

This paper was presented at the 18th Wind Integration Workshop and published in the workshop's proceedings.

Wind Power Forecasting Based on Deep Neural Networks and Transfer Learning

Stephan Vogt, Axel Braun, Jan Dobschinski
Forecasting in Energy Systems
Energy Economics and Grid Operation Division
Fraunhofer IEE
Kassel, Germany
Email: stephan.vogt@iee.fraunhofer.de

Bernhard Sick
Intelligent Embedded Systems
Electrical Engineering / Computer Science
University of Kassel
Kassel, Germany
Email: bsick@uni-kassel.de

Abstract—State-of-the-art forecasting systems are often based on single machine learning models that have been trained for individual wind farms. The data of a each wind farm is typically used exclusively for its "own" model. This article presents two approaches with deep neural networks to make the data usable across wind farms with transfer learning. In the first case, the adaptation to individual wind farms is achieved by an separate output layer for each wind farm. With the second approach, a Bayesian wind farm embedding is proposed. An experiment with realistic forecast conditions based on power measurements and weather forecasts of 19 wind farms is carried out. The proposed techniques are compared to established single wind farm models such as random forests, gradient boosted regression trees, and simple multi layer perceptrons. Our results indicate that a significant improvement in prediction quality can be achieved using multi-task learning, especially with a short time span of historical training data.

I. INTRODUCTION

Two factors represent the ongoing change in modern energy supply systems. On the one hand, we observe an increase in weather-dependent generation from renewable energies. On the other hand, this goes hand in hand with decentralisation through smaller generation units. As a result, there are specific demands on grid management and energy markets which can only be met with suitable forecasts.

Given these circumstances, modern forecasting systems produce forecasts for a large number of wind farms or photovoltaic plants. Machine learning (ML) models, such as researched in [10] or [9], become increasingly popular for this purpose. The usual setting is to train an individual ML model for each plant or park on the basis of historical weather forecasts and the corresponding power measurement data. This allows the ML model to learn the individual generation characteristics of the power plant and to give a good estimate of the expected power for future weather forecasts. However, it is assumed that sufficient historical training data is available. If this is not the case, physical models are often used. These cannot usually be used without uninformed, average assumptions, because the forecast provider often does not know the exact characteristics of the power plant. Furthermore, a detailed physical modelling is complex and often a time consuming engineering task.

The techniques of transfer learning (TL), as e.g. summarized in [20], and in particular multi-task learning (MTL), such as described in [22], offer the possibility of using ML models across different plants and gaining more general model knowledge from the training data. This article adopts

MTL to wind power forecasting and investigates the benefits of such models versus single task models depending on the length of available training data. The data from 19 operating wind farms will be used in a software experiment.

The article is divided into the following sections. Section II first explains the models and techniques used. Then, the developed approaches for MTL in wind power prediction are presented in Section III. Section IV contains explanations and information concerning the data used from 19 wind farms. Before carrying out and evaluating the corresponding data experiment, Section V provides a detailed analysis of the similarity between the wind farms. The evaluation of the experiments for comparison between MTL and single task models and the obtained results are discussed in Section VI. Finally, a discussion with conclusions and outlook to future research is given in Section VII.

II. METHODS

Machine learning models are widely used in wind power forecasting. This section provides an overview of the ML-methods used in the following sections. This includes random forests (RF), gradient boosted regression trees (GBRT) and artificial neural networks (ANN) as well as multi-task learning techniques. In addition, the procedure used for the optimization of hyperparameters, Bayesian optimization, is briefly examined.

A. Random Forests

RF are widely used for both, classification and regression. The use as regression model is of special interest for wind power forecasting purposes. The RF model is a bagging based ensemble [4] of randomized classification and regression trees (CART) [3].

The CART algorithm creates binary trees. Two branches are formed at each node on the basis of a decision, which further divides the data. Typically, individual features are compared to a threshold value. Each branching either leads to another node with branching or to a leaf. The leaf then contains the class or, here, the mean value to be predicted. In training, the feature and threshold of each branch are selected by a greedy search so that the tree is built in a recursive manner starting with the root node until the maximum number of branches has been reached or until each leaf covers at least a minimum number of samples.

For the bagging ensemble, several of these trees are trained on different subsets of the training data. For this purpose, samples are drawn by bootstrapping [6] for each tree in the ensemble. In addition, only a random subset of the features is considered when selecting the feature for branching, with the aim of further reducing the correlation between the trained trees and thus lowering the bias of the model.

B. Gradient Boosted Regression Trees

GBRT, as well as RF, are a combination of an ensemble technique, in this case gradient boosting, and CART trees, which represent the ensemble members. This technique is based on the idea of boosting, in which the $(k+1)$ -th member is trained in such a way that it corrects the weaknesses of the prediction $\hat{y}_i^{(k)}$ from the combined preceding members $\{1, \dots, k\}$. While other boosting techniques may apply a stronger weighting of samples with large errors, gradient boosting adapts the idea of gradient descent and trains each member as an approximation of the loss gradient.

