Application of Recurrent Graph Convolutional Networks to the Neural State Estimation Problem

Alexander Berezin[1,*], Stephan Balduin[1], Thomas Oberließen[2], Eric Veith[1], Sebastian Peter[2], and Sebastian Lehnhoff[1]
[1] R&D Division Energy - OFFIS e.V., Escherweg 2, 26121, Germany; Email: stephan.balduin@offis.de (S.B.); eric.veith@offis.de (E.V.); sebastian.lehnhoff@offis.de (S.L.)
[2] ie3 - TU Dortmund, Emil-Figge-Straße 70, 44227, Germany; Email: thomas.oberliessen@tu-dortmund.de (T.O.); sebastian.peter@tu-dortmund.de (S.P.)

***Abstract*—Neural State Estimation (NSE) is a novel application of deep learning which is concerned with interpolating the state of a distribution power grid from a limited amount of sensor data and can be represented as a non-linear graph time-series nowcasting problem. Although several authors have proposed their solutions for NSE, there is neither a comparison of approaches nor an industry-standard state of the art model yet. The main purpose of this paper is to compare these solutions to a promising new approach: recurrent graph convolutional neural networks. There are theoretical reasons to assume that this class of models is suited for solving NSE. Our experiments verify that they achieve similar performance while also presenting many unique advantages compared to the previously proposed models.**

***Index Terms*—Graph convolutional networks, neural state estimation, power system state estimation**

I. Introduction

State Estimation (SE) is the task of relating grid usage observations to the current grid state. For many years, SE was mainly performed for the transmission grids based on real and pseudo-measurement data. While the transmission grid is very extensively equipped with sensor technology, this is not the case for the distribution grids. Those sensors were not necessary to operate a distribution grid for a long time, but this has changed due to the increasing number of complex consumers. A grid operation closer to the grid's design limit caused by the increasing complexity of the distribution grid results in the need for efficient grid expansion and operational detection of power peaks. SE could provide the necessary grid transparency for the distribution grids, but some issues make it hard or impossible to do so. The required measurement data is missing since the distribution grids are much less permeated with sensor technology. Operators have to approximate load and generation with default load profiles, which do not describe the actual behavior of a given system's participants. More sensor technology would be needed, which is economically not feasible since distribution grids have many more nodes and lines to cover [1]. Furthermore, they are much more sensitive to changes in impedance than transmission grids. This characteristic invalidates some simplifications usually made for transmission grids [2], like assuming a near-DC power flow. Due to the X/R ratio in distribution grids, this assumption does not hold for them. To achieve grid transparency despite these issues, an intelligent approach is required [3, 4].

Artificial neural networks (ANNs) gained popularity in many different fields because of their ability to function as universal approximators for every Borel-measurable function, or even arbitrary dynamic systems in the case of Recurrent neural network (RNNs) [5–7]. Furthermore, ANNs have proven to be faster than iterative solutions for the PF calculation in the distribution grid and, depending on the quality of the data available for training, can achieve even higher solution quality [8]. Neural State Estimation (NSE), i.e. using ANNs to approximate the state of a given grid, was successfully applied in the past [9] and is the subject of active research indicated by an increasing number of publications [10, 11]. Generally, the approaches of these publications can be categorized into two types. The first approach is grid-agnostic neural networks, e. g., feed-forward ANNs [12] or autoencoders [13]. As the name implies, those approaches do not require any information about the grids' topology or parameters. However, that information can be exploited to reduce the complexity of the training procedure, which leads to the second category: the grid-aware neural networks [14], also referred to as physics-aware, physics-informed, or physics-based models.

To the best of our knowledge, there has not been a direct comparison of NSE approaches yet. Therefore, the contribution of our work is to fill this gap and introduce a new type of model for the given problem. The rest of this paper is structured as follows. In Section II we review state-of-the-art NSE approaches from the literature. In Section III we describe the considerations that lead to the new type of models we propose for NSE. In Section IV we describe the experiment setup used in order to compare selected NSE approaches, and we present the results from those experiments in Section V. We discuss the results and outline possible directions for future research in the conclusion.

Manuscript received September 1, 2022; revised October 15, 2022; accepted November 5, 2022.
*Corresponding author: Alexander Berezin (email: aleksandr.berezin @offis.de).

