Identification of line status changes using phasor measurements in transient states through deep learning networks

Natalja Gotman*, *Galina* Shumilova

Federal Research Center "Komi Scientific Center of the Ural Branch Russian Academy of Sciences", ISE and EPN, 167000, Russia

Abstract. The problem of detecting changes in a topology of an electrical network in real time is solved. This paper proposes a line state detection method based on a convolutional neural network (CNN) classifier using phasor measurements of bus voltages and currents in transient states.

1 Introduction

The creation of a topological model of an electric power system (EPS) is a very important stage in modeling an EPS in real time. Errors in this model can lead to incorrect and potentially dangerous control actions. In this situation, checking the correctness of an electrical power network topology model is a great importance.

Effective management of a modern power system with the presence of decentralized energy sources requires reliable methods for power distribution network topology detection. These methods must be supported by modern measuring systems provided with the phasor measurement units (PMUs). PMUs have the ability to record fast transients with high precision. Indeed, these events can occur in a few seconds, which hampers their detection by the traditional Supervisory Control And Data Acquisition (SCADA) systems. Time synchronization of geographically dispersed measurements provides better operational awareness about the network topology in real time.

With the advent of Wide Area Measurement Systems (WAMS) and the consequent deployment of such monitoring devices, control centers are being flooded with massive volumes of data. PMUs provides from 10 to several hundred samples per second, which is very high when compared to the traditional SCADA acquisition devices that sample every 2–4 seconds [1]. This new paradigm represents a huge amount of raw data collected every day. For instance, [1] refers that a single PMU sampling at 60 Hz can create roughly 721 MB of data per day, and 44 PMUs generate approximately 1 TB per month. Therefore, as stated here, large-scale PMU systems present challenges for such a volume of data processing. According to [2], this problem can be solved using techniques in the field of Artificial Intelligence, like Machine Learning, in particular, Deep Learning. These methods can be very helpful to extract features from raw data, and the latest machine learning algorithms are able to find the features themselves by which the input data can be classified.

A brief introduction to Deep Learning Neural Networks (DLNN), and, in particular, to Convolutional Neural Networks (CNNs), is presented in section 1. Section 2 is devoted to the study of the use of CNN to detect line status changes in transient states with the presentation of research results. Conclusion concludes results and discusses future research.

2 Deep Learning Neural Networks

Deep Learning Neural Networks are becoming one of the most popular Machine Learning Methods for creating Artificial Intelligence Systems in various fields, such as speech recognition, natural language processing, computer vision, medical informatics, etc. [1]. Their distinctive features are a larger number of neurons in layers, more complex connection methods, automatic feature extraction, and an increase in computation power and access to enough data to train the algorithms [3]. Many research works have been done recently on DLNN in power systems. DLNN application in power systems are following: load forecasting, power system restoration, detection of defect or faulty equipment, stability assessment, disturbance and emergency control, cyber security, power system fault diagnosis [4], real-time faulted line localization [5], etc.

One of the reasons for DLNN's success is that the network automatically extracts important features from the data that are necessary to solve the problem. When processing large amounts of data, the neural network copes with feature extraction much better than the person himself.

The multilayer perceptron [6] is an example of a deep neural network architecture. Such a network is called fully connected. There are other DLNN architectures, in particular, neocognitron, autoencoders, convolutional neural networks, limited Boltzmann machine, deep trust networks, long-short-term memory networks, driven

* Corresponding author: gotman@energy.komisc.ru

recurrent neural networks, and residual learning networks [7].

Convolutional neural networks are most suitable for solving the problem of detection of line status changes. CNN classifier is preferred considering its property of sparse connectivity and parameter sharing [5], so we will focus on a more detailed presentation of these networks.

CNNs are a special type of Feed-Forward Neural Networks for processing data that has grid-like topology. The CNN structure is aimed at efficient recognition of the input information. The CNN receives input data, transforms its using a number of interconnected layers. The output layer represents a set of probabilities (estimates). Although there is no uniform way of designing the structure of CNN, several basic components are typically considered together for better classification accuracy in a wide range of applications. These components include convolutional, Rectified Linear Unit (ReLU), Pooling, and fully connected operators (Fig. 1).

