Weather-Coupled Power Grid Predictive Modeling and Causality Inference

Wei-Chun Chang

February 7, 2025

1 Literature Review

Realistic large-scale synthetic power grids with electric substation latitudes and longitudes is presented in [1, 2]. Load profiles are generated by [3]. Direct inclusion of weather measurements to power flow simulation is proposed in [4]. Power flow simulation using weather measurements of higher-quality is presented in [5, 6].

Causal inference model with (V)AE is presented in [7]. Topology-aware OPF learning framework is proposed in [8].

2 Problem Statement

The interplay between weather conditions and the efficient operation of electric grids is well acknowledged. However, traditional grid analysis tools have often overlooked explicit integration of weather data. This paper presents novel advancements in leveraging weather information for enhancing power flow and optimal power flow (OPF) analyses.

Key Challenges This study tackles several pivotal challenges, including the accessibility of comprehensive weather data suitable for power flow analysis, the establishment of effective mappings between weather parameters and grid elements, the development of adaptable and scalable modeling techniques for incorporating weather variables into power flow models, and the visualization of weather-induced influences on resultant grid outcomes.

Introduction This paper introduces a novel approach utilizing a topology-informed graph neural network (GNN) to forecast voltage levels and detect line flow congestion events within power grids. By integrating grid topology into the neural network architecture, the GNN framework leverages the inherent locality property of locational

information [8], in our case, the weather information and power flow.
The primary objective of this study is to harness the capabilities of graph neural networks (GNNs) in embedding grid topology within a weather information framework. Our proposed methodology introduces an innovative approach to predict topology-dependent voltage magnitudes and line congestion events. Firstly, we train the model to predict voltage values. Subsequently, we train a model to identify congested events. Our GNN model effectively exploits the sparse graph structure inherent in power grids, thereby reducing model complexity and enhancing generalizability.

3 Learning Model

3.1 Weather Information

With the increasing prominence of renewable energy sources, the influence of weather conditions on energy production has become more pronounced. Among various weather parameters, temperature stands out as a dominant factor affecting energy load. Additionally, renewable sources such as wind turbines and solar panels are closely tied to wind speed and solar irradiance, respectively. To capture the impact of weather on power generation accurately, clear numerical values are essential. In this study, we focus on three primary weather variables: temperature, wind speed, and irradiance. These variables are chosen based on their strong correlation with both power generation and usage. It is important to note that while these variables are considered independent, they collectively provide valuable insights into the weather's effect on energy systems. This states the key issue we want to tackle using representation learning techniques. To account for the impact of cloud coverage on solar irradiance, we introduce a discount factor by performing element-wise multiplication between irradiance and cloud coverage. This approach acknowledges that greater cloud coverage can hinder solar irradiance, thereby affecting energy generation from solar sources. As illustrated in the following figure, irradiance typically reaches its peak during noon hours, as indicated by the x-axis representing the increment of hours. By incorporating this adjustment, our model provides a more comprehensive representation of weather effects on power systems.

3.2 GNN for direct PF prediction

3.2.1 Background(this is other's work)

Suppose we are given a graph $G = (\mathcal{V}, \mathcal{E}, A)$, which consists of $N = |\mathcal{V}|$ vertices and $|\mathcal{E}|$ edges such that an edge between any two vertices i and j represents their similarity. The corresponding adjacency matrix A is an $N \times N$ sparse matrix with (i, j) entry equaling to 1 if there is an edge between i and j and 0 otherwise. Also, each node

is associated with an F-dimensional feature vector and $X \in \mathbb{R}^{N \times F}$ denotes the feature matrix for all N nodes. An L-layer GCN [**?**] consists of L graph convolution layers and each of them constructs embeddings for each node by mixing the embeddings of the node's neighbors in the graph from the previous layer:

$$Z^{(l+1)} = A' X^{(l)} W^{(l)}, \;\; X^{(l+1)} = \sigma(Z^{(l+1)}), \tag{1}$$

where $X^{(l)} \in \mathbb{R}^{N \times F_l}$ is the embedding at the l-th layer for all the N nodes and $X^{(0)} = X$; A' is the normalized and regularized adjacency matrix and $W^{(l)} \in \mathbb{R}^{F_l \times F_{l+1}}$ is the feature transformation matrix which will be learnt for the downstream tasks. Note that for simplicity we assume the feature dimensions are the same for all layers ($F_1 = \cdots = F_L = F$). The activation function $\sigma(\cdot)$ is usually set to be the element-wise ReLU.

