Glassoformer: a query-sparse transformer for post-fault power grid voltage prediction

Yunling Zheng[1] Carson Hu[1] Guang Lin[2] Meng Yue[3]
Bao Wang[4] Jack Xin[1]

[1]Department of Mathematics, University of California, Irvine
[2]Department of Mathematics and School of Mechanical Engineering, Purdue University
[3]Interdisciplinary Science Department, Brookhaven National Laboratory
[4]Department of Mathematics and Scientific Computing and Imaging Institute, The University of Utah

April 17, 2022

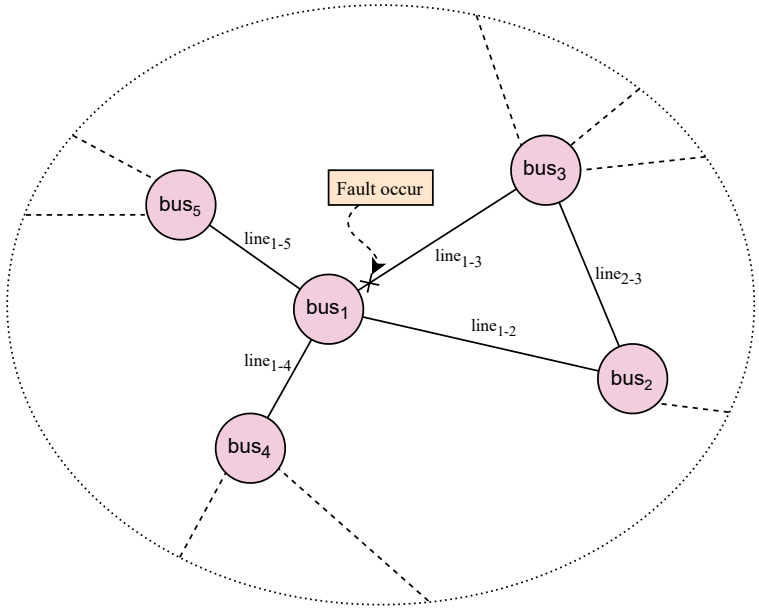
Table of Contents

Section 1

Introduction

Background

- Power grid system is a lifeline networks but fragile when interruptions occur.
- Have an online prediction for power grid stability after fault happens.
- Traditional physical simulation approach:
  - time-consuming
  - details of fault information
  - not suitable for online assessment



Power grid system with a fault on line_{1-3}.

Related work

- Data driven solutions
  - predict stability by binary indicator
  - estimate stability margin
  - not enough
- Classical signal processing: Prony's Method
  - linear time-invariant filters
  - limited to response with a rational function
  - require manual observation and parameter choice
  - not robust
- Improved with spatial coupling: 1D-CNN
  - deep feature extraction
- Our approach: Glassoformer
  - equip encoder-decoder with attentions
  - propose efficient sparse self-attention to reduce the cost
  - outperforms existing methods

