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

ABSTRACT

We propose GLassoformer, a novel and efficient transformer architecture leveraging group Lasso regularization to reduce the number of queries of the standard self-attention mechanism. Due to the sparsified queries, GLassoformer is more computationally efficient than the standard transformers. On the power grid post-fault voltage prediction task, GLassoformer shows remarkably better prediction than many existing benchmark algorithms in terms of accuracy and stability.

Index Terms— efficient transformer, query sparsity, group lasso, power grid prediction

1. INTRODUCTION

Lifeline networks such as power grids, transportation, and water networks are critical to normal functions of society and economics. However, rare events and disruptions occur due to natural disasters (earthquakes and hurricanes), aging and short circuits, to name a few. Knowledge of power grid transient stability after fault occurrences is crucial for decision making; see Fig.1 for an illustration of a power grid system with a line fault. In particular, an online tool for predicting the transient dynamics, if available, will be very beneficial to the operation of the increasingly dynamic power grid due to the increasing un-dispatchable renewables. Computationally, a traditional approach for transient assessment is to simulate a large physical system of circuit equations. The simulation-based approach is very time-consuming and requires detailed fault information such as type, location, and clearing times of fault. Therefore, it is not applicable for online assessment due to the unavailability of the fault information.

With the increasing deployment of phasor measurement units (PMUs) in power grids, high-resolution measurements provide an alternative data-driven solution, e.g., learning the post-fault system responses and predicting subsequent system trajectories based on post-fault short time observations. Machine learning (ML)-based online transient assessment methods have been developed using real-time measurements of the initial time-series responses as input. These ML-based studies aim to predict system stability by deriving a binary stability indicator [1, 2, 3] or estimating the stability margin [4, 5, 6]. However, knowing the system stability or the margin only is often not enough. Instead, having the knowledge of post-fault transient trajectories is more important for the system operators [7] to take appropriate actions, e.g., a load shedding upon a frequency or voltage violation.

Classical signal processing relies on linear time-invariant filters, such as Padé and Prony's methods [8], and can be used for online prediction. However, these methods are limited to fitting responses with a rational function, requiring users to choose integer parameters (filter orders) from observed data. The prediction is not robust with respect to such choices. In [9], Prony's method is generalized to 1D convolutional neural networks (1D-CNN) with additional spatial coupling (e.g., currents in power lines nearby). Temporal (1D) convolution exists in both methods, yet 1D-CNN also contains nonlinear operations and more depths in feature extractions. Predictions by 1D-CNN improve over Prony's method considerably [9]. To go further along this line, we notice that transformers ([10, 11] and references therein) have achieved state-of-the-art performance in machine translation and language modeling due to their more non-local representation power than convolutions. However, transformers suffer from quadratic costs in computational time and memory footprint with respect to the sequence length, which can be prohibitively expensive when learning long sequences. Moreover, redundant queries in transformers can degrade the prediction accuracy. Leveraging group Lasso-based structured sparsity regularization, we optimize the sparsity of queries of the self-attention mechanism, resulting in a reduced number of queries and computational cost. We further integrate 1D-CNN layers into transformers with sparse queries for power grid prediction and show remarkable improvement over existing methods.

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

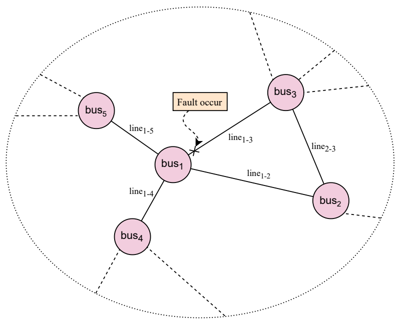

Fig. 1: Snapshot illustrates a faulty power grid system with voltages/currents recorded by sensors on buses/lines.

2. METHODOLOGY

The proposed model is based on the encoder-decoder architecture [10] with efficient attention. An overview is shown in the top panel of Fig. 2, and more details are described below.

2.1. Transformer

Embedding. In the voltage prediction of the post-fault power grid system, the ability to capture long-term voltage volatility patterns requires both global information like time stamps and early-stage temporal time-series features from connected neighbors. To this end, we use a uniform representation embedding layer based on padding and convolution to mitigate mismatch between time stamps and temporal inputs in the encoder and decoder.