For the purpose of regression, the gradient of the mean squared error (*MSE*) loss

$$\frac{\partial MSE}{\partial \hat{y}_i^{(k)}} = \frac{\partial}{\partial \hat{y}_i^{(k)}} \frac{1}{N} \sum_{i=1}^N (\hat{y}_i^{(k)} - y_i)^2 \propto (\hat{y}_i^{(k)} - y_i) \quad (1)$$

can be calculated based on the target y_i with respect to the prediction of the current ensemble $\hat{y}_i^{(k)}$. As can be seen in Eq. (1), the gradient is proportional to the residuals which can be approximated with a CART tree $h^{(k)}(\mathbf{x}_i) \approx \hat{y}_i^{(k)} - y_i$. In the sense of gradient descent, the tree trained in this way is scaled with a step size $\alpha \in (0, 1]$ and subtracted from the previous ensemble prediction

$$\hat{y}_i^{(k+1)} = \hat{y}_i^{(k)} - \alpha h^{(k)}(\mathbf{x}_i), \quad (2)$$

with $\hat{y}_i^{(0)} = \frac{1}{N} \sum y_i$, until k reaches the maximum number of trees. The step size and the number of trees, as well as the CART tree hyperparameter, influences the tendency of the model to over-fit. This technique was first introduced in [7].

C. Artificial Neural Networks

ANN are becoming increasingly popular due to recent developments with deep neural networks and their great flexibility. The progress made in image recognition [15] and speech recognition [24] should be mentioned in particular.

A modern ANN consists of a sequence of differentiable tensor operations, referred to as layers. The fully connected layer, a typical example and a central building block in all kinds of ANN, is defined with

$$\mathbf{h}^{(k)} = \varphi(\mathbf{W}^{(k)}\mathbf{h}^{(k-1)} - \mathbf{b}^{(k)}) \quad (3)$$

where $\mathbf{h}^{(k)} \in \mathbb{R}^n$ is a vector of so-called activations. The activations are a transformed representation of the data representation $\mathbf{h}^{(k-1)} \in \mathbb{R}^m$ from the preceding layer. The weight matrix $\mathbf{W}^{(k)} \in \mathbb{R}^{n \times m}$, the threshold or bias vector $\mathbf{b}^{(k)}$, and the nonlinear activation function $\varphi(\cdot)$ characterize the transformation of the k -th layer.

In the simplest case of a deep neural network, the multi layer perceptron (MLP), there is a sequence of several of

these layers. In a MLP, the input data vector \mathbf{x} of one sample is represented with the input layer $\mathbf{h}^{(0)} = \mathbf{x}$ while the output is computed with the last layer $\hat{\mathbf{y}} = \mathbf{h}^{(\text{end})}$. While the activation function of the output layer is usually linear in the case of regression, the activation functions of the previous layers, the so-called hidden layers, are non-linear. The weights and thresholds of each layer are trainable parameters. To train the MLP, the parameters of the model are adjusted so that a given loss function is minimized. With the help of the backpropagation method given in [23], the gradient of the loss function is formed with respect to the parameters.

D. Multi-Task and Transfer Learning

Transfer Learning (TL) in ML refers to techniques that allow knowledge to be transferred automatically from one problem, application, or task to another. A common application is the adaptation of models for image recognition. Typically, such models have been trained in a very time-consuming manner on a limited set of classes. The model found a certain understanding for the structures in images. If ones want to recognize a new class it, can be useful to build on the already gained knowledge. In practice, some of the last layers of a corresponding ANN are typically exchanged and re-trained, while the previous layers provide their knowledge unchanged in the form of suitable latent features.

Multi-Task Learning (MTL) is a variant of TL in which source and target tasks are rather simultaneously available. The goal here is a model which fulfils different tasks as best as possible in such a way that all tasks profit from each other. MTL as it is used in the following can be assigned to the category of homogeneous inductive TL where the input data of the source and target tasks originate from the same feature space [20]. At the same time, however, different tasks are to be accomplished (inductive).

A major challenge in MTL is to make common model knowledge usable in a beneficial way and at the same time to allow for a specification of the respective tasks. ANN are a viable way of doing this, as they are characterised by a learning of representations [22]. The technique of Hard Parameter Sharing (HPS) used in the following uses a small, individual part of the ANN parameters for certain tasks, while the mostly larger part of the model parameters is learned jointly across all tasks. A similar design is the described TL example where the last layers of an ANN are adapted to the individual task, while the layers closer to the model input form a common representation of the input data which is useful across several tasks.

Examples for TL in wind power forecasting are given in [21] and [25].

E. Hyperparameter Optimization

ML models strongly depend on the adjustable hyperparameters in their model properties. Examples of hyperparameters range from the maximum number of splits in a CART to regularization parameters in ANNs. The choice of hyperparameters influences in particular the ability of the model to learn complex dependencies in the data and to generalize.

In order to select the optimal hyperparameters for the desired problem, cross-validation is a suitable approach. To do this, the training data is divided several times into a training and validation set. For each training set, a model is then trained and evaluated on the validation set using a suitable error measure. A mean value of all validation errors is then evaluated. This is called a cross-validation error and estimates the quality of the model for new, unknown input data. The cross-validation error can be used to compare different hyperparameters.

A special feature in the field of wind power forecasting is the time dependency. The subdivision of the training and validation set should not be random, so that the generalization error is not falsified. In the following, the data is therefore divided into four chronologically successive sections. In four training sessions one of these blocks is used as validation data while the rest is used for model training.