Semi-supervised node classification is a popular application of GCN. When using GCN for this application, the goal is to learn weight matrices in (1) by minimizing the loss function:

$$\mathcal{L} = \frac{1}{|_L|} \sum_{i \in _L} \text{loss}(y_i, z_i^L), \tag{2}$$

where $_L$ contains all the labels for the labeled nodes; $z_i^{(L)}$ is the i-th row of $Z^{(L)}$ with the ground-truth label to be y_i, indicating the final layer prediction of node i. In practice, a cross-entropy loss is commonly used for node classification in multi-class or multi-label problems.

3.2.2 Voltage Prediction

The voltage values are derived from synthetic data representing the power grid. The goal of voltage prediction is to estimate voltage levels throughout the entire year. Our methodology focuses on the power network, comprising nodes and transmission lines, as the primary target for prediction. Weather stations provide data on various weather features, including temperature, wind speed, and solar radiation combined with cloud coverage. This information is collectively represented as an undirected acyclic graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. The node set $\mathcal{V}$ includes N nodes, which can correspond to the number of buses in one scenario or the transmission lines in another. The edge set $\mathcal{E}$ captures the relationships between weather stations and the prediction targets.

3.2.3 Dataset

The synthetic dataset used for this study was collected from the year 2016. The bus voltage values range from a minimum of 0.96 per unit (p.u.) to a maximum of 1.48 p.u., with a mean voltage of 1.039 p.u. These values have been normalized to fall within the range of 0 to 1.

Consider a graph with n features and k nodes. The voltage and line flow connectivity can be expressed as:

$$\mathbf{f}_{kn} = \mathcal{S}_{km}\mathbf{p}_{mn}$$

Where:
- $\mathbf{f}$ is the input to the graph neural network after multiplication, with dimensions of 6717 by the number of selected features (initially, 3 weather factors).
- $\mathbf{p}$ represents the weather stations with the three features, having dimensions of 137 by 3.
- The incidence matrix $\mathcal{S}$ has dimensions $(\text{number of nodes}, \text{number of nodes})$.

This formulation integrates weather data with the power grid structure, enabling the prediction of voltage levels through the application of graph neural networks.

3.3 Propagation and Transformation (l-th layer)

The feature transformation in each layer of the Graph Convolutional Network (GCN) can be expressed as:

$$Z^{(l-1)'} = HXW^{(l-1)} + b^{(l)}$$

where:
- H is the adjacency matrix with dimensions 6717×6717, representing the connectivity in the graph.
- X is the feature matrix from the previous layer.
- $W^{(l-1)}$ is the weight matrix of the $(l-1)$-th GCN layer, with dimensions corresponding to the number of features in the previous layer and the number of features in the current layer l.
- $b^{(l)}$ denotes the bias for each hidden node in the l-th layer.

In this formulation, $Z^{(l-1)'}$ is the result of multiplying the feature matrix X with the weight matrix $W^{(l-1)}$ and adding the bias term $b^{(l)}$. After the initial transformation, the element-wise sum of the propagated features $S^{(l)}$ and the bias term $b^{(l)}$ is calculated.

A non-linear activation function (ReLU, denoted by σ) is then applied to the result. The output feature matrix of the l-th GCN layer, denoted as $Z^{(l)}$, is obtained as follows:

$$Z^{(l)} = \sigma(Z^{(l-1)'})$$

This process effectively captures the complex relationships within the graph, incorporating both the structural information and the features associated with each node, facilitating accurate voltage level predictions across the power grid.

3.3.1 Line Congestion Classification

Graph Convolutional Networks (GCNs) share similarities with traditional Convolutional Neural Networks (CNNs) in their approach to feature learning. While CNNs operate on grid-like data structures, GCNs learn features by examining neighboring nodes in a graph. GCNs aggregate node vectors, pass the aggregated result through a dense layer, and apply non-linearity using an activation function. In essence, GCNs consist of three key components: graph convolution, a linear layer, and a non-linear activation function. There are two major types of GCNs: Spatial Convolutional Networks and Spectral Convolutional Networks.

