Voltage Instability Prediction Using a Deep Recurrent Neural Network

N.B. When citing this work, cite the original published paper.

(article starts on next page)

Voltage Instability Prediction Using a Deep Recurrent Neural Network

Hannes Hagmar, *Student Member, IEEE,* Lang Tong, *Fellow, IEEE,* Robert Eriksson, *Senior Member, IEEE,* and Le Anh Tuan, *Member, IEEE*

Abstract—This paper develops a new method for voltage instability prediction using a recurrent neural network with long short-term memory. The method is aimed to be used as a supplementary warning system for system operators, capable of assessing whether the current state will cause voltage instability issues several minutes into the future. The proposed method uses a long sequence-based network, where both real-time and historic data are used to enhance the classification accuracy. The network is trained and tested on the Nordic32 test system, where combinations of different operating conditions and contingency scenarios are generated using time-domain simulations. The method shows that almost all N-1 contingency test cases were predicted correctly, and N-1-1 contingency test cases were predicted with over 95 % accuracy only seconds after a disturbance. Further, the impact of sequence length is examined, showing that the proposed long sequenced-based method provides significantly better classification accuracy than both a feedforward neural network and a network using a shorter sequence.

Index Terms—Dynamic security assessment, long short-term memory, recurrent neural network, voltage instability prediction, voltage stability assessment.

I. INTRODUCTION

VOLTAGE instability is one of the main limitations for secure operation of a modern power system [1]. A voltage instability event can often be deceiving, where the system may seem stable for several minutes after a disturbance, only to end up in an unstable state within a short time [2]. When instability finally is detected, the system may already have become severely degraded and the risks of an extended blackout may have increased significantly.

To ensure a secure operation, system operators often use an approach called dynamic security assessment (DSA). DSA includes time-domain analysis to test a power system's dynamic response after a set of contingencies assure its ability to reach a stable post-disturbance operating point [3]. Time-domain analysis and assessment of the dynamic stability are complex tasks, and even with recent progress in high performance computing, it is generally not feasible to assess the dynamic stability in real-time [3].

To overcome this issue, various machine learning (ML) methods have been proposed in the literature. The main advantage of using ML is that high-cost computations can be performed off-line. Once the ML algorithm is trained, it

can almost instantaneously provide estimations and warnings to operators that otherwise would require time-consuming computations. Examples of DSA methods based on ML are found in [3]–[8], where mainly various decision tree (DT) or neural network (NN) methods are utilized.

Voltage security assessment (VSA) is a branch in DSA that specifically examines the impact of voltage instability events [9]. This paper deals with the *emergency* applications of VSA, where the *current* system state is assessed. Here, the stability of the system is not tested with respect to a set of contingencies; rather, the system may already have suffered a disturbance. The aim of these methods is to perform voltage instability prediction (VIP) and to detect the *onset* of instability, rather than its consequences. Fast prediction of voltage instability would then allow system operators to trigger fast remedial actions to control the system back into stable operation again.

A method for VIP based on ML was first proposed in [4], where a DT was trained on a generated database consisting of the intermediate, short-term equilibrium that follows a disturbance. This post-contingency state, where the majority of the electromechanical transients have died out, was referred to as the "just after disturbance" (JAD) state. Extensions of the method utilizing phasor measurements have later been proposed in [10]–[12], where the performance of different attributes or input data have been tested. A method based on support vector machines (SVM) was proposed in [13], where generators' predicted reactive power output was used as an indicator for voltage collapse. A method based on training a NN to online monitor voltage security was proposed in [14], and in [15], a NN was used to also indicate where in the system instability would emerge.

Most previously developed methods for VIP have in common that only instantaneous measurements are used as inputs to the VIP algorithms. These inputs represent the "state signal" that the ML algorithm uses to predict the future state. Ideally, the state signal should summarize all relevant information required to determine the future state of the system. A state signal achieving this is said to have Markov property [16]. However, the dynamic response of a power system cannot be modeled as a first-order Markov process using only the static states provided by available measurements in the power system. Rather, the future state of the system also depends on a range of unknown state variables such as the rotor speed of generators, tap positions, or rotor slips of induction motors.

An attempt to incorporate some time-related features to improve the performance in VIP was presented in [17], where

a temporal decision tree (TDT) approach was proposed. The TDT method, further discussed in [18] and [19], could incorporate some time-related variables, such as the difference between two measurements for specific value of elapsed time (Δt). However, the proposed TDT method did not allow efficient modeling of more complex time-dependent features and also required the user to define a finite number of candidate values of Δt, thus limiting the capability of the algorithm.

In response to these limitations, we propose a new method based on a recurrent neural network (RNN) with long short-term memory (LSTM). LSTM networks excel at capturing long-term dependencies [20], which is an inherent aspect in long-term voltage stability [2]. The method is, to the authors' knowledge, the first of its kind to use sequences of both current and past data with the aim to enhance the available state signal and implicitly take into account unknown state variables.

The main contributions of this paper are the following:

- A methodology for VIP using an LSTM network is developed. The LSTM network can utilize previous measurements, such as the trend of bus voltage magnitudes, tap changes, or fault locations, to improve the accuracy for VIP. The performance using the sequence-based approach is compared with an LSTM network using a shorter sequence and a conventional NN.
- A new training approach is developed to provide operators with an *online* assessment tool for potential voltage instability. The training approach allows the network to not only be trained on the measurements gathered from the JAD state but during the full dynamic trajectory that the system takes following a disturbance. As time progresses during a voltage instability event, the network is capable of incorporating new observations and continuously updating the assessment.
- A methodology for including consecutive contingencies (N-1-1) into the training data is presented. The paper also examines the ability of the LSTM network to generalize for VIP under N-1-1 contingencies. Such ability is especially valuable in overcoming the combinatorial increase of complexity in training.

The rest of the paper is organized as follows. In Section II, the theory regarding RNNs and LSTM is presented. In Section III, the proposed method is presented along with the steps for developing the training data and the training of the LSTM network. In Section IV, the results and discussion are presented. Concluding remarks are presented in Section V.

II. LONG SHORT-TERM MEMORY NETWORKS

Neural networks are a class of machine learning algorithms, highly capable of accurately approximating nonlinear functions, mapping a set of inputs to a corresponding set of target values. RNNs represent a specific type of NNs adapted for processing *sequential* input data [21]. However, the standard implementation of RNN has difficulties in capturing long-term dependencies of events that are significantly separated in time. In an LSTM network, such information can be propagated through time within an internal state memory cell, making the network capable of memorizing features of significance [22].
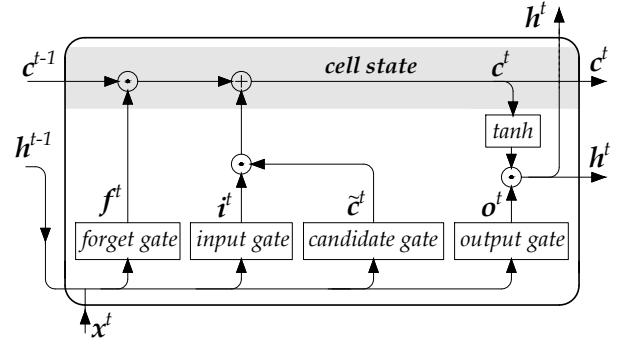

