Article

Distribution System State Estimation Using Model-Optimized Neural Networks

Doyun Kim, Justin Migo Dolot and Hwachang Song *

Department of Smart Energy System Engineering, Seoul National University of Science and Technology, Seoul 01811, Korea; 1007km@seoultech.ac.kr (D.K.); migo@seoultech.ac.kr (J.M.D.)

* Correspondence: hcsong@seoultech.ac.kr; Tel.: +82-2-970-6403

Featured Application: Power System State Estimation.

Abstract: Maintaining reliability during power system operation relies heavily on the operator's knowledge of the system and its current state. With the increasing complexity of power systems, full system monitoring is needed. Due to the costs to install and maintain measurement devices, a cost-effective optimal placement is normally employed, and as such, state estimation is used to complete the picture. However, in order to provide accurate state estimates in the current power system climate, the models must be fully expanded to include probabilistic uncertainties and non-linear assets. Recognizing its analogous relationship with state estimation, machine learning and its ability to summarize model unseen and complex relationships between input data is used. Thus, a power system state estimator was developed using modified long short-term (LSTM) neural networks to provide quicker and more accurate state estimates over the conventional weighted least squares-based state estimator (WLS-SE). The networks are then subject to standard polynomial scheduled weight pruning to further optimize the size and memory consumption of the neural networks. The state estimators were tested on a hybrid AC/DC distribution system composed of the IEEE 34-bus AC test system and a 9-bus DC microgrid. The conventional WLS-SE has achieved a root mean square error (RMSE) of 0.0151 p.u. for voltage magnitude estimates, while the LSTM's were able to achieve RMSE's between 0.0019 p.u. and 0.0087 p.u., with the latter having 75% weight sparsity, estimates about ten times faster, and half of its full memory requirement occupied.

Keywords: power system; state estimation; machine learning; distribution system; hybrid AC/DC

Citation: Kim, D.; Dolot, J.M.; Song, H. Distribution System State Estimation Using Model-Optimized Neural Networks. *Appl. Sci.* **2022**, *12*, 2073. <https://doi.org/10.3390/app12042073>

Academic Editor: Amjad Anvari-Moghaddam

Received: 12 January 2022

Accepted: 14 February 2022

Published: 16 February 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Reliable electric power transmission and distribution has ever been at the forefront of power system studies. Power system reliability, as defined by [1], encompasses both system adequacy and system security. The former aims to satisfy the balance between energy supply and energy demand within the system, while the latter aims to establish the system's ability to bounce back from disturbances. These days, more and newer devices such as renewable energy sources and converter interfaced generation are introduced to the network. These induce uncertainties as shown by [2] and harmonic disturbances in the voltage signal as shown by [3]. Further, integration of DC sources and loads into hybrid AC/DC distribution systems are also increasing [4]. To help maintain reliability, reference [5] mentions that operators need full system monitoring to maximize renewable energy production and to make a more versatile power system. However, system monitoring implements such as SCADA, AMI, and PMUs are expensive; thus, optimal use is normally required. As proposed in [6], while fully covering the system with measurement devices ensures reliable services, they experience diminishing returns on investment. Due to that, optimal placement papers such as [6–8] exist to attempt to find a configuration which makes the system observable and cost effective. This then leads to the problem of

completing the missing measurements of the current system state given limited information; thus, power system state estimation is developed.

State estimation works by pooling the limited measurement data available and finding a suitable system state that will approximate the given measurements with the least amount of error. This is largely analogous to regression analyses wherein a best fit function's parameters are changed in order to reconcile the data and the function mapping. However, instead of the usual forecasting paradigm, state estimation aims to deduce current values given current and historical input data. The key difficulty here is establishing an effective model to relate the measurement values to the system state, that is, completely modeling the power system including the non-linear components, the noise acquired during data acquisition, and the statistical uncertainties of the renewable energy sources. There are few references that consider hybrid distribution system implementations, some of which include [9,10] wherein the conventional state estimation method has been decentralized to evaluate the AC and DC subgrids separately. The work in [9] capitalized on the duality principle while the work in [10] introduced an intermediate non-linear variable in their three-stage approach. Since the conventional state estimation method is a snapshot in time, the distributed generation and renewable energy sources can be summarily modeled as current injections in the DC subgrid. For references [9,10] too, however, the conventional process is computationally intensive and is open to failures in assumption as [11] mentions.

Due to the nature of this problem and its analog, regression analysis, there exists a method that excels in summarily modeling unseen and complex relationships between data: machine learning, specifically, neural networks. There have already been other allied works that have used neural networks in some form or another for power system state estimation. Works such as [12,13] have used neural networks simply to prepare and clean measurement data to aid in state estimation. One work [14], far back in 1996, has already explored the use of neural networks for topology processing and static state estimation. A more recent work [15] adopts Bayesian state estimation using machine learning to power system state estimation so that estimation of unobservable parts of the system is made possible. Some other works try to include the original components of state estimation into the training process such as [16]. In the same category, another recent work utilizes the concept of optimal measurement device placement to dictate the structure of the neural network [17]. Another work used neural networks to make state *forecasts* instead of estimation such as [18]. These works show that neural networks can indeed be used for state estimation. However, for large, practical systems, the neural networks will tend to be large, heavy, and computationally taxing as well.

This paper proposes taking the neural network application three steps further by (i) adopting the historical measurements and states to improve the accuracy of state estimation, especially for those states near transformers, ii) utilizing the standardized model optimization techniques to lighten the neural network for state estimation, and (iii) expanding its target application to the state estimation for hybrid AC/DC distribution systems. To visualize the thought process, this paper is structured as follows: Section 2 discusses the basic intuition on power system state estimation and the conventional method: weighted least squares state estimation (WLS-SE). It also shows the main objective of state estimation, i.e., to minimize the error between the calculated and the measured system state. Section 3 discusses neural networks and its learning process which involves minimizing errors between initial and target predictions and using those errors to update its own parameters. The section also ties machine learning to state estimation by showing where it can be used within state estimation and the advantages it has over the conventional WLS-SE. Section 4 carefully lays out the process of developing and verifying the proposal from data preparation all the way through performance evaluation. Section 5 provides insight into the performance of the proposed neural networks and Section 6 provides a summary, conclusion, and the topic's possible future works.

