Highlights

Fast Small Signal Stability Assessment using Deep Convolutional Neural Networks

Tetiana Bogodorova, Denis Osipov, Luigi Vanfretti

- Fast small signal stability assessment on a short data window using deep learning
- A comparative study of convolutional neural network architectures that are adapted for time-series multidimentional data
- A superiority of deep learning in fast desicion making with respect to traditional Prony method
- The study is performed on a large 769-bus system that is currently exploited by the power system operators

Fast Small Signal Stability Assessment using Deep Convolutional Neural Networks

Tetiana Bogodorova[a], Denis Osipov[b], Luigi Vanfretti[a]

[a]*Rensselaer Polytechnic Institute*
Department of Electrical, Computer & Systems Engineering
Troy, NY, USA
[b]*New York Power Authority*
White Plains, NY, USA

Abstract

The paper proposes an approach for fast small signal stability assessment on a short data window using deep learning algorithms. This paper shows that the proposed deep convolutional neural networks (CNNs) are faster than traditional methods (i.e. Prony's method). The evaluated CNNs are fully convolutional network (FCN), CNN with sub-sampling steps performed through max pooling (Time LeNet), time CNN, fully convolutional network with attention mechanism (Encoder), and CNN with a shortcut residual connection (ResNet). The proposed approach is validated on different synthetic measurement data sets generated from the IEEE 9-bus system that is used as a reference, and further applied to a 769-bus system representing a region in the U. S. Eastern Interconnection. We show that precision and recall are more informative metrics than accuracy for the reliability of the stability assessment process using the proposed methodology. In addition, the method is compared to classical Prony method, training time and the classification performance time is evaluated.

Keywords: Convolutional neural networks, deep learning, small signal stability assessment, time-series classification

Email addresses: bogodt2@rpi.edu (Tetiana Bogodorova), Denis.Osipov@nypa.gov (Denis Osipov), vanfrl@rpi.edu (Luigi Vanfretti)

1. Introduction

Applications of Deep Learning (DL), a special class of Machine Learning (ML) techniques that employ multi-layered neural networks (NNs) to extract information from raw data, have recently emerged in power systems analysis. The typical applications of DL are classification of events from Phasor Measurement Unit (PMU) data (1), voltage stability (2), (3), and dynamic security assessment (4), among others.

To maintain the small signal stability (SSS) of a power system, the authors in (5) address the problem of parameter optimization for power system stabilizers (PSSs). The proposed approach employs a NN with Levenberg-Marquardt optimization, showing good performance. Another example of the successful applications of NNs is presented in (6), where optimal PSS parameters are determined for PSSs that do not provide enough damping for the system. Meanwhile, the work of (7) explores optimal and robust power system stabilizer design for multi-machine power systems. An heuristic search optimization that is known as a cuckoo search optimization is proposed to obtain PSS parameters that ensure SSS of a particular power system. Since heuristic algorithms empirically showed a convergence close to the global optimum, the work of (8) proposed to use such algorithms for coordinated tuning of PSS parameters too. In (9) the authors propose to employ the decision trees at each generation bus to evaluate the interarea oscillations damping. The method is of low computational burden and robust to loss of information.

As it can be noticed, most of the methods in the literature are concerned with the improvement of the SSS of a system after such instability has been detected. However, limited attention (10) has been paid to the automation of fast instability detection using a time window shorter than classical methods (i.e. Prony method (11), (12)). In contrast to the approach proposed in this work, other methods require meticulous and cumbersome data preparation for training to ensure good performance of the proposed machine learning algorithms. In this paper the deep convolutional neural networks with different architectures are studied to solve the small signal stability assessment (SSSA) task. These algorithms allow for extremely fast decision making after being adequately trained (13). In addition, the proposed approach demonstrates good performance under relatively short input data lengths and can perform classification online.

Because deep learning is a novel research area, the applications for power

system problems are limited, but are expected to grow (14). Another reason for the limited application is the requirement of careful data generation for training, which is a challenging task in the case of working with large power systems (15), (16). First, this task demands considerable computational power. Second, substantial effort is needed to design of numerical experiments together with simulation data cleaning and curating before training of the deep learning algorithms takes place. If irrelevant or invalid data is preserved in the training data set, the trained deep learning model may learn inappropriate rules and patterns in the data, that corrupt the performance of the algorithm.

In this paper we propose an approach for SSSA using novel deep convolutional neural network architectures adapted for time series data classification, employing data preparation techniques for training, and validating the approach on the large 769-bus system. In this approach, the classes are the system conditions, i.e. stable or unstable. Thus, the main contributions of this paper are:

1. To propose a novel fast approach employing FCN (17), Time LeNet (18), Time CNN (19), Encoder (20), ResNet (17) for SSSA. The proposed system extracts patterns in power system data that include both spatial and temporal dependencies that are used for SSSA, which is a novel approach for SSS analysis. We show that CNN is capable to perform faster than traditional Prony.

2. To show the importance of using precision and recall metrics (illustrated through case studies), how to contrast these metrics and to explain how they influence the trained model's evaluation.

3. To provide a comparison of the performance of trained models that use different signal types such as voltage phasor magnitude, voltage phasor angle, with and without Gaussian noise.

4. To evaluate how the length of the input signal influences model performance for each deep neural network studied, and compare it with the classical Prony method.

The reminder of the paper is organized as follows. Section 2 introduces the proposed method. Section 3 elaborates each of five algorithms applied for the small signal assessment task. Section 4 presents the assessment of the proposed method. Finally, Section 5 concludes this work.

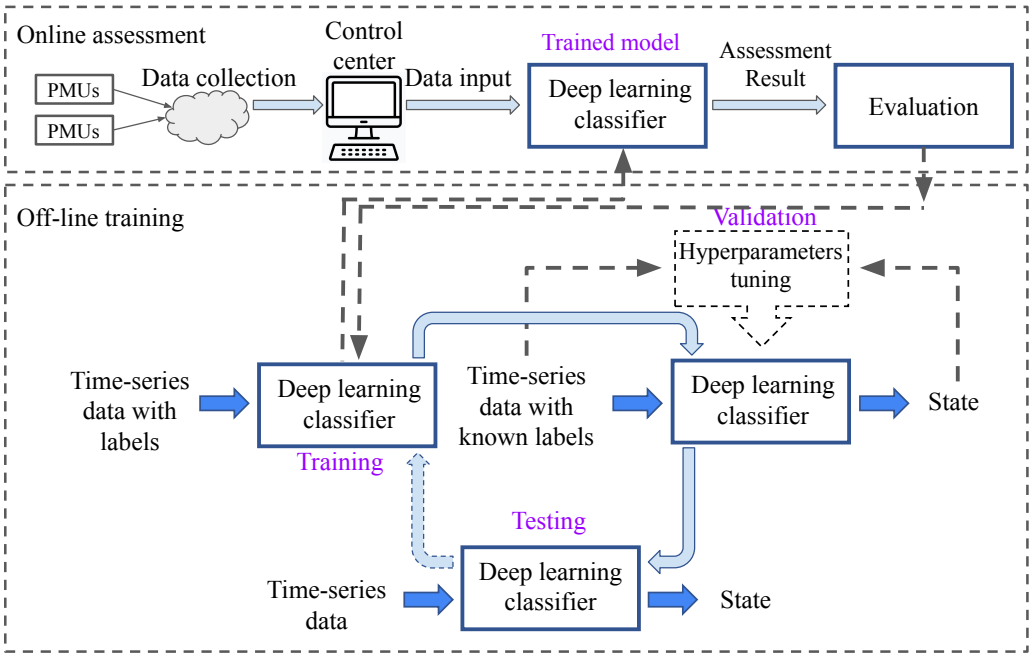

Figure 1: Overview of the proposed method.