Fig. 1. Detailed schematics of an LSTM block

A typical LSTM-block is illustrated in Fig 1. The state memory cell, illustrated by the light grey area, is controlled by nonlinear gating units that regulate the flow in and out of the cell [20]. Following [22] and [20], the forward operation of an LSTM block is summarized below. It should be noted that each block consists of a number of hidden LSTM cells. Vector notation is used, meaning that, for instance, the hidden state vector h^t is not the output of a single LSTM-cell at time t, but the output of a vector of N LSTM-cells. The operation of an LSTM block at a time t may then be summarized by:

$$\boldsymbol{f}^t = \sigma\left(\boldsymbol{W}_f \boldsymbol{x}^t + \boldsymbol{U}_f \boldsymbol{h}^{t-1} + \boldsymbol{b}_f\right) \tag{1}$$

$$\boldsymbol{i}^t = \sigma\left(\boldsymbol{W}_i \boldsymbol{x}^t + \boldsymbol{U}_i \boldsymbol{h}^{t-1} + \boldsymbol{b}_i\right) \tag{2}$$

$$\tilde{\boldsymbol{c}}^t = \tanh\left(\boldsymbol{W}_c \boldsymbol{x}^t + \boldsymbol{U}_c \boldsymbol{h}^{t-1} + \boldsymbol{b}_c\right) \tag{3}$$

$$\boldsymbol{c}^t = \boldsymbol{f}^t \odot \boldsymbol{c}^{t-1} + \boldsymbol{i}^t \odot \tilde{\boldsymbol{c}}^t \tag{4}$$

$$\boldsymbol{o}^t = \sigma\left(\boldsymbol{W}_o \boldsymbol{x}^t + \boldsymbol{U}_o \boldsymbol{h}^{t-1} + \boldsymbol{b}_o\right) \tag{5}$$

$$\boldsymbol{h}^t = \boldsymbol{o}^t \odot \tanh(\boldsymbol{c}^t), \tag{6}$$

where element-wise multiplication is denoted by $\odot$, σ is the logistic sigmoid function, $\tanh$ is the hyperbolic tangent function, and with the following variables:

- $\boldsymbol{x}^t \in \mathbb{R}^M$: input vector to an LSTM block
- $\boldsymbol{h}^t, \boldsymbol{h}^{t-1} \in \mathbb{R}^N$: output vector at time t respectively t-1
- $\boldsymbol{f}^t \in \mathbb{R}^N$: activation vector of the forget gate
- $\boldsymbol{i}^t \in \mathbb{R}^N$: activation vector of the input gate
- $\tilde{\boldsymbol{c}}^t \in \mathbb{R}^N$: vector of the the candidate gate
- $\boldsymbol{c}^t \in \mathbb{R}^N$: cell state memory vector
- $\boldsymbol{i}^t \in \mathbb{R}^N$: activation vector of the output gate

where $\boldsymbol{W}$, $\boldsymbol{U}$, and $\boldsymbol{b}$ represent the weight matrices and bias vectors for each gate. The superscripts M and N refer to the number of inputs and hidden LSTM cells in each LSTM block, respectively.

The information stored in the state memory cell is regulated by the operation of the different gates, as illustrated in Fig. 1. By the operation of (1), the forget gate controls what information should be stored from the previous memory cell state, and what can be discarded as irrelevant. The input gate and candidate gate control and update the memory cell state with new information by the operation of (2)–(3). In (4), the state memory cell is first updated by an element-wise multiplication of the previous cell state memory vector and

the resulting vector of the forget gate. Then, the state memory cell is updated with new values provided by an element-wise multiplication of the resulting vectors from the input gate and the candidate gate. Equations (5)–(6) show how the hidden state is updated by the operation of the output gate, modulated by the updated cell state memory vector.

An LSTM *network* may then be constructed by creating a sequence of several LSTM blocks. A partition of an LSTM sequence is illustrated in Fig. 2, where each block has a directed connection to the following block in the sequence. If the block is the first one in the sequence, the past system state is initialized with a preset value. For a deep LSTM network, with several stacked layers, the inputs to the deeper layers consist of the hidden states of LSTM blocks of previous layers. The cell state memory is only passed along the time sequence between LSTM blocks of the same layer. Typically, for classification purposes, an output vector y is generated by applying a nonlinear function of the hidden state implemented by a separate feedforward NN. Depending on the application of the network, output vectors may be computed for a single, or for several, LSTM block's hidden states.
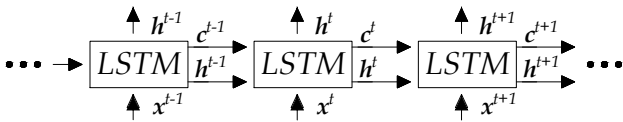

Fig. 2. An LSTM sequence with a directed connection between the blocks

The LSTM network can then be trained using a supervised approach, where a set of training sequences and an optimization algorithm are used to update and learn suitable values for the weights matrices and bias vector parameters. The training of an LSTM network is discussed in further detail in Section III-C.

III. METHODOLOGY

The proposed method for real-time VIP is based on off-line training of an LSTM network on a large data set consisting of time-domain simulation responses following a set of credible contingencies. The method is aimed to be used as a supplementary warning system that can assess the current state of the system in real-time. The LSTM network takes real-time and historic measurements and attempts to assess whether the *current* state will cause voltage stability issues several minutes into the future. As time progresses and if new events occur in the system, the network updates the assessment continuously. The network is also adapted to be able to indicate *where* in the system instability emerges, following the approach developed in [15], allowing more cost-effective countermeasures.

The first step of the method is the off-line generation of credible operating conditions (OCs) and contingency scenarios using time-domain simulations. The method is tested on a modified version of the Nordic32 test system with all data and models as presented in [23]. The load restoration following the contingencies is modelled by the actions of load tap changers that in steps restore the voltage-dependent loads. The corresponding generation changes are controlled mainly through
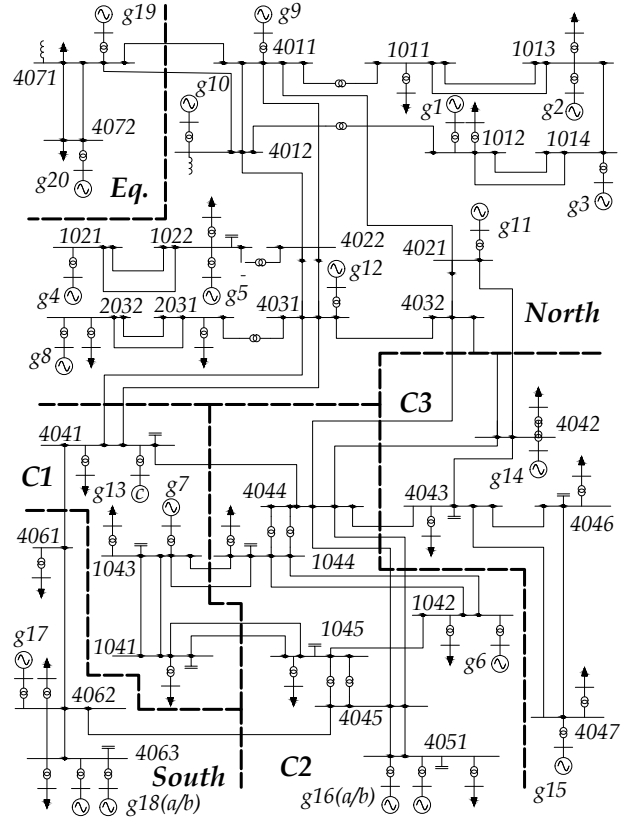

Fig. 3. One-line diagram of the modified Nordic32 system with subareas

governors that control the output of the hydro generators in the system. Schemes for automatic generation control or more dynamic load restoration models could be added in the dynamic models in the system but for the results to be more easily replicable, the same system as was presented in [23] was used. After a representative training set is generated, training of the LSTM network is performed. Each step in the methodology is described in the following subsections.

