energies

MDPI

Article

An End-to-End Deep Learning Method for Voltage Sag Classification †

Radovan Turović, Dinu Dragan *, Gorana Gojić, Veljko B. Petrović, Dušan B. Gajić, Aleksandar M. Stanisavljević and Vladimir A. Katić

Faculty of Technical Sciences, University of Novi Sad, 21460 Novi Sad, Serbia; radovan.turovic@uns.ac.rs (R.T.); gorana.gojic@uns.ac.rs (G.G.); pveljko@uns.ac.rs (V.B.P.); dusan.gajic@uns.ac.rs (D.B.G.); acas@uns.ac.rs (A.M.S.); katav@uns.ac.rs (V.A.K.)

* Correspondence: dinud@uns.ac.rs

† This paper is extended version of our paper published in the Proceedings of the 2021 21st International Symposium on Power Electronics (Ee); Novi Sad, Serbia, 27–30 October 2021.

Abstract: Power quality disturbances (PQD) have a negative impact on power quality-sensitive equipment, often resulting in great financial losses. To prevent these losses, besides detecting a PQD on time, it is important to classify it, so that appropriate recovery procedures are employed. The majority of research employs machine learning model PQD classifiers on manually extracted features from simulated or real-world signals. This paper presents an end-to-end approach that circumvents the manual feature extraction and uses signals generated from mathematical voltage sag type formulas. We developed a configurable voltage sag generator that was used to form training and validation datasets. Based on the synthetic three-phase voltage signals, we trained several end-to-end LSTM classifiers that classify voltage sags according to ABC classification. The best-performing model achieved an accuracy of over 90% in the real-world dataset.

Keywords: power quality; classification; neural networks; voltage sag; dataset

Citation: Turović, R.; Dragan, D.; Gojić, G.; Petrović, V.B.; Gajić, D.B.; Stanisavljević, A.M.; Katić, V.A. An End-to-End Deep Learning Method for Voltage Sag Classification. *Energies* **2022**, *15*, 2898. https://doi.org/10.3390/en15082898

Academic Editor: Fernando Sánchez Lasheras

Received: 7 February 2022
Accepted: 10 March 2022
Published: 15 April 2022

1. Introduction

With the latest trend of an increasing number of sophisticated equipment connected to the power grid, which is more susceptible to voltage disturbances while also contributing to them, power quality has gained significance. To address this problem, the effective characterization of power quality disturbances (PQDs) is crucial, i.e., quick and accurate detection and classification. Various methods have been proposed in the literature [1–10]. They enable fast and accurate PQD detection and classification based on mathematical PQD models or different signal-processing tools for feature and pattern recognition [2,7]. For automated classification, the application of artificial intelligence in a form of different machine learning algorithms, more specifically deep learning, proved as a good solution [3,8,9]. However, there are many PQD indices with various parameters, they have different causes, and their features and feature extraction methods are not standardized, so an application of the deep learning tools may be complex and time demanding [4–6]. Better performance may be achieved with methods specialized for a single disturbance. In that case, specific parameters of such a disturbance may be detected and more precise classification may be achieved.

Among the various types of disturbances, voltage sags often have severe negative effects. They are the most common cause of equipment malfunction and incur high time and revenue losses for businesses. Their detection, classification, and characterization are of special interest for variable-speed electric motor drive operations, power system stability, and the secure interconnection of distributed generation based on renewable energy sources, as well as for various devices for PQDs mitigation [1,11].

The conventional approach for their detection is to measure the RMS values of the signal over time and simply check whether the RMS value falls below a standardized threshold. However, there are several more advanced methods, which differ in speed, accuracy, and complexity [12,13]. They require some form of signal processing and may be classified into parametric, non-parametric, or hybrid methods, while the application of artificial intelligence is also possible [12]. The main parameters are the speed and accuracy of the detection. The comparison presented in [13] shows that a method based on harmonic footprint proposed in [14] has superior performances, so far.

Voltage sags can be classified into several types: (a) by using three-phase voltage phasors (there are seven types, A–G classification) [1]; (b) by applying the method of "voltage sags magnitude equality" (there are three types: Types I, II, and III) [15,16], and by the shape of the voltage sag RMS value (there are three types, rectangular, nonrectangular (triangular), or step change (multi-stage)) [17]. The most convenient method is the first one (A–G classification), and it will be used in this paper. For differentiation between specific types of voltage sags, there are various approaches, but all of them require specific feature extraction, optimization, and processing [8,10,17]. The main feature of voltage sag is the voltage RMS value, but it can be also recognized by using a space vector ellipse in a complex plane [9] or by harmonic footprint [14]. In this paper, we will use the RMS feature. However, the RMS feature is not always sufficient. In some cases, it is not straightforward, or it is not possible to differentiate between the types of voltage sags because of the complex interactions in the power grids. Therefore, the application of some artificial intelligence tools may be of interest.

Machine learning (ML) methods have the potential to bring additional generalization power to automatic classification. The feature extraction can be done manually or within an end-to-end approach. There are advantages and drawbacks to both approaches. When a manual feature extraction is done properly, ML models can be trained better, the model is less complex, and, as a result, the model calculation time is shorter. The biggest drawback is the process of manual feature engineering. To achieve the best results all the necessary features must be selected, and all of the unnecessary features must be removed to achieve the least complex and most accurate model. Additionally, when the feature extraction is computationally demanding, it can prolong overall classification time because extraction is done manually. With an end-to-end approach, ML methods learn the necessary features during training. As will be shown in the Literature Review Section, such an approach for the PQD classification has been reported in just a few papers. In this approach, it is necessary to rely on a model's ability to automatically infer useful features for the classification, and there is a potential suspect in whether the model can do it correctly. Still, end-to-end methods have been successfully applied in different fields, e.g., various applications in acoustics such as classification and transformation of sounds [18]. In an end-to-end approach, however, there is no feature extraction done beforehand, which is an advantage over the previous approach. Additionally, the time needed for the whole task can be shorter. The main drawback to this approach is the design of the model and the process of training. With the feature extraction responsibility shifted to the model, the complexity of the model must be high enough to enable learning of all necessary features and the process of training must be adjusted accordingly.

Ideally, a big dataset of real-world data is available to the ML model during training. Usually, this task is very time consuming and requires a professional in the required field to correctly label all the data. When an appropriate real-world dataset is not available, the usual solution is to augment the dataset with additional generated data or to generate the whole dataset. For PQD datasets, the usual approach is to generate data by simulation. Simulations output signals closer to the real-world recordings. The caveat to this approach is the time needed for the design of the grids used in simulations and the simulations themselves. In contrast, this paper shows the use of generalized mathematical models as a valid option to generate a synthetic PQD dataset. This approach shows greater flexibility in the variety of the generated signals and is faster with the ability to generate hundreds

of thousands of signals per hour, as we did it. These signals are simplified versions of real-world, measured data, so the networks trained on synthetic datasets in real-world applications often do not perform as well as those trained on datasets with measured data. However, there are advantages in training the network on a synthetic dataset including the following:

- Deep neural network training requires a lot of PQD signals (orders of 10 s to 100 s of thousands of instances or more) that can be generated in a few hours, contrary to, e.g., the signal measurement and collection from electrical grids that can last for months or years, making synthetic datasets more accessible.
- It is easier to produce a synthetic dataset with a balanced number of instances of different fault types than to collect the same percentage of all signal types from the electrical grid since some fault types occur rarely.
- There is no need for a professional to do the labeling of the data since it is already known what is being outputted during generation.
- The model trained on the synthetic dataset can be used in transfer learning [19] and uptrained with measured signals, satisfying the need for large and realistic datasets at the same time.

Although models trained on synthetic data do not tend to perform as well on real data as models trained on real data, they, in contrast, can have better generalization abilities when used on datasets with different distributions.

This paper represents a continuation of our work in [20]. In the previous study, we built an LSTM network and trained it on synthetic signals, and it worked wonderfully but had poor performance on real-world data. It had an accuracy of 50% or lower if real-world signals were applied. In this paper, we expand our research on voltage sag classification based on raw signals. We explore how improvements to the dataset and the model impact the performance of real-world voltage measurement data. These augmentations enabled us to train a model that achieved an accuracy of over 90% on the real-world dataset.

To create a synthetic dataset of voltage sag signals, we developed a voltage sag generator. The generator is written in Python. It is parameterized and extensible so it can be configured to output differently distributed datasets as well as extended with additional fault types. An additional benefit of using the generator would be for validation purposes and comparison of different models. To this end, the generator can be used to generate standardized datasets for these purposes. The generator, the models, the training, the validation code, and the best weights described in this paper are available online at: https://bitbucket.org/sara_e21/ee2021-mdpi-vsc/ (accessed on 30 January 2022).

Four variants of the synthetic dataset and three different models were made for this research. The dataset variants differ in respect to two augmentations that bring signals closer to the real-world measurements. The first is the amplitude variation of non-faulty segments and the second is the smoothing of transitions between faulty and non-faulty segments of the signal. Models differ in complexity in respect to the number, type, and arrangement of the layers. For all 12 possible combinations, a model instance was trained and tested on the real-world data. Each model instance is verified on the real-world dataset. The real-world dataset consists of real-world recordings made and labeled by professor Dr. Math H. J. Bollen [1].

The best model achieved around 90% accuracy on the real-world dataset. The achieved results show great promise and show that synthetic dataset can be used to successfully train classifiers for real-world fault recordings. The end-to-end approach demonstrated that manual feature extraction is not crucial for the classification of PQD. We can expect better-performing end-to-end models from future works. It would be beneficial for future research to see the standardization of training and/or verification datasets. This would make a comparison of the proposing ML methods more straightforward. Mentioned standardized datasets could be made with the use of the generator of synthetic signals proposed in the paper.