2. Power System State Estimation

The intuition behind state estimation (SE) for power systems is to solve for a system's state given a set of measurement data (making the system observable) while also considering possible data acquisition error. The basic formulation shown in Equation (1), and also shown in most state estimation works such as in [19–23], where z is the measurement data, h is a function that translates the true system states, x , to measurement values, and ϵ is the modeled acquisition error. For every state estimation problem, there exists required conditions so that the system state can be accurately and validly estimated. First is confirmation of network topology, achieved through confirmation and/or estimation of breaker statuses. Second is observability, which intuitively means, the state of all the nodes of the system can be solved given a set of state measurements in the form shown in Equation (1) [24].

$$z = h(x) + \epsilon \quad (1)$$

$$\min J = \sum_{i=1}^n \frac{(z_i - h_i(x))^2}{\sigma_i^2} \rightarrow [z - h(x)]^T \cdot W \cdot [z - h(x)] \quad (2)$$

Due to the strong dependency of this process on measured data, any estimate of the system state will only be as good as the measurement data it was fed [23,25]. Thus, the third requirement is bad data processing, which is why papers that deal with bad data itself [26], detection [27], and suppression [28] exist. Once data have been cleaned and prepared, all of this information is inserted into the weighted least squares (WLS) formulation as seen in Equation (2) with the aim of minimizing ϵ , thus called WLS-SE. σ_i represents the standard deviation of the measurements—or, loosely put, error—depending on the device and the quantity measured. Examples of this are $\sigma_i = 1\%$ to 3% for voltage-based measurements, 5% for current-based measurements, and 15% for power-based measurements. The general flow of WLS-SE is shown in Figure 1 which is closely similar to [19].

Figure 1. The general WLS-SE flowchart.

One of the more important relationships to be established is the relationship between the measurements and the different system states, i.e., the relationship of z_i with x_i , which is shown by $h(x)$. At this point, it is important to be clear about what parameters will

be the system state and which parameters will be measured. Based on traditional power system analysis methods, knowledge of system-wide bus voltages and angles are sufficient in order to fully establish current flows and phase differences. Thus, the system state, x_i , becomes a $2n \times 1$ vector consisting of voltage magnitude and angle quantities for all n buses; the measurements largely become a combination of either a subset of the bus voltage vector or a subset of the known real and reactive power injections. With that in mind, and depending on the measurement data used, $h(x)$ can take its forms from traditional power flow analysis, with power measurements derived from the system-wide power balance equation shown in Equation (3) and is related to the system state via nodal analysis as shown in Equation (4), as presented in textbooks such as [29].

$$P_{gen} = P_{load} + P_{loss} \quad (3)$$

$$Y_{bus} \cdot V = I \quad (4)$$

The solved state estimates will only be as precise as the model and data used, which is why papers that deal with parameter estimation also exist [30]. Other ways to improve the state estimate also include transforming the measurements into more indicative quantities such as branch current measurements [20]. The details about these steps have been neatly laid out by [21] from network topology modeling down to bad data processing. While this paper focuses on static state estimation, other forms also exist such as dynamic state estimation where [31] uses the Extended Kalman Filter to effectively consider unknown inputs of synchronous machines (generators). This requires complete modeling of all components of the power system including uncertain components such as RE sources. The authors of [11], one of whom also published the state estimation textbook [22], acknowledges that machine learning/deep learning-based state estimation is another direction that state estimation research can go to due to its ability to provide estimates without the need for power system models and not being vulnerable to assumptions that are not true for realistic cases. As [11] also mentions, deep learning-based state estimators can also be developed for dynamic state and parameter estimation. As will be mentioned in Section 3, the estimate is only true for the situations the network is trained in. This paper assumes a stance of static state estimation and is only trained for steady state measurements; therefore, should a fault or a slow acting stability issue occur, the *current trained model's* estimates will not be valid. However, should the proposed method be trained for dynamic state estimation—where generator rotor angles are also estimated—then these issues should be addressable as well.

3. Neural Network-Based State Estimation

The buzzwords big data and machine learning have been making the rounds in the academic community for years already. These fields of study go hand-in-hand in a multitude of proposals ranging from basic image classifiers, handwriting recognition, object detection, simple media recommendations, and even to self-driving cars, as popularly posited by [32]. Indeed, machine learning can be used to address various problems that can be solved by minimizing certain parameters of a given model iteratively. State estimation is by far one of those methods as seen in its formulation presented in Section 2. One of the more successful approaches available are neural networks, which have a plethora of parameters to tune and a good number of connected *neurons* to hold and remember certain information. A simple neural network is shown in Figure 2.

Figure 2. A neural network with two sets of two-unit fully connected hidden layers. The input and output layers are $[x_1, x_2]$ and $[y_1, y_2]$.

