I J C T A, 9(16), 2016, pp. 8033-8042
© International Science Press

Power Quality Signal Classification using Convolutional Neural Network

Binsha P.*, Sachin Kumar S.*, Athira S.* and K. P. Soman*

ABSTRACT

Researchers are exploring challenging techniques for the analysis of power quality signals for the detection of power quality (PQ) disturbances. PQ disturbances have become a serious problem for the end users and electric utilities. Numerous algorithms have been developed for the classification of unstructured data. In this paper, the classification of power quality signals are performed based on autoencoders and convolutional neural networks (CNN). The work focuses on classifying power quality signals into known categories of waveforms such as swell, sag, harmonics and their combinations. Signals are fed to the autoencoder for the extraction of features, which are then classified using Support Vector Machine. This classification result is compared with another technique, where the signals are given to convolutional neural network and are subjected to classification using softmax regression. The second method gave an accuracy of 97%, which is superior to that obtained by an autoencoder.

Keywords: Autoencoder, SVM, Convolutional Neural Network, Power quality signals, Neural networks, Classification.

1. INTRODUCTION

In recent years, power quality signal related issues are becoming a serious problem for the utilities and the customers. Therefore, it is essential to monitor these disturbances. Different types of signal processing techniques are used for monitoring the power quality. Monitoring systems processes power signals to distinguish the type of disturbances present in the signal. There are two types of power quality disturbances, Variations and Events. Variation refers to the small deviation from the normal waveform such as voltage fluctuations, voltage and the current imbalance, harmonic distortion and high frequency noise. Events are large disturbances that happens occasionally. Sudden disturbances like voltage swell, sag, interruptions, voltage dips, transients come under events [1]. This paper proposes a deep learning based approach for feature generation and classification of power quality signals into six classes like normal, harmonics, swell, sag, swell with harmonics and sag with harmonics. Deep learning technique, suggests incorporating artificial intelligence to the system by learning the hierarchical representations. The concept of deep learning is originated from artificial neural network. Autoencoders and convolutional neural networks are two artificial neural networks used in this paper. Autoencoder is a neural network model introduced by Geoffrey Hinton in 1980s [2]. It is an unsupervised learning algorithm, in which it automatically learns features from the unlabelled data. The second neural network used in the work is CNN. Convolutional Neural Network (CNN) is designed by motivation from the discovery of visual mechanism known as visual cortex in the brain. As the functions of a visual cortex are performed by cells, in CNN, the convolutional layer performs those functions. A convolutional neural network consists of many layers such as convolutional, pooling, activation and fully connected layers. It transforms the input data into an output with the help of some differentiable function. The classification methods used in this work are SVM and softmax regression. SVM is a supervised machine learning technique used for classification and regression. Softmax regression is a multiclass classification algorithm used when the classes are mutually

* Centre for Computational Engineering and Networking (CEN), Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, Amrita University, India, *Emails: binsha4030@gmail.com, sachinnme@gmail.com, athira3003@gmail.com, kp_soman@amrita.edu.*

exclusive. This paper focuses on the comparison of the classification accuracy of power quality signals by two methods. In the first method, feature extracted from autoencoder is classified using SVM whereas, in the second method, signals are fed to CNN which is followed by a softmax layer for classification.

In this paper, section 2 discusses about the methodologies such as autoencoder, CNN, SVM and softmax regression, that are required for the work. Section 3 describes the experiments done. In section 4 the results obtained from the experiment are discussed. Finally, conclusion is drawn in section 5.

