GridDS: An Open-Source Toolkit for Energy Forecasting

Indrasis Chakraborty
Senior Data Scientist, LLNL

Workshop on Optic Sensors For Energy Application, March 2-3 2023

Outline

- Context
  - Why we need a ML end-to-end toolkit
  - Why gridds is a generalized framework
- Overview of gridds
  - Data cleaning, Data structuring, ML experiments, Visualization
  - ML models
  - Results
- Conclusion and Ongoing effort

Why we need griddS

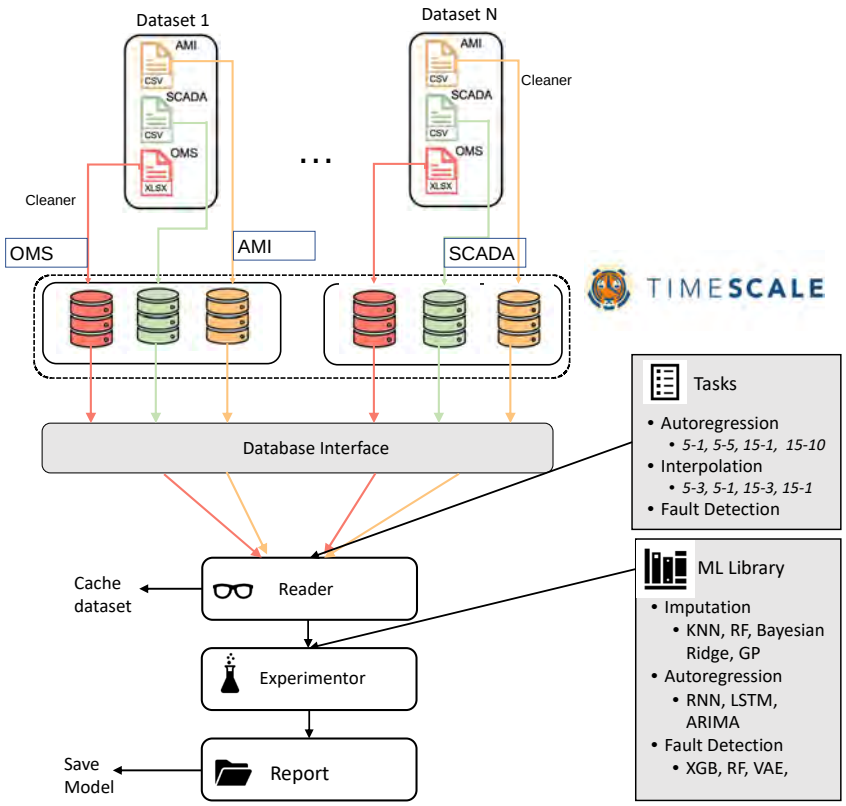
- DOE smart grid initiative has committed to advanced smart meter infrastructure which has installed approximately 10 million meters a year, totaling 100 million meters in 2020
- Wide Area Management Systems (WAMS) have enabled large scale data collection and storage, including AMI, OMS, SCADA, and GIS
- We leverage contributions from previous research on taxonomies for this data and surveys of state-of-the-art analysis performed on each data modality
- While there has been notable progress in WAMS infrastructure and energy data analytics systems, there are no existing open-source platforms that integrate the data from WAMS into a comprehensive set of ML models

Why we need griddS

- The few existing models are also strictly proprietary and are not available for broader research community
- gridds fills the gap by providing an integrative software platform to train and validate ML for central energy challenges must achieve modularity across datasets, models, and tasks
- gridds aims to provide open-source software tools for machine learning that can enable a community of researchers to build and share ML models for various challenges in the energy grid
- To accomplish this goal, we focus on integration and interoperability of four key components: (1) Data sources, (2) Task design of key energy grid challenges and (3) ML models, and (4) Visualization of findings

Generalizability of gridds

- We wanted gridds to be sharable across various energy infrastructure related research projects
- We proposed under gridds a workflow from raw data to a common representation (as shown in the schematic)



Generalizability of gridds

- In gridds we leverage TimeScaledDB for data storage and data augmentation
- Built on PostgreSQL, TimeScaleDB is an efficient means for storing finely sampled timeseries data. The database interface is a python class capable of flexibly executing modular queries to TimeScaleDB
- The “reader” class can compile a range of datasets using the database interface. Using a minimal database specification (we will talk more about this in the demonstration), the “reader” class can compile structural relationships between devices and their upstream and downstream components

