

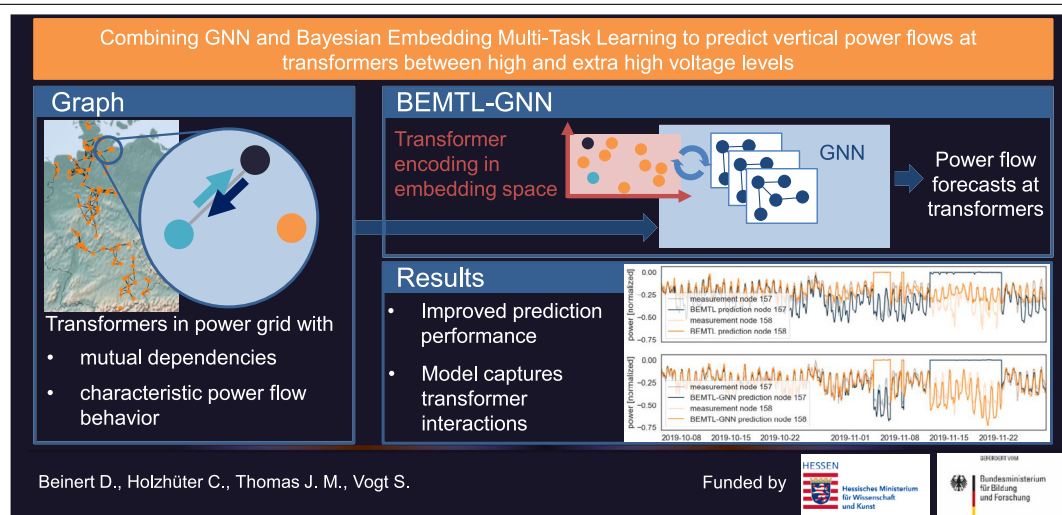
Power flow forecasts at transmission grid nodes using Graph Neural Networks

Dominik Beinert ^{a,1}, Clara Holzhüter ^{a,b,*}, Josephine M. Thomas ^b, Stephan Vogt ^b

^a Fraunhofer Institute for Energy Economics and Energy System Technology IEE, Joseph-Beuys-Straße 8, Kassel, 34117, Germany

^b Intelligent Embedded Systems Lab, University of Kassel, Wilhelmshöher Allee 73, Kassel, 34121, Germany

GRAPHICAL ABSTRACT



HIGHLIGHTS

- Combination of a Graph Neural Network and Multi-Task Learning.
- Power Flow Forecast at transformers performed on two real-world data sets.
- Bayesian multi-task embedding captures individual characteristics of transformers.
- Graph Neural Network architecture considers information from close-by transformers.
- Superior performance compared to benchmarks, particularly at interacting transformers.

ARTICLE INFO

Keywords:

Power flow forecasting
Graph Neural Network
Graph Convolutional Network
Embedding Multi-Task Learning

ABSTRACT

The increasing share of renewable energy in the electricity grid and progressing changes in power consumption have led to fluctuating, and weather-dependent power flows. To ensure grid stability, grid operators rely on power forecasts which are crucial for grid calculations and planning. In this paper, a Multi-Task Learning approach is combined with a Graph Neural Network (GNN) to predict vertical power flows at transformers connecting high and extra-high voltage levels. The proposed method accounts for local differences in power flow characteristics by using an Embedding Multi-Task Learning approach. The use of a Bayesian

* Corresponding author at: Fraunhofer Institute for Energy Economics and Energy System Technology IEE, Joseph-Beuys-Straße 8, Kassel, 34117, Germany.
E-mail addresses: dominik.beinert@iee.fraunhofer.de (D. Beinert), clara.juliane.holzhuetter@iee.fraunhofer.de (C. Holzhüter).

¹ Both authors contributed equally. Dominik Beinert mainly contributed the Multi-Task Approach and Clara Holzhüter mainly contributed the GNN Approach.

<https://doi.org/10.1016/j.egyai.2023.100262>

Received 27 December 2022; Received in revised form 29 March 2023; Accepted 5 April 2023

Available online 10 April 2023

2666-5468/© 2023 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

embedding to capture the latent node characteristics allows to share the weights across all transformers in the subsequent node-invariant GNN while still allowing the individual behavioral patterns of the transformers to be distinguished. At the same time, dependencies between transformers are considered by the GNN architecture which can learn relationships between different transformers and thus take into account that power flows in an electricity network are not independent from each other. The effectiveness of the proposed method is demonstrated through evaluation on two real-world data sets provided by two of four German Transmission System Operators, comprising large portions of the operated German transmission grid. The results show that the proposed Multi-Task Graph Neural Network is a suitable representation learner for electricity networks with a clear advantage provided by the preceding embedding layer. It is able to capture interconnections between correlated transformers and indeed improves the performance in power flow prediction compared to standard Neural Networks. A sign test shows that the proposed model reduces the test RMSE on both data sets compared to the benchmark models significantly.

1. Introduction

The expansion of renewable energies and an increasing number of new consumers, such as electric vehicles, contribute to an increased volatility in power grids. This poses challenges to grid operators because power flows are becoming harder to predict. In the past, power production used to be planned according to the expected consumption needs, but nowadays, it is also largely dependent on local weather and thus heavily fluctuating. At the same time, power consumption is changing due to the growth of e-mobility, batteries, and heat pumps, as well as trends in individual consumption behavior like the increasing occurrence of home office.

1.1. Problem statement

Vertical power flow denotes the power being transferred between two voltage levels in the electric grid measured at a transformer station. For grid operators, they are an essential input to grid calculations, which indicate grid congestion, allowing the operator to initiate countermeasures and thus maintain grid stability. Forecasts of vertical power flows are required for better planning and earlier identification of possible grid congestion.

While separate machine learning models can be trained for each transformer to predict its vertical power flow, the use case poses further challenges. The vertical power flows at transformers are not independent of each other but are interconnected due to the meshed power grid such that a dynamic balancing behavior of power flows can be observed. In contrast to standard Neural Networks, Graph Neural Networks (GNNs) can consider the interconnection between network components and leverage them to make predictions. Therefore, GNNs are particularly suitable for the task at hand.

1.2. Contribution

For accurate predictions of vertical power flows, a global model like a GNN is required to model the relations of power flows at different transformers in the power grid. By interpreting transformers as nodes in a graph, GNNs can take into account dependencies and balancing of power flows through message passing between nodes via the edges.

The main contribution of this paper is the novel combination of a Bayesian Multi-Task Embedding with a GNN architecture for power flow prediction that combines the following three aspects:

- Vertical Power flow at different transformers with individual latent characteristics, such as a specific share of renewable power plants and load characteristic, can be predicted by only one model using an Embedding Multi-Task Learning approach.
- The model generates forecasts of the vertical power flow at transformers taking into account relevant information such as weather conditions at neighboring transformers.
- Changes of how power flows at different transformers influence each other can be handled by an attention mechanism.

The paper aims to examine the proposed method for day-ahead forecasts of vertical power flows on real data sets provided by two German Transmission System Operators (TSO) and evaluate it against a local model benchmark. Here, local refers to a model that makes predictions for each transformer based only on local features of that single transformer, such as the wind speed at its location. This is done by, e.g., standard Neural Networks. The research questions to be answered are:

1. Can the combined architecture based on GNNs and Multi-Task Learning improve the prediction performance of vertical power flow forecasts compared to benchmark models based on only one of these approaches?
2. Can the proposed method learn relations between transformers in a dynamically changing power grid so that it correctly predicts the power flow for nodes that are influenced by the behavior of nearby transformers?

1.3. Related work

Power Flow approximation is required for a secure operation of the power grid in order to avoid blackouts. Security analysis typically relies on load flow solvers. Such methods estimate the power flow of transmission lines considering the physical constraints of the power grid. They are typically probabilistic Monte-Carlo methods using Newton–Raphson optimization to minimize the mismatch between the in- and outgoing power of grid nodes [1]. Since such solvers are computationally expensive, several machine learning-based approaches have been proposed to speed up computations. Duchesne et al. [2], for example, combine Neural Networks with variance reduction techniques to accelerate a Monte Carlo approach. Other Neural Network approaches are presented in, e.g., [3] or [4]. More traditional machine learning approaches have also been applied by, e.g. Yu et al. [5], who use support vector regression to learn the relationships of variables in the power flow equations. In contrast to the approach proposed in this paper, these methods do not take the grid structure and thus the interaction of network components into account.