2. METHODOLOGY

2.1. Autoencoder

Many simple neurons are interconnected to form a neural network. Neural network defines a complex and non-linear hypotheses H with parameters as weights (W) and biases (b) that fits the data. These weights and biases are fixed to an optimum value by minimizing the error between the predicted output $H_{w,b}(x)$ and actual output y. Autoencoders are simple learning networks that aims at transforming the input into some representation so as to reconstruct the input with least possible amount of distortion [3][4]. It is a multi layered neural network with a small intermediate layer that can be trained to convert the input data into a lower or higher dimensional code so that the input vectors are reconstructed. In such autoencoder networks, fine tuning of weights can be done using gradient descent method. An autoencoder has input, hidden and output layers. The network can have one or more hidden layers. Fig. 1 shows the network architecture of an autoencoder with one hidden layer. The values at the nodes in the hidden layer will be kept hidden there by providing data abstraction. All the connections between the nodes of different layers are associated with a weight and each node has a bias[5]. Consider the feed forward network Fig. 2. The network has three units in the input and hidden layer, and one unit is the output layer. The +1 units are the bias units, which represents the intercept term. Let N_l be the number of layers in the architecture shown in Fig. 2. l^{th} layer is represented as L_l. U_l is the total count of units in the layer $l.w_m^{(t)}$ denotes the weight of the connection between the q^{th} unit in the l^{th} layer and the p^{th} unit in the $(l + 1)^{th}$ layer. $b_p^{(t)}$ represents the bias associated with the p^{th} unit in the $(l + 1)^{th}$ layer. $A_p^{(t)}$ denotes the activation of p^{th} unit in l^{th} layer. For training an autoencoder, activations of each unit are found by performing feed forward pass. Thus, finally, the predicted outputs will be obtained as the activations of the output units. The partial derivatives of the cost function needed in gradient descent are found through back propagation algorithm. If $Z^{(l+1)}$ denotes the inputs to $(l +1)^{th}$ layer, recalling the inputs x as $A^{(t)}$, the activations can be found as

$$Z^{(l+1)} = w^{(l)} A^{(l)} + b^{(l)} \tag{1}$$

$$A^{(l+1)} = F\left(Z^{(l+1)}\right) \tag{2}$$

where $F(.)$ is the activation function. In case of a training data set with a single example (x, y), the squared error cost function is $C\left(w,b;x,y\right)=\frac{1}{2}\left\|H_{w,b}\left(x\right)-y\right\|^2$. For examples, the cost function becomes sum of squared error term and a regularization term to prevent over fitting. The intention of autoencoder is to minimize C with respect to W and b so as to reduce the error between the predicted and actual output. For training the neural network, some random values approximately equal to zero are assigned to both W and b. Applying gradient descent algorithm, parameters are fixed with an optimized value by minimizing the cost function. The parameters w and b are updated as:

$$C\left(w,b\right)=\frac{1}{m}\sum_{p=1}^{m}\frac{1}{2}\left\|H_{w,b}\left(x\right)-y\right\|^2+\left[\frac{\lambda}{2}\sum_{l=1}^{N_l-1}\sum_{p=1}^{U_l}\sum_{q=1}^{U_l+1}\left(w_{qp}^{(l)}\right)^2\right] \tag{3}$$

