Time Series-Based Small-Signal Stability Assessment using Deep Learning

Sergio A. Dorado-Rojas, Tetiana Bogodorova, Luigi Vanfretti
Rensselaer Polytechnic Institute
Troy, NY, USA
{dorads, bogodt2, vanfrl}@rpi.edu

***Abstract*—Power system operators obtain information about an electrical grid's current condition using available tools in control centers. These tools employ simple algorithms for data analysis and processing to expedite decision making. We propose to use Deep Learning algorithms to provide more information about the power system's operating condition without loss in computational performance. This work performs a comparison between several Deep Learning algorithms for time series-based classification of power system small-signal stability, which can be applied to both PMU data or synthetic measurements from simulations. In particular, several case studies are performed using line current and bus voltage data as input for the proposed algorithms. To find the best method for the classification task, the following neural network (NN) architectures are studied: a multi-layer perceptron, a fully-convolutional NN, an inception network, a time convolutional NN, and a multi-channel deep convolutional NN. Training and testing data sets were obtained from the IEEE 9 bus system by performing dynamic simulations subjected to a vast array of operating conditions (i.e., different power flow solutions, and contingencies). The computational time of the implemented algorithms is measured. The multi-channel deep convolutional NN shown the best performance in most of the reviewed cases.**

***Index Terms*—Deep learning, convolutional neural network, power systems, small signal stability**

I. Introduction

The standard approach of small-signal stability analysis (SSA) [1] is to quantify the effects of the small disturbances, such as a line trip, by analyzing the stability properties of the linearized power system model. The obtained model is usually presented in a form of a state-space representation. Then, linear system analysis is applied to assess the small-signal stability condition by evaluating the system's eigenvalues, thereby obtaining the damping ratio of the dominant modes [1]. When the model is not available, a measurement-based mode identification technique, such as Prony [2], is applied. Prony requires recording of several swing oscillations to get acceptable accuracy in mode identification. Despite the usefulness of the linear analysis and measurement-based mode estimation techniques to evaluate SSA, their implementation for a real-time analysis poses several challenges. Such drawbacks arise from the complexity of the required computations for a large-scale system (e.g. maintaining a validated model) or the measurement data requirements to obtain acceptable accuracy (e.g. filtering).

This work was funded in part by the New York State Energy Research and Development Authority (NYSERDA) under grant agreement numbers 137951 and 137940, and in part by the Center of Excellence for NEOM Research at King Abdullah University of Science and Technology.

Deep Learning (DL) is a family of Machine Learning (ML) algorithms that are based on artificial Neural Networks (NNs) with feature learning capability [3]. In other words, these algorithms allow extracting essential elements of the data that define an output that has to be learned by the DL algorithm. Some of ML algorithms have been recently applied in power systems to enhance and evaluate small-signal stability. For example, the work of [4] performs coordinated tuning of power system stabilizer (PSS) parameters using heuristic optimization algorithms. Likewise, in [5], a cuckoo search is employed to find optimal PSS parameters that guarantee small-signal stability. Moreover, in [6], the parameters of a unified power flow controller are tuned via an NN whose weights are optimized using the Levenberg-Marquardt algorithm. Despite some efforts to study SSA with conventional ML techniques, there is no research on the small-signal stability assessment using deep learning methods to the authors' best knowledge.

Deep Learning algorithms can be trained using collected measurements and data generated performing offline power system simulations. The trained algorithm has to be deployed for inference on real-time data to provide fast identification of the state in which the system operates. However, in this new approach, the main challenges are: (i) to choose the effective DL algorithm, (ii) to select the best data and appropriate amount of data to train the algorithm with sufficient accuracy while (iii) ensuring an acceptable real-time performance. This work aims to provide insight into these challenges.