II. Related Work

NSE is a relatively new research field, with the first notable works published in 2014. Reference [13] used autoencoders to estimate the state of distribution grids. They used a Particle Swarm Optimization to reconstruct values for missing signals, which are, in this context, voltage magnitudes and phases. Later publications used simple feed-forward ANNs to directly map the available inputs (mostly complex power values of load and generation) to the desired outputs [8, 15]. Another of those early approaches used ANNs to estimate inputs for the traditional SE methods like Gauss-Newton [12].

More recent publications seem to prefer grid-aware over grid-agnostic approaches. Those have some advantages, e. g., they require less tuning effort and may yield better performance. One, if not the first, of these grid-aware models, is the physics-aware neural network (PAWNN), proposed by [14]. The idea is to use the classic feed-forward neuron as a building block but prune its synapses according to the graph's adjacency matrix, i.e., the grids' topology. These neurons are stacked in a variable number of layers, which is equal to the maximum diameter of a vertex-cut partition of the original graph (D). The algorithm for calculating is explained in their paper.

An improved derivation of this model was proposed in [16]. Designing the ANN architecture based on the adjacency matrix, as it was done for the original PAWNN, may lead to unnecessary connections between layers. The pruned physics-aware neural network (P2N2) cut out those unnecessary connections and used separate weight matrices for the individual parts of the ANN, depending on the grid topology.

Another approach is the prox-linear neural network model, proposed in [11], which is based on a prox-linear solver for SE using the least absolute value (LAV) method. The main idea is to split the nonlinear problem of SE into several blocks that are proximal linear. The prox-linear neural network is built by unfolding these blocks and can achieve significant performance improvements. Additionally, the authors developed a deep RNN for power system state forecasting (PSSF) since such networks are capable of learning temporal correlations in (historical) data.

Finally, while this paper was being finalized, we discovered another study that tested a similar hypothesis [17] of applying recurrent graph convolutional neural networks (RGCNNs) to NSE. However, we believe that our research is still valuable, as it was conducted independently and tests many more models.

III. Recurrent Graph Convolutional Neural Networks

Now let us abstract away from the existing solutions and think about the characteristics required from an ANN model to tackle the NSE problem efficiently. For that, we need to start with the properties of datasets commonly used for SE. These datasets are, fundamentally, graphs with time series associated with every node. From that, we can assume that a desirable model has to be geometric in the sense that it can leverage the structure of the graph and also recurrent to capture temporal correlations in the time series. We also expect it to benefit from being convolutional because the graph convolution operation is a good first-order approximation of the current flow in the power grid [18]. From these considerations, we can reasonably expect the models residing at the intersection of these classes (depicted in Fig. 1) to be most suited for NSE.

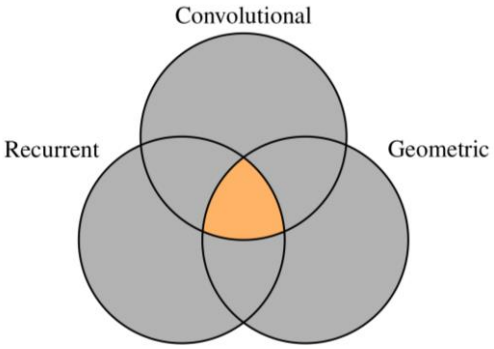

Fig. 1. Model selection rationale.

This class of models is called RGCNNs, originally proposed in [19], and it is represented by a significant number of different models in the literature. However, we have not been able to find a comprehensive review.

RGCNNs belong to a more general class of graph convolutional networks (GCNs), first proposed in [20], which can operate naturally on graph-structured data. By extracting and utilizing features from the underlying graph, GCNs can make more informed predictions about entities in these interactions, as compared to models that consider individual entities in isolation. This is achieved by employing a special convolution [21] on every node of the graph.

Since the convolution operation is local (only affects the adjacent nodes), the number of trainable parameters in GCNs is comparatively small. It also means that the model does not store information about the graph's structure but only about correlations between adjacent nodes. This makes GCNs ideal for transfer learning applications since a trained model can be transferred to a new graph and retain its predictive ability, provided that the interactions between nodes follow the same patterns. This ability is important for practical implementations of NSE, as discussed in [4].