A similar use case, i.e., vertical power flow prediction, is proposed in [6]. The authors Brauns et al. use a Long Short Term Memory (LSTM) approach to incorporate time series data, which comes with additional computational complexity. However the focus of the approach differs significantly from the approach presented in this paper, since the authors perform daily retraining of their local models to account for dynamic changes in time series characteristics. This is an alternative solution to the problem at hand. When using local models at each transformer, events at other transformers cannot be used directly to adjust model predictions. Instead, the models are retrained on new data which is affected by the mentioned events. In contrast, the approach proposed in this paper is a global model that is able to include events at other locations. Thus, a retraining of a global model is not necessarily required. Further differences in the approach by Brauns et al. are the longer forecast horizon of 48 h and a difference of the data set, which contains power flows between the medium and high voltage grid.

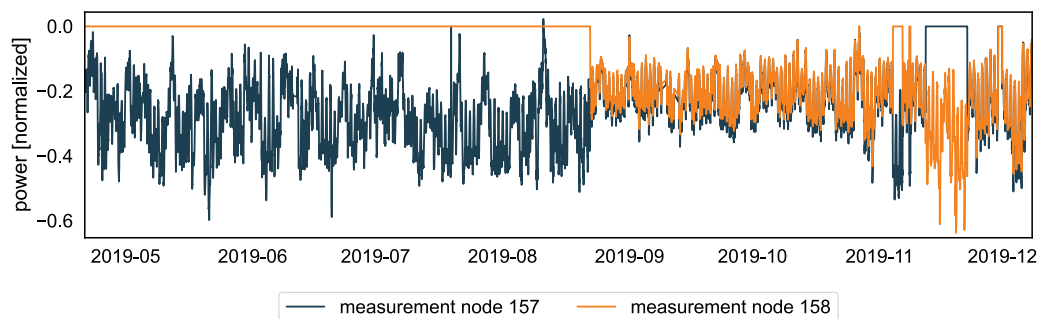


Fig. 1. Vertical power flows of two transformers at one substation with visible interaction after switching node 158 into operation in August 2019. The positive sign denotes power transmitting from the high into the extra-high voltage grid, whereas the negative sign refers to power transmitting from the extra-high into high voltage grid.

To consider that components in the power grid are not independent but highly correlated, several GNN approaches for power flow calculation have been proposed. Some of them focus on specific areas of power forecasting such as wind or solar power prediction [7–10]. Accordingly, these approaches tackle a related use case of renewable energies, but at another level, i.e., they directly estimate the production and not the power flow. General power flow approximation has been approached using GNNs as well. However, only a few works have been proposed so far. Bolz et al. [11], for example, apply a spatial Graph Convolutional Network (GCN) to approximate the loadings of lines based on reactive and active power flow. A similar use case is presented by Wang et al. [12] who use Cheb-Net to learn the distribution characteristics of power flow. Kundacina et al. [13] propose a similar approach that approximates the voltage angle and magnitude using a spectral GNN based on current measurements. All of these methods outperform the compared traditional methods [14]. Compared to our approach, these methods rather focus on the physical aspects and constraints of power flow approximation. Most of them take only technical variables such as reactive and active power flow into account. However, they show the potential of GNNs for such tasks since they are less computationally expensive than traditional methods, such as linear state estimators and solvers, while being more flexible than standard Neural Networks [14]. Unlike the approach presented in this paper, the input data for these approaches consider buses, loads and generators explicitly.

The above mentioned methods are to some extent related to physics inspired methods that use physical constraints such as Kirchhoff's law of the power grid as training loss for a neural network to learn the prediction of power flows. Such methods have been combined with GNNs as well [15–17]. Although GNNs have proven their applicability to the task of power flow estimation, these approaches are not comparable to the proposed approach, since they optimize a completely different target.

For the related use case of grid voltage prediction in the distribution grid, Fusco et al. [18] use a Gaussian graphical model expressed by a GNN, which incorporates domain knowledge as well as a physical constraints. It focuses more on grid congestion and market bidding.

To the best of the authors' knowledge, no other work proposed a Multi-Task GNN approach that embeds latent attributes of individual transformer stations and further processes the input data using a GNN to estimate the corresponding load flow based only on weather, price and consumption forecast and calendar information.

1.4. Outline

The paper is organized as follows: Section 2 provides details on the behavior of power flows in the electric grid. The subsequent Section 3 covers the proposed model consisting of a GNN method combined with task embeddings. In Section 4, the data set and experimental

setup, including model and training parameters, are described. Section 5 presents the obtained results and discusses them. Finally, a short conclusion is given in Section 6.

2. Power flows in the electric grid

The increasing amount of renewable energy sources has drastically changed the power flows in the electric grid. While in the past, most of the power generation took place at the extra-high voltage level, nowadays, much energy is fed into the distribution grid at lower voltage levels. In this section, the particularities of power flows at transformers are discussed as well as the implications for a suitable model.

2.1. Interactions between transformers

The power grid has a meshed structure, meaning that there are redundant lines and, thus, multiple paths for electricity to flow, making the grid more flexible as electricity can be redirected. Due to this meshed structure, the path the power takes when traveling between its point of generation or consumption and the extra-high voltage level can change dynamically, which results in changing characteristics of the vertical power flow. Typical events triggering such changes can be power grid switching or maintenance of single transformers. This behavior can particularly be observed in transformers situated at the same substation. Fig. 1 shows the vertical power flows at two such transformers between the high and extra-high voltage grid. When one transformer is inactive, i.e., its power flow measurements are zero, the other takes over. Consequently, its power flow differs visibly from periods of simultaneous activity.

While this kind of interaction can be seen clearly in time series plots, the balancing behavior of power flows in the electric grid is typically less obvious. Especially in the lower voltage levels, events can be hard to detect and even harder to model, as detailed knowledge about the grid topology is required.

2.2. Transformer-specific behavior

The topology of the distribution grid and the location and attributes of power plants and consumers contribute to the specific power flow characteristics of different transformers. Vertical power flows can be understood as the sum of generated and consumed power in the lower voltage levels. As seen in Fig. 2, the specific mix of production and consumption can be estimated from the power flow time series. While one vertical power flow shows a typical behavior of power consumption, which follows a periodical diurnal cycle with negative consumption peaks during mornings and evenings, the other is strongly influenced by power production, i.e. most of the time power is fed from the high voltage level into the extra-high voltage level, which is denoted by a positive sign. The energy sources contributing to this power flow are not known, but likely include wind power due to the high volatility of the power flow.

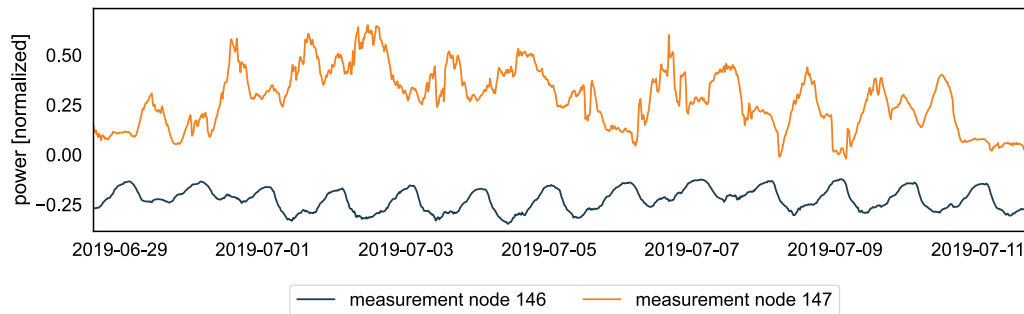


Fig. 2. Vertical power flows with a consumption characteristic at transformer node 146 and a mixed characteristic with mainly power generation at transformer node 147. Both transformers are situated at the same location. The positive sign denotes power transmitting from the high into the extra-high voltage grid, whereas the negative sign refers to power transmitting from the extra-high into high voltage grid.

2.3. Implications for a forecast model

A physical model approach for power flow forecasts is usually unfeasible or limited by strong simplifications, as it requires local modeling of all possible power generation and consumption types at any time, as well as complete knowledge of the transmission and distribution grid topology, including all connected consumers and generators.

For a machine learning approach, this means that it is not sufficient to treat all transformers' behavior and attributes identically. Instead, the individual, local properties of each transformer must be considered, for example, by means of an individual model per transformer. Another option is a multi-task learning model to allow for differences while leveraging similarities.

