Instability Prediction in Power Systems using Recurrent Neural Networks

Ankita Gupta, Gurunath Gurrala, Pidaparthy S Sastry
Indian Institute of Science, Bangalore
{ankitagupta, gurunath, sastry}@ee.iisc.ernet.in

Abstract

Recurrent Neural Networks (RNNs) can model temporal dependencies in time series well. In this paper we present an interesting application of stacked Gated Recurrent Unit (GRU) based RNN for early prediction of imminent instability in a power system based on normal measurements of power system variables over time. In a power system, disturbances like a fault can result in transient instability which may lead to blackouts. Early prediction of any such contingency can aid the operator to take timely preventive control actions. In recent times some machine learning techniques such as SVMs have been proposed to predict such instability. However, these approaches assume availability of accurate fault information like its occurrence and clearance instants which is impractical. In this paper we propose an Online Monitoring System (OMS), which is a GRU based RNN, that continuously keeps predicting the current status based on past measurements. Through extensive simulations using a standard 118-bus system, the effectiveness of the proposed system is demonstrated. We also show how we can use PCA and predictions from the RNN to identify the most critical generator that leads to transient instability.

1 Introduction

Early classification of time series data is crucial in many time-sensitive applications in medicine, environment, business etc. (See [Griffin and Moorman, 2001] for an interesting example of this in medical domain).

Recurrent Neural Networks (RNNs) are, in general, good at capturing temporal dependencies in data and hence are effective in many time-series analysis applications [Hüsken and Stagge, 2003]. A recently proposed architecture of RNN uses the so called Gated Recurrent Units (GRUs) which are good at capturing long range temporal dependencies [Cho *et al.*, 2014]. In this paper we explore application of GRU based RNN for an interesting early prediction application, namely, that of predicting the transient instability of a power system following a disturbance.

An operating power system is subjected to various disturbances like switching on and off of circuit elements, or occurrence & clearing of faults such as a line shorting to ground. The ability of a power system to regain its stable operating condition and maintain synchronism[1] when subjected to such a disturbance is referred to as the transient stability. Based on the severity of these disturbances and prior system operating conditions, the power system may loose synchronism which can lead to possible cascading effects causing system collapse. Hence, monitoring the stability status of a power system in real time is a task of primary importance. In such a scenario, early recognition of the potentially dangerous conditions is very crucial for allowing sufficient time to take emergency control actions.

Various techniques based on time domain simulations, transient-energy-functions and equal area criteria have been traditionally used to assess the transient stability of power system [Kundur *et al.*, 1994; Pai, 2012; Xue *et al.*, 1992].

However, these methods demand very accurate information about network configuration and are computationally intensive with high time complexity.

Since the advent of Phasor Measurement Units (PMUs), over the last decade, near-real-time data of system variables is now available in many modern power systems. This led to many machine learning (ML) techniques being explored for complex power system stability and control problems. (See, for example, [Jensen *et al.*, 2001; Kamwa *et al.*, 2009]).

An SVM based classifier is proposed for post-fault transient instability prediction in [Rajapakse *et al.*, 2010]. Similarity values of measured voltages with pre-identified voltage variation trajectory templates are the inputs to SVM. In [Gomez *et al.*, 2011], the SVM classifier directly uses sampled values of the measured voltage magnitudes at the generator buses to predict the transient stability status. In [Ji *et al.*, 2016], global trajectory cluster feature subset is extracted using RELIEF algorithm. These features are used as SVM inputs. SVM-Based Ensemble classifier is proposed in [Zhou *et al.*, 2016] for transient stability prediction. These SVM based methods are seen to be more effective in assessing transient stability compared to all earlier ML techniques used for this

[1]Synchronism refers to the operation of all generators in an interconnected power system at a common voltage level, frequency and phase sequence. Loss of synchronism occurs when relative motion between generator rotors is fluctuating widely.

problem.

However, all these approaches assume availability, at the common control center, of accurate information of the fault such as instance of fault clearance. This is because, the classifier needs, as input, the values of system variables starting from the instant when the fault is cleared. Acquiring such fault-related information involves extensive instrumentation throughout the power system network (e.g. PMUs and protection relays). Even then, information such as the instance of fault clearance would have some random errors due to relay-tripping-time and communication delay. As we show here, the performance of the SVM is poor when tested on data measurements delayed even by a few samples with respect to fault clearance. Thus, there is a need for a more robust system whose performance is independent of accurate fault information, specifically, the instance of fault clearance.

