applied sciences

MDPI

Article

A Comparison of Power Quality Disturbance Detection and Classification Methods Using CNN, LSTM and CNN-LSTM

Carlos Iturrino Garcia [1,2,*,†], Francesco Grasso [1,2,†], Antonio Luchetta [1,2,*,†], Maria Cristina Piccirilli [1,2,†], Libero Paolucci [1,2,*,†] and Giacomo Talluri [1,2,†]

1 Smart Energy Lab, University of Florence, 50139 Florence, Italy; francesco.grasso@unifi.it (F.G.); mariacristina.piccirilli@unifi.it (M.C.P.); giacomo.talluri@unifi.it (G.T.)
2 Department of Information Engineering, University of Florence, 50139 Florence, Italy
* Correspondence: carlos.iturrinogarcia@unifi.it (C.I.G.); antonio.luchetta@unifi.it (A.L.); libero.paolucci@unifi.it (L.P.)
† These authors contributed equally to this work.

Received: 30 July 2020; Accepted: 21 September 2020; Published: 27 September 2020

Abstract: The use of electronic loads has improved many aspects of everyday life, permitting more efficient, precise and automated process. As a drawback, the nonlinear behavior of these systems entails the injection of electrical disturbances on the power grid that can cause distortion of voltage and current. In order to adopt countermeasures, it is important to detect and classify these disturbances. To do this, several Machine Learning Algorithms are currently exploited. Among them, for the present work, the Long Short Term Memory (LSTM), the Convolutional Neural Networks (CNN), the Convolutional Neural Networks Long Short Term Memory (CNN-LSTM) and the CNN-LSTM with adjusted hyperparameters are compared. As a preliminary stage of the research, the voltage and current time signals are simulated using MATLAB Simulink. Thanks to the simulation results, it is possible to acquire a current and voltage dataset with which the identification algorithms are trained, validated and tested. These datasets include simulations of several disturbances such as Sag, Swell, Harmonics, Transient, Notch and Interruption. Data Augmentation techniques are used in order to increase the variability of the training and validation dataset in order to obtain a generalized result. After that, the networks are fed with an experimental dataset of voltage and current field measurements containing the disturbances mentioned above. The networks have been compared, resulting in a 79.14% correct classification rate with the LSTM network versus a 84.58% for the CNN, 84.76% for the CNN-LSTM and a 83.66% for the CNN-LSTM with adjusted hyperparameters. All of these networks are tested using real measurements.

Keywords: power quality disturbances; long short term memory; convolutional neural network; short time Fourier transform

1. Introduction

The wide diffusion of electronic loads in the industrial, household, commercial and public sectors has improved many aspects of everyday life. In other words, power electronics technologies have made life easier and more comfortable. On the other hand electronic devices have a nonlinear behavior that disturbs the power grid through voltage and current waveform distortions. The number of power electronics devices that are connected to the grid is constantly increasing; as a consequence, the waveform distortion levels have also increased in the last decades causing a degradation of the Power Quality (PQ) levels on the grid. Ideally, grid voltages and currents should have a purely sinusoidal behavior. If distorting components are injected, power losses can occur

as well as malfunctioning of electric devices. This can severely damage industrial equipment, household appliances and commercial business plants. They also cause disturbance to other consumers and interference in nearby communication networks [1]. Besides that, the energy providers can sanction the injection of such disturbances on the grid. These disturbances concern frequency, amplitude, waveform and—in three-phase systems—symmetry of voltages and currents. Moreover, high energy demanding companies are becoming more sensitive to loss of profit margins due to power losses and plant shoutdowns caused by low PQ levels [2]. In order to adopt countermeasures and to define the origin of the phenomena, it is important to detect and classify these disturbances. This information indeed can be used to define the PQ level of a grid, to understand their behavior and to assess the responsibilities. The operation can be carried on by acquiring and processing the voltage and current signals of a power line. To do this, several techniques are currently exploited using machine learning algorithms [3]. Among the extensive inventories of deep learning algorithms, for the present work, the Long Short Term Memory (LSTM) and the Convolutional Neural Networks (CNN) are being used to detect and classify these disturbances [4–6]. Other algorithms are being used to address these problems like the Kalman Filter, Wavelet Transform and the Support Vector Machine (SVM).

In [7], a Kalman Filter is used in an UPQC to extract the state components of the distorted supply voltage and load current. The algorithm can classify PQD internally enabling the conditioning of the PQ signals for power factor correction. The technique seems to work well with the detection of sag, swell and harmonic distortion, however it shows a certain lag between the disturbance starting condition and the detection [8]; furthermore, the algorithm is usually applied to a restricted number of disturbances. On the other hand, the wavelet transform is used as a tool for analyzing PQD as shown in [9]. The tool is very useful for the extraction of the signals features for learning algorithms like the SVM as shown in [10–12]. However, it does not perform disturbances detection by itself. The SVM showed interesting performances for the detection of a wide range of PQ disturbances and it is often used as a benchmark to assess the performances of other algorithms. The main disadvantage of the PQD detection techniques mentioned above is that, once the voltage and current waveforms are acquired, a preprocessing of the signal must be performed before feeding it to the algorithm. This usually consists of a signal features extraction. Deep learning algorithms solves this problem by implicitly applying a feature extraction for the classification of the signal. In other words, these algorithms could be fed with raw data and still make accurate classifications. This can help to speed up the identification and classification process especially in real time applications.

For the training and the validation of deep learning algorithms, it is necessary an extensive dataset in order to avoid overfitting and obtain generalization. Unfortunately, it is not easy to obtain such datasets with experimental data. One reason is that performing on-field data sampling through measurement campaigns is time consuming, many of these disturbances indeed are sporadic, and it is not always possible to record an event with a desired amplitude and duration. For that reason, simulated voltage disturbances are used in order to create the dataset for training and validation. For further generalizing the dataset, data augmentation is used, since it has proven to be efficient in improving accuracy by reducing overfitting [13,14].

This work explores different deep learning architectures which were trained and validated using simulated data and tested using experimental data. Once the simulated data are generated, it is then augmented, in order to obtain a generalized result and overcome any sampling discrepancy and phase difference between simulated data and measured data. The signals are pre-processed in order to compare the accuracy of each architecture in their proven classification tasks. With respect to other works in which the training, validation and testing steps are performed using purely simulated data or purely experimental data [4,15,16], in the present work the training and validation steps are performed with simulated datasets, while the testing one is performed with experimental datasets that were acquired on the field.

2. Power Quality Disturbance Simulations and Dataset Acquisition

The definition of Power Quality by the IEEE is: Power Quality is the concept of powering and grounding sensitive equipment in a matter that is suitable to the operation of that equipment [17]. The definition given by the International Electrotechnical Commission is: The characteristics of the electricity at a given point on an electrical system, evaluated against a set of reference technical parameters. These parameters might, in some cases, relate to the compatibility between electricity supplied on a network and the loads connected to that network. From these definitions it can be said that PQ always includes voltage quality and supply reliability. The Power Quality Disturbances (PQD) are broadly classified into three categories: magnitude variations, sudden transients and steady-state harmonics, as said in [18].

As a preliminary stage of this work, the voltage and current time signals were simulated using MATLAB Simulink. By doing this, it has been possible to recreate the disturbances on the line and see how they interact with the other devices connected to the grid. Thanks to the simulation results, it has been possible to acquire a current and voltage dataset with which the identification algorithms were trained and validated. This dataset includes simulations of several disturbances such as Sag, Swell, Harmonics, Transient, Notch and Interruption. After that, the deep learning algorithms were tested with an experimental dataset of voltage and current field measurements containing the disturbances mentioned above and it has been possible to perform a performance comparison.

For the generation of the PQD dataset a MATLAB/Simulink model of a micro grid has been implemented. The model is shown in Figure A1 and it includes several different industrial loads. It is possible to identify a three-phase dynamic load which could be associated to an electrical motor with variable load, a linear load and a nonlinear load which injects disturbances on the net [18]. These disturbances include: Sag, Swell, Harmonics, Transient, Notch and Interruption.

The Simulink schematic for the PQD simulation consist of a three-phase voltage source connected to a fault block, then to a line impedance, to a transformer and finally ending in a linear load. The voltages and currents are measured after the transformer which are going to be used for the classification. Inside the fault block, there are different types of disturbance generation blocks which were listed above. The sag block reduces the voltage from its nominal value. The swell block rises the voltage from its nominal value, and it is modeled with a switch that connects the grid to a capacitor bank. The harmonic distortion is modeled as a resistor in parallel with a capacitor in parallel with a free willing diode. The transient is modeled with an impulse generator. The notch block is modeled by a thrysistor in parallel with a resistance and an inductance. Finally the interruption block is simply a switch. The simulink schematic for the simulation of the PQD along with each disturbance block are shown in Appendix A.

3. Machine Learning Algorithms

There is a wide literature concerning the feature extraction and classification of PQD exploiting different types of Machine Learning architectures. In [19], Borges implements a feature extraction using statistical data of PQD for classification. Shen in [20], uses an Improved Principal Component Analysis (IPCA) to extract the statistical features of the PQD followed by a 1-dimensional Convolutional Neural Network (1-D-CNN) to extract other features of the signal and for the classification. The results of the classification in this work were compared to a Support Vector Machine (SVM) In terms of accuracy, which is the ratio between correct classification and total classifications, the SVM gave 98.55% accuracy while the 1-D-CNN scored 99.75%. These results proved that the 1-D-CNN is slightly superior to the SVM in classifying PQD.