In practical applications, one of our primary concerns is the proximity of a line to exceeding its maximum usage, which translates to the likelihood of line congestion. To address this, we use a binary label for each line to indicate whether congestion or a significant event is occurring. Our model is designed to predict line congestion events, effectively classifying lines based on their risk of becoming congested.

This predictive capability is crucial for maintaining the reliability and efficiency of the power grid. By accurately forecasting congestion events, we can implement preemptive measures to mitigate risks, ensuring the stability of the power network.

3.4 Causal inference for PF events

The fields of machine learning and graphical causality have historically developed independently. However, there is growing interest in leveraging advancements from both fields to enhance each other. In the context of weather and power flow prediction, the relationship is not straightforward, and simple statistical methods may not yield the best results. This is due to potential causal relationships within the weather data that need to be understood and accounted for.

The objective, therefore, is to uncover structural knowledge and unobserved variables within the weather data that influence power flow. This understanding allows for more effective interventions and adjustments. Unlike many traditional approaches in causality, which assume that causal variables are predefined, a significant challenge for AI and causality is to identify high-level causal variables from low-level observations. This process is known as causal representation learning.

Causal representation learning aims to discover and represent the underlying causal mechanisms that govern the observed data. By doing so, it provides a framework for making informed predictions and interventions, which is particularly valuable in complex systems like power grids where multiple interacting factors influence outcomes. Integrating these insights into machine learning models can significantly improve the accuracy and robustness of predictions, particularly in predicting events such as line congestion in power networks.

3.5 Autoencoder

Autoencoder for causal inference [xx]. Sparse autoencoder [9, 10]. The approach aims to better understand subsets of input features captured by each latent variable and to simplify the model by pruning small weight parameters, such as using only one encoder layer. We implement an autoencoder for predicting voltage at each bus. If there is an inherent structure in the data, such as correlations between input features, the autoencoder can learn this structure by forcing the input through the network's bottleneck.

Our goal is to extract features that dominate the causal connections. By taking an unlabeled dataset and framing it as a supervised learning problem aimed at reconstructing the original input $\mathcal{X}$, the network can be trained to minimize the reconstruction error $\mathcal{L}(x, \hat{x})$, which measures the differences between the original input and its reconstruction.

The bottleneck is a crucial element of our network design. It ensures that the network does not simply memorize the input values by passing them through without transformation. Instead, it forces the network to learn a compressed representation that captures the most relevant information for reconstruction.

The autoencoder can be mathematically described by the loss function:

$$\mathcal{L}(\mathbf{x}, g(f(\mathbf{x})))$$

Here, f and g are nonlinear functions representing the encoder and decoder, respectively. The encoder $f(\mathbf{x})$ maps the input $\mathbf{x}$ to a latent representation, and the decoder g reconstructs the input from this latent representation. The loss function $\mathcal{L}(\mathbf{x}, g(f(\mathbf{x})))$ quantifies the reconstruction error, guiding the training process to minimize this error and thus improve the quality of the learned representations.

By incorporating the information bottleneck, the autoencoder effectively captures the dominant causal features in the data, facilitating more accurate voltage predictions across the power grid. This method leverages the structural dependencies in the data to enhance the predictive capability of the model.

3.5.1 Loss Function Design

To effectively combine the reconstruction loss and the prediction loss, we formulate the total loss function as a weighted sum of both components. This allows us to proportionally balance the importance of accurately reconstructing the input features and predicting the target values. The total loss function is given by:

$$\text{Total loss} = \lambda_1 \times \text{MSE}(x, \hat{x}) + \lambda_2 \times \text{MSE}(y, \hat{y})$$

where: - $\text{MSE}(x, \hat{x})$ is the mean squared error between the original input x and its reconstruction $\hat{x}$, quantifying the reconstruction accuracy. - $\text{MSE}(y, \hat{y})$ is the mean squared error between the true target values y and the predicted values $\hat{y}$, quantifying the prediction accuracy. - λ_1 and λ_2 are the weighting coefficients that balance the contributions of the reconstruction loss and the prediction loss, respectively.

The combined loss function ensures that the autoencoder not only learns to reconstruct the input features accurately but also makes precise predictions for the target values. By adjusting λ_1 and λ_2, we can control the trade-off between reconstruction quality and prediction accuracy, tailoring the model to the specific requirements of the task.