The stacked GRU based RNN that we propose in this paper is a continuous Online Monitoring System (OMS) for assessment of transient stability of a power system. At each instant, based on the measurements over a few previous instants, the RNN outputs a classification of the current system state which, in turn, is used to predict transient instability. The system does not need any information regarding the faults (or any other disturbance in the system). The RNN is trained on system trajectories and essentially it learns the characteristics of the normal and abnormal variations in measured system variables. We use detailed simulation of a standard 118-bus system [Dehghanian *et al.*, 2015] to generate system voltage trajectories under normal operation as well as under different faults and these are used to train the RNN. Through extensive simulations we show that the trained RNN is very effective in early prediction of transient instability. We also show that it is fairly robust to measurement noise and other similar inaccuracies. In addition, we also present a method, based on PCA and some output of the RNN, to predict the critical generator in cases where there is instability. This is useful for the operator to plan an isolation operation to mitigate the effect of instability following a fault.

The remainder of this paper is organized as follows. In section 2 we describe the OMS and explain how the training set is generated and how the prediction on instability is obtained. In section 3 we describe the architecture of the GRU-based RNN that is used as OMS here. In section 4 we discuss simulation results and show the effectiveness of the proposed OMS and then conclude the paper in section 5.

2 The Proposed Online Monitoring System

The objective of the online monitoring system (OMS) is early prediction of transient instability of the power system following a disturbance such as a fault. Since disturbances occur at random times and we do not assume any knowledge about the faults, the OMS has to continuously keep analyzing the power system variables over a few cycles. (For a power system operating at 50 Hz, each cycle is of 20ms duration).

At any given instant, the OMS observes all the variables over a sliding window of, say, s cycles. The OMS is a Recurrent Neural Network (RNN) whose inputs are the voltages etc. of all the generators in the system. (See Sec. 2.1 for more details). The final objective is to predict instability. However, instead of making the output of RNN binary, we formulate it as a 5-class classification problem. That is, we train the RNN so that, on observing the variables in a time window, it predicts a system state to which this window belongs.

We explain in the next subsection these five states/classes that we define and the way in which we create the needed training data. The output of the RNN is finally used to generate a prediction about instability. Making the RNN into such a 5-class classifier enables it to learn the natural variations in the power system variables during different phases of a disturbance and to distinguish between such variations in the cases where the system regains a stable state and the cases where the system becomes unstable.

2.1 Preparation of Training Data

For training (as well as testing) our neural network, we need data in the form of temporal profiles of the power system variables in different scenarios. For this we use a simulation of the IEEE 118-Bus system.

This system has been widely used as a benchmark system for testing stability enhancement applications. This system comprises 118 buses, 19 generating units, 91 loads, and 177 transmissions lines.

Data is generated through offline dynamic simulations using MATLAB [Gurrala *et al.*, 2017]. In the simulations, we impose three-phase-to-ground faults on each bus as well as on each transmission line at three locations (at 25%, 50%, and 75% of the length). Clearing time for all the contingencies is randomly picked from 4-8 cycles. The above contingencies were repeated at three different loading levels (base load plus 5%, 7%, and 10%). The simulator records the variations over time of generator voltages (as complex quantities) over a large time window spanning both pre and post contingency period. A sampling frequency of 50 Hz is used to obtain synchronously sampled measurements. Each simulation is carried out for a time duration of 1000 cycles (or 20s).

The inputs to the RNN are magnitudes and phase angles of voltages at generator buses and the derivative of phase angles. The phase angles give information about rotor angles and their derivatives give information about frequency.

Using the simulator we generate a number of temporal profiles of these dynamic variables of the generators. We now explain how we label each profile as stable/unstable and how we prepare overlapping windows of data for training the RNN.