2. Proposed Method

2.1. Overview of the framework

The proposed method consists of two major processes: off-line learning and online assessment. The former consists of three phases: training, validation and testing, shown in Figure 1. In the training phase, the ML model learns from labeled data by tuning its parameters. The model training process involves iterations where the output of the algorithm and the true label are compared using a performance metric, e.g. accuracy. The validation phase includes tuning the hyperparameters of the classifier (internal parameters of the chosen CNN architecture). The testing phase involves exposing the trained CNN to unseen sets of data to verify if it has been trained and validated correctly. Once the model is trained, the online assessment process exploits the trained CNN model for SSS assessment. This process includes the deployment of the trained model at the control center with the online collection of the measurements and evaluation of CNN's output.

2.2. Data Preprocessing

The following data preprocessing steps are performed for the voltage angle signal at each generator bus: a) subtraction of the center of angle that is defined as the inertia weighted average of all rotor angles (21); b) unwrapping; c) subtracting the initial value to obtain a deviation signal; d) detrending. Let us present the input data as $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T]$, where T is the number of time-series signals. For the voltage magnitude of positive sequence data, the deviations of the voltage signals are presented as follows:

$$\mathbf{x}_t = [V_{1,t} - V_{i,0}, V_{2,t} - V_{i,0}, ..., V_{i,t} - V_{i,0}, V_{N,t} - V_{i,0}] \tag{1}$$

where N is the total no. of buses, $V_{i,t}$ is the voltage magnitude at bus i of length t, and $V_{i,0}$ the initial value of voltage for each bus i. In addition, to test the ability of the deep learning algorithms to learn from noisy data, 1% Gaussian noise has been added to the simulated signals that are used as pseudo-measurements in this work.

In the case of the voltage angle, the data is presented as a set of vectors of $\mathbf{x} = [\angle\theta_{1,t}, \angle\theta_{2,t}, ..., \angle\theta_{N,t}]$. Then, then angle unwraping is performed by computing:

$$\angle\theta_{j,i} = \angle\theta_{j,i} + (2\pi k) \text{ if } (\angle\theta_{j,i} - \angle\theta_{j,i-1}) \geq \pi \tag{2}$$

where j is the sample number in the dataset, i the identifier of a measurement at a particular moment in time, and k is updated after every large jump in the phase value (22).

It is assumed that measurements are made at key system locations where PMU are installed or simulated using the equivalent system model: generator terminal buses at major power plants, major transmission level substations and boundary buses of tie-lines between the study and neighbouring systems.

2.3. Offline Training

The collected data is divided into 3 parcels: (1) training, validation and testing data (see Section 2). The split is usually done with $\frac{2}{3}$ of data used for training, and $\frac{1}{3}$ of data used for testing. In case when there is a lack of data, k-fold cross validation is employed to generate a validation set of data, where k defines a number of groups the data is divided into, meaning that k-th part of the data will be left for validation of the trained model. Thus, the model is trained on $k-1$ folds of data, and evaluated on the k-th subset of data.

If the training set is presented as a collection of values $\{x_{(n)}, y_{(n)}\}_{n=1}^{N}$, the objective of the training is to find the parameters of a model (e.g. $\mathbf{W}$, $\mathbf{b}$ in equations (7)) when minimizing the categorical cross entropy error function L_{CE}, i.e.,

$$L_{CE} = \min \sum_{n=1}^{N} \sum_{c=1}^{C} y_{c,(n)} log(\hat{y}_{c,(n)}) \tag{3}$$

where $\hat{y}_{(n)}$ is the classification result of the input values $x_{(n)}$ for the trained model, C the number of classes, and N the number of training cases.

The classification is performed according to:

$$y_{(n)} = \begin{cases} 0 \text{ (Stable)}, & \text{if } \forall \zeta_i > 3\% \\ 1 \text{ (Unstable)}, & \text{if } \exists \zeta_i \leq 3\% \end{cases} \tag{4}$$

where ζ_i is the damping ratio of the i-th oscillatory mode, with $i = 1 \dots m$ for a power system with m modes.

2.4. Online Assessment

For small signal assessment, the online assessment is performed when the trained model is employed to classify the state of the system using measurements as input (e.g., PMU data) that contains pre-fault measurements, the measurements during the contingency itself, and post-contingency measurements. The set of measurements - voltage magnitude or voltage angle of the length T, the output h is evaluated and passed through the softmax function to make a decision on the $\hat{y} \in (0, 1)$ for each class. The parameter of the softmax function $\delta = 0.5$ defines the boundary between the classes. Thus, the prediction of the class is made in favour of the $\hat{y} > \delta$.

2.5. Evaluation Metrics

The training of the CNNs has been performed using accuracy metric, while precision and recall were measured on testing data.

Accuracy. This metric defines the general performance of the algorithm over all classes.

$$\text{accuracy} = \frac{N_{TP} + N_{TN}}{N_{TP} + N_{FP} + N_{TN} + N_{FN}} \tag{5}$$

where N_{TP} is the total of unstable cases (positive class corresponds to unstable labeling) correctly classified as unstable; N_{TN} is the total of stable

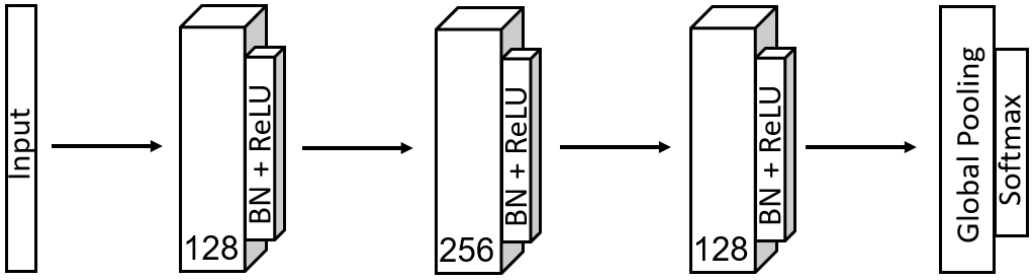

Figure 2: Fully convolutional neural network architecture

cases (negative class by choice of the authors) correctly classified; N_{FP} is the number of stable cases misclassified as unstable; and N_{FN} is the number of unstable cases misclassified as stable.

Precision and Recall. Precision relates accuracy of the model in classifying the data as positive sample. Recall evaluates the number of correct positive predictions over all positive predictions that are relevant.

$$\text{precision} = \frac{N_{TP}}{N_{TP} + N_{FP}};\ \text{recall} = \frac{N_{TP}}{N_{TP} + N_{FN}} \tag{6}$$

3. Deep Convolutional Neural Networks

In this section, the Deep CNN architectures that are used for time-series data classification are presented.

3.1. Fully convolutional neural network (FCN)

The architecture of FCN (17) is presented in Fig. 2. The basic convolution block can be described by equations (7):

$$\begin{aligned} y &= \mathbf{W} \otimes \mathbf{x} + \mathbf{b} \\ s &= BN(y) \\ h &= ReLU(s) \end{aligned} \tag{7}$$

where $\otimes$ - the convolution operator, BN is the batch normalization function, $ReLU$ is the rectified linear unit function.

Meanwhile, the objective function is a categorical cross entropy to be minimized. The Adam optimizer is a stochastic gradient descent method.