RGCNNs combine the convolutional units from GCNs with memory units from RNNs in different ways in order to capture temporal correlations in the data. These models have been successfully applied to several practical problems, such as predicting traffic [22] and infection spread [23].

IV. Experiment Setup

The question we want to answer with this experiment is whether generic RGCNNs can compete in solving the NSE problem with models developed explicitly for this task.

A. Data Set for Neural State Estimation

To generate reproducible results, we chose open data sources and simulated data as the basis of our model comparison. Using synthetic data circumvents the issue of incomplete data sets due to a lack of sensor technology and data privacy restrictions. The grid and training data we used are based upon a SimBench benchmark model (available under ODbL and DbCL licenses). SimBench provides grid models that resemble real-life grids, as well as typical nodal load and generation time series [24].

The specific grid chosen is the 1-MV-urban–1-sw, a 147-node, 10 kV medium voltage grid. Using the provided load and generation time series, we calculated the resulting grid state with the agent based power system simulation SIMONA [25, 26]. The resulting data relevant to the model comparison comprises complex nodal voltages for all nodes with a temporal resolution of 15 minutes. In total, there are 35135 timesteps.

For this experiment, the dataset is split 50:50 into training and validation data along the time axis, meaning that the models are trained on the first half of the time steps and have to forecast the second half, so no data points are repeated in the process.

By design, RGCNNs have a problem with SE: they require input data (labels) for all nodes of the graph, while the very nature of the task implies that input data is only available for the observable nodes. A possible solution for this issue is found in [20]. The unobservable nodes can be initialized with Gaussian white noise sampled from ground truth distribution. The output data (targets) is the ground truth itself. Specifically, we assume 30% grid observability, i.e., 30% of nodes in the input data are initialized with the ground truth, and the rest are randomized.

B. Metric

In order to perform a statistical comparison of models, a numerical metric for said comparison has to be agreed upon first. Of course, such a metric would necessarily have to reduce the complexity of the task and thus would miss important details of the result. We will attempt to partially compensate for this in the next section by plotting each model's per-node error.

We have chosen to use the metric that is most commonly used in deep learning for evaluating the performance in regression problems, which is Mean Squared Error (MSE). It is also used as the loss function for all models except the prox-linear Network.

C. Baseline Model

The best baseline for comparing SE models would be the conventional method of solving this problem, namely the weighted least squares algorithm. However, that algorithm has a high number of manually tuned parameters, and we could not obtain a representative open-source implementation. Instead, we use a simple RNN, passing graph nodes as features in order to capture temporal correlations, which neither the physics-aware neural network nor the prox-linear neural network does.

D. Physics-Aware Neural Network

We did not find a reference implementation for this model, proposed in [14], and implemented it ourselves using PyTorch. The implementation may not be entirely correct and thus may not accurately represent its' performance. Unfortunately, we were not able to replicate the P2N2 variant from [16] since the proposed algorithm is only explained in the paper on one trivial example, and it is not clear how to generalize that example to more complex data.

E. Prox-Linear Neural Network

A reference implementation is available on GitHub: https://github.com/LiangZhangUMN/PSSE-via-DNNs. We adapted the implementation to use PyTorch instead of TensorFlow but did not change anything else.

F. Recurrent Graph Convolutional Neural Networks

Fortunately, there exists a whole library dedicated to reference implementations of the most well-known RGCNN models called PyTorch Geometric Temporal (PyGT) [27].

We decided to only use models without hyperparameters from PyGT. There are two reasons for that. First, hyperparameter tuning significantly complicates a comparative study. Second, PyGT in its current implementation suffers from performance issues that we will discuss in more detail in Evaluation, making the tuning process far too slow. This left us with two models: EvolveGCN and Temporal GCN.

In both cases, we build the final model by stacking D RGCNN layers (like [14] did for their PAWNN model) to ensure that information from the observable nodes is propagated across the entire graph during the forward pass. The loss function used for training is MSE.