The optimal choice of hyperparameters is an optimization problem with the cross-validation error being the objective function. Typically, a gradient-free optimization method is required that works despite local minima and, ideally, is able to process stochastic objective functions for which the evaluation of a solution is non-deterministic (e.g., due to random weight initialization). One such method, which is enjoying increasing popularity (e.g., in [8] and [26]), is Bayesian optimization. The method estimates the objective function using the evaluations of the objective function already made by a Gaussian process model. Since this yields both, the expected value and the standard deviation of the objective function for each unevaluated point, the expected model improvement can be estimated in advance. The point with the greatest expected model improvement is then selected for the next evaluation. In our experiments we used the Matlab implementation in [16]. For each of the trained models, the same configuration of Bayesian optimization with 60 optimization steps was selected.

After optimization, the found hyperparameters are used to train a model with training and validation data. This is then evaluated against the previously omitted (chronologically subsequent) test set. On the basis of this test error, the comparison between the ML methods used is finally conducted.

III. MULTI-TASK LEARNING APPROACH FOR WIND FARM FORECASTING

This section describes two approaches developed to find a model for several wind farms using MTL. The first subsection describes the challenge of time-asynchronous MTL, followed by the description of the first MTL approach with a task-specific output layer. Then follows the description of the second MTL approach with a Bayesian task embedding and finally how TL can be applied to the MTL-trained model.

A. Asynchronous Multi-Task Learning for Wind Farm Data

MTL usually considers input data with several target values for one sample. In the case of wind power forecasting, this is not the case. Input data samples are only valid for the respective location of a wind farm. In addition, historical power measurements of several wind farms often only cover a small joint period of time.

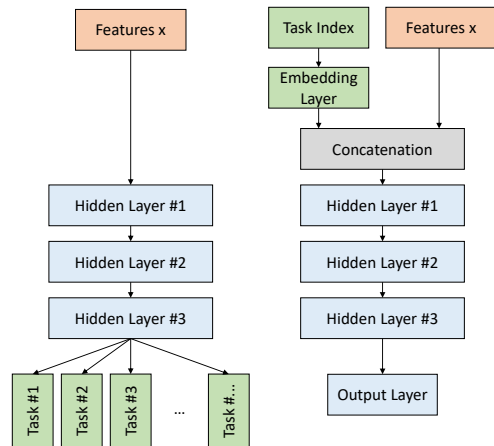


Fig. 1. The two model structures for hard parameter sharing in deep multi-task neural networks with task specific output layers (left, MTL-MLP) and a task embedding layer (right, BE-MTL-MLP). Light blue boxes represent the shared layers, while task specific layers are green. The feature input is represented with a light red color.

In the following, the wind farm tasks are trained asynchronously. Since our models assume i.i.d., this takes place in the form of a stochastic gradient descent with mini batches. Gradually, randomly drawn batches with, e.g., 32 samples are presented to the MLP, each with a different timestamp and task, i.e., wind farm. The output of a sample is calculated with the corresponding task-specific parameters. Using backpropagation, the gradient of the loss function is then computed with regard to the shared parameters and the respective task-specific parameters for each sample. The so gained sample gradients are applied in a stochastic gradient descent step to update these parameters. At the same time, however, the task-specific parameters of the other tasks remain unaffected by a sample, provided that this does not belong to the corresponding task.

How strongly a task affects the shared parameters in relation to the other tasks depends on the average gradients of the loss function and the number of samples of the respective task. Since the errors of all wind farms are expected in a similar range and since wind farms with more samples should be evaluated more strongly, no further measures are necessary here.

B. Wind Farm Specific Output Layer

An MLP learns a representation in each layer, which makes it easier for the subsequent layers to generate the desired output. In the case of regression, the last layer forms a pure linear combination of the activations of the previous layer. This corresponds to a linear regression, a rather simple model, which, however, can generate complicated, non-linear outputs by the learned non-linear representation of the input data in the second last layer. Classic MTL with HPS is therefore usually applied to one or more task-specific output layers. However, the more layers the individual tasks share and the less layers are trained task-specifically, the more the model is forced to use similar representations across tasks. Therefore, only the output layer was adapted to the respective task, i.e. the forecast for the respective wind farm.

In order to additionally control the strength of multi-task learning, the number of units of the second last layer was selected using hyperparameter optimization.

Fig. 1 shows such an MLP architecture with an exchangeable, task-dependent output layer on the left-hand side. The hidden layers are the same for all tasks. This MTL model is abbreviated as MTL-MLP for multi-task learning multi layer perceptron in the following.

C. Bayesian Wind Farm Embedding

Embeddings have become more and more popular in recent years to provide categorical entities (e.g., words) in a vector representation understandable for ANN as input data. Besides the application in language models such as word2vec, e.g. in [17] and [18], there is also the application for the representation of general categories in [12]. Embeddings replace a one-hot encoding, which generates high-dimensional vectors depending on the number of categories, by a real-valued lower-dimensional vector. The embedding vectors can be trained in a supervised way via backpropagation.

In the MTL application with many tasks we will encode the tasks in the form of an embedding vector. Fig. 1 shows an MLP architecture on the right side where the embedding of the respective task is fed into the model in addition to the weather-dependent input data. During the training, the weights of the ANN and the embedding vectors are trained simultaneously by minimizing the MSE. Due to the small number of 19 entities in the embedding and the almost unlimited flexibility of the ANN, similar tasks may not be close together. Therefore, for each entity of the embedding (task), an identical, independent prior probability distribution is chosen. This ensures that indistinguishably similar tasks form a close neighbourhood, while differences in tasks that are sufficiently supported by data lead to different embedding encodings. This model is abbreviated with BE-MTL-MLP for Bayesian embedding multi-task learning multi layer perceptron.