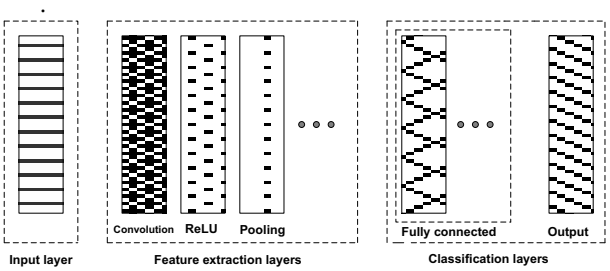

Fig. 1. Convolutional neural network architecture.

There are three main groups of layers: 1) input layer; 2) feature extraction layers; 3) classification layers.

The input layer accepts 3D signals. Feature extraction layers have a repeating structure: convolution (filter) → activation (ReLU) → pooling. The input and output of each layer are called feature maps. A filter layer convolves its input with a set of trainable kernels.

The convolutional layer is the core building block of a CNN and exploits spatially local correlation by enforcing a local connectivity pattern between neurons of adjacent layers. The connections are local, but always extend along the entire depth of the input volume in order to produce the strongest response to a spatially local input pattern. The convolution operation is shown in Fig. 2.

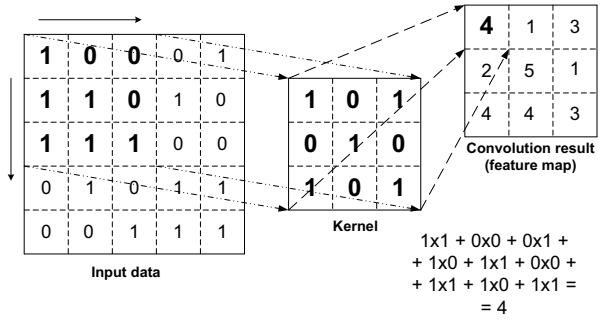

Fig. 2. The convolution operation.

The convolution input is the original data or a feature map calculated by another convolution. The size of all maps of the convolutional layer is the same and is calculated by the formulas [3]:

$$\left.\begin{aligned} w &= mW - kW + 1, \\ h &= mH - kH + 1, \end{aligned}\right\} \quad (1)$$

where w, h – calculated, respectively, the width and height of the convolution map; mW – the width of the previous card; mH – the height of the previous map; kW – kernel width; kH – kernel height.

During convolution, the kernel moves in two directions – along the width and height of the input feature map – thus the final output is 2D (Fig. 2). The result is the sum of the products of the kernel elements by the corresponding elements of the part of the input map. The convolution operation is mathematically described by the formula:

$$(F \times K)[a,b] = \\ = \sum_{x=1}^{kH} \sum_{y=1}^{kW} F[a + x - 1, b + y - 1] \times K[x, y] \quad (2)$$

where F is the input feature map, K is the convolutional kernel, FxK is the convolution result.

The activation function (such as sigmoid and tanh) introduces non-linearity into the networks and allows them to learn complex models. Here we applied ReLU (Rectified Linear Units) because it trains the neural networks several times faster without a significant penalty to generalisation accuracy.

ReLU which is actually an activation function is shown here as a layer since it is so accepted in the literature. The linear rectification function activates the block if the input signal is greater than the set value. The function is described by the formula $f(x) = \max(0, x)$, its diagram is shown in Fig. 3. ReLUs are widely used in modern deep networks because they work well in many situations [3].

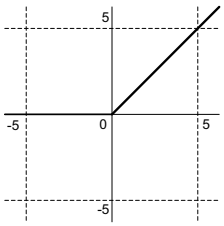

Fig. 3. Linear rectification function

Pooling reduces the resolution of input and make it robust to small variations for previously learned features. It combines the outputs of i –1th layer into a single input in i th layer over a range of local neighborhood. Pooling layer operates on each feature map independently. Two common functions used in the pooling operation are:

- Average Pooling: Calculate the average value for each patch on the feature map.
- Maximum Pooling (or Max Pooling): Calculate the maximum value for each patch of the feature map. The most common approach used in pooling is max pooling.

The result of using a pooling layer and creating down sampled or pooled feature maps is a summarized

version of the features detected in the input. They are useful as small changes in the location of the feature in the input detected by the convolutional layer will result in a pooled feature map with the feature in the same location. This capability added by pooling is called the model's invariance to local translation.

Fully connected layers are an essential component of CNNs. The result of the convolution and pooling feeds into a fully connected neural network structure that drives the final classification decision. The output of convolution/pooling is flattened into a single vector of values, each representing a probability that a certain feature belongs to a label. The fully connected part of the CNN network goes through its own backpropagation process to determine the most accurate weights.