Transient Stability Index

Suppose that a transient disturbance occurs at time instance t_F and is cleared at t_C. The system variables are observed until a later time $t_M > t_C$. Then, theoretically, the stability status of the system, post-contingency, is obtained using Transient Instability Index, (η), defined as [Gomez *et al.*, 2011].

$$\eta = \frac{360^0 - |\Delta\delta|_{max}}{360^0 + |\Delta\delta|_{max}} \tag{1}$$

where $|\Delta\delta|_{max}$ is the absolute value of the maximum angle of separation between any two generators during the post-fault period, $\{t : t_F < t < t_M\}$. System is considered as stable if $\eta > 0$; otherwise, the system is transiently unstable. We label

Table 1: Possible System States

No Disturbance	Class 0
Fault Occurrence	Class 1
Fault Duration	Class 2
Fault Clearance	Class 3
Alert	Class 4

the system profiles obtained through our simulator as stable or unstable based on the η value as defined above.

Remark: Based on η, the stability status can be judged, provided the post-fault system is observed for a long time.

The objective is to predict this sufficiently in advance. Thus, while η is useful to label each profile, it cannot be used in practice for prediction of impending instability.

Let (X_i, Y_i), $i = 1, \cdots, N$, be the dataset where X_i are generated by the simulator and (Y_i) is 0/1 depending on sign of η. Each X_i is a time series of length 1000 where each element is a 57-dimensional vector (consisting of the magnitude, phase angle and derivative of phase angle of voltages of 19 generators).

We consider windows of size *s*, with stride of 1 cycle, over entire time horizon. A window is specified as *w* = $[t_S, t_E]$, where t_S and t_E are the start and end times of the window and s = t_E-t_S+1. Each profile, X_i, is converted into a sequence of overlapping windows.

We wish to label each window based on what is happening in the system during that window. Accordingly, we propose five classes as described in Table 1. If the window covers the instant of fault occurrence, it would be labelled as Class 1. All the windows in the stable pre-fault operation part would be labelled as Class 0. If a window sees all samples after fault occurrence but before fault clearance, it would be labelled as Class 2. If (in the current profile) the system is stable after fault clearance, we categorize all windows after fault clearance as Class 0 again; otherwise, all windows after fault clearance would be labelled as *Alert* or Class 4.

In the simulation we know the instant of fault onset, fault clearance etc. and this information would be used for labelling the windows in the training data. Let t_F and t_C denote the onset and clearance times of a fault in a profile. Then, a window *w* = $[t_S, t_E]$, is labelled as y_w as given below:

$$y_w = \begin{cases} 0 & \text{if } (t_E \leqslant t_F) \text{ or } (t_S \geq t_C \text{ and } \eta > 0) \\ 1 & \text{if } (t_S \leqslant t_F \leqslant t_E) \\ 2 & \text{if } (t_F \leqslant t_S \leqslant t_E \leqslant t_C) \\ 3 & \text{if } (t_F \leqslant t_S \leqslant t_C \leqslant t_E) \\ 4 & \text{if } (t_S \geq t_C \text{ and } \eta < 0) \end{cases} \tag{2}$$

Data in the form of windows with these labels is used to train the RNN. It is easy to see that we would have many more windows of class 0 than other classes. To avoid such class imbalance in training set, we used subsampling where needed.

Fig. 1 shows some typical profiles for stable and unstable cases. The instants $t_C + 1$ and $t_C + 6$ are marked in the figures, where t_C is the instant of fault clearance. We normally need to predict within about 4-7 samples from t_C. While the instability is easy to see if we wait long enough, predicting within the interval shown is what is challenging.