In [4], Mohan explores the potential of deep learning architectures for PQ disturbances classification. In this work the convolutional neural network (CNN), recurrent neural network (RNN), identity-recurrent neural network (I-RNN), long short-term memory (LSTM), gated recurrent units (GRU) and convolutional neural network-long short-term memory (CNNLSTM) are studied and compared in order to find the best architecture for PQD data. The accuracy of each deep learning

architectures is: 98% for the CNN, 91.5% for RNN, 93.6% for the I-RNN, 96.7% for the LSTM, 96.4% for the GRU and 98.4% for the hybrid CNN-LSTM. These results proved that the hybrid CNN-LSTM is superior at classifying PQD.

The hallmark of deep learning architectures is that they are able to perform a feature extraction and classification by processing raw data. However, many other works proved that these techniques are also successful for the signal feature extraction for different applications including PQD [21–23].

3.1. Long Short Term Memory

A recurrent neural network (RNN) is a neural network that simulates a discrete-time dynamical system that has an input x_t, an output y_t and a hidden state h_t as defined in [24]. A drawback of the RNNs is that they suffer from vanishing or exploding gradient. By truncating the gradient where it does not harm, LSTM can learn to bridge minimal time lags in excess of 1000 discrete-time steps by enforcing constant error flow through constant error carousels within special units [25].

The LSTM has three states that help the network to reduce the long term dependency of the data. These states are called the Forget State, the Input State and the Output State. The Forget State eliminates redundant or useless data. The Input State processes the new data and finally the Output State processes the input data with the cell state. A block diagram of a LSTM cell is shown in Figure 1. In the following subsections a focus on the three steps is presented.

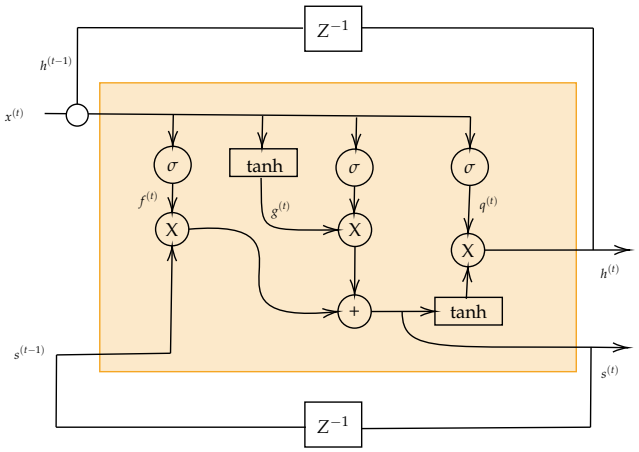


Figure 1. Block diagram of one cell of a long short term memory architecture.

3.1.1. Forget State

The forget state controls the state parameter $s^{(t)}$ via a sigmoid function σ. This state controls what the cell should remember through time and what to forget. The equation of the forget state is shown in Equation (1), where $f^{(t)}$ is the forget vector, x^t and $h^{(t-1)}$ are the input and previous output respectively. The input and the previous output are multiplied by the trained weights U and W with bias b. This result is then truncated between 0 and 1 via a sigmoid function. Basically the idea is to have an input vector added with the previous output vector passed through a neural network which outputs the values to keep with a 1 and the values to forget with a 0.

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^t + \sum_j W_{i,j}^f h_j^{(t-1)}\right) \tag{1}$$

3.1.2. Input State

The new state of the cell is defined in the input state where the previous state is multiplied by the forget state dropping off irrelevant information. This is shown in $f_i^{(t)} s_i^{(t-1)}$ of Equation (2). Now the relevant information gets updated in $g_i^{(t)} \sigma\left(b_i + \sum_j U_{i,j} x_j^t + \sum_j W_{i,j} h_j^{(t-1)}\right)$ which is the product between

the input and the previous neural network output times $g^{(t)}$ which is the candidate for the next time step of the cell state. The equation that generates the vector that contains the candidates for the next cell state is shown in Equation (3).

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\left(b_i + \sum_j U_{i,j} x_j^t + \sum_j W_{i,j} h_j^{(t-1)}\right) \tag{2}$$

$$g_i^{(t)} = \tanh\left(b_i^g + \sum_j U_{i,j}^g x_j^t + \sum_j W_{i,j}^g h_j^{(t-1)}\right) \tag{3}$$

3.1.3. Output State

The output state decides what should be the output of the cell and of the new cell state. The output of the cell is shown in Equation (4) where the cell state goes through a hyperbolic tangent and it is then multiplied by the output of another hidden layer as shown in Equation (5).

$$h_i^t = \tanh\left(s_i^{(t)}\right) q_i^{(t)} \tag{4}$$

$$q_i^{(t)} = \sigma\left(b_i^o + \sum_j U_{i,j}^o x_j^t + \sum_j W_{i,j}^o h_j^{(t-1)}\right) \tag{5}$$

3.2. Convolutional Neural Networks (CNN)

Convolutional neural networks or CNN, are a particular type of neural network for data processing that has a grid-like topology. Convolutional networks proved to be successful in several practical applications. They essentially consist of neural networks that use convolution in place of general matrix multiplication in at least one of their layers [26]. The convolutional layer is accompanied by a pooling layer which is a type of under sampling that helps with processing speed.

Since the signals of interest are 1-D signals and the CNN processes a 2-D signal, a pre-processing of each signals is necessary. Hence, the Short Time Fourier Transform is performed on each signal before feeding it to the CNN; by doing this, an image containing the spectral components and amplitude of the signal of interest is generated. The characteristics of this processing technique are highlighted in Section 4.1.

3.2.1. Convolutional Layer

Convolution leverages three important ideas that can help improve a machine learning system: sparse interactions, parameter sharing and equivariant representations. Moreover, convolution provides a means for working with inputs of variable size [26]. The convolution layer of a convolutional neural network operates by applying a convolution to each dataset. Since the hallmark of the CNN is image classification, the 2 dimensional version of the discrete convolution is used. To serve as a reminder, the 2 dimensional discrete convolution operation is shown in Equation (6). The convolutional layer works by adjusting the parameter ω for each backpropagation in order to maximize the features extracted by minimizing the error in classification. This in turn creates a set of filters which are the key of the feature extraction of the data.

$$S[n_1, n_2] = \sum_{m=1}^{M_1} \sum_{m=1}^{M_2} x[m_1, m_2] \omega[n_1 - m_1, n_2 - m_2] \tag{6}$$

3.2.2. Pooling Layer

A pooling function replaces the output of the layer with a summary statistic of the previous layer outputs [26]. The most popular pooling functions include the max of a rectangular neighborhood, the average, the L2 norm, or a weighted average based on the distance from the central datum.

This layer in the architecture speeds up the training and classification since it undersamples the dataset and helps the network to obtain a more generalized result. An illustration of the pooling function is shown in Figure 2 where $\mathcal{X}_n$ is the vector containing the pooled data of the dataset as shown in Equation (7). In Equation (8) the pooling function is shown.

$$\mathcal{X}_n = \{x_j, ..., x_N\} \tag{7}$$

$$\hat{x}_n = f(x_n, x_{n+1}, x_{n+2}) = f(\mathcal{X}_n) \tag{8}$$

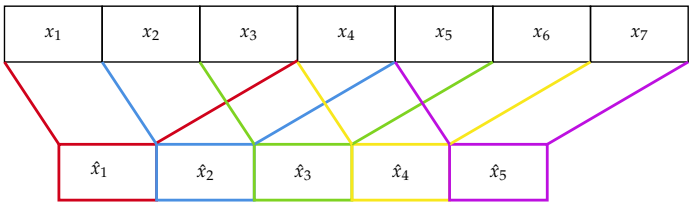


Figure 2. Pooling illustration for the CNN.

The pooling layer can contain either the maximum of the set, the average, the $\mathbf{L}_2$ norm or the weighted average of the pool. In this work the max pooling is used as shown in Equation (9).

$$f(\mathcal{X}_n) = argmax(\mathcal{X}_n) \tag{9}$$

4. Methodology

In this work, the end goal is to compare different Machine learning architectures in order to determine which is the best suited for the Power Quality disturbances identification task. The LSTM, the CNN, CNN-LSTM and the CNN-LSTM with adjusted hyperparameters were chosen for the comparison based on their performances on previous works [4]. The LSTM was designed specifically for time series data which is well suited for this task. The CNN has proven to be very effective in image classification tasks, hence it is necessary to treat the signal as an image. Since the success of CNN is attributed to its superior multi-scale high-level image representations [27], a time frequency analysis is executed to the time series data as shown in [28] so the signal can be treated as an image. To obtain this image, the Short-Time Fourier Transform is used. It has been selected because of its simplicity of implementation but others, like the wavelet transform, can be considered as well. Data Augmentation is executed in order to obtain a more generalized result for the training and validation. The different techniques are trained with simulated data and then tested with both simulated and experimental datasets. The datasets consist of three phase voltage signals which have one sinusoidal waveform shifted 120° with respect to the other; both in the experimental and in the simulated datasets, PQ disturbances are superposed to the three signals. The identification algorithms are fed with 1000 samples of the time series per classification. A rectangular sliding window is used with an overlap of 500 samples.