1) EvolveGCN

The Evolving Graph Convolutional Network (EvolveGCN) is a model proposed in [28]. It consists of a GCN, which is built for the graph structure, and a RNN, which tracks changes in the graph topology and regulates the GCN parameters in each step to reflect those changes. It has two slightly different variants, EvolveGCN-H and EvolveGCN-O, which differ in how the weight matrix is updated. The "-H" treats the weight matrix as a hidden state of the dynamical system, while the "-O" treats it as the system's output. The authors recommend the "-O" variant if the graph structure is more informative than node labels. We included both of them in the comparison.

2) Temporal GCN

The temporal graph convolutional network (TGCN) is a model originally proposed in [29] for traffic prediction. A GCN is used to capture spatial features and is combined with gated recurrent unit (GRUs) for temporal features. The authors provide a reference implementation, but there is also an implementation in the aforementioned PyGT library, which is the one that we chose.

V. Evaluation

The following experiments were run on an on-premise NVIDIA DGX Server, although a single consumer-class GPU with CUDA support is sufficient to replicate them.

A. Individual Model Evaluation

Below are the results of training the models in related work and experiment setup on the data from data set for

neural state estimation. MSE is used as the overall performance metric, while the per-node figures depict absolute error for better visual representation. We are omitting the figure for the RNN since we do not expect it to perform competitively.

Note that all models are trained in stochastic gradient descent (SGD) mode. This is because PyGT does not support batches for most of its models and is only able to process one data point at a time. Currently, this presents a disadvantage for RGCNNs. Still, since it is a technical rather than a theoretical limitation, we are using SGD on all models to make the comparison more representative, on the assumption that the training mode only affects the convergence speed and not the prediction quality once convergence has been reached.

First, looking at the convergence graphs, we can see Fig. 2 and Fig. 3 that EvolveGCN-H converges almost immediately and proceeds from the training data set to the testing one almost without loosing accuracy; EvolgeGCN-O shows similar behavior but is not as stable as the former. TGCN in Fig. 4 behaves similar as well but with more noise on the test set compared to the EvolveGCNs. PAWNN in Fig. 5 converges slower and to a lower point but generalizes to the test sequence less successfully. The convergence of the prox-linear network as shown in Fig. 6 is too noisy to analyze. These results are consistent with the number of trainable parameters for each model, which we will provide later in this section.

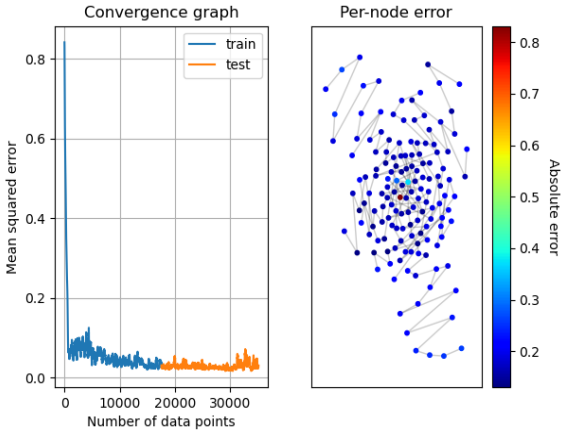

Fig. 2. EvolveGCN-H

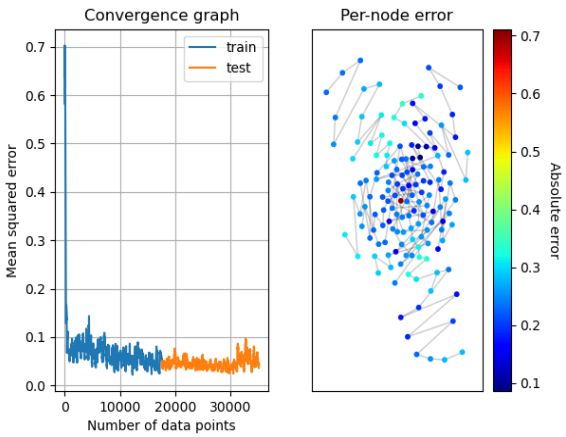

Fig. 3. EvolveGCN-O.