To find a posterior distribution of the embedding, *Bayes By Backprop* is applied to the embedding layer. This technique described in [2] is mainly based on the work in [13] and further aspects in [11]. In the concrete case, a standard normal distribution was used as a variational distribution. Thus, for each embedding parameter there are two parameters of the variational distribution to be learned. These are the mean $\mu_{i,j}$ and the standard deviation $\sigma_{i,j}$ of the i -th task and the j -th embedding dimension.

The local reparametrization trick described in [14] allows sampling from the variational distribution. For this, a value for ϵ is drawn from a standard normal distribution to compute the sampled embedding with

$$w_{i,j} = \mu_{i,j} + \sigma_{i,j}\epsilon \quad \text{with} \quad \epsilon \sim \mathcal{N}(0,1). \quad (4)$$

Each sampled embedding is randomly combined with a training data sample. Backpropagation yields the gradient of the training loss L with respect to the embedding parameter $w_{i,j}$. This is combined with the gradient of the Kullback-Leibler (KL) loss based on the KL-divergence between the variational distribution and the standard normal distribution

to obtain the following update rules:

$$\Delta\mu_{i,j} = \frac{\partial L}{\partial w_{i,j}} + \lambda\mu_{i,j} \quad (5)$$

$$\Delta\sigma_{i,j} = \frac{\partial L}{\partial w_{i,j}}\epsilon + \lambda\left(\sigma_{i,j} - \frac{1}{\sigma_{i,j}}\right). \quad (6)$$

The parameter λ controls the weighting of the KL loss against the prediction loss L . Since the uncertainty of the target value is not known and since the data cannot be assumed to be i.i.d., it is recommended to search for the parameter λ via hyperparameter optimization and cross-validation.

D. Model Transfer to Wind Farms

A forecast operator usually manages a portfolio of wind farms with sufficient historical data and often wants to add new wind farms with few historical data. In this scenario it makes sense to train an MTL model on the historical data and to adapt it to the new wind farms using TL. In the following, TL is realized by re-training the task-specific parameters while the shared MTL layers do not change their weights further. This makes sense especially with few historical data for a new wind farm. The more training data becomes available, the more fine-tuning of the shared layers should be considered. However, this will not be examined in the following.

IV. EXPERIMENTAL SETUP

A. Data Set

The data used in the experiments comprise 19 wind farms. All wind farms are located in the south-west of Germany. The data includes the measured aggregated power of the turbines of a wind farm as well as input features, such as variables from a numerical weather prediction (NWP) of the ECMWF from the deterministic IFS forecast described in [19]. The input features are linearly interpolated from a 3h resolution to the 15min sampling of the power measurements. To simulate day ahead forecasts only the forecast horizons between +24h and +48h of the 00:00 a.m. model run were selected.

In addition to the features primarily relevant to wind power, such as wind speed, wind direction, air pressure and temperature, other features such as dew point temperature and, global horizontal radiation are taken into account (air mas and mountain / valley breeze effects). For the wind speed, which is considered both in 10m and 100m height as well as the global horizontal radiation, three time-lags with a shift of -1h, 0h and 1h around the forecast time are formed. The wind direction is represented in the form of cosine and sine values in order to avoid apparent sudden dissimilarity between 0° and 360° . In addition to the NWP forecasts, temporal features such as legal night time and sun angles are used.

B. Test Procedure and Model Evaluation

A period of one year was chosen for the training of the models. The start of the training period is April 4, 2018. The test period is between April 4, 2019 and June 17, 2019. The main goal of the experiment is the comparison between classical single task ML models and MTL models in wind

power forecast. Each park is, therefore, evaluated with a trained model on the data in the test period. The single-task models are trained with the training data of the respective park. The MTL models, on the other hand, are trained with data from several parks. Under the assumption that MTL models show a particular advantage when the training period is shortened, the training is carried out with data segments of different durations. These different training durations include 3, 7, 14, 30, 90, 180 and 365 days. In any case, the end time of the training period is April 4, 2019, while the start time is chosen accordingly. In this way, even the shorter training phases have the most up-to-date data available.

For the MTL models, the 19 wind farms are divided into four subsets with four or five parks each. In order to keep the computational effort small, four MTL models are pre-trained, each with the wind farms from three of the four subsets. The pre-trained models always cover the entire training period of a year. This pre-trained MTL model is then re-trained on the shortened training period of a wind farm that was not included in the three subsets. This process is repeated for all wind farms. In this way, it can be simulated that a new wind farm with little data is integrated into a pool of wind farms with a longer historical data set.

C. Model Calibration

For each model, various hyperparameters including their expected value ranges were selected in advance. The more hyperparameters that are optimized, the larger the search space. Thus, the limited number of iterations usually leads to a suboptimal result. If only one hyperparameter is selected, this may also lead to suboptimal results, since the potential of model diversity is not fully exploited. This section documents the selected hyper parameters and their ranges.