UCI Department of Mathematics

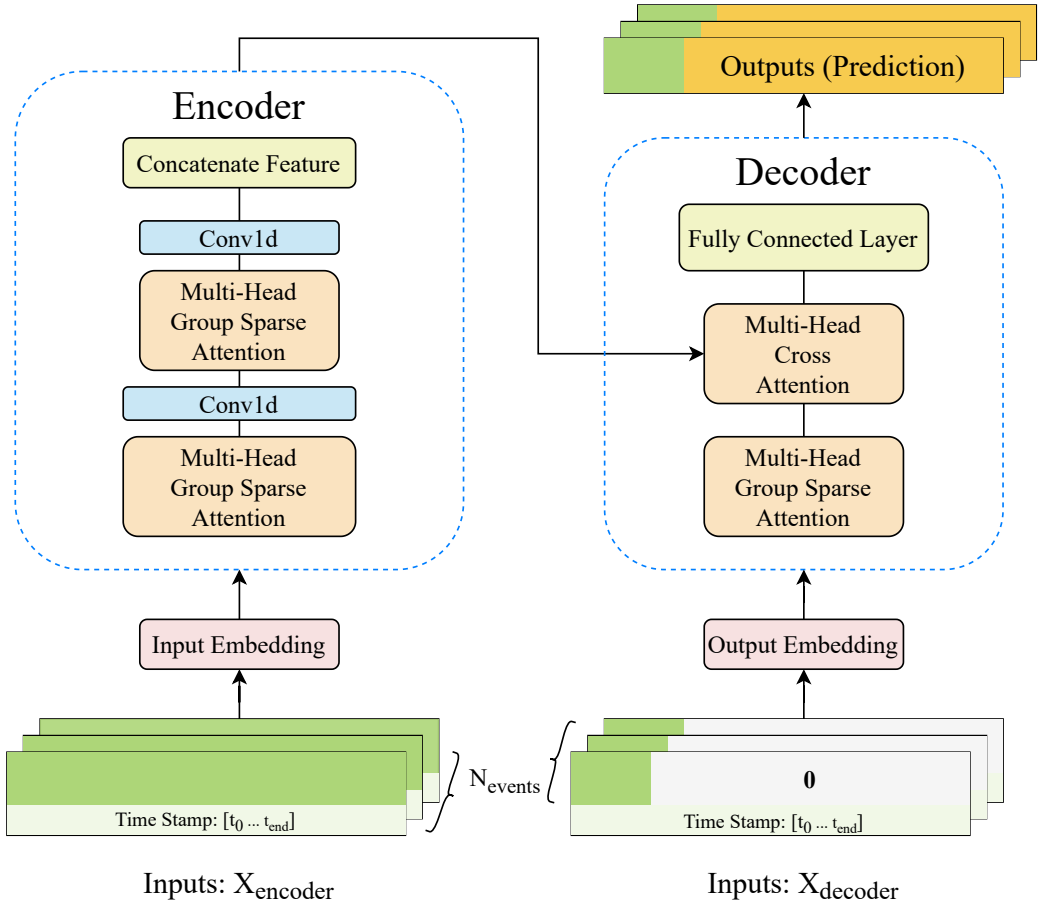
Table of Contents

Section 2

Our approach

Overview of the model

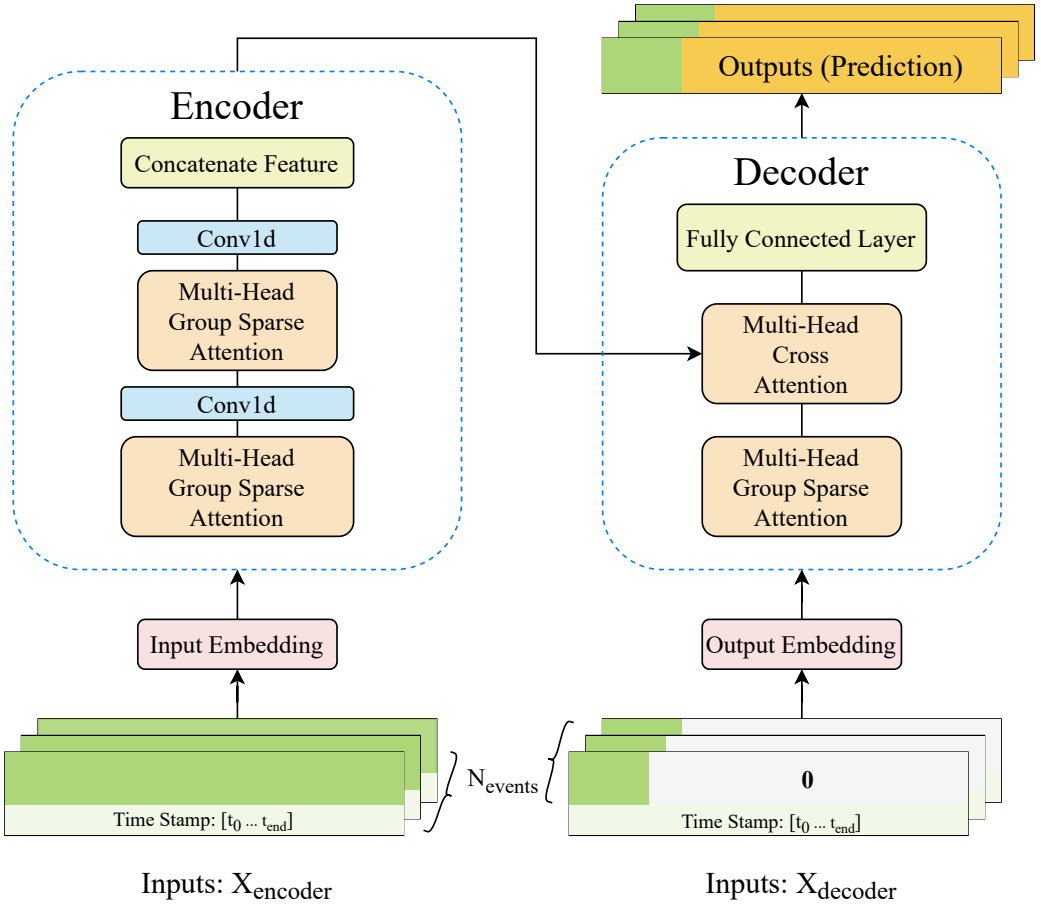
- Based on encoder-decoder structure.
- Adjusted embedding layers for post-fault prediction problems.
- Different inputs for encoder and decoder.