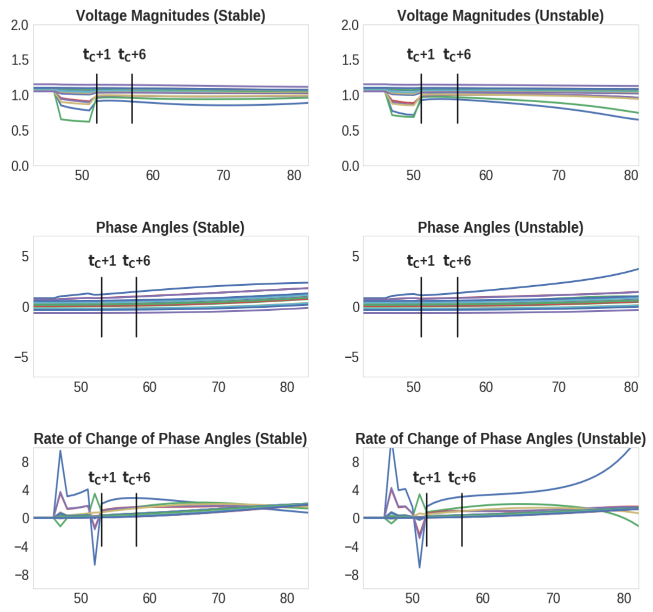

Figure 1: Variations of measured quantities for stable and unstable cases

The final output of OMS

The OMS is required to identify the cases that are going to be eventually unstable. We could predict instability as soon as our RNN classifies a window as Class 4. We have a softmax layer at the output of RNN and hence, normally, we would predict Class-4 if its probability is the highest. In a system that predicts instability, reducing false alarms is very important. Hence, we predict instability at the first instant where the probability of Class-4 (as given by the softmax layer of RNN) is above 0.99. Till the time the OMS predicts instability, we take it that it is predicting the system to be stable. While the threshold is arbitrary, we keep it high because even at this level our missed detection rate is zero.

2.2 Assessing OMS Performance

Since the final output of OMS is a stability/instability prediction, we assess it using false alarm and missed detection rates. In this application, we want zero missed detections and want to reduce false alarm rate as much as possible. This is one assessment of OMS performance that we present in our experiments section.

In this application, we also want to measure how early the OMS is able to predict. We can measure this by looking at the instance of instability prediction by OMS on the i^{th} profile, T_A^i, relative to the time instant where we anyway know the system to be unstable and this can be T_η^i, the instant at which the transient stability index, η, becomes negative. We define the *average latency of prediction*, Υ, by

$$\Upsilon = \frac{1}{M}\sum_{i=1}^{M}(T_\eta^i - T_A^i) \tag{3}$$

where M is total number of profiles in test set. In our experiments section we indicate this assessment of OMS also.

The average latency of prediction is, in general, important in any application where the goal is early prediction. However, in many applications, it may not always be easy to objectively identify an instant, such as T_η^i here, relative to which the early prediction can be assessed.

2.3 Identification of Critical Generator

The main utility of early prediction of transient instability is that the operator can take corrective action such as isolation of the generator that is likely to loose synchronism from rest. Hence, it is desirable to also identify the critical generator.

For this we consider the data between the first instant, say, t_F, when OMS predicts class-1 (which is onset of a fault) and the instant, say, t_A, when OMS predicts instability. We consider measurements over all these instants for each of the generators. Since the critical generator(s) is one that is going out of synchronism with the rest, we should be looking for some 'outliers' based on the data. We do that as follows.

For each generator, we have a data vector of dimension $3(t_A - t_F)$. Using PCA on the training data we obtain the first two principal directions. Now on any profile where we make an instability prediction, for each generator its data vector (of dimension $3(t_A - t_F)$) is projected onto $\Re^2$ using the first two principal components. If there are G generators, this gives us G number of 2-dimensional vectors. A two dimensional gaussian density is fitted to the projected data with its parameters estimated by Sample Mean and Minimum Covariance Determinant (MCD) [Rousseeuw and Driessen, 1999]. MCD is a robust estimator of covariance matrix, which is not affected much by outliers. The generator (vector) with minimum likelihood under the estimated density is identified as the outlier and hence as the critical generator.

3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are well-suited for analyzing time series data due to their recurrent connections. Consider a single hidden layer network with x_t, h_t and y_t denoting the input, hidden and output layer neuron outputs. Then a general recurrent network can be specified as

$$h_t = \sigma(W^{hh}h_{t-1} + W^{hx}x_t) \tag{4}$$

$$y_t = \text{Softmax}(W^S h_t) \tag{5}$$

where, W^{hh}, W^{hx} and W^S are the weight matrices across different connections and $\sigma(\cdot)$ denotes the sigmoid function. The output is obtained through a softmax because we are considering a classification scenario.