The **RF**, consisting of 100 CART trees, is optimized with three hyperparameters. Among them is the minimum number of samples per leaf with values between 1 and 100. In addition, the maximum number of splits between 1 and 20 and the number of features to sample between 1 and 17 (all features) were optimized.

The **GBRT** has been optimized analog to the RF regarding the maximum number of splits and the number of features to be sampled per split in the same ranges. The minimum number of samples per leaf were fixed to ten, while the step size has been optimized in the interval $\alpha \in [10^{-8}, 1]$. With α close to 0, the GBRT remains close to the mean of the training target. A small α can therefore reduce overfitting but could also cause underfitting.

In the case of **MLP**, four hyperparameters were optimized. These include the number of hidden layers between one and four, the number of units per hidden layer between 20 and 50 and the regularization strength (elastic net) between 10^{-8} and 1 as well as the weighting between L1 and L2 norm in the range between 0 and 1. The L1 and L2 regularization is applied in the same amount for the weights \mathbf{W} in all layers. In the case of the **MTL-MLP**, another hidden layer was added before the output layer. The number of units in this second last layer is individually optimized in the range between one and ten, because it controls how much the different tasks have to learn from each other.

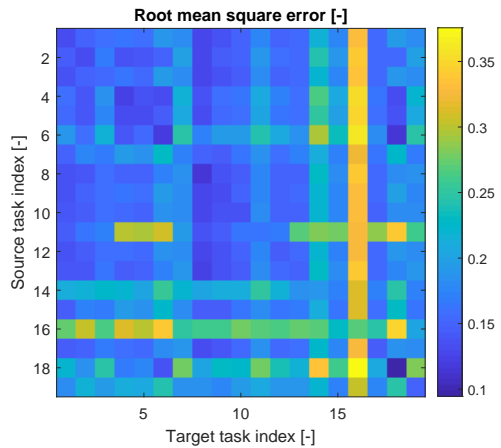


Fig. 2. Heat map with the RMSE error values of the models trained with the source task data while being evaluated on the target task data.

The ANN of the **BE-MTL-MLP** model is trained with simple L2 regularization since only the embedding is Bayes. In this case the regularization is between 10^{-8} and 10^3 . Additionally, hyperparameters are optimized to weight the KL divergence in the range $\lambda \in [10^{-8}, 1]$ and to select a number of hidden embedding dimensions between one and five.

V. ANALYSIS OF WIND FARM SIMILARITIES

The benefit of MTL methods depends to a large extent on the similarity of the tasks considered. The application of multi-task learning to wind power forecasting described here falls in the category of homogeneous transfer learning in which the input data of the different tasks come from the same feature domain. This allows the model of one (source) task to be applied directly to another (target) task and to generate a prediction without further adjustment. Measuring the quality of such a forecast provides an indication of the similarity or dissimilarity of both tasks.

Fig. 2 shows a heatmap in which the RMSE in the training period was color coded depending on the pairing of source and target task. A simple regression tree was trained as a prediction model for each source task wind farm where the maximum number of splits was optimized using hyperparameter optimization and cross-validation. Each model created this way is in turn used to make a forecast using the features for each target wind farm. Finally, the target data of the target wind farm is then compared to the forecast using the RMSE. The RMSE refers to the nominal power-standardized power measurement time series in order to achieve better comparability between wind farms of different nominal power.

The RMSE heatmap shows a largely symmetrical matrix. This matrix is square because each wind farm is combined with each wind farm as source and target. The symmetry of the matrix results from the circumstance that a source model on target task data is similarly bad as using the model trained on the target task data on the source data instead. The reason that the matrix is not perfectly symmetrical is that some parks are generally easier to forecast than others (e.g. because of a general low power production).

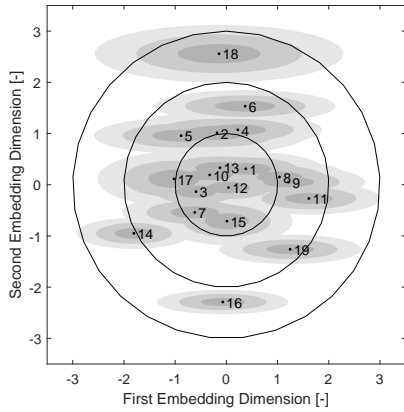


Fig. 3. Bayesian wind farm embedding based on deep neural network and standard normal distributed prior ($\lambda = 10^{-4}$).

In the concrete example it is noticeable that a handful of wind farms distinguish themselves by larger RMSEs. In particular, no trained model seems to be able to predict wind farm 16 with a low RMSE. On the other hand, a relatively low RMSE for wind farm 19 can be achieved with the model trained with wind farm 16. Also of interest is wind farm 11, which seems to be suitable as a model for farm 1 to 3 and farm 7 to 12, while it usually fails for the other wind farms. However, wind farm 11 can be predicted relatively well with almost all other wind farms.

Fig. 3 shows a 2-dimensional embedding which was found using a BE-MTL-MLP model trained with all 19 wind farms. The grey ellipses around the points with the farm indices indicate the posteriori distribution with one, two, and three standard deviations of the respective wind farm. The euclidean distances of the points in the embedding partially reflect the differences in RMSE as seen in the heatmap in Fig. 2. For example, the heatmap shows a distinct increase in the RMSE if the model from wind farm 18 is used for wind farm 16 or vice versa. At the same time, wind farm 18 and wind farm 6 seem to be similar, as a particularly small increase in the RMSE occurs when the models are exchanged. These differences in the RMSE are reflected in the distances between the wind farm embedding points in Fig. 3.