A. Generation of training data

The generation of a training set is a critical step and a range of different initial OCs and contingencies were included to generate a representative training set. Dynamic simulations were performed using PSS®E 34.2.0 with its built-in models [24]. The steps of generating the training data are illustrated as a flowchart in Fig. 4 and can be summarized as follows:

1) Initial OCs: For the Nordic32 system, the initial OCs were randomly generated around the stable operation point denoted as "operating point B" in [23]. A large number of possible OCs were simulated by randomly initiating the loads from a uniform distribution around the base case load levels (80 % of the original load as a lower limit and 120 % as an upper limit), while the power factor of the loads was kept constant. The total load change was distributed among the generators based on a weighted random distribution, where a higher rated capacity of a generator results in a higher probability to cover a larger share of the total load change. All generation that could not be supplied by the regular generators were distributed to the slack bus generator g20, see Fig. 3.
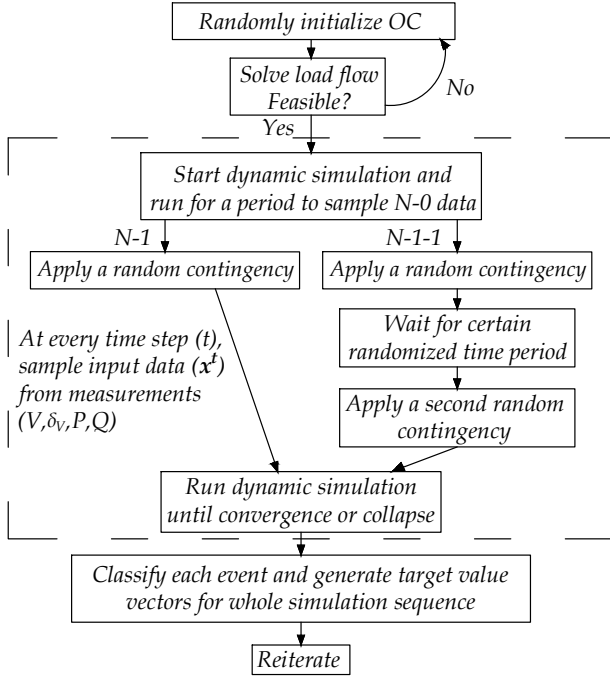
Fig. 4. Flowchart for generating input data and target values

In real applications, more delicate methods for efficient database generation and more careful generation of relevant OCs should be used [3], [25], where for instance the impact of unit commitment and topology changes are taken into account.

2) Solve and check for feasibility: The generated OCs were solved with a power flow simulator, which served as a starting point for the dynamical simulation. If the system load flow did not converge, the initial OC was re-initialized.

3) Start dynamic simulation and introduce contingencies: Two separate dynamic simulations were then initiated for the N-1 and the N-1-1 cases. The process is illustrated in Fig. 5. For each of the two cases, the system runs without any contingencies for 65 seconds to generate a sufficient amount of N-0 data for the network to train on. At $t = 66$ seconds, the *same* first contingency was applied to both of the cases. After an additional uniformly distributed random time in $[10 - 30]$ seconds after the first contingency, a secondary consecutive contingency was applied for the N-1-1 cases. Events resulting in several (near-)simultaneous contingencies were not taken into account (N-k events).

The considered contingencies in the simulations were either (i) tripping of a generator, or (ii) a three-phased fault during 0.1 seconds, followed by tripping the faulted line, which was then kept tripped during the remaining time of the simulation. The first contingency was chosen to be a major fault, meaning a fault on any transmission line connecting the different main areas in the system (excluding the "Eq." area, see Fig. 3), *or* any larger thermal generator in the "Central" area. The second contingency, for the N-1-1 cases, included tripping of *any* transmission line in the whole system, excluding lines in the "Eq." area. No variations of load and generation were taken into account during the dynamic simulations as these, in the relatively short time of the simulations, are presumed to have a small impact on the system stability.

4) Sample inputs and run until stopping criteria: For each of the two cases, an input vector x^t consisting of measurements of all bus voltage magnitudes (V_{mag}) and angles (V_{θ}), active and reactive power flows (P_{flow}, Q_{flow}), were sampled every second ($\Delta t = 1s$) and saved in a data file. No information regarding the type and location of applied the contingencies were sampled, as this information implicitly can be learned by the LSTM network. For instance, the LSTM network should be able to correlate a zero power flow in a transmission line with that line being out of service.

Each dynamic simulation ran for a total of 560 seconds, but was, in the case of a major voltage collapse, stopped in advance. The simulation interval of 560 seconds was chosen to allow time for *all* dynamic events to occur and for the system to either fully stabilize or collapse. It should be noted that systems with different dynamics may require longer or shorter simulation times. To reduce the computational burden when generating the training set for a real power system, more intricate methods to determine whether the system has stabilized can be adopted. For instance, by monitoring the actions of components with slow dynamics, such as load tap changers or excitation limiters of generators, a more efficient way of determining whether the system has been stabilized can be developed. If none of these components has acted in a time frame of the component with the longest activation time, the power system can be assumed to have fully stabilized.

5) Classification: For each case, a sequence of true target value vectors $y^1, ..., y^{560}$ was generated for every time step in the time-domain simulation. Each y^t in these sequences represents the classification of the system if the system is allowed to run from time t up until 560 seconds without any changes to the current system. As time progresses and new events occur, the class of y^t may change. The sequences consist of multidimensional vectors where the actual class is encoded using one-hot (binary) encoding.

The classification was performed according both to the severity and the location of the system degradation at the *end* of the time-domain simulation. The system was defined as stable if *all* transmission bus voltage magnitudes were above or equal to 1 pu, in an alert state if *any* transmission bus voltage magnitude ranged between $0.9 < V < 1.0$ pu, and in an emergency state if *any* transmission bus voltage magnitude was below 0.9 pu. Overvoltages were not taken into account.

The target values for the *alert* cases were also classified according to *where* the lowest bus voltage magnitudes were found at the end of each dynamic simulation.