The rest of the paper goes as follows. The second section gives insights into deep learning theory closely related to the classification method presented in this paper. It is followed by an overview of the recent literature in PQD and voltage sag classification research in the third section. The fourth section explains in detail the methodology behind the experiment preparation and followed protocols that led to the results. The fifth section focuses on the acquired results and provides an analysis of the results. Finally, the last section gives a summary of the results and proposes possible future work paths.

2. Background Theory

Deep neural networks (DNNs) [21] are a part of a broader family of deep learning algorithms. Nowadays, DNNs are widely used due to their ability to automatically extract multi-scale features from large amounts of data and, optionally, perform tasks like detection or classification based on learned features. Automatic feature extraction has an advantage over its predecessors, hand-crafted feature extraction algorithms, since no assumptions on data patterns are required to find and learn the patterns. A DNN consists of a large number of stacked layers. The attribute "deep" in deep neural networks is due to the number of layers in the network, which can be very large and depend on the complexity of the data used for learning. DNNs learn layer by layer by feeding the first, input layer with data, while all subsequent layers except the output one learn significant features. Effectively, layers refine and pass information among themselves to obtain requested generalization from the data. The typical structure of a DNN consists of an input layer, multiple hidden layers, and an output layer. Some of the commonly used DNN specializations are convolutional neural networks (CNNs), which are heavily utilized in image processing and recurrent neural networks (RNNs) used in fields where the input data are in the form of a time series.

2.1. Recurrent Neural Networks

Recurrent neural networks (RNNs) [22] are good at processing sequence data for predictions. That is the reason why they are widely used in tasks naturally producing sequence data, such as natural language processing and speech recognition, where the input data are in the form of text or audio sequences. Vanilla RNNs consist of input and output layers and one or more hidden layers. The hidden layer contains many neurons, or cells, as they are usually called in RNN terminology, being responsible for "memorizing" the content the network has learned and maintaining the network's state. The RNN is trained by feeding it with pieces of sequential data, one chunk at a time to update the network's state. However, what differs RNN from a standard feed-forward neural network is that calculation of the network state in time step t depends not only on the input data for that time step, but also on the network state in time step $t-1$. This mechanism enables RNNs to incorporate past knowledge from all previous time steps when predicting the output for time moment t. Figure 1. shows the RNNs computational graph across the time for the same RNN.

RNNs have multiple advantages over standard feed-forward networks such as the ability to handle sequences of variable length and constant model size for a variable input size. However, vanilla RNNs also have a significant limitation—they have short-term memory, meaning that they cannot properly learn long-term dependencies in the input sequence. This limitation is introduced by the algorithm used to update RNNs weights in the training process.

2.2. Long Short-Term Memory Networks

Long short-term memory (LSTM) [23] neural network architecture is a variation of RNN introduced to mitigate short-term memory issues of vanilla RNNs by advanced modifications in RNN cell design. Similar to RNN cells, LSTM cells are capable of memorizing the state, but unlike RNN cells, that is not done unconditionally. LSTM cells contain gates that are used to control information flow through the cell in the training process, thus limiting cell state perturbations that might negatively affect learning. The intuition behind

the LSTM gate concept is to enable the cell to forget irrelevant history and use only new and important data to update its state and generate the output. LSTMs are appropriate to use when it is expected that underlying data have long-term dependencies. Otherwise, vanilla RNNs can perform well enough in a shorter time, since updating the RNNs state is less computationally intensive compared to LSTM, where more calculations are needed to calculate the cell's next state.

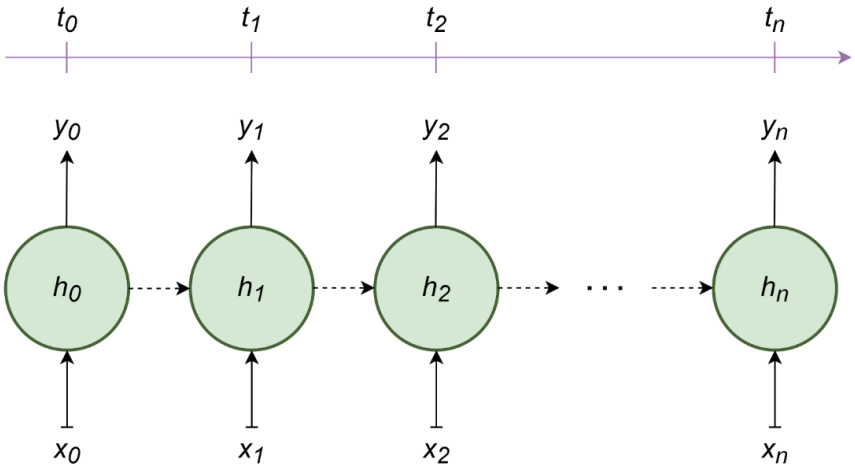

Figure 1. RNN computational graph across the time. t_i represents different time points in which the RNN for a given input x_i and current state h_i produces the output y_i.

3. Literature Review

The development of machine learning techniques has influenced the work on the classification of PQDs in electrical grids, which is visible by an increased number of published papers presenting classification solutions that rely on machine learning. Commonly used machine learning techniques to classify PQ signal disturbances are artificial neural networks (ANNs), support vector machines (SVM), decision trees, logistic regression (LR), and extreme learning machine (ELM) [10,24].

All the above-mentioned methods involve a process known as 'feature engineering' to extract characteristics significant for PQ classification from a time-series electrical signal. Usually that means applying a hand-crafted transformation to extract different, spatio-temporal, stationary, non-stationary, or other signal characteristics. Some of the most used feature extraction methods in literature are Wavelet transform (WT) Kalman filter, S-transform, Fourier transform, Gabor–Wigner transform, Hilbert–Huang transform, Harmonic footprint, and statistics-based methods [10,12,14,24].

However, feature engineering has certain drawbacks: it requires an expert to find the optimal choice for the feature extraction algorithm, and it adds computational cost to the classification pipeline; thus, it limits the method's applicability for real-time purposes. Sometimes, even with the best effort, it is not possible to design a feature extraction function that will preserve all information that might benefit classification and simultaneously reduce signal complexity. That motivated later research on PQ classification to converge towards deep learning, where significant features are learned automatically by deep neural networks [9]. In some cases, the deep neural networks are used in the feature engineering process to automatically find significant features that are further used to train a simpler classification method [25] or, more often, both extract significant features and classify PQ signal disturbances [26–29]. In [25], a CNN is used to automatically extract features from input signals that are used to train distance-weighted k-nearest neighbors. There are also approaches based on two-dimensional (2D) CNNs that make use of the plotted disturbance signals [26,29,30], which practically classify waveform images instead of one-dimensional time series. This can be seen in [30], where PQ voltage disturbances are transformed using continuous wavelet transform (CWT), CWT coefficients are plotted, and those plots are used as inputs to train a CNN. Work in [26] transforms the raw signal

to root mean square (RMS) voltage waveform and uses waveform plots for classification. While this approach is suitable for post-fault signal classification, it can be problematic in time-critical applications, like real-time fault detection or classification, where the signal to image conversion adds up on classification time. While early work on deep learning used mainly transformed signals as an input to the network, using raw time series to classify disturbances is more common in recent work [22,23,27]. By operating on a raw waveform in the time domain, the network can learn low-level patterns encoded directly into the signal, without any assumptions on the underlying patterns introduced by feature engineering or input signal transformations. Training networks on raw signals have so far demonstrated at least comparable or even superior classification results compared to previously proposed machine learning methods [31,32]. Similar performance trends have also been demonstrated in other fields where DNNs are trained on raw wavelets such as automatic speech recognition [18,33], acoustic scene classification [34,35], and motor fault detection [36,37].

Since deep learning methods require large datasets for training, there are two mainstream directions in literature when training these methods for PQ voltage disturbance classification: to train the method on a dataset of measured signals from an electrical grid [26,30,38] or to synthesize signals based on mathematical models [27,28,32]. The first approach is widely adopted in literature when it comes to the training of traditional machine learning methods, such as SVM classifiers or shallow neural networks, since they can be trained with smaller amounts of data. Training the method on measured signals is a better option, when possible since the performance of a trained model is a better approximation of the model's behavior in industry applications when compared to the models trained on synthetic data. There are two main concerns in training the network on real-world datasets that emerged with the development of deep learning models: existing publicly available datasets are too small to be used in deep learning training and collecting a huge number of measured signals is an expensive and time-consuming process aggravated by class imbalance problems since some disturbances appear more often in electrical grids than others. While dataset collection lags in time compared to deep learning model development speed, it is accepted in the literature to train deep neural networks on datasets with synthetically generated signals and, if possible, test the model on signals measured in electrical grids [27,32,39].