The total loss function can be expressed as:

$$\text{Total loss} = \lambda_1 \times \text{MSE}(x, \hat{x}) + \lambda_2 \times \text{MSE}(y, \hat{y})$$

This formulation leverages the autoencoder's ability to capture latent structures in the input data while simultaneously ensuring that the model is optimized for predictive performance on the target variable.

3.6 Sparse Autoencoder

(this is from the SAE class notes https://graphics.stanford.edu/courses/cs233-21-spring/ReferencedPapers/SAE.pdf) The autoencoder is designed to learn a function $h_{W,b}(x) \approx x$, effectively approximating the identity function to produce an output $\hat{x}$ that closely resembles the input $\mathbf{x}$. By imposing constraints on the network, such as limiting the number of hidden units, we can force the model to uncover and learn the underlying structure of the input data. In our specific architecture, we use only 64 hidden units, compelling the network to learn a compressed representation of the input.

This constraint means that given only the vector of hidden unit activations $a^{(2)} \in \mathbb{R}^{64}$, the network must attempt to reconstruct the 411-unit input. If there are correlations among the input features, this approach will enable the algorithm to identify and leverage those correlations. To maintain simplicity and interpretability of the weights, we implement a single hidden layer.

The linear layers in our autoencoder perform a linear transformation of the input data, defined by:

$$y = Wx + b$$

where: - W is the weight matrix, - x is the input vector, - b is the bias vector, - y is the output vector after the linear transformation.

Given this framework, the autoencoder's structure can be summarized as follows:

- Encoder: Maps the input $\mathbf{x} \in \mathbb{R}^{411}$ to a hidden representation $a^{(2)} \in \mathbb{R}^{64}$.
- Decoder: Attempts to reconstruct the input from the hidden representation.

The loss function for training the autoencoder combines the reconstruction error with the prediction error:

$$\text{Total loss} = \lambda_1 \times \text{MSE}(\mathbf{x}, \hat{\mathbf{x}}) + \lambda_2 \times \text{MSE}(y, \hat{y})$$

This setup encourages the network to not only reconstruct the input accurately but also make precise predictions for the target variable, balancing both aspects through the weighting coefficients λ_1 and λ_2. By learning the compressed representation and the associated linear transformations, the autoencoder can effectively capture the essential structures and correlations within the input data, facilitating more accurate voltage predictions across the power grid. In order to single out the weights, we only implement one layer in our sparse autoencoder set up. Among the numbers of latent variables, empirically, 64 being the size of latent variables yields the best reesults.

3.6.1 Sparse Loss

To enhance the interpretability and efficiency of our model, we aim to constrain the neurons to be inactive most of the time. This approach helps in identifying the neurons that learn the most informative features. To achieve this, we impose a LASSO (Least Absolute Shrinkage and Selection Operator) penalty in our loss function. The sparsity loss function incorporates the mean absolute value of the activations in each layer (excluding the last layer) to promote sparsity.

The mathematical formulation of our objective with the LASSO penalty is:

$$\text{minimize} \left(\text{SSE} + \lambda \sum_{j=1}^{p} |\beta_j| \right)$$

Where: - SSE (Sum of Squared Errors) is calculated as:

$$\text{SSE} = \frac{1}{N} \sum_{i=1}^{N} |a_i|$$

Here, N is the number of elements in the activation vector, and a_i is the value of the i-th activation in the layer. - λ is the regularization parameter controlling the sparsity. - β_j represents the coefficients of the activations.

The LASSO penalty constrains the size of the coefficients such that any increase in the coefficients must be accompanied by a comparable decrease in the sum of squared errors (SSE). This constraint forces the hidden unit activations to be mostly near zero, thus promoting sparsity.

Incorporating this penalty into our autoencoder, the total loss function becomes:

$$\text{Total loss} = \lambda_1 \times \text{MSE}(\mathbf{x}, \hat{\mathbf{x}}) + \lambda_2 \times \text{MSE}(y, \hat{y}) + \lambda_3 \sum_{j=1}^{p} |\beta_j|$$

Where: - λ_1 and λ_2 are the weighting coefficients for the reconstruction and prediction losses, respectively. - λ_3 is the coefficient for the LASSO penalty.