The Nordic32 test system is predefined into four different regions, namely: "Eq", "North", "Central", and "South" [23]. The regions "North", "South", and "Eq." were found to be stable regions, and no alert events were found in these regions for any of the simulated cases. To test the capability of the network to also indicate where instability emerges in the system, the "Central" area was divided into three separate regions (indicated by **C1**, **C2**, **C3** in Fig. 3). The classification for each time step of each simulation belonged then to one of 5 different possibilities. Either the whole system was predicted stable; it ended up in an emergency state; or an alert state
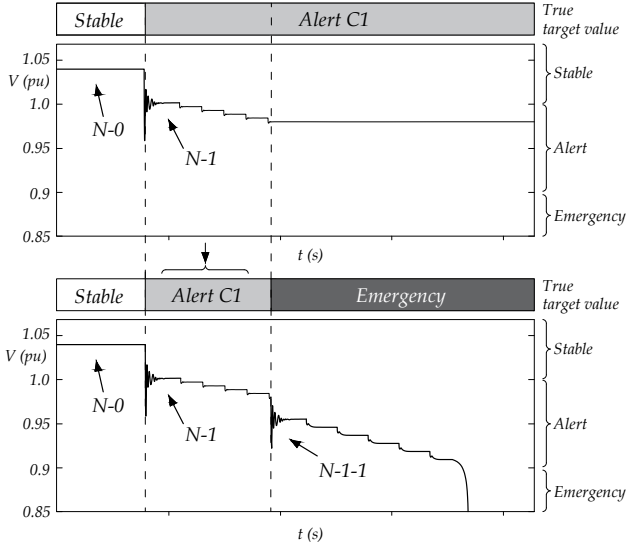
Fig. 5. Example of classification of an N-1 and an N-1-1 case

Fig. 6. The proposed LSTM network architecture

was predicted in one of the three defined regions (**C1**, **C2**, or **C3**) where the *lowest* occurring transmission bus voltage was found.

The classification process is illustrated in Fig. 5. The target values are always classified as stable up until the first contingency. From different combinations of OCs and contingencies, the system may then end up being in a stable state, an alert state in area **C1**, **C2**, or **C3**, or in an emergency state. For the N-1 case, the sequence of true target value vectors from the time of the contingency to the end of the simulation are classified depending on which of these five states the systems end up in. For the example of the N-1 case in Fig. 5, the system ends up in an alert state in the **C1** area. For the N-1-1 case, the target values are classified as stable up until the first contingency. The target values are then gathered from the N-1 case, using the end state of that simulation for classifying the state *between* the first and the consecutive contingency. After the second consecutive contingency, the system runs until it either collapses or until 560 seconds. Depending on this final state, the sequence of true target value vectors from the second contingency until the end of the simulation is classified. In the example in Fig. 5, an emergency state is reached. Note that the scales in Fig. 5 are different from those in the simulations for easier interpretation.

It should be noted that the classification of the different states (stable, alert, emergency) could be performed more intricately to satisfy other criteria of stability. For instance, these could be related to a minimum level of loadability of the system in its post-disturbance state. The loadability limit could then be computed by, for instance, parameterized continuation methods such as the continuation power flow (CPF) method [26], or by certain line indicators [27]. Other stability criteria could include the capability of the system in its post-disturbance state to handle yet another disturbance.

6) Reiteration: The described steps are reiterated until a sufficiently large training set is generated.
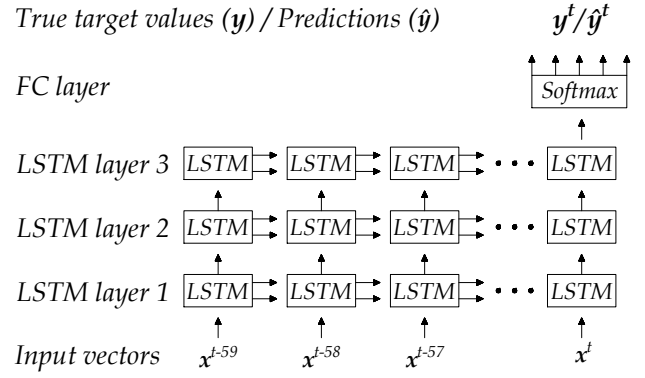
B. Architecture of the LSTM network

The proposed LSTM network architecture, shown in Fig. 6, is generally referred to as a "many-to-one" architecture, where previous measurements in the time sequence are used for the classification in the final block. The network consists of three stacked LSTM layers which are used to capture different levels of features from the inputs. Each LSTM block consists of 32 individual LSTM cells. The first layer of LSTM-blocks takes a generated sequence of input vectors as inputs; then by mathematical operation as presented in Section II, the output of each block is forwarded both to the following block in the sequence, as well as to the upper layer of LSTM-blocks. The inputs to the deeper layers consist only of the hidden states of LSTM blocks of previous layers, while both the hidden state and the cell state memory is passed along the time sequence between LSTM blocks of the same layer.

The LSTM network is designed to take sequences of 60 time steps of measurements as inputs. The internal architecture of each LSTM cell and functionality of the nonlinear gating units as presented in Section II, allows the LSTM network to fully utilize and pass forward the information from the first to the final time step in the sequence. The third layer of LSTM-blocks only passes the output forward along the time sequence. The output layer at time t is a fully connected network with softmax activation for classification. In training, the network uses the true target vector y^t at time t, while during the test or prediction phase, the network estimates a prediction vector $\hat{y}^t$ at time t. The interpretation of the prediction problem is further explained in section III-D.

C. Training the LSTM network

Different data sets were used for training, validation, and testing of the method on a mix of N-1 and N-1-1 cases. The training data set has the dimension $(135,000 \times 364 \times 560)$, where the dimension represents the number of training cases, the number of inputs, and the total interval in seconds for each simulation, respectively.

Before training, a process generally referred to as sequence preprocessing was performed to prepare batches of sequences with suitable length. The network is designed to take a sequence of 60 time steps of measurements as inputs and subsequences with a length of 60 time steps $\left(x^{t-59}, ..., x^t\right)$

TABLE I
DESIGN AND HYPERPARAMETERS USED IN TRAINING

	Parameter	Values and size
Data	Simulation interval	560
	Input dimension	364
	Input data type	$V_{mag}, V_\theta, P_{flow}, Q_{flow}$
	Target classes	5
	Training cases (N-1+N-1-1)	45,000 + 90,000
	Validation cases (N-1/N-1-1)	5,000 / 10,000
	Test cases (N-1/N-1-1)	10,000 / 10,000
Architecture	LSTM layers	3
	LSTM sequence length	60
	FC activation function	Softmax
	LSTM hidden cells	32
	LSTM Activation function	Tanh
Training	Max Epochs	500
	Learning rate (α)	0.0001
	Dropout & recurrent dropout	50 % / 50 %
	Optimizer	Adam [28]
	Loss metric	Categorical cross-entropy

were thus extracted from the 560 seconds long simulation intervals, for different values of t. For each subsequence of input vectors, a corresponding target value ($\boldsymbol{y}^t$) at time t was gathered. The sequence preprocessing was performed 120 times for *each* training and validation case by varying t between values of $t = [60, 180]$. The lower bound of t is required to always allow historic data to be included into the sequence. The LSTM network could have been trained on the whole simulation interval by increasing the upper bound of t from 180 to 560. However, since the method is proposed to be used in fast VIP applications, there is less usefulness of predicting instability long after the contingencies have occurred.