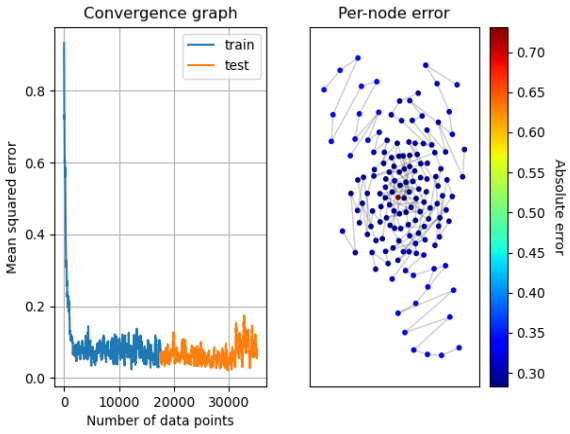

Fig. 4. TGCN.

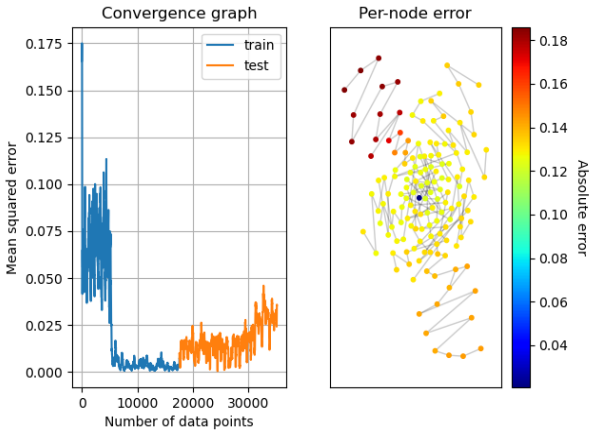

Fig. 5. PAWNN.

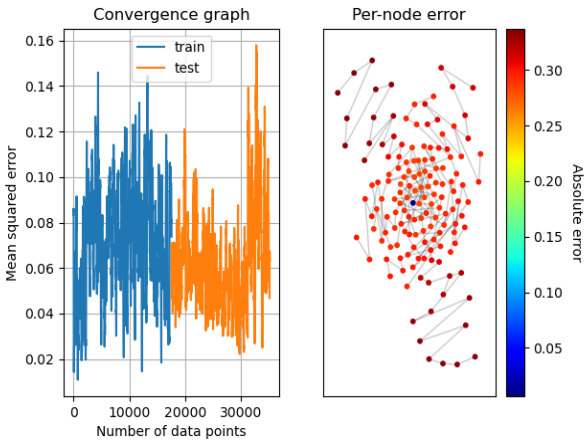

Fig. 6. Prox-linear network.

Second, the per-node error shows an important distinction between RGCNNs and other models: the former perform better on long branches of the graph, while the latter produce more consistent error values over all nodes. This becomes clear with a look on the scale of the color-coding. The EvolveGCN-H scales from 0.2 to 0.8 with low error values especially in long branches and a few high values in the center, while EvolveGCN-O has error values between 0.1 and 0.7 with similar characteristic to the former, as shown in Fig. 2 and Fig. 3 respectively. Fig. 4 shows that TGCN has a lower delta than the former two models but, again, a similar

characteristic. The PAWNN in Fig. 5 has lower error values with a smaller delta between 0.04 and 0.18 but the error in the long branches is higher than in the center. Finally, the prox-linear network as shown in Fig. 6 still has a small delta compared to the RGCNNs with 0.05 to 0.3 with a similar characteristic to the PAWNN.

B. Model Comparison

We have used 10 most common random seeds [30] to collect performance statistics, which is presented in Fig. 7. All the data collected is also available in a supplementary file results.xlsx in our repository (see Availability of data and code).

Looking at Fig. 7 in isolation, it is hard to determine the best performing model: PAWNN and RGCNNs produce very similar average results. The Prox-linear network is remarkably invariant of the random seed, but its performance is worse than the competition. Finally, RNN is far behind all other models. This suggests that temporal correlations in isolation are significantly less important for the SE problem than spatial ones.

In terms of variation, we observe that RGCNN are relatively more consistent at their performance level, with TGCN being the most consistent of the RGCNNs. Since the main differentiating feature between TGCN and EvolveGCN is the ability of the former to better leverage temporal correlations, this suggests that incorporating them into the model is still valuable. These observations point us to a more rigorous statistical analysis of the dataset as a venue for further research into the problem.