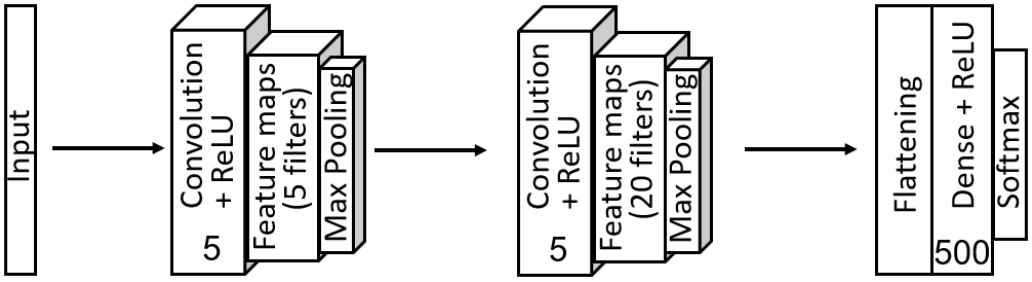

Figure 3: Time-LeNet architecture

3.2. Time LeNet

Time-LeNet architecture (19) (Fig. 3) includes two convolutional hidden layers that are followed by a pooling layer, a feature layer, and an output layer. The Time-LeNet NN is described by

$$\begin{aligned} y_r(t) &= ReLU(\mathbf{W}_r \otimes \mathbf{x}(t) + \mathbf{b}_r) \\ s_r(t) &= MP(y_r((t-1)l+1), y_r((t-1)l+2), ..., y_r(tl)) \\ h &= ReLU(\mathbf{Z} * \mathbf{s} + \mathbf{b}) \end{aligned} \tag{8}$$

where $*$ denotes dot product, MP denotes max pooling operation, r is the number of filters (feature maps), l is the length of a convolution window.

The objective function is a loss function that is defined as categorical cross entropy to be minimized. The optimizer is Adam (see Section 3.1) with learning rate equal to 0.01, and decay equal to 0.005.

3.3. Time-CNN

Time-CNN (19) shown in Fig. 4 is a traditional deep CNN, but with the output consisting of a fully convolutional layer and using a local average pooling operation instead of max pooling operation in the Time-LeNet architecture (Fig. 8). The Time-CNN can be described by

$$\begin{aligned} y_r(t) &= S(\mathbf{W}_r \otimes \mathbf{x}(t) + \mathbf{b}_r) \\ s_r(t) &= AP(y_r((t-1)l+1), y_r((t-1)l+2), ..., y_r(tl)) \end{aligned} \tag{9}$$

where S is the sigmoid activation function, AP is the average pooling operation. The objective function is a mean squared error to be minimized. The optimizer is Adam with learning rate equal to 0.001.

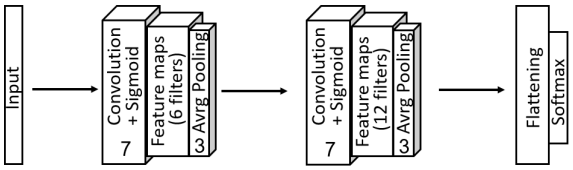

Figure 4: Time CNN neural network architecture

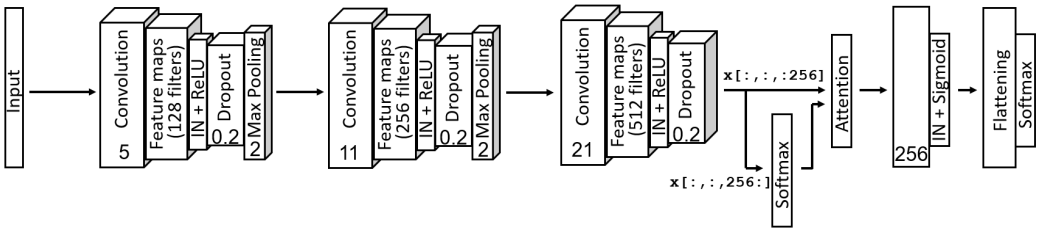

Figure 5: Encoder neural network architecture

3.4. Encoder

The Encoder (Fig. 5) (20) is a hybrid deep CNN architecture that is similar to FCN with the difference in one layer: the GAP layer is replaced with an attention layer. This architecture is larger in number of feature maps in each convolutional layer than Time-LeNet or Time-CNN. The operations within the hidden layers are presented in the set of the equations (10).

$$\begin{aligned} h_1 = Conv_{k_1}(x);\ h_2 = Conv_{k_2}(h_1) \\ x = Conv_{k_3}(h_2) \\ y = ATN(x[:,:,:256] * S(x[:,:,256:])) \\ z = \mathbf{W} \otimes \mathbf{y} + \mathbf{b};\ s = IN(z);\ u = S(s) \end{aligned} \tag{10}$$

where IN is an instance normalization operation, k_1, k_2, k_3 are the number of output filters equal to 128, 256, and 512, respectively; ATN is the attention mechanism, and S is the softmax function.

The attention $ATN(\cdot)$, is implemented by splitting the data in equal parcels. Softmax function is applied to one parcel, and then two parcels are multiplied. Thus, each element of the softmax-treated parcel serves as a weight for another one. This mechanism enables the model to learn which parts of the time series are important for classification.

The objective function is a loss function that is defined as categorical cross entropy to be minimized. The optimizer is Adam with the learning rate equal to 1×10^{-6}.

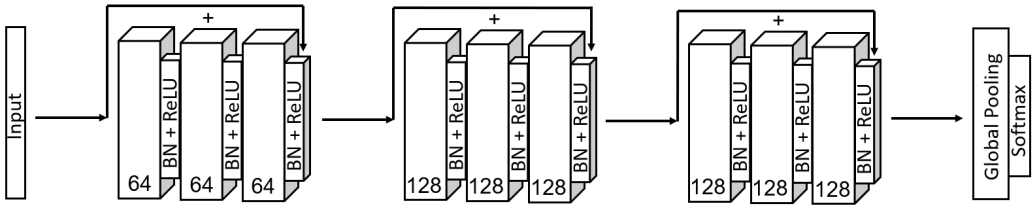

Figure 6: ResNet neural network architecture

3.5. Residual Neural Network (ResNet)

Finally, the last NN architecture used in this work is the Residual Neural Network (ResNet) (17), which is a deep NN with 11 layers among each 9 layers that are convolutional (see Fig. 6). The main characteristic of this type of network is the shortcut residual connection between consecutive convolutional layers. Thus, each colvolutional layer characterized by the number of filters k, as shown by

$$\begin{aligned} h_1 = Conv_{k_1}(x);\ h_2 = Conv_{k_2}(h_1);\ h_3 = Conv_{k_3}(h_2) \\ y = h_3 + x \\ \hat{h} = ReLU(y) \end{aligned} \tag{11}$$

where $k_1 = 64$, $k_2 = 128$, $k_3 = 128$ is the number of filters in each layer. The final layer includes both global average pooling and softmax operations.

4. Case Studies and Analysis

The following case studies were conducted to employ the proposed CNNs of different architectures and show their superiority on shorter data window than required for Prony algorithm. To determine if the deep CNNs are capable to perform well, several approaches to prepare training and testing data were used. In order to limit the possible space of states that define the dynamics to be learnt and, therefore, to verify the capability of each deep CNNs architecture for SSSA, the IEEE 9-bus system is chosen for proof-of-concept. Using the IEEE 9-bus system, seven case studies were developed as shown in Table 1.

The labeling of the data is done using modal analysis of the power system (see Section 2.3). The simulation data was obtained using the approach for realistic contingency scenario generation in (16). In addition, the shifting

Table 1: Case Study Setup. IEEE 9-bus system

Case Study	Data	Preprocessing	Noise	Prefault interval
1	$\angle\theta$	Yes	No	1 sec
2	$\angle\theta$	Yes	1% Gaussian	1 sec
3	V	No	No	No
4	V	No	No	0.5 sec
5	V	No	No	1 sec
6	V	Yes	No	1 sec
7	V	Yes	1% Gaussian	1 sec

where $\angle\theta$ - voltage angle, V - voltage magnitude. For all the cases the data are generated using (16),the data length is 10 sec.

of the measurements window to include the pre-contingency interval is performed to observe if the data during steady state operation influences the deep neural networks' training.

The number of cases of stable and unstable scenarios were balanced to allow the deep CNNs to learn both classes with the same importance. The evaluation metrics are accuracy, precision, and recall. The two latter metrics are not typical for the evaluation, however, we show that in some cases they are decisive while accuracy give incomplete information.

