ECSE 6110 Power Engineering Analysis
Fall 2020 Final Project
Admittance Matrix Estimation with Deep Learning

Hayleigh Sanders

Abstract: This project successfully implemented power system admittance matrix estimation using deep learning. Given a set of system measurements, a neural network was trained to predict the power system admittance matrix parameters with high accuracy and robustness to realistic levels of PMU noise. In both the noiseless and noisy cases, the MAPE (mean average percent error) of the predicted parameters was well below 1%.

1 Introduction

Accurate knowledge of grid topology is essential for proper power system management. Efficient methods of topology estimation are necessary due to missing or imprecise admittance information present in many distribution grids. This project will implement and evaluate an admittance matrix estimation method using deep learning. A deep neural network will be trained to predict the admittance matrix parameters based on a set of power flow measurements with Python using the Tensorflow package. The previous work of Albuquerque [1] is highly relevant to this approach. In this article, highly accurate Ybus estimation is achieved through a neural network trained on system measurement data. This project utilized the general neural network architecture outlined in this reference and implemented the same evaluation procedure for noiseless vs noisy system measurement data.

Other relevant background literature is primarily focused on Ybus estimation through statistical methods. Liu [2] estimates the power system topology and Ybus values using a conservative self-adaptive algorithm. Given an initial state vector of current measurements, first and second order optimization using the "Adam" optimizer is performed in combination with the Armijo–Goldstein condition to find a stable solution. Saadeh [3] frames Ybus estimation as a cross-covariance matrix estimation problem. Because the magnitudes and angles from a PMU measurement follow a normal distribution, the relationship between current and voltage measurements can be understood as a cross-covariance matrix, so estimating the cross-covariance for random vectors will approximate the Ybus. Du [4] applies methods from optimal experimental design to online Ybus parameter estimation by incorporating multiple measurement snapshots into a Fisher information matrix, which is then used to obtain the Ybus parameters. Zhao [5] applies a Schweppe-type Hubber estimator to PMU measurements in order to find incorrect branch parameters within a localized solution area.

2 Modeling Approach

2.1 Power System Model

The simple three-bus power system used as a model for this project is shown in the one-line diagram given by Figure 1. The model has two scheduled loads and the line impedences are per unit on a 100MVA base.

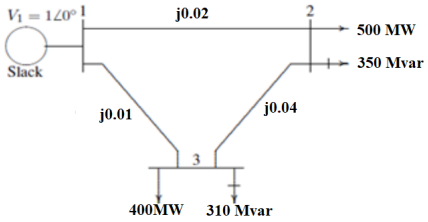

Figure 1: Power system model

2.2 Neural Network Model

2.2.1 Training Data Generation

The power flow equations relating the admittance matrix parameters and system measurements are given by Equations 1, 2, 3 and 4.

$$P_i = V_i \sum_{j=1}^{n} V_j (G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}) \tag{1}$$

$$Q_i = V_i \sum_{j=1}^{n} V_j (G_{ij} \sin\theta_{ij} - B_{ij} \cos\theta_{ij}) \tag{2}$$

$$P_{ij} = V_i^2 (g_{si} + g_{ij}) - V_i V_j (g_{ij} \cos\theta_{ij} + b_{ij} \sin\theta_{ij}) \tag{3}$$

$$Q_{ij} = -V_i^2 (b_{si} + b_{ij}) - V_i V_j (g_{ij} \sin\theta_{ij} - b_{ij} \cos\theta_{ij}) \tag{4}$$

The number of buses is given by n, and $\theta_{ij} = \theta_i - \theta_j$. V_i and θ_i are the voltage magnitude and phase at bus i. Each i,j element of the admittance matrix has the form $G_{ij} + jB_{ij}$. The admittance of each series branch connecting buses i and j has the form $g_{ij} + jb_{ij}$, and the admittance of the shunt branch connected to bus i has the form $g_{si} + jb_{si}$.