However, to include the interactive behavior of the electric grid in power flow forecasts, the model needs to consider neighboring node interactions. In this paper, a combination of a multi-task learning model and a GNN is proposed. The multi-task approach accounts for the individual behavior of the transformer, whereas the GNN takes the information of neighboring transformers into account to model the dependencies between transformers. Furthermore, the goal is a mainly data-driven method to be independent of grid topology models as an input. For TSOs, this can pose an advantage, as it does not require other grid operators to provide grid topology and switching state information in lower voltage levels.

In summary, the interactive behavior displayed in Fig. 1 demands for a global GNN model approach where information of neighboring nodes is made available. On the other hand, the different specific time series characteristics as seen in Fig. 2 require the model to distinguish between nodes, which in this paper is achieved by the Bayesian Task Embedding. This additional aspect has to be considered as the graph in this use case does not contain a complete replica of the real electric grid with every single power plant, consumer and power line. Instead, the model must be able to predict different types of time series behavior from local input features. In fact, the two nodes in Fig. 2 are located at exactly the same location at the same substation, which means that the same input values will be provided to the model for both nodes. While it is still possible to consider varying node behavior in a GNN by message passing, i.e., if the nodes have different connections, relying on it will most likely not yield good results. In the example of node 147 in Fig. 2, wind power is generated by near-by wind farms, so the local wind speed will likely be higher correlated to the target time series than the wind speeds at other nodes that are available due to message passing. By integrating a Bayesian Task Embedding approach, the model is allowed to use local input variables differently depending on the specific node.

3. Graph neural networks with multi-task embedding

The method proposed in this paper combines a GNN with a Bayesian Embedding Multi-Task Learning approach, enabling the model to learn

both interactions between nodes and node-specific time series characteristics. Given inputs $X_t \in \mathbb{R}^{d \times n}$ with d features at n nodes for a specific timestamp t , the model will predict the vertical power flows $y_t \in \mathbb{R}^{1 \times n}$. For training, a data set $(\mathbf{X}, \mathbf{y}) = (X_t, y_t)_{t \in \mathcal{T}}$ is provided, where \mathcal{T} denotes the given time range of training data.

3.1. Graph Convolutional Networks (GCNs)

Compared to standard Neural Networks, the usage of GNNs to predict power flow at nodes in the electricity network has a theoretical advantage: the information exchange between connected nodes. Note that the mentioned information exchange is invariant of the absolute node position in the graph. For standard Neural Networks, the input samples are treated as independent samples. For the given use case and data set, this means that the transformers would be assumed to be independent of each other. However, this is not the case since the power flow depends on several features that can be correlated across different transformers. On one hand, features such as wind speed or solar irradiance are locally correlated, i.e., transformers at close locations will have a similar weather forecast. On the other hand, the transformers are directly correlated in the sense that they show compensatory behavior, e.g., if a transformer close by is inactive due to maintenance. GNNs are designed for such data and enable the propagation of information through the graph via the edges. Therefore, they can model relationships between nodes in the graph.

More specifically, GNNs operate on graphs $G = (E, V)$ where E denotes the edges and V the nodes. The attributes of node v are represented as a feature vector. Accordingly, the set of all nodes is represented as a feature matrix of shape $d \times n$ where n is the number of nodes, and d is the dimension of the features. The edges are represented as an adjacency matrix A of shape $n \times n$ in which $A_{u,v} = 1$ if node u and v are connected and $A_{u,v} = 0$ otherwise. In the case of a weighted graph, $A_{u,v}$ is set to the weight associated with the edge between node u and v .

Due to the great success of Convolutional Neural Networks (CNNs), Graph Convolutional Networks (GCNs) have been developed to generalize convolutions to graph data. In the proposed approach, spatial graph convolutions are used, which resemble the standard convolution such that in each layer, each node v aggregates the features of its neighbors, i.e., adjacent nodes. Similar to standard CNNs in which filters are learned, GCNs aggregate neighboring features using learnable weight matrices as well. Since each node aggregates information from direct neighbors in each layer, the receptive field size of the network increases by one with each layer [19]. This way, the GNN can learn a high-level node representations which cover the interaction with a larger portion of the underlying electrical grid.

The following exemplary basic spatial graph convolution as defined in, e.g., [20] is given by:

$$h_u^{(l+1)} = \sigma(W_1 h_u^{(l)} + W_2 \sum_{v \in \mathcal{N}(u)} h_v^{(l)}). \quad (1)$$

$h_u^{(l)} \in \mathbb{R}^{d^{(l)}}$ denotes the d -dimensional node representation of node u in layer l and $\mathcal{N}(u)$ describes the neighborhood of node u . W_1 and W_2 are shared learnable weight matrices and σ is an activation function such as ReLU. In the first layer, h_u is the input features x_u of node u . In this graph convolution, every neighbor is weighted equally in the neighborhood aggregation. Using the adjacency matrix and notation $H^{(l)} = (h_1^{(l)}, \dots, h_n^{(l)})$, Eq. (1) can be written in matrix form as multiplication with the adjacency matrix A described above:

$$H^{(l+1)} = \sigma(W_1 H^{(l)} + W_2 H^{(l)} A) \quad (2)$$

Note that A does not contain self-loops which means that the diagonal is set to zero. The features of the target node itself are transformed using matrix W_1 . To learn individual weights for each neighbor, basic graph convolutions can be equipped with attention mechanisms as presented in [21]. The new node representation for a node i changes Eq. (1) as follows:

$$h_u^{(l+1)} = \sigma(W_1 h_u^{(l)} + \sum_{v \in \mathcal{N}(u)} \alpha_{u,v} W_2 h_v^{(l)}). \quad (3)$$

In contrast to the standard convolution from Eq. (1), every neighbor v is weighted by attention coefficient $\alpha_{u,v}$, which is computed individually for each pair of nodes. The coefficient, which represents the importance of one node to another, is computed as follows:

$$\alpha_{u,v} = \text{softmax} \left(\frac{(W_3 h_u)^T W_4 h_v}{\sqrt{d_l}} \right) \quad (4)$$

where d_l is the number of outputs for the hidden layer l , i.e., the hidden dimension and $\sum_{v \in \mathcal{N}(u)} \alpha_{u,v} = 1$. Although α is usually different for each node and neighbor, the weights used to compute α are shared across all nodes. To include edge features in this attention mechanism, the equation changes as follows:

$$h_u^{(l+1)} = \sigma(W_1 h_u^{(l)} + \sum_{v \in \mathcal{N}(u)} \alpha_{u,v} (W_2 h_v^{(l)} + W_5 e_{u,v})). \quad (5)$$

$e_{u,v}$ denotes the weight associated with the edge between node u and v and W_5 corresponds to a learnable weights matrix also used to compute α with edge weights included:

$$\alpha_{u,v} = \text{softmax} \left(\frac{(W_3 h_u)^T (W_4 h_v + W_5 e_{u,v})}{\sqrt{d_l}} \right). \quad (6)$$

To learn different relations between nodes in one layer, multi-head attention can be used. For this purpose, the attention mechanism is applied multiple times, each learning its individual weights. The resulting feature maps, called attention heads, are then concatenated afterwards [21]. The concept is similar to producing multiple feature maps in standard CNNs. Stacking multiple convolutional layers builds a GCN that outputs a graph with new node representations that can be used to perform specific prediction tasks such as node regression or classification. For this purpose, the node embeddings can be used directly by applying a suitable activation function, or they are further transformed by a suitable readout network. Typically, such a network consists of a set of fully connected layers that transform each node separately.

3.2. Embedding multi-task learning

Multi-Task Learning (MTL) refers to a machine learning setting in which a model is trained to solve multiple similar problems, also referred to as tasks. The goal is to use the combined knowledge of all tasks during training such that all tasks benefit from each other. In contrast to GNNs, an MTL model does not exchange current information between tasks when making predictions.