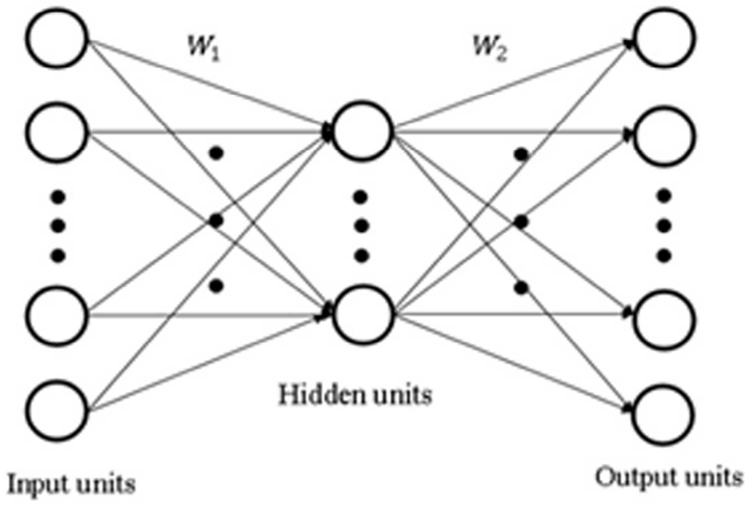


Figure 1: Autoencoder with one hidden layer

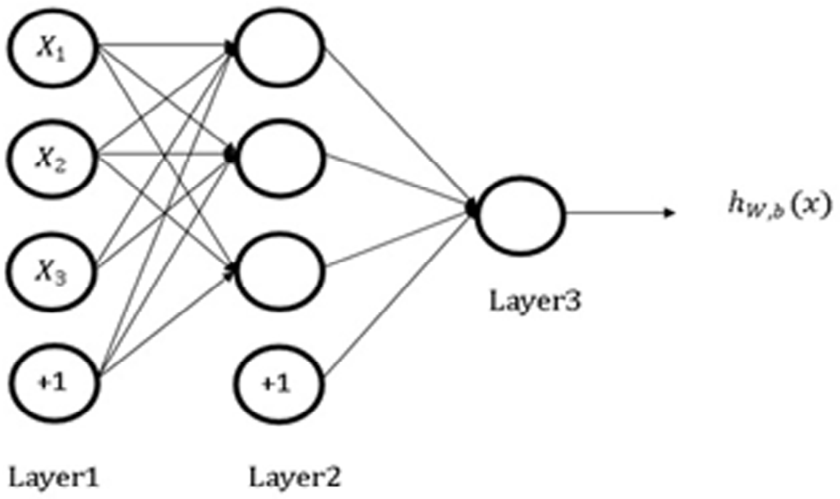


Figure 2: Feed forward network

$$w_{pq}^{(l)} = w_{pq}^{(l)} - \alpha \frac{\delta C(w,b)}{\delta w_{pq}^{(l)}} \tag{4}$$

$$b_{p}^{(l)} = b_{p}^{(l)} - \alpha \frac{\delta C(w,b)}{\delta b_{p}^{(l)}} \tag{5}$$

where α is the learning rate and

$$\frac{\delta C(w,b)}{\delta w_{pq}^{(l)}} = \left[\frac{1}{m}\sum_{p=1}^{m} \frac{\delta C\left(w,b;x^{(p)},y^{(p)}\right)}{\delta w_{pq}^{(l)}}\right] + \lambda w_{pq}^{(l)} \tag{6}$$

$$\frac{\delta C(w,b)}{\delta b_{p}^{(l)}} = \left[\frac{1}{m}\sum_{p=1}^{m} \frac{\delta C\left(w,b;x^{(p)},y^{(p)}\right)}{\delta b_{p}^{(l)}}\right] \tag{7}$$

These partial derivatives of the cost function, which are needed in gradient descent method, are computed by back propagation algorithm as follows:

1. Perform a feed forward algorithm to find activation of all the nodes in the network.
2. In the output layer N_l, compute $e^{(N_l)} = \left(y - A^{(N_l)}\right).F'\left(Z^{N_l}\right)$; which measures how long the node is subjected to the error in the output.
3. Find the error terms of the nodes from the last to the first hidden layer. For

$$l = N_l - 1, N_l - 2, \ldots\ldots.., 2,\ e^{(l)} = \left(w^{(l)}\right)^T e^{(l+1)}.F'\left(Z^{(l)}\right)$$

4. Partial derivatives can be calculated as:

$$\nabla_{w^{(l)}} C\left(w, b; x, y\right) = e^{(l+1)}\left(A^{(l)}\right)^T, \tag{8}$$