Generalizability of gridds

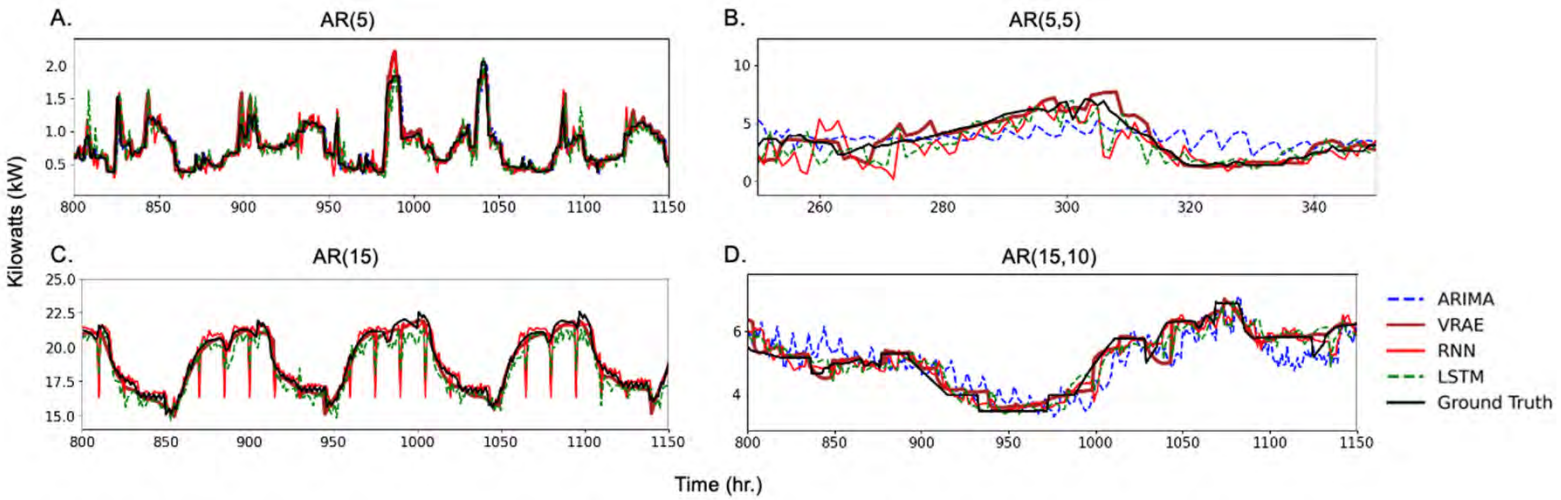
- After data has been aggregated and a subset of data has been selected using the “reader” and “database interface”, it is used to instantiate the “dataset” class
- The “dataset” class, supports common transforms to data such as normalization, cross-validation, and signal preprocessing
- Our schematic shows that after the “reader” provides the “dataset” class, the “experimenter” class requires an input of a set of models and a task and runs experiments on several ML objectives

Model Applications

- Autoregression
  - VRAE: encoder/decoder learns latent representation for hidden state trajectory.
  - RNN: standard vanilla neural networks for time series modeling.
  - LSTM: improves upon RNN for modeling longer term temporal dynamics.
  - ARIMA: commonly used baseline for forecasting timeseries data.
- Interpolation
  - KNN: nearest neighbors has a very straightforward application to interpolation.
  - Bayesian Ridge: introduces simple probabilistic model baseline for a diverse approach that performs well.
  - Random Forest: simple SkLearn baseline.
  - Gaussian Process: GP kriging is a well known method for interpolation (Kleijnen 2017).
- Fault Detection
  - LSTM: models longer term temporal dynamics giving good model capacity to predict faults.
  - VRAE: commonly used in anomaly detection (Wang 2018, Chen 2019).
  - XGBoost: commonly used baseline on all problems involving tabular data. Works well on time series.
  - Random Forest: simple baseline classification system.

