CS230

Learning Power Flow Mappings for Power Grid Simulation

Lily Buechler
Department of Mechanical Engineering
Stanford University
`ebuech@stanford.edu`

1 Introduction

Power system simulation engines are essential tools for power grid operation and planning by utility companies and academic researchers. However, simulations can be computationally expensive given the wide range of timescales of grid operation. Additionally, the accuracy of utility distribution system models has traditionally been limited, although operational data is now becoming more abundant. The objective of this project is to develop a deep learning framework for approximating the outputs from a power flow simulation, and evaluate performance for a variety of power network characteristics. This model could be used for: (1) accelerating simulation speed when trained on the outputs from a traditional power flow solver, or (2) improving network model accuracy when trained on real operational data. These methods would be particularly useful in applications requiring real-time simulation or optimization.

The purpose of a power flow solver is to perform the inverse power flow mapping. That is, given the real and reactive power injections at each bus (node) in the power network and the admittance matrix of the system, solve for the magnitude and phase of the voltage at each bus in the network. For a deep learning model, the output is a vector of the voltage magnitudes for every bus and phase, and the input is a feature vector composed of the real and reactive power injections at nominal voltage for each bus and phase. We also investigate how incorporating other types of network information improves model accuracy.

2 Related work

The type of power flow simulation under consideration is a quasi-static timeseries (QSTS) simulation, which uses the Newton Raphson method to solve the static nonlinear power flow equations (1). Power flow solutions must be performed chronologically which prevents sequential solutions from being parallelized. Literature has investigated methods for accelerating power flow simulations (2; 3; 4; 5; 6), including using machine or deep learning methods to approximate the powerflow mapping (7; 6). These studies have focused primarily on linear regression (6), SVR (7), multilayer perceptron neural networks (8), and radial basis function neural networks (9). The deep learning approaches focus entirely on fully-connected architectures and typically use the second-order Levenberg-Marguardt method for training (9; 8). Some studies train a different neural network for each node in the system. Most prior studies have focused on relatively small power systems and assume that no information is available on power system topology. In this study, we test methods on a range of power system sizes to evaluate model scalability and demonstrate that performance can be improved by utilizing additional information about the phase of the power injections.

3 Dataset and Features

The datasets used for this study are the input and output data from a physics-based power flow simulator GridLAB-D (10). Datasets were generated for six different power networks (11; 12) under a variety of loading conditions. The input and output dimensions of the power networks are shown in Table 1. Sixty-one days of simulation was performed at a 1-minute timestep for each

Table 1: Input and output dimensions of the evaluated power flow problems.

Power Network	Output (voltage) dimension	Input (power) dimension
IEEE 4 bus	12	3
IEEE 13 bus	48	20
IEEE 123 bus	402	95
PNNL GC-12.47-1	108	9
PNNL R1-12.47-3	297	37
PNNL R2-12.47-2	2553	214

network scenario (87,840 samples) to generate training (82,080 samples), validation (2,880 samples), and test (2,880 samples) sets, which come from roughly the same distribution. For simplicity, we assume that the power factor is constant over time, so that real and reactive power injections are proportional. Therefore only the real power injections at each node are used as features. Both the inputs and outputs were normalized to have zero mean and unit variance. For most networks, the power-voltage relationship is fairly linear for low loading conditions, and highly nonlinear for high loading conditions. When generating datasets, the power injections were calibrated such that the power network voltages were in both the linear and nonlinear regimes. The adjacency matrix of the network and the phase of each power injection was also recorded.

4 Methods

Current literature on using neural networks for power flow approximation has been limited to single-layer fully-connected architectures (9; 8). Therefore, we use a fully-connected architecture as a baseline model and perform hyperparameter tuning to optimize the number of layers, amount of L2 regularization, type of activation (tanh and ReLU), and number of hidden units. The dimension of the input layer (n_x) is equal to the number of power injections and the dimension of the output layer (n_y) is equal to the number of power network buses times the number of phases, as shown in Fig. 1a for a single hidden layer. Therefore a single neural network is used to predict all of the voltage outputs at all nodes, unlike previous studies that train separate models for each node, which is very computationally expensive for large networks. A mean squared error loss was used for model training using Adam optimization ($\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$). Since power flow mappings tend to be fairly linear in the low loading regime, linear regression was also used as a baseline model.