3.1. Neural Network Theory

On the surface, the output of a neural network can be treated as a very long linear equation as shown by Equation (5) for an N_L -layer, N_x -input, N_y -output fully connected neural network as shown in [33–35], where the j th neuron in the r th layer is the sum of the i weighted and connected neurons in the $(r - 1)$ layer and the r th layer's bias vector—all of which are subject to the activation function, h . An important thing to note with the activation function, h , is that it squeezes the numbers to digestible ranges with different behaviors depending on the choice. Usual choices of activation functions include sigmoid, hyperbolic tangent, and the rectified linear unit. In the case of dimensions, the 2nd layer's neurons take from the N_x weighted input vector, $\mathbf{x}$, and the N_L th layer gives the N_y output vector, $\mathbf{y}$. Equation (6) shows this relationship in matrix form with n_r and $n_{(r-1)}$ representing the number of neurons of the r th and $(r - 1)$ th layer and H representing the activation function applied to each row. This is the equation generally followed during testing and actual implementation, called the feed-forward path.

$$a_j^{(r)} = h \left(\sum_i W_{j,i}^{(r \leftarrow r-1)} \cdot a_i^{(r-1)} + b_j^{(r)} \right) \quad (5)$$

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n^{(r)} \end{bmatrix}^{(r)} = H \left(\begin{bmatrix} W_{1,1} & W_{1,2} & \dots & W_{1,n^{(r-1)}} \\ W_{2,1} & W_{2,2} & \dots & W_{2,n^{(r-1)}} \\ \vdots & \vdots & \ddots & \vdots \\ W_{n^{(r)},1} & W_{n^{(r)},2} & \dots & W_{n^{(r)},n^{(r-1)}} \end{bmatrix}^{(r \leftarrow r-1)} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n^{(r-1)} \end{bmatrix}^{(r-1)} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n^{(r)} \end{bmatrix}^{(r)} \right) \quad (6)$$

During the process of training, however, there is an additional step to reliably tune the weights to its proper values, called the back-propagation. Intuitively, this step takes the gradient between the error and the individual weights as shown in [33–35] and uses it to update the weight values in order to minimize the error. The loss function, which is the “modeled difference” between the predicted output, $\hat{y}_i$, and the ground truth, y_i , is at the forefront of this step. One of the most common loss functions is the mean square error (MSE), which is shown in Equation (7).

$$C = L(y_i, \hat{y}_i) \rightarrow C = \frac{1}{N} \sum_i^N (\hat{y}_i - y_i)^2 \quad (7)$$

The challenge is finding the gradient of the cost function with respect to the weights. Shown in the succeeding equations below is the derivation to achieve the desired form. To obtain $\frac{\partial C}{\partial W}$, remember from Equation (7) that C is dependent on $\hat{y}_i$ and y_i , and by chain, $\hat{y}_i$ is dependent on the sum of the weighted neuron values, seen in Equation (5) (when r is the last/output layer, $a_i^{(r)} = \hat{y}_i$). Performing the simplified differentiation and applying chain rule, we obtain Equation (8).

$$\frac{\partial C}{\partial W_{i,j}^{(r)}} = \frac{\partial C}{\partial a_i^{(r)}} \cdot \frac{a_i^{(r)}}{W_{i,j}^{(r)}} \quad (8)$$

$$a_i^{(r)} = h\left(\sum_j W_{i,j}^{(r)} \cdot a_j^{(r-1)} + b_j^{(r)}\right) \rightarrow \frac{\partial a_i^{(r)}}{\partial W_{i,j}^{(r)}} = a_j^{(r-1)} \quad (9)$$

Now, Equation (9) recognizes that the weighted neuron values, $a_i^{(r)}$, are dependent also on their own sum of weighted neuron values. Therefore, it will propagate back up to the first layer, as implied by Equation (10).

$$\frac{\partial C}{\partial W_{i,j}^{(r)}} = \frac{\partial C}{\partial a_i^{(r)}} \cdot a_j^{(r-1)} \quad (10)$$

$$W_{i,j}^{(r)*} = W_{i,j}^{(r)} - \alpha \frac{\partial C}{\partial a_i^{(r)}} \cdot a_j^{(r-1)} \quad (11)$$

Thus, we arrive at the final point which is Equation (11), where the change to the weight value, $W_{i,j}^{(r)}$, is a decrement by a fraction of the gradient dictated by the learning rate, α , as shown in [33–35]. The differentiation while including specific activation functions is left to the reader for personal exploration. Here, the activation function is simply a pass through, that is, $y = h(x) = x$.

As mentioned before, this paper proposes the use of machine learning, specifically model optimized neural networks, to solve the state estimation problem. Using neural networks eliminates the need to formulate and compute the measurement function, $h(x)$, the gain matrix, G , which also stems from the measurement Jacobian, H , and the covariance matrix, R . All of the information that is contained within those quantities can be assumed to be summarily modeled within the neural network's weight values, thus making neural networks a worthy candidate for a state estimator. In particular, the model will *grow into* $h(x)$, which is mathematically shown in Equation (12).

$$x = LSTM(z, \epsilon) \quad (12)$$

3.2. Implementation to State Estimation

This paper takes off from a previous work shown in [36], where the standard MSE loss function has been replaced with the WLS loss function, which is used traditionally by power system state estimation to help the neural network model learn which of the measurements to trust more. By coming off of Equation (2), the weight, W , is used with predetermined weights that correspond to the measurement devices known to the operator. The resulting equation is shown in Equation (14)

$$W = \frac{1}{\sigma} \quad (13)$$

$$\min J = \sum_{i=1}^n \frac{(z_i - h_i(x))^2}{\sigma_i^2} \rightarrow [z - h(x)]^T \cdot \frac{1}{\sigma} \cdot [z - h(x)] \quad (14)$$

Note that, in this paper's notation, the system states x_i are the neural network's target prediction, $\hat{y}_i$, and the system measurements z_i are the neural network's inputs, x_i . The paradigm this paper takes is inferring the system state given the known system states some time steps before as shown in Figure 3, i.e., approximating $h(x)$ and learning to consider or reject ϵ . Furthermore, reference [36] used a special kind of neural network structure to improve the accuracy of estimation. This is the long short-term memory (LSTM) neural network, which is an improved variation of the recurrent neural networks (RNNs). It works better for time series data by taking into account the measurements observed some few

time steps prior and introduces new weights that correspond to remembering, forgetting, and passing certain information from the input data. These weights are what sets it apart from standard fully connected feed-forward or multilayer perceptron (MLP) networks.

