Augmented Pre-trained Graph Neural Networks for Grid-Supportive Flexibility Control

Simon Grafenhorst
Karlsruhe Institute of Technology
Institute for Automation and Applied Informatics
Karlsruhe, Germany
simon.grafenhorst@kit.edu

Gökhan Demirel
Karlsruhe Institute of Technology
Institute for Automation and Applied Informatics
Karlsruhe, Germany
goekhan.demirel@kit.edu

Kevin Förderer
Karlsruhe Institute of Technology
Institute for Automation and Applied Informatics
Karlsruhe, Germany
kevin.foerderer@kit.edu

Veit Hagenmeyer
Karlsruhe Institute of Technology
Institute for Automation and Applied Informatics
Karlsruhe, Germany
veit.hagenmeyer@kit.edu

Abstract

With the ongoing transformation in electricity grids, new components such as decentralized generation systems and high-load electric vehicle charging infrastructures are installed. To foresee congestion scenarios and lay out flexibility-providing distributed energy resources, power flow calculations are generally required. Since power flow calculations are computationally expensive, various neural network architectures have been developed in the literature to infer power flow estimates. In the present paper, we extend the application of a graph neural network (GNN) architecture initially developed for power flow approximation to infer minimum load or generation requirements for congestion mitigation. Evaluating different training procedures and models, we find that a two-step training process using a pre-trained power flow inferring GNN has a head start in training compared to end-to-end training of the GNN. The pre-trained model demonstrates robust generalization capabilities, effectively inferring optimal power values for congestion mitigation across grid topologies outside of the training distribution. Furthermore, as the second training step is not limited to the task of congestion mitigation, the approach can easily be adapted to other use cases.

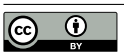
CCS Concepts

• **Computing methodologies** → **Neural networks**; • **Hardware** → **Power networks**.

Keywords

Graph neural networks, power flow approximation, distributed energy resource, flexibility control

ACM Reference Format:
Simon Grafenhorst, Gökhan Demirel, Kevin Förderer, and Veit Hagenmeyer. 2025. Augmented Pre-trained Graph Neural Networks for Grid-Supportive Flexibility Control. In *The 16th ACM International Conference on Future and Sustainable Energy Systems (E-ENERGY '25), June 17–20, 2025, Rotterdam, Netherlands.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3679240.3734663



ACM ISBN 979-8-4007-1125-1/25/06
https://doi.org/10.1145/3679240.3734663

1 Introduction

Electricity grids worldwide are undergoing a substantial transformation, primarily driven by the declining cost of renewable energy generation and the implementation of policies promoting renewable energy sources' utilization. These developments are reshaping power distribution systems and necessitating adaptation to the inherent variability of weather-dependent power generation. Consequently, distribution grids must evolve to enhance their flexibility and adaptability to balance supply and demand effectively. The widespread installation of Electric Vehicle (EV) charging infrastructure is introducing new demands and, through bidirectional charging, facilitates the integration of residential and commercial entities into flexibility markets. Consequently, conventional radial distribution networks, characterized by unidirectional power flow from centralized generation sources to end-users, are transitioning towards decentralized, bidirectional grids capable of dynamic power generation, storage, and distribution. To support this transformation, Distribution System Operators (DSOs) incorporate flexibility-providing Distributed Energy Resources (DERs) and advanced grid management software. These technologies ensure that grid operators can manage and balance loads, thereby maintaining stability and efficiency even with power flows becoming more variable.

For grid operation as well as grid-aware flexibility scheduling, fast and computationally inexpensive power flow calculations are becoming increasingly important. By determining how the power flows through the grid, the infrastructure utilization, voltage levels, and other vital parameters can be computed. As a result, congestion scenarios become predictable, allowing for effective mitigation by controlling flexibility providers.