One possible way to share knowledge during training is the concept of hard parameter sharing, where a certain part of model parameters is shared between all tasks while other parameters are trained specifically

for each task. In the Bayesian Embedding Multi-Task Learning Multi-Layer Perceptron (BEMTL) model as proposed in [22], all weights in an Artificial Neural Network (ANN) are shared, and only a Bayesian task embedding is used to differentiate between tasks. A task embedding can be understood as a vector encoding a specific task by placing it in a low-dimensional embedding space. It is given into the network's first hidden layer concatenated to the task's input variables and can be trained similarly to ANN weights by back-propagation. A Bayesian approach enforces a stable embedding training that maps similar or even indistinguishable tasks close to each other in the embedding space. Thus, instead of specific vectors $w_u \in \mathbb{R}^m$ in the embedding space, multidimensional probability distributions $q(w_u)$ are used to represent tasks $u \in \{1, \dots, n\}$. To simplify calculations, independent multivariate normal distributions are chosen, and an identical, independent prior normal distribution $p(w)$ is introduced as regularization for all tasks. The posterior probability distribution can be trained by using *Bayes By Backprop* as described in [23] and used in [22]. The results are independent multivariate normal distributions $q(w_u | \mu_u, \sigma_u) = \mathcal{N}(\mu_u, \text{diag}(\sigma_u^2))$ with $\mu_u, \sigma_u \in \mathbb{R}^m$ for tasks $u \in \{1, \dots, n\}$. With $\mathbf{w} = (w_1, \dots, w_n)$, the probability distribution parameters $\boldsymbol{\mu} = (\mu_1, \dots, \mu_n)$ and $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_n)$ minimize the cost function

$$\begin{aligned} \arg \min_{\boldsymbol{\mu}, \boldsymbol{\sigma}} \mathcal{F}(\mathbf{X}, \mathbf{y}, \boldsymbol{\mu}, \boldsymbol{\sigma}) \\ = \arg \min_{\boldsymbol{\mu}, \boldsymbol{\sigma}} \{ KL[q(\mathbf{w} | \boldsymbol{\mu}, \boldsymbol{\sigma}) \parallel p(\mathbf{w})] - \mathbb{E}_{q(\mathbf{w} | \boldsymbol{\mu}, \boldsymbol{\sigma})} [\ln p(\mathbf{y} | \mathbf{X}, \mathbf{w})] \}, \end{aligned} \quad (7)$$

where the first term is the Kullback–Leibler (KL) divergence between the approximation $q(\mathbf{w} | \boldsymbol{\mu}, \boldsymbol{\sigma})$ and the chosen prior $p(\mathbf{w})$, and the second term represents the negative log-likelihood cost function. The KL divergence is a measure of the difference between two probability distributions. Thus, it serves as a penalty for diverging too far from the given prior and can, in practice, be weighted by a hyperparameter λ_{bayes} for appropriate regularization.

3.3. Proposed architecture

In the given use case, transformers are represented by nodes in a graph. Since different transformers display different characteristic time series behavior, as explained in Section 2, they can be seen as different but also similar tasks in a Multi-Task Learning setting. A standard GNN, which can be interpreted as a particular type of hard parameter-sharing ANN, cannot distinguish between the graph nodes. Therefore, a Bayesian task embedding is used as an additional node input into the GNN by introducing two embedding layers for $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. The proposed architecture BEMTL-GNN with the novel combination of GNN with a Bayesian task embedding for node distinction is shown in Fig. 3. For n nodes and d input features, X_t is a $d \times n$ matrix containing inputs for one timestamp, while $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are $m \times n$ matrices with m being the dimension of the embedding space. With $w_u \sim \mathcal{N}(\mu_u, \text{diag}(\sigma_u^2))$ a sample for node u can be chosen from the embedding probability distribution, and $\mathbf{w} = (w_1, \dots, w_n)$ for nodes $\{1, \dots, n\}$. It should be noted that samples are drawn from the distribution only during the training process and $\mathbf{w} = \boldsymbol{\mu}$ when making predictions. The encoded task representations \mathbf{w} from embedding space are then concatenated with X_t to create the initial $(d+m) \times n$ node representations $H^{(0)} = (h_1^{(0)}, \dots, h_n^{(0)})$.

These representations are passed to the GNN, which applies the attention convolution defined in Eq. (3). As a readout function the convolutional layers are followed by a set of fully connected layers to produce the final $1 \times n$ output y_t , i.e., one value for each transformer for the given timestamp t .

4. Experiments

An experiment was carried out to test the approach on a real-world data set and evaluate it against models without GNN architectures and a standard GNN without the embedding layer of the proposed model.

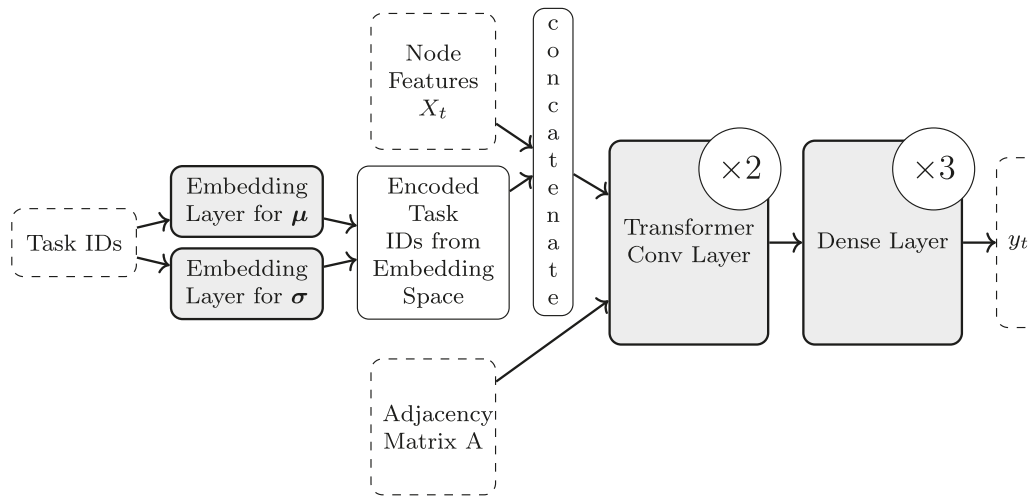


Fig. 3. Proposed model architecture BEMTL-GNN.

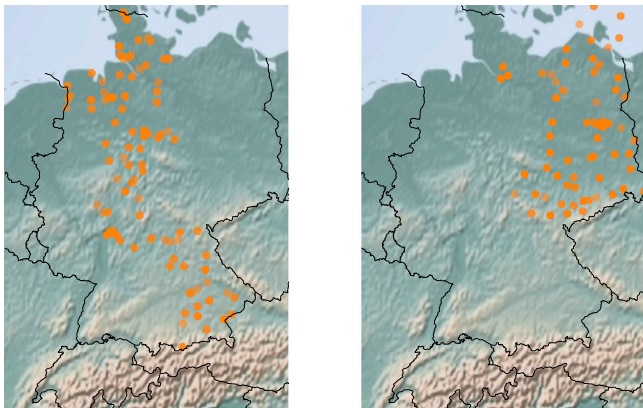


Fig. 4. Location of transformers for TSO 1 TenneT (left) and TSO 2 50 Hz (right). Multiple transformers at the same location are possible.

4.1. Data set

This study has been conducted on two real-world data sets containing vertical power flow measurements at transformers of the German Transmission grid, respectively. The overall problem size considered in the data set corresponding to TSO1 (TenneT) includes 176 transformers with 37 features each and 175 transformers with 28 features for TSO2 (50 Hz). Together, the data sets cover about half of the transmission grid in Germany which make them a real-world use case at scale of the real system. The transformers connect the high voltage distribution grid to the extra-high voltage transmission grid. Additionally, the data set includes metadata comprising coordinates for each transformer. Apart from that, no exact information about the actual grid topology is available for this experiment; nonetheless, it can be assumed that the existence of an overhead line correlates strongly with geographic proximity. Fig. 4 shows the location of the given transformers for both data sets. The measurement time series are given in a 15 min resolution with available data from Jan 5, 2018 to Dec 31, 2019 for the TenneT data set and from Jan 2, 2020 to Feb 2, 2022 for 50 Hz. Power flows from the distribution grid to the transmission grid are interpreted as positive values and power flows from the transmission to the distribution grid as negative values. The measurements are normalized by each transformer's estimated capacity rating, receiving target values between -1 and 1 . Furthermore, gaps in the measurements with a duration of up to one hour are interpolated linearly.

As local inputs at each node, a mix of numerical weather predictions (NWP) and other features are used to include explanatory variables for each possible type of power generation and consumption. Table 1 shows the input features used in both data sets. As NWP inputs, the IFS weather model by ECMWF [24] with day-ahead forecast horizons of $+24$ to $+48$ h and at transformer locations is used. All input features get standardized before training.

4.2. Experimental setup

In this section the experimental setup is discussed. First, the graph construction is explained, followed by an introduction of benchmark models that are compared to the novel approach. Lastly, the choice of model parameters as well as the process of hyperparameter tuning are elaborated.

4.2.1. Graph construction

In this use case, the graph represents the power grid, in which the nodes correspond to the individual transformers between high and extra-high voltage levels. The edges that connect the transformers are defined by the distance between them, since the true wiring is not given in the data set. Transmission lines, bus bars or other grid components are not considered in the graph representation. Using a graph in which each node is connected to every other node would increase the computational complexity significantly. At the same time, the mutual influence of two transformers is expected to decrease with larger geographical distances, such that edges between transformers that are far apart from each other are likely less beneficial for the prediction performance. To affirm this hypothesis and test the effect of edge density in the graph, the experiment is performed on two differently constructed graphs.