Auto-encoder architecture

Embedding

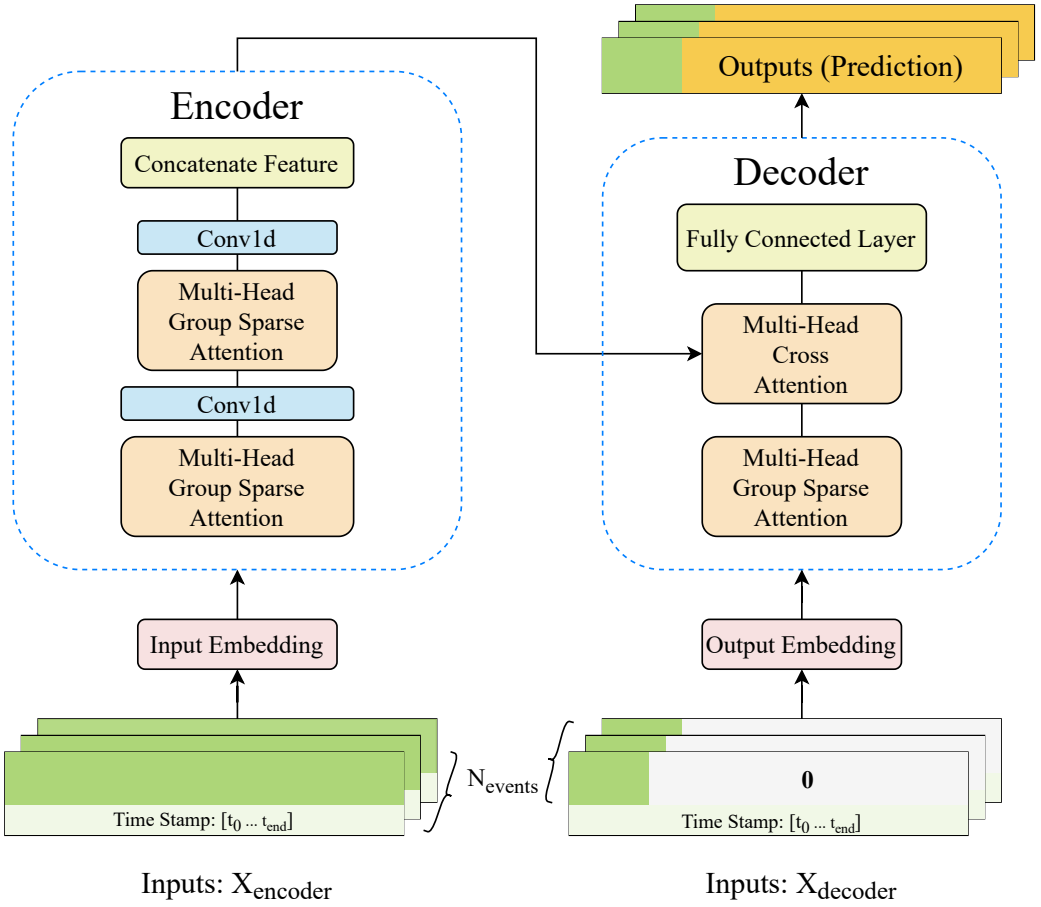
- In transient prediction, full period time stamps ($\mathbf{x}^s$) and temporal time-series features $\mathbf{x}$ are required.
- Input embedding:
  - positional embedding
  - 1D convolution
- Output embedding:
  - zero padding
  - 1D convolution