$$\nabla_{b^{(l)}} C\left(w, b; x, y\right) = e^{(l+1)}. \tag{9}$$

2.2. Convolutional Neural Network

Convolutional Neural Network (CNN) is an artificial neural network. Different layers are stacked together to form CNN architecture. Commonly, four types of layers used to build convolutional neural networks are: convolutional layers, pooling layers, nonlinear or activation layers and fully connected layers. Four hyper parameters of CNN are number of filters (*K*), their spatial extend (*F*), stride (*S*) and zero padding (*P*). Convolution layer plays a key role in feature extraction. It contains filters that are convolved with the input and the features are extracted. Convolution is performed by sliding each filter over the input, thereby summing up the dot product of the elements of the input and filter[5]. Each filter extracts one feature from the input. The weights of the filters in the convolution layer are computed as part of the training process. Each feature of a layer receives an input from a small neighborhood of the previous layer, known as receptive field. Convolutional layers extracts the local features because, they restrict the receptive fields of the hidden layers to be local. Thus, CNN has a locally connected structure. CNN may contain one or more convolutional layers. In the case of more than one convolution layer, the activation map from one layer is fed as input to the next convolutional layer and so on. Activation map is a sheet of neuron output in which, each neuron is connected to a small region in the input called receptive field and all of them share same parameters. But, the parameters will be different for different filters; (*K*) filters maps to (*K*) different activation maps. Higher level features are extracted by higher level layers. The convolved results are then activated by the activation functions. Second layer of a CNN contains activation functions for the identification of like features. CNNs uses some specific functions such as Rectified Linear Units (ReLUs) [6] and continuous trigger functions. ReLU is the rectified linear unit. It implements the function $y = \max(x, 0)$. Continuous trigger functions includes tanh and sigmoid functions. The third layer, pooling layer is the sub sampling or down sampling layer. In convolution layer, the spatial volume of the input is preserved. The pooling layer shrinks the input volume spatially. It reduces the resolution of the features. Down sampling operations are performed on every single activation map independently. Two types of pooling that are commonly used are: average pooling and max pooling. Input is divided into non overlapping regions. In average pooling, the average and in max pooling, the maximum of the data points in those non overlapping regions is the output. Fully connected layers are the final layers of the CNN. This layer contains neurons that connect to the entire input volume as in ordinary neural network. The specified target output scores is obtained from this layer.

Fig. 3 shows a CNN with one convolution layer of filters represented as , which computes certain features of the input . Then, the output is fed into a fully connected layer F. Here, A considers only two data points. The window size of the convolutional layer would be much larger. Also, more convolutional layer can be used in a CNN. By adding more convolutional layers, more abstract features can be extracted. A max pooling layer considers small blocks of features from the previous layer and finds the maximum from each block. The output from the pooling layer gives information regarding the presence of a feature in a region of the preceding layer, but not the precise location of it's presence. Max-pooling layers can be thought of as a zoom out [7]. The Fig. 4 shows a CNN with two convolutional layers, one with filters A and the other with filters B. Both figures show the architecture for one dimensional data. Higher dimensional data can be fed to CNN. While using higher dimensional data, instead of segments, filters will look for patches.

2.3. Support Vector Machine

SVM [8] is a supervised learning approach for classification and regression. SVM aims at finding a target value for the test data based on the trained model. In SVM, a linear classifier (hyperplane) separates different classes with a maximum separation. The hyperplane that separates two classes is represented as:

$$w^T x_i - \xi = 0 \tag{10}$$

where $w = [w_1, w_2, ...w_n]^T$, $\xi \in \mathbb{R}$, x_i is a training sample with a label $y_i \in \{-1,1\}$. The decision function is given by $\operatorname{sign}\left(w^T x_i - \xi\right)$. The problem formulation of SVM in matrix format for this case is:

$$\begin{gathered} \min_{w,\xi} \frac{1}{2} w^T w \\ subject\,to : D\left(Xw - \xi e\right) \geq e \end{gathered} \tag{11}$$

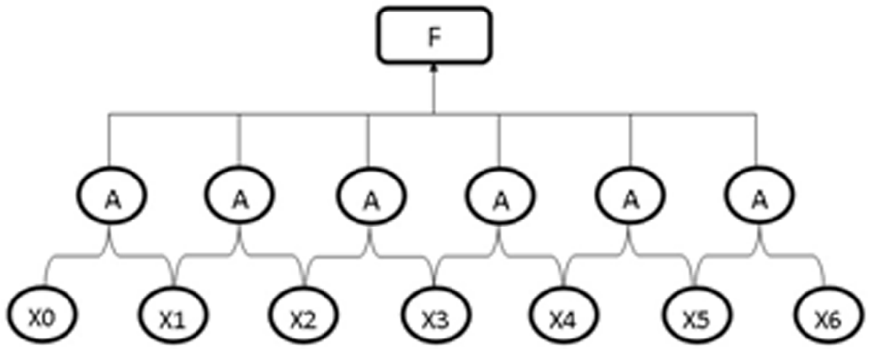


Figure 3: CNN with one convolution layer and one fully connected layer.