Voltage Sag Classification

Voltage sag is one of the classes commonly predicted by PQD classification. It is of interest to classify voltage sags in finer subcategories, as well as to identify voltage sag sources to timely and properly react to the sag and minimize the probability of damage in the grid. The majority of current research centers on the latter, with traditional pipelines utilizing feature engineering [40,41] in combination with some shallow machine learning methods like SVMs [42–44] and ANNs [40,44] or discrete algorithms [45]. To the best of the authors' knowledge, just a few papers are demonstrating the application of deep learning models in voltage sag classification for time domain signals. A work from [46] uses the LSTM network to train the model on RMS sequences obtained by transformation of raw signals. The proposed LSTM network is trained and tested on measured data labeled according to the ACD classification scheme and achieves a classification accuracy of 93.40%. Subsequent work in [47] improves on work in [46] by increasing classification accuracy on measured test data to 97.72%. The method proposed in [47] transforms input signals from a time domain to a 2D space-phasor model (SPM) and employs a CNN to perform classification into one of the classes of the ACD classification scheme. We find that methods from [46,47] have multiple differences compared to the method proposed in this paper. Firstly, they classify signals according to the ACD classification scheme, which is derived from ABC classification by aggregating seven classes from ABC classification into three ACD classification schemes. Secondly, although it is desirable to use real-world data in network training, a shortcoming of using private datasets, as it is done in [46,47], is a lack

of experiment reproducibility. In this work, using synthetic signals are made publicly available and the method achieves accuracy close to that reported in [46] on a more detailed classification scheme. The third meaningful difference to the [46,47] is that we use raw signals as an input to classification algorithm contrary to the transformed signals such as RMS in [46] and SPM in [47]. Transforming the input signal to emphasize certain properties to facilitate classification may lead to suboptimal results as is suggested in [48]. Since signal transformations are targeted to emphasize certain signal properties, it is natural that other, possibly significant properties that are present in a raw signals get omitted from a transformed signal. Simplified signals are a requirement when it comes to shallow machine learning models training since these models have a limited ability to learn complex patterns due to their fairly simple design. On the other hand, by adjusting the depth of the DL model, such as the one proposed in this paper, one can increase the model's ability to learn complex patterns and avoid information loss caused by signal simplification. Work in [48] recognizes the shortcomings of training a DL model on transformed signals and proposes improved, bidirectional LSTM (Bi-LSTM) with an attention mechanism to classify voltage sag causes with high accuracy for each predicted class. Although the proposed work addresses voltage sag source instead of voltage sag classification, it is significant as a demonstration of successful LSTM application in raw waveform-based classification for the task closely related to the one addressed in this paper. Inspired by the growing success of DL methods in raw signal-based classification in voltage sag cause and PQ disturbance classification, this paper demonstrates usage of the LSTM neural network for voltage sag subtype classification according to standard ABC categorization [1]. To the best of the authors' knowledge, this is the first work applying DL models on raw, three-phase, voltage series time data to predict a voltage sag subtype in accordance with the ABC scheme.

4. Methodology

As a continuation of the previous research, to increase the accuracy and applicability of the model, the synthetic dataset was augmented resulting in four dataset variants, and three improved models that were trained on each of them and tested on a real-world dataset. In the case of datasets, two upgrades were introduced to make the dataset a step closer to the real-world data. Three models were designed to test how the changes to the depth and complexity of the model affect the results. All the data and model combinations were trained and tested on a set of real signals. The overview of the experimental setup is shown in Figure 2. There are two stages. The first stage is the training of the models, and the second stage is the testing of the trained models on the real-world data. In the first stage, one model per each synthetic dataset and model combination are trained, as depicted by the left half of Figure 2. The second stage is depicted by the right half of Figure 2.

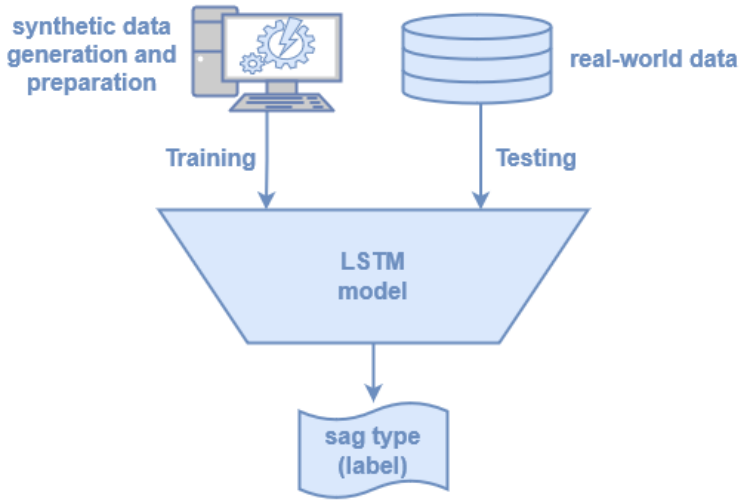

Figure 2. Overview of the experimental setup.

In the following subsections, each stage will be described in more detail.

4.1. Synthetic Signal Generation

Synthetic signal generation consists of generating the signals themselves with their corresponding labels. The synthetic signals are then used as an input for the models and the labels are used as expected output. The place of the signal generation in the experiment is shown in Figure 2 as a computer icon.

Three-phase voltage signals by their nature are three sinusoids with different starting phase angles [1]. Because of this, the generation of signals is mainly based on the manipulation of sinusoids. Many disturbances can be simulated with simple amplitude scaling and phase angle shifting. For this research, the generator was designed to output non-faulty signals and all types of voltage sags proposed in the ABC classification. The classification includes seven basic voltage sag types named, respectively, with letters A to G. Each generated signal is labeled with one of seven types, or with a special NF label, which stands for No-Fault (NF) and is used for signals without disturbances or with disturbances, where the RMS of all phases is above 90% of the nominal voltage. In this research, we focused only on simple fault signals with single class faults. From the ground up, the generator was built with extensibility in mind so that more fault types can be added later.

The generator is also parameterized to enable greater flexibility of the process. The parameters can be divided into two main groups. Signal parameters, which pertain to the characteristics of the signals and are irrelevant to the fault types, and fault parameters, which induce the fault deformations in the non-faulty signal.

A comprehensive list of parameters used for signal generation is shown in Table 1. Out of these parameters, *signal duration*, *fault start time*, and *fault duration* are temporal parameters. *Signal duration* controls the time length of a single sample, *fault starting time* controls the moment at which the fault will occur in the signal, and *fault duration* controls the duration of the fault. The main signal characteristics that differ between grids are *nominal voltage* and *signal frequency*. *Nominal rms* determine the expected voltage to be delivered in the grid. *Signal frequency* allows for the generator to output signals of any frequency to accommodate the variations from real-world grids such as the 50 and 60 Hz signals. *The sampling frequency* is a parameter that accommodates variations in resolutions of measuring devices used for voltage measurements from grids. To take into consideration the possibilities of signal amplitudes varying and still not being faulty and fault events happening during different phases of a signal cycle, *no-fault amplitude delta* and *phase angle delta* parameters change the elementary parameters of sinusoidal signal generation, namely the amplitude and phase angle. *Non-fault amplitude delta* scales nominal voltage to the (90%, 110%), while *phase angle delta* is phase angle added to the phase angle. The *no. of smoothing time points* parameter is the number of time points, i.e., number of samples, which are used in smoothing between a normal signal and a fault. Lastly, *noise percentage* controls the percentage of the white noise compared to the nominal value of the signal. *Label* controls what kind of a signal is being generated. In the case of the NF label, a signal without faults is generated. In all other cases, a type of voltage sag is being generated and is dependent on additional parameters.

There are two fault-specific parameters: *fault severity* and *signal rotation*. These two parameters are specific to voltage sag generation. *Fault severity* controls the intensity of the deformations of amplitudes and phase angles of each of the phases individually. With *fault severity* being equal to 1-V (Appendix A; Formulas (A1)–(A7)), the multiplicative amplitude delta and additive phase angle delta are calculated. After applying V as 1—the *fault severity* in the adequate set of formulas—newly calculated phasors represent phasors during the faulty state. Default scaling of the amplitudes of new phasors is already in (0, 1) the interval, so the calculated amplitudes represent the multiplicative amplitude delta. On the contrary, the phase angle delta is calculated as a difference between the faulty and non-faulty phasor.

Table 1. The list of generator parameters with their meanings and acceptable values.

Parameter Name	Meaning	Acceptable Values
signal duration	Time duration of the sample	A real number higher than 0
fault start time	The moment at which the fault occurs in the sample	A real number from [0, signal_duration]
fault duration	Time duration from the fault start time to the moment of recovery	A real number from [0, signal_duration-fault_start_time]
nominal rms	Voltage value in the grid set by the regulation laws	A real number higher than 0
signal frequency	No. of sinusoids per second	A real number higher than 0
sampling frequency	No. of sampling points per second	A real number higher than 0
no-fault amplitude delta	Non-faulty scaling amplitude values, i.e., 90% to 110%	A real number from (0.9, 1.1)
phase angle delta	Phase angle offset of the whole signal	A real number from [0, 2π}
no. of smoothing timepoints	No. of points that smoothen the transition from normal to faulty state and vice-versa.	An integer higher than or equal to 0
fault severity	The intensity of the changes during fault; equal to 1-V (Appendix A; formulas (A1)–(A7))	A real number from [0, 1]
signal rotation	Changes the phasor around which the fault is based for asymmetric faults	{'**A**BC', '**B**CA', '**C**AB'}
noise percentage	White noise in percent in respect to the *nominal rms*	A real number higher than or equal to 0
label	Type of the fault to be generated	{NF, A, B, C, D, E, F, G}

The adequate formula set is picked concerning the label when its value is not NF. All voltage sag types except for type A are not symmetrical around each of the phasors and are based around phase A. To cover cases of these faults occurring based around phases B and C, a simple signal rotation is introduced. The comprehensive list of the voltage sag formulas used is shown in Appendix A in Table A1. A step-by-step algorithm for generating a signal is shown in Appendix B, Algorithm A1.

4.2. Data Preparation

Each synthetic dataset consists of two sub-datasets, one for training and one for validation. There are four variants of the synthetic datasets made by introducing two types of augmentations to the data. The instantaneousness of the change from normal to faulty state is unnatural, so the first augmentation smooths this transition. The second augmentation is varying the base voltage amplitude of the normal signal within the bounds of 90% to 110% of the nominal. This simulates the imperfections that fall short of errors in the normal operation of an electrical grid. The dataset without any modifications we termed 'sterile.' The set with just smoothing we labeled 'smooth', and the set with just varied base voltage amplitude we labeled 'varied_normal.' The set with both modifications applied was labeled 'varied_smooth.' The effects of the upgrades are illustrated in Figure 3. The parameters used for the generation of the synthetic datasets are shown in Appendix C, Table A2.