Auto-encoder architecture

Encoder and Decoder

- Encoder consists of two identical layers:
  - multi-head group sparse attention sub-layer
  - 1D-convolution sub-layer
- Decoder:
  - multi-head group sparse self attention
  - multi-head cross attention



Auto-encoder architecture

Self-attention

- long-range dependency and enable parallel processing
- Self-attention transforms input sequence $\tilde{\boldsymbol{X}} \in \mathcal{R}^{N,D_x}$ into an output sequence $\hat{\boldsymbol{V}}$ by two steps:
  - Step 1: Project input into 3 matrices:

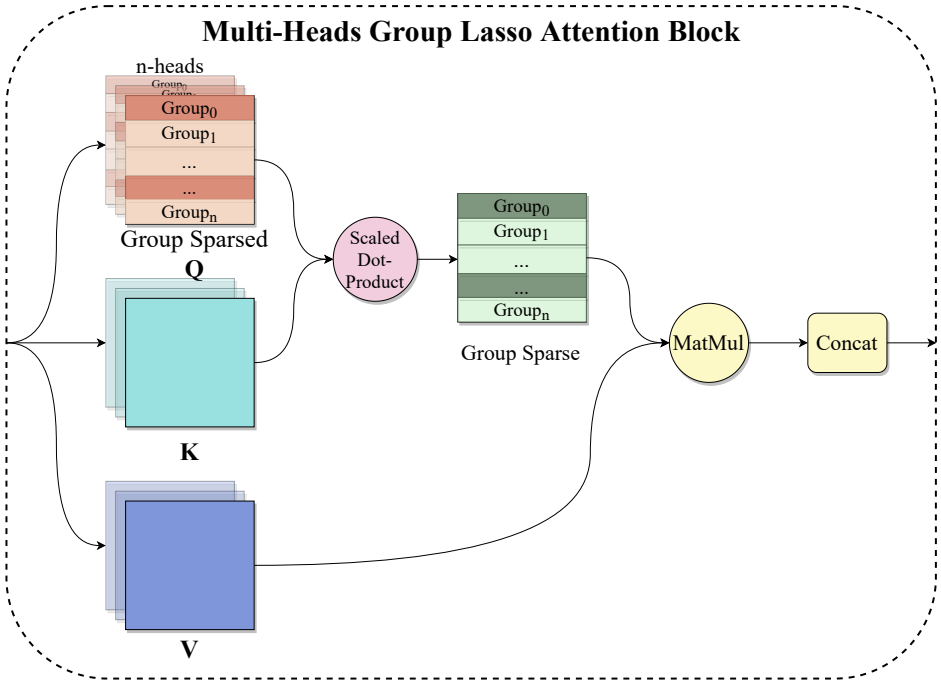
$$\boldsymbol{Q} = \boldsymbol{W}_Q^\top \tilde{\boldsymbol{X}}^\top; \boldsymbol{K} = \boldsymbol{W}_K^\top \tilde{\boldsymbol{X}}^\top; \boldsymbol{V} = \boldsymbol{W}_V^\top \tilde{\boldsymbol{X}}^\top,$$

