April 15, 2020
Webinar Questions and Answers

NASPI Webinar Panel Discussion
National Infrastructure for Artificial Intelligence on the Grid (NI4AI)

Sensor Data Rates and Platform Performance

50kHz data, 9 locations, 4 years is closer to a petabyte, not 4 terabytes... what am I missing?
Answer: The data set in question was sampled at 50KHz (and not 100KHz) and was capturing data at 2 bytes per sample. This yields approximately 6.3TB per year per stream per sensor and the total archive is approximately 90.1TB (compressed). While data was collected for 4 years, there are some gaps due to sensor issues.

Which is the reporting rate of PMUs?
Answer: The reporting rate varies from one report every other cycle (i.e., 25 or 30 frames per second (fps)) to two reports per cycle (100 or 120 fps). Some PMUs report at 10, 15, 30, 60, 120, and 240 fps. The platform can accommodate diverse reporting rates, including point-on-wave (POW) data at frequencies up over 1 million samples per second per stream.

Could you mention briefly the order-of-magnitude of your planned ingest rate? i.e. how many terabytes per second, or whatever? Thanks!
Answer: We have benchmarked the platform as being capable of handling a sensor data load that is 100x larger than Dominion Energy (which is in the hundreds of PMUs, nearing 1,000). This means ingesting, processing, compressing, and storing over 150 million points per second. The platform was architected to be horizontally scalable to much larger flows or loads of data.

Do I understand correctly that we're presently capable of ingesting 150E6 data points per second? For 3-phase voltage-only sensors running at 100k samples per second, that means we can only support 500 sensors? It seems to me that, even for North America alone, it would be good to ingest at a rate that is at least 3 to 5 orders of magnitude higher - is that realistic?
Answer: The system's horizontal scalability should allow us to go dramatically beyond 150M points per second (that was actually done on a pretty small cluster). The issue for us is that our

customers are not quite there yet so we have focused benchmarking on much smaller, utility scale deployments.

What are you using to store the network model?
Answer: PingThings has built a custom solution to ingest, store, and make available programmatically the Network Model.

Data Types and Sources

Do you have the Pecan street datasets in NI4AI?
Answer: No, last time we checked, Pecan Street charged money for access to datasets.

How to integrate data from different infrastructures (electricity, gas, water, communication, etc) to properly address the issue of resilience?
Answer: The PredictiveGrid platform is agnostic to the type of data, and capable of aligning any time-series stream from other infrastructures. Typically, these will have much slower data rates.

Are there any sources for real time data? or if anyone is willing to share?
Answer: A major part of the project is to make live, real-time streaming sensor data available via the platform and this effort. We have not deployed the sensors yet but are on track to start building out a very large collection of data.
There is some precedent for real-time data collected independently on customer premises that have been publicly shared. Notable resources include FNET and http://map.pqube.com/.

What is the preliminary data filtration technique used?
Data pre-processing methods largely depend on the application you are targeting. The most basic ones consist in outlier removal and detrending. An example with pre-processing techniques for a specific PMU application are detailed in this paper:

> L. Vanfretti, S. Bengtsson, and J.O. Gjerde, Preprocessing synchronized phasor measurement data for spectral analysis of electromechanical oscillations in the Nordic Grid, Int. Trans. Electr. Energ. Syst., 25, 348–358, 2015. [On-line]: http://dx.doi.org/10.1002/etep.1847
> Author's copy: here

Will a mixture of PMU and Smart meter data be helpful for distribution system?
Answer: Yes, there are a number of applications in distribution systems that can make use of smart meter or AMI data. One example is distribution state estimation algorithms that use meter data (which comes in at much slower intervals) as pseudo-measurements. The problem with AMI data is not only that it tends to reside in a separate silo, but that it is harder to scrub sensitive private information about customers because its location is so granular.

One more question I would like to ask regarding the total vector error (TVE).
Answer: TVE is defined by the C37.118 standard. Sources for more information about TVE include the NASPI Distribution Task team report titled Synchrophasor Monitoring for Distribution Systems: Technical Foundations and Applications as well as www.openpmu.org/pmu-fundamentals.

Will the model of the feeder made available with the PMU data?

Answer: NI4AI will aim to make as much information about real feeder models available as possible, but we are constrained by permission from the original owners of the data. The distribution feeder presently featured has been anonymized and we are not authorized to share model information. We hope to publish datasets along with circuit models when possible, or make these available to a restricted group of authorized researchers. We also look forward to hosting synthetic (realistic but not real) datasets with matching model information. Some of these are being produced in the ARPA-E funded GRID DATA program, and are publicly accessible at https://egriddata.org/. Also, for a repository of publicly accessible circuit models, see BetterGrids.org.

Accessing the Data

Is there any timeline on when these data sets can be accessible to Universities?
Answer: Some are accessible now! Sign up at ni4ai.org.