For this paper, the LSTM networks are sequentially stacked with three hidden LSTM layers and a fully connected output layer as shown in Figure 4. The network structure may vary depending on the scale of application. The activation function for the layers is the default linear (or pass-through) activation. This is possible because the measurements are already acquired using the per-unit system; thus, the numbers are already within manageable ranges and typical machine learning problems such as the vanishing and exploding gradient are prevented. Furthermore, the learning rate, α , is also set to be reasonably small, at 0.001. The loss functions, both MSE and WLS, are minimized using the adaptive moments optimization algorithm.

Figure 3. The visual representation of the LSTM-based state estimator. The historical sequence input of measurement values (circles) to the LSTM from $t - 4$ to t will be considered in estimating the unknown (X-marks) value at the current time step, t .

Figure 4. The structure of the LSTM-based state estimators. The AC subgrid neural network takes in 52 measurements for 5 time steps and returns 68 values, 2 for each bus corresponding to bus voltage magnitude and angle. The DC subgrid neural network takes in 12 measurements for the same time steps and returns 9 values, one for each bus voltage magnitude.

3.3. Neural Network Model Optimization

The key proposal of this paper is the use of model optimization techniques to reduce the model's size. Proposed as early as 1990 by [33], the concept of neural network pruning is removing weights that are not responsive to back-propagation as [33] mentions. As discussed in Section 3, back-propagation is in charge of updating the weight values given

the gradient of the error with the weights. If the responsiveness of the weights to back-propagation—or in this paper’s terms, the gradient—is negligible, then it is plausible to conclude that the weight (neural connection) is not needed. These days, not only the gradient is evaluated but the value itself is also checked, that is, if the weight is zero (or plays around zero), that connection is also removed. The work by [37] has only recently proposed a standard way to benchmark gains received from pruning. Popular machine learning frameworks such as TensorFlow and pyTorch have also only recently released their pruning libraries at nearly the same time as [37] was published.

There are a few general trends between pruning methods as [37] points out: Structure, Scoring, Scheduling, and Fine-tuning. Each of these differences can exploit advantages but have their own intrinsic disadvantages too. An example would be networks whose structure has been pruned, which is largely removing weight connections as shown in Figure 5. Since the parameters are exactly zero, deploying these models in small-scale devices such as micro-controllers and remote terminal units is largely plausible due to the advantage they obtain when their model files are being compressed. Compression algorithms work better when there are true zeros in the data, more so if the file is sparse in general. However, speed up features such as those in CUDA-enabled hardware may or may not be available to these models as the APIs themselves may or may not be tailored to handle random sparsity. Scheduling, on the other hand, prunes the network bit by bit at every step of the training process or immediately to target sparsity. An example of a pruning schedule is shown in Figure 6. It is also possible to be creative with the training process and start the network with all the weights active to help the network reach near optimum point before pruning. Once there, pruning can proceed as scheduled to save space and to focus on updating the responsive weights, which leads to this paper’s main proposal: the use of scheduled weight pruning to optimize the neural network-based power system state estimator.

Figure 5. Visual representation of neural network weight pruning. Each line segment connecting the neurons has a weight value. For weight pruning, these weight values are zeroed out, which is equivalent to removing the connection between the neurons.

The pruning schedule used is Polynomial Decay that starts from 0% sparsity at training step 0 and ends with the target (25%, 50%, 75%) sparsity. The rate at which the weights are pruned follows the default power setting, which is a cubic ramp-up function as stated in the TensorFlow-Model Optimization Toolkit documentation. It is important to remember here that the TensorFlow way of weight pruning does not immediately cull the weights, rather, a masking layer is created alongside the neural network. The mask layer is Boolean, 1 for an active weight and 0 for an inactive (or pruned) weight. Thus, it is important at the end of the process to strip the mask layer using the `strip` pruning function such that when the model is saved, the active and inactive weights are committed.

Figure 6. An example of weight matrix sparsity given a pruning schedule—weights are removed at each pruning step until the sparsity reaches the specified targets. Training frameworks such as TensorFlow provide the option to set the initial and target sparsity and end step.

4. Methodology

To properly evaluate the proposed state estimator, measurement data must first be prepared together with the test system model. Moreover, the conventional WLS-SE should serve as the control method or the score to beat. Further, the proposed neural networks should be trained separately without using transfer learning to ensure learning capability. Finally, the WLS-SE's input measurement data should also be the inputs to the other state estimators to ensure input–output consistency.

4.1. System Model and Data Preparation

To capture the most realistic system behavior possible in the absence of actual full historical AMI data, it is assumed that the loads follow an actual day load profile as shown in Figure 7. It is also assumed that there is only one type of load in this particular microgrid system. The day load profile is sampled every five minutes and is used to scale the loads of the test system appropriately. The final system state to use for state estimation is recorded once the system reaches steady state after applying the load change which takes about five seconds in simulation time (not real time). These metered values are then subject to data augmentation.

Figure 7. The day load profile to be followed by all loads in the system.

The test system is a modified IEEE 34-bus test system from [38] containing a 9-bus DC microgrid model developed by [36] connected together via a bidirectional interlinking converter as shown in Figure 8. The IEEE 34-bus test system is a grid-connected distribution system composed of all kinds of loads by phase connection and by bus distribution with a maximum rating of about 150 kW per phase. It also has under load tap changers and

a 24.9 kV/4.16 kV transformer. The interlinking converter employs the droop control strategy to manage the power sharing between the AC and DC subgrids. The DC subgrid contains 20 kW loads at the far ends of the network. There are also photovoltaic (PV) power sources coupled with battery energy storage systems (BESS) which are all capable of supporting the DC subgrid in islanded mode. The test system is realized in the PSCAD simulation software.

Figure 8. The test system to perform state estimation on. IEEE 34-bus test system to serve as the AC subgrid with the IEEE 9-bus test system at the far end of the IEEE 34-bus network to serve as the DC subgrid.