A. Contribution

This paper's main contribution is a proposal to use and a comparison of the training results of the state-of-the-art Deep Learning algorithms for small-signal stability assessment using time-series input power system data (either synthetic (simulated) or the measurements from a real power system). Special attention is paid to the case studies on the measurement selection (voltage or current measurements) and data preparation for the algorithms' training. For this purpose, case studies are performed for voltage and current measurements with 1% Gaussian noise (added in the preprocessing) and without noise of a simulated model to assess small-signal stability. The performance of studied architectures is discussed and compared using the performance evaluation metrics such as

accuracy, precision and recall. The architectures are: a Multi-Layer Perceptron, a Fully Convolutional NN, a Time Convolutional NN (CNN), an inception network, and a Multi-Channel Deep CNN. Hyperparameters of the algorithms, such as the effective number of epochs, are also reported. Furthermore, the trained algorithms' prediction/classification time per data sample is presented to analyze which model is most suitable for a real-time application.

This article expands the authors' previous work in [7] by considering a time-series data as input directly, rather than the set of eigenvalues describing a particular operating condition. In [7], we used the output from several dynamic simulations to compute the eigenvalues characterizing a particular condition. Eigenvalues had to be further preprocessed before classifying the operational condition of the system using the decision boundaries learned during training. In this work, the pre-processing task of identifying the system eigenvalues is a feature learning task performed by the particular layers of the proposed NN architectures.

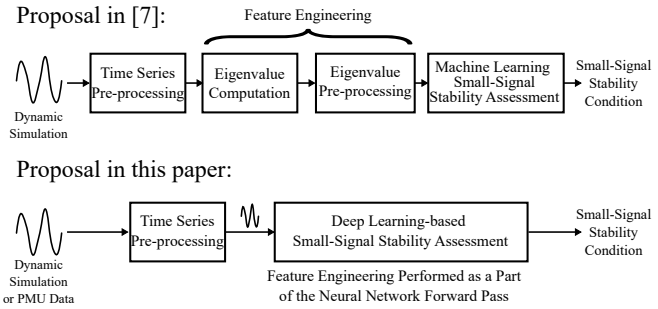

Fig. 1. Approaches to small-signal stability assessment by conventional ML and DL.

Since unstable operating conditions are rare, they are created via model-based simulation. To this end, we follow the automated phasor time-domain simulation approach that is based on realistic selection of a set of contingencies for power systems described in [8] to produce training data. The procedure in [7] follows a different contingency generation algorithm.

In contrast to our previous work, using time series data as input expands the applicability of the methodology and enables the potential inclusion of PMU data for training. Both approaches require a time series pre-processing stage, but the approach of this paper does not require any eigenvalue computation. We emphasize that the feature engineering task (i.e., computing eigenvalues from time-series data) is carried out in the forward-pass of the NN. This is not possible with the classic ML methods applied in [7] where the pre-processing had to be done beforehand. The difference between both approaches is underlined in Figure 1.

B. Paper Organization

This paper is structured as follows: Section II presents an overview of SSA from time-series data and a description of the studied DL architectures. Section III describes the data generation and pre-processing stages. In Section IV, we show the results of each case study and discuss NN performance for SSA. Finally, Section V elaborates on common challenges of the proposed deep learning application and Section VI concludes the work.

II. Small-Signal Stability and Deep Learning

A. Time Series-Based Small-Signal Stability Assessment

The proposed approach (Fig. 2) to perform SSA consists of an offline and an online step. The former includes data preparation (time series preprocessing and labeling) and the deep NN training; the latter refers to exploitation with the trained NN on non-labeled data to carry out SSA.

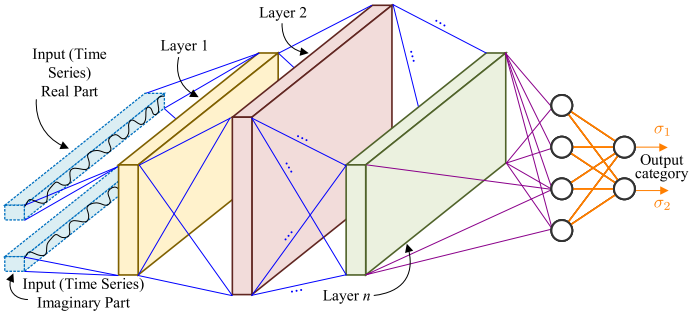

Fig. 2. Convolutional neural network structure for time series classification.