The shares between classes in datasets are shown in Table 2. For most of the classes, the number of classes among the data is identical. Only NF and A classes do not have a proportional share of datapoints because signal rotation is not present in classes NF and A. The reason for the higher share of NF class compared to class A is the labeling of faulty signals as not faulty when the RMS in any of the phases does not drop below the 90% nominal RMS value. The sizes of the datasets are shown in Table 3. Figure 4 shows an example from each of the classes from the datasets with voltage sag types shown with a 0.66 (66%) severity each.

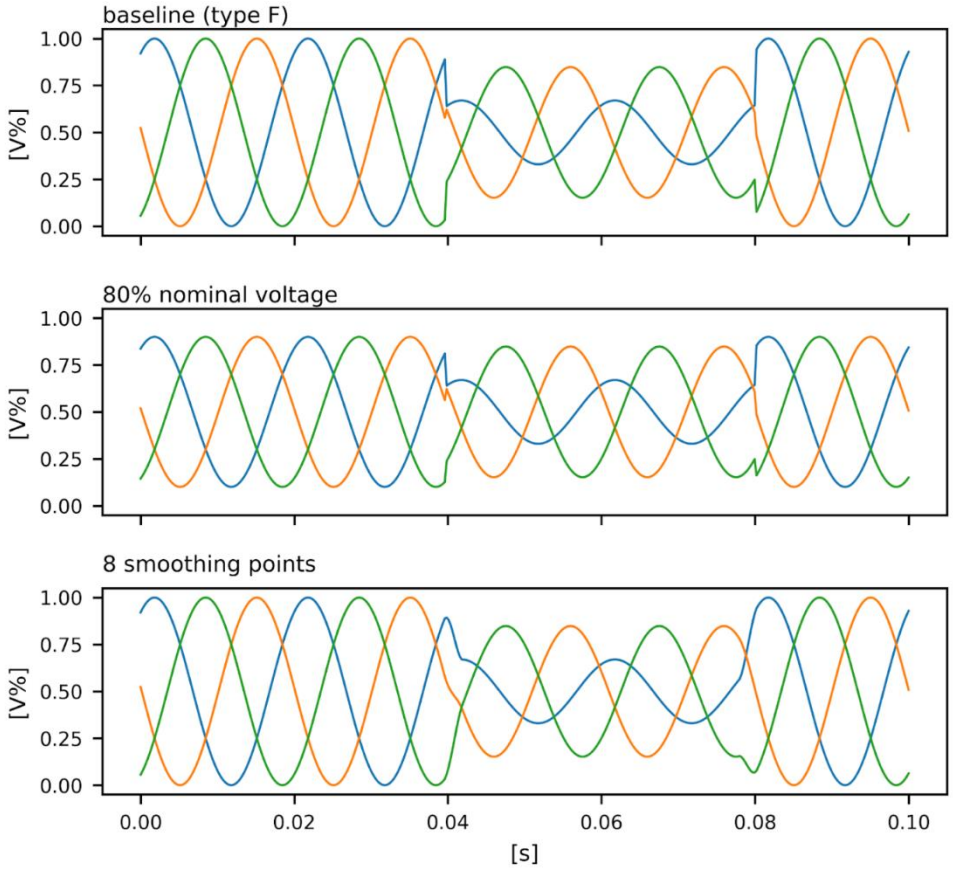

Figure 3. An example of augmentations introduced with smoothing and varying the amplitudes of normal segments.

Table 2. Shares of classes in the datasets. NF and A classes are lesser due to the lack of signal rotation. The higher share of NF class to A is a result of faulty signals being labeled as NF when the RMS is higher than 90% of the nominal RMS value.

None	A	B	C	D	E	F	G
~10%	~4%	~14%	~14%	~14%	~14%	~14%	~14%

Table 3. The number of instances per each set in the training and validation datasets. The 'varied_' datasets have higher numbers because of additional variations to normal segments.

Subsets	Sterile, Smooth	Varied_Normal, Varied_Smooth
training	311,520	1,557,728
validation	166,784	833,984

The final step is normalizing the data based on the nominal RMS value. The normalization is designed in such a way that if the signal is in the range from 0% to 100% of its nominal voltage value, then the resulting normalization interval is [0, 1]. The formula for this would be as follows:

$$V_{new} = \frac{V_{old}}{2 \cdot V_{nominal}} + 0.5 \quad (1)$$

That means if the nominal voltage value for the signal is 240 V, then a signal with a range of [−264 V, 264 V] (110% of the nominal value) is normalized to the interval [−0.05, 1.05].

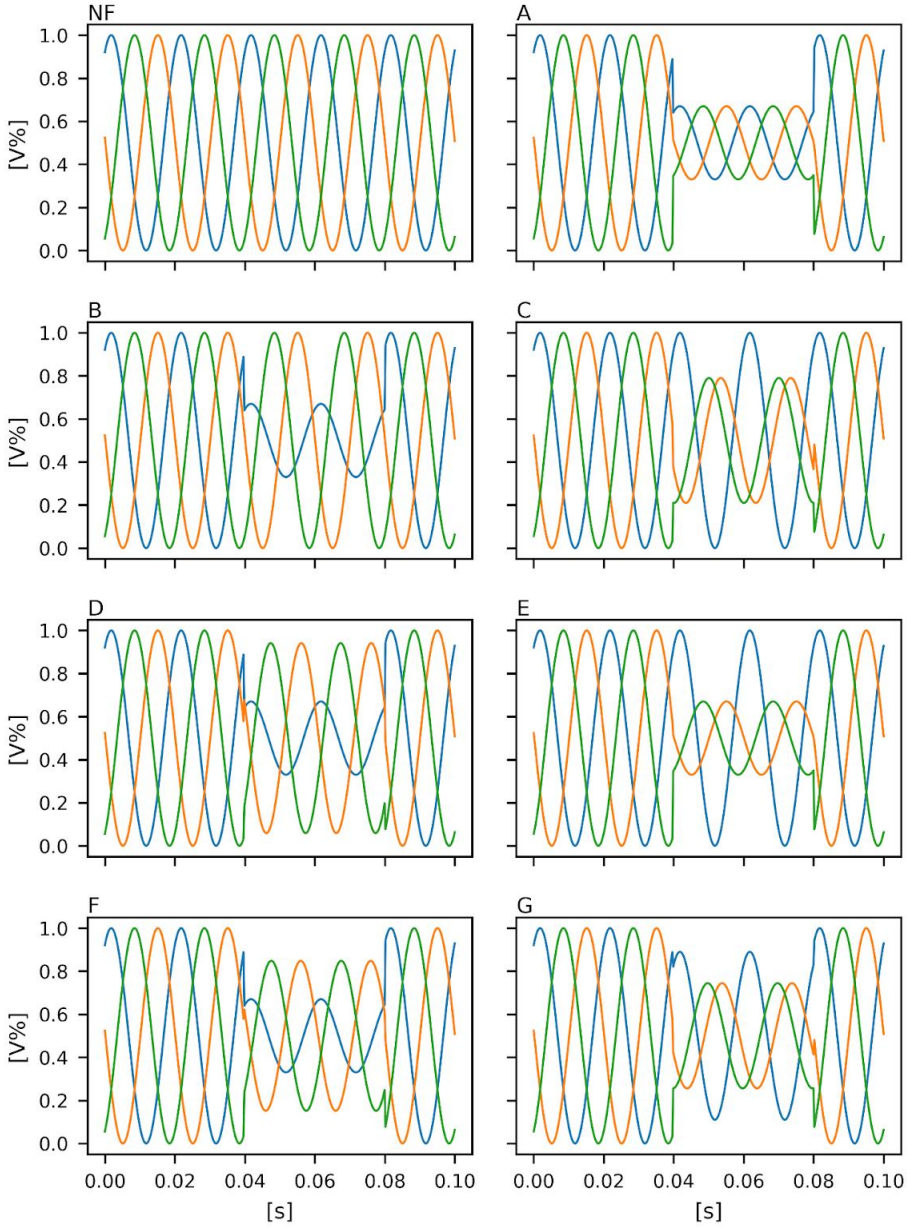

Figure 4. Examples of signals for each class with fault severity being 0.66 for classes other than NF.

Each dataset instance consists of two parts. The first part is the generated signal. It is an array of 410 timepoints for each of the three phases. The second part is the label. It is a one-hot vector where the class is represented by the placement inside of the vector, and the label itself is one for the one class that is present in the data and zeros in all the other places. The label vector contains eight elements because there are seven different voltage sag classes and one class that represents that, while there may be a voltage sag, it is not severe enough to count it as a fault according to prevailing standards, i.e., the RMS value of all three phases never drops below the 90% nominal RMS. The main reason for the output vector being a one-hot vector is because this work focuses on single voltage sag type faults. We labeled the phases 'A, B, and C' in a counterclockwise direction.

4.3. Model Training

In the case of models, the original LSTM model from [20] was improved with batch normalization layers, so now it has the structure as shown in Figure 5a. In Figure 5, each block represents one layer or a stack of multiple layers, in which case the layers are separated with ">" character symbolizing the order of layers. The layers are presented

top-to-bottom meaning that input data are fed to the topmost layer, whilst the bottommost layer produces the class/label. The width of the blocks depicts the change in volume of the feature space. Figure 5b presents the second model. Differences to the first model are colored blue. This upgrade was introduced by adding more convolutional layers at the beginning. The convolutional layers are computationally less demanding than the LSTM layers, so they are added to do the low-level feature extraction before feeding the feature data to the LSTM layers. The third model was designed with a few more upgrades so its architecture is significantly more complex than the previous models. This model's architecture is shown in Figure 5c.