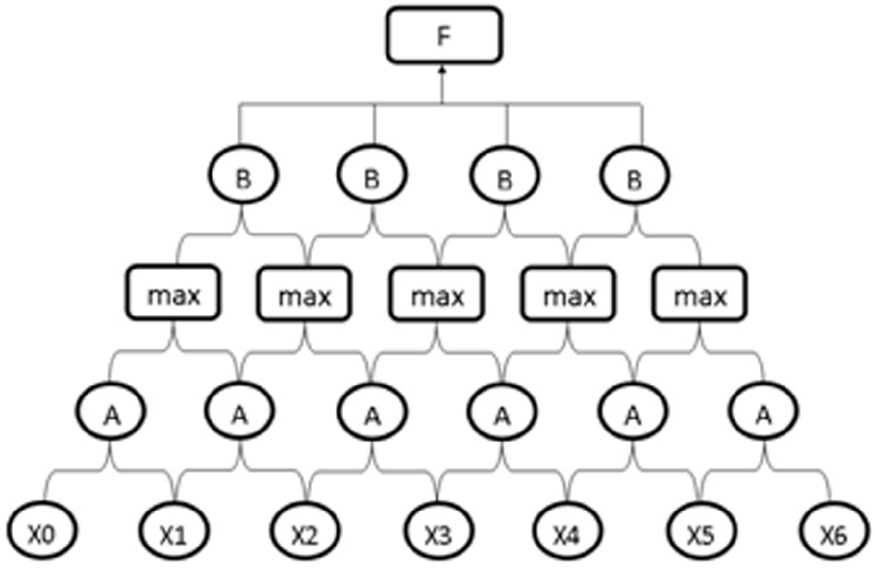


Figure 4: Convolutional Neural Network.

where D is a $m \times m$ diagonal matrix with elements of , the label set as diagonal elements, X is the $m \times n$ data matrix and e is $m \times 1$ column vector of ones. This formulation is valid only for linear data. In case of non-linearly separable data, a mapping operator ϕ is employed to map the data to a relatively manageable higher dimension. The above mentioned formulation is efficiently implemented in LibSVM [9] a kernel based software library which supports multiclass classification [10].

2.4. Softmax Regression

Softmax regression [5] is a multinomial logistic regression. Logistic regression is a binary classifier whereas, softmax regression is a multi class classifier. In softmax regression, the hypothesis is to find the probability of getting each classes. Let the training dataset contains $m(x_i, y_i)$ pairs and K classes. The output of the hypothesis function will be the probabilities estimated:

$$\left(x^{(i)}\right) = \begin{bmatrix} p(y^{(i)} = 1 \mid x^{(i)}; \theta) \\ p(y^{(i)} = 2 \mid x^{(i)}; \theta) \\ \cdots\cdots\cdots \\ p(y^{(i)} = K \mid x^{(i)}; \theta) \end{bmatrix} = \frac{1}{\sum_{1 \le j \le K} \exp(\theta_j^T x^{(i)})} \begin{bmatrix} \exp(\theta_1^T x^{(i)}) \\ \exp(\theta_2^T x^{(i)}) \\ \cdots\cdots \\ \exp(\theta_K^T x^{(i)}) \end{bmatrix} \tag{12}$$

where θ represent the parameters. On classifying the test data, a category with highest probability will be selected. In softmax regression, the parameters are found by optimizing the cost function

$$C(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} \sum_{j=1}^{K} 1\{y^{(i)} = j\} \log \frac{\exp(\theta_j^T x^{(i)})}{\sum_{1 \le l \le K} \exp(\theta_l^T x^{(i)})}\right] \tag{13}$$

3. EXPERIMENT

The experiments are done on the data set of power quality signals that are generated in Matlab. The data set consists of six classes of signals, each representing the disturbances that may occur in power line signals. Each class contains 100 signals of length 2560. Normally, the features that are extracted from the data are classified using some algorithms.