To reconcile the models to use between the WLS-SE and the PSCAD, certain assumptions were made. Since the power sharing between the subgrids are outside the scope of this paper’s proposal, it is assumed that the power flow within the converter is unidirectional towards the DC subgrid only. Thus, the loading conditions and renewable energy source operating points are modified such that power is not transmitted from the DC subgrid back up to the external system (IEEE 34-bus’ slack bus). Second, the three-phase unbalanced modeling of the AC subgrid’s lines are replaced with its positive sequence equivalent, Z_1 . The impedance values are calculated via Equation (15), with the A transformation matrix shown in Equation (16). The Z_{abc} model of all the lines are neatly presented in the documentation of the IEEE 34-bus test system [38].

$$Z_{012} = A^{-1} \cdot Z_{abc} \cdot A \quad (15)$$

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1\angle-120^\circ & 1\angle120^\circ \\ 1 & 1\angle120^\circ & 1\angle-120^\circ \end{bmatrix} \quad (16)$$

4.2. Data Augmentation

The measurement data are composed of real and reactive power measurements from the load buses, voltage measurements from choice buses following optimal measurement device placement results presented by [39], and some power flow measurements for the AC subgrid and current measurements for the DC subgrid. It is also assumed to be transmitted through a noisy channel that has additive Gaussian noise. The mean and variance that the noise introduces is influenced by the choice of measurement devices as specified in Section 2. To populate the dataset, random values are acquired that follow the normal distribution of the measurements with sample mean, $\mu = 0$, and standard deviation, $\sigma = \{1\%, 3\%, 15\%\}$. In MATLAB, this is realized by the `normrnd()` function. To ensure that the distribution is well represented, `normrnd()` is arbitrarily set to generate $n_a = 1000$ samples. Therefore, for each measurement z_t at time t , the noise, ϵ_i , is a random sample generated by `normrnd()`

following the ziggurat rejection algorithm proposed by [40]. The augmented data are mathematically shown in Equation (17).

$$Z = \begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_T \end{bmatrix} \rightarrow \forall z_t \in Z, z_t = z_t + \begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \vdots \\ \epsilon_{n_d} \end{bmatrix} \quad (17)$$

The next step is preparing the data for neural network training. For supervised neural network training, the data are fed in the format of (inputs, targets), or mathematically, (x, y) . Recall that the reason why data are augmented, aside from lack of actual data, is due to the modeled uncertainty from the measurement device of choice. So, when forming the input–target pairs, the inputs encompass the whole augmented dataset, while the targets come only from the pseudo-measurement set. Mathematically, this is shown in Equation (18).

$$\begin{bmatrix} z_t \rightarrow y_t \\ z_t + \epsilon_0 \rightarrow y_t \\ \vdots \\ z_t + \epsilon_{n_d} \rightarrow y_t \end{bmatrix} \quad (18)$$

This tells the neural network that the measurements will be noisy; thus, the network must learn to account for the specified uncertainties. The data are also transformed such that the training process is regularized: a process to prevent the neural network from overfitting into the small dataset which results in generalization failures, especially when deployed.

4.3. Training

For this paper, there are six (6) state estimators to pit against each other:

- Weighted Least Squares-State Estimation;
- Standard LSTM;
- Loss Modified LSTM;
- 25% Pruned LSTM;
- 50% Pruned LSTM;
- 75% Pruned LSTM.

To ensure consistency of structure among comparisons, the Modified and Pruned LSTMs followed the Standard LSTM's structure and were trained separately from zero, that is, transfer learning between models is not used. All networks are trained over about 13,000 data points corresponding to two hours' worth of measurement values. Half of the data are used for training, the other half are hidden and used for validation. It is important that the training process does not see the validation set so that the aim of generalization is reliably tested. A third hour of the dataset, hidden from the training process, is used for final evaluation which is shown in Section 5. The optimizer used is the adaptive moments algorithm with a learning rate of 0.0001 to aid in preventing explosive and vanishing gradients during the training process. The loss function used is the weighted least squares for the loss modified LSTM and standard mean square error for the rest. The training session repeats for 30 epochs, with batch sizes of 9 for the DC subgrid and 68 for the AC subgrid due to limitations in implementing the WLS loss function in the TensorFlow framework. The length of the training session should be kept minimum so as not to overfit into the dataset. The models are evaluated for learning fitness depending on the ratio between the final training and validation loss values. Ideally, these values should be as close to each other as possible. These configurations are kept the same everywhere for consistency.

4.4. Testing

The testing dataset is composed of the augmented data points at the third hour, which is at a slightly higher loading point than the training dataset. This is to also be able to test the amount of generalization the LSTM networks have learned. Since the WLS-SE assumes that the system state variables are the voltage magnitudes and angles, the neural network outputs are single columns in order of increasing bus number, with the magnitudes in the first half and the angles at the second half. Voltage magnitudes are measured in per-unit while the angles are normalized to 90° . The error between them is measured with the mean absolute error (MAE) and the root mean square error (RMSE) metrics. These metrics are shown in Equations (19) and (20), respectively.

$$C = \frac{1}{N} \sum_i^N (|\hat{y}_i - y_i|) \quad (19)$$

$$C = \sqrt{\frac{1}{N} \sum_i^N (\hat{y}_i - y_i)^2} \quad (20)$$

5. Results and Discussion

Shown in Tables 1 and 2 are the errors in state estimate per state estimator and subgrid. As can be observed, the LSTM networks performed better than the conventional WLS-SE, largely due to the learning capability of neural networks and to the relative simplicity of the network and data. Another important note is that the error metrics increase steadily as the target sparsity increases, all while the estimation time decreases. This is largely due to the fact that there are less weights to hold information and less weights to actually multiply. It is also important to note that the speed up for each method may or may not be accurately presented because this is only tested for one PC unit equipped with an Intel Core i5, 8GB RAM, and a hard disk drive. However, it can be inferred that with better equipment, such as a GPU with support for Nvidia's CUDA cores which are optimized for neural network operations, the estimation time will decrease. This event will be more pronounced when the methodology is subject to large, practical system configurations, that is, networks with hundreds or thousands of buses, tens or hundreds of measuring devices, and of course, millions of Y_{bus} entries.