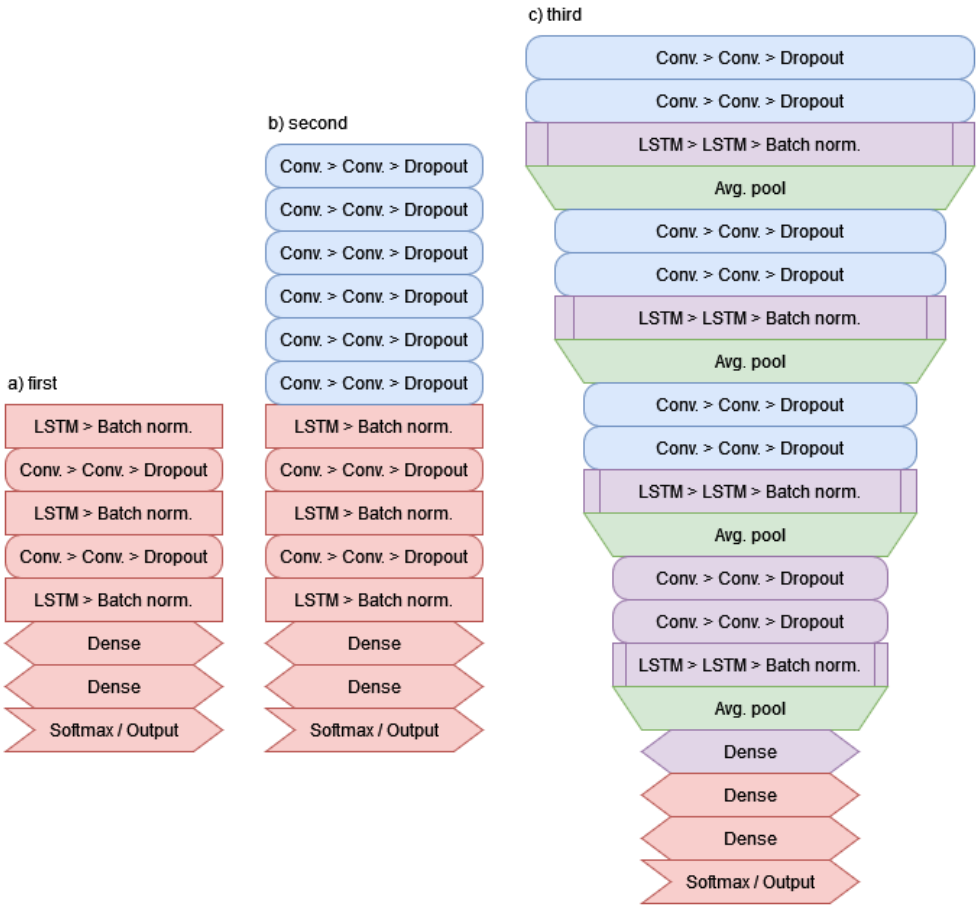

Figure 5. Diagrams of the three model variants used in the research. (**a**) Initial model. (**b**) Upgraded initial model with changes colored blue. (**c**) Most complex model. The purple color shows improvements to the parts of (**a**,**b**) models with additional layers. The green color is an addition to a new type of layer.

The purple model accents the changes to the parts of the first and second models' architectures by stacking additional layers to the existing ones. The idea was to design a few stages of feature calculation. There are four convolutional layers at the beginning of each stage calculating the lower-level features. Following are two LSTM layers that calculate the temporally coupled features. After the batch normalization layers, the average pooling layers, colored green, are introduced to force the model to extract and map the significant features to a smaller feature space, which is then used as an input to the next stage. The ending stage consists of fully connected layers.

Listed three models are used in place of the "LSTM model" in Figure 2, where for each model, one trained model is produced per variant of synthetic dataset during the training stage, and later each trained model is tested on real-world data during the testing stage, counting 12 variants in total.

All the models have the same input and output shapes. Input shape for each model is a tuple (batch_size, 410, 3) and the output shape is (batch_size, 8). The models were developed using Python 3.9 with Conda environment, and the TensorFlow (2.4.1) framework. The hardware used for the experiment is a workstation with an AMD Ryzen 7 1700 CPU, 32 GB of RAM, and a GeForce GTX Titan X GPU. The training of the models was done using Adam [49] as an optimization algorithm with a learning rate of 1e-4. The loss function used is categorical cross-entropy. The training lasted for 10 epochs with a batch size of 32. The training parameters not listed are left with their default implementation values.

4.4. Voltage Sag Classification and Data Analysis

To test how LSTM models trained on synthetic data perform on real-world data, we formed a dataset of real-world voltage measurements. The trained models are designed and trained for the classification of single voltage sag type data, which are the constituents of the compiled real-world dataset. The real-world dataset consists of a total of 47 real-world voltage measurements. Table 4 shows the distribution of measurements per class of the real-world dataset.

Table 4. The first row shows the total number of instances per class. The second row shows the share of the class in the dataset.

NF	A	B	C	D	E	F	G
10	10	2	18	3	2	2	0
21.27%	21.27%	4.26%	38.30%	6.38%	4.26%	4.26%	0%

All the signals present in the real-world dataset have some form of voltage sag present in the signal. The NF-labeled instances are signals for which the RMS does not drop below 90% of the nominal value at any point in any of the phases. As can be seen from the table, signals for classes B, E, F, and G were not present in sufficient numbers to make any sort of determination of accuracy and were therefore elided from the final tally of results, with the inclusion limit being at least 5%.

Before the data are used in validation, there are four preprocessing steps done to classify the data. The first one is fixing the orientation of ABC phases to match the counterclockwise direction. The second one is resampling the signals to match the frequency at which the synthetic data were generated. After this step, data normalization was done using the recorded nominal voltage values of each of the signals. Finally, considering that, in the training data the signals were trained with signals that start with 0.04 s of the normal state, each of the signals was sampled for 410 time points in a sliding window manner 20 times, capturing the starting moment of the fault in the vicinity of the 0.04 s mark. These steps are shown in Appendix B, Algorithm A2.

This network is foremost designed to be used as a classification step after the successful detection of the anomaly in advance. Without the loss of generality, it can be assumed that the detection step will detect the approximate moment of the beginning of the anomaly and that the recorded window will be long enough to extract 410 time point samples with the anomaly beginning approximately 0.4 s from the start of the sample. While testing the models, we measured MCC and three different versions of accuracy. Accuracy over instances, accuracy over windows of a signal with 50%+ correctness, and accuracy over the whole window of a signal. Accuracy of the instances is the number of correctly classified samples divided by the total number of samples. All of the signals were sampled 20 times using the sliding window technique, so there are 20 samples per signal. Window accuracies are the numbers of correctly classified signals divided by the number of signals where in the first case the correctly classified window has at least 11 correctly classified samples, and in the case of the second version, all 20 samples in a window must be correct. For this paper, all combinations of models and datasets were trained and each of the models is tested on the real-world dataset. During training after every epoch, the state of the model,

i.e., weights, is saved. For each trained model, four to six weights were picked based on the performance on the validation dataset, and only the weights with the best result on real-world dataset were included in the final results.

5. Experimental Results, Analysis, and Discussion

The results for each variant of the third model, trained on each synthetic dataset variant, are shown in Table 5.

Table 5. Results of the best-performing model on the real-world dataset without the results for B, E, F, and G classes.

Training Dataset	MCC	Instance Acc	Window Acc	Whole Window Acc
sterile	0.8787	91.46%	92.68%	87.80%
smooth	0.8526	89.51%	90.24%	85.36%
varied_normal	0.8318	87.56%	87.80%	85.36%
varied_smooth	0.8448	88.65%	87.80%	85.36%

The best performance on the real-world dataset was exhibited by the third model trained on the sterile dataset variant. The full results of the experiments, not shown here, suggest that higher model complexity has the highest impact on performance improvements. The variations in training datasets gave better performances for the first and second models, but the third model did not show such trends. During an inspection of the behavior of the models during training, it was observed that the models tend to overfit relatively quickly. This is something that should be addressed in future work by further researching architectures and training processes.

It was expected for the most complex model to achieve the best results. Higher model complexity allows the model to properly learn the needed features. Since the proposed classifier is supposed to be used after the detection step, it can be expected that instances that would fall under the NF class would not come to the classification step in most cases. Because of this, if the classifier finds the appropriate type of voltage sag in the signal, which is otherwise labeled as NF, then that classification is counted as correct. In the real-world dataset, each such instance is a type A voltage sag for which the RMS value is near 90% of the nominal voltage value. Since the number of instances of classes B, E, F, and G are two or less, i.e., less than 5% of the total number of samples, they were omitted from the overall results.

Table 6 provides insight into the accuracy of the method per each class for the best-performing model, i.e., the third model trained on the sterile version of the dataset. The NF class has lower accuracy than the rest, but that is to be expected since deformations of voltage sags with low severities are very slight and are hard to differentiate between in real-world signals.

Table 6. Results per class of the best-performing model on the real-world dataset.

NF	A	B	C	D	E	F	G
68.04%	90.29%	100%	97.77%	100%	50%	0%	None

The performance of the classifier can be seen in detail in Figure 6. The y-axis in the figure shows the true classes, as they are labeled in the dataset. The x-axis shows the labels predicted by the classifier. Correct classification increases the number in the diagonal fields of the confusion matrix. Because of this, the lighter color on the diagonal and the darker color on the rest of the matrix are better. The darkest color means 0 instances of the given combination of true and predicted classes, and the lightest color is over 350 instances.