The generated subsequences were then used to train the LSTM network. Due to memory limitations, a method called mini-batch gradient descent was utilized where mini-batches of 1000 subsequences were used separately to train the network. The training was performed for a maximum of 500 epochs. An epoch is finished when all generated batches have been used to update the network parameters. Adam [28], an adaptable algorithm suitable for gradient-based optimization of stochastic objective functions was used in training the network. The algorithm used default parameters according to [28], except for the learning rate which was tuned. The loss function on which the optimizer is applied is the categorical cross-entropy function, which is suitable for multi-classification problems. To avoid overfitting the data, two regularization techniques were used during the training. First, early stopping was implemented, and the training of the network was stopped in case the performance on the validation set did not improve after six epochs. Second, a technique called dropout was applied, where a certain percentage of the connections between inputs and the LSTM cells were randomly masked (or "dropped") to reduce overfitting on the data. Both conventional dropout and recurrent dropout between consecutive blocks were applied during the training phase.

All other parameters related to the training of the network are presented in Table I. The LSTM network was trained and implemented in Python, using the Keras library with TensorFlow backend. The architecture and parameters used to train the network have been iteratively tuned to increase the classification accuracy. It should be noted that the tuning could be extended even further to allow an even better classification accuracy.

D. Interpretation and intuition of the VIP problem

By the proposed training and architecture of the LSTM network, a classification problem is solved where the *current* system state space is separated into different regions. Every state on a trajectory to a stable, alert (in **C1**, **C2**, or **C3**), or emergency state is labeled accordingly. The LSTM network is then trained on this data to implicitly learn these asymptotic properties of solutions and the trajectories of the system state. Once trained, the network can correlate the inputs, current and historic measurements, with a certain state-space region and trajectory, allowing warnings of voltage instability only moments after a contingency have occurred in a system. The classification is performed under the assumption that the current system is unchanged, meaning that no additional contingencies or changes in generation and load configuration will occur. However, as time progresses, new observations are used as inputs to the LSTM network to continuously update and incorporate such changes in the system.

This VIP problem should be interpreted as a fixed horizon prediction problem, where the prediction horizon always is the final state given by the trajectories of the (dynamical) system. This interpretation assumes that the simulation horizon of the generated time-domain simulations are sufficiently long so that extending the simulation horizon even further, for this particular system beyond 560 seconds, would not change the partitioning of the state space.

IV. RESULTS AND DISCUSSION

A. Test results

The developed VIP methodology was tested on two separate test sets, one containing only N-1 cases, the other containing N-1-1 cases. Each test set was composed of 10,000 cases of dynamic simulations. The test results of the predictions are presented using categorical accuracy, where the *indices* of the true target values are compared to the argument maxima of the predictions. The accuracy at *each* time step is then calculated over time for each of the two test sets.

The data were fed into the network in the form of a rolling window, with subsequences generated in the same manner as described in Section III-C. As time t progresses, new measurements entered the network from the rightmost block in the input layer and were shifted to the left in each time increment. Since the LSTM network require a sequence of 60 time steps of data, no predictions were made before $t = 60$. To facilitate the presentation in the following figures, a new time index T is introduced here. The relationship between the two time-indices is $T = t - 60$. The LSTM network's performance for VIP is not only tested during the short JAD state but during a longer period of the dynamic trajectory the system takes following the disturbances. This is performed to test the network's capability to incorporate new observations and improving its assessment as time progresses throughout

TABLE II
CONFUSION TABLE SHOWING PREDICTION RESULTS AND ACCURACY OF THE LSTM NETWORK EVALUATED AT $T = 50$ SECONDS

| | | Predicted states (N-1 / N-1-1) | | | | | |
| | | Stable state | | Alert state | | Emergency state | Accuracy |
	Classification	All areas	C1	C2	C3	All areas	
Actual states	**Stable state** All areas	2766 / 1147	0 / 36	0 / 11	0 / 8	0 / 8	100 / 94.8 %
	Alert state C1	0 / 0	856 / 562	0 / 3	0 / 0	0 / 5	100 / 98.6 %
	C2	0 / 5	0 / 5	1874 / 1222	0 / 0	0 / 109	100 / 91.1 %
	C3	0 / 0	0 / 0	0 / 12	0 / 208	0 / 0	- / 94.5 %
	Emergency state All areas	0 / 0	0 / 0	0 / 10	0 / 0	4504 / 6649	100 / 99.8 %
	Accuracy	100 / 99.6 %	100 / 93.2 %	100 / 97.1 %	- / 96.3 %	100 / 98.2 %	**100 / 97.9 %**

a voltage instability event. The classification accuracy is only plotted for 120 seconds after T to better visualize the changes in accuracy after the contingencies.

The classification accuracy *over time* is presented in Fig. 7. The classification accuracy for the N-1 test set dropped significantly at $T = 6$ seconds, which is the same instant that the first contingency is applied. The large drop in classification accuracy can be attributed to low bus voltages instantaneously following the first contingency, which the LSTM network has learned to correlate to a voltage instability event. After the first contingency, the classification accuracy increased and remained constant at 100 % for the rest of the simulations.

The classification accuracy for the N-1-1 test set was identical up until the time when the consecutive contingencies were randomly applied. During this time, illustrated by the arrows in Fig. 7, the classification accuracy decreased slightly. Since these contingencies do not occur at the same time instant in each test case, the same instant drop in accuracy as for the N-1 cases was not seen. The accuracy then gradually increased and stabilized at around 97-98 %.

The results show that the LSTM network can classify and predict future stability almost perfectly for the N-1 contingency cases and with good accuracy for the N-1-1 cases. To examine which cases were misclassified, the prediction accuracy for the two test sets, evaluated at $T = 50$ seconds, are presented in Table II in the form of a confusion table. Each number in the column in the table represents instances of the predicted classes and each number in the row represents the instances of the actual classes. The (empirical) conditional probabilities of correctly classifying a certain state is presented in the column furthest to the right. Similarly, the conditional probability of a state *actually* belonging to the predicted state is presented in the bottom row of the table. The total accuracy is presented in the lower right corner of the table. The accuracy for all N-1 cases is 100 % and no cases are falsely classified. For the N-1-1 test set, the lowest classification accuracy occurred for the alert states. After inspection of the falsely classified cases, it was found that several of these were borderline cases where the transmission bus voltage magnitude used in the classification were very close to what was used in the other classes. The highest classification accuracy occurred for the emergency cases with 99.8 %.

It should be noted that the test and training sets were weighted with more cases ending up in certain classes than others. It is thus probable that the results are slightly biased with higher accuracy for these classes, and that the classification accuracy of the other classes may be lower as an effect.

B. Impact of sequence length

In this section, the performance of the sequence-based approach is tested and compared against a conventional feedforward NN, which only uses a single snapshot of measurements as inputs. Further, to test the impact of a *shorter* time sequence, the results of an LSTM network using a time sequence of 30 time steps, instead of 60, are presented.

To allow a fair comparison between the two approaches, the feedforward NN used in this comparison was designed to be as similar as possible to the LSTM network. Essentially, the design of the NN in the comparison is identical to the *final* time step in the LSTM network presented in Fig. 6, with the difference that each layer consists of a hidden layer of neurons. The designed NN has thus three hidden unit layers, each layer with 32 hidden nodes. The same FC layer with a softmax activation function was used. The training for the NN was performed identically as for the LSTM network, with the exception that instead of a sequence of input values, a single snapshot was used. The LSTM network using a shorter time sequence was trained identically to that of the longer LSTM network with the exception that a shorter sequence of 30 instead of 60 time steps was used.