Table 1. Summary of errors in state estimates for each AC subgrid state estimator.

State Estimator	Voltage Magnitude (MAE)	Voltage Angle (MAE)	Voltage Magnitude (RMSE)	Voltage Angle (RMSE)	Estimation Time (s)
Conventional WLS	0.0128	0.1691	0.0151	0.2498	3.5828
Normal LSTM	0.0025	−0.0005	0.0032	0.0007	0.2834
Modified LSTM	0.0013	−0.0002	0.0019	0.0005	0.2805
25%-Pruned LSTM	0.0053	−0.0005	0.0055	0.0009	0.2764
50%-Pruned LSTM	−0.0061	−0.0005	0.0063	0.0011	0.2734
75%-Pruned LSTM	−0.0086	−0.0006	0.0087	0.0012	0.2701

Table 2. Summary of errors in state estimates for each DC subgrid state estimator.

State Estimator	Voltage Magnitude (MAE)	Voltage Magnitude (RMSE)	Estimation Time (s)
Conventional WLS	0.0035	0.0036	0.9689
Normal LSTM	−0.0005	0.0005	0.3043
Modified LSTM	0.0002	0.0003	0.2989
25%-Pruned LSTM	0.0008	0.0008	0.2863
50%-Pruned LSTM	0.0013	0.0015	0.2814
75%-Pruned LSTM	−0.0017	0.0020	0.2797

Shown in Figures 9–17 are the graphical representations of the state estimation errors against the true system states. This is where we can see individually how the methods perform and at which points they could possibly fail. One possible trend that can be inferred is that non-pruned networks tend to under predict, which is not the case. The under and over prediction of the model is highly dependent on the sequence of data it has seen during training. Recall that the data are always shuffled every training step in a session and that the weights that will be pruned are nearly random and are at the mercy of the scheduler and its gradients. However, this does not mean that there is absolutely no definite and concrete answer. One of the possible reasons for this is the modeled (or anticipated) errors in the measurement units, which is why it tends to under (or over) predict. What is certain, however, is that a neural network’s aim is always to generalize the dataset; thus, the testing methodology exhausts the dataset. Therefore, by consequence, the error metrics are good enough approximations of their performances. The file sizes of the neural networks after compression are 837 kB for the normal and modified LSTM, 380 kB for the 25% pruned LSTM, 375 kB for the 50% pruned LSTM, and 370 kB for the 75% pruned LSTM. The reasons behind the compression are outside the scope of this paper, but loosely put, it is due to the compression algorithms and its fundamental limits and tradeoffs, i.e., processing time, memory, chunk size, etc. [41].

Figure 9. Conventional weighted least square state estimation results: (a) AC voltage magnitudes and (b) AC voltage angles. y -axis corresponds to estimated bus voltage in (p.u.), while the x -axis corresponds to the bus number.

Figure 10. Standard LSTM results: (a) AC voltage magnitudes and (b) AC voltage angles. y -axis corresponds to estimated bus voltage in (p.u.), while the x -axis corresponds to the bus number.

Figure 11. Modified LSTM results: (a) AC voltage magnitudes and (b) AC voltage angles. y -axis corresponds to estimated bus voltage in (p.u.), while the x -axis corresponds to the bus number.

Figure 12. The 25% Pruned LSTM results: (a) AC voltage magnitudes and (b) AC voltage angles. y -axis corresponds to estimated bus voltage in (p.u.), while the x -axis corresponds to the bus number.

Figure 13. The 50% Pruned LSTM results: (a) AC voltage magnitudes and (b) AC voltage angles. y -axis corresponds to estimated bus voltage in (p.u.), while the x -axis corresponds to the bus number.

Figure 14. The 75% Pruned LSTM results: (a) AC voltage magnitudes and (b) AC voltage angles. y -axis corresponds to estimated bus voltage in (p.u.), while the x -axis corresponds to the bus number.

Figure 15. (a) Weighted least square state estimation DC voltage magnitudes and (b) standard LSTM DC voltage magnitudes; y -axis corresponds to estimated bus voltage in (p.u.), while the x -axis corresponds to the bus number.

Figure 16. (a) Modified LSTM DC voltage magnitudes and (b) 25% Pruned LSTM DC voltage magnitudes; y-axis corresponds to estimated bus voltage in (p.u.), while the x-axis corresponds to the bus number.

Figure 17. (a) 50% Pruned LSTM DC voltage magnitudes and (b) 75% Pruned LSTM DC voltage magnitudes; y-axis corresponds to estimated bus voltage in (p.u.), while the x-axis corresponds to the bus number.

Finally, one of the telling signs of the graphs is the profile of the system state. The x-axis represents the bus numbers of the system. It is here we can see that there are two big dipping points in the voltage and angle profiles, both of which are interfaces between an under load tap changer (ULTC) and a three-phase transformer, respectively. Recall that the true system state is acquired from the PSCAD simulations; thus, we can be confident in the models, that is, the behavior is consistent with the documentation for the IEEE 34-bus test system. An example of this sign is that, in between the big voltage drops, we see long line segments which introduce relatively large impedances. Bulk of the spot loads are located at the far ends of the network; thus, the required current flow forces the voltage difference. It is also important to note that the sharp dip of angle measurement is not what it seems to be—mind the scale of the y-axis (0° to -4°).

6. Conclusions

Using historical data, several configurations of model optimized neural networks have been developed, trained, and tested against their standard implementations and the conventional state estimation method. As a consequence of the standardized model optimization process, i.e., scheduled weight pruning, the neural network's size and potency is reduced. In spite of this, the neural network still provides satisfactory state estimation results and is more accurate over the conventional WLS-SE. The errors in estimation range from 0.0003 to 0.0087 per-unit for the hybrid AC/DC distribution system composed of the IEEE 34-bus test system and the 9-bus DC microgrid. Furthermore, with the neural

network's high scalability, this proposal is good for modeling and estimating practical hybrid distribution systems.