The input signals of a single bus in a fault event is $\boldsymbol{X} = \{\dots, \boldsymbol{x}_i, \dots\}$, $1 \le i \le F$ and $\boldsymbol{x}_i \in \mathbb{R}^L$. Where F is he number of input signals, counting both voltage and current connected or adjacent to the bus and L is length of signal measured. The global time stamp $\boldsymbol{x}^s \in \mathbb{R}^L$ records temporal (positional) relationship to the fault time t_f. The encoder input is $\boldsymbol{x}^{en} = \{\boldsymbol{x}; \boldsymbol{x}^s\}$. For input of decoder, only the features of initial period before time t_f is used. We take zero padding to match dimension, and get $\boldsymbol{x}^{de} = \{\dots, \boldsymbol{x}_i^{de}, \dots; \boldsymbol{x}^s\}$, where $\boldsymbol{x}_i^{de} = \{x_{i,t_0}, \dots, x_{i,t_f}, 0, \dots, 0\}$. This embedding ensures causality and is written as

$$\boldsymbol{X} = \mathrm{ELU}(\mathrm{Concat}(\mathrm{Conv1d}(\boldsymbol{x}_{in}), \mathrm{Conv1d}(\boldsymbol{x}^s))),$$

where $\boldsymbol{x}_{in}$ is either $\boldsymbol{x}^{en}$ or $\boldsymbol{x}^{de}$; and $\mathrm{ELU}(u) = \exp\{u\} - 1$ if $u < 0$, $\mathrm{ELU}(u) = u$ if $u \ge 0$, component-wise on a vector.

Encoder. After embedding and representations, the input signal is formed as $\boldsymbol{X}_{en} \in \mathbb{R}^{L \times d_{model}}$, see Fig.2 (top). In the encoder, there are two identical layers; each consists of multi-head Group Sparse attention and a 1-D convolution layer. The sub-layer, Group Sparse attention, is discussed in detail in Sec.2.2 for removing redundancy in the vanilla attention module [10]. The 2nd sub-layer going forward is written as

$$\mathrm{ELU}(\mathrm{Conv1d}(\mathrm{Attn}(X_{en})))$$

where Conv1d performs 1-D convolutional filtering in time dimension [8], followed by the ELU activation function [12].

Decoder. The embedding layer before decoder shapes the inputs feature as $\boldsymbol{X}_{de} \in \mathbb{R}^{L \times d_{model}}$, where d_{model} is the

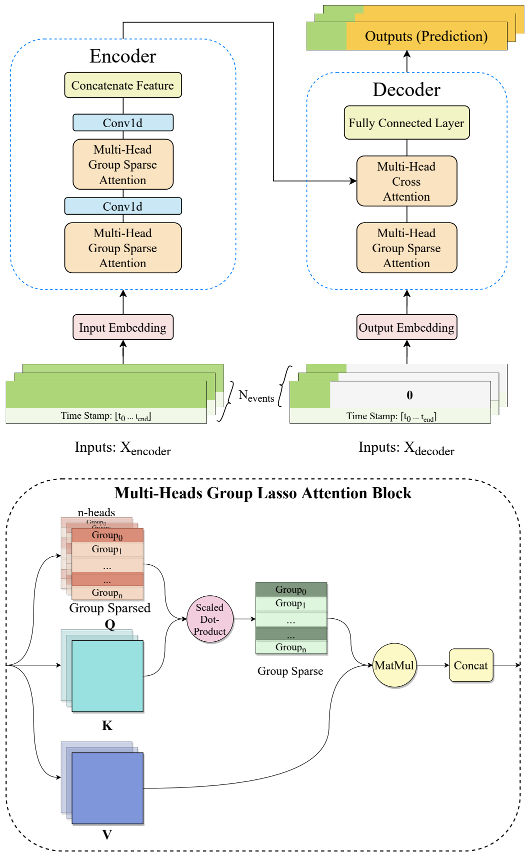

Fig. 2: Top panel: overall encoder-decoder architecture of GLassoformer, bottom panel: GLassoformer's attention block.

hidden dimension of model. This layer enables Group Sparse attention to perform the same as in the encoder. After multi-head Group Sparse attention, a canonical multi-head cross attention [12] combines the hidden feature from encoder and $\mathrm{Attn}(\boldsymbol{X}_{de})$. A fully connected layer after cross attention matches the output dimension with that of the prediction variables. The final output is

$$\boldsymbol{y}^n = \mathrm{Full}(\mathrm{CrossAttn}(F_{hidden}, \mathrm{Attn}(\boldsymbol{X}_{de}))),$$

where Full and CrossAttn stand for the fully connected and cross attention layers respectively, and F_{hidden} is the hidden feature from the encoder layer.