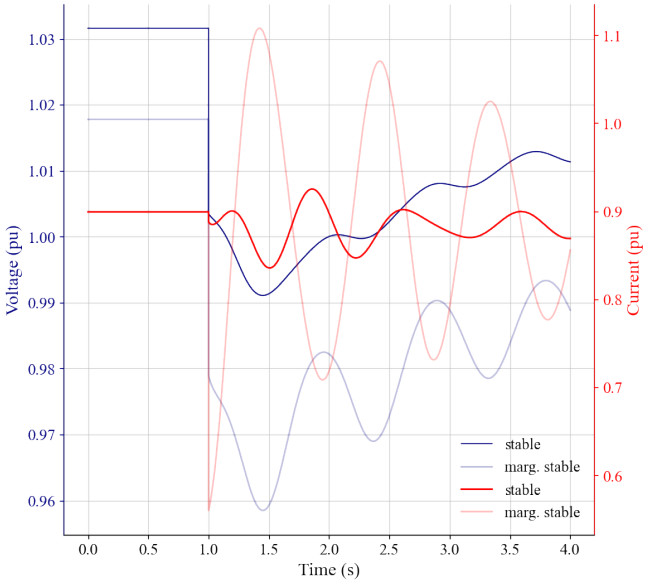

Fig. 3. An example of synthetic current and voltage magnitude measurements for stable and marginally stable scenarios (the disturbance occurs at $t = 1.0$ s).

B. Deep Learning for Time-Series Classification

Since this is a supervised learning problem, we need to provide a set of examples for the DL algorithms to learn how to categorize the data. These examples are the time series traces with labels. The labels indicate whether a trace corresponds to either of the classification categories. Labeling is performed using symbolic linearization of the grid model to get a system matrix. Then, the dominant eigenvalue is used to compute a damping ratio that defines the corresponding label

(stable/unstable) for a given scenario. Thus, in our experiments a training instance is composed of a phasor time-domain profile of a generator bus voltage or line current (see Fig. 3) and a binary label calculated offline via model linearization. During training, the NN learns to identify the characteristic patterns of each category by updating its parameter values iteratively. After training, a deployed NN can classify/define the label for the operating condition by feeding voltage or current data directly.

DL stands from a NN architecture with (deep) numerous interconnected layers of reduced number of neurons in each layer without loss in performance. In general, DL algorithms consist of three main components: a NN architecture, a cost/loss function, and an optimization method. The architecture is related to the NN structure, which includes the number of neurons in each layer, the number of layers, and the type of layers. The cost/loss function represents the criterion based on which the NN improves its performance while learning from the given examples. Finally, the optimization method provides the mechanism to update the NN parameters, such as weights in the functions that represent the neurons. In this section, several deep NN architectures recommended for Time Series Classification (TSC) are introduced (see Table I).

1) Multi-layer Perceptron (MLP): An MLP [9], or a deep feedforward network, computes an output by the weighted sum of every input signal component, followed by a pointwise nonlinear activation. When deployed for TSC, an MLP's performance may downgrade due to its architectural properties. The temporal interdependence in the input is not captured because all components are weighted individually before being passed to each layer. Despite this drawback, an MLP [9] is used as a base case to compare with more sophisticated NNs for TSC since it represents a lower performance bound.

2) Convolutional Neural Networks (CNNs): CNNs use convolution as a mapping operator in at least one of their layers. Convolution has several advantages: sparse interactions, parameter sharing, equivariant representations, and ability to work with inputs of variable size. Sparse interactions mean that the output does *not* interact with every input unit, allowing memory reduction (fewer parameters) and efficiency boost (fewer operations). Likewise, parameter sharing indicates that the same parameter is used on several layers, which improves computational efficiency. The equivariance property for dealing with time series data means that when different features appear in the input, and a particular event is time-shifted, the same signal will appear in the output, shifted equally [3].