By adding the LASSO penalty, we ensure that only a small subset of the features, those exhibiting the strongest effects, are selected. This adheres to the "bet on sparsity" principle (Hastie, Tibshirani, and Wainwright, 2015). Preliminary results show that with different selections of λ_3, the LASSO penalty effectively pushes many coefficients to zero, thus enforcing sparsity and improving the model's ability to identify key features.

This design leads to a more interpretable and efficient model, where the activations of the neurons that contribute the most to learning are emphasized, ultimately enhancing the overall performance of the autoencoder in voltage prediction tasks across the power grid.

3.6.2 Regularizer Selection

(This part is from https://bradleyboehmke.github.io/HOML/regularized-regression.html) Linear models (LMs) provide a simple and effective approach to predictive modeling. When the assumptions required by LMs, such as constant variance, are met, the estimated coefficients are unbiased and have the lowest variance among all linear unbiased estimates. To improve the robustness of our linear models and prevent overfitting, we apply multiple regularization techniques.

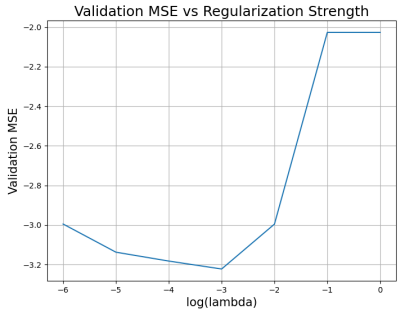
In our approach, we explore a range of regularization strengths by using 10 as the base for the penalty parameter, with the exponent ranging from -6 to 0. This allows us to systematically evaluate the impact of regularization on model performance.

Our observations reveal a slight fluctuation in the mean squared error (MSE) as the penalty parameter $\log(\lambda)$ increases. This suggests that a standard ordinary least squares

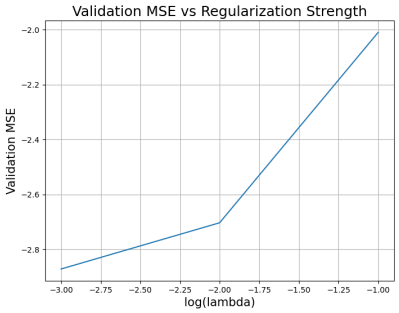
(OLS) model likely overfits the training data, as indicated by lower MSE values with minimal or no regularization. However, as we further constrain the model by increasing the penalty, the MSE initially decreases, reaching an optimal point, and then starts to increase again.

Specifically, we find that the optimal regularization occurs at $\log(\lambda) = 0.1$. At this point, the MSE is minimized, indicating a balance between bias and variance that prevents overfitting while maintaining good generalization performance. Beyond this optimal point, increasing the regularization strength further leads to a rise in MSE, likely due to excessive bias introduced by the regularization.

Thus, our findings underscore the importance of selecting an appropriate regularization strength to achieve the best predictive performance in linear models. The use of regularization helps in mitigating overfitting and enhancing the model's ability to generalize well to unseen data.

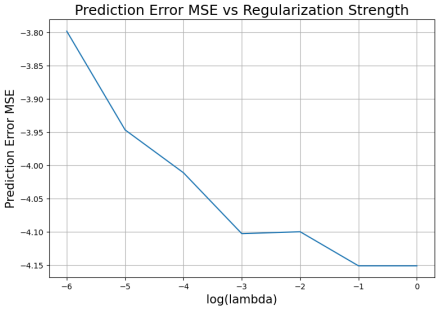


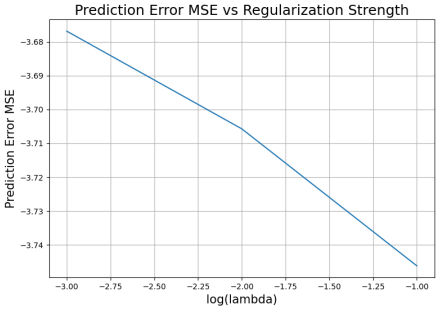
(a) MSE across regularizers(-6 to 0)



(b) MSE across regularizers(-3 to 0)

Therefore, we select the exponent -1 as the optimal regularizer, as it also corresponds to the lowest prediction mean square error, as indicated by our figures.