  - Step 2: Compute output with softmax:

$$\hat{\boldsymbol{V}} = \mathrm{softmax}\Big(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D}}\Big)\mathbf{V} := \boldsymbol{A}\boldsymbol{V}$$

Query sparsification

- For long sequence, the computational and memory cost are both $\mathcal{O}(N^2)$.
- Efficient transformers have been proposed: Informer, cluster transformer, etc.
- Our approach: Query sparsification
  - apply group Lasso (GLasso) penalty to sparsify $\boldsymbol{W}_Q$
  - reduce number of nonzero rows of $\boldsymbol{Q} = \boldsymbol{W}_Q^\top \tilde{\boldsymbol{X}}^\top$ to $\mathcal{O}(1)$ (row-wise group sparsity)
  - complexity of $\boldsymbol{AV}$ goes down to $\mathcal{O}(N)$



Glassoformer attention block

Algorithm

- Propose Relaxed Group-wise Splitting Method (RGSM) to achieve row-wise sparsity in query matrix $\boldsymbol{Q}$:
- Consider group lasso penalty $\|\cdot\|_{GL}$, the loss is:

$$L(\theta, \boldsymbol{W}_Q) = f(\theta, \boldsymbol{W}_Q) + \lambda \|\boldsymbol{W}_Q\|_{GL}$$

- Solve proximal problem in closed-form:

$$\boldsymbol{y}_g^* = \arg\min_{y_g} \lambda \|\boldsymbol{y}_g\|_2 + \sum_{i \in I_g} \frac{1}{2} \|\boldsymbol{y}_{g,i} - \boldsymbol{w}_{g,i}\|_2^2$$

  here $\boldsymbol{W}_Q = \{\cdots, \boldsymbol{w}_g, \cdots\}$
- Obtain GLasso proximal operator:

$$y_g^* = \mathbf{Prox}_{GL,\lambda}(\boldsymbol{w}_g) \triangleq \boldsymbol{w}_g \max(\|\boldsymbol{w}_g\|_2 - \lambda, \mathbf{0}) / \|\boldsymbol{w}_g\|_2 .$$

Algorithm

RGSM algorithm

Update for each iteration:

$$\boldsymbol{u}_g^t = \mathbf{Prox}_\lambda(\boldsymbol{w}_g^t), \qquad \text{for } g = 1, \ldots, N$$

$$(\theta, \boldsymbol{w})^{t+1} = (\theta, \boldsymbol{w})^t - \eta \nabla f(\theta^t, \boldsymbol{w}^t) - (0, \eta \beta(\boldsymbol{w}^t - \boldsymbol{u}^t))$$

Convergence

Lemma

$$L_\beta(\boldsymbol{v}^{t+1}, \boldsymbol{u}^t) \leq L_\beta(\boldsymbol{v}^t, \boldsymbol{u}^t) + \left(\frac{L_{ip}}{2} + \frac{\beta}{2} - \frac{1}{\eta}\right) \|\boldsymbol{v}^{t+1} - \boldsymbol{v}^t\|^2$$

Theorem

If f is coercive (f bounded implies that of its independent variables, true when standard weight decay is present in network training) and the learning rate $\eta < 2/\beta + L_{ip}$, then $L_\beta(\boldsymbol{v}^t, \boldsymbol{u}^t)$ decreases monotonically in t, and $(\boldsymbol{v}^t, \boldsymbol{u}^t)$ converges sub-sequentially to a limit point $(\bar{\boldsymbol{v}}, \bar{\boldsymbol{u}})$, from which $\bar{\boldsymbol{w}}$ is extracted to speed up inference.