A typical layer of a CNN includes three stages: *convolution*, *detector*, *pooling*. In the *convolution* stage, the layer performs convolutions on the input to get a set of outputs that run through a nonlinear activation function (e.g., ReLU) in the *detector* step. Finally, a *pooling* function replaces the output at a certain location with a summary statistic of the nearby outputs (e.g., max pooling returns the maximum value of the particular neighborhood). Thus, the output is invariant to noise.

In this work, four CNN architectures –fully convolutional NNs (FCNs), inception, Multi-channel Deep CNN (MCDCNN), and Time-CNN- are studied. The general structure of the CNNs is shown in Fig. 2. The real and imaginary parts of a phasor-domain time series are passed as inputs. The CNN is composed of n convolutional layers (layers $1-n$ in Fig. 2) that followed by a fully-connected layer (purple in Fig. 2) and a softmax activation in the output layer (orange in Fig. 2). Let $\mathbf{z}$ be the vector of values at the last NN layer. A softmax activation σ for the binary case (i.e., $\mathbf{z} \in \mathbb{R}^2$) is computed and the category predicted as follows):

$$y = \text{argmax}\left[\sigma\left(\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}\right)\right] = \text{argmax}\begin{bmatrix} \frac{e^{z_1}}{e^{z_1}+e^{z_2}} \\ \frac{e^{z_2}}{e^{z_1}+e^{z_2}} \end{bmatrix} \quad (1)$$

A FCN (fcn in Table I) has three convolutional layers followed by a batch normalization stage and a ReLU activation function each. The training is performed minimizing a cross-entropy loss function. The third convolutional layer's output is averaged over time dimension (global average pooling) before being fed into an output softmax layer.

Time-CNN (cnn in Table I) [10] has two convolutional layers followed by softmax activation and average pooling operation each. The loss function is a mean squared error.

An inception network (inception in Table I) [11] consists of 6 inception blocks –concatenated outputs of 4 conv layers and one max pooling connected in parallel- followed with batch normalization operation and activation function, a global average pooling layer, and a fully-connected output layer. Thus, it takes a previous input and passes it to the parallel convolutional layers concatenating the outputs together with the output of max pooling operation over the input. So, a bigger variety of filters can be chosen in each layer.

A Multi-Channel Deep Convolutional Neural Network (mcdcnn in Table I) [12] consists of typical CNN layers where the convolutions are performed in parallel on each dimension of the time series data. Each two convolutional layers have 8 output filters of length 5 with a ReLU activation function, followed by a max-pooling operation. The convolutional layers' output is flattened before entering a fully-connected layer with a ReLU activation function. Finally, the output layer is fully-connected of the size that corresponds to number of classes with a softmax activation function (see Eq. (1)).

III. Case Study

1) Data Generation: We have generated trajectories using the IEEE 9 bus system (24 state variables; 203 algebraic variables) initialized for a vast array of operating conditions (i.e. power flows) and subjected to realistic contingencies that are generated using the algorithm in [8]. Once the contingency scenario is applied, a dynamic simulation is carried out for 4 s to generate trajectories (see Fig. 3). A total of 1805 simulations were generated, from where $n_{\text{training}} = 1083$, $n_{\text{testing}} = 602$, and $n_{\text{validation}} = 120$.