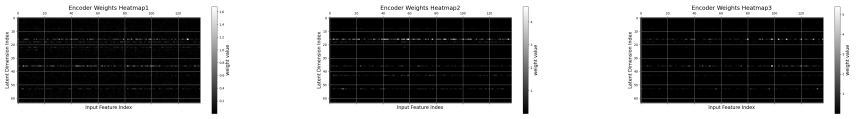


(a) Prediction MSE across regularizers(-6 to 0) (b) Prediction MSE across regularizers(-3 to 0)

3.6.3 Weight Analysis

By observing which weights are pushed to zero, we can identify the active latent variables, indicating the latent variables that inherit non-zero weights. This allows us to pinpoint the weather stations that possess the most important information. Consequently, we can reduce the dimensions of the input and gain insight into subgroups of weather stations that sufficiently represent the dataset.

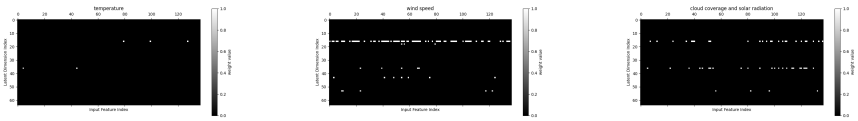
To achieve this, we set the regularizer as 0.1 and plotted the absolute weight matrix as shown below:



(a) Temperature (b) Wind Speed (c) Solar Irradiance

Figure 3: Weight matrix for 3 features

To identify the significant weights, we filter out entries close to zero and emphasize those with higher absolute values. This process allows us to pinpoint the weights that carry the most importance in our model.



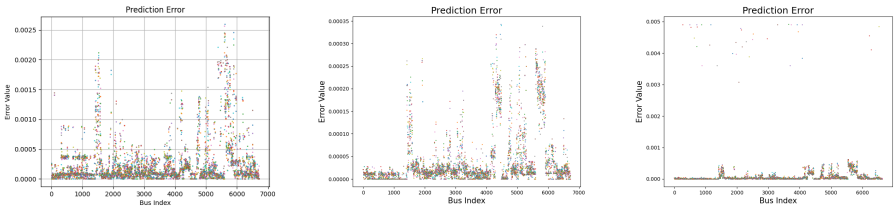
(a) Temperature (b) Wind Speed (c) Solar Irradiance

Figure 4: Weight matrix classification for 3 features

4 Preliminary Results

4.1 Prediction Error

We utilized a pretrained model to predict a sample size of (1757, 411), resulting in predictions of shape (1757, 6717). Using mean square error (MSE) as our criterion, we compared the prediction errors for each model in a table. The results indicate low prediction errors across all models, with the autoencoder (AE) demonstrating the lowest error. Following closely, the Sparse AE model achieved the second-best performance in terms of MSE.



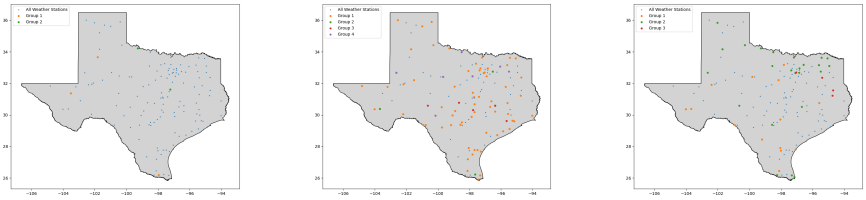
(a) GNN (b) Autoencoder (c) Sparse Autoencoder

Figure 5: Prediction Error for Training Models

Table 1: Error Prediction Results

Prediction Error			
Modeling Approach	GNN	Autoencoder	Sparse Autoencoder
MSE for sample (Max/Min/Mean)	0.00047, 0.00974, 0.00084	3.216e-06, 0.00194, 3.642e-05	2.707e-05, 0.0022, 7.063e-05
Square root of MSE for sample (Max/Min/Mean)	0.02159, 0.09871, 0.02889	0.00179, 0.04406, 0.00604	0.00520, 0.0470 , 0.00840
MSE for buses (Max/Min/Mean)	2.6806e-08, 0.00959, 0.00084	5.1197e-12, 0.00034, 3.6422e-05	0.0, 0.01455, 7.0633e-05
Square root of MSE for buses (Max/Min/Mean)	0.00016372, 0.09793676, 0.02889975	2.2627e-06, 1.8511e-02, 6.0350e-03	0. , 0.12062, 0.00840