In this work, the signals are first fed to autoencoder with single hidden layer for feature extraction. Different types of features are extracted by varying the number of hidden units in the autoencoder. The features extracted are then classified using Support Vector Machine. LibSVM package is used for classification. It is commonly used for binary classification. In this work, multiclass classification is performed

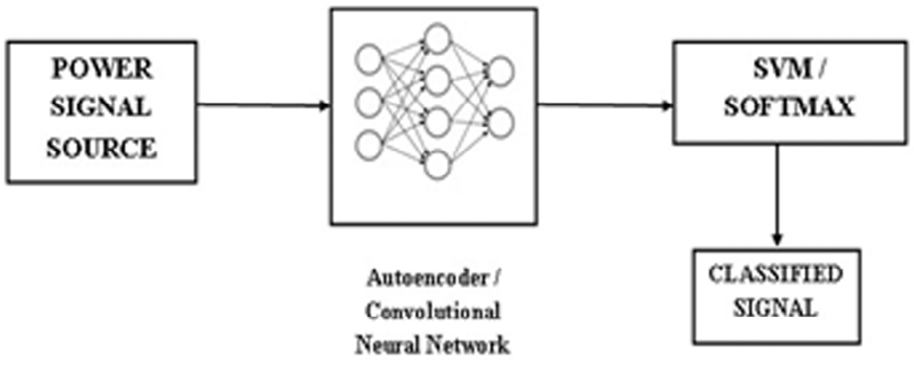


Figure 5: Flow diagram of classification using neural networks.

using LibSVM. Fig. 5 shows the flow graph of the steps performed in this work. Similarly, the signals are fed to convolutional neural network followed by a softmax layer. Feature extraction and classification accuracy were compared by varying the number of filters in the convolutional layer. Higher classification accuracy was observed on using CNN.

4. RESULTS AND DISCUSSIONS

4.1. Power signal generation

The power signals for classification are synthetically generated in Matlab. Fig. 6 shows the signals generated. Six classes of signals, one of normal signal and the other five with disturbances, such as swell, sag, harmonics, swell with harmonics and sag with harmonics are generated in Matlab using the following parametric equations as in Eq.14-Eq. 19[11]:

- Normal Signal

$$V = Amp * \sin(\omega * t) \tag{14}$$

where, *Amp* is the signal magnitude, ω is the power system frequency and *t* is the sampling time.

- Sag Signal

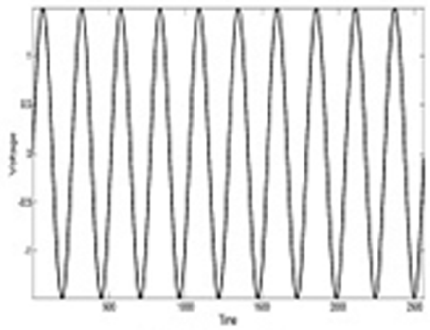
$$V = Amp * \left(1 - A_s \left(u_s\left(t - t_1\right) - u_s\left(t - t_2\right)\right)\right) * \sin(\omega * t) \tag{15}$$

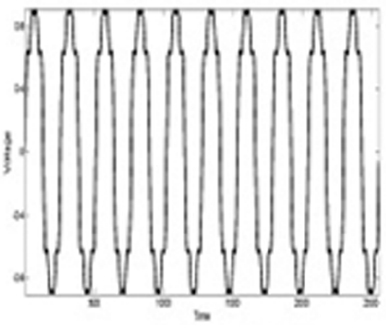
where $u_s(t)$ is the unit step function and A_s represents the magnitude of sag.

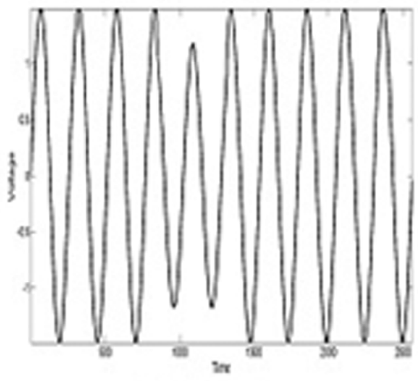
- Swell Signal

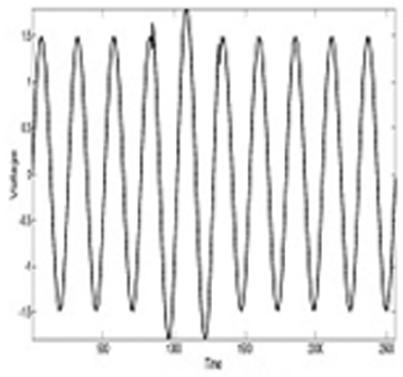
$$V = Amp * \left(1 + A_s \left(u_s\left(t - t_1\right) - u_s\left(t - t_2\right)\right)\right) * \sin(\omega * t) \tag{16}$$