The training data set is formed from n feature vectors and n corresponding label vectors, where n is the number of training samples. The true value of the branch admittances can be represented by Equation 5.

$$Y_{true} = (g_{12}, b_{12}, g_{23}, b_{23}, g_{13}, b_{13}) \tag{5}$$

The elements of each label vector Y[n] were computed from the normal distributions as shown in Equation 6.

$$\begin{aligned} Y[n] = (&N(g_{12}, 0.1g_{12}), N(b_{12}, 0.1b_{12}), N(g_{23}, 0.1g_{23}), \\ &N(b_{23}, 0.1b_{23}), N(g_{13}, 0.1g_{13}), N(b_{13}, 0.1b_{13})) \end{aligned} \tag{6}$$

6,000 label vectors were generated using the above equation. Using these branch admittance values, the power flow result for each $Y[n]$ was computed using the Gauss-Seidel method. The feature vector $X[n]$ was then formed as shown in Equation 7.

$$\begin{aligned} X[n] = (&V1.real, V1.imag, V2.real, V2.imag, V3.real, \\ &V3.imag, Psb.real, Psb.imag, i12.real, i12.imag, \\ &i21.real, i21.imag, i13.real, i13.imag, i31.real, \\ &i31.imag, i23.real, i23.imag, i32.real, i32.imag) \end{aligned} \tag{7}$$

The feature vector is formed from the real and imaginary parts of the bus voltages, slack bus voltage, and line flows for each branch.

According to Brown [8], random noise in the PMU measurement can vary from 0.5% to 15%. A second set of feature vectors X_{noisy} was generated along the uniform distribution $U(p-.15p, p+.15p)$, where p is the true value of each feature, so each element will deviate up to 15% from its true value.

2.2.2 Neural Network Architecture

A deep neural network with one input layer, four hidden layers and one output layer was constructed using Tensorflow, which can be visualized in Figure 2. The input layer accepts a tensor of size (n, 20) and the output layer produces a tensor of size (n, 6). A weight matrix exists between each layer. The weights are initialized to random values and an initial prediction $\hat{y}[n]$ for each $x[n]$ is produced. This output is used to produce the output gradient, which is backpropagated through each layer of the neural network to obtain the weight gradients for each layer. Based on these gradients, the weight matrices are updated and the process is repeated for a number of iterations.

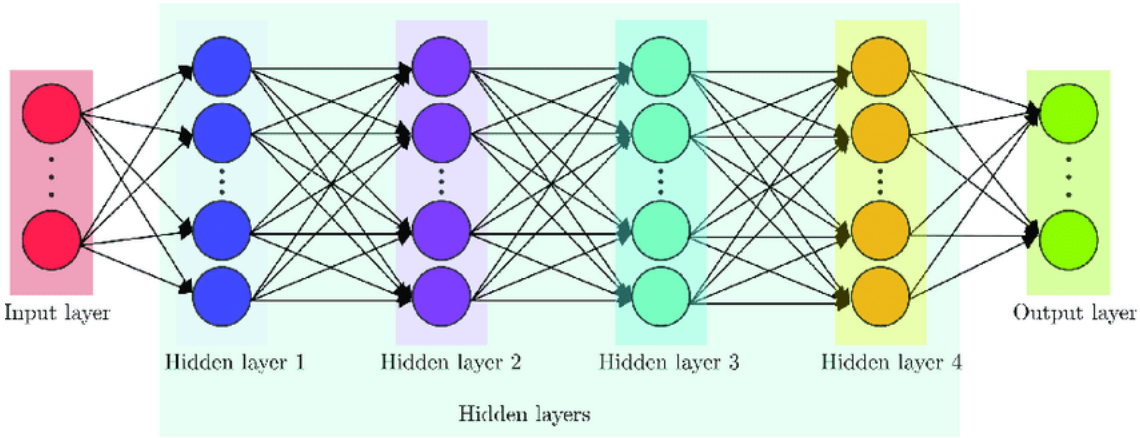

Figure 2: Neural network with four hidden layers. Image credit: Galinos [6]

2.2.3 Loss Function

This project utilized the mean square error (MSE) loss function as shown in Equation 8.