2.2. Self-attention mechanism and query sparsification

The self-attention mechanism [10] is used to learn long-range dependencies and enable parallel processing of the input sequence. For a given input sequence $\tilde{\boldsymbol{X}} := [\tilde{\boldsymbol{x}}_1, \cdots, \tilde{\boldsymbol{x}}_N]^\top$, $\tilde{\boldsymbol{x}} \in \mathbb{R}^{D_x}$, self-attention transforms $\tilde{\boldsymbol{X}}$ into an output sequence $\hat{\boldsymbol{V}}$ in the following two steps[1]:

Step 1. Project the input sequence $\tilde{\boldsymbol{X}}$ into three matrices via the following linear transformations

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

[1]For simplicity of the following discussion, we formulate the self-attention mechanism slightly differently from that in [10].

where $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{W}_V \in \mathbb{R}^{D_x \times N}$ are the weight matrices. We denote $N \times N$ matrices $\boldsymbol{Q} := [\boldsymbol{q}_1, \cdots, \boldsymbol{q}_N]^\top$, $\boldsymbol{K} := [\boldsymbol{k}_1, \cdots, \boldsymbol{k}_N]^\top$, and $\boldsymbol{V} := [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_N]^\top$, where the vectors $\boldsymbol{q}_i, \boldsymbol{k}_i, \boldsymbol{v}_i$ for $i = 1, \cdots, N$ are query, key, and value vectors respectively.

Step 2. For each query vector $\boldsymbol{q}_i$ for $i = 1, \cdots, N$, we compute the output vector $\hat{\boldsymbol{v}}_i$ as follows

$$\hat{\boldsymbol{v}}_i = \sum_{j=1}^{N} \text{softmax}\Big(\frac{\boldsymbol{q}_i^\top \boldsymbol{k}_j}{\sqrt{D}}\Big)\boldsymbol{v}_j, \tag{1}$$

i.e.,

$$\hat{\boldsymbol{V}} = \text{softmax}\Big(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D}}\Big)\mathbf{V} := \boldsymbol{A}\boldsymbol{V}, \tag{2}$$

where the softmax is applied row-wise.

For long sequences, the computational and memory costs of transformers are dominated by (1). It is evident that the memory cost is $\mathcal{O}(N^2)$ to store the attention matrix $\boldsymbol{A}$. Also, the computational complexities of computing the matrix-matrix products $\boldsymbol{Q}\boldsymbol{K}^\top$ and $\boldsymbol{A}\boldsymbol{V}$ are both $\mathcal{O}(N^2)$. In response, efficient transformers have been proposed leveraging low-rank and/or sparse approximation of $\boldsymbol{A}$ [13, 14, 15, 16], locality-sensitive hashing [17], clustered attention [18], etc.

In [11], k-means clustering identifies most relevant keys to reduce the number of query-key pairs. In [12], an empirical formula based on Kullback-Leibler divergence is derived to score the likeness of $\boldsymbol{Q}$ and $\boldsymbol{K}$, and select top few row vectors in $\boldsymbol{Q}$. To reduce the score computation to $\mathcal{O}(N \log N)$ complexity, random sampling of $\boldsymbol{Q}$ vectors is used to approximate the score formula. The resulting top score $\boldsymbol{Q}$ vectors remain and others are zeroed out. Such a group sparsification of $\boldsymbol{Q}$ helps remove insignificant part of the full attention matrix, also lower computation and storage costs to $\mathcal{O}(N \log N)$.