One approach for solving power flow calculations is the utilization of machine learning techniques [15]. Typically, the results of a Newton-Rhapson power flow solver are learned by a neural network, which then infers new power flow results with little computational effort. Recent publications have advanced this approach to utilize Graph Neural Networks (GNNs) [3, 10, 11]. These graph-based neural networks have the key advantage of being able to

learn the fundamental structure of the data and infer power flow calculations even for electricity grids that are not included in the training data. In this paper, we propose a novel GNN architecture, which is not only able to infer power flows, but also the power required by a DER to mitigate congestion of a power line. This downstream task is achieved by adding further layers to the Artificial Neural Network (ANN). It represents a category of tasks fundamentally reliant on power flow computations, including, but not limited to, the calculation of the available transfer capability of a grid, minimization of power loss, and the optimal placement of DERs, among others [20]. We hypothesize that employing an existing power flow inferring GNN as a foundation accelerates the training process for downstream tasks and enhances performance compared to a single model. By training the models on a diverse set of electrical grid topologies and realistic loading scenarios, we aim to improve generalization across multiple grid configurations and test cases. Although the idea of pre-trained GNN architectures has been explored in many contexts [12], to the best of our knowledge, no previously published work has used pre-trained GNNs in the energy domain. Beyond pre-training, our approach distinguishes itself from prior research through the utilization of authentic load profiles and network topologies sourced from the Simbench dataset, in contrast to relying on synthetic profiles generated based on load distributions [1, 3, 11]. Moreover, we demonstrate the model's capability for cross-grid generalization. To facilitate reproducibility, all code developed in the present work is available open source on GitHub [1].

The remainder of the paper is organized as follows: In the next chapter, we first outline related work on the application of ANNs and GNNs in power systems. Additionally, we explain how our presented approach is similar to other methods for power flow analysis and how it differs from them. In Section 3, we present the GNN architecture, as well as the test and training datasets. In Section 4, we provide an evaluation of our experiments under different configurations and datasets. Section 5 discusses the results, potential shortcomings, and improvements, comparing them with the existing literature. Section 6 concludes the paper.

2 Related Work

ANNs have been applied to power system problems for more than 30 years [9, 21]. They can be used to solve power flows for specific electricity grids, for example, by training the neural network on the diagonal elements of the grid's matrix representation as input data [15]. This obviously has the deficit of the training only being useful for the specific electricity grid, and generalization is limited to other load scenarios in the same grid. In more recent times, power system analysis has also been conducted using GNNs [3, 11]. This variety of neural networks leverages the structural relationships in the data to understand the underlying properties, therefore being invariant to permutations [16]. To approximate power flows and exploit the spatial grid topology, spatial-based Graph Convolutional Networks (GCNs) are widely used [1, 3, 10].

These GNNs operate by diffusing information across the graph through a message-passing algorithm. Individual nodes exchange information with their neighbors over K hops throughout multiple iterations (i.e., neural network layers). Given the intuitive nature of power grids as graphs, in which loads and generators are represented as nodes and lines are represented as edges, the application of GNNs to power flow calculations is a self-evident proposition.

Power flow and optimal power flow solvers are often based on iterative optimization algorithms [3], resulting in high computational costs for large systems. A physics-informed GNN is used in [3] to estimate grid states accurately. The architecture consisting of Topology Adaptive Graph Convolutional (TAGConv) [5] layers is highlighted in [3] as it shows low Mean Square Error (MSE) values for different amounts of missing sensor data. In a recent publication by Lin et al. [11], a GNN architecture for the approximation of power flows is presented. They achieve high speedups of Alternating Current (AC) powerflow calculations with an accuracy superior to simple Direct Current (DC) powerflow approximations. Their large model consists of five layers, a hidden dimension of 512, and messages are passed for three hops ($K = 3$).

The output tensor $\mathbf{X}'$ of the TAGConv operator presented by Du et al. [5] is described by

$$\mathbf{X}' = \sum_{k=0}^{K} \left(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)^K \mathbf{X}\mathbf{W}_K. \tag{1}$$

With the adjacency matrix $\mathbf{A}$ and it's diagonal degree matrix $D_{ii} = \sum_{j=0} A_{ij}$, the term $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ denotes the normalized adjacency matrix $\hat{\mathbf{A}}$. The parameter K denotes the number of hops that are taken into consideration in this layer, and $\mathbf{W}_K$ denotes the weights that are multiplied with the input tensor $\mathbf{X}$.