Table of Contents

Section 3

Experiment and result

Experimental setting

- **Dataset**: Simulation of power system in New York/New England
  - 16 generators 68 buses
  - 2248 fault events, and each 10 seconds signal
  - voltage from bus and current from line
  - graph structure of buses and lines
- Experiment details:
  - baseline: 1D-CNN, Informer, Prony's method, Transformer with lasso
  - platform: GTX-1080Ti

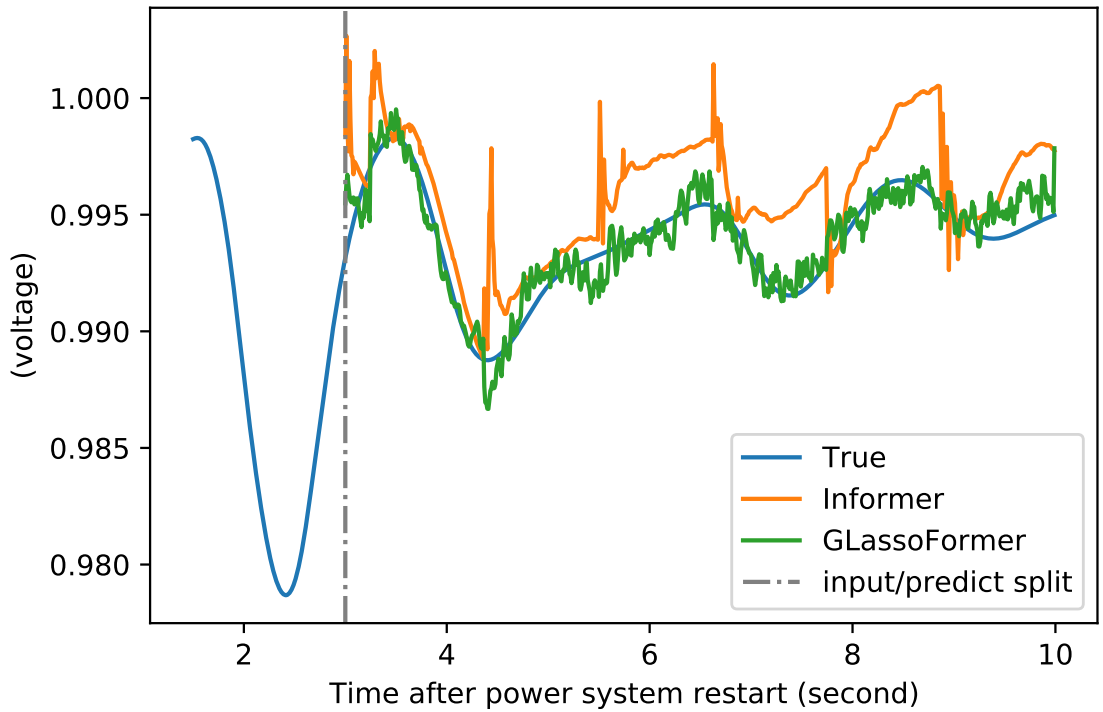
Result

	Models	GLasso	Informer	Lasso	1DCNN	Prony
I	MSE($\times 10^{-5}$)	**3.189**	3.662	3.532	8.014	–
	MAE($\times 10^{-3}$)	**2.374**	2.684	2.543	6.087	–
II	MSE($\times 10^{-5}$)	**6.115**	6.599	6.501	17.21	397.6
	MAE($\times 10^{-3}$)	**3.520**	3.877	3.611	9.264	47.3

Table: Voltage prediction error comparison.

I (II): input data with (without) neighbor voltage and current features.