Autoregression

- Autoregression is defined as the capacity to forecast future values given a history of previous observations.
- Understanding what conditions and factors are correlated with disparities in performance is important for future research and task design for load consumption forecasting



ARIMA: Autoregressive integrated moving average, VRAE: Variational recurrent autoencoder, RNN: Recurrent neural network, LSTM: Long short-term memory

Autoregression

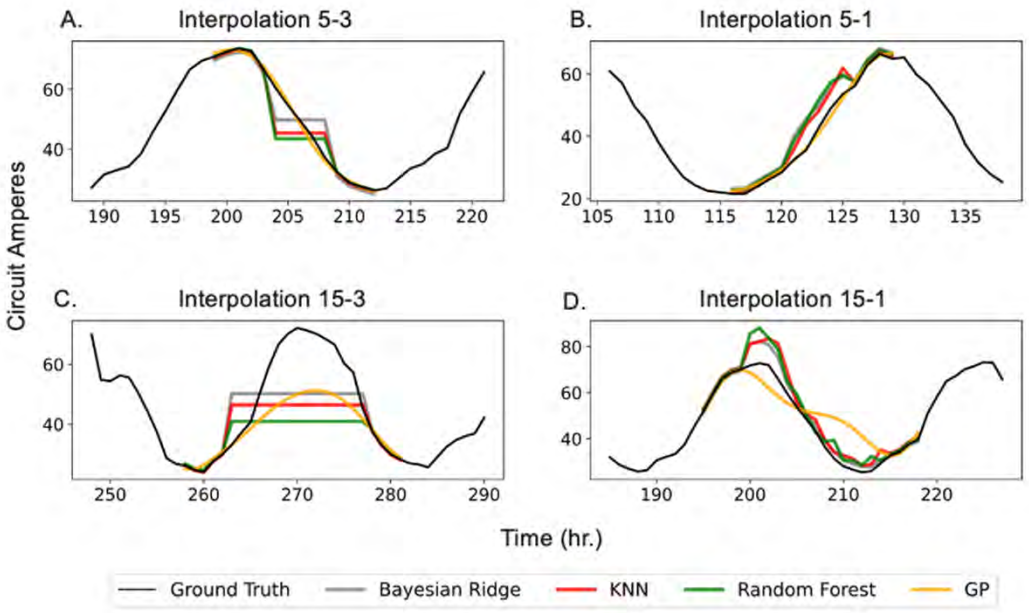
Method	AR Task Metric	*AR(5)*	*AR(5,5)*	*AR(15)*	*AR(15,10)*
ARIMA	MAE	**0.08 ± 0.04**	0.97 ± 0.28	**0.14 ± 0.13**	1.09 ± 0.52
	MBE	**-0.00 ± 0.00**	**0.07 ± 0.16**	**-0.00 ± 0.01**	-0.49 ± 0.41
	RSE	**0.04 ± 0.04**	0.91 ± 0.24	**0.08 ± 0.06**	1.60 ± 0.14
LSTM	MAE	0.38 ± 0.35	0.96 ± 0.20	0.34 ± 0.30	0.84 ± 0.32
	MBE	-0.02 ± 0.24	0.16 ± 0.29	0.01 ± 0.15	0.06 ± 0.27
	RSE	0.23 ± 0.22	1.17 ± 0.28	0.38 ± 0.23	1.12 ± 0.02
RNN	MAE	0.41 ± 0.39	0.96 ± 0.20	0.40 ± 0.38	0.82 ± 0.27
	MBE	0.02 ± 0.24	0.18 ± 0.27	-0.05 ± 0.10	-0.04 ± 0.29
	RSE	0.27 ± 0.26	1.15 ± 0.37	0.45 ± 0.36	1.06 ± 0.17
VRAE	MAE	0.25 ± 0.19	**0.74 ± 0.20**	0.24 ± 0.23	**0.58 ± 0.27**
	MBE	0.07 ± 0.09	0.12 ± 0.07	0.04 ± 0.01	**0.04 ± 0.16**
	RSE	0.13 ± 0.10	**0.89 ± 0.32**	0.23 ± 0.16	**0.69 ± 0.06**

- We analyzed four different autoregression tasks:
- AR(5): autoregression with a history of 5.
- AR(5,5): autoregression with a history of 5 and a horizon of 5.
- AR(15): autoregression with a history of 15.
- AR(15,10): autoregression with a history of 15 and a window of 5.

Interpolation

- Interpolation is defined as the capacity to construct new data points based on the range of a discrete set of known data points.

Interpolation 5-3: Gap size of 5, 3 channels are missing data.
Interpolation 5-1: Gap size of 5 and 1 channel is missing data.
Interpolation 15-3: Gap size of 15 , 3 channels are missing data.
Interpolation 15-1: gap size of 15, 1 channel is missing data.