In the literature, machine learning models are often trained in a two-stage process [8, 18, 23]. In the first training stage, one or more layers of a neural network are trained to learn certain fundamental relationships. This is undertaken, for instance, with the help of imitation learning [8]. In imitation learning, an "agent uses instances of performed actions to learn a policy that solves a given task.", see [8] for details.

In a second training stage, the neural network is extended by more layers without resetting the previously trained layers. The subsequent layers are then trained to optimize further or abstract the previously learned layers. The goal is either to improve the existing representations or to learn additional relationships that were not captured before. The network's first layers reduce the depth of the problem so that the subsequent layers have to model a less complex relationship [17]. This concept of first training on broad data that can later be adapted to a range of downstream tasks is similar to the definition of foundation models [2]. However, the model presented in the following chapter is not trained in a self-supervised manner and contains a comparably low number of trainable parameters. It, therefore, differs in key aspects from the definition of foundation models given by Bommasani et al. [2].

When optimizing the schedule of flexibility providing DERs, authors often neglect the grid serving aspect and only minimize costs or emissions [4]. For instance, Geidl et al. presented the Energy Hub concept in 2007, describing a multi-energy DER optimized to be as cost-efficient as possible [6]. In order to incorporate grid constraints, algorithms may linearize power flow equations for small grids to solve them quickly [13, 19]. However, to solve power flows

[1] https://github.com/KIT-IAI/PretrainedPowerflowGNN

for large grids, a numerical Newton-Rhapson power flow solver is frequently employed [7, 13, 22].

3 Method

This chapter introduces the implemented GNN architectures for both the pre-trained model and the cold-start model. During pre-training, a GNN model is trained to infer power flow solutions. This model serves as the foundation and is then extended to approximate the power (positive or negative) necessary for congestion mitigation.

In Medium Voltage (MV) grids with a significant presence of distributed generation and large numbers of EVs and heat pumps, line loads may exceed the permissible limits. The proposed solution is a flexible DER acting as a load or generator to reduce a line's load below a predefined limit. However, due to monetary expenses linked to the use of the flexibility, the requested change in power should be as low as possible.

3.1 Training Data

Power Flows. The training data is generated from SimBench [14] MV grids. The SimBench dataset provides realistic topologies of different grids and includes load and generation profiles at 15-minute intervals for an entire year. Each time step is treated as an independent data point for the models. A Newton-Raphson solver from Pandapower [22] computes the power flows for each time step to generate the ground truth data.

We represent the distribution grid as a graph $G = (N, E)$, where the set of nodes $N = \{1, \ldots, n\}$ represents the buses, and the set of edges $E \subseteq N \times N$ corresponds to power lines. One data point consists of a graph with the described topological information. The node feature input tensor includes the active and reactive power values (P and Q), while the edge feature tensor consists of the line conductance ($G = 1/R$). The output tensor contains the voltage magnitude and angle ($|V|, \theta$) at every node.

To generate sufficient training samples, we apply the load profiles to the SimBench grids `"1-MV-urban"`, `"1-MV-comm"`, and `"1-MV-rural"`. These distribution grids represent typical urban, commercial, and rural medium voltage grids. For each grid, 70.000 power flows are calculated. This includes two whole years of load profiles, ensuring that the model is trained on a wide range of possible situations.

To obtain edge features, the line resistances are scaled to the range from zero to one and inverted. These edge weights represent respective conductances in the electrical grid. An electrical line with no resistance would result in a weight of one, indicating maximum conductance and maximum influence on neighboring nodes. Conversely, higher resistance leads to lower weights, reducing the line's impact on adjacent nodes. This scaling and inversion approach mirrors the physical properties of the network.

After preprocessing, the samples of two grids are shuffled to break any inherent order or patterns that might lead to biased learning. The randomized data is then split into training and validation sets with an 70% to 30% ratio, respectively. The test dataset contains samples from the third grid.

Calculating load or generation required to mitigate congestion. To mitigate congestion, Algorithm 1 is employed [7]. The algorithm computes the power required at a particular bus to relieve congestion of a single line so that the utilization is below a specified upper limit. The result is not necessarily the global optimum. Instead, the algorithm determines how the isolated action at each bus can solve the congestion, which is, for example, relevant for non-centralized DER coordination approaches. This process can take a significant amount of time, even with high-performance computing.