3.1 Stacked GRU Based RNN

Although theoretically RNNs can model arbitrarily long memories (that is, dependence on inputs from remote past), in practice this is difficult due to the problems associated with backpropagation through time (BPTT) over many time steps [Pascanu *et al.*, 2013]. Some of the specialized RNNs proposed are Long Short Term Memories (LSTMs) [Hochreiter and Schmidhuber, 1997] and Gated Recurrent Units (GRUs) [Cho *et al.*, 2014]. They are designed to have more persistent memory, hence making it simpler for RNNs to capture long-term dependencies. We preferred the GRUs owing to lesser number of trainable parameters as compared to LSTMs (2 gates versus 3 gates). As earlier, let x_t be the input and h_t be the output of a hidden layer. A GRU based cell computes h_t through the following steps.

Table 2: Model Parameters for RNN based OMS

Parameters	Description
Batch Size	32
Number of Hidden Layers	2
Hidden Dimensionality	128
Optimizer	Adam [Kingma and Ba, 2014]
Learning Rate	Adaptive (Default Adam)

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1}) \tag{6}$$

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}) \tag{7}$$

$$\tilde{h}_t = \tanh(r_t \circ Uh_{t-1} + Wx_t) \tag{8}$$

$$h_t = (1 - z_t) \circ \tilde{h}_t + z_t \circ h_{t-1} \tag{9}$$

The new memory $\tilde{h}_t$ is a summarization of new input x_t and past hidden state h_{t-1}. The reset signal r_t decides how relevant is h_{t-1} to the computation of new memory. The h_t is a convex combination of $\tilde{h}_t$ and h_{t-1} with the relative weights decided by the update gate z_t. Trainable parameters are the weight matrices $U^{(z)}$, $W^{(z)}$, $U^{(r)}$, $W^{(r)}$, U, and W. These are learnt using BPTT algorithm. (The symbol $\circ$ denotes element-wise multiplication and $\sigma(x) = \frac{1}{1+e^{-x}}$).

3.2 RNN Architecture for OMS

We use a 2 layer stacked-GRU based RNN architecture implemented in TensorFlow 0.8 with GPU (CUDA 7.5) support. As mentioned earlier, the input is a window of size s. We train the network with Adam Optimizer [Kingma and Ba, 2014]. We employ Cross Entropy loss and a *Softmax* output layer. Other details of the architecture are mentioned in Table 2.

4 Experimental Results

4.1 The Experimental Set-up

As described in Section 2.1, we generate a large number of temporal profiles of the power system variables for the IEEE-118 Bus System. Each profile comprises of voltage magnitudes, voltage phase angles and derivative of phase angles for all the 19 generators for 1000 time instants.

A total of 6046 profiles are generated using the simulator out of which 592 are unstable based on η. We then randomly split this data into training and test sets with one-tenth of the data kept as test set. As the RNN takes overlapping windows as inputs, we cut each profile in the training set into overlapping windows and label them according to Eq.(2). We consider a window size of 5 samples. Some windows are used to train the RNN and the rest are used for validation. For each test profile, the overlapping windows are fed to RNN and its output noted. If at any time the probability with which Class-4 is predicted (which is the output of that node in the softmax layer) is greater than 0.99, the corresponding profile is predicted unstable by OMS. If for the entire duration of a profile, the probability of Class-4 does not go above 0.99 then that profile is predicted stable by the OMS.

We perform the random split of data into training and test sets ten times and all results shown are averages over these trials. Every time we generate a test set, we randomly choose 60 unstable cases and 540 stable cases to form the test set. The remaining cases form the training set. In our results we report the false alarm (FA) rate, missed detection (MD) rate and also the average latency of prediction (cf. Eq.(3)). We also show effectiveness of our system under scenarios where measurements are corrupted with noise and where system topology changes due to some transmission lines being down.

4.2 Performance of the SVM Classifier That Needs Fault Information

Many ML approaches have been used for transient stability prediction. A good comparative analysis among all of them is given in [Zhou *et al.*, 2016], where it is shown that the SVM based method proposed by [Gomez *et al.*, 2011] and ensemble of SVMs performs the best. Here we present results obtained with SVM method on our data. As mentioned in section 1, all these methods assume that we know instance of fault clearance, t_C.