Fully-connected architectures do not account for the sparsity of the topology of the power network. While utilities may have limited knowledge of power network parameters, the network topology is more likely to be known. Convolutional graph neural networks make use of network sparsity and extend the convolution operation traditionally applied in the Euclidean domain to graph structured data (13). We investigated the use of graph convolutional networks (GCNs) proposed in (14) for this application. However the model resulted in poor performance, which can be attributed to two main factors. First, the voltage outputs depend on the power injections at all nodes in the network. Since graph convolution operations aggregate feature information only from neighboring nodes, this requires extremely deep neural networks to propagate information through large power networks. Second, graph convolutions assume parameters can be shared among nodes, while in reality, power flow is spatially dependent since branch impedances can vary significantly throughout the network. Prediction errors of the GCN model were more than an order of magnitude larger than all other models and therefore have not been included in the results section. While other types of graph convolutional networks might work well for this problem, this project focused on other approaches.

Modern power systems have three phases of sinusoidal alternating current distribution, where each phase is offset by approximately 120°. QSTS simulation assumes that the power injections may be unbalanced among the three phases. Therefore, the voltage magnitude depends not only on the location and magnitude of the power injections but also the phase to which the load is connected. We incorporate knowledge about which phase each power injection is connected to into the model to analyze the effect on prediction accuracy. This is done by formatting the power injection features into three channels corresponding to the three different phases. A sequence of convolutional and fully-connected layers is used to transform these features, as shown in Fig 1b. The input dimensions of the different channels may be different depending on the number of power injections connected to each phase. Note that in the more general case where reactive power is included as a feature, the filter

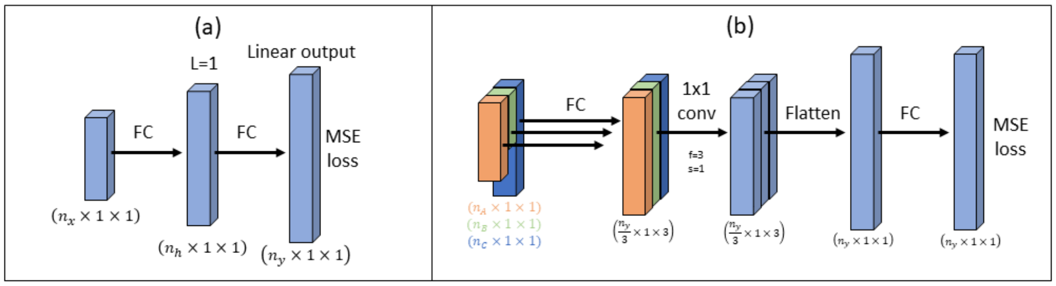
(a)
L=1
Linear output
FC
FC
MSE loss
$(n_x \times 1 \times 1)$
$(n_h \times 1 \times 1)$
$(n_y \times 1 \times 1)$
(b)
FC
1x1 conv
f=3
s=1
Flatten
FC
MSE loss
$(n_A \times 1 \times 1)$
$(n_B \times 1 \times 1)$
$(n_C \times 1 \times 1)$
$(\frac{n_y}{3} \times 1 \times 3)$
$(\frac{n_y}{3} \times 1 \times 3)$
$(n_y \times 1 \times 1)$
$(n_y \times 1 \times 1)$

Figure 1: Model architectures for the (a) fully connected and (b) convolutional models.

dimension would be 1x2 and the second dimension of the input layer and first hidden layer would be equal to 2. Adam optimization was used for training ($\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$) and both tanh and ReLU activations were evaluated.

The fully-connected, convolutional, and GCN models were implemented using the Tensorflow (15) and Keras (16) deep learning frameworks on an AWS EC2 instance. The Scikit-Learn package (17) was used for data pre-processing and learning the linear regression model.