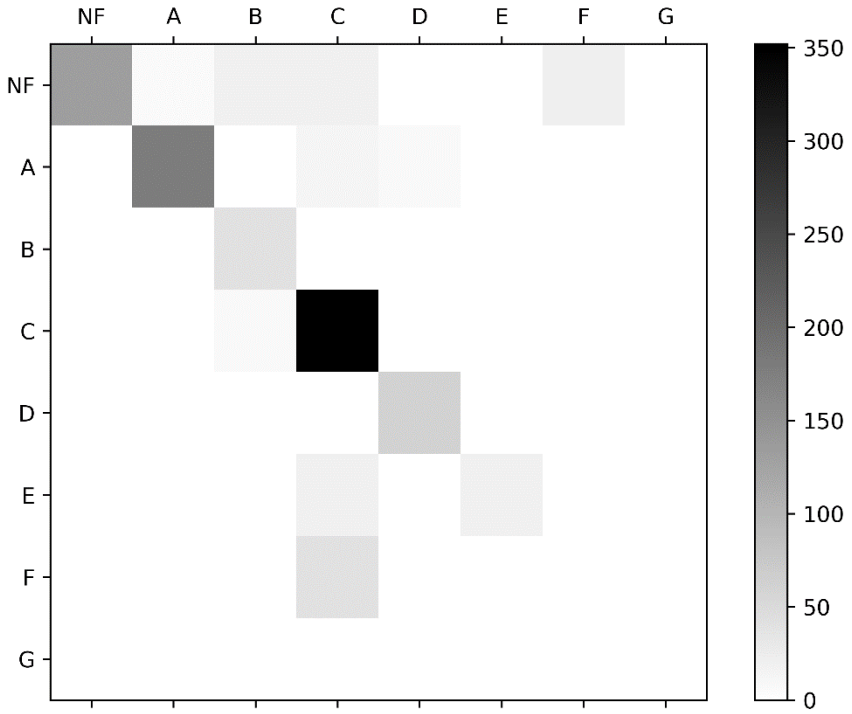

Figure 6. Confusion matrix for results of the best performing model on the real-world dataset.

Classes B, E, and F have results that are difficult to give credence to because of the extremely low number of examples, which is why they were not a part of our final tally of model performance. G was not present in our real-world dataset at all and remains entirely untested. For classes A, C, and D, performance is excellent with nearly 100% accuracy. The NF class is more complex since it includes examples of both signals with no voltage sag whatsoever and signals with a voltage sag too small to qualify as a fault. The results for this class depend heavily on the somewhat arbitrary dividing line between what constitutes a fault and what does not. Classifying the type correctly when the overall severity of the fault is minute remains beyond the predictive power of our model since it is difficult to distinguish noise from a very low-severity fault. To be able to accurately measure the performance of the trained models a better real-world set should be assembled, which would have instances of each of the classes with varying levels of severities of the voltage sags and be balanced with approximately equal representation for each class.

In terms of time, Table 7 lists average times spent per epoch of training for each model and each dataset variant. The total time spent on training in our case can be approximately calculated as 10 times the time needed for one epoch. This means that training the most complex model on variants with varied normal segments took roughly 1 day and 13 h.

Table 7. The time needed for each model trained on each dataset for one epoch.

Model	Sterile, Smooth	Varied_Normal, Varied_Smooth
1. model	~770 s	~3720 s
2. model	~1280 s	~6220 s
3. model	~1850 s	~13,400 s

The classification performance in respect to time is shown in Table 8. Times shown are the time demands of each model per batch. There are 32 signals in one batch, so the last column shows the estimated number of classifications each model can do per one second.

Table 8. The time needed for classification of one batch (32 signals) and estimated number of classifications per second.

Model	Batch Classification Time	Classifications per Second Est.
1. model	~77 ms	415
2. model	~128 ms	250
3. model	~210 ms	152

To put the achieved results in perspective, we selected a few of the similar solutions that use machine learning in Table 9.

Table 9. Overview of a selection of similar solutions.

Method	Classification	Training Dataset	Testing Dataset	Feature Extraction	Accuracy
2D CNN [47]	ACD 7-class	real	real	2D polarized eclipse images	97.72%
LSTM [46]	ACD 7-class	real	real	RMS	93.40%
Bi-LSTM [48]	7-class sag sources	synthetic	real	no	99.07%
Proposed	ABC 7-class	synthetic	real	no	92.68%

The accuracies listed in Table 9 show that the solution proposed in this paper is relevant in comparison. Direct comparison was not possible since none of the related works used the ABC classification. The closest in terms of methodology is the work presented in [48], which differs in the classification and training dataset. In [48], seven voltage sag source classes are combinations of one-phase faults (simple and multi-stage), generators, and transformer-induced voltage sags. The dataset used consists of simulated signals from the MATLAB/Simulink scheme. The rest of the presented works all use some form of feature extraction. In comparison, since in [46] only RMS is the only extracted feature, it is not possible to do ABC classification. In that sense, feature extraction can help in achieving higher accuracy in some cases, i.e., like in [47], but it also results in a loss of potentially useful features. In terms of the training dataset, presented works have relatively small datasets, which show the difficulty of compiling a big reliably labeled dataset. In this work, data generation proved flexible and reliable and provided the ability to design a dataset and train a deep NN model, which has great generalization ability.

Compared to the rest of the approaches, there are two main differences in the proposed approach. The first difference is in the training dataset. Other works use real-world datasets or datasets made from simulations of various grid configurations. Here, as can be seen from the results, the models were able to generalize enough to be able to classify the voltage sags in real-world signals from being trained on very simplified signals. This opens the possibilities of making arbitrarily large datasets in a short amount of time compared to modeling different grid configurations or measuring real-world signals, where the frequencies of various types of faults are highly disproportional. Additionally, with the generation of the signals, greater variations can be achieved in the dataset. The second difference is in the end-to-end approach, where, contrary to the usual approaches, manual feature engineering was circumvented. Additionally, this approach allows for the specific deformations in the signal, which would not be calculated as features otherwise and would not be present as input data, to be used for classification.

The results that the proposed model achieved are comparable to the state-of-the-art models, particularly when it is taken into account that we used an end-to-end approach. The state-of-the-art models achieve accuracies of 95% or more, but they do not employ an end-to-end approach, with the key difference being mainly the classification of various PQDs such as voltage sags, voltage swells, harmonics, notches, etc. The few researchers that use an end-to-end approach do not go further into the classification of subtypes of PQDs. The paper that is closest to our work classifies voltage sag causes and not types of voltage sags. Our work further confirms the validity of presented approaches, namely,

usage of end-to-end approach and training a model on a highly synthetic dataset for use on real-world signals.

6. Conclusions

This paper presents an end-to-end deep LSTM classifier of voltage sags that achieves an accuracy up to 92.68% (Table 5, sterile dataset, window accuracy) for the best model. This result shows the potential in an end-to-end approach. In place of manual feature extraction, the end-to-end approach shows that feature extraction and selection can be done automatically. It is a general opinion that data preparations take about 80% of the time spent on the research and that the end-to-end approach improves this time significantly [50]. For an end-to-end approach to be done properly, a big amount of reliably labeled data is needed, which is non-trivial to obtain in the case of PQ events. Results presented in this work show that deep machine learning models have great generalization ability, and this allowed for the use of a synthetic dataset. The voltage sag generator presented in this paper proved to be a reliable and flexible way of attaining a big, labeled dataset for training. In general, it is better to have real-world data; however in the light of difficulties in obtaining an adequate PQ events dataset, the use of simplified generated data is at least a great starting point for deep ML models, which can later be refined using real-world datasets.

The results also show that there is room for improvement of the proposed method. One path can be aimed towards exploring the design of standardized synthetic datasets for training and possibly validation of ML models for the classification of PQ events. It would also be interesting to have a dataset that would estimate the cost of misclassification. Another path would be the exploration of improvements to the model architecture.

Author Contributions: Conceptualization, D.D., D.B.G. and V.A.K.; methodology, R.T., A.M.S. and V.B.P.; software, R.T., G.G. and V.B.P.; validation, D.B.G., A.M.S. and V.B.P.; formal analysis, V.B.P. and D.B.G.; investigation, G.G. and V.B.P.; resources, R.T.; data curation, V.A.K. and A.M.S.; writing—original draft preparation, R.T., G.G. and D.D.; writing—review and editing, R.T., D.D., G.G., D.B.G., V.B.P., V.A.K. and A.M.S.; visualization, R.T., G.G. and D.D.; supervision, V.A.K., D.B.G. and D.D.; project administration, R.T. and D.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The code for the generator, the models, and training and testing, as well as the weights for the best-performing model, are open-sourced and available at https://bitbucket.org/sara_e21/ee2021-mdpi-vsc/ (accessed on 30 January 2022).

Acknowledgments: This research (paper) has been supported by the Ministry of Education, Science, and Technological Development through project no. 451-03-68/2022-14/ 200156 "Innovative scientific and artistic research from the FTS (activity) domain".

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Formulas used to generate datasets used in this paper are presented in Table A1. In these formulas, parameter V is equal to 1—fault severity. For more details, read Section 4.1.

Table A1. List of phasor formulas for each phase per voltage sag type.