After analyzing the performance achieved by the CNNs for the IEEE 9-bus system, the measurements of the particular type that correspond to the best performance are selected for the further training and tuning using a large-scale power system model, i.e. a 769-bus system of a region in the U.S. Eastern Interconnection. For this system the time-series data was labeled by performing mode estimation using Prony method (11) to replicate scenarios when the model of the system is not available. However, the data window for the proposed CNNs approach for SSSA is much shorter than for Prony analysis.

4.1. Proof of Concept: Training on voltage angle data from the IEEE 9-bus system

The data reporting rate is 60 times per second. The training is done on input data of size 4,498 data parcels (corresponding to the same number of different nonlinear simulations, i.e. 4,498), each of a 10 sec length. The experiment settings are presented in Table 1, the results of the training for each case study are summarized in Table 2.

Table 2: Performance evaluation of Deep Convolutional NNs. IEEE 9-bus system

Case Study	Metrics	FCN	Time-LeNet	Time-CNN	Encoder	ResNet
1	Accuracy/ Precision/ Recall	**0.993 / 0.991 / 0.977**	**0.993 / 0.991 / 0.977**	0.989 / 0.992 / 0.971	0.990 / 0.992 / 0.973	**0.992 / 0.994 / 0.978**
2		**0.992 / 0.994 / 0.978**	**0.992 / 0.994 / 0.978**	0.990 / 0.992 / 0.975	0.990 / 0.991 / 0.973	0.990 / 0.988 / 0.977
3		0.778 / 0.781 / 0.815	0.989 / 0.992 / 0.971	0.986 / 0.989 / 0.963	0.989 / 0.992 / 0.972	**0.991 / 0.991 / 0.979**
4		**0.989 / 0.992 / 0.971**	**0.989 / 0.992 / 0.971**	0.986 / 0.989 / 0.963	0.989 / 0.992 / 0.972	0.450 / 0.390 / 0.576
5		**0.989 / 0.992 / 0.971**	0.427 / 0.142 / 0.333	0.986 / 0.989 / 0.963	**0.989 / 0.992 / 0.972**	0.988 / 0.991 / 0.970
6		**0.989 / 0.992 / 0.971**	0.989 / 0.992 / 0.972	0.987 / 0.989 / 0.963	**0.989 / 0.991 / 0.972**	**0.989 / 0.992 / 0.971**
7		0.985 / 0.984 / 0.965	0.987 / 0.986 / 0.970	0.985 / 0.989 / 0.963	**0.989 / 0.992 / 0.972**	0.986 / 0.986 / 0.967

4.1.1. Case Study 1: Voltage angle data without noise, with a pre-contingency interval

In this case study the generated data does not contain noise, but includes a steady state interval of 1 *sec* before the contingency occurs. The results of training the algorithms are marked as Case study 1 in Table 2.

For all the CNN models recall is lower than accuracy or precision. Lower recall than precision means that the number of false negative examples (unstable cases misclassified as stable) is bigger than the number of false positives (stable cases misclassified as unstable). Thus, we consider that the better CNN model is the one with higher recall, meaning less misclassified unstable cases. For the system operator the minimization (preferably zero) of the misclassified unstable cases is crucial to trust the classifier's output.To summarize, the best training results considering all applied metrics are achieved utilizing FCN, ResNet NNs, and Time-LeNet.

4.1.2. Case Study 2: Voltage angle data with 1% Gaussian noise, and with a pre-contingency interval

A typical value of 1% Gaussian noise has been added to the voltage angle signal. The results of this case study, shown in Table 2, suggest that the best performing deep NNs among tested are FCN and Time Le-Net.

The difference between FCN and Time Le-Net is that FCN has a GAP layer with average pooling, while Time Le-Net has max pooling after each convolutional layer. Nevertheless, both networks showed good performance in this case study.

4.2. Training on voltage magnitude data from the IEEE 9-bus system

In this case study the aim is to compare the CNNs trained using voltage magnitudes measured at the generator buses in IEEE 9-bus system.

4.2.1. Case Study 3: Voltage magnitude data without noise, after the contingency

In this case study the data has been collected right after the contingency starts, for a duration of 10 sec.

Following the results in Table 2 the performance of the convolutional architectures, especially of FCN, has decreased in comparison to the results of Case Study 1 and 2. ResNet showed the best performance.

Table 3: Case Study 7: Training on voltage amplitude data with 1% Gaussian noise

Algorithm	Train loss	Valid. loss	Train accuracy/ valid. accuracy	N. of epoch
FCN	0.000	1.496	1/0.985	1999
Time-LeNet	0.064	0.075	0.987/0.985	5
Time-CNN	0.007	0.010	0.987/0.985	1987
Encoder	0.000	0.134	1/0.989	85
ResNet	0.000	0.208	1/0.986	1477

4.2.2. Case Study 4: Voltage magnitude data without noise, with pre-contingency interval

In this case study the data of steady state operation ($0.5sec$) before a contingency with the data ($9.5sec$) after the contingency is used for training. In Table 2 Case 4 results show performance improvements for the FCN with respect to Case 3, while Time-CNN, Time Le-Net and Encoder demonstrated better robustness towards the data change. ResNet showed the worse performance. This can be explained by the special architecture of ResNet that allows the NN to remember longer the past patterns in the data. In this case this memory did not allow convergence to a better solution. To conclude, the results of Table 2 suggest that Encoder is the best choice for training on the data in this case study.

To verify these conclusions, additional data corresponding to the steady state regime has been added to the data sets. Thus, the simulations start 1 sec. before the contingency is applied.

4.2.3. Case Study 5: Voltage magnitude data without noise, with enlarged pre-contingency interval

In this case (see Tables 2) the Encoder shows the best performance. In addition, ResNet and FCN showed good performance as well. ResNet has improved its performance with respect to the previous case study. In contrast, Time Le-Net demonstrates the worst performance among all the CNNs.

The results showed that the best performance of all deep NNs is achieved when using data from the preprocessed angles deviations (see Section 2.2

for data preprocessing details). Therefore, considering the effectiveness of the data preprocessing, the next case study is designed using the voltage magnitude deviations measured on the generator buses in the IEEE 9-bus system.

4.3. Case Study 6. Voltage magnitude deviation data

Results for this case study are shown in Table 2, were an improvement (w.r.t. Case 5) of performance of all the NNs can be observed. We can conclude that the deviations allow the models to focus more on the system dynamics, which are masked when evaluating the raw magnitude values. FCN and ResNet performed equally good.

To consider training on data that is closer to "real-world" measurements, 1% Gaussian noise is added to the voltage magnitude deviations in the next case study.

4.4. Case Study 7. Voltage magnitude deviations data with 1% Gaussian noise

In this study, when the 1% Gaussian noise has been added to the measurements, one can observe an example of overfitting of the trained NNs. If we judge on the results in Table 2, we can draw a conclusion that the best NN architecture for this case is the Encoder. However, as Table 3 clearly shows, the FCN, Encoder and ResNet experience overfitting when exposed to noisy data. In other words, the trained NNs remembered noise as useful dynamics, which is unwanted behaviour.

In contrast, using the results of Table 3 together with Table 2, it can be observed that Time Le-Net and Time-CNN show good performance and did not overfit. Therefore, one of these architectures has to be chosen for implementation when considering noisy data. Otherwise, the extra tuning of NN is required to avoid overfiting. In addition, comparing Time Le-Net and Time-CNN using the computational efficiency metric, Time Le-Net is superior in the training time showing the fastest convergence (in 5 epochs) to the best model.

4.5. Training on voltage angle deviations data of the 769-bus system

This case study has been developed to ensure that the proposed methodology is valid for a large-scale power system.