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \tag{8}$$

2.2.4 Forward Pass

Starting at the input layer, for each $x[n]$ in the training data set of size n, its corresponding output $\hat{y}[n]$ is computed from the current weights $W = [W^1, W^2, W^3, W^4, W^5]$ and their biases W_0^1, W_0^2, W_0^3, W_0^4, W_0^5 with the following:

$$H^1[n] = ReLu((W^1)^T * x[n] + W_0^1) \tag{9}$$

$$H^2[n] = ReLu((W^2)^T * H^1 + W_0^2) \tag{10}$$

$$H^3[n] = ReLu((W^3)^T * H^2 + W_0^3) \tag{11}$$

$$H^4[n] = ReLu((W^4)^T * H^3 + W_0^4) \tag{12}$$

$$\hat{y}[n] = ReLu((W^5)^T * H^4 + W_0^5) \tag{13}$$

The ReLu function is given by,

$$ReLu(z) = max(0, z) \tag{14}$$

This process is split into multiple steps:

$$Z^1 = W^1 X + W_{0,1} \tag{15}$$

$$H^1 = ReLu(Z^1) \tag{16}$$

$$Z^2 = W^2 H^1 + W_{0,2} \tag{17}$$

$$H^2 = ReLu(Z^2) \tag{18}$$

$$Z^3 = W^3 H^2 + W_{0,3} \tag{19}$$

$$H^3 = ReLu(Z^3) \tag{20}$$

$$Z^4 = W^4 H^3 + W_{0,4} \tag{21}$$

$$H^4 = ReLu(Z^4) \tag{22}$$

$$Z^5 = W^5 H^4 + W_{0,5} \tag{23}$$

$$\hat{Y} = ReLu(Z^5) \tag{24}$$

2.2.5 Backward Pass

Starting at the output layer, the output gradient is computed and passed over the neural network layers. The weight gradients were found recursively with the following process.

Find the error at the output layer:

$$dZ^5 = (\hat{Y} - Y) \tag{25}$$

Find the gradients at the output layer:

$$dW^5 = (1/n) * dZ^5 * H^{4T} \tag{26}$$

$$dW_{0,5} = (1/n) * sum(dZ^5) \tag{27}$$

Backpropagate through the fourth layer:

$$dH^4 = W^{5T} * dZ^5 \tag{28}$$

$$dZ^4 = dH^4 * dReLu(Z^4) \tag{29}$$

Find the gradients at the fourth layer:

$$dW^4 = (1/n) * dZ^4 * H^{3T} \tag{30}$$

$$dW_{0,4} = (1/n) * sum(dZ^4) \tag{31}$$

Backpropagate through the third layer:

$$dH^3 = W^{4T} * dZ^4 \tag{32}$$

$$dZ^3 = dH^3 * dReLu(Z^3) \tag{33}$$

Find the gradients at the third layer:

$$dW^3 = (1/n) * dZ^3 * H^{2T} \tag{34}$$

$$dW_{0,3} = (1/n) * sum(dZ^3) \tag{35}$$

Backpropagate through the second layer:

$$dH^2 = W^{3T} * dZ^3 \tag{36}$$

$$dZ^2 = dH2 * dReLu(Z^2) \tag{37}$$

Find the gradients at the second layer:

$$dW^2 = (1/n) * dZ^2 * H^{1T} \tag{38}$$

$$dW_{0,2} = (1/n) * sum(dZ^2) \tag{39}$$

Backpropagate through to the first layer:

$$dH^1 = W^{2T} * dZ^2 \tag{40}$$

$$dZ^1 = dH^1 * dReLu(Z^1) \tag{41}$$

Find the gradients at the first layer:

$$dW^1 = (1/n) * dZ^1 * X^T \tag{42}$$

$$dW_{0,1} = (1/n) * sum(dZ^1) \tag{43}$$

Where dReLu is the derivative of the Relu function given by,

$$dReLu(z) = \begin{cases} 0 & z < 0 \\ 1 & z > 0 \\ undefined & z = 0 \end{cases} \tag{44}$$