Algorithm 1 Reduction of the utilization of line l, adapted from [7]

$g \leftarrow Grid$; $b \leftarrow Bus$; $l \leftarrow Line$; $lim \leftarrow UtilLim$
$l_{util} \leftarrow$calcPowerFlow(g) {line utilization}
$m \leftarrow 1$ {modifier}
$\epsilon \leftarrow 0.1$
while $m > \epsilon$ **do**
  $g(b)$.add(m) {modify DER power}
  $l_{util}^{prev} \leftarrow l_{util}$
  $l_{util} \leftarrow$calcPowerFlow(g)
  if abs(l_{util} - lim) > abs(l_{util}^{prev} - lim) **then**
    $m \leftarrow m \cdot -0.5$
  end if
end while
return $g(b)$

Algorithm 1 begins by calculating a power flow without any modification. However, due to the lack of consideration for grid constraints, the line utilization of line l can exceed design limits. To solve this shortcoming, the iterative algorithm adjusts the load ($m > 0$) and generation ($m < 0$) located at bus b of a flexibility providing DER. Minimizing the change in power required to meet the line utilization limit mitigates the congestion in the most economical manner possible. Due to the iterative nature of the algorithm and the power flow calculation in every iteration, the computation is inherently slow.

The optimal power value calculated by the algorithm is used as a label for the node where the controllable DER is located. The input data is in the same form as for the power flow analysis but includes additional binary features, which identify the node the DER is placed at and the congested line. The training data only includes time steps with voltage values in the grid exceeding the regulatory guidelines because samples without voltage exceedance and no necessary DER power adjustment could comprise training effectiveness.

Training data is generated with the iterative algorithm using one DER location in each grid. For each of the three grid topologies, 70,000 load cases are evaluated. The training and validation datasets are comprised of two grid topologies, while the testing dataset contains a third topology. This configuration enables the evaluation of the models' capability to generalize.

3.2 Power Flow Model

The core of the proposed model comprises $\mathcal{L}_{\mathcal{PF}}$ PowerFlowNet [11] layers. Each layer consists of a two-layer multilayer perceptron to process the messages passed by the neighboring nodes and a

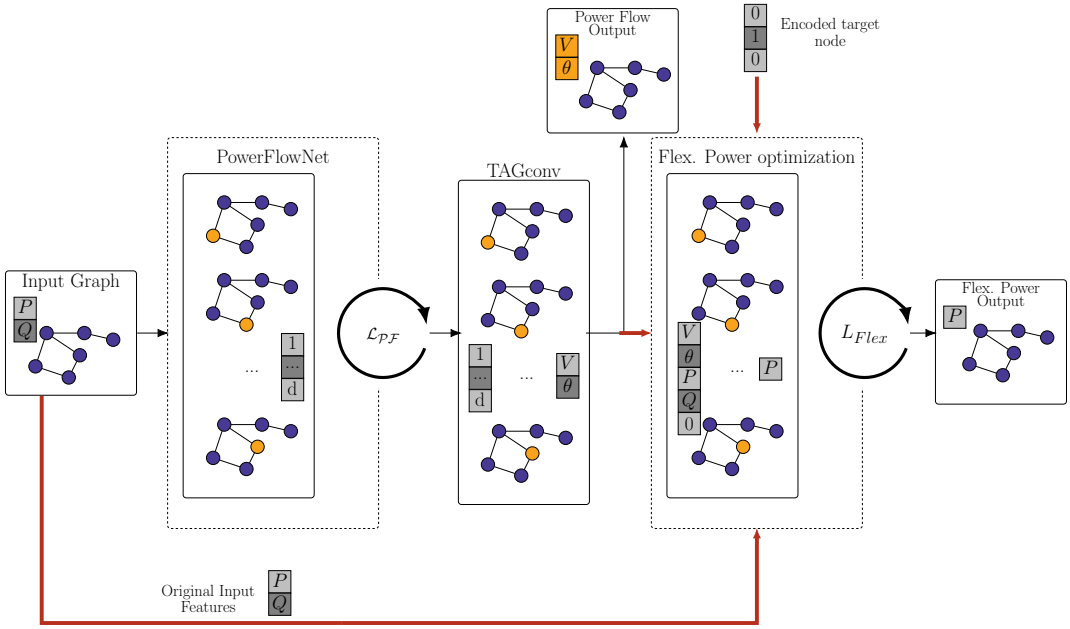


Figure 1: GNN Architecture for the pre-trained model for congestion mitigation. The red arrows denote inputs for calculating the flexibilities power requirements.