5 Results and Discussion

For a single voltage magnitude output vector $\hat{V}^{(i)}$, the performance is evaluated in terms of the normalized voltage magnitude error $\epsilon_v^{(i)}$:

$$\epsilon_v^{(i)} = \left\| \frac{\hat{V}^{(i)} - V^{(i)}}{V^{(i)}} \right\|_\infty \tag{1}$$

The max norm is used to consider the least accurate prediction for all nodes in the power network. The mean value over m examples is given by:

$$\mu_{\epsilon_v} = \frac{1}{m} \sum_{i=1}^{m} \epsilon_v^{(i)} \tag{2}$$

Fig. 2 shows the mean normalized voltage prediction error μ_{ϵ_v} on the validation set as a function of the size of the training set for the single layer fully-connected network, linear regression, and the convolutional model. Each model was trained for 500 epochs, which was more than sufficient to fully minimize the loss function. Increasing the number of layers in the fully-connected network did not significantly improve performance. The best performance was achieved with a small amount of L2 regularization ($\alpha = 1e$-6) for the fully-connected network and no regularization for the convolutional model.

In all power networks except for the IEEE 4 bus network, the convolutional model performs as well as or better than the fully-connected network. This demonstrates that the additional structure provided by the convolutions allows the model to learn the dependencies across phases. The relative performance of the fully-connected and convolutional models for tanh and ReLU activations depends on the power network and the amount of training data available. However the tanh activation tends to most reliably produce the most accurate results across a variety of networks and operating conditions. The validation and test errors (μ_{ϵ_v}) for the convolutional model with tanh activation are shown in Table 2. The errors are similar across the validation and test sets for all models, which is expected given that the validation and test sets come from the same distribution.

Results indicate that for both the fully-connected and convolutional models, the amount of required training data scales with the size of the power network. Approximately 10,000 samples are necessary for the three smaller networks (IEEE 4, IEEE 13, and GC-12.47-1) and 20,000-40,000 samples are required for the three larger networks (IEEE 123, GC-12.47-1, and R2-12.47-2) to mitigate variance issues. The performance of the linear regression model does not improve with more than 1,000 samples. The improvement in performance of the fully-connected network over the linear regression baseline diminishes with the size of the power network, however the performance of the convolutional model is more scalable.

For the fully-connected model, the effect of the number of units in each layer on the prediction error for the validation set is shown in Fig. 3 for the six different power networks. As expected, the

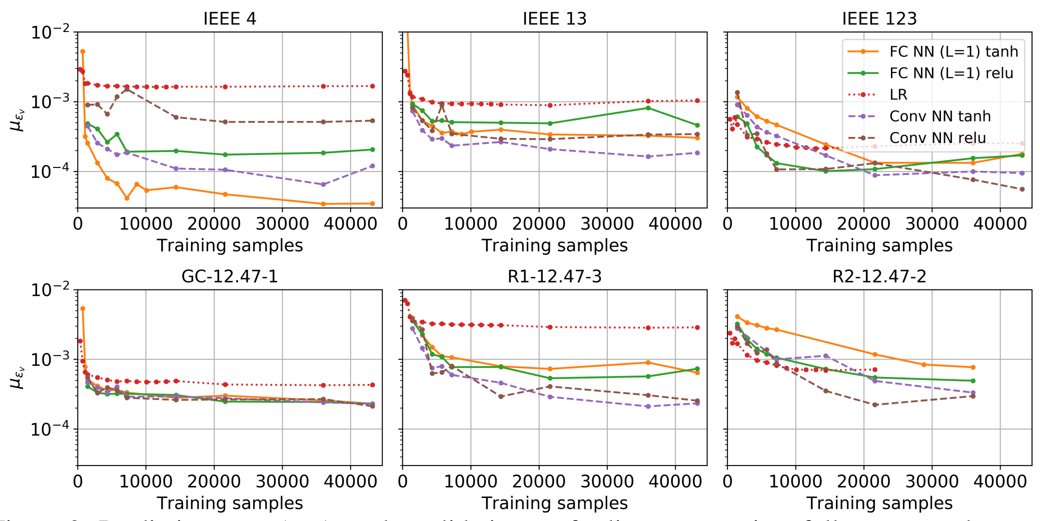

Figure 2: Prediction error (μ_{ϵ_v}) on the validation set for linear regression, fully-connected neural network (FC NN) and convolutional neural network (Conv NN) as a function of the number of training samples and fully-connected layers. Results are shown for both tanh and ReLU activations.