The weights and their biases for each layer are then updated using the gradients found in the backward pass process with the following:

$$W^1 = W^1 - \eta dW^1 \tag{45}$$

$$W_{0,1} = W_{0,1} - \eta dW_{0,1} \tag{46}$$

$$W^2 = W^2 - \eta dW^2 \tag{47}$$
$$W_{0,2} = W_{0,2} - \eta dW_{0,2} \tag{48}$$

$$W^3 = W^3 - \eta dW^3 \tag{49}$$
$$W_{0,3} = W_{0,3} - \eta dW_{0,3} \tag{50}$$

$$W^4 = W^4 - \eta dW^4 \tag{51}$$
$$W_{0,4} = W_{0,4} - \eta dW_{0,4} \tag{52}$$

$$W^5 = W^5 - \eta dW^5 \tag{53}$$
$$W_{0,5} = W_{0,5} - \eta dW_{0,5} \tag{54}$$

Where η is the learning rate.

2.2.6 Stochastic Gradient Descent

Stochastic gradient descent (SGD) was employed through the use of the Tensorflow "Adam" optimizer to increase the learning efficiency. The SGD algorithm is given in Equation 55,

$$W_k^{t+1} = W_k^{t+1} - \eta * [\frac{1}{S} * \sum_{x[m],y[m] \in D} \frac{\partial L(x[m]y[m]W)}{\partial W}] \tag{55}$$

Instead of using the entire dataset to compute the gradients, the algorithm uses a subset of the data.

2.2.7 Model Hyperparameters

The model hyperparameter choices were guided by Albuquerque [1]. The learning rate $\eta = 0.001$ was the default value for the Tensorflow "Adam" optimizer. Four hidden layers were selected for the neural network, and each hidden layer was composed of 250 nodes. The model was trained for 400 epochs.

3 Analysis and Results

3.1 Model Evaluation Method

Model performance will vary among training sessions. K-fold cross validation was utilized as a result to evaluate the model. The dataset is first split into k equally sized partitions. The first partition is retained as testing data and the other k-1 partitions are used as training data for a model instance. The model is trained and evaluated, then discarded. The next partition is used as testing data with a new model instance and the cycle is repeated until each partition is used as validation data. The performance results from each iteration are averaged. This project used k = 10.

3.2 Model Evaluation Metrics

The root mean-square error (RMSE) is given by Equation 56,

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \tag{56}$$

The mean absolute percentage error (MAPE) is given by Equation 57,

$$MAPE = \frac{\sum_{i=1}^{n} \frac{y_i - \hat{y}_i}{y_i}}{n} \tag{57}$$

According to Wang [7], only results with a MAPE below 1% are suitable for parameter estimation, so this value will be used as a benchmark for interpreting the model evaluation results.

3.3 Results

3.3.1 Noiseless Case

This section will investigate the noiseless case for the power flow measurements. The neural network was trained with the feature vectors X and label vectors Y outlined in Section 2.2.1. The training loss vs epochs is shown in Figure 3. The model appears to converge very quickly.

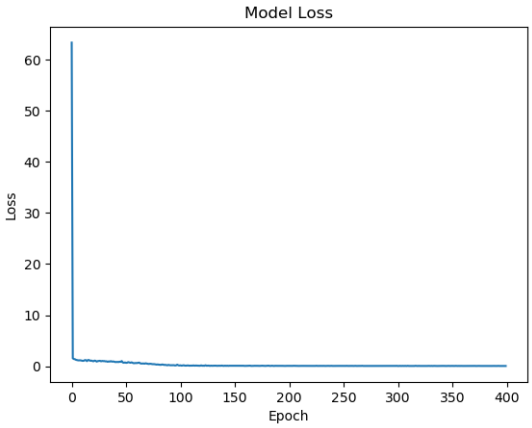

Figure 3: Model loss vs epochs

The model evaluation results are shown in Table 1.

Table 1: Model Evaluation Results

MAPE (%)	RMSE
0.47%	.10

The overall MAPE of the predicted Ybus parameters is 0.47%, which is well below the threshold of 1% needed for parameter estimation. Thus it can be concluded that the model can predict Ybus parameters from system measurement data with high accuracy.

The MAPE for each of the predicted Ybus parameters is shown in Figure 4.