In the first step, the benefit of the proposed approach shall be shown in a proof of concept. For this purpose, a very sparsely connected graph with edges only between transformers situated at the same substation is chosen. Accordingly, the distance between that are connected by edges is not allowed to exceed 0 km. This experiment is performed on both data sets resulting in 102 edges for TSO1 and 171 edges for TSO2.

In the second step, an experiment is run on a more densely connected graph for data set 1 with edges between all transformers with a maximum geographical distance of 50 km. This results in a graph with 769 edges. No topological grid information is used to define edges. Additionally, for the more densely connected graph, the edge weights are set to the exponential inverse of the distance between the two transformers connected by an edge.

Table 1
Input features.

	TSO 1	TSO 2
Wind speed at 10m and 100m, with ± 1 h time lag	x	x
Wind direction at 100 m	x	x
Temperature at 2m level	x	x
Dew point temperature at 2m level	x	x
Air pressure at surface level	x	x
Global horizontal irradiation with ± 1 h time lag	x	x
Precipitation as a moving mean, as well as accumulated precipitation of the last 1, 2, 3 and 4 days	x	
Solar height and solar azimuth	x	x
A night/day logical with value 1 from 10 pm to 6 am and 0 otherwise to accommodate noise protection regulations	x	x
Calendar information (hour of the day, day of the week, day of the year; logicals for holidays, sundays and saturdays)	x	x
Consumption day-ahead forecasts for Germany and all four German TSO regions [25]	x	x
	x	
Price day-ahead forecasts for Germany [26]	x	x
Active/inactive logical with value 1 for a detected active transformer state and 0 for inactive state. Inactivity is assumed for five or more consecutive zeros or missing values in the vertical power flow measurements	x	x

Table 2
Parameters of the compared models.

	BEMTL-GNN	BEMTL	St. GNN	STL
Embedding	$d = 8, \lambda = 1e - 10$	$d = 8, \lambda = 1e - 10$	–	–
Conv. Layers	2	–	2	–
Lin. Layers	3	5	3	5
Neurons per layer	100			
Activation function	ReLU			
Batch size	128			
L2-regularization	$1e - 8$			
Learning rate	0.001			

4.2.2. Benchmark models

Four different models are compared in total, of which two models are GCNs. The other two models serve as benchmark models. The first benchmark model is a Bayesian Embedding Multi-Task Learning approach (BEMTL) as described in Section 3.2, in which a set of standard fully connected layers is applied to the input features of the transformers after the Bayesian embedding layer. This model is equivalent to the proposed BEMTL-GNN model applied to the transformer graph without considering the edges. Accordingly, no information is exchanged across nodes. The other non-GNN benchmark model is a single-task model, which trains a fully connected Neural Network (STL ANN) for each transformer separately. Hence, each model learns individual weights optimized for the corresponding transformer. Both GNN models apply the convolution from Eq. (3) with one attention head per layer. The standard GNN, which serves as a benchmark, does not include a Bayesian embedding layer. Accordingly, the model allows information exchange between different transformers, including individual attention coefficients, but it cannot distinguish between different transformers. The proposed architecture BEMTL-GNN includes both multi-task embedding and information exchange across nodes. Although a temporal approach might be beneficial for the predictive performance, a time series-based approach was not included as benchmark, since it comes with an additional increase of complexity. Instead, this paper focuses on the aspect of combining MTL Learning with GNN for power flow prediction. To analyze the impact of this approach, a temporal model is not needed. However, for future work and further model improvement, both the benchmark and the BEMTL-GNN model can be expanded by temporal layers.

4.2.3. Model and training parameters

The exact parameters of all models, such as the number of layers and neurons, are shown in Table 2. All non-GNN architectures and the model training are implemented using pytorch [27]. All graph-related methods, including the GNN architectures, and the graph data handling are implemented using PyTorch Geometric [28], which provides the applied GNN layers.

Preceding the experiment, a small range of hand-picked hyperparameters were investigated. These include different network depths and numbers of attention heads (1–3 layers/heads), and several different l2-regularization values to fine-tune the model broadly. As part of the hyperparameter tuning, two different architectures of GCNs were applied, one based on the layer defined in Eq. (1) and the other one using the attention-based convolution from [21] defined in Eq. (3). Results showed an advantage of the attention-based convolution used for further experiments. Other aspects that were considered in a small study comprise the embedding regularization parameter λ_{bayes} , the convolutional layers' aggregation function, and the order of convolutional and standard dense layers. Of course, the hyperparameter optimization has been performed exclusively on the training set, defined as described below.

The data described in Section 4.1 was split into a training and test set containing data of individually consecutive periods. Furthermore, a 2-month period at the end of the training period is used as a validation set for hyperparameter tuning. Table 3 shows the exact data split for both data sets.

Using consecutive periods for each split is necessary to avoid including data points from inside the test period in the training set. Sampling randomly from the entire period would lead to a higher correlation between the sets, which would reduce the reliability of the results on the test set. Additionally, the power production is highly dependent on the weather and, thus, on the seasons. Therefore results are most representative when an entire year can be included in the test set.

In the experiments, all models were trained over 20 epochs using the Adam optimizer. By tracking the validation error while training, the epoch with lowest validation error was identified and the corresponding model weights chosen as final model. As a loss function, the mean squared error has been used. For all models which include the Bayesian embedding, an additional loss term was introduced to measure the quality of the embeddings. For this purpose, the KL loss has been computed as described in Eq. (7) and weighted by hyperparameter λ_{bayes} . The exact training parameters, such as learning rate, are shown in Table 2.

To evaluate the performance of each model, the root mean squared error (RMSE) on the test set averaged over five runs was compared.

Table 3
Split of training, validation and test periods.

	Training	Validation	Test
TSO 1	5 Jan–31 Dec 2018	1 Jan–28 Feb 2019	1 Mar–31 Dec 2019
TSO 2	2 Jan–31 Dec 2020	1 Jan–28 Feb 2021	1 Mar 2021–7 Feb 2022

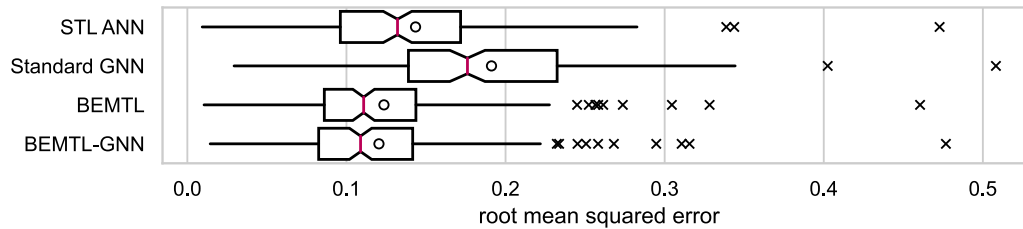


Fig. 5. Test RMSE of normalized predictions on TSO 1 data set for benchmark models STL ANN, Standard GNN and BEMTL and proposed model BEMTL-GNN. One box represents 176 error samples, where each sample is the test error for one transformer averaged over 5 runs. The red line indicates the median over all samples and the black circle indicates the mean.

5. Results and discussion

In this section, the proposed approach is evaluated and compared to benchmark models described in Section 4.2. For evaluation the models' test RMSE on all transformers is compared and significance of observed RMSE differences between models are asserted by a sign test. As error values alone do not give any insight in how the model uses the additional information provided by the graph, forecast time series of the proposed model and the strongest benchmark are compared and discussed. The results for the sparsely and densely connected graph setup are examined separately.

5.1. Sparsely connected graph

For each method, five experimental runs were performed to reduce random effects in the evaluation. Thus, in each run and for each method one RMSE value per transformer is calculated on the test set. By averaging over the five runs, for each method a mean RMSE per transformer is obtained. Fig. 5 shows those averaged test RMSEs of data set 1 as boxplots, i.e. each method's boxplot depicts 176 values. The results show that BEMTL-GNN achieves the lowest median and mean RMSE but only with a slight advantage compared to the pure Multi-Task Model BEMTL. Noticeably, the standard GNN performs poorly with the highest average and median RMSE, indicating that in the given use case it is essential for the model to be able to distinguish between nodes. These results correspond to the authors' expectations that the differing behavior of the transformers requires a multi-task approach that captures individual characteristics.