Training the above CNN architecture is similar to the MLPs. Gradient-based optimization method (error back-propagation algorithm [7]) is utilized to estimate parameters of the model. For faster convergence, the stochastic gradient descent is used for updating the parameters. The training phase has two main steps: propagation and weight update. Each propagation involves feedforward and error back-propagation passes. Former determines the feature maps on input vector by passing from layer to layer until reaching the output (left to right in Figure 1). Latter, calculates the propagation errors according to the loss function for the predicted output (error propagates from right to left in Figure 1). Predicted error on each layer is used for calculating the derivatives by taking advantage of chain-rule of derivative. Once the derivatives of parameters obtained, the weight is updated as follows: the weight's output delta and input activation are multiplied to find the gradient of the weight. And then, a ratio (learning rate) of the weight's gradient is subtracted from the weight. This learning cycle is repeated until the network reaches a satisfactory validation error.

Each neuron receives weights that prioritize the most appropriate label. Finally, the neurons "vote" on each of the labels, and the winner of that vote is the classification decision.

3 Experiments and results

To detect power network topology changes the CNN was used oriented to the IEEE 14-bus test system (Fig. 4).

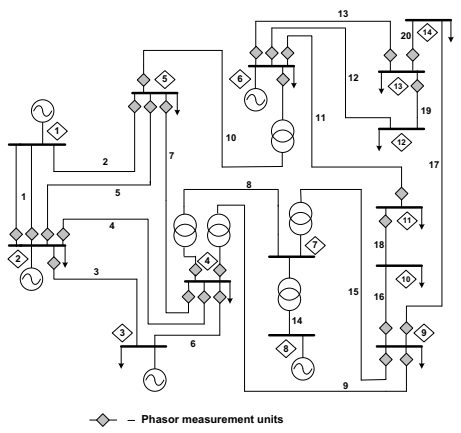

Fig. 4. IEEE 14-bus test system

To simulate outage/ turning on of line 500 modes were calculated by means of the load changes in all load buses in the range of 70–150 percent of the base level and adding to the obtained values a random value equal to 0–20 percent of the base load of the bus. For each mode the calculations of transients when there was an outage of one of the lines and turning on the line by an auto-reclosing device after three seconds were performed. Total 500 datasets (200 are training datasets and 300 as validating datasets) are employed to train the CNN classifier for the 14-bus power system. The measurement values obtained by calculations using PCC "RastrWin3" were used with a randomly added noise: a change of the voltage phase by ±0.5° and the current magnitude by ±0.5%. The relative error of noise is slightly increased, since phasor measurements contain not only the error of the PMU (± 0.2%), but also the error of the measuring transformers.

The problem of the line status detection at any moment of the transient process was considered. For this, calculations were carried out with three options of input datasets. We used as input datasets the bus voltage phase changes and the line current module changes reported by PMUs during the transient process. We generate datasets with the simulation step of 0.1 second.

The first option. The number of input parameters for the CNN equal to 27. There are 7 voltage PMU measurements at the buses where PMUs are installed and 20 current PMU measurements in lines incident to the buses where PMUs are located.

Changes in the measurement values for one time slice were used (one time slice – the difference between measurements of the current time slice and measurements of the previous time slice). The input of the CNN is the three-dimensional matrix 1x27x1.

The second option. Changes in the measurement values for two consecutive time slices were used. The number of input parameters for the CNN equal to 54. The input of the CNN is the three-dimensional matrix 2x27x1.

The third option. Changes in the measurement values from 1 to 20 time slices were used. The number of input parameters for the CNN varied from 27 to 540. The inputs of the CNN are the three-dimensional matrices nx27nx1, where n = 1, ..., 20.

The calculations were carried out using a program developed in the programming language Julia (version 1.4) implementing the Flux package (a machine learning library that also includes functions for creating CNN models). The number of datasets for training and testing of the CNN varied depending on the version of the input dataset.

All options used the ReLU activation function for convolutional layers. The classification task is performed by employing a *softmax* logistic regression layer as the output layer. To update the weights when the neural network is train a loss function is used. The loss function for all tested versions is a cross entropy. The cross entropy with "Adam" optimizer is used.

The calculation results with the first two options of the input parameters are given in Table 1 and Table 2.