Author Contributions: Conceptualization, H.S., D.K. and J.M.D.; methodology, D.K. and J.M.D.; software, J.M.D.; validation, D.K. and J.M.D.; formal analysis, D.K.; investigation, D.K.; resources, J.M.D.; data curation, D.K.; writing—original draft preparation, D.K. and J.M.D.; writing—review and editing, H.S. and J.M.D.; visualization, D.K.; supervision, H.S.; project administration, H.S.; funding acquisition, H.S. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by the Advanced Research Project funded by the SeoulTech (Seoul National University of Science and Technology).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank Seoul National University of Science and Technology for providing the research funding needed to pursue this topic.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SCADA	Supervisory Control and Data Acquisition
AMI	Advanced Metering Infrastructure
PMU	Phasor/Phase Measurement Unit
SE	State Estimation
WLS	Weighted Least Squares
WLS-SE	Weighted Least Squares State Estimation
BESS	Battery Energy Storage System
CUDA	Nvidia Compute Unified Device Architecture
API	Application Program Interface
MAE	Mean Absolute Error
MSE	Mean Square Error
RMSE	Root Mean Square Error
MLP	Multilayer Perceptron
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory Neural Network
RAM	Random Access Memory
ULTC	Under Load Tap Changer

References

1. Billinton, R.; Allan, R. Power-System Reliability in Perspective. *Electron. Power* **1984**, *30*, 231. [[CrossRef](#)]
2. Ahmed, R.; Sreeram, V.; Mishra, Y.; Arif, M. A Review and Evaluation of the State-of-the-Art in PV Solar Power Forecasting: Techniques and Optimization. *Renew. Sustain. Energy Rev.* **2020**, *124*, 109792. [[CrossRef](#)]
3. Ghandehari, R.; Shoulaie, A. Evaluating Voltage Notch Problems Arising from AC/DC Converter Operation. *IEEE Trans. Power Electron.* **2009**, *24*, 2111–2119. [[CrossRef](#)]
4. Liu, X.; Wang, P.; Loh, P.C. A Hybrid AC/DC Microgrid and Its Coordination Control. *IEEE Trans. Smart Grid* **2011**, *2*, 278–286. [[CrossRef](#)]
5. Bevrani, H.; Watanabe, M.; Mitani, Y. *Power System Monitoring and Control*; John Wiley & Sons: Hoboken, NJ, USA, 2014.
6. Aminifar, F.; Fotuhi-Firuzabad, M.; Safdarian, A. Optimal PMU Placement Based on Probabilistic Cost/Benefit Analysis. *IEEE Trans. Power Syst.* **2013**, *28*, 566–567. [[CrossRef](#)]
7. Saha Roy, B.; Sinha, A.; Pradhan, A. An Optimal PMU Placement Technique for Power System Observability. *Int. J. Electr. Power Energy Syst.* **2012**, *42*, 71–77. [[CrossRef](#)]
8. Gou, B. Generalized Integer Linear Programming Formulation for Optimal PMU Placement. *IEEE Trans. Power Syst.* **2008**, *23*, 1099–1104. [[CrossRef](#)]
9. Xia, N.; Gooi, H.B.; Chen, S.; Hu, W. Decentralized State Estimation for Hybrid AC/DC Microgrids. *IEEE Syst. J.* **2018**, *12*, 434–443. [[CrossRef](#)]