A comparison between the two non-GNN models, STL ANN and BEMTL, points towards an advantage of the BEMTL model. However, note that hyperparameters were tuned only for BEMTL-GNN. This hyperparameter tuning translates better to a Standard GNN and the Multi-Task model BEMTL since those models are trained on the same amount of training data given by all transformers, while STL ANNs are trained separately for each transformer and thus are provided with fewer training samples. Accordingly, a lower error might be achieved by STL ANN when performing a separate hyperparameter optimization. However, in an experiment on wind power forecasting, [22] showed that BEMTL can match or even outperform STL ANNs with individual hyperparameter optimization for each STL model, such that focusing on BEMTL as the strongest benchmark seems justified.

The results show that test errors vary widely across different transformers. Thus, a better comparison of models can be made by looking at RMSE differences at each transformer. In Fig. 6 the test error differences between BEMTL and BEMTL-GNN are shown, with both median and mean being greater than zero, indicating that BEMTL-GNN outperforms BEMTL on the majority of transformers. To be exact, on data set 1,

the proposed approach achieves lower RMSEs at 130 of 176 (74%) of transformers. A sign test performed on these results testifies that the null hypothesis that BEMTL achieves lower errors than BEMTL-GNN can be dismissed with a p -value of $8.88e-11$. This attests that the result of the proposed model being more likely to outperform the benchmark is significant. Furthermore, the median is greater than zero with a confidence interval of 95%, as indicated by the notches. This shows that the proposed model achieves higher performance on a large fraction of the transformers.

The results on data set 2 are less clear but point in the same direction, with BEMTL-GNN being the better choice of model for 109 of 175 (62%) of transformers. Here, the null hypothesis that BEMTL achieves lower errors than BEMTL-GNN can be dismissed with a p -value of $7.14e-4$.

However, evaluating error values does not give any insight into the model's ability to learn relations between transformers. For this purpose, forecasts of BEMTL and BEMTL-GNN at example transformers situated at the same substation are compared. As can be seen in Fig. 7, forecasts by both models accurately predict zero values during times of inactivity, but only BEMTL-GNN forecasts show a clear impact of one transformer's inactivity on the other. It can be observed that BEMTL-GNN indeed learns the relation that these two transformers strongly influence each other. This shows that the proposed BEMTL-GNN model has successfully leveraged the ability of GNNs to capture such relationships. Since the BEMTL model lacks information flow between transformers, such effects cannot be explored, resulting in the model ignoring the inactivity of correlated transformers. In this example, node 158 was inactive for 4 months in the training set, while node 157 only had short periods of inactivity. Thus, BEMTL forecasts of node 157 are biased towards behavior that assumes the inactivity of node 158. The predictions of BEMTL-GNN appear to be better than the ones of BEMTL, even when both transformers are active. This could be due to the fact that BEMTL has to average over the possible activity states to some extent to make good predictions during times of activity and inactivity of partner nodes.

While this example portrays the positive effect of the convolutional layers provided by the GNN, the advantage of including the Bayesian Embedding approach in the model is best illustrated by Fig. 8. Here the center plot shows forecasts of the benchmark Standard GNN at two transformer nodes located at the same substation. Being situated at the same location, input values and edges are identical for both nodes. Therefore, the standard GNN model without a Multi-Task Embedding produces the same output for both transformers. As it can be observed from the true measurements, the forecasts of the standard GNN are not a good match for either node. However, the proposed BEMTL-GNN model shown at the bottom of the figure is able to make two different predictions using the same graph and input features, which results in

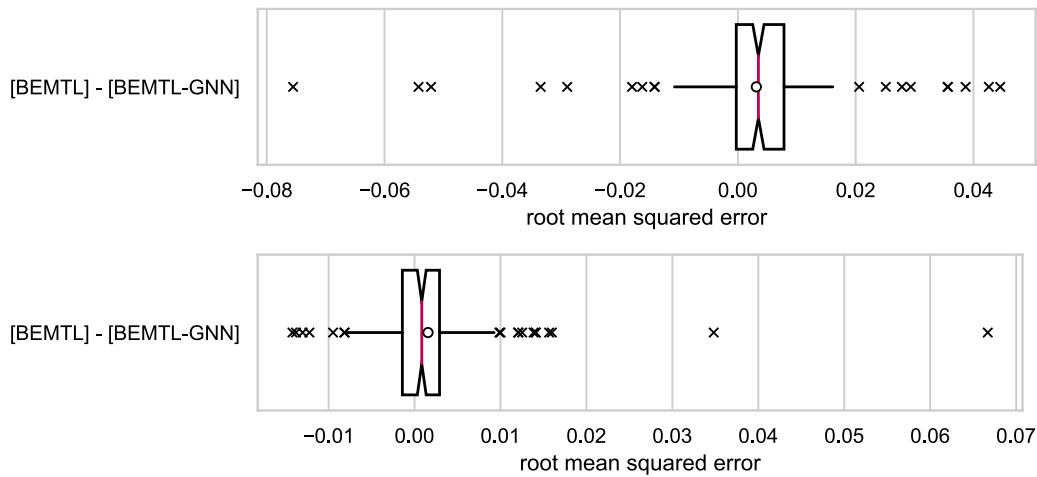


Fig. 6. Top: TSO 1 data set, bottom: TSO 2 data set. Test RMSE differences between normalized predictions for benchmark model BEMTL and proposed model BEMTL-GNN. One box represents 176 and 175 error samples, respectively, where each sample is the test error difference for one transformer averaged over 5 runs. Note that axes are not scaled and clipped equally.

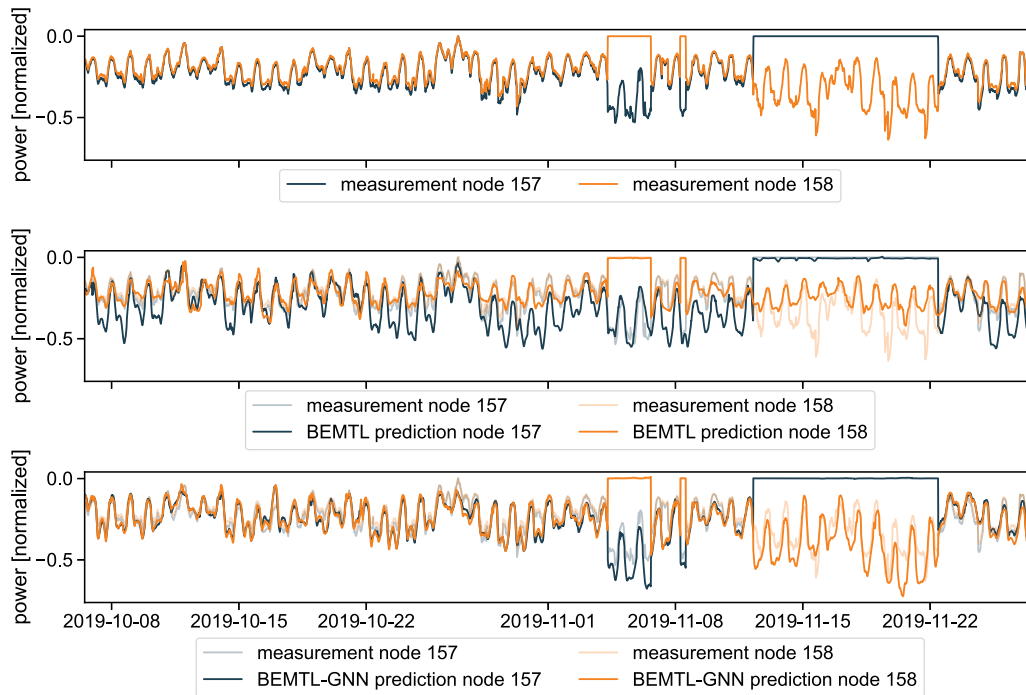


Fig. 7. Vertical power flow predictions during test period by benchmark BEMTL (center plot) and proposed model BEMTL-GNN (bottom plot) at two transformers by TSO 1.

more accurate power flow forecasts for both nodes. This is due to the task embeddings that were optimized during the training process and resulted in dissimilar embedding values for both nodes, providing the necessary variance of input. Consequently, including the Bayesian task embedding is a valuable addition to the model in this use case.