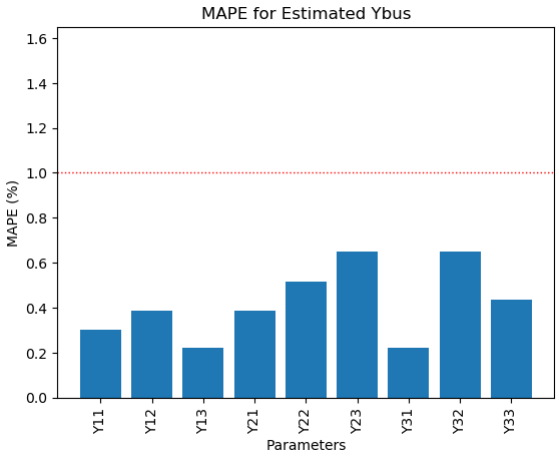

Figure 4: MAPE for each Ybus element

The RMSE for each of the predicted Ybus parameters is shown in Figure 5.

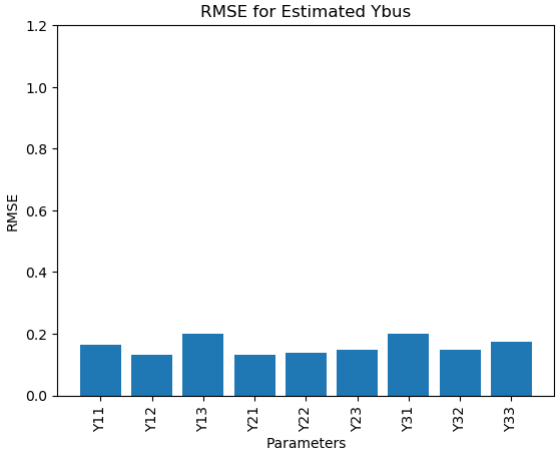

Figure 5: RMSE for each Ybus element

The predicted Ybus values were used to generate power flow results $\hat{X}[n]$, which were then compared to their true values $X[n]$. The MAPE for each of the features is shown in Figure 6.

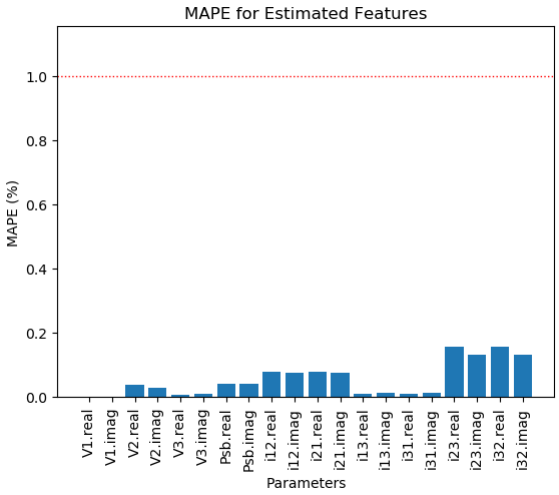

Figure 6: MAPE for each feature

The RMSE for each of the features is shown in Figure 7.

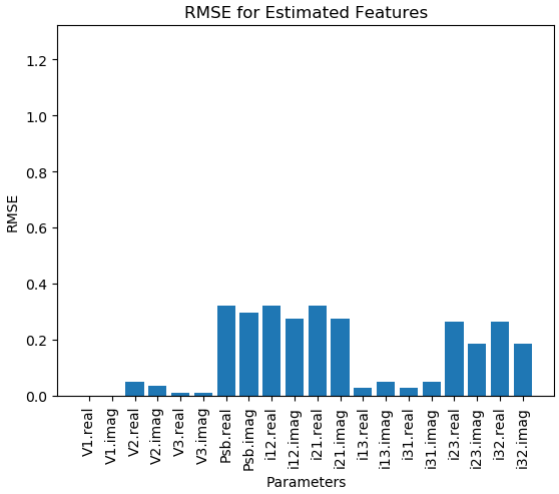

Figure 7: RMSE for each feature

3.3.2 Noisy Case

This section will investigate the robustness of the model against a realistic amount of noise in the power flow measurements. The neural network was trained with the feature vectors X_{noisy} and label vectors Y outlined in Section 2.2.1. The training loss vs epochs is shown in Figure 8.