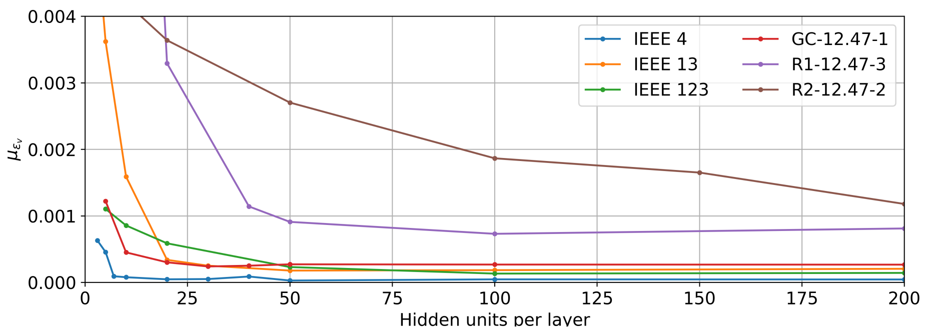

Figure 3: Prediction error (μ_{ϵ_v}) for the single layer fully-connected neural network with tanh activation as a function of the number of units.

optimal number of hidden units scales with both the input and output dimension of the mapping problem. The R2-12.47-2 network requires at least 200 units, the IEEE 123 bus and R1-12.47-3 networks require a minimum of 100 units, while 20 units is sufficient for the smaller power networks. Results indicate that the prediction error is less sensitive to the amount of regularization relative to the other hyperparameters.

The prediction accuracy for all models depends highly on the spatial location in the power network. As shown in Fig. 4 for the IEEE 123 bus network, the prediction error tends to increase further away from the slack bus (bus 150, bottom left), as the voltage deviation becomes larger and the power flow equations become more nonlinear. A level of error of $\mu_{\epsilon_v} < 0.0005$ is acceptable for most applications, which can be achieved with the convolutional model for all power networks. An example timeseries voltage prediction for node 92 phase A in the IEEE 123 bus network is shown in Fig. 5 along with the ground truth data, indicating that they are barely visually distinguishable.

6 Conclusions and Future Work

This study investigated different neural network architectures for approximating the inverse power flow mapping for three-phase unbalanced power system simulation. Results show that a fully-connected architecture can produce accurate results for small networks, but performance diminishes with the size of the power network. Providing additional structure in the model to account for the phase of the power injections using convolutions improves accuracy for larger networks. the prediction error is low enough to be useful for various applications in power system simulation.

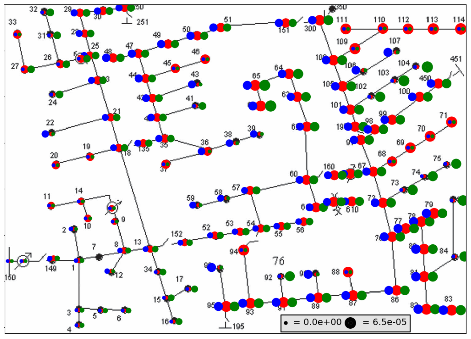

Figure 4: Prediction error μ_{ϵ_v} (indicated by marker size) for each phase (a=blue, b=red, c=green) in the IEEE 123 bus network for a fully-connected single layer neural network.

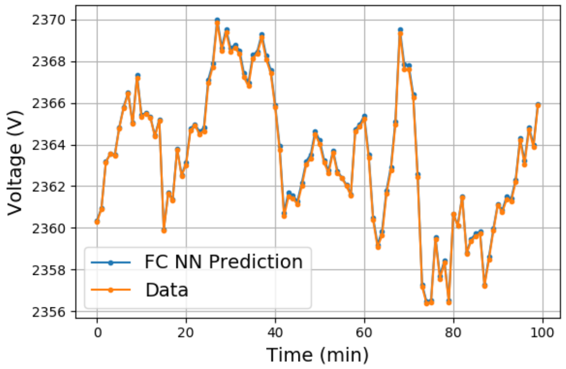

Figure 5: Timeseries prediction of the voltage magnitude for bus 92 phase A in the IEEE 123 network using a single layer fully-connected neural network.

Table 2: Validation and test error (μ_{ϵ_v}) for the convolutional model with tanh activation using 21,600 samples for training.