5.2. Densely connected graph

The second experiment was performed on a more densely connected graph in which two nodes are connected if their distance does not exceed 50 km. Again, five experimental runs were performed and the error values averaged as described in 5.1. The results show a reduction of the test RMSE compared to the benchmark BEMTL, as can be observed in Fig. 9. The plot shows the error differences between BEMTL and BEMTL-GNN operating on the dense graph. As before on the sparse graph, BEMTL-GNN performed better on 130 of 176 (74%)

transformers. However, in this experiment, forecasts by BEMTL-GNN do not clearly show the expected balancing behavior that the model was able to learn on the sparse graph. Since closer transformers show more adaptive behavior towards each other, a more densely connected graph probably includes too many factors in the neighborhood aggregation of the graph convolution. Hence, the adaptive behavior is harder to recognize for the model. To be able to predict such relationships also on densely connected graphs, further research on edge attributes or attention mechanisms could be of interest.

5.3. General remarks on efficiency and scale

Regarding efficiency, the following observations can be made: Firstly, STL has by far the most trainable parameters since it includes one model per transformer, whereas the other approaches learn only one model in total. With approximately 62,000 trainable parameters,

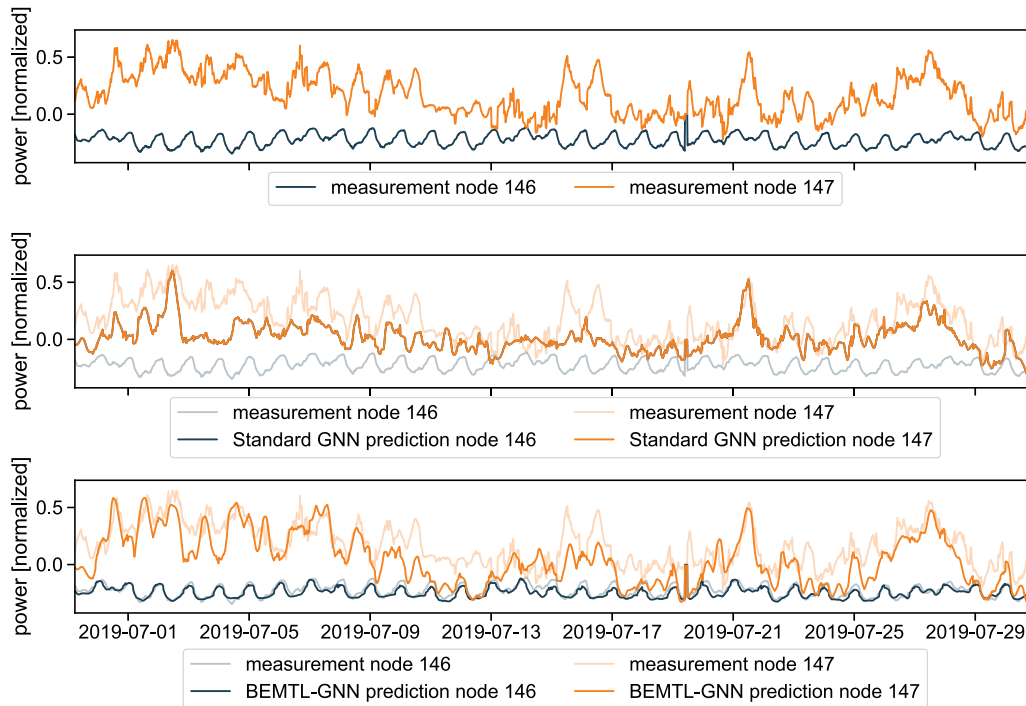


Fig. 8. Vertical power flow predictions during test period by the benchmark Standard GNN (center plot) and proposed model BEMTL-GNN (bottom plot) at two transformers at the same substation by TSO 1. Note that in the center plot, the blue line of Standard GNN prediction for node 146 is overlaid by the orange line for node 147. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

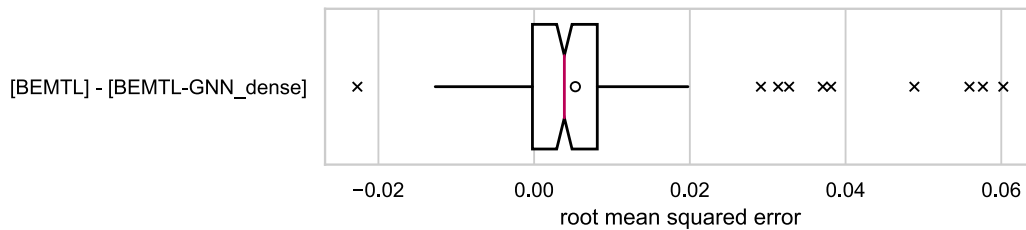


Fig. 9. Test RMSE differences between normalized predictions for benchmark model BEMTL and model BEMTL-GNN operating on a densely connected graph on TSO 1 data set. The box represents 176 error samples, each sample being the test error difference for one transformer averaged over 5 runs.

BEMTL is the least complex model, followed by the standard GNN, and BEMTL-GNN with approximately 86,000 and 92,000 parameters, respectively. Each STL model contains about 24,000 parameters multiplied by the number of nodes in the graph. Accordingly, the model in total is about 125 times larger than the BEMTL model.

Regarding the inference and train times, one STL model is roughly three times faster than the BEMTL model, which, in turn, requires about half the inference and train time of the GNN models. Conclusively, the superior performance of the proposed BEMTL-GNN model comes with an increased training and inference time compared to the STL and BEMTL models. The reason for this is the higher number of parameters, which reside mainly from the attention convolutions.

The inference time of the proposed model increases linearly with the number of nodes since for each node a separate node representation has to be computed. However, the size of the weight matrices in the proposed model does not depend on the number of nodes, but only on the number of node features. A more important factor regarding the runtime is connectivity of the input graph. The reason is that for the computation of a node embedding, the embeddings of all neighbors of that node are involved. Accordingly, the more edges are present in the graph, the more computationally expensive is the GNN layer computation. This holds especially for the applied attention convolution, since it computes an attention score for each edge (c.f. Eq. (4))

For the experiments on the densely connected graph including 769 edges, which is more than 7 times more edges than in the sparsely connected graph, the inference time increases to 1.75 of the sparse graph. When considering graphs with edges based on 100 kms (2400 edges) and 150 kms (4100 edges) distance between two transformers, an increase of complexity to approximately 2.8 and 3.7 compared to the sparse graph can be observed. Accordingly, in this range of connectivity, the inference time increases linearly with the number of edges. Since a higher connectivity, i.e. more edges, did not enhance the performance in the experiments, a larger distance than 150 is not suitable for the conducted experiments.

6. Conclusion and outlook

This paper proposes BEMTL-GNN, a GNN with a node-specific embedding layer and an attention mechanism to forecast vertical power flows at transformers. The model successfully learns individual latent characteristics of the transformers, while also taking into account information from close-by transformers. It has been shown that a preceding embedding layer enhances the per-node predictions of the applied GNN significantly, since it can capture the characteristic behavior of the individual transformers. Compared to an equivalent multi-task network without a GNN architecture, the novel approach indeed improves the

prediction performance of vertical power flow forecasts. More specifically, the proposed model reduces the test loss at 74% of transformers from a real-world data set. This includes situations in which transformers are directly influenced by nearby transformers. A sign test asserts the significance of the result that BEMTL-GNN is more likely to outperform the pure multi-task model. However, a visible balancing relationship between forecasts at different transformers can only be retraced in time series plots when the model is trained on a sparse graph with few edges. In summary, the proposed method delivers a clear advantage by providing both the ability to use message passing of GNN layers to include transformer interactions in forecasts, and the ability to distinguish between nodes to allow a characteristic usage of input features at different transformers. This is achieved by the novel combination of GNN layers and a Bayesian task embedding. On the other hand, the approach does not provide a solution for defining relevant graph edges yet. Further research is also necessary to extend the desired behavior seen in the experiment with a sparse graph to densely connected graphs.

To enable the model to learn the more complex relations of several transformers in a neighborhood, more than one attention head might be necessary. Further research could also focus on reducing irrelevant edges, as a large number of edges between transformers with little impact on each other hampers model training. For this, a two-step approach might be helpful, where the existence of edges is learned in the first step before applying the model proposed in this paper in the second step. Another approach could address the problem of unbalanced data sets. Generally, periods in the training set from which specific relations between nodes could be learned are relatively short. This is because some switching states of the grid occur in exceptional situations like maintenance or special weather conditions only. Thus, data augmentation with a more frequent appearance of different grid situations could improve model training but requires detection and distinction of grid states in the first place, which is not trivial. Lastly, including real grid topology information where possible might help the model to learn unknown relations.