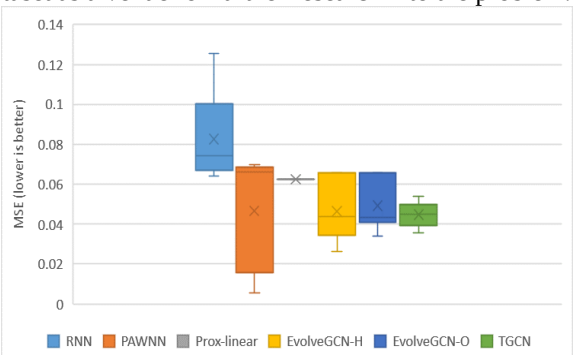

Fig. 7. Comparing the accuracy

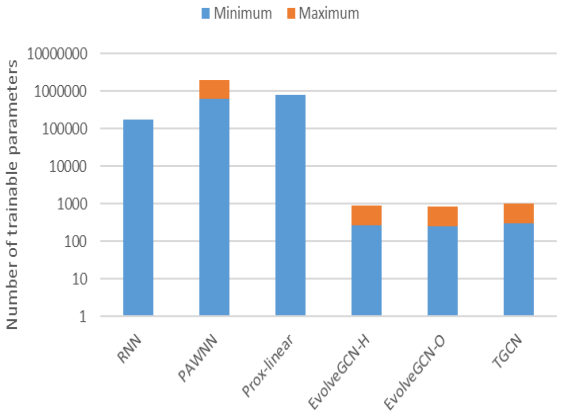

Fig. 8. Comparing the parameter count.

A much more drastic difference between conventional models and RGCNNs can be observed in Fig. 8, which depicts the number of trainable parameters of each model. Here we can see that this number is roughly three orders of magnitude lower for the RGCNNs. Note that the parameter axis is logarithmic and that some models have a variable number of parameters.

It would be reasonable to assume that this difference in complexity translates into a significantly faster training process for the RGCNNs, but that is not the case in practice. Unfortunately, it is currently impossible to perform a representative comparison of training time between conventional and RGCNN models due to computational inefficiencies in the PyGT library [27]. However, we can assume that the training time for ideal implementations would be roughly proportional to the number of parameters in each model. By that logic, RGCNNs should have the advantage in this metric.

VI. Conclusion

With the experimental data from Evaluation we can claim that the question posed Experiment Setup in is answered in the affirmative. In other words, generic RGCNN models applied to the NSE problem offer results competitive with those from models developed specifically for this task and at a much lower cost in model complexity. While specialized models rely on a large number of parameters to fit all the data at once and have problems generalizing, RGCNNs focus on approximating the local interactions between nodes in a graph. As a result, they can better generalize to new data and possibly even to new grid topologies.

The next step in research on this topic would be developing more specialized RGCNNs focused on NSE that can also leverage the inherent transferability of GCNs. There is also much room for improving the computational performance of these models.

Availability of Data and Code

We welcome the reviewers and other authors to replicate our experiments. For this purpose, we provide the complete source code along with the dataset in the following GitLab repository: https://gitlab.com/funbotan/transense-model-comparison.

The repository README will provide detailed instructions for setting up the environment.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Alexander Berezin and Stephan Balduin conducted the research and wrote the paper; Thomas Oberließen and Sebastian Peter provided the dataset; Eric Veith and Sebastian Lehnhoff provided scientific advisory; all authors approved the final version.

Acknowledgment

This research is a part of project TRANSENSE, funded by the German Federal Ministry for Economic Affairs and Energy (FKZ 03EI6044A).

ACRONYMS

ANN: Artificial Neural Network.
EvolveGCN: Evolving Graph Convolutional Network.
GCN: Graph Convolutional Network.
GRU: Gated Recurrent Unit.
LAV: Least Absolute Value.
MSE: Mean squared error.
NSE: Neural State Estimation.
P2N2: Pruned Physics-Aware Neural Network.
PAWNN: Physics-Aware Neural Network.
PF: Power Flow.
PSSF: Power System State Forecasting.
PyGT: PyTorch Geometric Temporal.
RGCNN: Recurrent Graph Convolutional Neural Network.
RNN: Recurrent Neural Network.
SE: State Estimation.
SGD: Stochastic gradient descent.
TGCN: Temporal Graph Convolutional Network.