2) Data Preparation: The voltage signals at generator buses contain system dynamics that, in practice, can be measured by PMUs. Thus, the real and imaginary parts of voltage at some generator buses and selected line current signals are

TABLE I
MODEL CHARACTERISTICS

Model	Parameters		Optimizer	Layers		
	Trainable (T)	Non-trainable (NT)		Layer Number	Type (activation)	Hyperparameters
fcn	265,986	1,024	Adam	8 (4T + 4NT)	3x conv1d (relu) 1x dense (softmax)	$n_{\text{filters}} = (128, 256, 218)$ kernel size = (8, 5, 3) padding = same
inception	422,850	2,048	Adam	64 (17T + 47NT)	16x conv1D (relu) 1x dense (softmax)	$n_{\text{filters}} = 32$ kernel size = 41 padding = same
mcdcnn	1,443,526	0	SGD	15 (8T + 7NT)	7x conv1D (relu) 1x dense (softmax)	$n_{\text{filters}} = 8$ kernel size = 5 padding = same
cnn	1880	0	Adam	7 (3T + 4NT)	2x conv1D (sigmoid) 1x dense (softmax)	$n_{\text{filters}} = (6, 12)$ kernel size = (7, 7) padding = valid
mlp	1,007,502	0	Adadelta	8 (4T+ 4NT)	3x dense (relu) 1x dense (softmax)	$n_{\text{neurons}} = 500$

used to construct training and testing datasets. Normalization is not required since all waveforms are per-unitized.

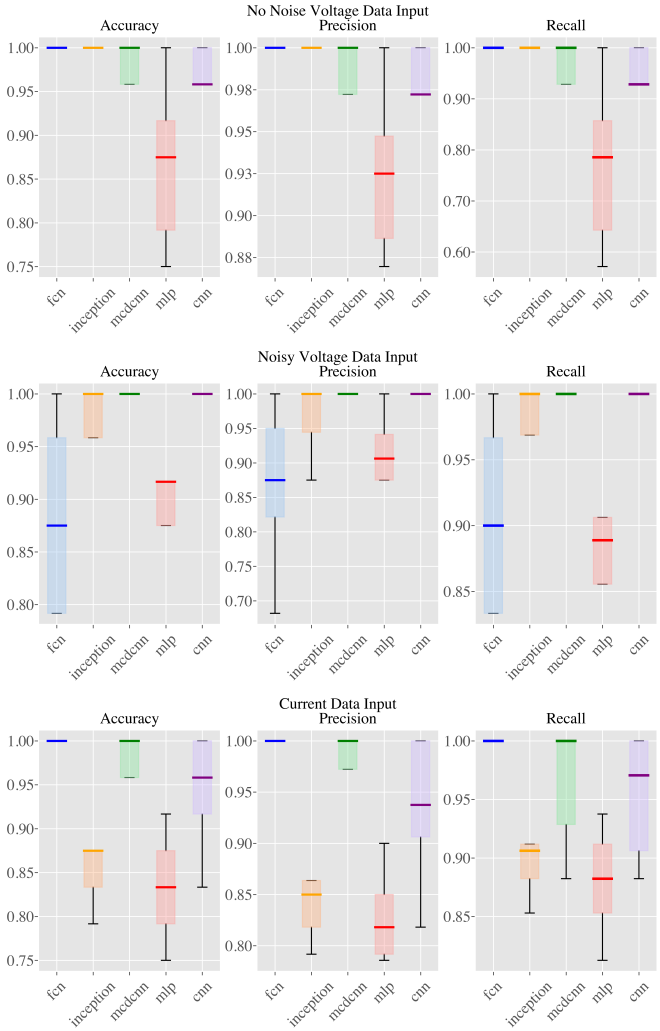

Fig. 4. Training results for 5-fold cross-validation (full length data set).

Also, 1% Gaussian noise, typical for PMU measurements, is added to the voltage signals. In summary, three datasets that include pairs of real and imaginary parts of each signal (voltage, noisy voltage, and current) were generated.

The labeling of the datasets for training of the deep learning NNs is performed using the classic small-signal stability analysis. Small-signal stability is analyzed either by linearizing a nonlinear grid model or identifying the excited dynamics from measurements (e.g., PMU data).

For ease of interpretation, we limit the NN task (Fig. 2) to the binary (two classes) problem of stable (when damping ratio of the system $\zeta > 5\%$) and marginally stable ($0\% < \zeta < 5\%$) classification. To train and validate the models on the more complex learning problem, the resulting data sets are further preprocessed. Thus, to expedite SSA, the measurement length has been reduced to 75% of the original simulation time. The resulting length corresponds to a measurement window of 3 s. This would require less information to be passed to the NN. Likewise, the number of training instances is reduced by a factor of 3 ($n_{\text{training}} \approx 300$) to account for data scarcity.