4.5.1. Voltage angle deviations data preparation

Considering the results from Case Studies 1-7, the best performance of the deep CNN algorithms has been achieved on preprocessed voltage angle deviations data. Therefore, this case study has been designed using the same kind of data as in Section 4.1. These synthetic measurements are collected locally at the terminal bus of the largest generator in the 769-bus system. As mentioned earlier, the data has been generated using the realistic contingency generation algorithm (16). The labeling of the generated data was performed using Prony's algorithm starting from the 2 seconds mark of the simulation until 10 seconds, having a 8-second window. Since Prony is a simple estimation algorithm, the estimation problem is constrained to fit the pole that is closest to 0.8 Hz which is a frequency of a typical inter-area mode in the 769-bus system (23). Within the generated simulations, there is a lack of cases with negative damping. Consequently, for this case study, we consider two classes: stable and unstable (when the damping is less than 3%). The data contains $7,878$ time-series trajectories in total. The data reporting rate is 60 samples per second. It has been divided to $\frac{2}{3}$ of data for training, and $\frac{1}{3}$ of data for testing. The validation procedure has been performed using a k-fold cross validation algorithm.

4.5.2. Voltage angle deviations data with and without 1% *Gaussian noise*

The dependency of the performance metrics on presence of noise in the data is more prominent in precision and recall change, when accuracy remains very close to the value trained without noise. The significant difference between the accuracy of training with and without noise data is observed in Encoder (around 5%) without dependency on the batch size in Fig. 7 and Fig. 8, and ResNet when the batch size is 128 in Fig. 7, less prominent (range of $2-4\%$) for every CNN when the data length is 6.67 sec. in Fig. 9.

When recall drops significantly for FCN (around 14%) the batch size increased to 128, for ResNet (around 26%), and it improves for the Encoder (up to 10%) with the presence of noise. This improvement is interpreted as the decrease in misclassification of unstable scenarios with respect to truly unstable cases. Therefore, the architecture of Encoder with the attention mechanism allows to classify unstable scenarios better in the presence of noise when the data interval is short. The best performance on the short data length is received using the Encoder and Time-LeNet architectures. The latter uses max pooling (propagating maximum out of the region where the operation is applied) after each convolutional layer.

4.5.3. Batch size tuning

The batch size defines how often the NNs parameters will be updated during the training. Larger batch size speeds up the training, but the accuracy of the trained NN may suffer due to rare updates. According to the results in Fig. 7, 8, and 9 the optimal batch size is 64 when considering the improvement of the performance metrics. The Encoder has shown good robustness of the precision towards the noise. However, when the training data length is shorter, the precision of the Encoder and Time-LeNet drops with the increase of the batch size. Thus, if the goal is to utilize shorter input data (decreasing the data collection window), and therefore, the total processing time for the whole method when performing the online assessment, the batch size has to be limited to 64.

4.5.4. Dependency of performance metrics on the data length. Comparison with Prony method

Considering the conclusion on the optimal batch size value, in Fig. 10 the dependency of the metrics on the training data length for the fixed batch size is shown. The best performance is shown by the Encoder and Time-LeNet, while there is a decrease in metrics for the data length of 200 samples, the difference is metrics is negligible for the length of 300 and 400 samples. In addition, the classic SSSA approach - Prony method - is 10% less accurate than the proposed methodology. The ability of the CNN perform fast and accurate classification is explained by the meticulous training phase.

4.5.5. Performance Time. Training and Online Assessment

In Fig.11 the benefit of Time-LeNet is that of offering the fastest online assessment time performance, however, this comes at the cost of requiring substantial offline training time to be allocated. The opposite case is shown by the Encoder, the online assessment time is larger but still small (less than 1.8×10^{-3} sec.), while the training time is the smallest among the considered CNNs.

4.5.6. Analysis of acceptable level of missclassification

In Fig.12 the input data for qualitative analysis is shown. These help to understand the difference in misclassified unstable cases that were overlooked by the Encoder, but correctly classified by Time-LeNet. It worth noticing that among the cases that where misclassified by the Encoder, only one is obviously unstable. The potential explanation for this misclassification can

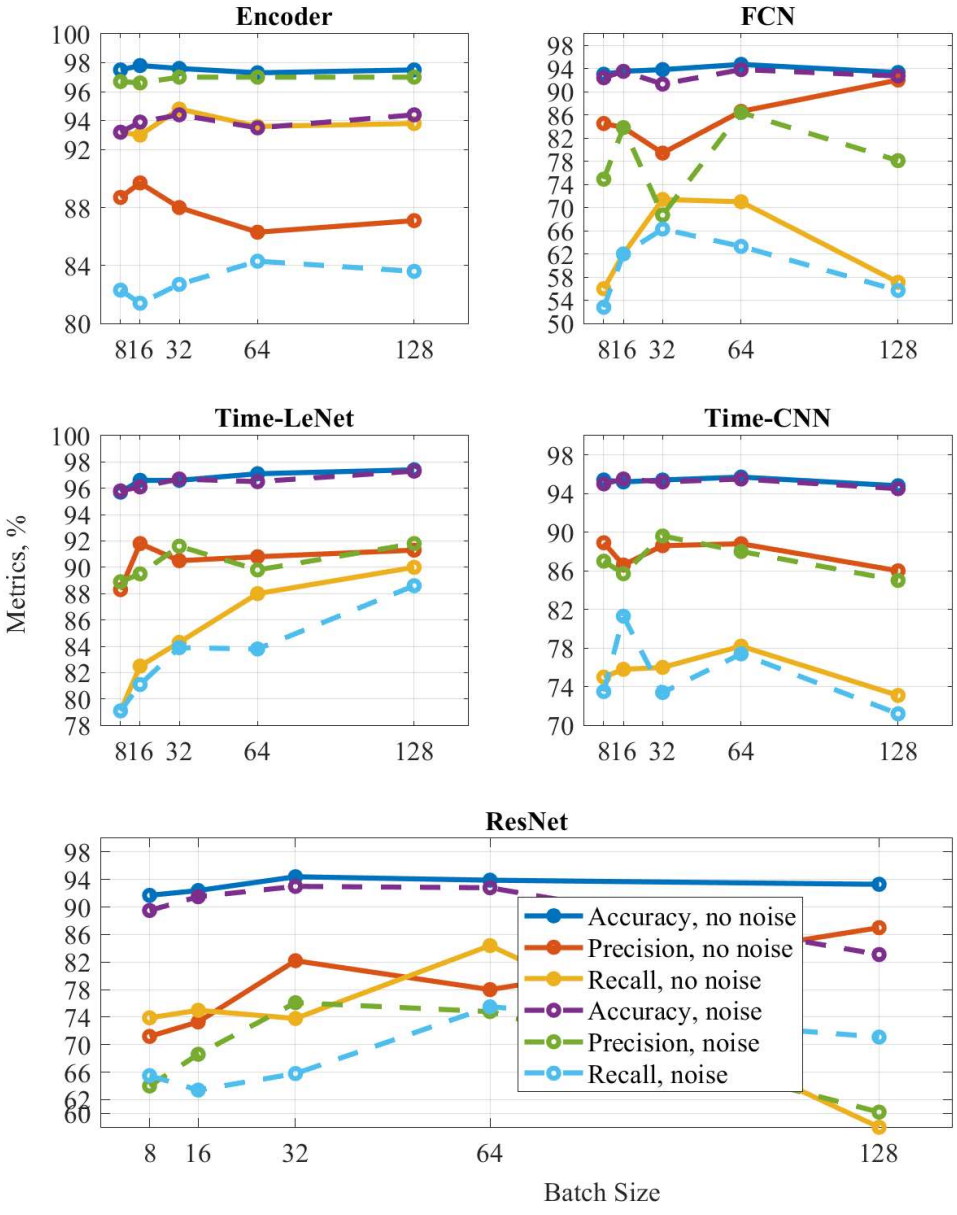

Figure 7: Dependency on the batch size. Length=200 samples (3.33 sec)