4.1.1 Weather Station and Renewables Buses Visualization



(a) Temperature (b) Wind Speed (c) Solar Irradiance

Figure 6: Active Buses in Latent Variable Groups

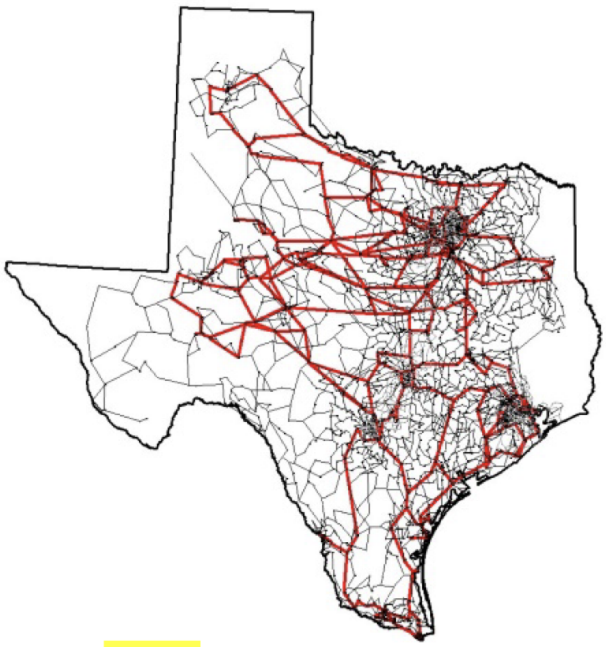


Figure 7: One-line diagram of the synthetic 6700-Bus Texas Grid.

The oneline for the synthetic grid, which has 6700 buses and covers a geographic footprint of most of the US state of Texas, is shown in Figure 7. The grid's nominal voltage levels are 345 kV (red), 138 kV (black) and 69 kV (gray). The grid has a total of about 105 GW of generation, with 25 GW of wind and 2.3 GW of solar. Datasets are generated using PowerWorld Simulator Version 23.

References

[1] Adam B Birchfield, Ti Xu, Kathleen M Gegner, Komal S Shetye, and Thomas J Overbye. Grid structural characteristics as validation criteria for synthetic networks. *IEEE Transactions on power systems*, 32(4):3258–3265, 2016.

[2] Ti Xu, Adam B Birchfield, Kathleen M Gegner, Komal S Shetye, and Thomas J Overbye. Application of large-scale synthetic power system models for energy economic studies. In *50th Annual Hawaii International Conference on System Sciences, HICSS 2017*, 2017.

[3] Hanyue Li, Ju Hee Yeo, Ashly L. Bornsheuer, and Thomas J. Overbye. The creation and validation of load time series for synthetic electric power systems. *IEEE Transactions on Power Systems*, 36(2):961–969, 2021.

[4] Thomas Overbye, Farnaz Safdarian, Wei Trinh, Zeyu Mao, Jonathan Snodgrass, and Juhee Yeo. An approach for the direct inclusion of weather information in the power flow. In *56th Annual Hawaii International Conference on System Sciences, HICSS 2023*, 2023.

[5] Farnaz Safdarian, Melvin Stevens, Jonathan Snodgrass, and Thomas J Overbye. Detailed hourly weather measurements for power system applications. In *2024 IEEE Texas Power and Energy Conference (TPEC)*, pages 1–6. IEEE, 2024.

[6] H. Hersbach, B. Bell, P. Berrisford, G. Biavati, A. Horányi, J. Muñoz Sabater, J. Nicolas, C. Peubey, R. Radu, I. Rozum, D. Schepers, A. Simmons, C. Soci, D. Dee, and J-N Thépaut. Era5 hourly data on single levels from 1940 to present (accessed on 23-01-2024). *Copernicus Climate Change Service (C3S) Climate Data Store (CDS)*, 2023.

[7] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

[8] Shaohui Liu, Chengyang Wu, and Hao Zhu. Topology-aware graph neural networks for learning feasible and adaptive ac-opf solutions. *IEEE Transactions on Power Systems*, 2022.

[9] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[10] Alireza Makhzani and Brendan Frey. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.