TAGConv [5] layer. The first layer has an input dimension of two, corresponding to the two node features P and Q. The output dimension of this first layer is set to the hidden dimension d. The value of d determines the dimension of the node representation after each message-passing step and essentially determines model complexity and number of learnable parameters. A comprehensive hyperparameter sweep is conducted to find the optimal configuration. This involves varying the number of layers $\mathcal{L}_{\mathcal{PF}}$, the hidden dimension d of the node representation vector in the l-th layer, and the K-hop number, which defines the depth of neighborhood aggregation. The results suggest that increasing the complexity of the network allows it to capture more detailed and nuanced patterns in the data, which is essential for accurately modeling power flow dynamics. In training, using multiple grid topologies and about 15.000 load cases with about 1.6 million nodes in total, various hyperparameter configurations are compared. A final configuration of five layers, a hidden dimension of 256, and a K-hop of four are chosen.

3.3 Model Architecture for Mitigating Congestion

To approximate the power needed to mitigate congestion, two main models with independently optimized hyperparameters are constructed. However, one model is pre-trained with power flow data, and the other model is trained from a cold start. The hidden dimension is varied between 128 and 512, the number of hops (K) is varied between three and five and the number of layers L_{Flex} is varied between two and seven. The models are trained with a learning rate of $l_r = 0.001$ using the ADAM optimizer for 100 epochs, which is more than enough for both models to minimize their respective losses.

The pre-trained model consists of the power flow model and a separate GNN, which outputs the approximate optimal power replacing the previously described iterative algorithm. After training the power flow model, its weights stay fixed for the second training procedure. In addition to the estimated voltage magnitude $|\hat{V}|$ and phase angle $\hat{\theta}$, the first of the additional layers receives the active power P, reactive power Q, and an indicator that specifies the node with the flexibility as inputs. A visual representation of this architecture is shown in Figure 1. In contrast, the cold-start model consists of a single GNN.

4 Evaluation

	Test		
	Grid 0	Grid 1	Grid 2
cold-start	0.67 (0.60 %)	1.18 (0.81 %)	0.92 (0.72 %)
pre-trained	0.02 (0.36 %)	0.01 (0.17 %)	0.01 (0.23 %)
	Validation		
cold-start	4.01 (1.91 %)	3.84 (1.87 %)	3.64 (1.82 %)
pre-trained	1.22 (0.91 %)	1.09 (0.86 %)	0.78 (0.74 %)

Table 1: MSE in W and Mean Absolute Percentage Error (MAPE) of different models, inferring the flexibilities power for a node in a known topology and in a new topology.

The power flow GNN implemented as the first stage of the pre-trained model contains four layers with a hidden dimension of $d = 128$ and $K = 4$ hops. Tests conducted on the three medium voltage SimBench grids reveal an average MSE of $2.7 \cdot 10^{-4}$ p.u.. After an extensive hyperparameter sweep, a GNN containing four

	Grid 0	Grid 1	Grid 2
pre-trained	4.834	8.574	4.001
perfect input pre-trained	1.738	2.160	0.989

Table 2: Normalized test MSE multiplied by 10^5 of the pre-trained model with inputs from the power flow model compared to the results obtained using perfect inputs without an additional error introduced by the power flow inference.

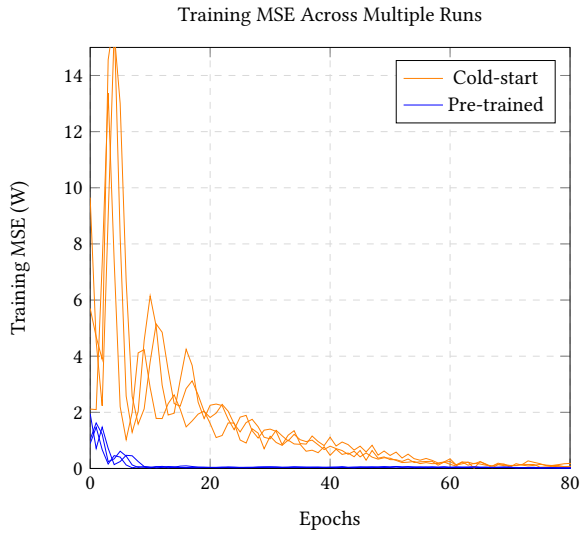


Figure 2: Comparison of the training MSE between the pre-trained models (blue) and cold-start models (orange). Each training run is conducted on an individual dataset containing two of the three grid topologies.