The inputs to the SVM are the generator voltages over 4-5 cycles *immediately* after t_C. However, in a practical scenario, we may not know t_C exactly because, for example, the protective relay may report it with an error of up to two samples. Hence, we analyze the SVM performance under the cases of (a) when exact t_C is known. (b) when t_C is known with 1 sample delay. (c) when t_C is known with a random error of upto 2 samples. The industry-grade PMUs often have some measurement noise. Hence we also analyze SVM performance under 1% and 2% noise in measurements. We consider the cases of noise in both training and test data as well as noise in test data only (with noise-free training data).

Table 3 shows the performance of SVM for all these cases. As can be seen from the table, the performance of the SVM degrades very significantly when there is error in t_C and/or when there is noise in measurements. Thus the existing ML approaches based on training the system only on a few cycles of data immediately following clearance of fault, are brittle and are not very attractive for more realistic scenarios.

We note here that our OMS does not need any information about t_C and hence there are no existing methods with which we can compare it directly. We present performance results of OMS in the next few subsections which show that it is much better than the SVM even though we do not need t_C. It is also observed in our simulations that in all cases the OMS predicts instability in about 4-7 cycles from t_C.

4.3 OMS Performance

We test the performance of the OMS under zero noise as well as under ±1% to ±3% uniform noise. We add the noise to both the training as well as test data.[2] These results are shown in Table 4. As can be seen from the table, even at 3% noise, our missed detection rate remains zero though our false alarm rate slightly increases. (We have a very small missed detection rate under no noise. This is because of only one out of 60 unstable cases missed in only one of the ten random repetitions of training and test splits).

We term the above noise which is added in every measurement at every instant as Type I noise. We also consider what we call type II noise where at any given time instant each generator has probability 0.1 of its measured values being noisy and when they are, the noise is uniform with rate varying between ±1% and ±3%. The missed detection rate remains at zero in all cases of type II noise also showing our method is very reliable under measurement noise unlike the SVM based method. Figure 2 shows false alarm rates for the two types of noise when we use different noise rates in train and test data. As can be seen from the figure, our method is very robust to noise in measurements.

4.4 Robustness to Topology Changes

The OMS is also tested under some changes to network topology in the form of certain transsmission lines being disconnected from base topology of IEEE-118 Bus system. For the illustration here we consider the case where transision lines interconnecting following Bus pairs being put out of service : Bus 119 to Bus 120, Bus 23 to Bus 25, Bus 30 to Bus 135, Bus

Table 3: Performance of the existing SVM classifier

Noise Level	**Case**	**FA Rate** (%)	**MD Rate** (%)
None	Exact	0.15	1.67
	+1 Delay	0.52	11.5
	±2 Error	5.80	18.83
±1 %	Exact	0.43	9.00
	+1 Delay	0.69	24.83
	±2 Error	4.31	35.83
±2 %	Exact	0.63	17.83
	+1 Delay	0.96	42.17
	±2 Error	1.72	47.83
±1 % (only Test)	Exact	0.88	7.41
	+1 Delay	2.80	18.70
	±2 Error	5.27	29.30
±2 % (only Test)	Exact	4.78	9.16
	+1 Delay	12.83	22.00
	±2 Error	12.64	25.33

Table 4: OMS performance under Type I Noise

Noise Level	**FA Rate** (%)	**MD Rate** (%)	Υ **(cycles)**
None	1.61	0.16	35.52
±1% Uniform	1.89	0.00	38.44
±2% Uniform	2.79	0.00	38.97
±3% Uniform	3.09	0.00	40.06

[2]According to *"C37.118.1-2011 IEEE Standard for Synchrophasors for Power Systems"* tolerable measurement noise is 1% of Total Vector Error and thus the noise levels considered here are relatively high. See https://standards.ieee.org/findstds/standard/C37.118.1-2011.html