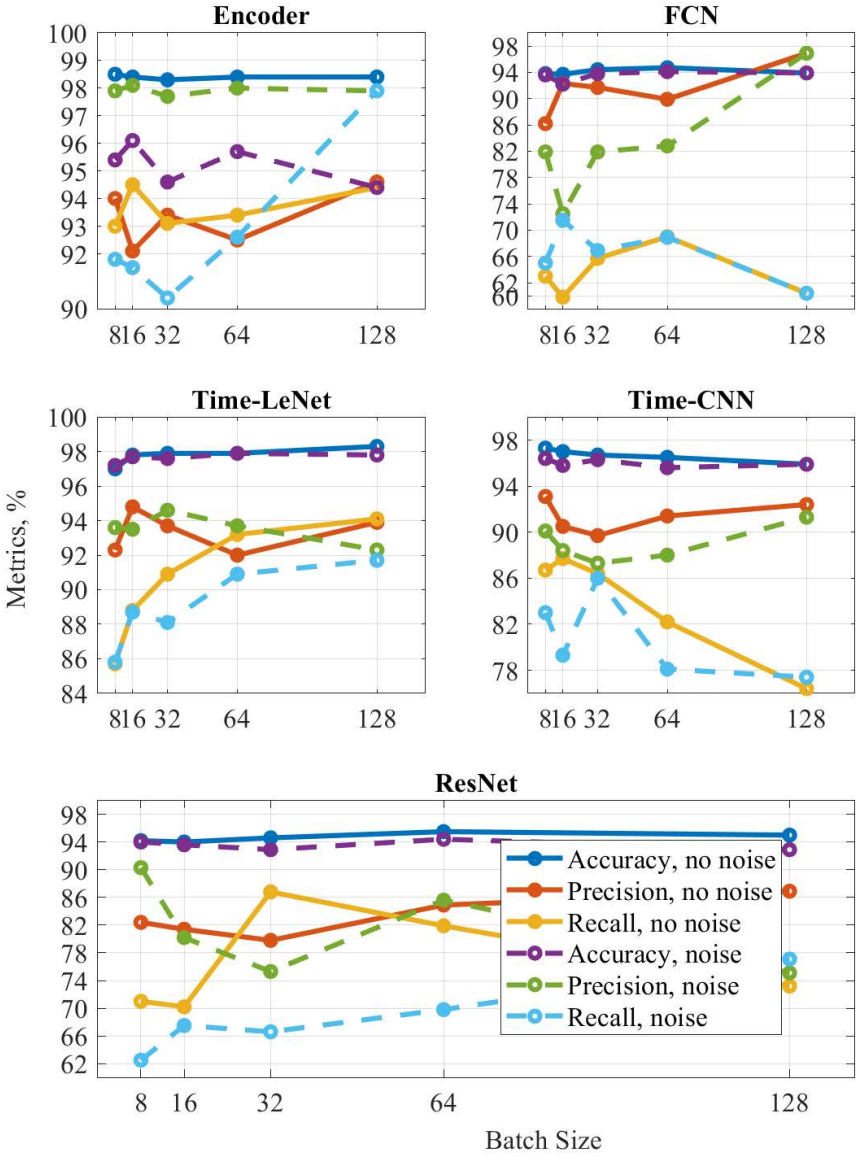

Figure 8: Dependency on the batch size. Length=300 samples (5 sec)

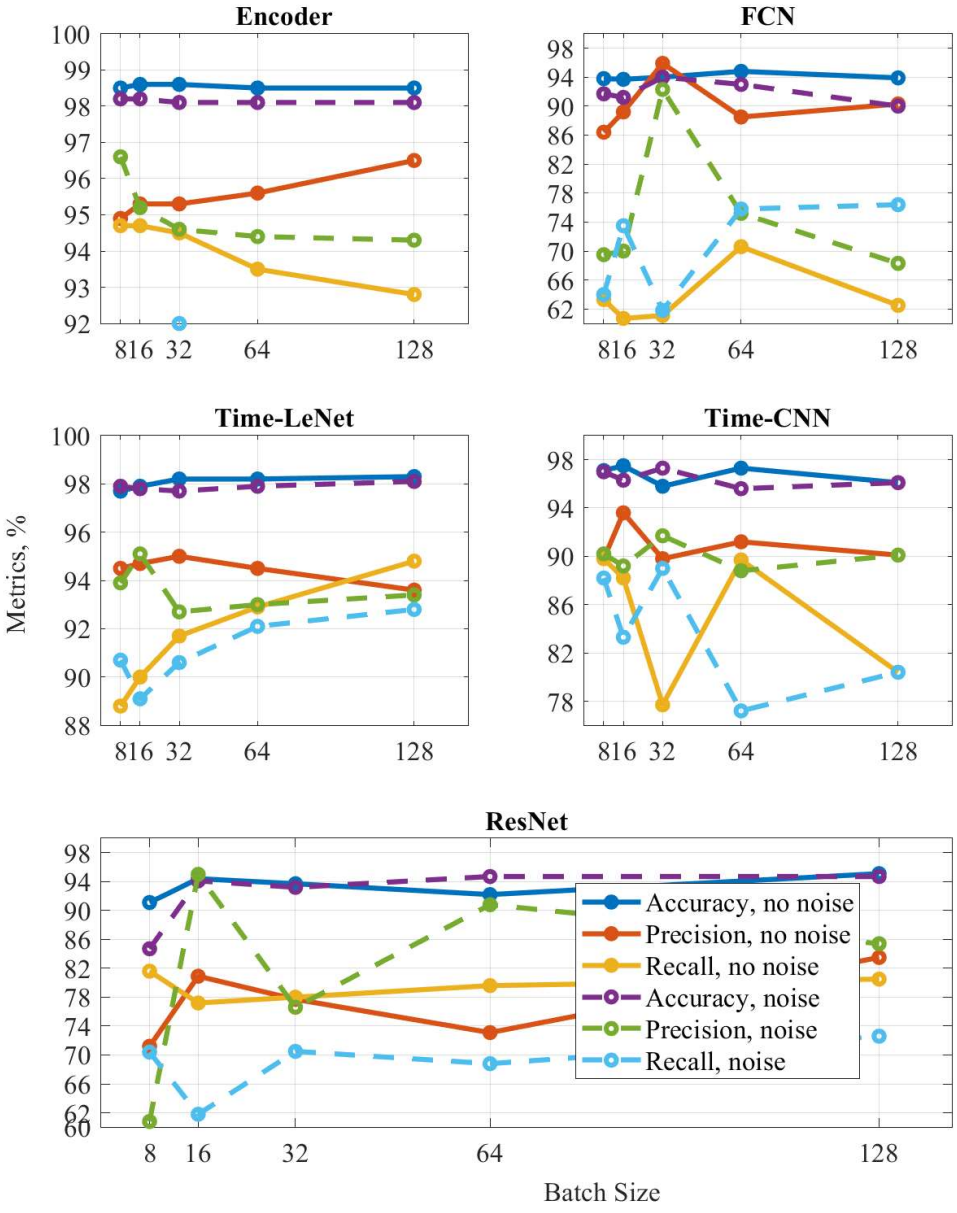

Figure 9: Dependency on the batch size. Length=400 samples (6.67 sec)