PowerFlowNet layers is used as the second stage of the pre-trained model. With a hidden dimension of $d = 128$ and messages passing for $k = 4$ hops, the lowest MSE were achieved. For the cold-start model, a similar architecture with five modified PowerFlowNet layers, $d = 128$, and $k = 4$ shows the best scores.

The test dataset for evaluating the performance of the models in different scenarios consists of a new grid topology that is not included in the training dataset. The training procedure is performed three times for each model, with the training and evaluation datasets containing two grid topologies and the test dataset containing the third topology. The MSE always refers to the power of the flexibility in W, and training is always performed for 100 epochs. The input data for the pre-trained model is generated using the previously detailed power flow model, with an added error to the input data. It is observed that by using perfect input data without the errors introduced by inferring the power flow, the MSE of the results obtained from the pre-trained model can be improved (see Table 2).

The training procedure for the five-layer cold-start GNN takes about 9:40 minutes on an NVIDIA A100 graphics card, while the second stage training for the four-layer pre-trained GNN takes about 9:00 minutes. Figure 2 depicts 80 epochs of training for the pre-trained model (in blue) and the cold-start model (in orange). The pre-trained model consistently outperforms the cold-start model. However, the difference between the models and the test cases diminishes progressively.

5 Discussion

Evaluation of the power flow model reveals a low test MSE of $2.7 \cdot 10^{-4}$ p.u.. This error score is comparable to the results published by Lin et al. [11]. During the training process, the pre-trained model initially possesses a knowledge advantage. As more information about the electricity grid is available to the pre-trained GNN, this is to be expected. However, the performance of the pre-trained model is superior to the cold-start model even after 100 epochs of training. Similarly, the pre-trained model outperforms the cold-start model in every metric in Table 1. The test datasets consist of load cases only containing a grid topology that is not included in the training datasets. Both models can infer the optimal power the DER needs to provide with good accuracy and are able to generalize. Using perfect power flow data for the pre-trained model and not using power flow results inferred by the power flow model improves the MSE of the pre-trained model by a factor of approximately 2.8 to 4, depending on the grid. The cold-start model exhibits a longer training duration compared to the pre-trained model, mainly due to the cold-start model's increased architectural complexity, specifically the inclusion of an additional layer. However, this training efficiency advantage of the pre-trained model is diminished when considering the supplementary training of the power flow model, which takes about 30 minutes. The pre-trained model's head-start in training partially offsets this effect. For instance, achieving an MSE below 0.5 W necessitates approximately 10 training epochs for the pre-trained model, while the cold-start model requires approximately 50 epochs. Regardless, the present paper does not quantify the total training effort, as the generalizing power flow model serves as a foundation for multiple downstream applications. Even when establishing a precise correlation between training effort and model performance, its optimization is dependent on the task at hand. By using a GNN, the optimal load for mitigating a line overload can be approximated around 500 times faster than by using the iterative Algorithm 1 (≈ 5 ms vs. 3 s using an office PC without inference on a GPU). This is especially relevant for DSOs controlling large amounts of DERs as the cost of computation is reduced drastically. The pre-trained model offers vastly improved performance compared to the cold-start model.

The task of optimizing the power needed to mitigate congestion demonstrates the ability of GNNs to learn optimization tasks related to power flow calculations. Particularly, the pre-trained model is able to generalize with good accuracy. This allows the use of the trained model in grid topologies that are not included in the training dataset. The fast inference enables fast responses to congestion scenarios such that DERs are able to provide the needed flexibility with low latency.