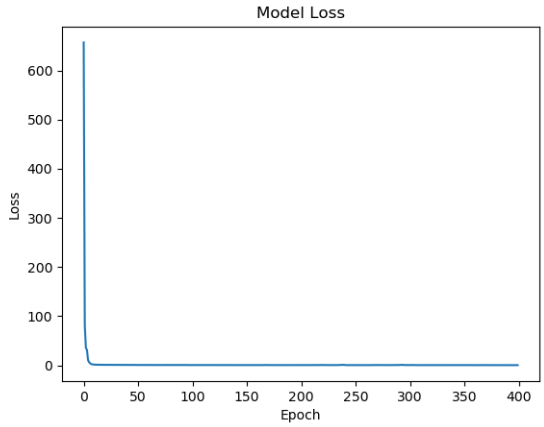

Figure 8: Model loss vs epochs

The model evaluation results are shown in Table 2.

Table 2: Model Evaluation Results

MAPE (%)	RMSE
0.51%	.15

The average MAPE and RMSE barely increased with the inclusion of error in the training data feature vectors, so the model can be considered highly robust against noisy PMU measurements.

The MAPE for each of the predicted Ybus parameters is shown in Figure 9.

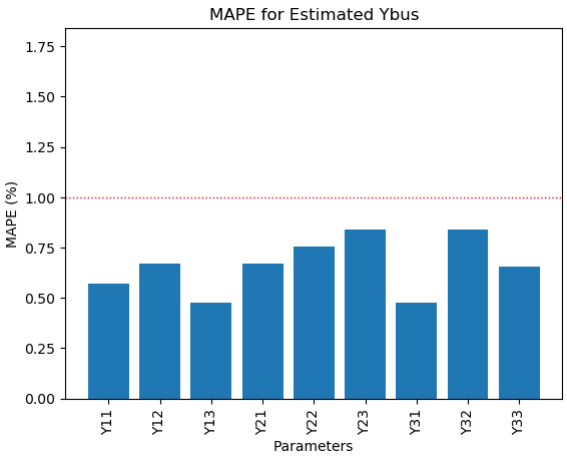

Figure 9: MAPE for each Ybus element

The RMSE for each of the predicted Ybus parameters is shown in Figure 10.

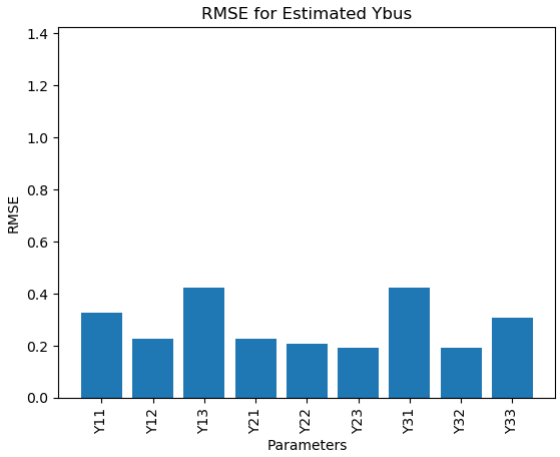

Figure 10: RMSE for each Ybus element

The predicted Ybus values were used to generate power flow results $\hat{X}[n]$, which were then compared to their true values $X[n]$. The MAPE for each of the features is shown in Figure 11.

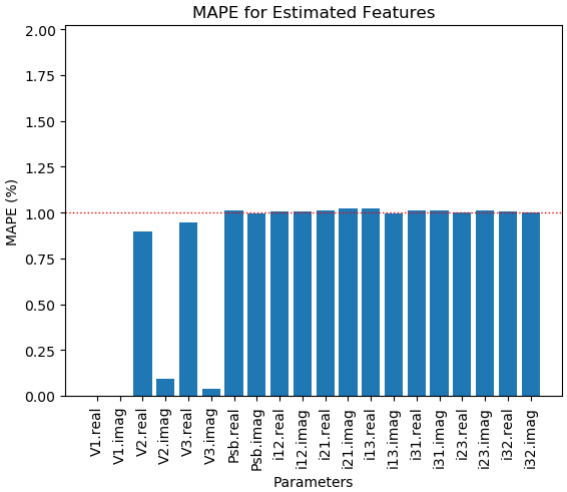

Figure 11: MAPE for each feature

The RMSE for each of the features is shown in Figure 12.