We observe from (2) that if query matrix $\boldsymbol{Q}$ only has $\mathcal{O}(1)$ many nonzero rows (or group sparsity), the complexity of $\boldsymbol{A}\boldsymbol{V}$ goes down to $\mathcal{O}(N)$. This amounts to replacing the empirical score formula in [12] by a data-driven row selection in $\boldsymbol{Q}$. To this end, we adopt the group Lasso (GLasso [19]) penalty *to sparsify columns of $\boldsymbol{W}_Q$ so that the number of nonzero rows of $\boldsymbol{Q} = \boldsymbol{W}_Q^\top \tilde{\boldsymbol{X}}^\top$ is reduced to $\mathcal{O}(1)$ which in turn lowers complexity of $\boldsymbol{A}\,\boldsymbol{V}$ to $\mathcal{O}(N)$*. Though GLasso computation without any query vector sampling does not reduce $\mathcal{O}(N^2)$ complexity of attention computation in training (it surely does so at inference), this is a minor problem for our moderate-size data set and network with a few million parameters in this study. The training times are about the same with or without a random sampling of query vectors on the power grid data set (Tab. 4). More importantly, our resulting network (GLassoformer) is faster and more accurate than Informer [12] during testing and inference, see Tabs. 1 and 2.

3. ALGORITHM AND CONVERGENCE

We present our algorithm to realize row-wise sparsity in query matrix $\boldsymbol{Q}$ and show its convergence guarantee.

3.1. Algorithm

Let $(\theta, \boldsymbol{W}_Q)$ denote model parameters, where $\boldsymbol{W}_Q$ is the query weights in the attention mechanism. We divide $\boldsymbol{W}_Q$ into column-wise groups: $\boldsymbol{W}_Q = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_g, \ldots, \boldsymbol{w}_N\}$. The penalized total loss is:

$$L(\theta, \boldsymbol{W}_Q) = f(\theta, \boldsymbol{W}_Q) + \lambda \left\|\boldsymbol{W}_Q\right\|_{GL} \tag{3}$$

where f is the mean squared loss function, and $\left\|\boldsymbol{W}_Q\right\|_{GL} \triangleq \sum_{g=1}^{N} \left\|\boldsymbol{w}_g\right\|_2$ is the GLasso penalty [19]. For network training, we employ the Relaxed Group-wise Splitting Method (RGSM, [20]) which outperforms standard gradient descent on related group sparsity (channel pruning) task of convolutional networks for loss functions of the type (3). To this end, we pose the proximal problem:

$$\boldsymbol{y}_g^* = \arg\min_{y_g} \lambda \left\|\boldsymbol{y}_g\right\|_2 + \sum_{i \in I_g} \frac{1}{2} \left\|\boldsymbol{y}_{g,i} - \boldsymbol{w}_{g,i}\right\|_2^2 \tag{4}$$

where I_g is the column index set of $\boldsymbol{w}$ in group g. Solution of (4) is a soft-thresholding function:

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

The RGSM algorithm minimizes a relaxation of (3):

$$L_\beta(\theta, \boldsymbol{w}, \boldsymbol{u}) := f(\theta, \boldsymbol{w}) + \lambda \left\|\boldsymbol{u}\right\|_{GL} + \frac{\beta}{2} \left\|\boldsymbol{w} - \boldsymbol{u}\right\|_2^2 \tag{5}$$

by iterations:

$$\boldsymbol{u}_g^t = \mathbf{Prox}_\lambda(\boldsymbol{w}_g^t), \qquad \text{for } g = 1, \ldots, N \tag{6}$$

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

with $\nabla := \nabla_{\theta,\boldsymbol{w}}$, learning rate η, relaxation parameter $\beta > 0$.

3.2. Convergence Theory

Thanks to the ELU activation function (continuously differentiable with piece-wise bounded second derivative), the network mean squared loss function f satisfies Lipschitz gradient inequality for some positive constant L_{ip}:

$$\|\nabla f(\boldsymbol{v}_1) - \nabla f(\boldsymbol{v}_2)\| \leq L_{ip} \|\boldsymbol{v}_1 - \boldsymbol{v}_2\| \tag{7}$$

and any $\boldsymbol{v}_i := (\theta_i, \omega_i)$, $i = 1, 2$. Notice that $\|\cdot\|_{GL}$ violates (7). The advantage of splitting in RGSM (6) is to overcome this lack of non-smoothness in convergence theory. We state the following (the proof follows from applying (7) and (6), similar to Appendix A of [21])

Lemma 1 (Descent Inequality)

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

which implies

Theorem 1 *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.*

4. EXPERIMENTAL SETTINGS

Datasets. We carry out experiments on the simulated New

	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 1: Voltage prediction error comparison.
I (II): input data with (without) neighbor voltage and current features.