6 Conclusion and Outlook

Many fundamental operations within electricity grids involve power flow analysis using numerical solvers, and efforts are made to infer

power flows using machine learning techniques. In this paper, we not only confirm the viability of inferring power flows with GNNs but also exemplify the potential of augmented use cases. While GNNs inferring power flows and GNNs optimizing various tasks exist, the connection of these two domains has not yet been made. We advance the research in this field by showing that a power flow inferring GNN can be extended to predict solutions for problems based on power flows. A head-start of the pre-trained model and vastly improved performance can be observed during the conducted training processes, compared to a cold-start model.

The low error scores achieved by the pre-trained model are indicative of its robust generalization and inference capabilities across grid topologies that were excluded from the training dataset. This performance suggests the model's potential for practical application, particularly in scenarios requiring a rapid response to congestion or in environments with limited computational resources. The low computing requirements for inference suggest the feasibility of a decentralized calculation of the power to be provided by a flexibility.

In future studies, reinforcement learning may offer an alternative approach, as generating extensive training data is a significant challenge. Reinforcement learning has the potential to accelerate the training process by enabling models to learn from simulated interactions rather than relying exclusively on pre-existing data sets. This eliminates the need for training data that represents the "ground truth", making it applicable to problems for which perfect solutions are unavailable or computationally infeasible, such as the control of several flexibilities in a grid. Moreover, an extensive evaluation employing real-world datasets and grid topologies is a part of future work.

Acknowledgments

The authors gratefully acknowledge funding by the German Federal Ministry of Education and Research (BMBF) within the Kopernikus Project ENSURE "New ENergy grid StructURes for the German Energiewende".

References

[1] Valentin Bolz, Johannes Rueß, and Andreas Zell. 2019. Power Flow Approximation Based on Graph Convolutional Networks. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE, Boca Raton, FL, USA, 1679–1686. https://doi.org/10.1109/ICMLA.2019.00274

[2] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2022. On the Opportunities and Risks of Foundation Models. https://doi.org/10.48550/arXiv.2108.07258 arXiv:2108.07258 [cs].

[3] Steven De Jongh, Frederik Gielnik, Felicitas Mueller, Loris Schmit, Michael Suriyah, and Thomas Leibfried. 2022. Physics-informed geometric deep learning for inference tasks in power systems. *Electric Power Systems Research* 211 (Oct. 2022), 108362. https://doi.org/10.1016/j.epsr.2022.108362

[4] Tao Ding, Wenhao Jia, Mohammad Shahidehpour, Ouzhu Han, Yuge Sun, and Ziyu Zhang. 2022. Review of Optimization Methods for Energy Hub Planning, Operation, Trading, and Control. *IEEE Transactions on Sustainable Energy* 13, 3 (July 2022), 1802–1818. https://doi.org/10.1109/TSTE.2022.3172004

[5] Jian Du, Shanghang Zhang, Guanhang Wu, Jose M. F. Moura, and Soummya Kar. 2017. Topology Adaptive Graph Convolutional Networks. https://doi.org/10.48550/ARXIV.1710.10370 Version Number: 5.

[6] Martin Geidl, Gaudenz Koeppel, Patrick Favre-Perrod, Bernd Klockl, Goran Andersson, and Klaus Frohlich. 2007. Energy hubs for the future. *IEEE Power and Energy Magazine* 5, 1 (Jan. 2007), 24–30. https://doi.org/10.1109/MPAE.2007.264850

[7] Simon Grafenhorst, Gökhan Demirel, Kevin Förderer, and Veit Hagenmeyer. 2024. Grid Aware Portfolio Optimization of a Multi-Energy DER. In *2024 IEEE Sustainable Power and Energy Conference (iSPEC)*. IEEE, Kuching, Sarawak, Malaysia, 245–250. https://doi.org/10.1109/iSPEC59716.2024.10892444

[8] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. 2018. Imitation Learning: A Survey of Learning Methods. *Comput. Surveys* 50, 2 (March 2018), 1–35. https://doi.org/10.1145/3054912

[9] Soteris A. Kalogirou. 2000. Applications of artificial neural-networks for energy systems. *Applied Energy* 67, 1-2 (Sept. 2000), 17–35. https://doi.org/10.1016/S0306-2619(00)00005-2