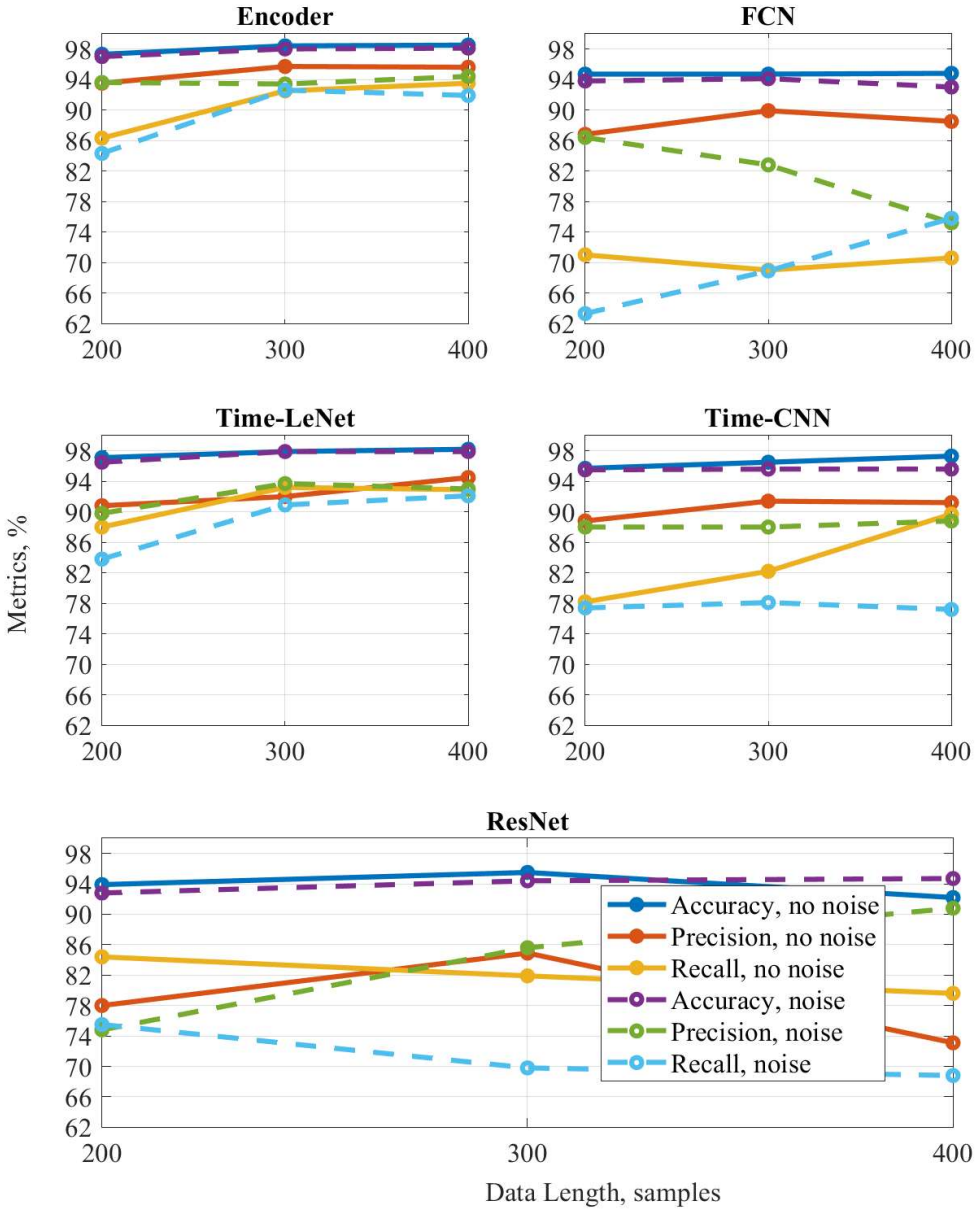

Figure 10: Dependency of metrics on the length of data. Batch size=64. Accuracy of Prony method: for 200 samples - 80.54%, 300 samples - 88.99%, 400 samples - 92.69%

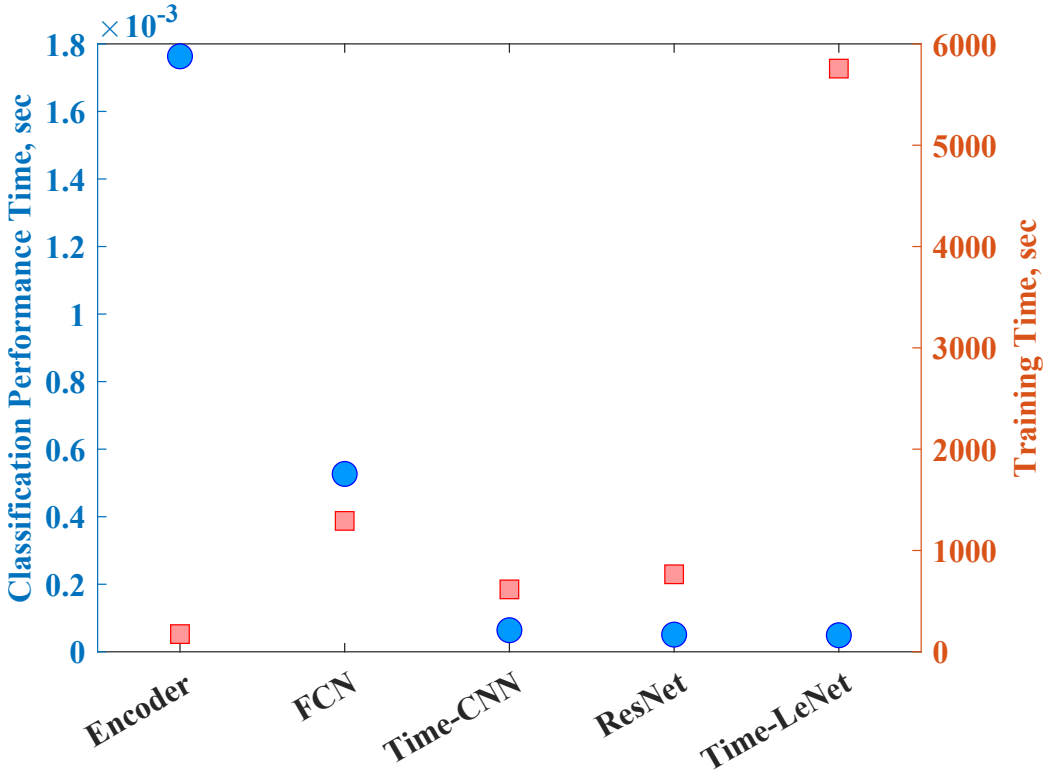

Figure 11: Performance time. Online assessment and training

be the lack of similar cases in the training set, as the majority of cases are marginally stable. The case with indx=2570 in Fig. 12 has prominent unstable upper envelope of the voltage angle signal and marginally stable behaviour for the lower envelope of the signal.

5. Conclusion

In this paper we propose a novel method to perform small signal stability assessment of a power system using deep convolutional neural networks to learn the oscillatory pattern of power system dynamics by capturing time and spatial characteristics in the measurements. The main advantage of the proposed approach is the ability of performing substantially faster than classical algorithms on the shorter data window with substantially higher accuracy (10%) with online execution times as low as 1.8×10^{-3} sec. by exploiting the trained CNNs.

To validate the method, the synthetic realistic measurements were generated using the IEEE 9-bus system and the 769-bus system. Among all data types used for the validation of the CNNs performance, the best performance was achieved for the preprocessed voltage angle differences. Time-LeNet and Encoder have shown the best performance in terms of performance metrics such as accuracy, precision, and recall. While accuracy is less susceptible to the changes made in each study, precision and recall shown how sensitive the method is towards unstable or marginally stable conditions.

Acknowledgement

This research was supported by the New York State Energy Research and Development Authority though Grant No. A50626 and National Science Foundation, Grant No. 2231677.