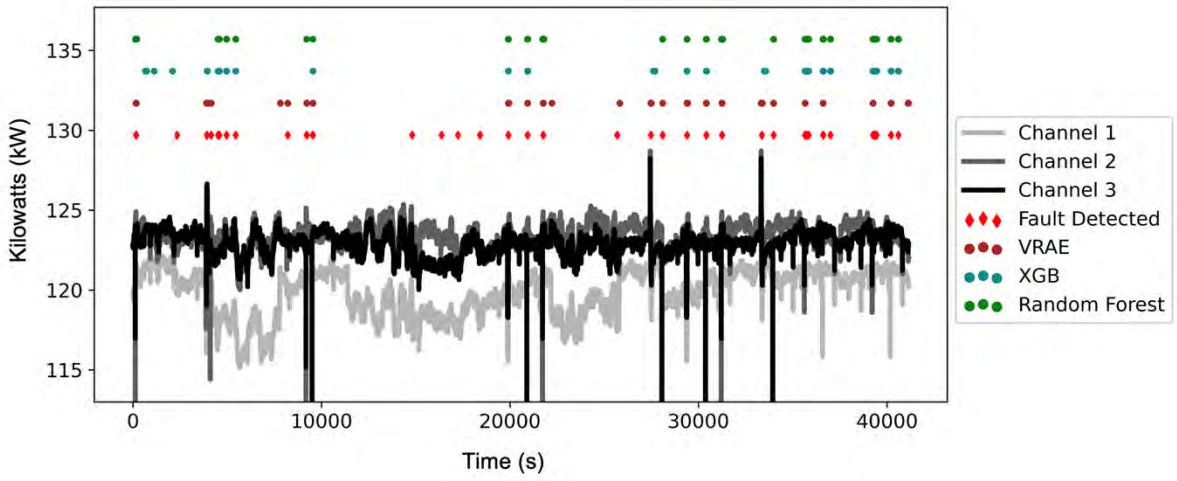
Interpolation

Method	Interpolation Task / Metric	*5-3*	*5-1*	*15-3*	*15-1*
BRR	MAE	15.70 ± 0.80	5.20 ± 0.56	14.98 ± 2.00	**4.28 ± 0.43**
	RMSE	18.26 ± 0.69	6.01 ± 0.55	17.82 ± 2.39	**5.24 ± 0.25**
GP	MAE	**2.05 ± 0.43**	**2.42 ± 0.21**	**9.19 ± 2.67**	8.63 ± 0.32
	RMSE	**2.84 ± 0.82**	**3.15 ± 0.13**	**12.62 ± 2.83**	11.07 ± 0.35
KNN	MAE	16.31 ± 1.02	5.54 ± 0.05	14.63 ± 2.25	4.56 ± 0.30
	RMSE	19.39 ± 0.49	6.41 ± 0.05	18.16 ± 3.03	5.59 ± 0.23
RF	MAE	14.44 ± 1.27	5.62 ± 0.35	14.90 ± 2.83	4.74 ± 0.36
	RMSE	17.98 ± 2.06	6.68 ± 0.17	19.20 ± 3.76	5.87 ± 0.21

Interpolation 5-3: Gap size of 5, 3 channels are missing data.
Interpolation 5-1: Gap size of 5 and 1 channel is missing data.
Interpolation 15-3: Gap size of 15 , 3 channels are missing data.
Interpolation 15-1: gap size of 15, 1 channel is missing data.

Fault Prediction

- Fault detection models predict whether a device has gone into a faulty state using anomaly detection.
- Models for this classification task are evaluated using precision, recall, accuracy, and F1.
- Performance disparity with supervised and unsupervised.