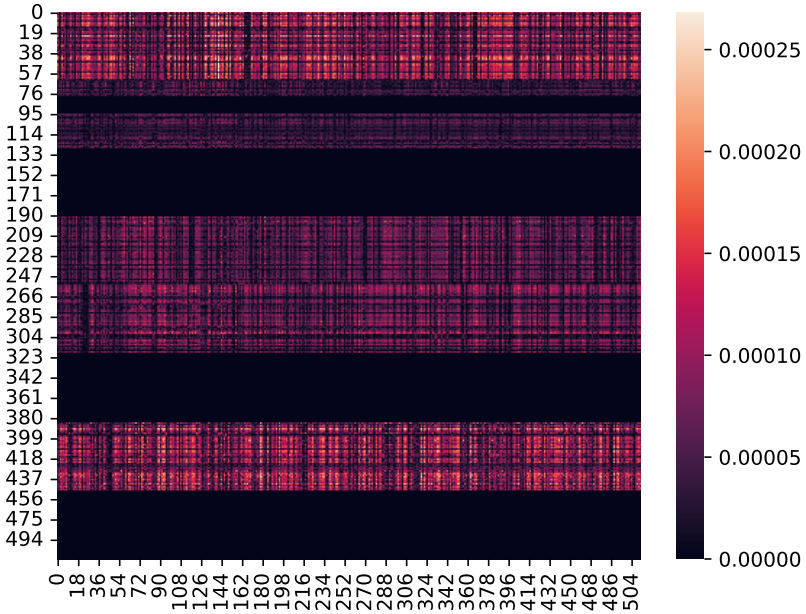
Result



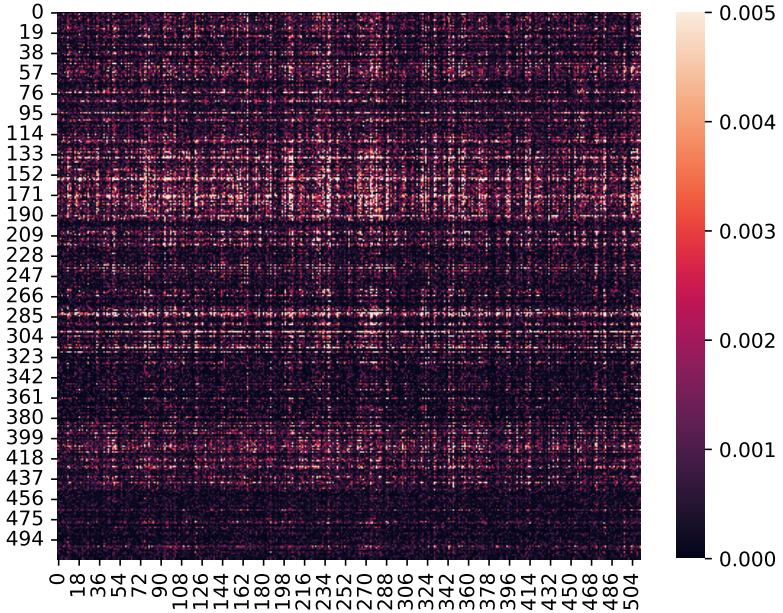
Sample post-fault voltage predictions in time (to the right of the vertical dashed line at time 3) with input from the left of the line.

Result

Visualization of sparsity patterns (black pixel) in $\boldsymbol{Q}$. GLassoformer obtains a better group (row-wise) sparsity.



Sample sparsity pattern of $\boldsymbol{Q}$ by GLassoformer



Sample sparsity pattern of $\boldsymbol{Q}$ by Lasso penalty

Result

Our network: Glassoformer is faster and sparser.

Models	GLasso	Lasso	Informer	1DCNN
Num of Params (M)	5.827	5.707	7.257	0.706
Inference Time (ms)	18.98	14.92	29.76	0.5969

Table 1: Comparison of model parameter size (M: million), and inference time (ms: millisecond) on GTX-1080Ti.

Models	GLasso	Informer	Lasso
Pruning rate [1] (%)	19.09	4.220	2.674
Training time [2] (second)	9.215	8.975	8.987

Table 2: Pruning rate and training time comparison.

[1]: fraction of zero query vectors (threshold = 1e-5); [2]: time per epoch

Table of Contents

Acknowledgments

The work was partially supported by NSF and DOE grants DMS-1924935, DMS-1952644, DMS-1924548, DMS-1952339, DE-SC0021142.

The End