For all experiments, the input tensors' shape is $(n_{\text{scenarios}}, T, 2)$, where T is the number of points in the time-series. The last element of the tuple, '2', indicates that real and imaginary parts are input data features.

IV. RESULTS

Results for the experiments with full and reduced length time series, with all and fewer instances, are presented in Figs. 4 and 5. Conventional splitting results are shown in Fig. 6.

1) Result Analysis: The number of samples for the presented case studies could be considered small when applying DL. Thus, k-fold cross-validation ($k = 5$) was performed to validate the performance of each DL model.

In Fig. 4, we observe that the CNN architectures with the tuned hyperparameters (fcn and cnn) can achieve 100% performance on the data generated for the experiment. This does *not* mean that the NN will be fully accurate when deployed but rather that the performance will be very high. Thus, in our setup, CNNs successfully detect the oscillation patterns

allowing for distinguishing a stable operating condition from a marginally stable one.

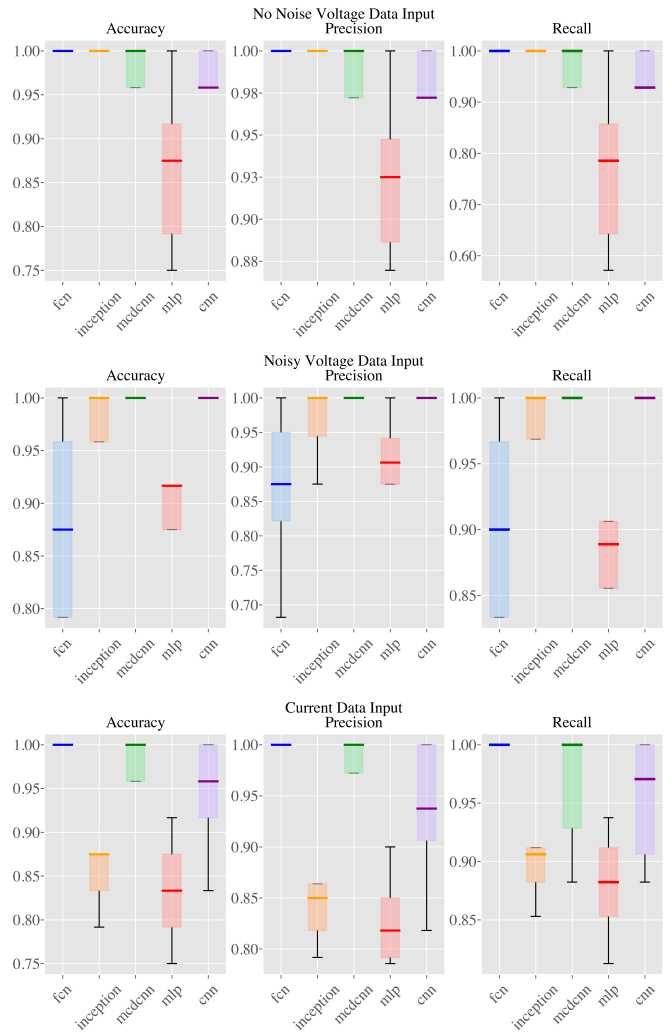

Fig. 5. Training results for 5-fold cross-validation (reduced length data set).

Fig. 5 shows that performance is not significantly downgraded after reducing the time series length. Some architectures can relate the oscillating behavior with a category better than in the previous case. Nevertheless, we observe that assessing the system's condition from current measurements could be more challenging than for voltage inputs. This is due to presence of more prominent oscillations in voltage measurements in comparison to the current traces from the training data.