Fig. 4 shows power curves, i.e. the wind power depending on the 100m wind speed computed with the trained BE-MTL-MLP model. The two diagrams show the power curves of all 19 wind farms (each marked with the farm id). The colour coding corresponds to the position of the respective wind farm in the embedding in Fig. 3. The first and second embedding dimensions correspond to the color of the upper and lower diagram respectively.

In this case, the second embedding dimension shows a clear dependence on the shape of the power curve. The first embedding dimension shows no clear relationship to the shape of the curve. The exact wind farm characteristics that influence this embedding dimension have not yet been determined.

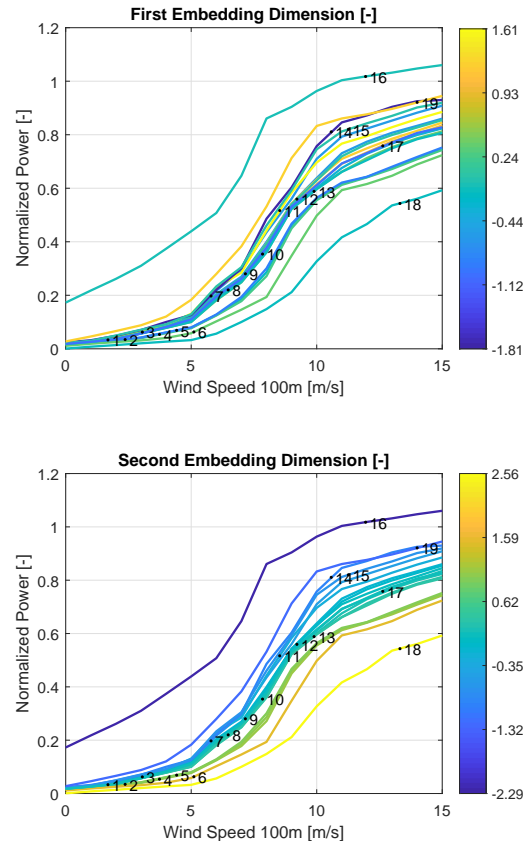


Fig. 4. Effect of the wind farm embedding parameters from Fig. 3 on the shape of the power curve.

VI. RESULTS

In this section, the approaches are evaluated and compared. The main question here is whether the use of multi-task learning methods is justified.

A. Evaluation of errors

Fig. 5 shows several box plots with the RMSE values of the different models depending on the duration of the training data. The individual boxes represent the distribution of the 19 RMSE values during the test period. A value of 0.15 corresponds to a standardized RMSE of 15%, which was determined with the normalized power with respect to the nominal power.

For 90 training days or more, very similar distributions of model errors occur. This apparently applies both to the errors of a model with different training durations and to a training duration with different models. With 30 training days or less, the MLP is particularly noticeable with large errors and strong differences between the wind farms. Apart from BE-MTL-MLP, the remaining models also show a relatively distinct increase in the test error with decreasing training duration.

Overall, apart from MLPs, the distributions of the models do not appear to be significantly different. This is mainly due to the different RMSE of different wind farms, depending on the location and number of turbines.

A more compact representation of the results is given in Fig. 6. This figure contains the average RMSE over all wind farms of the individual models plotted over the training

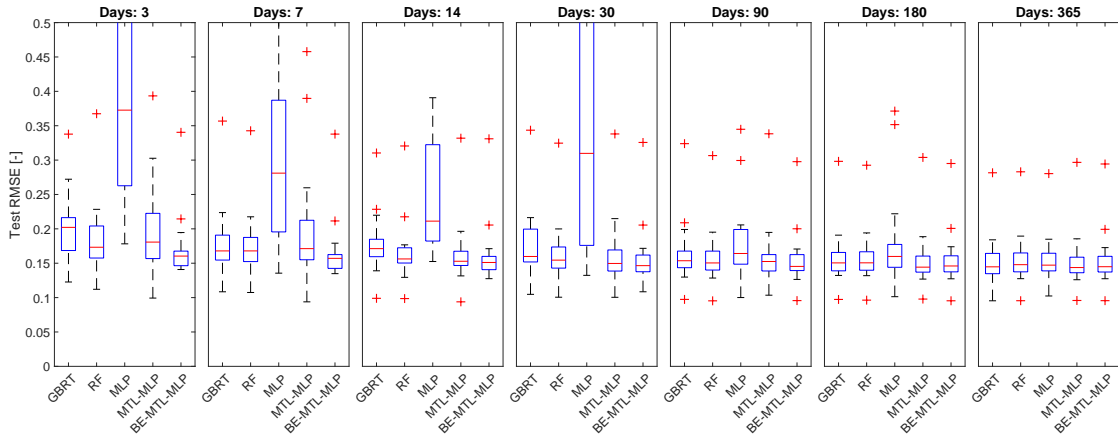


Fig. 5. Evaluation of the models depending on the length of the training data period per subplot. Each box represents the distribution of the RMSE of the 19 wind farms of the model given below.