Power Network	Validation error	Test error
IEEE 4 bus	1.103e-4	1.027e-4
IEEE 13 bus	1.771e-4	1.718e-4
IEEE 123 bus	9.552e-5	9.936e-5
PNNL R1-12.47-3	2.684e-4	2.586e-4
PNNL R2-12.47-2	4.627e-4	4.775e-4

Future work may investigate formulating the neural network inputs and outputs as complex-valued quantities. While prediction of the voltage magnitude and phase can be performed by separate neural networks, the problem is more naturally formulated in the complex domain.

Additional research is needed to determine how to best incorporate different power system controllers, such as voltage regulators and capacitor banks, into a neural net architecture. Accounting for these resources is necessary to perform more complicated types of power grid simulation analysis. The control variable for a voltage regulator is an integer value which indicates its tap changer setting. This information could be used as a feature using a one-hot encoding or the scalar value. The on/off status of a capacitor bank could be encoded as a boolean variable.

The analysis in this project assumed that the topology of the power system is static. However in actual systems, the topology can change due to switches being opened and closed. These topology changes affect the underlying dynamics of the system. Additional research could analyze the best approach to dealing with these changes in the neural network model. While it may be necessary to train entirely different neural networks for each configuration, it may be possible to retrain only a subset of the weights for each topology.

7 Contributions

Lily Buechler worked independently on this project.

8 Acknowledgements

Thank you to David Chassin at SLAC National Accelerator Laboratory for feedback on using the GridLAB-D power system simulator and suggestions on the modeling strategy.

9 Code

Code for this project is available at https://github.com/ebuech/cs230

References

[1] J. Deboever, X. Zhang, M. J. Reno, R. J. Broderick, S. Grijalva, and F. Therrien, "Challenges in reducing the computational time of qsts simulations for distribution system analysis," *SAND2017-5743, Albuquerque, NM*, 2017.

[2] Z. K. Pecenak, V. R. Disfani, M. J. Reno, and J. Kleissl, "Multiphase distribution feeder reduction," *IEEE Transactions on Power Systems*, vol. 33, no. 2, pp. 1320–1328, 2017.

[3] D. Montenegro, R. C. Dugan, and M. J. Reno, "Open source tools for high performance quasi-static-time-series simulation using parallel processing," in *2017 IEEE 44th Photovoltaic Specialist Conference (PVSC)*. IEEE, 2017, pp. 3055–3060.

[4] M. J. Reno and R. J. Broderick, "Predetermined time-step solver for rapid quasi-static time series (qsts) of distribution systems," in *2017 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 2017, pp. 1–5.

[5] J. Deboever, S. Grijalva, M. J. Reno, and R. J. Broderick, "Fast quasi-static time-series (qsts) for yearlong pv impact studies using vector quantization," *Solar Energy*, vol. 159, pp. 538–547, 2018.

[6] S. Powell, A. Ivanova, and D. Chassin, "Fast solutions in power system simulation through coupling with data-driven power flow models for voltage estimation," *arXiv preprint arXiv:2001.01714*, 2020.

[7] J. Yu, Y. Weng, and R. Rajagopal, "Robust mapping rule estimation for power flow analysis in distribution grids," in *2017 North American Power Symposium (NAPS)*. IEEE, 2017, pp. 1–6.

[8] H. H. Muller, M. J. Rider, C. A. Castro, and V. L. Paucar, "Power flow model based on artificial neural networks," in *2005 IEEE Russia Power Tech*. IEEE, 2005, pp. 1–6.

[9] A. Karami and M. Mohammadi, "Radial basis function neural network for power system load-flow," *International Journal of Electrical Power & Energy Systems*, vol. 30, no. 1, pp. 60–66, 2008.

[10] D. P. Chassin, K. Schneider, and C. Gerkensmeyer, "Gridlab-d: An open-source power systems modeling and simulation environment," in *2008 IEEE/PES Transmission and Distribution Conference and Exposition*. IEEE, 2008, pp. 1–5.

[11] W. H. Kersting, "Radial distribution test feeders," *IEEE Transactions on Power Systems*, vol. 6, no. 3, pp. 975–985, 1991.

[12] K. P. Schneider, Y. Chen, D. P. Chassin, R. G. Pratt, D. W. Engel, and S. E. Thompson, "Modern grid initiative distribution taxonomy final report," Pacific Northwest National Lab.(PNNL), Richland, WA (United States), Tech. Rep., 2008.

[13] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *arXiv preprint arXiv:1901.00596*, 2019.

[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[15] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[16] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.