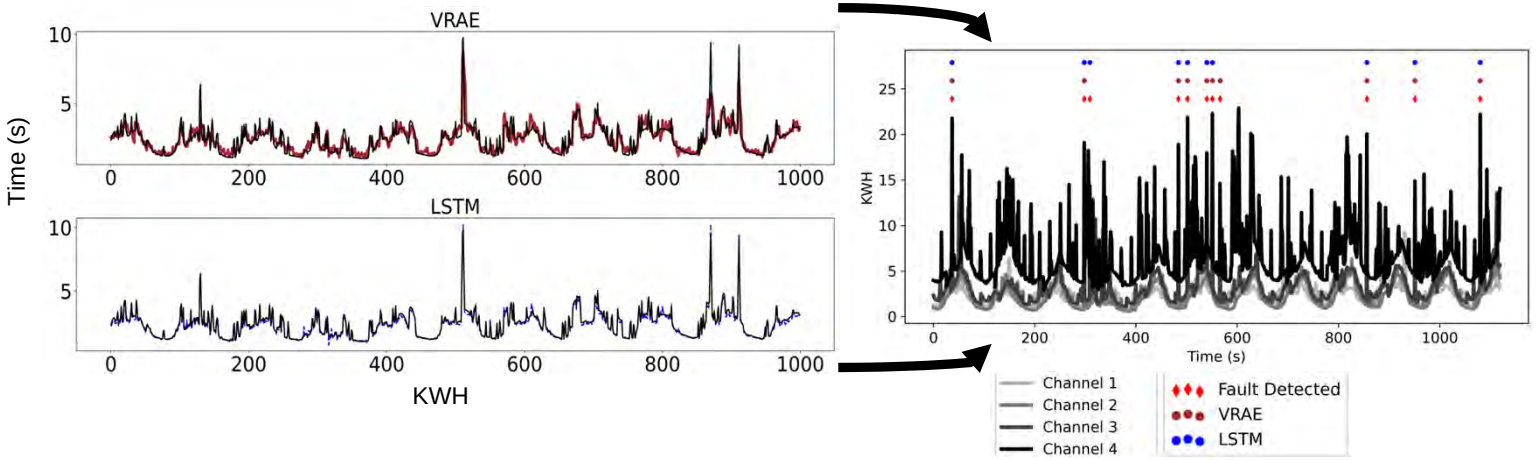
Fault Prediction

Method	RECALL	ACCURACY	PRECISION	F1
LSTM	**0.43 ± 0.22**	0.87 ± 0.04	0.13 ± 0.07	0.20 ± 0.09
Random Forest	0.31 ± 0.30	**0.97 ± 0.01**	**0.66 ± 0.29**	**0.34 ± 0.30**
VRAE	0.29 ± 0.17	0.85 ± 0.06	0.09 ± 0.07	0.14 ± 0.09
XGBoost	0.30 ± 0.32	0.96 ± 0.02	0.42 ± 0.36	0.28 ± 0.28

- RF and XGB demonstrated similar capacity for classifying faulty states but RF was able to classify these faulty states with less false positives. Thus RF achieved a higher F1 score.
  - F1 is a compound metric that represents a trade-off between precision and recall.
- While VRAE and LSTM have similar recall in comparison to other methods, they have worse precision.
  - The reason for this is, VRAE LSTM are not trained using fault labels; instead, these methods detect faults when streaming data diverges from the reconstructed time-series by a fixed threshold.

Composite Experimentation

- Fault prediction for VRAE and LSTM can be broken into two steps to show that our experiments can build upon each other.
- First, we calibrate VRAE and LSTM using autoregression.
- Then we use the reconstructed time series and compare it against the incoming measurements to detect faulty states.
- Thus, we stack autoregression and fault detection experiments. Other combinations are possible (e.g., interpolate then perform autoregression).



Conclusion

- The previous experiments show that the best ML model for a given energy problem is highly context dependent.
- This means researchers are better off trying many approaches and seeing what is best for their specific context.
- While off-the-shelf libraries like SkLearn are helpful (and even built into gridds), they don't allow researchers to have flexibility with task design and datasets. They also don't offer models as complex as our pytorch models.
- Our platform allows researchers to:
  - make small tweaks to task design such as horizon/history in autoregression.
  - Carry over ML models between datasets, enabling transfer learning and broader model validation.
  - Rapidly and efficiently test many approaches to many energy / sensor time series problems.
  - Train model hyperparameters.
- Ultimately, we show that gridds has the capacity to take general approaches to time series to machine learning, apply them to highly specific energy tasks, and evaluate / validate their performance.

Ongoing Efforts

- Switching from fault detection to fault prediction.
- Visualization.
- Task compositionality.
- Incorporating more datasets.

Acknowledgment

- LLNL team
  - **Indrasis Chakraborty** (PI), Alexander Ladd, Pedro Sotorrio, Hannah Burroughs
- ORNL, SNL, NETL, UPitt, NRECA, Corning
- DOE Office of Electricity, GMLC

Thank you for attending!