This paper was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring

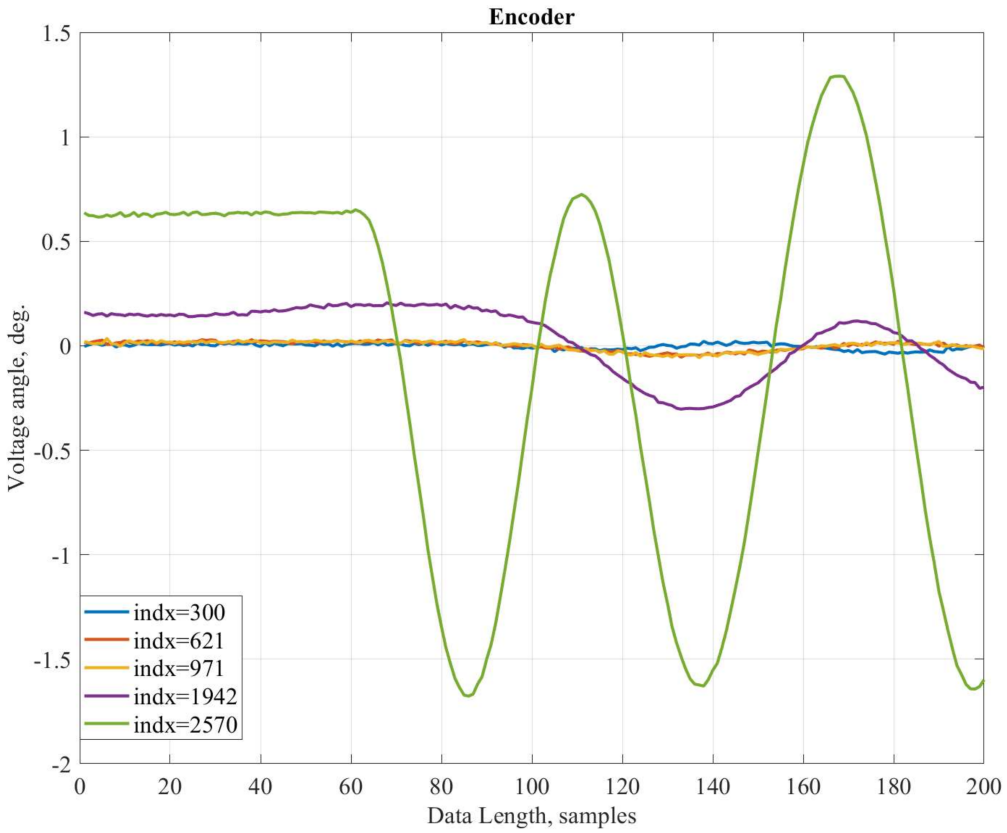

Figure 12: Difference between missclassified data by Encoder and Time-LeNet. Batch = 64

by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

References

[1] R. Huang, Y. Li, False phasor data detection under time synchronization attacks: A neural network approach, IEEE Transactions on Smart Grid 13 (6) (2022) 4828–4836. doi:10.1109/TSG.2022.3179277.

[2] W. Huang, W. Zheng, D. J. Hill, Distribution network reconfiguration for short-term voltage stability enhancement: An efficient deep learning approach, IEEE Transactions on Smart Grid 12 (6) (2021) 5385–5395. doi:10.1109/TSG.2021.3097330.

[3] M. Zhang, J. Li, Y. Li, R. Xu, Deep learning for short-term voltage stability assessment of power systems, IEEE Access 9 (2021) 29711–29718. doi:10.1109/ACCESS.2021.3057659.

[4] J. J. Q. Yu, D. J. Hill, A. Y. S. Lam, J. Gu, V. O. K. Li, Intelligent time-adaptive transient stability assessment system, IEEE Trans. Power Systems 33 (1) (2017) 1049–1058. doi:10.1109/TPWRS.2017.2707501.

[5] M. J. Rana, M. S. Shahriar, M. Shafiullah, Levenberg–Marquardt neural network to estimate UPFC-coordinated PSS parameters to enhance power system stability, Neural Computing and Applications 31 (4) (2019) 1237–1248. doi:10.1007/s00521-017-3156-8.

[6] V. S. Perić, A. T. Sarić, D. I. Grabež, Coordinated tuning of power system stabilizers based on fourier transform and neural networks, Electric power systems research 88 (2012) 78–88. doi:10.1016/j.epsr.2012.01.017.

[7] D. Chitara, K. R. Niazi, A. Swarnkar, N. Gupta, Cuckoo search optimization algorithm for designing of a multimachine power system stabilizer, IEEE Transactions on Industry Applications 54 (4) (2018) 3056–3065. doi:10.1109/TIA.2018.2811725.

[8] W. Peres, E. J. De Oliveira, J. A. Passos Filho, I. C. da Silva Junior, Coordinated tuning of power system stabilizers using bio-inspired algorithms, International Journal of Electrical Power & Energy Systems 64 (2015) 419–428. doi:10.1016/j.ijepes.2014.07.040.

[9] G. L. da Cunha, R. A. Fernandes, T. C. C. Fernandes, Small-signal stability analysis in smart grids: An approach based on distributed

decision trees, Electric Power Systems Research 203 (2022) 107651. doi:10.1016/j.epsr.2021.107651.

[10] S. K. Azman, Y. J. Isbeih, M. S. El Moursi, K. Elbassioni, A unified online deep learning prediction model for small signal and transient stability, IEEE Trans. on Power Systems 35 (6) (2020) 4585. doi:10.1109/TPWRS.2020.2999102.

[11] J. Sanchez-Gasca, J. Chow, Performance comparison of three identification methods for the analysis of electromechanical oscillations, IEEE Transactions on Power Systems 14 (3) (1999) 995. doi:10.1109/59.780912.

[12] S. A. Dorado-Rojas, F. Fachini, T. Bogodorova, G. Laera, M. de Castro Fernandes, L. Vanfretti, ModelicaGridData: Massive power system simulation data generation and labeling tool using Modelica and Python, SoftwareX 21 (2023) 101258. doi:10.1016/j.softx.2022.101258.

[13] S. A. Dorado-Rojas, T. Bogodorova, L. Vanfretti, Time series-based small-signal stability assessment using deep learning, in: 2021 North American power symposium (NAPS), IEEE, 2021, pp. 1–6. doi:10.1109/NAPS52732.2021.9654643.

[14] J. Wang, P. Pinson, S. Chatzivasileiadis, M. Panteli, G. Strbac, V. Terzija, On machine learning-based techniques for future sustainable and resilient energy systems, IEEE Transactions on Sustainable Energy (2022). doi:10.1109/TSTE.2022.3194728.

[15] T. Zufferey, A. Ulbig, S. Koch, G. Hug, Unsupervised learning methods for power system data analysis, in: Big data application in power systems, Elsevier, 2018, pp. 107–124. doi:10.1016/B978-0-12-811968-6.00006-1.

[16] T. Bogodorova, D. Osipov, L. Vanfretti, Automated design of realistic contingencies for big data generation, IEEE Transactions on Power Systems 35 (6) (2020) 4968–4971. doi:10.1109/TPWRS.2020.3020726.

[17] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: A strong baseline, in: 2017 Int. joint conf. on neural networks (IJCNN), IEEE, 2017, pp. 1578–1585. doi:10.1109/IJCNN.2017.7966039.

[18] A. Le Guennec, S. Malinowski, R. Tavenard, Data augmentation for time series classification using convolutional neural networks, in: ECML/PKDD workshop on advanced analytics and learning on temporal data, 2016. doi:halshs-01357973.

[19] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Deep learning for time series classification: a review, Data Mining and Knowledge Discovery 33 (4) (2019) 917–963. doi:10.1007/s10618-019-00619-1.

[20] J. Serrà, S. Pascual, A. Karatzoglou, Towards a universal neural network encoder for time series., in: CCIA, 2018, pp. 120–129. doi:10.3233/978-1-61499-918-8-120.

[21] C. J. Tavora, O. J. Smith, Characterization of equilibrium and stability in power systems, IEEE Transactions on Power Apparatus and Systems (3) (1972) 1127–1130. doi:10.1109/TPAS.1972.293468.

[22] V. Venkatasubramanian, Real-time strategies for unwrapping of synchrophasor phase angles, IEEE Transactions on Power Systems 31 (6) (2016) 5033–5041. doi:10.1109/TPWRS.2016.2538209.

[23] J. H. Chow, J. J. Sanchez-Gasca, Power system modeling, computation, and control, John Wiley & Sons, 2020.