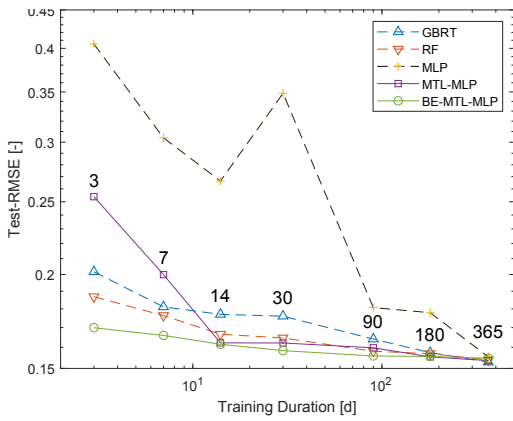


Fig. 6. Double logarithmically scaled plot of the wind farm portfolio mean test RMSE per model plotted over the length of the training data.

duration. The axes are both scaled logarithmically. From this illustration it becomes clear that a decrease of the error is to be expected with increasing training duration. At the same time, this applies to different models at different degrees.

Remarkable are the large mean errors of the MLP. As the amount of training data increases, the training errors are aligned with the other models. One reason could be that a smaller number of samples requires a higher number of training epochs. Since this has not been further investigated, the seemingly inferior quality for MLPs in this application cannot be proven.

Compared to MLP, tree-based methods seem to be more robust with a small number of training samples. One possible cause is that the output of a regression tree is the mean value of some power measurements in the training set. On the other hand, the output of MLPs for new input data may be random if there were no comparable examples in the training data.

The BE-MTL-MLP shows low RMSE and a much lower dependency on training duration. With a length of the training period of three days, this achieves the model quality of the GBRT for even more than 30 days. If the BE-MTL-MLP is trained with 30 days it is with an error of 15.8% approx. 0.5% points away from the best test error, which is the GBRT with 365 training days and an RMSE of 15.3%.

The MTL-MLP seems to take at least 14 training days to

stand up to the single task models. The difference between BE-MTL-MLP and MTL-MLP is mainly to be found in the missing prior of the MTL-MLP. The weights in the task output layer of the MTL-MLP can deviate more easily from the already found weights of the other tasks and thus overfit more quickly.

In order to show whether one method performs better than another, the wind farm specific error portion must be removed. A simple way to do this is to calculate the difference between the errors of two models per wind farm and to depict this distribution of the pairwise difference, as shown in Fig. 7. The BE-MTL-MLP was selected as the comparison model. If the differences between the model RMSEs and the BE-MTL-MLP are predominantly in the positive range, a higher quality of the BE-MTL-MLP can be expected in these cases. In addition, triangular markings are shown next to the boxes. These indicate the 95% confidence range of the estimated median (red line in the box). If both triangles are above the red line, this is a first, important indication of a significant result in favor of the BE-MTL-MLP model.

VII. CONCLUSION

The aim of this work is to compare multi-task learning methods for cross-farm model training to individual wind farm models in power forecasting. An experiment with historical data from 19 wind farms is described. This shows that MTL methods are particularly advantageous when only a small amount of training data is available. In some cases results are achieved with a training period of 3 days, while other models must be trained for at least 30 days to achieve a similar result. With one month of training days we can achieve similar results as with one year without MTL.

Random Forests, Gradient Boosted Regression Trees, and simple Multi Layer Perceptrons were investigated as single wind farm models. Two approaches based on multi-task learning were presented. Both methods are based on deep neural networks with Hard Parameter Sharing. They stand out due to their asynchronous training periods, as the training data of the individual systems do not have to cover the same period of time.

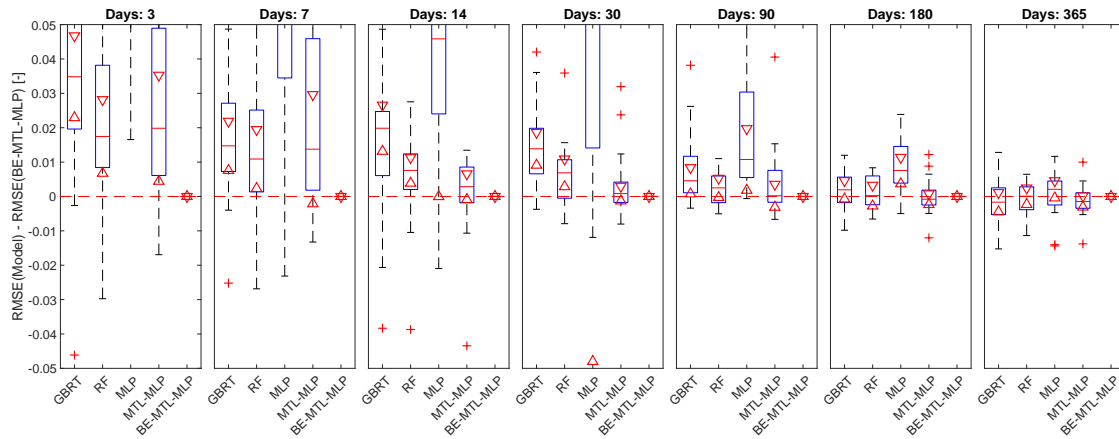


Fig. 7. Distribution of the pairwise differences between the RMSEs of the individual wind farms of a model with the RMSEs of the individual wind farms and the multi-task MLP in analogy to Fig. 5.