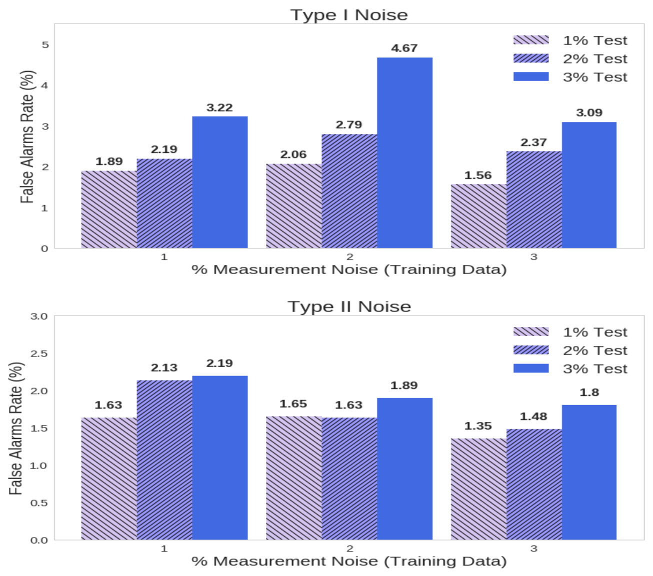

Figure 2: Robustness of OMS: False Alarm Rate under Type I and Type II noise

Table 5: OMS performance under Topology Changes

Noise Level	**FA Rate (%)**	**MD Rate (%)**	Υ **(cycles)**
None	1.94	0.00	22.50
$\pm 1\%$	2.52	0.00	23.03
$\pm 2\%$	3.00	0.00	21.36
$\pm 3\%$	3.02	0.00	23.53

35 to Bus 37, Bus 60 to Bus 61 and Bus 95 to Bus 96. Such changes in the network are common in the day-to-day operation of power systems. Lines are often put out of service to accommodate maintenance or operational requirements. These topology changes can alter the pre-fault power flow and the evolving voltage profiles of the network.

We train the RNN with data generated using the original topology and test it with data generated using the altered topology. We also add different levels of Type I noise. The results, as given in Table 5, show that the proposed RNN based OMS is quite robust to network topology changes.

4.5 Identification of Critical Generator

As described in Section 2.3, for identification of critical generator we project the data (over relevant time interval) onto $\Re^2$ using the first two principal components. These 2D-

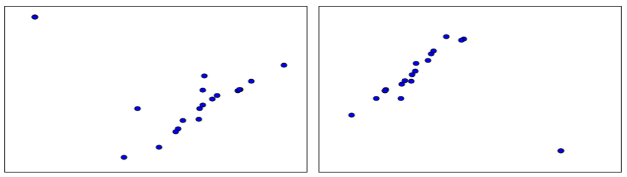

Figure 3: Projected Measurements in $\Re^2$ showing outliers

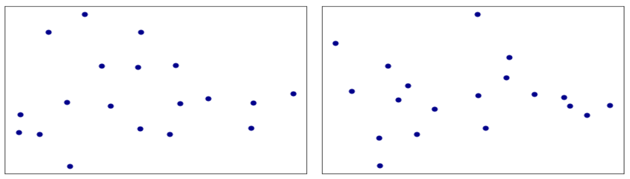

Figure 4: Projected Measurements in $\Re^2$ for False Alarms

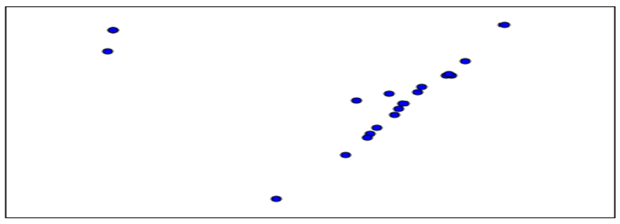

Figure 5: Projected measurements when multiple generators are critical

vectors corresponding to all generators for two unstable profiles is shown in Fig. 3. It is easily seen that the critical generator is an outlier. Our technique of robustly fitting a Gaussian density and looking for a point with least likelihood, is able to identify a critical generator in about 85% of unstable cases even under measurement noise. While our algebraic technique can identify only one critical generator, the visualization in $\Re^2$ is a good aid for the operator to quickly see all the critical generators as illustrated in Fig. 5.

We also found that this visualization can help reduce the effect of our false alarms. When OMS predicts instability wrongly, there would not actually be a critical generator and hence in the projected data there may be no outliers. We show the visualization in a false alarm case in Fig. 4. Thus, this PCA-based visualization can help the operator to properly respond to the alarms generated by OMS.