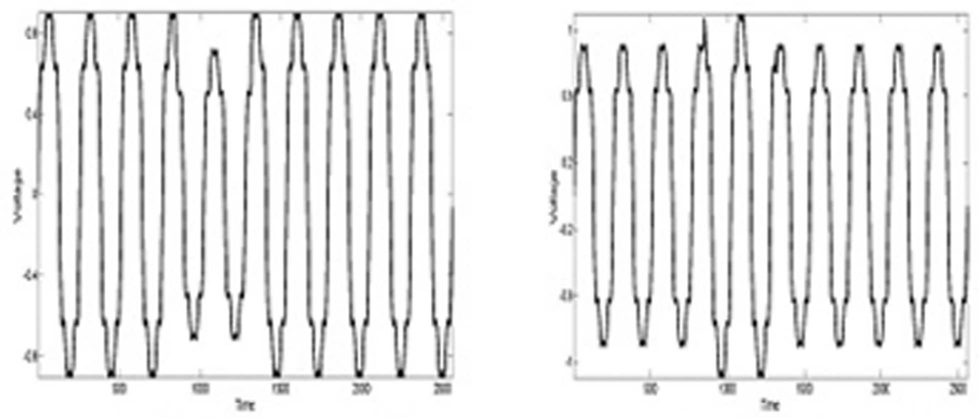
where, A_3 represents the magnitude of swell.


(a) Normal


(b) Harmonics


(c) Sag


(d) Swell



(e) Sag with harmonics (f) Swell with harmonics

Figure 6: Power signals

- Harmonics

$$V = Amp * \sin(\omega * t) + Amp * \sin(3 * \omega * t) + \\ Amp * \sin(5 * \omega * t) + Amp * \sin(7 * \omega * t) \tag{17}$$

- Sag with harmonics

$$V = Amp * \left(1 - A_s \left(u_s (t - t_1) - u_s (t - t_2)\right)\right) * \\ \sum_{i \in \{1,3,5,7\}} \alpha_i * \sin(i * \omega * t) \tag{18}$$

where, A_3 is the magnitude of sag and $\alpha_1, \alpha_3, \alpha_5, \alpha_7$ are the magnitudes of 1st, 3rd, 5th and 7th harmonics respectively.

- Swell with harmonics

$$V = Amp * \left(1 + A_s \left(u_s (t - t_1) - u_s (t - t_2)\right)\right) * \\ \sum_{i \in \{1,3,5,7\}} \alpha_i * \sin(i * \omega * t) \tag{19}$$

where, A_3 is the magnitude of the swell and $\alpha_1, \alpha_3, \alpha_5, \alpha_7$ are the magnitudes of 1st, 3rd, 5th and 7th harmonics respectively.

Table 1
Classification accuracy using SVM

No. of hidden units	*Overall accuracy (%)*	*Classwise Accuracy (%)*					
		Harmonics	*Normal*	*Sag*	*Sag with harmonics*	*Swell*	*Swell with harmonics*
500	36.49	58.42	49.5	54.7	10.9	15.9	29.5
1500	18	19.22	13.4	16	8.8	22.8	28.2
5000	16.61	24.8	10.2	12	20.3	18.9	13.2

4.2. Classification using SVM

Power signals are fed to an autoencoder with single hidden layer. The signals that are considered for classification has a length of 2560. Thus, 2560 units are considered in the input and output layer of autoencoder. The signals are mapped into feature space by the autoencoder. The features are extracted at the hidden layer and then classified using support vector machines. Number of nodes taken in the hidden layer are varied, such as 500, 1500, 5000 and the classification accuracy on each case are noted. The algorithm of Support Vector Machine is used for classification. The LibSVM library is used for training and the prediction of class value of the test data. LibSVM has options for selecting the type of classification, kernel, and different parameters. 50% of the signals from each class is used for training and the machine is tested using the remaining 50% for finding the accuracy. Table 1 shows the accuracy obtained on classifying power signals using SVM. Overall accuracy of classification decreased on increasing the number of units in the hidden layer of autoencoder.