4.1. Short Time Fourier Transform (STFT)

As said in [29], one of the approaches for using Machine Learning techniques in the frequency domain is to transform the time series into the frequency domain first, and then apply conventional neural network components. As specified before, the transformation chosen for this task was the Short Time Fourier Transform. It is characterized by a Fourier transform executed in a fixed windowed interval. The window function $g(n)$ is called Blackman's window function and it is used to multiply a short segment of the signal by the window function. This avoids sharp sections and redundant information. A Fourier transform of this small windowed section $X_n(j\omega)$ is calculated and stacked up to form a matrix. The STFT equation is shown in Equation (10). An illustration of the Blackman's window and of the algorithm can be shown in Figures 3 and 4, respectively. The resulting matrix

can now be treated as an image to train a Convolutional Neural Network which is capable of feature extraction of the frequency components of the signal. The Fourier Transform equation is shown in Equation (10) and the Blackman Window Function is shown in Equation (11).

$$X(j\omega) = \sum x(n)g(n - mR)e^{-2j\pi fn} \tag{10}$$

$$w[n] = a_0 + a_1 cos\left(\frac{2\pi n}{N}\right) + a_2 cos\left(\frac{4\pi n}{N}\right) \tag{11}$$

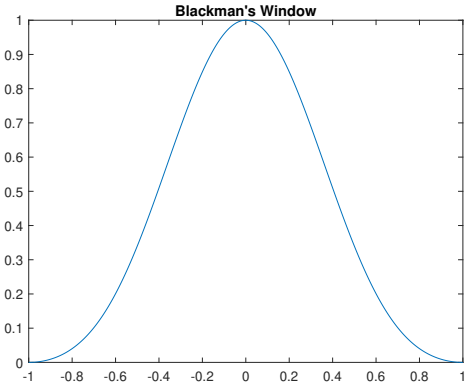


Figure 3. Blackman's window.

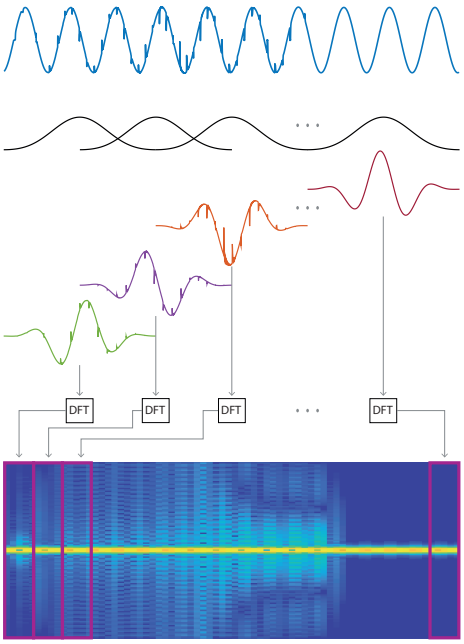


Figure 4. Short time Fourier transform illustration.

4.2. Data Augmentation

Although training and validation of the network using simulated data can be easily implemented and it is time efficient, it can often result in overfitting. To solve this issue and in order to obtain a more generalized result, data augmentation is necessary. Data augmentation consists of manipulating the training and validation set to obtain more data with small variations. The PQD waveforms obtained through simulations are limited by sampling time and starting time. In a simulated voltage signal the disturbance which is superposed to the signal is characterized by a fixed time interval. Through data augmentation, the PQD can be shifted of 0°, 60°, 120°, 180°, 240°, and 300°. These signals are then

oversampled by 2 and 4 to deal with possible sampling discrepancies. In other words, for each of the three voltage waveforms, after data augmentations, six more signals are generated returning a total of 18 new waveforms.

$$\mathbf{Voltage}_{3\Phi}^{T} = \begin{bmatrix} \mathbf{V}_{\Phi_1}^{T} \\ \mathbf{V}_{\Phi_2}^{T} \\ \mathbf{V}_{\Phi_3}^{T} \end{bmatrix} \quad \mathbf{Voltage}_{3\Phi}^{0.5T} = \begin{bmatrix} \mathbf{V}_{\Phi_1}^{0.5T} \\ \mathbf{V}_{\Phi_2}^{0.5T} \\ \mathbf{V}_{\Phi_3}^{0.5T} \end{bmatrix} \quad \mathbf{Voltage}_{3\Phi}^{0.25T} = \begin{bmatrix} \mathbf{V}_{\Phi_1}^{0.25T} \\ \mathbf{V}_{\Phi_2}^{0.25T} \\ \mathbf{V}_{\Phi_3}^{0.25T} \end{bmatrix} \tag{12}$$

$$\mathbf{Class}_d = \begin{bmatrix} \mathbf{Voltage}_{3\Phi}^{T} \\ \mathbf{Voltage}_{3\Phi}^{0.5T} \\ \mathbf{Voltage}_{3\Phi}^{0.25T} \\ -\mathbf{Voltage}_{3\Phi}^{T} \\ -\mathbf{Voltage}_{3\Phi}^{0.5T} \\ -\mathbf{Voltage}_{3\Phi}^{0.25T} \end{bmatrix} \tag{13}$$

4.3. Implementation of the Simulink Schematic for the Generation of the Simulated Dataset

For the generation of the simulated dataset, a Simulink model of the grid has been used [18]. With this model it has been possible to generate several PQ disturbances and organize it in a cell array. The disturbances that were implemented in the simulink model are the sag, the voltage rise, the harmonic distortion, the transient, the notch and the interruption [30]. After the Simulink simulation is completed, a dataset is generated. Hence the data are gathered with a script which generates a structured cell array. Once the process is completed, each fault is labeled with a target number as shown in Equation (14). To train the neural network, the data for the Class and Target are shuffled together in order to obtain a generalized solution for the network.

$$Class = \begin{bmatrix} \textbf{Normal} \\ \textbf{Sag} \\ \textbf{Swell} \\ \textbf{Harmonics} \\ \textbf{Transient} \\ \textbf{Notch} \\ \textbf{Interruption} \end{bmatrix} \quad Target = \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{4} \\ \mathbf{5} \\ \mathbf{6} \end{bmatrix} \tag{14}$$

After the time series signals are stored in a structured array they are shuffled. The next step is to train the network, validate and test it. For this task, the dataset containing the time series signals is separated in two groups: one containing 75% of the signals, used for the training and the other group containing the remaining 25%, used for the validation. These networks are trained with a batch size of 20 data points with a maximum of 30 epoch making a total of 15 iterations per epoch. After training is completed for all architectures, the performance of the networks are evaluated using a confusion matrix. With this, the precision and recall are calculated for each class.

4.4. Deep Learning Architectures

Deep learning in neural networks is the approach of composing networks into multiple layers of processing with the aim of learning multiple levels of abstraction [26]. By doing so, the network can adaptively learn low-level features from raw data and higher-level features from low-level ones in a hierarchical manner, nullifying the over-dependence of shallow networks on feature engineering [31]. The two most popular types of networks are the feed forward networks and the recurrent networks. Both have evolved in what is known today as deep learning architectures. Concerning the recurrent networks the LSTM is considered, it is mostly used for time series data; with regards to feed forward networks the CNN is considered, it is used mostly for image classification. Both are found to be successful in classification problems.

4.4.1. Long-Short Term Memory

Concerning the architecture under evaluation, 100 hidden units were used, that is, 100 LSTM blocks were used for the classification of time series data. Since in this architecture it is not possible to use a pooling layer, a drop out layer is used in order to achieve generalization. After that, a fully connected layer, a soft max layer and a classification layer, which outputs the final result, are added to the architecture. The architecture is shown in Figure 5.

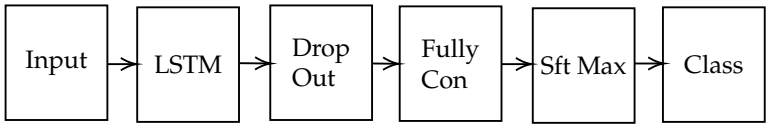


Figure 5. LSTM architecture block diagram.

4.4.2. Convolutional Neural Networks

The architecture used in this experiment has 3 stages of convolution, that is, 3 convolutional layers with 3 batch normalizations, 3 rectifier linear units or ReLU's and 2 under samplings using the max pooling layers. After the three stages of convolution, the network has a fully connected layer, a softmax layer and finally the output with the classification layer. Concerning the input, as specified before, it is necessary to first preprocess the training data using the STFT. The architecture can be seen in Figure 6 with all the details of the hyperparameters. Examples of each disturbance STFT are shown in Figure A2a–l.

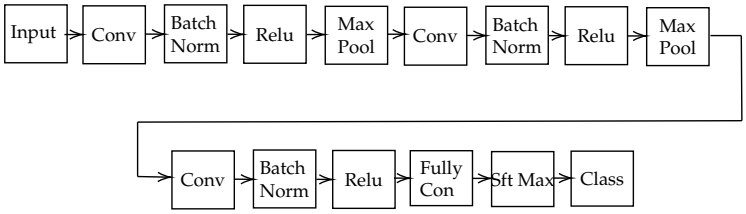


Figure 6. CNN architecture block diagram.

4.4.3. Convolutional Neural Networks—Long-Short Term Memory

This architecture mixes the CNN with the LSTM. In order to do this a sequence folding layer right after the input layer is added. The sequence folding layer converts a batch of data sequences to a batch of data. After this layer, the CNN comes into play. After the CNN, there is a sequence unfolding layer used to convert the batch of data in a batch of sequenced data. The sequence data are the input to the LSTM. Before the LSTM layer, there is a flattening layer that reshapes the input data to the input of the LSTM layer. Then a fully connected layer, a soft max and finally a classification layer are added respectively. The block diagram of the architecture is shown in Figure 7. By adjusting the parameters of the above mentioned architecture the CNN-LSTM with adjusted hyperparameters is obtained.