In Fig. 8, the classification accuracy on the N-1-1 test set is presented for the two LSTM networks with the different
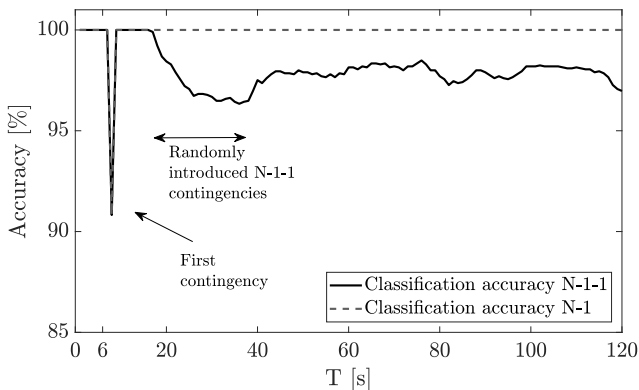

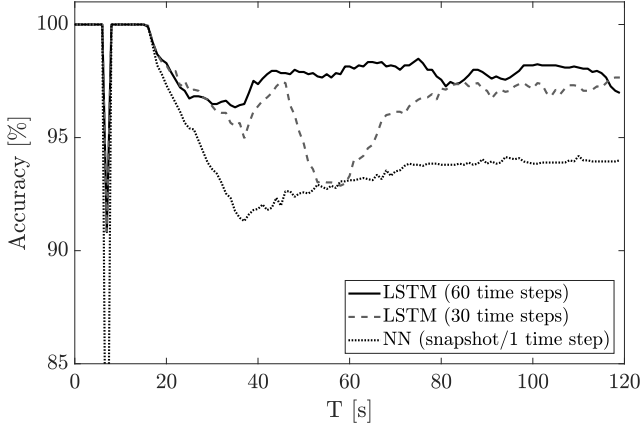Fig. 7. Classification accuracy over time for the proposed LSTM network

Fig. 8. Impact of sequence length on classification accuracy

Fig. 9. Classification accuracy over time for different values of Δt

time sequence length and for the conventional NN. The classification accuracy for the conventional NN was around 93 % after all the consecutive contingencies had been applied, while that of the proposed LSTM network is around 97-98 %. The results show that the performance of the LSTM network using 60 time steps in the sequence significantly exceeded that of the conventional NN, generally providing better classification accuracy over the whole time frame of the simulation cases.

The classification accuracy of the LSTM network using a shorter sequence was similar to the one using a longer sequence, with the difference of a large drop in classification accuracy occurring at around $T = 46$ seconds, see Fig. 8. The accuracy declined for 20 seconds and was then restored to around 97 % accuracy. A similar decline in classification accuracy, though less significant, can be noted for the LSTM network using the longer time sequence at $T = 76$. Thus, a decline in classification accuracy started exactly 60 respectively 30 seconds after the consecutive contingencies were introduced (at $T = 16$) for the two networks, corresponding to the network's respective sequence length. One explanation of these results is that the LSTM networks utilize information concerning the contingency and *pre-contingency* state to enhance the classification accuracy. When the networks starts to lose the information about the pre-contingency state, the chance of a misclassification increases. The results strengthen the hypothesis that a long sequence LSTM network could be used to enhance the state signal to provide better classification accuracy. Theoretically, an even longer sequence could be used to increase the accuracy even further. However, this would increase the computational cost of training, and a balance between classification accuracy and computational cost should be sought.

C. Impact of measurement update rate

The performance of the LSTM network is in this section tested for different values of the measurement update rate. The performance is compared between the previously assumed available measurement update rate of $\Delta t = 1s$ and the slower update rates of $\Delta t = 3s$ and $\Delta t = 5s$. Due to the slower
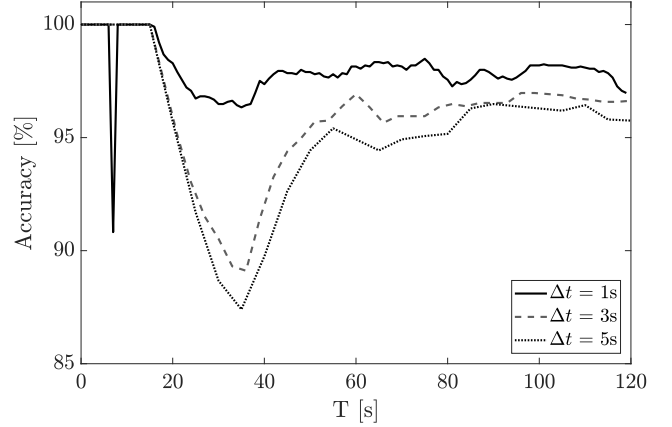
update rates, the architecture and the number of LSTM blocks along the time sequence had to be reduced accordingly. The original LSTM network was designed to take subsequences of 60 time steps of measurements as inputs. Thus, for the LSTM network adapted for $\Delta t = 3s$, the number of LSTM blocks along the time sequence was reduced to a third (20 blocks along the time sequence), while the number of blocks for the LSTM network adapted for $\Delta t = 5s$ was reduced to a fifth (12 blocks along the time sequence). The LSTM networks adapted for the new measurement update rates were then trained identically to the original LSTM network, with the difference that now only every third, respectively fifth, measurement in each generated subsequence were passed on the networks.

The classification accuracy for the different values of Δt is presented in Fig. 9 using the same N-1-1 test set as in previous sections. The results show that the performance when using a measurement update rate of $\Delta t = 1s$ exceeds those using a slower update rate. The largest difference can be identified during the period when the second consecutive contingencies are applied, which indicates that a lower value of Δt is especially valuable for classification during the short time that follows a disturbance. It should be noted that due to the slower update rates of Δt, there is no dip in the classification accuracy following the first contingency.

A larger value of Δt may also increase the *time* it takes to accurately predict instability, as new measurements are being passed less frequently to the LSTM network. In Table III, the average time, after a contingency, to accurately predict the future state of the system is presented for the different values of Δt. The average time is only presented for the time it takes to correctly classify the system states following the *second* consecutive contingency, since correct classification following the first contingency was almost instantaneous in all test cases. The time was computed as the averaged passed time after the second contingency, up until the time when the LSTM network could consistently and accurately predict the state of the system. For cases that took *longer* time than 100 seconds to be correctly classified, a detection time of 100 seconds was

TABLE III
AVERAGE TIME TO PREDICT THE ONSET OF VOLTAGE INSTABILITY

	Measurement update rates		
	$\Delta t = 1s$	$\Delta t = 3s$	$\Delta t = 5s$
Average prediction time [s]	6.6	8.7	10.7

assumed to avoid skewed averaged values.

The average time to correctly predict the system state was found to be 6.6 seconds for the proposed LSTM architecture using a measurement update rate of $\Delta t = 1s$. The corresponding values for the LSTM networks using the slower update rates of $\Delta t = 3s$ and $\Delta t = 5s$, were 8.7 seconds and 10.7 seconds, respectively. The longer time longer time to accurately predict instability for the slower update rates of Δt can be attributed partly to a lower classification accuracy, and partly to the fact that measurements are being updated less frequently.