The first method, MTL-MLP, uses a task-specific output layer. In the case of this model, it shows that training data should be available for at least two weeks. The second method, BE-MTL-MLP, is even more robust in this aspect. It uses a stochastic wind farm embedding, which is trained with a Bayesian prior. This approach allows for a training with extremely few training samples (3 days).

In addition, the derived embedding offers the possibility of model interpretation. This shows that the learned parameters can, among other things, give an indication of the shape of the power curve. Another application of embedding could be to automatically identify wind farms with unusual or erroneous power data.

Future work will relate to the application in regional power forecasts. The main problem is the prediction of unmeasured power plants in the region. Here, the shown probabilistic modelling of the wind farm embedding can find a further useful application.

In addition, the methods presented can be used for other types of power plants, e.g. photovoltaics or biomass. In combination with different types of power plants, an application in load flow forecasting in electrical grids is especially of interest.

ACKNOWLEDGMENT

This work has been carried out within the German research project gridcast (Fkz. 0350004A) funded by the German Federal Ministry for Economic Affairs and Energy.

REFERENCES

- [1] Bishop, C.: *Neural Networks For Pattern Recognition*. Clarendon Press, Oxford, 1995.
- [2] Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. *Weight uncertainty in neural networks*. arXiv preprint arXiv:1505.05424, 2015.
- [3] Breiman, L., J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Boca Raton, FL: CRC Press, 1984.
- [4] Breiman, L. *Bagging Predictors*. Machine Learning. Vol. 26, pp. 123-140, 1996.
- [5] Breiman, L. *Random Forests*. Machine Learning. Vol. 45, pp. 532, 2001.
- [6] Efron, B. *Bootstrap methods: Another look at the jackknife*. The Annals of Statistics. 7 (1): 126., 1979.
- [7] Friedman, J. *Greedy function approximation: A gradient boosting machine*. Annals of Statistics, Vol. 29, No. 5, pp. 1189-1232, 2001.
- [8] Gelbart, M., J. Snoek, R. P. Adams. *Bayesian Optimization with Unknown Constraints*. <https://arxiv.org/abs/1403.5607>, 2014.

- [9] Gensler, A., Henze, J., Sick, B., & Raabe, N. (2016, October). *Deep Learning for solar power forecasting: An approach using AutoEncoder and LSTM Neural Networks*. In 2016 IEEE international conference on systems, man, and cybernetics (SMC) (pp. 002858-002865). IEEE.
- [10] Gensler, A. *Wind Power Ensemble Forecasting: Performance Measures and Ensemble Architectures for Deterministic and Probabilistic Forecasts* (Vol. 12). kassel university press GmbH, 2019.
- [11] Graves, A. (2011). Practical variational inference for neural networks. In Advances in neural information processing systems (pp. 2348-2356).
- [12] Guo, C., & Berkhahn, F. *Entity embeddings of categorical variables*. arXiv preprint arXiv:1604.06737, 2016.
- [13] Hinton, G. E., & Van Camp, D. *Keeping the neural networks simple by minimizing the description length of the weights*. In Proceedings of the sixth annual conference on Computational learning theory (pp. 513). ACM, 1993.
- [14] Kingma, D. P., Salimans, T., & Welling, M. *Variational dropout and the local reparameterization trick*. In Advances in Neural Information Processing Systems (pp. 2575-2583), 2015.
- [15] LeCun, Y. et al.: *Backpropagation Applied to Handwritten Zip Code Recognition*. Neural Computation, Vol. 1, pp. 541-551, 1989.
- [16] *MATLAB and Statistics Toolbox Release 2018b*, The MathWorks, Inc., Natick, Massachusetts, United States.
- [17] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. *Distributed representations of words and phrases and their compositionality*. In Advances in neural information processing systems (pp. 3111-3119), 2013.
- [18] Mikolov, T., Chen, K., Corrado, G., & Dean, J. *Efficient estimation of word representations in vector space*. arXiv preprint arXiv:1301.3781, 2013.
- [19] Owens, R. G., Hewson, T. D.: *ECMWF Forecast User Guide*. Reading: ECMWF. doi: 10.21957/m1cs7h, 2018.
- [20] Pan, S. J., & Yang, Q. A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10), 1345-1359, 2009.
- [21] Qureshi, A. S., Khan, A., Zameer, A., & Usman, A. *Wind power prediction using deep neural network based meta regression and transfer learning*. Applied Soft Computing, 58, 742-755, 2017.
- [22] Ruder, Sebastian. *An overview of multi-task learning in deep neural networks*. arXiv preprint arXiv:1706.05098 (2017).
- [23] Rumelhart, D. E. and Hinton, G. E. and Williams, R. J.: *Learning Internal Representations by Error Propagation*. In D. E. Rumelhart and J. L. McClelland and the PDP Research Group (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, MIT Press, pp. 318-362, 1986.
- [24] Sak, H. and Senior, A. and Beaufays, F.: *Long short-term memory recurrent neural network architectures for large scale acoustic modeling*. INTERSPEECH, pp. 338-342, 2014.
- [25] Schreiber, J. *Transfer Learning in the Field of Renewable Energies—A Transfer Learning Framework Providing Power Forecasts Throughout the Lifecycle of Wind Farms After Initial Connection to the Electrical Grid*. arXiv preprint arXiv:1906.01168, 2019.
- [26] Snoek, J., H. Larochelle, R. P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. <https://arxiv.org/abs/1206.2944>, 2012.