms.

IV. Conclusions

Power quality monitoring tools are becoming a necessity, and many recent studies focus on detection and classification of power quality disturbances. However, presently a core obstacle that prevents the direct comparison of PQD classification techniques is the lack of a standard database that can be used as a benchmark. In this light, we propose here an open-source software which enables the creation of synthetic power quality disturbances, and is designed specifically for comparison of PQD classification algorithms. The software produces several types of standard disturbances from the literature, with varying repetitions and random parameters of the labeled disturbances, and includes two reference classifiers that are based on deep-learning techniques. Due to the good performance of these classifiers, we suggest that they can be used by the community as benchmarks for the development of new and better PQD classification algorithms.

Downloading The Code

The software is free and publicly available, and can be downloaded from https://github.com/chachkes247/Power-Quality-Disturbances. We kindly ask that researchers who use this software will cite this paper. In case the CNN reference classifier is used, kindly cite [18] as well.

References

[1] M. H. J. Bollen, *Understanding Power Quality Problems*. John Wiley & Sons, 1999.

[2] C. Sankaran, *Power Quality*. CRC Press, 2017.

[3] A. Ghosh and G. Ledwich, *Power Quality Enhancement Using Custom Power Devices*. Springer US, 2002.

TABLE X: Simulation Accuracy (%) Results per Disturbance for Testing Set

#	Disturbance	CNN without noise	CNN with noise	BiLSTM without noise	BiLSTM with noise
1	Normal	98.507	99.577	100	99.792
2	Sag	99.182	100	76.25	98.75
3	Swell	99.805	99.79	97.708	98.958
4	Interruption	97.189	100	99.167	99.375
5	Harmonics	99.782	98.36	100	99.583
6	Flicker	98.571	99.59	100	100
7	Oscillatory transient	99.791	100	97.684	96.875
8	Impulsive transient	100	100	95.208	94.583
9	Notch (periodic)	98.932	99.775	95.833	96.042
10	Spike	100	100	96.458	99.792
11	Sag with harmonics	100	99.8	88.125	96.25
12	Swell will harmonics	100	100	98.958	99.167
13	Interruption with harmonics	99.779	99.097	95.208	99.375
14	Flicker with harmonics	100	100	100	94.792
15	Flicker with sag	96.991	100	98.542	97.292
16	Flicker with swell	100	100	98.75	99.583

[4] M. H. J. Bollen and I. Y.-H. Gu, *Signal Processing of Power Quality Disturbances*. John Wiley & Sons, Inc., 2006.

[5] E. Hossain, M. R. Tur, S. Padmanaban, S. Ay, and I. Khan, "Analysis and mitigation of power quality issues in distributed generation systems using custom power devices," *IEEE Access*, vol. 6, pp. 16 816–16 833, 2018.

[6] S. K. Khadem, M. Basu, and M. Conlon, "Power quality in grid connected renewable energy systems: role of custom power devices," *Renewable Energy and Power Quality Journal*, vol. 1, no. 08, pp. 878–881, 2010.

[7] Z. Moravej, A. A. Abdoos, and M. Pazoki, "Detection and classification of power quality disturbances using wavelet transform and support vector machines," *Electric Power Components and Systems*, vol. 38, no. 2, pp. 182–196, 2009.

[8] D. D. Yong, S. Bhowmik, and F. Magnago, "An effective power quality classifier using wavelet transform and support vector machines," *Expert Systems with Applications*, vol. 42, no. 15-16, pp. 6075–6081, 2015.

[9] S. He, K. Li, and M. Zhang, "A real-time power quality disturbances classification using hybrid method based on s-transform and dynamics," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 9, pp. 2465–2475, 2013.

[10] R. Kumar, B. Singh, D. T. Shahani, A. Chandra, and K. Al-Haddad, "Recognition of power-quality disturbances using s-transform-based ANN classifier and rule-based decision tree," *IEEE Transactions on Industry Applications*, vol. 51, no. 2, pp. 1249–1258, 2015.

[11] W. Qiu, Q. Tang, J. Liu, Z. Teng, and W. Yao, "Power quality disturbances recognition using modified s transform and parallel stack sparse auto-encoder," *Electric Power Systems Research*, vol. 174, p. 105876, 2019.

[12] F. A. S. Borges, R. A. S. Fernandes, I. N. Silva, and C. B. S. Silva, "Feature extraction and power quality disturbances classification using smart meters signals," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 2, pp. 824–833, 2016.

[13] M. S. Manikandan, S. R. Samantaray, and I. Kamwa, "Detection and classification of power quality disturbances using sparse signal decomposition on hybrid dictionaries," *IEEE Transactions on Instrumentation and Measurement*, vol. 64, no. 1, pp. 27–38, 2015.

[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[15] N. Mohan, K. P. Soman, and R. Vinayakumar, "Deep power: Deep learning architectures for power quality disturbances classification," in *2017 International Conference on Technological Advancements in Power and Energy (TAP Energy)*. IEEE, 2017.

[16] H. Liu, F. Hussain, Y. Shen, S. Arif, A. Nazir, and M. Abubakar, "Complex power quality disturbances classification via curvelet transform and deep learning," *Electric Power Systems Research*, vol. 163, pp. 1–9, 2018.

[17] Y. Deng, L. Wang, H. Jia, X. Tong, and F. Li, "A sequence-to-sequence deep learning architecture based on bidirectional GRU for type recognition and time location of combined power quality disturbance," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 8, pp. 4481–4493, 2019.

[18] S. Wang and H. Chen, "A novel deep learning method for the classification of power quality disturbances using deep convolutional neural network," *Applied Energy*, vol. 235, pp. 1126–1140, 2019.

[19] S. Khokhar, A. A. B. M. Zin, A. S. B. Mokhtar, and M. Pesaran, "A comprehensive overview on signal processing and artificial intelligence techniques applications in classification of power quality disturbances," *Renewable and Sustainable Energy Reviews*, vol. 51, pp. 1650–1663, 2015.

[20] G. S. Chawda, A. G. Shaik, M. Shaik, P. Sanjeevikumar, J. B. Holm-Nielsen, O. P. Mahela, and K. Palanisamy, "Comprehensive review on detection and classification of power quality disturbances in utility grid with renewable energy penetration," *IEEE Access*, pp. 1–1, 2020.

[21] R. Igual and C. Medrano, "Research challenges in real-time classification of power quality disturbances applicable to microgrids: A systematic review," *Renewable and Sustainable Energy Reviews*, vol. 132, p. 110050, 2020.

[22] O. Florencias-Oliveros, M. J. Espinosa-Gavira, J. J. G. de-la Rosa, A. Agüera-Pérez, J. C. Palomares-Salas, and J. M. Sierra-Fernóndez, "Real-life power quality sags," IEEE Dataport, 2017, [Online] Available https://ieee-dataport.org/documents/real-life-power-quality-sags, Accessed October 22, 2020.

[23] O. Florencias-Oliveros, M. J. Espinosa-Gavira, J. J. G. de-la Rosa, A. Agüera-Pérez, J. C. Palomares-Salas, and J. M. Sierra-Fernóndez, "Real-life power quality transients," IEEE Dataport, 2017, [Online] Available https://ieee-dataport.org/documents/real-life-power-quality-transients, Accessed October 22, 2020.

[24] IEEE, "IEEE recommended practice for monitoring electric power quality," *IEEE Std. 1159-1995*, pp. i–81, 2009.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2012.

[26] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2017.

[27] A. Graves, N. Jaitly, and A.-R. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2013.

[28] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv*, 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[30] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.