Voltage Sag Type	Phasor Formulas	
A	$V_a = V$ $V_b = V\left(\frac{-1}{2} - \frac{\sqrt{3}}{2}j\right)$ $V_c = V\left(\frac{-1}{2} + \frac{\sqrt{3}}{2}j\right)$	(A1)
B	$V_a = V$ $V_b = \frac{-1}{2} - \frac{\sqrt{3}}{2}j$ $V_c = \frac{-1}{2} + \frac{\sqrt{3}}{2}j$	(A2)
C	$V_a = 1$ $V_b = \frac{-1}{2} - \frac{\sqrt{3}}{2}jV$ $V_c = \frac{-1}{2} + \frac{\sqrt{3}}{2}jV$	(A3)
D	$V_a = V$ $V_b = V\frac{-1}{2} - \frac{\sqrt{3}}{2}j$ $V_c = V\frac{-1}{2} + \frac{\sqrt{3}}{2}j$	(A4)
E	$V_a = 1$ $V_b = V\left(\frac{-1}{2} - \frac{\sqrt{3}}{2}j\right)$ $V_c = V\left(\frac{-1}{2} + \frac{\sqrt{3}}{2}j\right)$	(A5)
F	$V_a = V$ $V_b = \frac{-\sqrt{3}}{3}j + V\left(\frac{-1}{2} - \frac{\sqrt{3}}{6}j\right)$ $V_c = \frac{\sqrt{3}}{3}j + V\left(\frac{-1}{2} + \frac{\sqrt{3}}{6}j\right)$	(A6)
G	$V_a = \frac{2}{3} + \frac{1}{3}V$ $V_b = \frac{-1}{3} + V\left(\frac{-1}{6} - \frac{\sqrt{3}}{2}j\right)$ $V_c = \frac{-1}{3} + V\left(\frac{-1}{6} + \frac{\sqrt{3}}{2}j\right)$	(A7)

Appendix B

The process of generating one signal is shown in Algorithm A1. The first step of the generation of one three-phase voltage signal is a calculation of a set of time points based on the duration of the signal and sampling frequency. The next step is to separate this set into normal and faulty segments for signals that are faulty based on the faults' starting and ending time points. With the time points set segmented properly, the signal of the normal segments can be already generated concerning signal frequency, RMS of the nominal voltage value, non-fault amplitude, and phase angles. If there is no fault in the signal, there will be no faulty segments and the algorithm ends here. In the case of the generation of a faulty signal, the next step would be the acquisition of the function that is used for the calculation of amplitude and phase deltas for that fault. With this function, they are calculated based on the fault's severity by inputting it into the adequate formula (as listed in Appendix A; formulas (A1)–(A7)) as $V = 1$—*fault_severity*.

The faulty signal is generated the same way as the no-faulty one and is then perturbed using the computed amplitude and phase deltas. If smoothing is enabled, the number of points specified by *no. of smoothing time points* is replaced in the faulty signal segment, where it borders a no-faulty segment. They are replaced by points computed to blend the faulty and no-faulty segments smoothly. Before the final step, if there is signal rotation to be applied and the voltage sag type is not symmetrical around all three phasors, then the signals are rotated. Finally, the last step is to concatenate generated normal and faulty segments back into one signal.

Algorithm A1 Step-by-step process of generating one synthetic signal.

Generate one three-phase signal:
Generate time points(signal_duration, sampling_frequency)
Slice time time points into groups for normal and faulty state(fault_start_time, fault_duration)
Generate normal state segments(nominal_rms, signal_frequency, non_fault_amplitude_delta, phase_angle_delta, noise_percentage)
IF *label is not NF* **THEN**
For voltage sag class acquire function for calculating amplitude and phase deltas based on severity(label)->function
*Calculate amplitude and phase deltas(*function*)*->fault_amp_delta, fault_ang_delta
Generate faulty state segment applying calculated deltas(nominal_rms, signal_frequency, non_fault_amplitude_delta, phase_angle_delta, noise_percentage, fault_start_time, fault_duration, signal_rotation, fault_amp_delta, fault_ang_delta*)*
IF *no_of_smoothing_time points > 0* **THEN**
Cut out designated no. of points from the start of the faulty segment(no_of_smoothing_timepoints)
IF *normal segment follows the faulty segment* **THEN**
Cut out designated no. of points from the end of the faulty segment(no_of_smoothing_timepoints)
END IF
Interpolate the missing values from the surrounding points;
END IF
IF *signal rotation is in* {'**B**CA', '**C**AB'} *and label is not A* **THEN**
Rotate the signals(signal rotation)
END IF
Concatenate normal and faulty states into one signal;
END IF

A step-by-step process of preparing a real-world measurement for verification is described in Algorithm A2.

Algorithm A2 Step-by-step process of preparing a real-world measurement for verification.

Preparing a single real signal for verification:
IF *phases are rotated in clockwise direction* **THEN**
Switch A and B phases;
END IF
IF *signal sampling is not 4096 Hz* **THEN**
Resample the signal to 4096 Hz;
END IF
Normalize the signal using the nominal voltage value (formula 1);
IF *there is a deformation in the signal measurement* **THEN**
Find the time point of the beginning of the deformation;
Pick a sample of 0.1 s duration so that deformation starts at 0.04s;
Using sliding-window with 1 time point step pick also 9 samples before and 10 samples after the already chosen sample;
ELSE
Choose 20 samples using the sliding window technique with 1 time point step from anywhere in the signal;
END IF

Appendix C

A detailed list of values used during the generation of training and validation datasets compiled are described in Table A2.

Table A2. List of all values used during generation for each of the sets, subsets, and parameters.

Parameter	Subset	Sterile	Smooth	Varied_Normal	Varied_Smooth
phase angle delta	training validation	256 values chosen uniformly from [0, 6.25] 19 values chosen uniformly from [0.5, 6]			
signal duration	training validation	0.1 s 0.1 s			
fault duration	training validation	{0.02, 0.04} {0.01, 0.02, 0.03, 0.04, 0.05, 0.06}			
sampling frequency	training validation	4096 Hz 4096 Hz			
signal frequency	training validation	50 Hz 50 Hz			
fault severity	training validation	32 values chosen uniformly from [0, 1] 77 values chosen uniformly from [0.01, 0.99]			
nominal rms	training validation	230 V 230 V			
fault start time	training validation	0.04 s 0.04 s			
noise percentage	training validation	0 0			
no-fault amplitude delta	training validation	1 1		5 values chosen uniformly from [0.91, 1.09] 5 values chosen uniformly from [0.91, 1.09]	
no. of smoothing timepoints	training validation	0 0	4 4	0 0	4 4

References

1. Bollen, M.H. *Understanding Power Quality Problems: Voltage Sags and Interruptions*; IEEE Press: Piscataway, NJ, USA, 2000; ISBN 978-0-7803-4713-7.
2. Lee, C.-Y.; Shen, Y.-X. Optimal Feature Selection for Power-Quality Disturbances Classification. *IEEE Trans. Power Deliv.* **2011**, *26*, 2342–2351. [CrossRef]
3. Ma, J.; Zhang, J.; Xiao, L.; Chen, K.; Wu, J. Classification of Power Quality Disturbances via Deep Learning. *IETE Tech. Rev.* **2017**, *34*, 408–415. [CrossRef]
4. Ribeiro, E.G.; Mendes, T.M.; Dias, G.L.; Faria, E.R.S.; Viana, F.M.; Barbosa, B.H.G.; Ferreira, D.D. Real-Time System for Automatic Detection and Classification of Single and Multiple Power Quality Disturbances. *Measurement* **2018**, *128*, 276–283. [CrossRef]
5. Jeba Singh, O.; Prince Winston, D.; Chitti Babu, B.; Kalyani, S.; Praveen Kumar, B.; Saravanan, M.; Cynthia Christabel, S. Robust Detection of Real-Time Power Quality Disturbances under Noisy Condition Using FTDD Features. *Automatika* **2019**, *60*, 11–18. [CrossRef]
6. Liu, J.; Tang, Q.; Qiu, W.; Ma, J.; Qin, Y.; Sun, B. Automatic Power Quality Disturbance Diagnosis Based on Residual Denoising Convolutional Auto-Encoder. *Appl. Sci.* **2021**, *11*, 7637. [CrossRef]
7. Alam, M.R.; Bai, F.; Yan, R.; Saha, T.K. Classification and Visualization of Power Quality Disturbance-Events Using Space Vector Ellipse in Complex Plane. *IEEE Trans. Power Deliv.* **2021**, *36*, 1380–1389. [CrossRef]
8. Rahul. Review of Signal Processing Techniques and Machine Learning Algorithms for Power Quality Analysis. *Adv. Theory Simul.* **2020**, *3*, 2000118. [CrossRef]
9. Ozcanli, A.K.; Yaprakdal, F.; Baysal, M. Deep Learning Methods and Applications for Electrical Power Systems: A Comprehensive Review. *Int. J. Energy Res.* **2020**, *44*, 7136–7157. [CrossRef]
10. Chawda, G.S.; Shaik, A.G.; Shaik, M.; Padmanaban, S.; Holm-Nielsen, J.B.; Mahela, O.P.; Kaliannan, P. Comprehensive Review on Detection and Classification of Power Quality Disturbances in Utility Grid With Renewable Energy Penetration. *IEEE Access* **2020**, *8*, 146807–146830. [CrossRef]
11. Han, Y.; Feng, Y.; Yang, P.; Xu, L.; Xu, Y.; Blaabjerg, F. Cause, Classification of Voltage Sag, and Voltage Sag Emulators and Applications: A Comprehensive Overview. *IEEE Access* **2020**, *8*, 1922–1934. [CrossRef]
12. Stanisavljević, A.M.; Katić, V.; Dumnic, B.; Popadic, B. A Comprehensive Overview of Digital Signal Processing Methods for Voltage Disturbance Detection and Analysis in Modern Distribution Grids with Distributed Generation. *Acta Polytech. Hung.* **2019**, *16*, 125–149. [CrossRef]