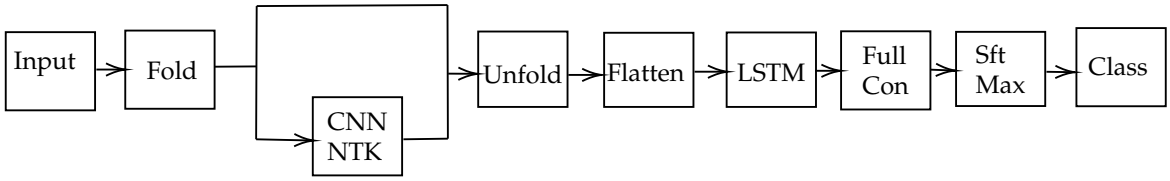


Figure 7. LSTM-CNN architecture block diagram.

5. Experimental Setup

To provide an experimental testbed for the validation of the simulation results, it has been necessary to test the identification techniques with experimental datasets. It is possible to extract this data by acquiring voltages and currents directly from the grid; however, to operate a comprehensive testing, it is necessary to reproduce different fault types by varying the amplitude, duration and intensity of the phenomena. Hence an experimental setup has been designed and implemented with which it is possible to reproduce and record several real-time PQDs. The system has been designed by the Smart Energy Lab of the University of Florence [18] and it is shown in Figure 8. To simulate the grid with PQD, a Chroma 61,500/61,600 series programmable AC source is used. To perform the measurement, a Fluke 435 Series II Power Quality Energy Analyzer (Leasametric, Villebon-sur-Yvette, France) and two Yokogawa PX8000-F-HE/G5/M1 (Yokogawa, Musashino, Japan). The first instrument was used to monitor the loading of the loads and check for any malfunctions, while with the Yokogawa instruments it was possible to obtain measures of the electrical quantities of the system recording up to 100,000 samples per second (100 kS/s) and averaging the measurement over a set period (minimum 10 ms). The load connected is given by a 1 kW linear load, a group of switching power supplies for a total of 2 kW and a three-phase inverter of 2 kW. The signal generated by the Chroma 61,500/61,600 is acquired by the Yokogawa PX8000 via the PowerViewerPlus software and is sent to the remote PC for data storage. By doing this, it is possible to test and compare the performances of different algorithms and identify strengths and weaknesses while assuring repeatability of the experiments.

Figure 8. Experimental setup for power quality disturbances generation.

6. Testing Results

6.1. Testing of the Detection Techniques Using Simulated Signals

Simulations of the six different PQDs were made using the MATLAB Simulink model shown in Figure A1. In total, 300 simulations of each PQD were generated to create the dataset, each one containing different start times and duration of the disturbance. This signals were stacked together in a cell array along with its corresponding labels. Data augmentation was applied as explained in Section 4.2. The cell array was shuffled and divided into 75% for training and 25% for testing. This architecture had a training time of 671 min or 11.18 h with an accuracy of 79.14% for the validation.

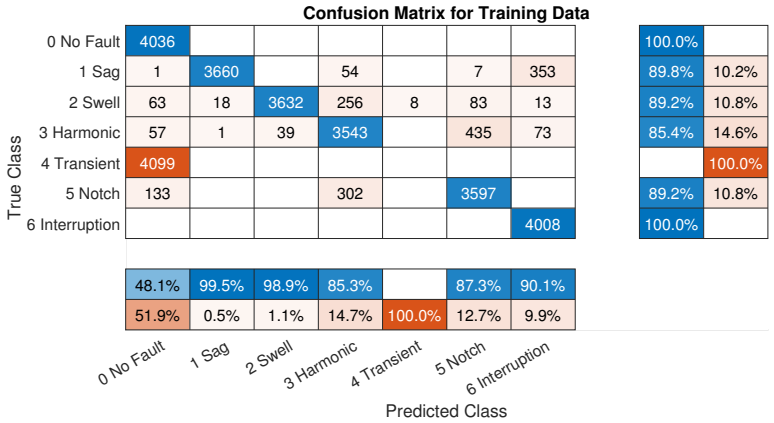
The confusion matrix of the LSTM for training and validation is shown in Figure 9a,b along with the testing in Figure 9c. This architecture had a training time of 443 min or 7.38 h with an accuracy of 84.58% for the validation. The confusion matrix of the CNN for training and validation is shown in Figure 10a,b along with the testing in Figure 10c. This architecture had a training time of 751 min or 12.52 h with an accuracy of 83.66% for the validation. The confusion matrix of the CNN-LSTM for training and validation is shown in Figure 11a,b along with the testing in Figure 11c. This architecture had a training time of 51 min or 0.85 h with an accuracy of 84.78% for the validation. The confusion matrix of the CNN-LSTM for training and validation is shown in Figure 12a,b along with the testing shown in Figure 12c.

On each of the confusion matrix of the compared architectures, the precision and the recall was calculated. The precision of a classifier is defined as the number of retrieved relevant items as a proportion of the number of retrieved items for any given class [32]. In other words, it is the ratio between the positive identifications that are actually correct and the entire set of positive identifications of any given class. Recall, on the other hand, is defined as the number of retrieved relevant items as a proportion of all the relevant items, for any given retrieved set [32]. In other words, the proportion of the actual positives that where identified correctly. The comparison of precision and recall of different architectures are shown in Figure 13. The precision and recall of the LSTM-CNN with adjusted hyperparameters had superior results with respect to the other architectures due to the fact that it had better scores for classifying the transient in both training and testing.

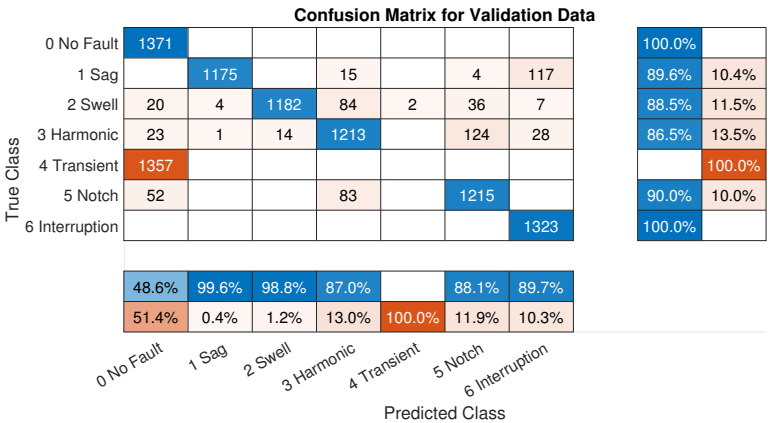
The LSTM training gave an accuracy of 79.14% where most of the problems were found in the Transient disturbance as shown in the precision and recall plots. The architecture was not able to detect the transient disturbance either in the training or the validation signals, that is, the LSTM classified the Transient signal as a No Fault in 100% of the cases. Concerning the other classes it resulted in 10.2%–14.6% misclassification.

The CNN needs to be fed with an image which, in this case, represents the spectral components and amplitude of the signal of interest; hence a STFT was done on the augmented dataset. The training of the CNN gave an accuracy of 84.58% which is an improvement with respect to the LSTM. The problem with this architecture is that it classified 89% of the No Fault as a Transient from the training dataset and an 88.6% from the validation. Again, it showed confusion between the two classes as shown in the precision and recall plots.

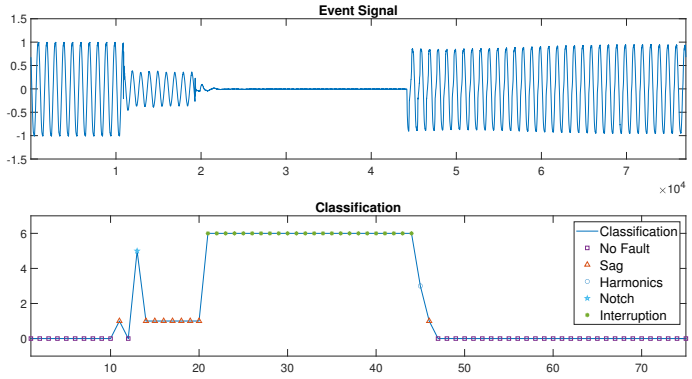
As regards to the the hybrid CNN-LSTM, two similar strategies were tested. The first strategy consisted in joining the LSTM and CNN that were used in the previous experiment and the second was using the combined architectures while adjusting the hyperparameters. The first strategy, exploiting the CNN-LSTM, resulted in an improvement of the classification performances of almost all of the the disturbances except for the Transient which resulted in 100% misclassification as with the LSTM. The other disturbances misclassifications ranged between 2%–10%, which was an improvement. Concerning the second strategy exploiting the hybrid architecture, a significant improvement on the Transient response recognition was reached resulting in a 51.1% misclassification with the No Fault condition. In both, precision and recall, it showed more or less a 50% chance of miss-classification. For the other disturbances the misclassification ranges between 1.3% and 4.8% which is also an improvement. The hybrid CNN-LSTM with adjusted hyperparameters was able to detect the Transient disturbance in 48.9% of the signals where the transient was present. The other architectures failed completely in this task. Furthermore, this architecture obtained better results on the other disturbance classifications.