4.3. Classification using CNN

CNN with two convolutional layers and two fully connected layers is considered. 50% signals from each class is used for training and the remaining 50% for testing. Classification accuracy of the test input data is calculated by changing the number of filters in each convolutional layer. Table 2 shows the accuracy obtained on using CNN for classification. The number of filters is varied as 512, 256 and 128 in first convolution layer and 8 and 16 in second convolution layer. Accuracy of 97% is obtained when 256 and 16 filters are used in first and second convolutional layers respectively.

Table 2
Classification accuracy using CNN

No. of filters in layer 1	*No. of filters in layer 2*	*Accuracy (%)*
512	8	90.19
256	8	83.39
128	8	87.09
512	16	78.19
256	16	96.99
128	16	69.49

5. CONCLUSION

Power quality signals are generated in Matlab and are classified using neural networks. Six types of signals where generated, one of normal type and other five with disturbances like swell, sag, harmonics, swell with harmonics and sag with harmonics. Signals of length 2560 were mapped on to a dimension of 500, 1500 and 5000 by using autoencoder and these are classified using SVM. Test data were not correctly classified. Classification using CNN resulted in better accuracy. On using 256 filters in the first convolution layer and 16 filters in the second convolution layer, 97% of test data were correctly classified by CNN.

ACKNOWLEDGMENT

We would like to show our gratitude to CEN Department and Neethu Mohan, Premjith, Research scholars of CEN Department, for giving us a good guideline during the period of our work. We would also like to express our deepest gratitude to all those who have directly and indirectly guided us in writing this paper.

REFERENCES

[1] Ms. Ankita Dandwate, Dr. Prof. K. B. Khanchandani. “Generation of Mathematical Models for various PQ Signals using MATLAB”. *International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622, International Conference On Industrial Automation And Computing (ICIAC- 12-13 April)* 2014.

[2] Hinton, Geoffrey E. and Salakhutdinov, Ruslan R. “Reducing the dimensionality of data with neural networks”. *Science*. American Association for the Advancement of Science; 2006; 313:504-507

[3] Baldi, Pierre. “Autoencoders, unsupervised learning, and deep architectures”. *Unsupervised and Transfer Learning Challenges in Machine Learning*. 2012; 7:43.

[4] Bengio, Yoshua. “Learning deep architectures for AI”. *Foundations and trends R in Machine Learning*. Now Publishers Inc. 2009; 2:1-127.

[5] Unsupervised Feature Learning and DeepLearning *http://udl.stanford.edu/tutorial/unsupervised/Autoencoders/*.

[6] “Using Convolutional Neural Networks for Image Recognition”. *Cadence Design Systems*. 2015.

[7] Conv Nets: A Modular Perspective. *http://colah.github.io/posts/2014-07-ConvNets-Modular/*. 2014.

[8] Soman KP, Loganathan R, Ajay V. Machine learning with SVM and other kernel methods. *PHI Learning Pvt. Ltd.* 2009.

[9] Chang, Chih-Chung and Lin, Chih-Jen. “LIBSVM: a library for support vector machines”. *ACM Transactions on Intelligent Systems and Technology (TIST).* 2011.

[10] Nikhila Haridas, V. Sowmya and K. P. Soman. “GURLS vs LIBSVM: Performance Comparison of Kernel Methods for Hyperspectral Image Classification”. *Indian Journal of Science and Technology, Vol 8(24), DOI: 17485/ijst/2015/v8i24/ 80843, September*. 2015.

[11] Ravi Saxena, A.K.Swami, Sanjay Mathur. “A Power Quality Signal Generator in Lab View Environment”. *Proc. of the Intl. Conf. on Advances in Electronics, Electrical and Computer Science Engineering EEC.* 2012.