While the above suggestions for improvement can pave the way for a more exhaustive, data-driven graph representation of the power grid, including more system components and their interactions, the proposed method can already enhance vertical power flow forecasts without requiring additional information about grid topology. This is a valuable step for grid operators towards a more accurate grid calculation for increasingly fluctuating electricity networks.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data

Acknowledgments

This publication was supported by the Hessian Ministry of Higher Education, Research, Science and the Arts, Germany, through the Competence Center for Cognitive Energy Systems of the Fraunhofer IEE with reference number 511/17.001. Additionally, this work was supported by TRANSFER (grant number 01IS20020A) funded by the German Federal Ministry of Education and Research. Further, the authors would like to thank their partners TenneT TSO GmbH and 50 Hz Transmission GmbH for their support, as well as the GAIN research group at University Kassel funded by the German Federal Ministry of Education and Research, under the funding code 01IS20047A.

References

- [1] Donon B, Donnot B, Guyon I, Marot A. Graph neural solver for power systems. In: 2019 International joint conference on neural networks (IJCNN). 2019, p. 1–8. <http://dx.doi.org/10.1109/IJCNN.2019.8851855>, ISSN: 2161-4407.
- [2] Duchesne L, Karangelos E, Wehenkel L. Using machine learning to enable probabilistic reliability assessment in operation planning. In: 2018 Power systems computation conference (PSCC). 2018, p. 1–8. <http://dx.doi.org/10.23919/PSCC.2018.8442566>.
- [3] Cai M, Chen R, Kong L. Hyper-Chaotic neural network based on Newton iterative method and its application in solving load flow equations of power system. In: 2009 International conference on measuring technology and mechatronics automation, Vol. 3. 2009, p. 226–9. <http://dx.doi.org/10.1109/ICMTMA.2009.531>, ISSN: 2157-1481.
- [4] Karami A, Mohammadi MS. Radial basis function neural network for power system load-flow. *Int J Electr Power Energy Syst* 2008;30(1):60–6. <http://dx.doi.org/10.1016/j.ijepes.2007.10.004>, URL <https://www.sciencedirect.com/science/article/pii/S0142061507001135>.
- [5] Yu J, Weng Y, Rajagopal R. Robust mapping rule estimation for power flow analysis in distribution grids. In: 2017 North American power symposium (NAPS). Morgantown, WV: IEEE; 2017, p. 1–6. <http://dx.doi.org/10.1109/NAPS.2017.8107397>, URL <http://ieeexplore.ieee.org/document/8107397/>.
- [6] Brauns K, Scholz C, Schultz A, Baier A, Jost D. Vertical power flow forecast with LSTMs using regular training update strategies. *Energy AI* 2022;8:100143. <http://dx.doi.org/10.1016/j.egyai.2022.100143>, URL <https://www.sciencedirect.com/science/article/pii/S2666546822000064>.
- [7] Bai W, Zhu X, Lee KY. Dynamic optimal power flow based on a spatio-temporal wind speed forecast model. In: 2021 IEEE Congress on evolutionary computation (CEC). 2021, p. 136–43. <http://dx.doi.org/10.1109/CEC45853.2021.9504847>.
- [8] Khodayar M, Wang J. Spatio-Temporal graph deep neural network for short-term wind speed forecasting. *IEEE Trans Sustain Energy* 2019;10(2):670–81. <http://dx.doi.org/10.1109/TSTE.2018.2844102>, Conference Name: IEEE Transactions on Sustainable Energy.
- [9] Park J, Park J. Physics-induced graph neural network: An application to wind-farm power estimation. *Energy* 2019;187:115883. <http://dx.doi.org/10.1016/j.energy.2019.115883>, URL <https://www.sciencedirect.com/science/article/pii/S0360544219315555>.
- [10] Chen R, Liu J, Wang F, Ren H, Zhen Z. Graph neural network-based wind farm cluster speed prediction. In: 2020 IEEE 3rd student conference on electrical machines and systems (SCEMS). 2020, p. 982–7. <http://dx.doi.org/10.1109/SCEMS48876.2020.9352310>.
- [11] Bolz V, Rueß J, Zell A. Power flow approximation based on graph convolutional networks. In: 2019 18th IEEE International conference on machine learning and applications (ICMLA). 2019, p. 1679–86. <http://dx.doi.org/10.1109/ICMLA.2019.00274>.
- [12] Wang D, Zheng K, Chen Q, Luo G, Zhang X. Probabilistic power flow solution with graph convolutional network. In: 2020 IEEE PES Innovative smart grid technologies Europe (ISGT-Europe). 2020, p. 650–4. <http://dx.doi.org/10.1109/ISGT-Europe47291.2020.9248786>.
- [13] Kundacina O, Cosovic M, Vukobratovic D. State Estimation in Electric Power Systems Leveraging Graph Neural Networks. 2022, URL <http://arxiv.org/abs/2201.04056>, arXiv:2201.04056 [cs, eess, math] (Apr. 2022).
- [14] Liao W, Bak-Jensen B, Pillai JR, Wang Y, Wang Y. A review of graph neural networks and their applications in power systems. *J Mod Power Syst Clean Energy* 2022;10(2):345–60. <http://dx.doi.org/10.35833/MPCE.2021.000058>, Conference Name: Journal of Modern Power Systems and Clean Energy.
- [15] Jeddi AB, Shafieezadeh A. A physics-informed graph attention-based approach for power flow analysis. In: 2021 20th IEEE International conference on machine learning and applications (ICMLA). 2021, p. 1634–40. <http://dx.doi.org/10.1109/ICMLA52953.2021.00261>.
- [16] Lopez-Garcia TB, Domínguez-Navarro JA. Power flow analysis via typed graph neural networks. *Eng Appl Artif Intell* 2023;117:105567. <http://dx.doi.org/10.1016/j.engappai.2022.105567>, URL <https://www.sciencedirect.com/science/article/pii/S0952197622005577>.
- [17] Donon B, Clément R, Donnot B, Marot A, Guyon I, Schoenauer M. Neural networks for power flow: Graph neural solver. *Electr Power Syst Res* 2020;189:106547. <http://dx.doi.org/10.1016/j.epsr.2020.106547>, URL <https://linkinghub.elsevier.com/retrieve/pii/S0378779620303515>.
- [18] Fusco F, Eck B, Gormally R, Purcell M, Tirupathi S. Knowledge- and data-driven systems for energy systems using graph neural networks. In: 2020 IEEE International conference on big data (Big Data). 2020, p. 1301–8. <http://dx.doi.org/10.1109/BigData50022.2020.9377845>.
- [19] Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS. A comprehensive survey on graph neural networks. *IEEE Trans Neural Netw Learn Syst* 2021;32(1):4–24. <http://dx.doi.org/10.1109/TNNLS.2020.2978386>, Conference Name: IEEE Transactions on Neural Networks and Learning Systems.
- [20] Morris C, Ritzert M, Fey M, Hamilton WL, Lenssen JE, Rattan G, Grohe M. Weisfeiler and leman go neural: Higher-order graph neural networks. In: Proceedings of the AAAI Conference on artificial intelligence. 33, 2019, p. 4602–9.

- [21] Shi Y, Huang Z, Feng S, Zhong H, Wang W, Sun Y. Masked label prediction: Unified message passing model for semi-supervised classification. In: Proceedings of the thirtieth international joint conference on artificial intelligence. Montreal, Canada: International Joint Conferences on Artificial Intelligence Organization; 2021, p. 1548–54. <http://dx.doi.org/10.24963/ijcai.2021/214>, URL <https://www.ijcai.org/proceedings/2021/214>.
- [22] Vogt S, Braun A, Dobschinski J, Sick B. Wind power forecasting based on deep neural networks and transfer learning. In: Wind integration workshop. Dublin, Ireland; 2019.
- [23] Blundell C, Cornebise J, Kavukcuoglu K, Wierstra D. Weight uncertainty in neural networks. 2015, <http://dx.doi.org/10.48550/ARXIV.1505.05424>, URL <https://arxiv.org/abs/1505.05424>.
- [24] Owens RG, Hewson T. ECMWF forecast user guide. 2018, <http://dx.doi.org/10.21957/m1cs7h>, URL <https://www.ecmwf.int/node/16559>.
- [25] Entsoe transparency platform. 2022, Last accessed: 2022-02-03, <https://transparency.entsoe.eu>.
- [26] Volue insight. 2022, Last accessed: 2022-05-27, <https://www.volue.com/insight>.
- [27] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S. Pytorch: An imperative style, high-performance deep learning library. In: Advances in neural information processing systems 32. Curran Associates, Inc.; 2019, p. 8024–35, URL <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- [28] Fey M, Lenssen JE. Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on representation learning on graphs and manifolds. 2019.