D. Generalization capability and training set requirement

The generalization capability of a ML method refers to the capability to generalize the learning from the actual training set to other, yet unseen, cases. Such capability is especially valuable in overcoming the combinatorial increase of complexity in the training when N-1-1 cases are also considered [29].

In Fig. 10, the classification accuracy is presented on the N-1-1 test set when the LSTM network have been trained on three different training sets. The results are presented when the network was trained on i) the full training set with all N-1 and N-1-1 cases included, ii) a smaller training set with all N-1 cases but where only a small batch $(5,000)$ of N-1-1 cases have been included, and iii) a training set where the network is *only* trained on N-1. The same training approach as previously described were used. According to Fig. 10, the classification accuracy was significantly reduced when no N-1-1 cases are included in the test set. When including the small batch $(5,000)$ of N-1-1 cases, the classification accuracy increased significantly. However, the accuracy is still lower than when the full training set is used. Thus, the importance of obtaining a representative training set is still imperative if a high classification accuracy is to be achieved.

E. Practical applications and requirements

The method is proposed to be used as an online tool for system operators to monitor the current state of a power system. It should be stressed that the method is not proposed to replace conventional voltage instability detection methods, but rather function as a supplementary tool to provide early warnings. The instantaneous prediction capability of the proposed method has to be weighed against the possibility of misclassification of the system's future stability. When comparing the proposed method to other conventional indicators for voltage instability detection (see [2]), it is important to remember that these might be more accurate once instability detected, but generally take *significantly* longer time to indicate instability,
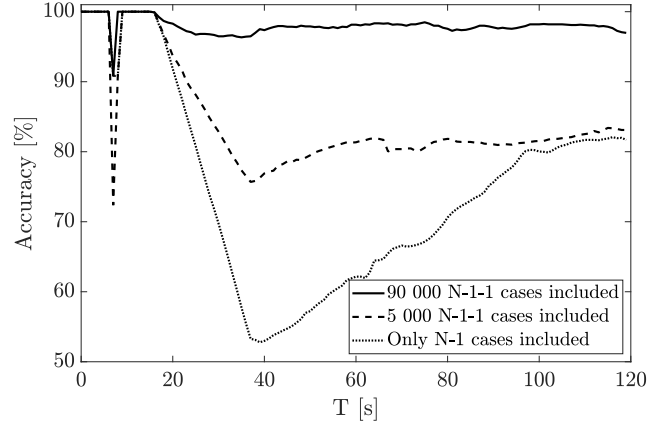

Fig. 10. Classification accuracy over time when varying the number of N-1-1 cases included in the training data

thus reducing the time frame that system operators have to steer the system back into stable operation.

The proposed method is mainly intended for predicting mid-term or long-term voltage instability where system operators will have the possibility to act on the warnings provided by the network. Theoretically, the method could be adapted to also handle short-term voltage instability. However, this would require more frequent measurement updates to ensure that the onset of short-term instability is detected in time. Because of the difference in the dynamical trajectories of the system for the two different types of instability events, training a separate LSTM network would likely provide better performance. Furthermore, the signals provided by the network would have to automatically trigger emergency controls, since the available time for system operators to act on the signals would be too short for manual control actions.

For the proposed method to be effective in long-term voltage instability, measurement updates should be available within a few seconds. In this paper, a measurement update rate each second have been assumed to be available. As was found in Section IV-C, slower measurement update rates lead both to lower classification accuracy and slower predictions. To assure that errors and missing values are filtered out, measurements should always be preceded by a state estimator. However, state estimates from a non-linear state estimator based on remote terminal units may be too slow to be effective. Thus, time-synchronized measurements from wide-area phasor measurements filtered through a linear state estimator would be preferred.

The softmax classifier of the LSTM network outputs a probability vector, where each class is given a certain probability. It should be noted that this probability vector does not provide a *true* representation of the model confidence. However, it can still be useful as a proxy by system operators to track the network's confidence in each prediction. Thus, the operator can use the probability vector directly in an online interface to track the network's belief in each prediction. Alternatively, argument maxima or other functions could be used to present the most probable prediction of the network, or, for instance,

to avoid predictions of falsely labeled stable states.

The *practical* classification accuracy of the proposed method will be affected by many aspects and will generally be lower than on a simulated test set. One of the more important aspects are modeling errors, including erroneous system parameters or inaccurate modeling of parameter values for dynamic models. Such aspects will introduce a difference between the simulated and the actual dynamic response after a contingency. However, it should be noted that such limitations are not limited only to ML based approaches for VIP. All methods for DSA require that the dynamic models used in assessing the system response are accurately modeled.

High penetration of variable generation may also impact the proposed ML methodology; primarily by increasing the combinatorial difficulty in generating a representative training set. To reduce the dimensionality of the state space that the power system can operate in, it is thus required that smaller variable generation units can be merged and modelled as equivalent units. Such simplifications are already to an extent required in the modelling of large power systems, where underlying distribution grids are often reduced to single load buses. However, it is important to verify that such simplifications of the simulation models do not affect the dynamical trajectories the system can take under the different contingency scenarios.

V. CONCLUSIONS AND FUTURE WORK

This paper presents a new approach for online voltage instability prediction using an LSTM network capable of utilizing a sequence of measurements to improve classification accuracy. Once trained, the LSTM network can allow system operators to continuously assess and predict whether the present system state is stable, or will evolve into an alert or an emergency state in the near future. The network is also adapted to be able to indicate *where* instability emerges, allowing system operators to perform more cost-effective control measures.

The LSTM network was proposed to improve the available state signal by implicitly learning the dynamical trajectories of a power system following a disturbance. The LSTM architecture and the operation of the gating units ensure that the network is capable of capturing the long-term dependencies that are common in voltage instability events. The results presented in the paper are highly encouraging and the proposed method is shown to have high accuracy in predicting voltage instability only seconds after a disturbance.

The impact of sequence length of the LSTM network was tested and showed that a longer sequence provided a significantly better classification capability than both a feedforward NN and a network using a shorter sequence. The paper also examined the generalization capability of the proposed LSTM network, where the classification accuracy on N-1-1 cases was assessed when the system was only trained on N-1 cases. It was found that this reduced the classification accuracy significantly, whereas including a smaller subset of N-1-1 cases into the training set resulted in significantly better performance.

Future research work includes examining the impact that emergency control have on the prediction accuracy of the method. Although emergency control will generally be activated too slowly to affect the system in the time frame it takes for the LSTM-network to predict instability, it will affect the following predictions after the control actions have been issued. If such control aspects are included in the training scenarios, the LSTM network could also be used to assess whether the emergency actions indeed succeeded in restoring the system to a stable state again, or if further actions would be required. Moreover, further studies are required in examining how the generalization capability of the LSTM network can be improved. Methods to also evaluate the accuracy of the predictions, such as providing a confidence estimate of the predictions, would also be valuable to ensure the robustness of the method.

REFERENCES

[1] P. Kundur *et al.*, "Definition and classification of power system stability IEEE/CIGRE joint task force on stability terms and definitions," *IEEE Trans. Power Syst.*, vol. 19, no. 3, pp. 1387–1401, Aug 2004.