13. Usman, A. An Efficient and High-Speed Disturbance Detection Algorithm Design with Emphasis on Operation of Static Transfer Switch. *Adv. Electr. Comp. Eng.* **2021**, *21*, 87–98. [CrossRef]
14. Katić, V.A.; Stanisavljević, A.M. Smart Detection of Voltage Dips Using Voltage Harmonics Footprint. *IEEE Trans. Ind. Appl.* **2018**, *54*, 5331–5342. [CrossRef]
15. Djokic, S.Z.; Milanovic, J.V.; Chapman, D.J.; McGranaghan, M.F.; Kirschen, D.S. A New Method for Classification and Presentation of Voltage Reduction Events. *IEEE Trans. Power Deliv.* **2005**, *20*, 2576–2584. [CrossRef]
16. Djokic, S. *Voltage Dip Immunity of Equipment Used in Installations: CIGRE Technical Report TR412*; CIGRE: Paris, France, 2010; ISBN 978-2-85873-099-5.
17. Bollen, M.H.J.; Gu, I.Y.H.; Axelberg, P.G.V.; Styvaktakis, E. Classification of Underlying Causes of Power Quality Disturbances: Deterministic versus Statistical Methods. *EURASIP J. Adv. Signal Process.* **2007**, *2007*, 79747. [CrossRef]
18. Sainath, T.N.; Weiss, R.J.; Wilson, K.W.; Li, B.; Narayanan, A.; Variani, E.; Bacchiani, M.; Shafran, I.; Senior, A.; Chin, K.; et al. Multichannel Signal Processing With Deep Neural Networks for Automatic Speech Recognition. *IEEE/ACM Trans. Audio Speech Lang. Processing* **2017**, *25*, 965–979. [CrossRef]
19. Tan, C.; Sun, F.; Kong, T.; Zhang, W.; Yang, C.; Liu, C. A Survey on Deep Transfer Learning. In Proceedings of the Artificial Neural Networks and Machine Learning—ICANN 2018, Rhodes, Greece, 4–7 October 2018; Kůrková, V., Manolopoulos, Y., Hammer, B., Iliadis, L., Maglogiannis, I., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 270–279.
20. Turović, R.; Dragan, D.; Stanisavljević, A.; Gojić, G.; Petrović, V.; Katić, V.; Gajić, D. Training an LSTM Voltage Sags Classificator on a Synthetic Dataset. In Proceedings of the 2021 21st International Symposium on Power Electronics (Ee), Novi Sad, Serbia, 27–30 October 2021; pp. 1–6.
21. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
22. Sherstinsky, A. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. *Phys. D Nonlinear Phenom.* **2020**, *404*, 132306. [CrossRef]
23. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
24. Khokhar, S.; Mohd Zin, A.A.B.; Mokhtar, A.S.B.; Pesaran, M. A Comprehensive Overview on Signal Processing and Artificial Intelligence Techniques Applications in Classification of Power Quality Disturbances. *Renew. Sustain. Energy Rev.* **2015**, *51*, 1650–1663. [CrossRef]
25. Sun, H.; Yi, H.; Yang, G.; Zhuo, F.; Hu, A. Voltage Sag Source Identification Based on Few-Shot Learning. *IEEE Access* **2019**, *7*, 164398–164406. [CrossRef]
26. Balouji, E.; Salor, O. Classification of Power Quality Events Using Deep Learning on Event Images. In Proceedings of the 2017 3rd International Conference on Pattern Recognition and Image Analysis (IPRIA), Shahrekord, Iran, 19–20 April 2017; pp. 216–221.
27. Mohan, N.; Soman, K.P.; Vinayakumar, R. Deep Power: Deep Learning Architectures for Power Quality Disturbances Classification. In Proceedings of the 2017 International Conference on Technological Advancements in Power and Energy (TAP Energy), Kollam, India, 21–23 December 2017; pp. 1–6.
28. Rai, P.; Londhe, N.D.; Raj, R. Fault Classification in Power System Distribution Network Integrated with Distributed Generators Using CNN. *Electr. Power Syst. Res.* **2021**, *192*, 106914. [CrossRef]
29. Ren, H.; Hou, Z.J.; Vyakaranam, B.; Wang, H.; Etingov, P. Power System Event Classification and Localization Using a Convolutional Neural Network. *Front. Energy Res.* **2020**, *8*, 607826. [CrossRef]
30. Ekici, S.; Ucar, F.; Dandil, B.; Arghandeh, R. Power Quality Event Classification Using Optimized Bayesian Convolutional Neural Networks. *Electr. Eng.* **2021**, *103*, 67–77. [CrossRef]
31. Rodrigues Junior, W.L.; Silva Borges, F.A.; Lira Rabelo, R.d.A.; de Lima, B.V.A.; Almeida de Alencar, J.E. Classification of Power Quality Disturbances Using Convolutional Network and Long Short-Term Memory Network. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; pp. 1–6.
32. Wang, S.; Chen, H. A Novel Deep Learning Method for the Classification of Power Quality Disturbances Using Deep Convolutional Neural Network. *Appl. Energy* **2019**, *235*, 1126–1140. [CrossRef]
33. Golik, P.; Tüske, Z.; Schlüter, R.; Ney, H. Convolutional Neural Networks for Acoustic Modeling of Raw Time Signal in LVCSR. In Proceedings of the 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, 6–10 September 2015.
34. Sainath, T.N.; Weiss, R.J.; Wilson, K.W.; Narayanan, A.; Bacchiani, M.; Senior, A. Speaker Location and Microphone Spacing Invariant Acoustic Modeling from Raw Multichannel Waveforms. In Proceedings of the 2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), Scottsdale, AZ, USA, 13–17 December 2015; pp. 30–36.
35. Salvati, D.; Drioli, C.; Foresti, G.L. Urban Acoustic Scene Classification Using Raw Waveform Convolutional Neural Networks. 2019. Available online: https://dcase.community/documents/challenge2019/technical_reports/DCASE2019_Salvati_35.pdf (accessed on 30 January 2022).
36. Li, C.; Yu, L.; Zhang, A.; He, Q.; Yang, W.; Duan, Z. A Novel Bearing Fault Diagnosis of Raw Signals Based on 1D Residual Convolution Neural Network. In Proceedings of the 2020 International Conference on High Performance Big Data and Intelligent Systems (HPBD IS), Shenzhen, China, 23 May 2020; pp. 1–6.
37. Eren, L.; Ince, T.; Kiranyaz, S. A Generic Intelligent Bearing Fault Diagnosis System Using Compact Adaptive 1D CNN Classifier. *J. Signal Processing Syst.* **2019**, *91*, 179–189. [CrossRef]

38. Valtierra-Rodriguez, M.; de Jesus Romero-Troncoso, R.; Osornio-Rios, R.A.; Garcia-Perez, A. Detection and Classification of Single and Combined Power Quality Disturbances Using Neural Networks. *IEEE Trans. Ind. Electron.* **2014**, *61*, 2473–2482. [CrossRef]
39. Axelberg, P.G.V.; Gu, I.Y.-H.; Bollen, M.H.J. Support Vector Machine for Classification of Voltage Disturbances. *IEEE Trans. Power Deliv.* **2007**, *22*, 1297–1303. [CrossRef]
40. Subhani, S.; Gang, M.; Cobben, J.F.G. Automatic Classification of Voltage Dip Root Causes via Pattern Recognition. In Proceedings of the 2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC), Florence, Italy, 7–10 June 2016; pp. 1–5.
41. Alam, M.R.; Muttaqi, K.M.; Bouzerdoum, A. A New Approach for Classification and Characterization of Voltage Dips and Swells Using 3-D Polarization Ellipse Parameters. *IEEE Trans. Power Deliv.* **2015**, *30*, 1344–1353. [CrossRef]
42. Erişti, H.; Demir, Y. Automatic Classification of Power Quality Events and Disturbances Using Wavelet Transform and Support Vector Machines. *IET Gener. Transm. Distrib.* **2012**, *6*, 968–976. [CrossRef]
43. Sha, H.; Mei, F.; Zhang, C.; Pan, Y.; Zheng, J. Identification Method for Voltage Sags Based on K-Means-Singular Value Decomposition and Least Squares Support Vector Machine. *Energies* **2019**, *12*, 1137. [CrossRef]
44. Nagata, E.A.; Ferreira, D.D.; Bollen, M.H.J.; Barbosa, B.H.G.; Ribeiro, E.G.; Duque, C.A.; Ribeiro, P.F. Real-Time Voltage Sag Detection and Classification for Power Quality Diagnostics. *Measurement* **2020**, *164*, 108097. [CrossRef]
45. Chia, M.H.; Khambadkone, A.M. Subcycle Voltage Dip Classification Using Matrix Pencil Method With Ellipse Fitting Algorithm. *IEEE Trans. Ind. Appl.* **2015**, *51*, 1660–1668. [CrossRef]
46. Balouji, E.; Gu, I.Y.H.; Bollen, M.H.J.; Bagheri, A.; Nazari, M. A LSTM-Based Deep Learning Method with Application to Voltage Dip Classification. In Proceedings of the 2018 18th International Conference on Harmonics and Quality of Power (ICHQP), Ljubljana, Slovenia, 13–16 May 2018; pp. 1–5.
47. Bagheri, A.; Gu, I.Y.H.; Bollen, M.H.J.; Balouji, E. A Robust Transform-Domain Deep Convolutional Network for Voltage Dip Classification. *IEEE Trans. Power Deliv.* **2018**, *33*, 2794–2802. [CrossRef]
48. Wang, H.; Qi, L.; Ma, Y.; Jia, J.; Zheng, Z. Method of Voltage Sag Causes Based on Bidirectional LSTM and Attention Mechanism. *J. Electr. Eng. Technol.* **2020**, *15*, 1115–1125. [CrossRef]
49. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization 2017. *arXiv* **2014**, arXiv:1412.6980.
50. Press, G. Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says. *Forbes* **2016**. Available online: https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/?sh=7b5865716f63 (accessed on 30 January 2022).