10. Kong, X.; Yan, Z.; Guo, R.; Xu, X.; Fang, C. Three-Stage Distributed State Estimation for AC-DC Hybrid Distribution Network Under Mixed Measurement Environment. *IEEE Access* **2018**, *6*, 39027–39036. [\[CrossRef\]](#)
11. Zhao, J.; Qi, J.; Huang, Z.; Meliopoulos, A.P.S.; Gomez-Exposito, A.; Netto, M.; Mili, L.; Abur, A.; Terzija, V.; Kamwa, I.; et al. Power System Dynamic State Estimation: Motivations, Definitions, Methodologies, and Future Work. *IEEE Trans. Power Syst.* **2019**, *34*, 3188–3198. [\[CrossRef\]](#)
12. Manitsas, E.; Singh, R.; Pal, B.C.; Strbac, G. Distribution System State Estimation Using an Artificial Neural Network Approach for Pseudo Measurement Modeling. *IEEE Trans. Power Syst.* **2012**, *27*, 1888–1896. [\[CrossRef\]](#)
13. Salehfar, H.; Zhao, R. A Neural Network Preestimation Filter for Bad-Data Detection and Identification in Power System State Estimation. *Electric Power Syst. Res.* **1995**, *34*, 127–134. [\[CrossRef\]](#)
14. Vinod Kumar, D.; Srivastava, S.; Shah, S.; Mathur, S. Topology Processing and Static State Estimation Using Artificial Neural Networks. *IEE Proc. Gener. Transm. Distrib.* **1996**, *143*, 99. [\[CrossRef\]](#)
15. Mestav, K.R.; Luengo-Rozas, J.; Tong, L. Bayesian State Estimation for Unobservable Distribution Systems via Deep Learning. *IEEE Trans. Power Syst.* **2019**, *34*, 4910–4920. [\[CrossRef\]](#)
16. Wang, L.; Zhou, Q.; Jin, S. Physics-Guided Deep Learning for Power System State Estimation. *J. Mod. Power Syst. Clean Energy* **2020**, *8*, 607–615. [\[CrossRef\]](#)
17. Zamzam, A.S.; Sidiropoulos, N.D. Physics-Aware Neural Networks for Distribution System State Estimation. *IEEE Trans. Power Syst.* **2020**, *35*, 4347–4356. [\[CrossRef\]](#)
18. Zhang, L.; Wang, G.; Giannakis, G.B. Real-Time Power System State Estimation and Forecasting via Deep Unrolled Neural Networks. *IEEE Trans. Signal Process.* **2019**, *67*, 4069–4077. [\[CrossRef\]](#)
19. Meriem, M.; Bouchra, C.; Abdelaziz, B.; Jamal, S.O.B.; Faissal, E.M.; Nazha, C. Study of State Estimation Using Weighted-Least-Squares Method (WLS). In Proceedings of the 2016 International Conference on Electrical Sciences and Technologies in Maghreb (CISTEM), Marrakech, Morocco, 26–28 October 2016; pp. 1–5. [\[CrossRef\]](#)
20. Baran, M.; Kelley, A. A Branch-Current-Based State Estimation Method for Distribution Systems. *IEEE Trans. Power Syst.* **1995**, *10*, 483–491. [\[CrossRef\]](#)
21. Monticelli, A. Electric Power System State Estimation. *Proc. IEEE* **2000**, *88*, 262–282. [\[CrossRef\]](#)
22. Abur, A.; Exposito, A.G. *Power System State Estimation: Theory and Implementation*; Marcel Dekker, Inc.: New York, NY, USA, 2004.
23. Wang, H.; Schulz, N. A Revised Branch Current-Based Distribution System State Estimation Algorithm and Meter Placement Impact. *IEEE Trans. Power Syst.* **2004**, *19*, 207–213. [\[CrossRef\]](#)
24. Clanents, K.; Krutnpholz, G.; Davis, P. Power System State Estimation with Measurement Deficiency: An Observability/Measurement Placement Algorithm. *IEEE Trans. Power Appar. Syst.* **1983**, *PAS-102*, 2012–2020. [\[CrossRef\]](#)
25. Muscas, C.; Pau, M.; Pegoraro, P.A.; Sulis, S. Effects of Measurements and Pseudomeasurements Correlation in Distribution System State Estimation. *IEEE Trans. Instrum. Meas.* **2014**, *63*, 2813–2823. [\[CrossRef\]](#)
26. Monticelli, A.; Garcia, A. Reliable Bad Data Processing for Real-Time State Estimation. *IEEE Trans. Power Appar. Syst.* **1983**, *PAS-102*, 1126–1139. [\[CrossRef\]](#)
27. Chen, J.; Abur, A. Placement of PMUs to Enable Bad Data Detection in State Estimation. *IEEE Trans. Power Syst.* **2006**, *21*, 1608–1615. [\[CrossRef\]](#)
28. Merrill, H.; Schweppe, F. Bad Data Suppression in Power System Static State Estimation. *IEEE Trans. Power Appar. Syst.* **1971**, *PAS-90*, 2718–2725. [\[CrossRef\]](#)
29. Saadat, H. *Power System Analysis*; McGraw-Hill: Singapore, 2004.
30. Zarco, P.; Exposito, A. Power System Parameter Estimation: A Survey. *IEEE Trans. Power Syst.* **2000**, *15*, 216–222. [\[CrossRef\]](#)
31. Ghahremani, E.; Kamwa, I. Dynamic State Estimation in Power System by Applying the Extended Kalman Filter With Unknown Inputs to Phasor Measurements. *IEEE Trans. Power Syst.* **2011**, *26*, 2556–2566. [\[CrossRef\]](#)
32. Bojarski, M.; Del Testa, D.; Dworakowski, D.; Firner, B.; Flepp, B.; Goyal, P.; Jackel, L.D.; Monfort, M.; Muller, U.; Zhang, J.; et al. End to End Learning for Self-Driving Cars. *arXiv* **2016**, arXiv:1604.07316.
33. Karnin, E. A Simple Procedure for Pruning Back-Propagation Trained Neural Networks. *IEEE Trans. Neural Netw.* **1990**, *1*, 239–242. [\[CrossRef\]](#) [\[PubMed\]](#)
34. Hecht-Nielsen, R. Theory of the Backpropagation Neural Network**Based on “Nonindent” by Robert Hecht-Nielsen. Which Appeared in Proceedings of the International Joint Conference on Neural Networks 1, 593–611, June 1989. © 1989 IEEE. In *Neural Networks for Perception*; Elsevier: Washington, DC, USA, 1992; pp. 65–93. [\[CrossRef\]](#)
35. Werbos, P. Backpropagation through Time: What It Does and How to Do It. *Proc. IEEE* **1990**, *78*, 1550–1560. [\[CrossRef\]](#)
36. Adi, F.S.; Lee, Y.J.; Song, H. State Estimation for DC Microgrids Using Modified Long Short-Term Memory Networks. *Appl. Sci.* **2020**, *10*, 3028. [\[CrossRef\]](#)
37. Blalock, D.; Ortiz, J.J.G.; Frankle, J.; Gutttag, J. What Is the State of Neural Network Pruning? *arXiv* **2020**, arXiv:2003.03033.
38. Kersting, W. Radial Distribution Test Feeders. In Proceedings of the 2001 IEEE Power Engineering Society Winter Meeting, Conference Proceedings (Cat. No.01CH37194), Columbus, OH, USA, 28 January–1 February 2001; Volume 2, pp. 908–912. [\[CrossRef\]](#)
39. Manousakis, N.M.; Korres, G.N. A Weighted Least Squares Algorithm for Optimal PMU Placement. *IEEE Trans. Power Syst.* **2013**, *28*, 3499–3500. [\[CrossRef\]](#)

-
40. Marsaglia, G.; Tsang, W.W. The Ziggurat Method for Generating Random Variables. *J. Stat. Softw.* **2000**, *5*, 1–7. [[CrossRef](#)]
 41. Mittal, S.; Vetter, J.S. A Survey of Architectural Approaches for Data Compression in Cache and Main Memory Systems. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 1524–1536. [[CrossRef](#)]