York/New England 16-generator 68-bus power system [22]. The data set takes over 2248 fault events, where each event has signals of 10 seconds long. These signals contain voltage and frequency from every bus, current from every line. The dataset also records the graph structure of buses and lines, see Fig. 1. The system has 68 buses and 88 lines linking them, i.e., 68 nodes with 88 edges in the graph. To explore the prediction accuracy of the GLassoformer model, we take the voltage of a bus as the training target, and voltage with currents from locally connected buses and lines as input features. The splitting for train/val/test is 1000/350/750 fault events.

Experimental Details. We implemented our model on PyTorch. **Baselines:** we selected several time-series prediction models to compare, including 1D-CNN, Informer, Lasso[2], and Prony's method (for voltage signal of single bus, case II in Tab 1). **Architecture:** The encoder contains 2 Group Sparse attention layers, and decoder consists of 1-layer Group Sparse self-attention and 1-layer cross-attention. We use Adam with a learning rate η of $1e-4$ and rate decay by 0.8 after every 10 epochs. For hyper-parameters in group Lasso, β is 0.9 and λ is 0.01. During training, we use a batch size of 30. The maximal training epoch number is 80 with early stopping patience of 30. **Setup:** The input is zero-mean normalized. **Platform:** The model is trained on Nvidia GTX-1080Ti.

5. RESULTS

Fig. 4 shows that group and regular Lasso affect our network training as seen from visualizations of structured/unstructured sparse query $\boldsymbol{Q}$ (left/right), where entries with magnitudes below 1e-5 are zeroed out. In Fig. 3, two sample predictions from GLassoformer (green) and Informer (orange) are compared with the blue test data (to the right of the vertical dashed line). The blue curve to the left of the dashed line is short time observation after the fault (network input). Informer's prediction contains several spurious spikes that conceivably come from the random sampling procedure in their attention module for linear complexity. In Tab. 3, we see that the Informer's training time is slightly shorter even though it has three attention layers, while G/LassoFormer has two attention layers in the encoder. The training time saving is limited on our data set. However, GLassoformer has much better prediction accuracies with and without nearby voltage and line current features, as shown in Tab. 1 in terms of mean squares error (MSE) and mean absolute error (MAE). The Glassformer model is smaller in parameter size and faster at inference than Informer (Tab. 2).

[2]Transformer with regular (un-structured) Lasso penalty.

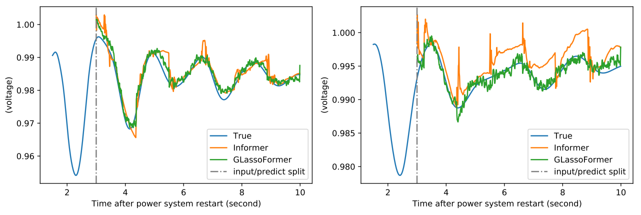

Fig. 3: Sample post-fault voltage predictions in time (to the right of the vertical dashed line at time 3) with input from the left of the line.

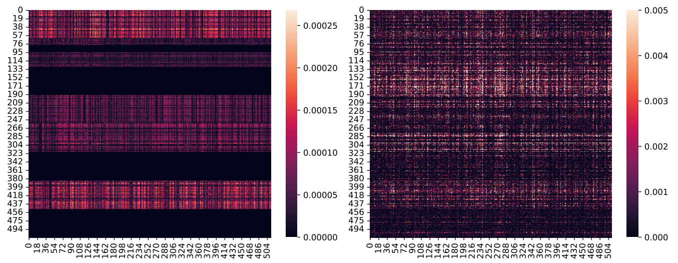
Fig. 4: Visualization of sparsity (black pixel) patterns in a sample query matrix $\boldsymbol{Q}$ by GLassoformer (left) and Lassoformer (right).

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 2: 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 3: Pruning rate and training time comparison.
[1]: fraction of zero query vectors (threshold = 1e-5); [2]: time per epoch

	Vanilla
MSE(e-5)	6.32
MAE(e-3)	3.93
Pruning rate (%)	2.3
Training time (s)	8.75
Inference Time (ms)	15.32

Table 4: Vanilla Transformer

6. CONCLUSION

We presented GLassoformer, a transformer neural network model with query vector sparsity for time-series prediction, and applied it to power grid data. The model is trained through group Lasso penalty and a relaxed group-wise splitting algorithm with theoretical convergence guarantee. The model's post-fault voltage prediction is much more accurate and rapid than the recent Informer [12], also outperformed other benchmark methods in accuracy remarkably.