(**a**) Training LSTM

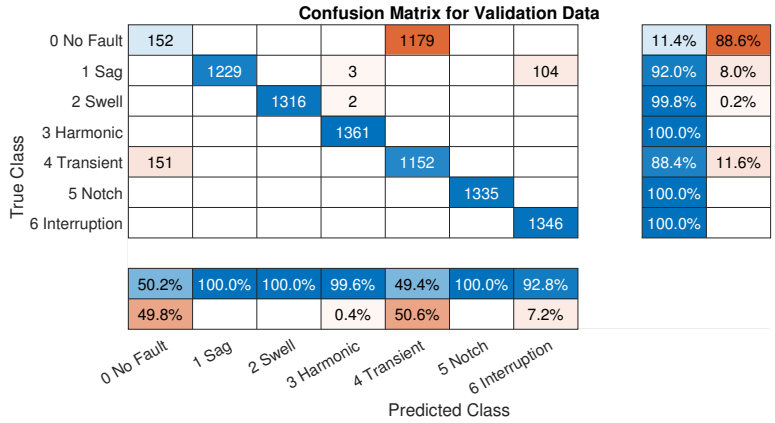


(**b**) Validation LSTM

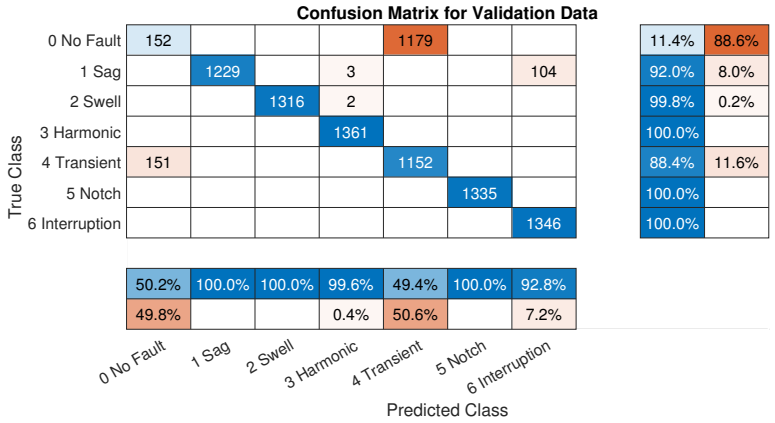


(**c**) Testing LSTM

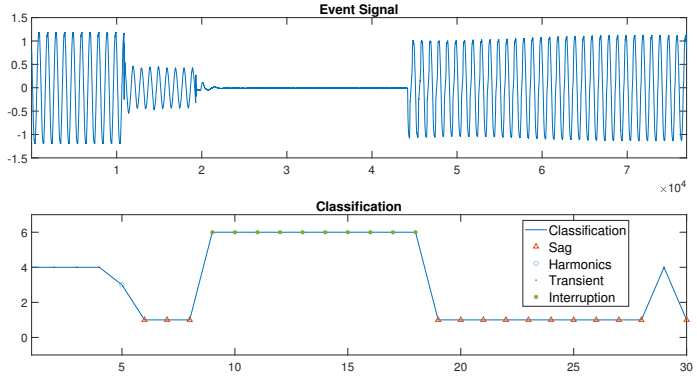
Figure 9. (**a**) Confusion matrix of the LSTM training, (**b**) Confusion matrix of the LSTM validation, (**c**) Testing of the LSTM using a simulated voltage waveform.



(**a**) Training CNN

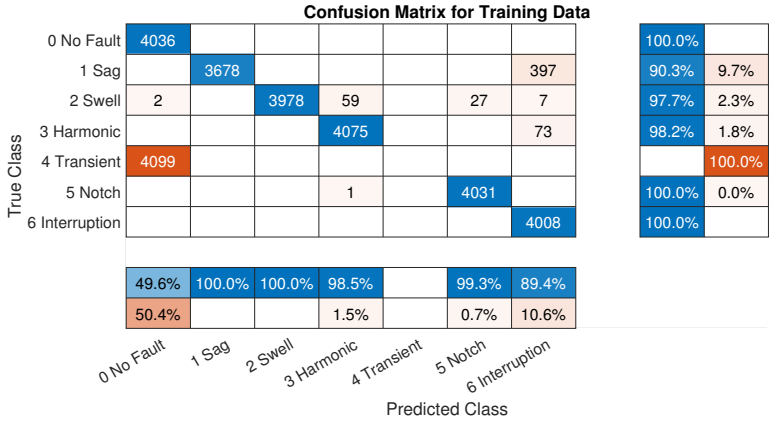


(**b**) Validation CNN

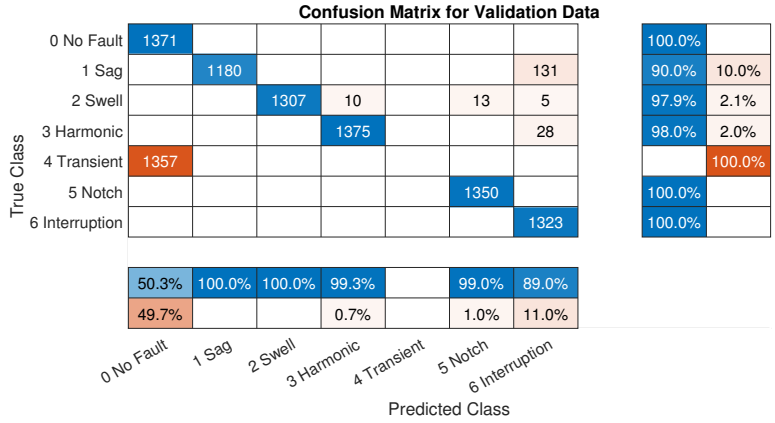


(**c**) Testing CNN

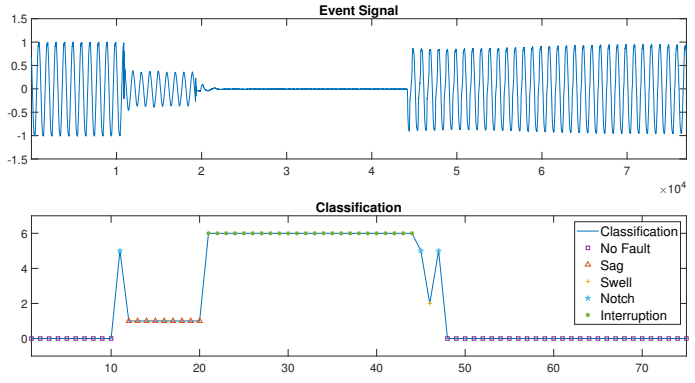
Figure 10. (**a**) Confusion matrix of the CNN training, (**b**) Confusion matrix of the CNN validation, (**c**) Testing of the CNN using a simulated voltage waveform.



(**a**) Training CNN-LSTM

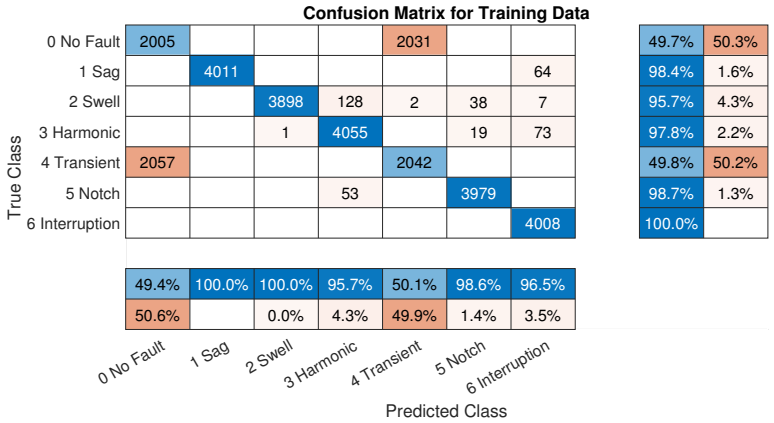


(**b**) Validation CNN-LSTM

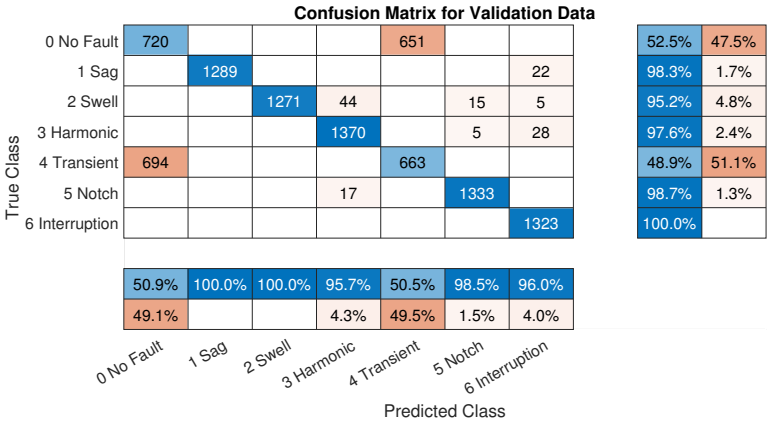


(**c**) Testing CNN-LSTM

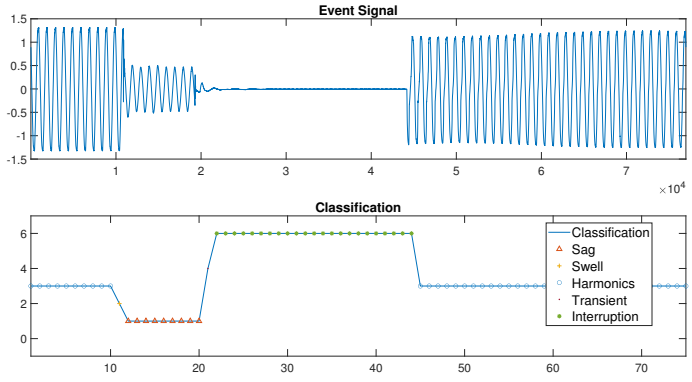
Figure 11. (**a**) Confusion matrix of the CNN-LSTM training, (**b**) Confusion matrix of the CNN-LSTM validation, (**c**) Testing of the CNN-LSTM using a simulated voltage waveform.