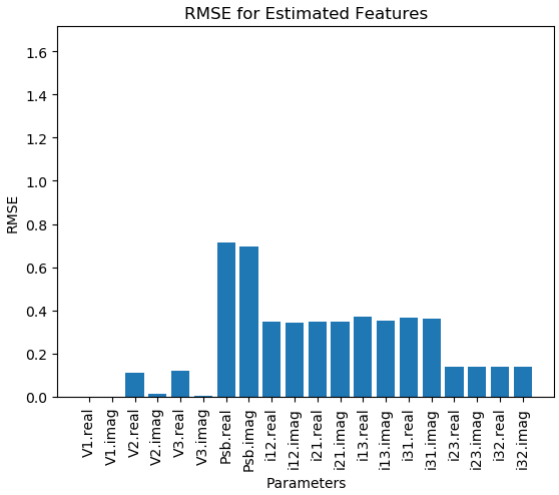

Figure 12: RMSE for each feature

4 Conclusion

A deep neural network model is able to predict admittance matrix parameters from power system measurements with high accuracy and robustness to realistic levels of measurement noise. In both the noiseless and noisy cases, the MAPE (mean absolute percent error) of the predicted parameters was well below 1%.

The model has its limitations despite these promising results. A known grid topology was assumed, which is not always the case for this parameter estimation problem. Likewise, measurement data from each bus and branch was used to form the feature vector for the neural network. In a real world model, measurement data is available for only a fraction of the grid topology because PMUs are expensive and obtaining measurement data from each grid component is not feasible. An improved iteration of this project would implement the sparse feature vector method developed by Albuquerque [1] in which an optimal PMU placement is obtained. In addition, a larger and more realistic power system model should be used in the future, such as the IEEE 33 or 141 bus systems.

References

[1] F. P. de Albuquerque, F. R. Lemes, E. C. M. da Costa, P. T. Caballero, and R. F. R. Pereira, "A Robust machine learning solution for admittance matrix estimation of electric power networks," 2023.

[2] Y. Liu, N. Zhang, Q. Hou, A. Botterud, and C. Kang, " Topology and Admittance Estimation: Precision Limits and Algorithms)," *iEnergy*, 12 2023.

[3] M. Saadeh, R. McCann, M. Alsarray, and O. Saadeh, " A new approach for evaluation of the bus admittance matrix from synchrophasors:(a statistical ybus estimation approach)," *International Journal of Electrical Power Energy Systems*, vol. 93, pp. 395–405, 2017.

[4] X. Du, A. Engelmann, Y. Jiang, T. Faulwasser, and B. Houska, " Optimal Experiment Design for AC Power Systems Admittance Estimation)," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 13 311–13 316, 2020.

[5] J. Zhao, S. Fliscounakis, P. Panciatici, and L. Mili, " Robust parameter estimation of the french power system using field data)," *IEEE Transactions on Smart Grid*, vol. 10, no. 5, pp. 5334–5344, 2018.

[6] C. Galinos, J. Kazda, A. W. H. Lio, and G. Giebel, " T2FL: An Efficient Model for Wind Turbine Fatigue Damage Prediction for the Two-Turbine Case)," *Energies*, vol. 13, no. 6, 2020.

[7] Y. Wang, W. Xu, and J. Shen, " Online tracking of transmission-line parameters using scada data," *IEEE Transactions on Power Delivery*, vol. 31, no. 2, pp. 674–682, 2015.

[8] M. Brown, M. Biswal, S. Brahma, S. J. Ranade, and H. Cao, " Characterizing and quantifying noise in pmu data," *IEEE Power and Energy Society General Meeting (PESGM)*, pp. 1–5, 2016.