7. REFERENCES

[1] W. Bo, B. Fang, Y. Wang, H. Liu, and Y. Liu, "Power system transient stability assessment based on big data and the core vector machine," *IEEE Transactions on Smart Grid*, vol. 7, no. 5, pp. 1–1, 2016.

[2] J. James, D. Hill, A. Lam, J. Gu, and V. Li, "Intelligent time-adaptive transient stability assessment system," *IEEE Transactions on Power Systems*, vol. 33, no. 1, pp. 1049–1058, 2017.

[3] R. Yan, G. Geng, Q. Jiang, and Y. Li, "Fast transient stability batch assessment using cascaded convolutional neural networks," *IEEE Transactions on Power Systems*, pp. 2802–2813, 2019.

[4] C. Liu, S. Kai, Z. H. Rather, C. Zhe, and P. Lund, "A systematic approach for dynamic security assessment and the corresponding preventive control scheme based on decision trees," *IEEE Transactions on Power Systems*, vol. 29, no. 2, pp. 717–730, 2014.

[5] A. Lotufo, M. Lopes, and C. Minussi, "Sensitivity analysis by neural networks applied to power systems transient stability," *Electric Power Systems Research*, vol. 77, no. 7, pp. 730–738, 2007.

[6] L. Zhu, D. Hill, and C. Lu, "Hierarchical deep learning machine for power system online transient stability prediction," *IEEE Transactions on Power Systems*, 2020.

[7] J. Li, M. Yue, Y. Zhao, and G. Lin, "Machine-learning-based online transient analysis via iterative computation of generator dynamics," *in Proc. IEEE SmartGridComm*, 2020.

[8] M. Hayes, *Statistical Digital Processing and Modeling*, John Wiley & Sons, 1996.

[9] C. Hu, G. Lin, B. Wang, M. Yue, and J. Xin, "Post-fault power grid voltage prediction via 1d-cnn with spatial coupling," *in Proc. of International Conference on AI for Industries*, 2021.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[11] A. Roy, M. Saffar, A. Vaswani, and D. Grangier, "Efficient content-based sparse attention with routing transformers," *arXiv:2004.05997v5*, Oct 2020.

[12] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proc. of the Association for the Advancement of Artificial Intelligence*, 2021.

[13] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are RNNs: Fast autoregressive transformers with linear attention," in *Proceedings of the 37th International Conference on Machine Learning*, Hal Daumé III and Aarti Singh, Eds., 13–18 Jul 2020, vol. 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165.

[14] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlós, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, D. Belanger, L. Colwell, and A. Weller, "Rethinking attention with performers," in *International Conference on Learning Representations*, 2021.

[15] I. Beltagy, M. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[16] J. Ainslie, S. Ontanon, C. Alberti, V. Cvicek, Z. Fisher, P. Pham, A. Ravula, S. Sanghai, Q. Wang, and L. Yang, "ETC: Encoding long and structured inputs in transformers," in *Proc. of the 2020 Conference on Empirical Methods in Natural Language Processing*, Online, Nov. 2020, pp. 268–284, Association for Computational Linguistics.

[17] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *International Conference on Learning Representations*, 2020.

[18] A. Vyas, A. Katharopoulos, and F. Fleuret, "Fast transformers with clustered attention," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[19] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society, Series B*, vol. 68(1), pp. 49–67, 2007.

[20] B. Yang, J. Lyu, S. Zhang, Y-Y Qi, and J. Xin, "Channel pruning for deep neural networks via a relaxed groupwise splitting method," *In Proc. of International Conference on AI for Industries, Laguna Hills, CA*, 2019.

[21] T. Dinh, B. Wang, A. Bertozzi, S. Osher, and J. Xin, "Sparsity meets robustness: Channel pruning for the Feynman-Kac formalism principled robust deep neural nets," *in Proc. of International Conference on Machine Learning, Optimization, and Data Science*, 2020.

[22] "Power system toolbox," Available at `https://www.ecse.rpi.edu/~chowj/`.