[10] Wenlong Liao, Birgitte Bak-Jensen, Jayakrishnan Radhakrishna Pillai, Yuelong Wang, and Yusen Wang. 2022. A Review of Graph Neural Networks and Their Applications in Power Systems. *Journal of Modern Power Systems and Clean Energy* 10, 2 (2022), 345–360. https://doi.org/10.35833/MPCE.2021.000058

[11] Nan Lin, Stavros Orfanoudakis, Nathan Ordonez Cardenas, Juan S. Giraldo, and Pedro P. Vergara. 2024. PowerFlowNet: Power flow approximation using message passing Graph Neural Networks. *International Journal of Electrical Power & Energy Systems* 160 (Sept. 2024), 110112. https://doi.org/10.1016/j.ijepes.2024.110112

[12] Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S. Yu, and Chuan Shi. 2024. Towards Graph Foundation Models: A Survey and Beyond. https://arxiv.org/abs/2310.11829 _eprint: 2310.11829.

[13] Salman Mashayekh, Michael Stadler, Gonçalo Cardoso, and Miguel Heleno. 2017. A mixed integer linear programming approach for optimal DER portfolio, sizing, and placement in multi-energy microgrids. *Applied Energy* 187 (Feb. 2017), 154–168. https://doi.org/10.1016/j.apenergy.2016.11.020

[14] Steffen Meinecke, Džanan Sarajlić, Simon Ruben Drauz, Annika Klettke, Lars-Peter Lauven, Christian Rehtanz, Albert Moser, and Martin Braun. 2020. SimBench—A Benchmark Dataset of Electric Power Systems to Compare Innovative Solutions Based on Power Flow Analysis. *Energies* 13, 12 (June 2020), 3290. https://doi.org/10.3390/en13123290

[15] V.Leonardo Paucar and Marcos J Rider. 2002. Artificial neural networks for solving the power flow problem in electric power systems. *Electric Power Systems Research* 62, 2 (June 2002), 139–144. https://doi.org/10.1016/S0378-7796(02)00030-5

[16] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* 20, 1 (Jan. 2009), 61–80. https://doi.org/10.1109/TNN.2008.2005605

[17] Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61 (Jan. 2015), 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

[18] Ruibo Shang, Kevin Hoffer-Hawlik, Fei Wang, Guohai Situ, and Geoffrey P. Luke. 2021. Two-step training deep learning framework for computational imaging without physics priors. *Optics Express* 29, 10 (May 2021), 15239. https://doi.org/10.1364/OE.424165

[19] Chengcheng Shao, Xifan Wang, Mohammad Shahidehpour, Xiuli Wang, and Biyang Wang. 2017. An MILP-Based Optimal Power Flow in Multicarrier Energy Systems. *IEEE Transactions on Sustainable Energy* 8, 1 (Jan. 2017), 239–248. https://doi.org/10.1109/TSTE.2016.2595486

[20] Bindeshwar Singh and Janmejay Sharma. 2017. A review on distributed generation planning. *Renewable and Sustainable Energy Reviews* 76 (Sept. 2017), 529–544. https://doi.org/10.1016/j.rser.2017.03.034

[21] D.J. Sobajic and Y.-H. Pao. 1989. Artificial neural-net based dynamic security assessment for electric power systems. *IEEE Transactions on Power Systems* 4, 1 (Feb. 1989), 220–228. https://doi.org/10.1109/59.32481

[22] Leon Thurner, Alexander Scheidler, Florian Schafer, Jan-Hendrik Menke, Julian Dollichon, Friederike Meier, Steffen Meinecke, and Martin Braun. 2018. Pandapower—An Open-Source Python Tool for Convenient Modeling, Analysis, and Optimization of Electric Power Systems. *IEEE Transactions on Power Systems* 33, 6 (Nov. 2018), 6510–6521. https://doi.org/10.1109/TPWRS.2018.2829021

[23] Efthymios Tzinis, Shrikant Venkataramani, Zhepei Wang, Cem Subakan, and Paris Smaragdis. 2020. Two-Step Sound Source Separation: Training On Learned Latent Targets. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Barcelona, Spain, 31–35. https://doi.org/10.1109/ICASSP40776.2020.9054172