Can you give us information on how we can have access to the datasets that are already available?
Answer: Create an account at ni4ai.org and log in directly. The WARP tool allows users to visualize and download the data. Users also receive an API key (listed under “User Profile”). Python code on how to access the API are published on the blog.

Application and Use Case Questions

How can one distinguish between voltage sags and bad data outliers with similar voltage dip signatures? How could this voltage sag function work with "bad" PMU data? Is it possible that bad data could be falsely identified as "sag"?
Answer: There are several ways proposed to detect bad data. One simple way is to look at correlations across nearby measurement points. If we see correlated dips across points, this is likely a physical voltage sag, not bad data.

How are you placing the microPMU in distribution network? Are you using deterministic or stochastic method to deploy the microPMU or any other method?
How are decisions for siting microPMUs in the distribution network made?
Answer: Optimal PMU placement is an open research topic. Generally speaking placement depends on control application, observability, protection and monitoring applications that process the synchrophasor data. There is also research happening to examine applications for sparse sensor networks, which are more common among existing sensor deployments.

Luigi & Tetiana’s Presentation

When you calculated accuracy, how much data had forced oscillation out of how much data? If the model was not detecting any forced oscillation, say it just says all data are normal, what will be the accuracy?

With regards to the first question, it is assumed it refers to the training using synthetic data. For details please refer to Section III.A of the paper, here. In the case of synthetic data, the sinusoidal frequency is changed from 1Hz to 15Hz for each export, and the output is plotted on a time delta of 1 second. Exports of the events during oscillation are merged in the same folder so that ML training is done with only two folders of pictures (inside and outside event), as shown in Fig. 6 of the paper. Observe that because we are using simulation, we can potentially generate as much data as we want. For a single simulation, normal conditions are applied from t = [0 120) sec., the forced oscillation is injected from t = [120 200) sec., and normal conditions again from t = [200 300]. Roughly speaking, for a single simulation, this corresponds to ~27% of the data containing oscillations, and 73% containing normal conditions.

With respect to the 2nd question, please see Table 3 of the paper. Normal data being miss-classified is regarded as a false-positive. In our work we tested a number of different neural networks, and thus, different ML models will yield different results. The number of false positives are shown in the table for each of them. The accuracy computed includes both false-positives and missed events, because we are interested in the ability to classify both types of conditions, so I cannot specify the accuracy that you are looking for. What Table 3 shows clearly is that the number of false-positives is lower than the number of missed events, so this implies that the models are more accurate in detecting normal conditions, which is somehow expected as per the amount of training data available to train for those.

How do you detect forced oscillations with frequencies in range of inter-area modes?
The goal of this work was to identify oscillations resulting from sub-synchronous control interactions (SSCI) in wind farms, which are not within the inter-area mode range. Hence, the proposed methodology was designed to identify forced oscillations which have a unique signature, i.e. well defined frequency with little to no variance, and zero damping).

When forced oscillations appear close by or on top of well damped modes their time-and-frequency features are different, and therefore, they will require a different approach to training and verification. One could exploit the knowledge of the features of the signals in both the time-and-frequency domain to train the algorithms, for more details on the characteristics of forced oscillations within the range of inter-area modes see the following publication:

> L. Vanfretti, S. Bengtsson, Vedran Perić, and Jan O. Gjerde, "Effects of Forced Oscillations on Power System Damping Estimation," International Workshop on Applied Measurements for Power Systems (AMPS), September 22-24, 2012, Aachen, Germany. Online: https://ieeexplore.ieee.org/abstract/document/6344015
> Author's copy: here

Have you tried to generate the extreme contingencies n-k?
Yes, it is possible to generate and stimulate n-k contingencies. However, after a certain number of contingencies you would have separate islands that need to be considered as individual networks, and deal with concurrently.

How does quantify if the ML model is losing accuracy due to new events or anomalies?
We did not carry out this analysis. There are different methods to measure and improve accuracy on ML applications, a laymans' explanation on the need for continuous learning is given in this Medium article here. In the case of the target application in our work, i.e. a ML application at the edge, there are approaches where the edge device will send relevant new data to the training server in order to improve the model itself. This is known as Distributed or Federated Learning.

This has not been explored in our work so far, but there are plenty of results and problems in this area, as shown in the following papers: here and here.

Luigi, good work! As you said, correct labeling is important in this forced oscillation detection approach. Sometimes it is not easy to differentiate between forced oscillations and natural oscillations only by looking at the curves. How was this handled? Any thoughts?
Please see the response to questions QL1-QL2 above which provide insight into this question.

Luigi, have you any ideas how to calculate parameters of oscillation such as frequency, amplitude and damping?
Yes, I have several ideas, but I'll save those for a grant application, 😂
Joking aside, there are a great number of methods already in production that deal with this issue quite well. I am not sure if throwing machine learning at the mode estimation problem is the best approach.