REFERENCES

[1] H. Seidl, S. Mischiner, and R. Heuke, "Beobachtbarkeit und steuerbarkeit in energiesystemen – eine handlungsanalyse der dena-plattform systemdienstleistungen," Deutsche Energie-Agentur GmbH, 2016.

[2] F. F. Wu, "Power system state estimation: A survey," *International Journal of Electrical Power & Energy Systems*, vol. 12, no. 2, pp. 80–87, 1990.

[3] O. Krause, D. Martin, and S. Lehnhoff, "Under-determined WLMS state estimation," in *Proc. of 2015 IEEE PES Asia-Pacific Power and Energy Engineering Conf*., 2015.

[4] S. Balduin, *et al*., "Towards a universally applicable neural state estimation through transfer learning," in *Proc. of 2021 IEEE PES Innovative Smart Grid Technologies Europe*, 2021.

[5] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[6] H. T. Siegelmann and E. D. Sontag, "On the computational power of neural nets," *Journal of Computer and System Sciences*, vol. 50, no. 1, pp. 132–150, 1995.

[7] M. Bodén and J. Wiles, "Context-free and context-sensitive dynamics in recurrent neural networks," *Connection Science*, vol. 12, no. 3-4, pp. 197–210, 2000.

[8] K. R. Mestav, J. Luengo-Rozas, and L. Tong, "State estimation for unobservable distribution systems via deep neural networks," in *Proc. of 2018 IEEE Power & Energy Society General Meeting*, 2018.

[9] T. Nakagawa, Y. Hayashi, and S. Iwamoto, "Neural network application to state estimation computation," in *Proc. of the First Int. Forum on Applications of Neural Networks to Power Systems*, 1991, pp. 188–192.

[10] A. S. Zamzam, X. Fu, and N. D. Sidiropoulos, "Data-driven learning-based optimization for distribution system state estimation," *IEEE Trans. on Power Systems*, vol. 34, no. 6, pp. 4796–4805, 2019.

[11] L. Zhang, G. Wang, and G. B. Giannakis, "Real-time power system state estimation and forecasting via deep unrolled neural networks," *IEEE Trans. on Signal Processing*, vol. 67, no. 15, pp. 4069–4077, 2019.

[12] A. S. Zamzam, X. Fu, and N. D. Sidiropoulos, "Data-driven learning-based optimization for distribution system state estimation," *IEEE Trans. on Power Systems*, vol. 34, no. 6, pp. 4796–4805, 2019.

[13] P. P. Barbeiro, J. Krstulovic, H. Teixeira, J. Pereira, F. J. Soares, and J. P. Iria, "State estimation in distribution smart grids using autoencoders," in *Proc. of 2014 IEEE 8th Int. Power Engineering and Optimization Conf.*, 2014, pp. 358–363.

[14] A. S. Zamzam and N. D. Sidiropoulos, "Physics-aware neural networks for distribution system state estimation," *IEEE Trans. on Power Systems*, vol. 35, no. 6, pp. 4347–4356, 2020.

[15] M. Abdel-Nasser, K. Mahmoud, and H. Kashef, "A novel smart grid state estimation method based on neural networks," *Int. Journal of Interactive Multimedia and Artificial Intelligence*, vol. 5, no. 1, pp. 92–100, 2018.

[16] M. Q. Tran, A. S. Zamzam, and P. H. Nguyen, "Enhancement of distribution system state estimation using pruned physics-aware neural networks," *arXiv*, arXiv:2102.03893, 2021.

[17] M. J. Hossain and M. Rahnamay–Naeini, "State estimation in smart grids using temporal graph convolution networks," in *Proc. of 2021 North American Power Symposium*, 2021.

[18] V. Bolz, J. Rueß, and A. Zell, "Power flow approximation based on graph convolutional networks," in *Proc. of 2019 18th IEEE International Conference on Machine Learning and Applications*, 2019, pp. 1679–1686.

[19] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," *arXiv*, arXiv:1612.07659, 2016.