Purely convolutional architectures (fcn, mcdcnn, and cnn) show metrics beyond 99% regardless of noise for both voltage and current inputs. In fact, the performance of all models is above 99% for noiseless voltage. However, the inception model lost its classification capability when exposed to noisy measurements (accuracy: 0.6761; precision: 0.3380; recall: 0.5000). For current data inputs (Fig. 6), the performance of the MLP degrades which is expected due to the limitations of this architecture for identifying patterns in time series data.

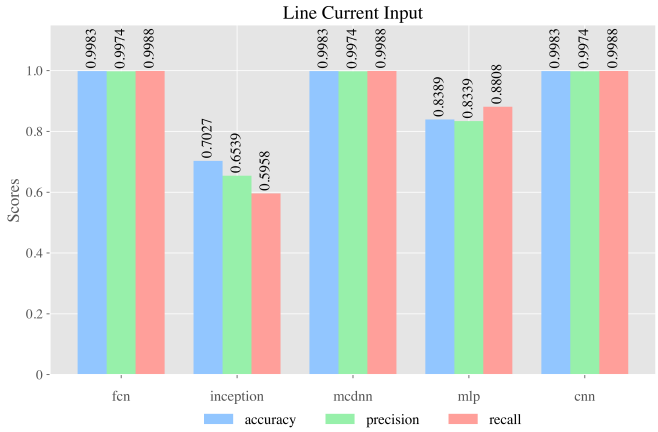

Fig. 6. Prediction results for line current input (reduced training data set).

In the reviewed cases, the oscillatory pattern is easily learned by the convolutional layers since it contains the excited dynamics of the small system. In the cases of presence of more complex dynamics in the large systems, achieving high performance would require finer hyperparameter tuning such as bigger number of neurons and larger data sets for learning, which would be more computationally expensive. Despite this, we conclude that DL methods are powerful to learn classifying patterns that are typical for SSA.

2) Computational Performance Analysis: To measure the computational performance of the NN models in production, the prediction time per sample and the number of epochs to train the best model (see Table II) were computed. The prediction time per sample indicates how fast the NN computes a prediction of the system stability given a time series input. The number of epochs is a relative measure of the training effort required to optimally tune the model. For all architectures, prediction time is in the order of milliseconds which favors the deployment in real-time pipelines.

MLP is the fastest architecture in terms of prediction, but also the one requiring a larger number of epochs to achieve its best performance. Moreover, there is a tradeoff between production performance and training effort for all models. The *mcdcnn* (Table II) is optimal in both senses.

TABLE II
PREDICTION TIME AND NUMBER OF EPOCHS FOR BEST MODEL

Model	Voltage		Noisy Voltage		Current	
	Pred. Time	Eph Best	Pred. Time	Eph Best	Pred. Time	Eph Best
fcn	1.20 ms	58	1.21 ms	209	1.21 ms	133
inception	2.50 ms	**8**	2.43 ms	**11**	2.48 ms	**8**
mcdcnn	0.477 ms	59	0.478 ms	59	0.491 ms	45
mlp	**0.234 ms**	2226	**0.225 ms**	1760	**0.235 ms**	211
cnn	2.75 ms	248	0.282 ms	235	0.261 ms	165

V. DISCUSSION

The presented training pipeline of classical state-of-the-art deep learning architectures is very common in the computer science community. However, when solving specific field engineering tasks such as small-signal stability assessment, several issues arise and must be addressed. The first issue is

which data to use as an input to train the models and process these data. In this case, when PMUs are widespread in the power grid, we chose to use voltage or current data, either noisy or noiseless, to validate the selected NN architectures' ability to catch the patterns that define the system's state (stable or marginally stable). The second issue is to elicit the data enriched with such distinctive patterns, meaning to find the measured values after a contingency or create the big enough dataset of such synthetic measurements to achieve a good performance of the models. This challenge has been addressed in [8] whose algorithm has been adopted in this work to generate the data.