5 Conclusions

In this paper we presented a GRU based RNN for predicting transient instability in a power system. All the existing ML approaches for this assume precise knowledge of fault occurence and clearance in the system. Our proposed RNN continuously monitors system variables in a sliding window and it does not need any fault information. Its performance is very good as seen from our experimental results. Further, the proposed RNN is very robust to measurement noise and topology changes. We also presented a PCA based visualization method for isolating the critical generator in case of instability. The OMS presented in this paper assumes that measurements from all generators are available at a central location synchronously. Extending this into a distributed system is an interesting problem for future work.

Acknowledgments

This work is supported by the FIST program grant of DST, Government of India and also by Robert Bosch Centre for Cyber-Physical Systems, IISc.

References

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. 2014.

[Dehghanian *et al.*, 2015] Payman Dehghanian, Yaping Wang, Gurunath Gurrala, Erick Moreno-Centeno, and Mladen Kezunovic. Flexible implementation of power system corrective topology control. *Electric Power Systems Research*, 128:79–89, 2015.

[Gomez *et al.*, 2011] Francisco R Gomez, Athula D Rajapakse, Udaya D Annakkage, and Ioni T Fernando. Support vector machine-based algorithm for post-fault transient stability status prediction using synchronized measurements. *IEEE Transactions on Power Systems*, 26(3):1474–1483, 2011.

[Griffin and Moorman, 2001] M Pamela Griffin and J Randall Moorman. Toward the early diagnosis of neonatal sepsis and sepsis-like illness using novel heart rate analysis. *Pediatrics*, 107(1):97–104, 2001.

[Gurrala *et al.*, 2017] Gurunath Gurrala, Disha Dinesha, Aleksandar Dimitrovski, Srdjan Simunovic, Sreekanth Pannala, and Michael Starke. Large multi-machine power system simulations using multi-stage adomian decomposition. *IEEE Transactions on Power Systems*, 2017.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Hüsken and Stagge, 2003] Michael Hüsken and Peter Stagge. Recurrent neural networks for time series classification. *Neurocomputing*, 50:223–235, 2003.

[Jensen *et al.*, 2001] Craig A Jensen, Mohamed A El-Sharkawi, and Robert J Marks. Power system security assessment using neural networks: feature selection using fisher discrimination. *IEEE Transactions on Power Systems*, 16(4):757–763, 2001.

[Ji *et al.*, 2016] Luyu Ji, Junyong Wu, Yanzhen Zhou, and Liangliang Hao. Using trajectory clusters to define the most relevant features for transient stability prediction based on machine learning method. *Energies*, 9(11):898, 2016.

[Kamwa *et al.*, 2009] I Kamwa, SR Samantaray, and Geza Joos. Development of rule-based classifiers for rapid stability assessment of wide-area post-disturbance records. *IEEE Transactions on Power Systems*, 24(1):258–270, 2009.

[Kingma and Ba, 2014] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.

[Kundur *et al.*, 1994] Prabha Kundur, Neal J Balu, and Mark G Lauby. *Power system stability and control*, volume 7. McGraw-hill New York, 1994.

[Pai, 2012] Anantha Pai. *Energy function analysis for power system stability*. Springer Science & Business Media, 2012.

[Pascanu *et al.*, 2013] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013.

[Rajapakse *et al.*, 2010] Athula D Rajapakse, Francisco Gomez, Kasun Nanayakkara, Peter A Crossley, and Vladimir V Terzija. Rotor angle instability prediction using post-disturbance voltage trajectories. *IEEE Transactions on Power Systems*, 25(2):947–956, 2010.

[Rousseeuw and Driessen, 1999] Peter J Rousseeuw and Katrien Van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, 1999.

[Xue *et al.*, 1992] Yusheng Xue, Louis Wehenkel, Regine Belhomme, Patricia Rousseaux, Mania Pavella, Edwige Euxibie, Bertrand Heilbronn, and J-F Lesigne. Extended equal area criterion revisited (EHV power systems). *IEEE Transactions on Power Systems*, 7(3):1012–1022, 1992.

[Zhou *et al.*, 2016] Yanzhen Zhou, Junyong Wu, Zhihong Yu, Luyu Ji, and Liangliang Hao. A hierarchical method for transient stability prediction of power systems using the confidence of a svm-based ensemble classifier. *Energies*, 9(10):778, 2016.