[20] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *Trans. Neur. Netw*., vol. 20, no. 1, pp. 61–80, Jan. 2009.

[21] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *arXiv*, arXiv:1606.09375, 2016.

[22] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv*, arXiv:1707.01926, 2017.

[23] B. Rozemberczki, P. Scherer, O. Kiss, R. Sarkar, and T. Ferenci, "Chickenpox cases in hungary: A benchmark dataset for spatiotemporal signal processing with graph neural networks," *arXiv*, arXiv:2102.08100, 2021.

[24] S. Meinecke, D. Sarajlić, S. R. Drauz *et al*., "SimBench—A benchmark dataset of electric power systems to compare innovative solutions based on power flow analysis," *Energies*, vol. 13, no. 12, Jan. 2020.

[25] J. Hiry, "Agent-based discrete-event simulation environment for electric power distribution system analysis," Dissertation, TU Dortmund, 2022.

[26] C. Kittl, "Entwurf und Validierung eines individualitätszentrierten, interdisziplinären Energiesystemsimulators basierend auf ereignisdiskreter Simulation und Agententheorie," Dissertation, TU Dortmund, Dortmund, 2022.

[27] B. Rozemberczki, P. Scherer, Y. He, *et al*., "PyTorch geometric temporal: spatiotemporal signal processing with neural machine learning models," in *Proc. of the 30th ACM Int. Conf. on Information and Knowledge Management*, 2021, pp. 4564–4573.

[28] A. Pareja, G. Domeniconi, J. Chen, *et al*., "EvolveGCN: Evolving graph convolutional networks for dynamic graphs," *arXiv*, arXiv:1902.10191, 2020.

[29] L. Zhao, Y. Song, C. Zhang, *et al*., "T-GCN: A temporal graph convolutional network for traffic prediction," *IEEE Trans. on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3848–3858, Sep. 2020.

[30] A. Bilogur. Most common random seeds. Kaggle. Dec. 2017. [Online]. Available: https://www.kaggle.com/code/residentmario/kernel16e284dcb7/notebook

Alexander Berezin was born in Moscow on September 27, 1996, studied applied mathematics at the Moscow Institute of Electronic Technology and made his Bachelor's and Master's degree in 2018 and 2020, respectively. Since 2021, he is employed at OFFIS - Institute for Information Technology in Oldenburg and working primarily on project TRANSENSE. His primary research interest at

the moment is the application of graph convolutional neural networks to the power system state estimation problem.

Stephan Balduin was born in Aachen, Germany on 5th September 1987 has studied computer science at the University of Oldenburg and made his Bachelor's and Master's degree in 2014 and 2017, respectively. Since 2017, he is employed at OFFIS - Institute for Information Technology in Oldenburg and has worked on different projects in the context of artificial intelligence for the electrical energy system. He focuses his research on AI-based analysis of energy systems using surrogate models.

Thomas Oberließen was born in Paderborn, Germany on 19th August 1994 has studied industrial engineering at the Technical University of Dortmund and made his Bachelor's and Master's degrees in 2018 and 2020, respectively. Since 2020, he is employed at the Technical University of Dortmund as a research associate, where he focuses on agent-based distribution grid simulation and other research topics around distribution grid planning and operation.

Sebastian Peter was born in Witten, Germany on 7th June 1992, has studied Computer Science at the Technical University of Dortmund and completed his Master's degree in 2021. Since 2021 he focuses on agent-based discrete-event distribution grid simulations as part of his employment as a research associate at the Technical University of Dortmund, while also working on various other topics related to distribution grid planning and operation.

Eric MSP Veith was received his Ph.D. degree in computer science from the Freiberg University of Mining and Technology, Germany, in 2017. He currently leads the junior research group Adversarial Resilience Learning at the University of Oldenburg, Germany, and is a member of OFFIS. Previously, he was an R&D group manager and researcher at OFFIS, Oldenburg, Germany. His research interests are focused on competing agents in the ARL methodology, applying Evolutionary Deep Reinforcement Learning to critical national infrastructures for their resilient operation. He is a member of the ACM, IEEE, German VDE, and DIN. He serves as a TPC member at numerous conferences such as ACM e-Energy.