(**a**) Training adjusted CNN-LSTM

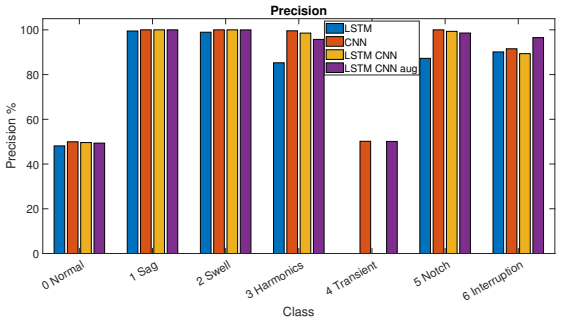


(**b**) Validation adjusted CNN-LSTM

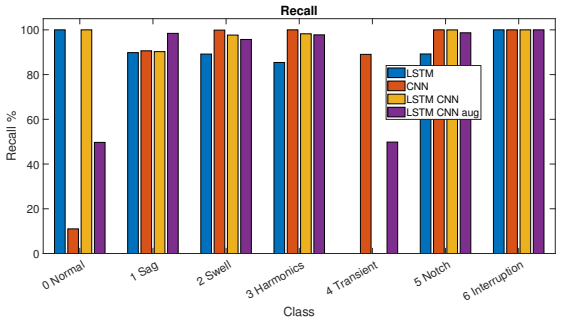


(**c**) Testing adjusted CNN-LSTM

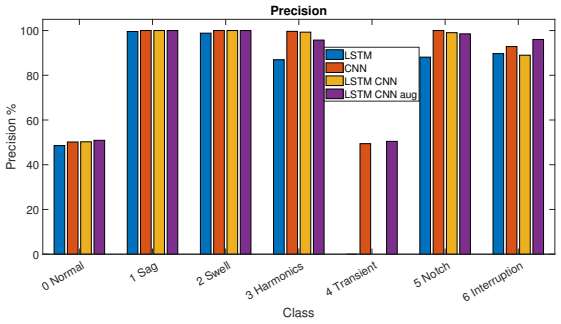
Figure 12. (**a**) Confusion matrix of the CNN-LSTM with adjusted hyperparameters training, (**b**) Confusion matrix of the CNN-LSTM with adjusted hyperparameters validation, (**c**) Testing of the CNN-LSTM with adjusted hyperparameters using a simulated voltage waveform.



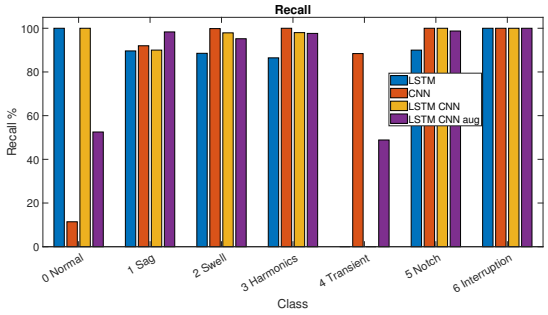
(**a**) Precision for Training Data



(**b**) Recall for Training Data



(**c**) Precision for Testing Data



(**d**) Recall for Testing Data

Figure 13. Bar chart of the comparison of the precision and recall of the LSTM (blue), CNN (red), LSTM-CNN(yellow) and LSTM-CNN with adjusted hyperparameters(purple). (**a**) Precision Training Data, (**b**) Recall Training Data, (**c**) Precision Testing Data, (**d**) Recall Testing Data.

6.2. Testing of the Detection Techniques Using Experimental Datasets

Other tests were conducted with experimental datasets using the test bench shown in Figure 8 in order to compare and prove the effectiveness of all the architectures previously mentioned. It has been possible to generate several experimental datasets of the interruption and of the sag disturbances. The experimental measurments are shown in Figures 14 and 15 along with the plots of the classification results below. Each classification point consist of 1000 samples of the measured signal. Once again, the CNN-LSTM with adjusted hyperparameters was the most consistent in classifying all the disturbances without misclassification. As mentioned before, the identification algorithms were tested with exerimental datasets containing interruption and sag disturbances. Concerning the sag disturbance, all of the four architectures performed correct identification. Some misclassifications occurred when testing the interruption disturbance with the CNN-LSTM and with the LSTM. Combining these results with the ones previously mentioned, the CNN-LSTM with adjusted hyperparameters is the is the one which performed best.

The event signal, shown in Figures 9c, 10c, 11c and 12c, is a voltage signal with a harmonic distortion, sag and an interruption. Each architecture had good results when tested using this signal. However, the LSTM classified the harmonics as a no fault and misclassified a section as a notch. The LSTM-CNN also showed the same problem. On the other hand the CNN misclassified the harmonic disturbance as a transient disturbance. While all architectures successfully classified only the Interruption and the Sag in the testing, the LSTM-CNN with adjusted hyperparameters was the one that had better results because it classified the harmonics, sag, interruption correctly without misclasification.

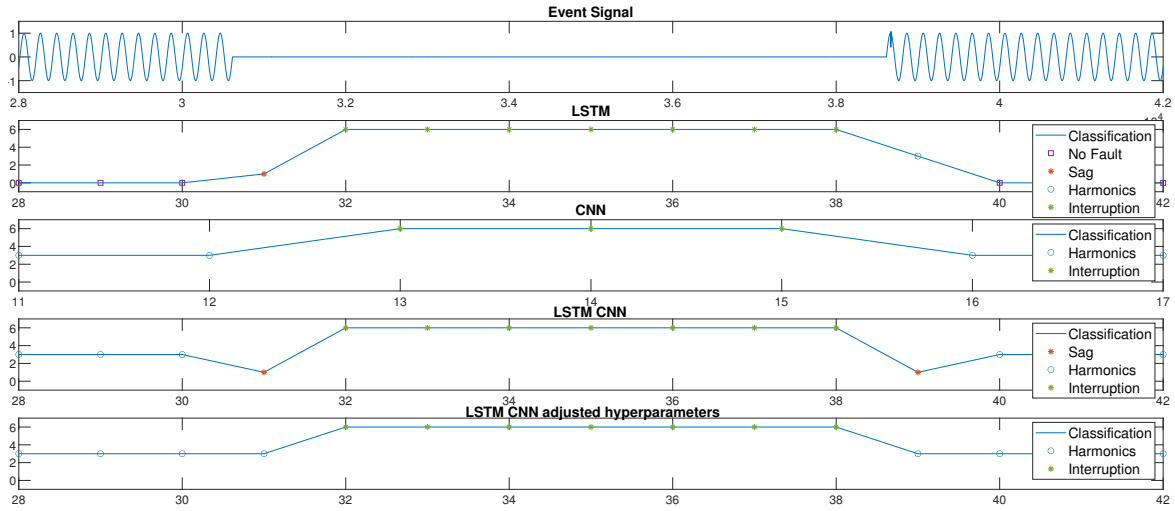


Figure 14. Voltage signal with an interruption measured on the test bench (top plot). From top to bottom, the classification performances of each architecture: LSTM, CNN, LSTM-CNN and the LSTM-CNN with adjusted hyperparameters.

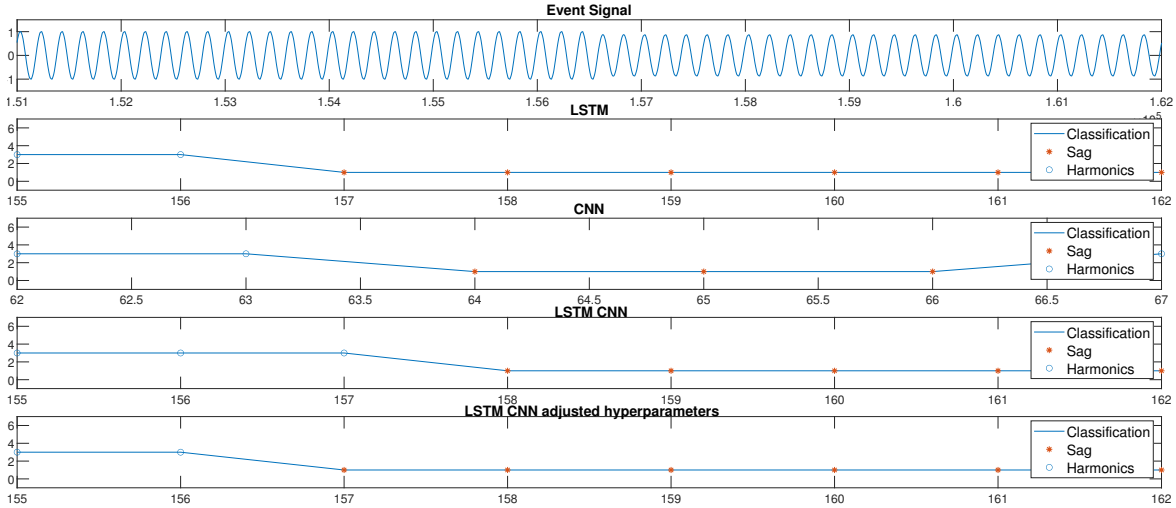


Figure 15. Voltage signal with an sag measured on the test bench (top plot). From top to bottom, the classification performances of each architecture: LSTM, CNN, LSTM-CNN and the LSTM-CNN with adjusted hyperparameters.