Analysis of the accuracy of calculations and localization of errors showed that calculations with data of later time slices relative to the moment of the line fault show a decrease in accuracy. The error in detecting the line status increases for 115 kV lines (lines 11-20) and changes a little for 230 kV lines (lines 1-7).

Table 1. The comparison of the results between two calculation options of the topology detection.

Option	Data (the number of time slices when there was the line outage and it was turned on by an auto-reclosing device)	The number of samples		The number of misclas-sified samples at testing	Accuracy calculation, %
		for training	for testing		
1	1	6000	9000	0	100
	2	12000	18000	437	97,57
	3	18000	27000	1169	95,67
	4	24000	36000	4043	88,77
	5	30000	45000	5571	87,62
	6	36000	54000	7587	85,95
	7	42000	63000	9866	84,34
	8	48000	72000	10858	84,92
	9	54000	81000	11729	85,52
	10	60000	90000	13428	85,08
2	by 2 from 2	6000	9000	7	99,92
	by 2 from 3	12000	18000	347	98,07
	by 2 from 4	18000	27000	1209	95,52
	by 2 from 5	24000	36000	2110	94,14
	by 2 from 6	30000	45000	3150	93,00
	by 2 from 7	36000	54000	4463	91,55
	by 2 from 8	42000	63000	6256	90,07
	by 2 from 9	48000	72000	7582	89,47
	by 2 from 10	54000	81000	8011	90,11
	by 2 from 11	60000	90000	9486	89,46

This happens because the phase of the voltage in the buses where the PMUs are located changes a little, and small power flows along the 115 kV lines incident to these buses. For 230 kV lines, the error is less than 1% with the second option of the input parameters for the CNN (Table 2).

The calculation results with the first two data sets suggested the third dataset version. The essence of this version is as follows. At each time slice, the amount of the input data increases by 27 parameters, and the CNN input is a matrix, the number of rows of which is increased by one in comparison with the previous matrix. Based on this, each time slice requires its own CNN architecture. At the same time, the number of samples for training and testing does not change (6000 for training and 9000 for testing), and the accuracy of line status detection with the data of the third option is almost 100% (Table 3).

Table 4 and 5 show the CNN architecture when the input of the CNN is the matrix 1x27x1 (one time slice) and the matrix 11x27x1 (eleven time slices).

The number of feature extraction layers is the same for both architectures, but an increase in the number of rows of the input data matrix in the second case made it possible to increase the kernel dimension for both convolution and pooling. Both cases are 100% accurate.

Table 2. The number of misclassified samples in different emergency situations for two calculation options.

Lne	First option			Second option		
	The number of errors / relative error (%) when using data of:					
	2 time slices	5 time slices	10 time slices	3 time slices by 2	6 time slices by 2	11 time slices by 2
1	31 / 2,58	8/ 0,27	161/ 2,68	0 / 0	0 / 0	35 / 0,58
2	4 / 0,33	27/ 0,9	62/ 1,03	0/ 0	0 / 0	6 / 0,1
3	1 / 0,08	71/ 0	298/ 4,97	0 / 0	0 / 0	4 / 0,07
4	29/ 2,42	37/ 2,37	78/ 1,3	0 / 0	1 / 0,03	2 / 0,03
5	39 /3,25	72/ 2,4	230/ 3,83	0 / 0	0 / 0	6 / 0,1
6	38 / 3,17	52/ 1,73	385/ 6,42	0 / 0	1 / 0,03	25 / 0,42
7	40 / 3,33	11/ 0,37	35/ 0,58	0 / 0	0 / 0	0 /0
11	15 / 1,25	351/ 11,7	796/ 13,27	44 / 3,67	236 / 7,87	869 / 14,48
12	59 / 4,92	1259/ 41,97	2928/ 48,8	61 / 5,08	783 / 26,1	2650 / 44,17
13	2 / 0,17	313/ 10,43	751/ 12,52	1 / 0,08	60 / 2,0	327 / 5,45
16	33 / 2,75	485/ 16,17	1412/ 23,53	12 / 1,0	119 / 3,97	936 / 15,6
17	7 /0,58	191/ 6,37	401/ 6,68	1 / 0,08	30 / 1,0	165 / 2,75
18	35 / 2,92	966/ 32,2	2073/ 34,55	52 / 4,33	584 / 19,47	1020 / 17,0
19	84 / 7,0	1443/ 48,1	2955/ 49,25	174 / 14,5	1024 / 34,13	2860 / 47,67
20	20 / 1,67	285/ 9,5	863/ 14,38	2 / 0,17	76 / 2,53	581 / 9,68
The number of tested samples for one line						
	1200	3000	6000	1200	3000	6000

Table 3. Calculation results for the third calculation option after outage/turning on of line.