Another issue that is left open is a transfer learning possibility that can be applied to the best performance model. Transfer learning is characterized as a possibility to use the trained model for measurements collected from other power systems without significant degradation of the model performance. This point requires special attention and is left as future work. The next issue connected with the NN architecture's performance is how much data is enough to get an acceptable performance of the trained model. In this work, we gradually enlarged the amount of data until at least one model of the compared deep learning algorithms showed a good performance. But generally speaking, the larger the model, the more extensive dataset is required for training. In particular, if the power system is characterized by the significant number of eigenvalues that define the system's state, more combinations of the excited modes can be present after a contingency is applied. Therefore, more data is needed to train a deep neural network.

Eventually, the issue of the computational performance is significant if the trained model is applied in any control center. In Table II the prediction time per data sample is presented for each studied model. We consider that the trained model is used in a plug-and-play manner. This means that the presented time is enough to perform the small-signal assessment if the input data chunk is available. But one has to be aware that this chunk is of the length of 3 seconds. This is the time series length to capture enough patterns to classify the state of the system. Comparing the trained models to the classic Prony method performance, the deep learning models give a significant speedup in performance since the Prony method is effective only after the oscillations caused by contingency are fade out [2], [13]. Thus, for the Prony method to achieve a good performance, the needed time series length is of approximately 10 seconds in comparison to the 3 seconds that have been used in the presented case studies.

VI. Conclusions

This work proposed a novel methodology for time series-based power system small-signal stability assessment using deep learning. The models are tested using the labeled current and voltage measurements generated using massive dynamic simulations after a realistic contingency scenarios were applied. The accuracy, precision and recall show a good performance of the trained models with the selected hyperparameters and for the generated data. Prediction time and performance show the potential of NNs to be deployed in real-time control center tools for operator decision support.

References

[1] J. H. Chow and J. J. Sánchez-Gasca, "Linear Analysis and Small-Signal Stability," in *Power System Modeling, Computation and Control*. Wiley-IEEE, 2020.

[2] J. F. Hauer, C. J. Demeure, and L. L. Scharf, "Initial results in prony analysis of power system response signals," *IEEE Transactions on Power Systems*, vol. 5, no. 1, pp. 80–89, 1990.

[3] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, no. 2.

[4] W. Peres, E. J. de Oliveira, J. A. Passos Filho, and I. C. da Silva Junior, "Coordinated tuning of power system stabilizers using bio-inspired algorithms," *International Journal of Electrical Power & Energy Systems*, vol. 64, pp. 419–428, jan 2015.

[5] D. Chitara, K. R. Niazi, A. Swarnkar, and N. Gupta, "Cuckoo Search Optimization Algorithm for Designing of a Multimachine Power System Stabilizer," *IEEE Transactions on Industry Applications*, vol. 54, no. 4, pp. 3056–3065, jul 2018.

[6] M. J. Rana, M. S. Shahriar, and M. Shafiullah, "Levenberg–Marquardt neural network to estimate UPFC-coordinated PSS parameters to enhance power system stability," *Neural Computing and Applications*, vol. 31, no. 4, pp. 1237–1248, apr 2019.

[7] S. A. Dorado-Rojas, M. de Castro Fernandes, and L. Vanfretti, "Synthetic Training Data Generation for ML-based Small-Signal Stability Assessment," in *2020 IEEE SmartGridComm*, 2020.

[8] T. Bogodorova, D. Osipov, and L. Vanfretti, "Automated Design of Realistic Contingencies for Big Data Generation," *IEEE Transactions on Power Systems*, vol. 35, no. 6, pp. 4968–4971, nov 2020.

[9] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 Int. joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 1578–1585.

[10] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162–169, 2017.

[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[12] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Exploiting multi-channels deep convolutional neural networks for multivariate time series classification," *Frontiers of Computer Science*, vol. 10, no. 1, pp. 96–112, 2016.

[13] J. Sanchez-Gasca and J. Chow, "Performance comparison of three identification methods for the analysis of electromechanical oscillations," *IEEE Transactions on Power Systems*, vol. 14, no. 3, pp. 995–1002, 1999.