7. Conclusions

This work investigates the effectiveness of various deep learning architectures for Power Quality disturbances detection and classification. To do this, it is imperative to study the mechanism of these algorithms to extract the unique features of each disturbance and obtaining an efficient and accurate classification. The training and validation of deep learning architectures depend on a large number of data to better generalize the classification results. A Matlab/Simulink model has been designed and implemented in order to generate these disturbances. To improve the classification performances of the strategies under evaluation and converge to a generalized result, the data in the simulated dataset was augmented. Using the resulting datasets the authors have proposed a comparison among the LSTM, the CNN and a joint architecture that uses both the LSTM and CNN. All of the architectures were trained and validated using the augmented datasets and then tested using experimental data.

Concerning the experimental validation of the algorithms, it has been possible to generate an experimental dataset of the interruption and of the sag disturbances. The two datasets were processed by exploiting the four previously mentioned architectures. The first signal contained a train of interruptions and the second signal a train of sags. All of the four architectures successfully classified the sag signal. There were some discrepancies between the architectures while classifying the signal containing interruptions. Again, the LSTM-CNN with adjusted hyperparameters proved to be superior in classifying the disturbances.

These results show that it is possible to train deep learning architectures with simulated data and operate disturbance identification on experimental data. The transient disturbance appears to be hardly detectable for all of the architectures under evaluation, mainly due to the small duration of the disturbance. The architecture that best performed while classifying the transient disturbance was the LSTM-CNN with adjusted hyperparameters. Furthermore, concerning the classifications of other disturbances, the LSTM-CNN with adjusted hyperparameters was the most performing one, both considering the simulated and the experimental datasets.

Author Contributions: Data curation, C.I.G. and L.P.; Formal analysis, C.I.G.; Funding acquisition, F.G., L.P. and G.T.; Methodology, A.L., F.G. and G.T.; Software, C.I.G.; Supervision, A.L.; Validation, L.P. and G.T.; Writing—original draft, C.I.G. and M.C.P.; Writing—review and editing, M.C.P. All authors have read and agreed to the published version of the manuscript.

Funding: E-CUBE(E^3) Project, Energy Exchange and Efficiency, project funded by POR FESR 2014-2020 ASSE 1 AZIONE 1.1.5.

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

Appendix A

Appendix A.1. Simulink Schematic

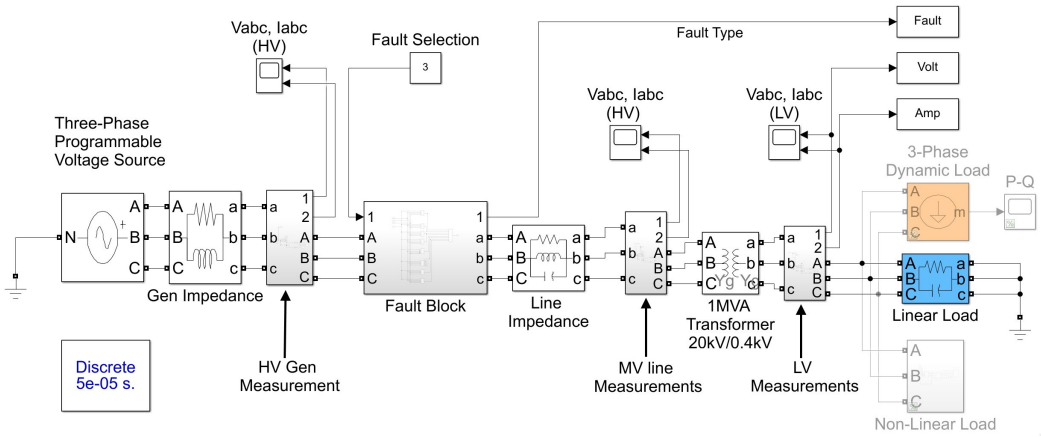
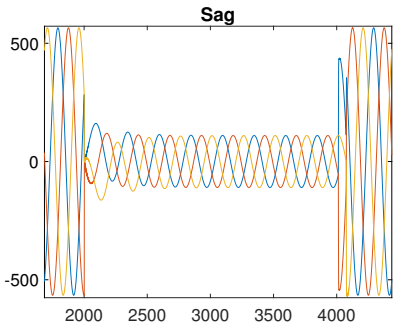
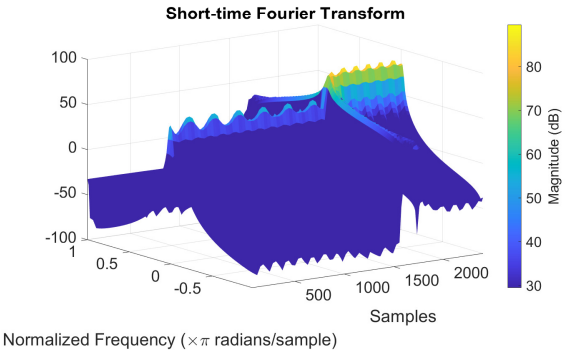


Figure A1. Simulink Model.