The number of time slices	Dimension of input data sets	Accuracy calculation, (%)
1	1x27x1	100
2	2x27x1	99,92
3	3x27x1	99,66
4	4x27x1	100
5	5x27x1	99,98
6	6x27x1	100
7	7x27x1	99,97
8	8x27x1	99,99
9	9x27x1	100
10	10x27x1	99,99
11	11x27x1	100
12	12x27x1	99,99
13	13x27x1	99,49
14	14x27x1	99,91
15	15x27x1	99,96
16	16x27x1	99,94
17	17x27x1	99,07
18	18x27x1	99,72
19	19x27x1	99,26
20	20x27x1	99,77

Table 4. Structural parameters of the CNN for the matrix of input parameters 1x27x1.

Layers	Operation	Input dimen-sion	Kernel	Output dimen-sion
1st feature extraction layer	Convolu-tion	1x27x1	1x2	1x26x6
	Pooling	1x26x6	1x2	1x13x6
2nd feature extraction layer	Convolu-tion	1x13x6	1x2	1x12x8
	Pooling	1x12x8	1x2	1x6x8
Classifica-tion layer	Fully connected layer	48	-	16

Table 5. Structural parameters of the CNN for the matrix of input parameters 11x27x1.

Layers	Operation	Input dimension	Kerne l	Output dimen-sion
1st feature extraction layer	Convolution	11x27x1	2x2	10x26 x36
	Pooling	10x26x36	2x2	5x13x 36
2nd feature extraction layer	Convolution	5x13x36	2x2	4x12x 14
	Pooling	4x12x14	2x2	2x6x1 4
Classifica-tion layer	Fully connected layer	168	-	16

4 Conclusion

In this paper, a solution for detecting the line status change in a transient states through the convolutional neural network classifier using phasor measurements of bus voltages and line currents in real time is proposed. The important role of the joint use of PMUs and CNN in solving this problem is emphasized. The ability of PMU to record fast transients with high accuracy at the same time provides a large amount of data that can be processed so far only with the help of the latest machine learning algorithms and, in particular, convolutional neural networks.

A high accuracy (up to 100%) in determining the line status was obtained, regardless of the presence of noise in the input data. A change in the network topology is detected at the very beginning of the transient process almost instantly. The operator can identify the line status several times during the first seconds to make sure that the actions are being taken correctly.

Further areas of research include the network topology detection when two or more lines change status simultaneously.

References

1. Cardoso P.E.A. Deep learning applied to PMU data in power systems: Ph.D. thesis. Faculdade De Engenharia Da Universidade Do Porto, 2017, 105 p.
2. Schmidhuber J. Deep Learning in Neural Networks: an Overview // Neural Networks, 2015, Vol. 61, pp. 85–117. doi: 10.1016/j.neunet.2014.09.003.
3. Паттерсон Дж., Гибсон А. Глубокое обучение с точки зрения практика / пер. с анг. М.: ДМК Пресс, 2018, 418 с.
4. Muhammad A., Lee J.M., Hong S.W., Lee S.J., and Lee E.H. Deep learning application in power system with a case study on solar irradiation forecasting // in Proc. Int. Conf. Artif. Intell. Inf. Commun. (ICAIIC), Feb. 2019, pp. 275–279.
5. Li W.T., Deka D., Chertkov M., Wang M. Real-time faulted line localization and PMU placement in power systems through convolutional neural networks // IEEE Transactions on Power Systems, 2019, Vol. 34, No. 6, pp. 4640–4651.
6. Хайкин С. Нейронные сети: полный курс. 2-е изд., испр. / пер. с англ. М: ООО «И.Д.Вильямс», 2006, 1104 с.
7. Созыкин А.В. Обзор методов обучения глубоких нейронных сетей // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика, 2017, Т.6, № 3. с. 28–59.
8. Хохлов М.В., Голуб И.И. Унифицированный подход к оптимизации размещения PMU в сети для обеспечения надежности наблюдаемости ЭЭС // Методические вопросы исследования надежности больших систем энергетики: Вып. 65. Надежность либерализованных систем энергетики / Отв. ред. Н.И. Воропай, А.Н. Назарычев. Иркутск, 2015, с. 591–601.