In our work we are targeting to detect/classify this behavior so that we can use it for control purposes, for example, we can use the output of the classification and pass it through an inverse-time logic in order to decide to trip the wind farm/turbine, to automatically ramp down its dispatch (known solution applied in the field) or to arm a control scheme to counter measure it (e.g. at a STATCOM or SVC).

So, from an academic research perspective, of course there are many ways to develop an ML-based mode meter. However, from the practical point of view, we already have mode meter methods based on signal processing deployed in the field and it would be a gigantic task to try to arrive at the level of maturity of those methods and have them deployed. I think the role of AI/ML in this specific topic would be more on how to compliment the existing mode estimators with auxiliary tasks, such as the application we presented.

I think sharing labeled data (training dataset) with the community is crucial to develop new ML algorithms.
Yes, I agree. However, based on my experience, preparing, documenting and publishing data is even more challenging to do than a regular article and the cost/benefit is usually very low due to the poor culture of the power engineering community on valuing and giving proper attribution to the originators.
Luckly, there are now some incentives to do so outside of the traditional venues and circles. For example, if time permits, we aim to release the data that we are generating in the project funded by NYSERDA when the project is completed in a "Data in Brief" publication. Nevertheless, even when a citable paper is available, people using the resource do not necessarily cite it. This human factor is one of the challenges that perhaps the NI4AI can help to address.

What about the speed of Python?
We have not carried out any performance analysis. In all honesty, this is a question that a computer scientist could answer better than me. Anecdotally, what I have seen on my own and

with students is that the speed is more than acceptable and even more competitive than the usual Matlab. Here are my two cents.
Scripting languages (e.g. Matlab, Python) tend to be slower than compiled languages (C, Fortran). However, keep in mind that Python is a “glue-language” and a lot of the code is actually just being used as an interface to call compiled binaries of routines originally developed in C or C++ in the case of numerical computations. For example, the widely used numpy library integrates C/C++ and Fortran code: https://numpy.org/. So, ultimately, the speed of a particular software pipeline will depend on the mix of code and most importantly on the hardware architecture and how it is exploited for performance gains. The following website discusses some of these aspects: https://wiki.python.org/moin/PythonSpeed

In the case of GPU speed performance for computations, there are an array of interfaces to give the ability to Python programmers to exploit the GPUs, and their performance differs. For example, NVIDIA provides a number of Python API’s such as PyCUDA (https://developer.nvidia.com/pycuda), CUDA Python and others: https://developer.nvidia.com/language-solutions
A number of approaches to use GPU accelerators from Python are described in this recent article: https://towardsdatascience.com/python-performance-and-gpus-1be860ffd58d

In the case of embedded systems, such as the NVIDIA Jetson, the manufacturer of the embedded system provides some tools to facilitate inference computations. See the following link for the case of the Jetson, of course, other platforms may have similar support. Speed in this case is more a factor of the particular embedded system’s hardware resources.

How do you compare regular neural networks vs deep neural networks for network load, PV and Wind generation forecasts in short-term?
This is an interesting question. With respect to the forecasting problem, my first question would be, do I need AI/ML? One would need to first have a baseline on the performance of existing methods and see what are the weaknesses that would merit the use of AI/ML. This follows the same rationale as in [QL6] above.

Do you also use a real time simulator?
Yes, we are using different types of tools for different types of purposes.
For the research presented, we plan to use our real-time simulator to test the performance of the embedded system to do inference through Hardware-in-the-Loop testing where the simulator will have a detailed model of a wind farm and we will simulate the scenario to generate an oscillation event. You can use any real-time simulator for this purpose, for example, we plan to use both Opal-RT and Typhoon-HIL targets.

Moreover, real-time simulators can be used for more than “real-time” simulation and for more than HIL. For example, we are using the Opal-RT system at RPI to do simulations of very complex models of transmission + distribution that would take gigantic amounts of time to run otherwise. Opal-RT’s software RT-Lab has a Python interface that allows it to submit a model for simulation and to return the results to the main software routine. Similarly, in the case of micro-grids, we are using a Typhoon-HIL simulator, that has a similar Python API. In these cases we are not doing hardware-in-the-loop simulation, but using the hardware to run models as fast as possible.

How does accuracy change for oscillation detection, when PMU Data has all other kind of events and anomalies. Training might be difficult to differentiate between Force oscillation and other events.

The data we used for training contained all other kinds of events and anomalies and as it can be seen from our results, the accuracy is surprisingly good even for simple ML models. The key for training is to have very well labeled data of what you want to detect/classify. Labeling data is the most time consuming and important step. Otherwise the saying “garbage in, garbage out” fully applies.