[2] M. Glavic and T. Van Cutsem, "A short survey of methods for voltage instability detection," in *Proc. (IEEE) PES General Meeting*, Detroit, MI, Jul 2011, pp. 1–8.

[3] I. Konstantelos *et al.*, "Implementation of a massively parallel dynamic security assessment platform for large-scale grids," *IEEE Trans. Smart Grid*, vol. 8, no. 3, pp. 1417–1426, May 2017.

[4] T. Van Cutsem *et al.*, "Decision tree approaches to voltage security assessment," *IEE Proceedings C - Generation, Transmission and Distribution*, vol. 140, no. 3, pp. 189–198, May 1993.

[5] Y. Mansour *et al.*, "Large scale dynamic security screening and ranking using neural networks," *IEEE Trans. Power Syst.*, vol. 12, no. 2, pp. 954–960, May 1997.

[6] K. Sun *et al.*, "An online dynamic security assessment scheme using phasor measurements and decision trees," *IEEE Trans. Power Syst.*, vol. 22, no. 4, pp. 1935–1943, Nov 2007.

[7] H. Khoshkhoo and S. M. Shahrtash, "Fast online dynamic voltage instability prediction and voltage stability classification," *IET Generation, Transmission & Distribution*, vol. 8, no. 5, pp. 957–965, May 2014.

[8] C. Liu, F. Tang, and C. L. Bak, "An accurate online dynamic security assessment scheme based on random forest," *Energies*, vol. 11, no. 7, 2018.

[9] C. Vournas and T. Van Cutsem, "Online voltage security assessment," in *Real-Time Stability in Power Systems - Techniques for Early Detection of the Risk of Blackout*. Springer, Jul. 2014, ch. 10, pp. 305–333.

[10] R. F. Nuqui, "State estimation and voltage security monitoring using synchronized phasor measurements," Ph.D. dissertation, Virginia Tech, 2001.

[11] R. Diao *et al.*, "Decision tree-based online voltage security assessment using PMU measurements," *IEEE Trans. Power Syst.*, vol. 24, no. 2, pp. 832–839, May 2009.

[12] H. Khoshkhoo and S. M. Shahrtash, "On-line dynamic voltage instability prediction based on decision tree supported by a wide-area measurement system," *IET Generation, Transmission & Distribution*, vol. 6, no. 11, pp. 1143–1152, November 2012.

[13] H. Nguyen Duc *et al.*, "A novel approach for early detection of impending voltage collapse events based on the support vector machine," *International Transactions on Electrical Energy Systems*, vol. 27, no. 9, p. e2375, 2017.

[14] M. L. Scala, M. Trovato, and F. Torelli, "A neural network-based method for voltage security monitoring," *IEEE Trans. Power Syst.*, vol. 11, no. 3, pp. 1332–1341, Aug 1996.

[15] H. Hagmar *et al.*, "On-line voltage instability prediction using an artificial neural network," in *2019 IEEE Milan PowerTech*, June 2019, pp. 1–6.

[16] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts London, England: The MIT Press, 2015.

[17] P. Geurts and L. Wehenkel, "Early prediction of electric power system blackouts by temporal machine learning," in *Proceedings of the ICML98/AAAI98 Workshop on "Predicting the future: AI Approaches to time series analysis*, 1998, pp. 24–26.

[18] L. Wehenkel *et al.*, "Probabilistic design of power-system special stability controls," *Control Engineering Practice*, vol. 7, no. 2, pp. 183–194, 1999.

[19] P. Geurts and L. Wehenkel, "Temporal machine learning for switching control," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2000, pp. 401–408.

[20] K. Greff *et al.*, "LSTM: A Search Space Odyssey," *IEEE Trans. Neural Netw. & Learning Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct 2017.

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, p. 533, 1986.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[23] T. Van Cutsem *et al.*, "Test systems for voltage stability analysis and security assessment," IEEE/PES Task Force, Tech. Rep. PES-TR19, Aug. 2015. [Online]. Available: http://resourcecenter.ieee-pes.org/pes/product/technical-publications/PESTR19

[24] *PSS®E 34.2.0 Model Library*, Siemens Power Technologies International, Schenectady, NY, Apr. 2017.

[25] F. Thams *et al.*, "Efficient database generation for data-driven security assessment of power systems," *IEEE Trans. Power Syst.*, pp. 1–1, 2019.

[26] V. Ajjarapu and C. Christy, "The continuation power flow: a tool for steady state voltage stability analysis," *IEEE Trans. Power Syst.*, vol. 7, no. 1, pp. 416–423, Feb 1992.

[27] S. Yari and H. Khoshkhoo, "A comprehensive assessment to propose an improved line stability index," *International Transactions on Electrical Energy Systems*, vol. 29, no. 4, p. e2806, 2019.

[28] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints*, p. arXiv:1412.6980, Dec 2014.

[29] P. Mitra *et al.*, "A systematic approach to n-1-1 analysis for power system security assessment," *IEEE Power and Energy Technology Systems Journal*, vol. 3, no. 2, pp. 71–80, June 2016.

Le Anh Tuan (S'01–M'09) received the M.Sc. degree in energy economics from the Asian Institute of Technology, Bangkok, Thailand, in 1997, and the Ph.D. degree in power systems from the Chalmers University of Technology, Gothenburg, Sweden, in 2004. He is currently a Senior Lecturer with the Division of Electric Power Engineering, Department of Energy and Environment, Chalmers University of Technology. His current research interests include power system operation and planning, power market and deregulation issues, grid integration of renewable energy, and plug-in electric vehicles.

Hannes Hagmar (S'17) received the M.Sc. degree in electric power engineering from Chalmers University of Technology, Gothenburg, Sweden in 2016. Between 2016 to 2017, he worked at RISE Research Institutes of Sweden with research in electric transmission systems and measurement technology. He is currently pursuing the PhD degree at Chalmers University of Technology. His research interest includes power system dynamics and stability, integration of renewables, and machine learning.

Lang Tong Lang Tong (S'87,M'91,SM'01,F'05) is the Irwin and Joan Jacobs Professor of Engineering at Cornell University and the Cornell site Director of Power Systems Engineering Research Center (PSERC). He received the B.E. degree from Tsinghua University and the Ph.D. degrees in electrical engineering from the University of Notre Dame. He was a Postdoctoral Research Affiliate at the Information Systems Laboratory, Stanford University held visiting positions at Stanford University, the University of California at Berkeley, the Delft University of Technology, and the Chalmers University of Technology in Sweden. Lang Tong's current research focuses on optimization, machine learning, AI, and economic problems in energy and power systems. He received several IEEE society transaction prize papers and conference best paper awards. He was a Distinguished Lecturer of the IEEE Signal Processing Society and the 2018 Fulbright Distinguished Chair in Alternative Energy.

Robert Eriksson (SM'16) received the M.Sc. and Ph.D. degrees in electrical engineering from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2005 and 2011, respectively. He held an associate professor position at the Center for Electric Power and Energy, DTU Technical University of Denmark, from 2013 to 2015. He is currently with the Swedish National Grid, Department of Markets and System Development. His current research interests include power system dynamics and stability, automatic control, HVDC systems, and DC grids.