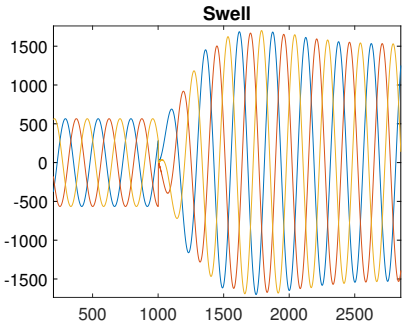
Appendix A.2. PQD and STFT



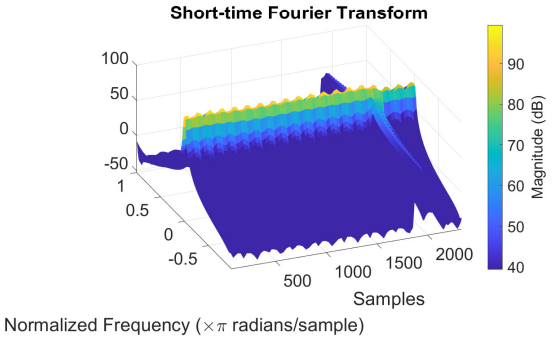
(**a**) Sag PQD



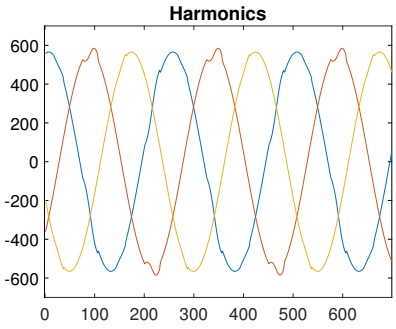
(**b**) Sag PQD STFT



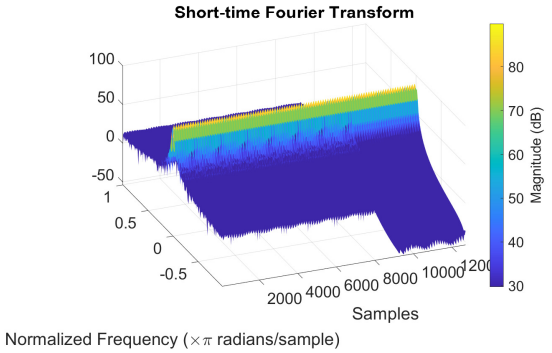
(**c**) Swell PQD



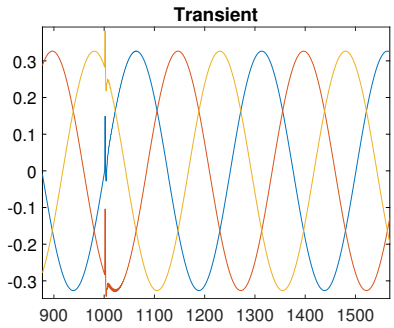
(**d**) Swell PQD STFT



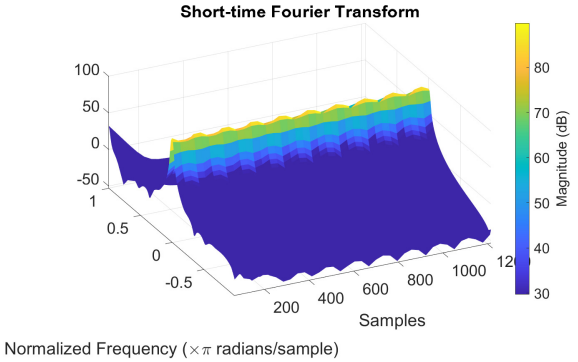
(**e**) Harmonics PQD



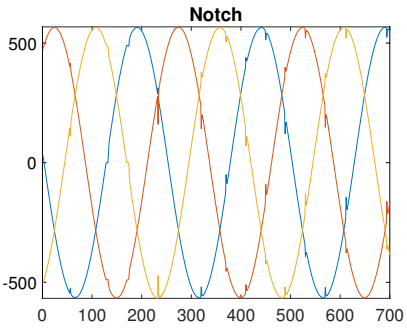
(**f**) Harmonics PQD STFT



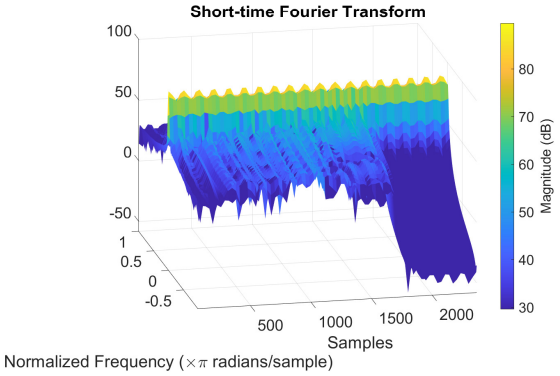
(**g**) Transient PQD



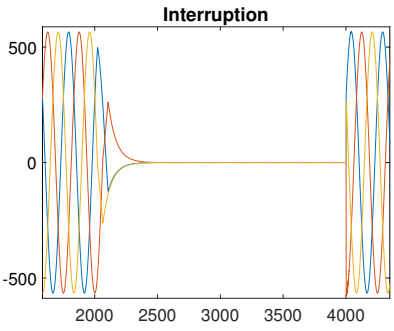
(**h**) Transient PQD STFT



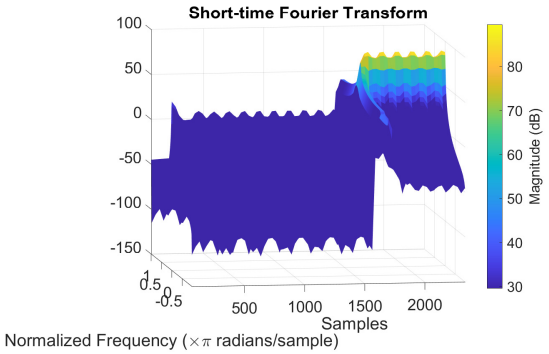
(**i**) Notch PQD



(**j**) Notch PQD STFT



(**k**) Interruption PQD STFT



(**l**) Interruption PQD STFT

Figure A2. Disturbance plots (Left) and the STFT (Right). (**a**,**b**) Sag, (**c**,**d**) Swell, (**e**,**f**) Harmonics, (**g**,**h**) Transient, (**i**,**j**) Notch, (**k**,**l**) Interruption.

References

1. Singh, B.; Al-Haddad, K.; Chandra, A. A review of active filters for power quality improvement. *IEEE Trans. Ind. Electron.* **1999**, *46*, 960–971. [CrossRef]
2. Chung, Y.; Kwon, G.H.; Park, T.; Kim, H.; Moon, J.I. Voltage sag, swell and flicker generator with series injected inverter. In Proceedings of the IEEE Power Engineering Society General Meeting, San Francisco, CA, USA, 16 June 2005; Volume 2, pp. 1308–1313.
3. Hafiz, F.; Swain, A.; Naik, C.; Abecrombie, S.; Eaton, A. Identification of power quality events selection of optimum base wavelet and machine learning algorithm. *IET Sci. Meas. Technol.* **2019**, *13*, 260–271. [CrossRef]
4. Mohan, N.; Soman, K.P.; Vinayakumar, R. Deep power: Deep learning architectures for power quality disturbances classification. In Proceedings of the International Conference on Technological Advancements in Power and Energy (TAP Energy), Kollam, India, 21–23 December 2017; pp. 1–6.

5. Alshahrani, S.; Abbod, M.; Alamri, B.; Taylor, G. Evaluation and classification of power quality disturbances based on discrete Wavelet Transform and artificial neural networks. In Proceedings of the 50th International Universities Power Engineering Conference (UPEC), Stoke on Trent, UK, 1–4 September 2015; pp. 1–5.
6. Bhavani, R.; Prabha, N.R. A hybrid classifier for power quality (PQ) problems using wavelets packet transform (WPT) and artificial neural networks (ANN). In Proceedings of the IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS), Srivilliputhur, India, 23–25 March 2017; pp. 1–7.
7. Kwan, K.H.; So, P.L.; Chu, Y.C. An Output Regulation-Based Unified Power Quality Conditioner With Kalman Filters. *IEEE Trans. Ind. Electron.* **2012**, *59*, 4248–4262. [CrossRef]
8. Soo-Hwan, C.; Jeong-Uk, K.; Il-Yop, C.; Jong-Hoon, H. Determination of Power-Quality Disturbances Using Teager Energy Operator and Kalman Filter Algorithms. *Int. J. Fuzzy Log. Intell. Syst.* **2012**, *12*, 42–46.
9. Kumawat, P.N.; Verma, D.K.; Zaveri, N. Comparison between Wavelet Packet Transform and M-band Wavelet Packet Transform for Identification of Power Quality Disturbances. *Power Res.* **2018**, *14*, 37–45. [CrossRef]
10. Liquan, Z.; Meijiao, G.; Lin, W. Classification of multiple power quality disturbances based on the improved SVM. In Proceedings of the International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, India, 22–24 March 2017; pp. 2625–2628.
11. Milchevski, A.; Taskovski, D. Classification of power quality disturbances using wavelet transform and SVM decision tree. In Proceedings of the 11th International Conference on Electrical Power Quality and Utilisation, Lisbon, Portugal, 17–19 October 2011; pp. 1–5.
12. Eristi, H.; Yıldırım, Ö.; Eristi, B.; Demir, Y. Optimal feature selection for classification of the power quality events using wavelet transform and least squares support vector machines. *Int. J. Electr. Power Energy Syst.* **2019**, *49*, 95–103. [CrossRef]
13. Cubuk, E.D.; Zoph, B.; Mane, D.; Vasudevan, V.; Le, Q.V. AutoAugment: Learning Augmentation Strategies From Data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019.
14. Fawaz, H.I.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P. Data augmentation using synthetic data for time series classification with deep residual networks. *arXiv* 2018, arXiv:1808.02455.
15. Xue, H.; Chen, A.; Zhang, D.; Zhang, C. A Novel Deep Convolution Neural Network and Spectrogram Based Microgrid Power Quality Disturbances Classification Method. In Proceedings of the 2020 IEEE Applied Power Electronics Conference and Exposition (APEC), New Orleans, LA, USA, 15–19 March 2020; pp. 2303–2307.
16. Aziz, S.; Khan, M.U.; Abdullah; Usman, A.; Mobeen, A. Pattern Analysis for Classification of Power Quality Disturbances. In Proceedings of the 2020 International Conference on Emerging Trends in Smart Technologies (ICETST), Karachi, Pakistan, 26–27 March 2020; pp. 1–5.
17. Benysek, G. *Improvement in the Quality of Delivery of Electrical Energy Using Power Electronics Systems*; Power Systems; Springer: Dordrecht, The Netherlands, 2007.
18. Grasso, F.; Paolucci, L.; Bacci, T.; Talluri, G.; Cenghialta, F.; D'Antuono, E.; Giorgis, S.D. Simulation Model and Experimental Setup for Power Quality Disturbances Methodologies Testing and Validation. In Proceedings of the 2019 IEEE 5th International Forum on Research and Technology for Society and Industry (RTSI), Florence, Italy, 9–12 September 2019; pp. 359–363.
19. Borges, F.A.S.; Fernandes, R.A.S.; Silva, I.N.; Silva, C.B.S. Feature Extraction and Power Quality Disturbances Classification Using Smart Meters Signals. *IEEE Trans. Ind. Inform.* **2016**, *12*, 824–833. [CrossRef]
20. Shen, Y.; Abubakar, M.; Liu, H.; Hussain, F. Power Quality Disturbance Monitoring and Classification Based on Improved PCA and Convolution Neural Network for Wind-Grid Distribution Systems. *Energies* **2019**, *12*, 1280. [CrossRef]
21. 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]
22. 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]

23. Ahajjam, M.A.; Licea, D.B.; Ghogho, M.; Kobbane, A. Electric Power Quality Disturbances Classification based on Temporal-Spectral Images and Deep Convolutional Neural Networks. In Proceedings of the 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 15–19 June 2020; pp. 1701–1706.
24. Pascanu, R.; Gulcehre, C.; Cho, K.; Bengio, Y. How to Construct Deep Recurrent Neural Networks. *arXiv* 2013, arXiv:1312.6026.
25. Hochreiter, S.; Schmidhuber, A.J. Long Short- Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
26. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; The MIT Press: Cambridge, MA, USA, 2016.
27. Han, D.; Liu, Q.; Fan, W. A new image classification method using CNN transfer learning and web data augmentation. *Expert Syst. Appl.* **2018**, *95*, 43–56. [CrossRef]
28. Huang, J.; Chen, B.; Yao, B.; He, W. ECG Arrhythmia Classification Using STFT-Based Spectrogram and Convolutional Neural Network. *IEEE Access* **2019**, *7*, 92871–92880. [CrossRef]
29. Yao, S.; Piao, A.; Jiang, W.; Zhao, Y.; Shao, H.; Liu, S.; Liu, D.; Li, J.; Wang, T.; Hu, S.; et al. STFNets: Learning Sensing Signals from the Time-Frequency Perspective with Short-Time Fourier Neural Networks. In Proceedings of the The World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019.
30. Tan, R.H.; Ramachandaramurthy, V.K. A Comprehensive Modeling and Simulation of Power Quality Disturbances Using MATLAB/SIMULINK. In *Power Quality Issues in Distributed Generation*; Luszcz, J., Ed.; IntechOpen: Rijeka, Croatia, 2015; Chapter 3.
31. Fayek, H.M.; Lech, M.; Cavedon, L. Evaluating deep learning architectures for Speech Emotion Recognition. *Neural Netw.* **2017**, *92*, 60–68. [CrossRef] [PubMed]
32. Buckland, M.; Gey, F. The relationship between Recall and Precision. *J. Am. Soc. Inf. Sci.* **1994**, *45*, 12–19. [CrossRef]

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).