Appendix A. Data

The following section details challenges with the different datasets. We also include a description on the overall process for generating power forecasts for a wind or photovoltaic (PV) park. Fig. Appendix A.1 summarizes this process. Due to the weather dependency of renewable power plants, we require weather predictions from so called numerical weather prediction (NWP) models. The NWP model has input from sensors that approximate the current weather situation. Based on the last sensory data, a so-called model run is calculated. This model run is, e.g., 0.00 am. Due to complex and manifold stochastic differential equations involved in predicting the weather, such a model run typically requires about six hours until it is finished. Afterward, the NWP provides forecasts, e.g., up to 72 hours into the future. In our case, we are interested in so-called day-ahead power forecasts based on weather forecasts between 24 and 48 hours into the future. Based on these weather forecasts and historical power measurements, we can train a machine learning (ML) model to predict the expected power generation for day-ahead forecasting problems.

However, due to the dependency of renewable power forecasts on weather forecasts as input features, substantial uncertainty is associated with these forecasts making it a challenging problem. At the same time, weather forecasts are valid for larger grid sizes, e.g., three kilometers and a mismatch between these grids and the location of a wind or PV causes additional uncertainty in the power forecasts. These mismatches and the non-linearity of the forecasting problem are visually accessible in Fig. Appendix A.2. We can observe various mismatches between the predicted wind speed (or radiation) with historical power measurements. For instance, we can observe (outliers) where a large amount of power is generated for low values of those features. Those mismatches are also visually accessible in the time-series plots in Fig. Appendix A.3 and Appendix A.4. This observation indicates that the weather forecasts were wrong. Various examples are also present where we observe a large value of wind speed or radiation, but no or little power is generated. A wrong weather forecast can cause this problem. However, often, it is associated with regular interventions. For instance, in some regions in Germany, wind turbines need to limit the rotation speed during the night. Also, there is a large portion of feed-in management interventions in Germany. These interventions are used to stabilize the electrical grid. A typical pattern for those interventions is given in Fig. Appendix A.4c. We can observe an initial large power production associated with large wind speed values. At this point, the power generation drops to zero, while the wind speed remains high. Such a sudden drop is typically associated with feed-in management interventions. As those interventions depend on the power grid's state, we typically have no information about such drops, making the forecasting problem even more challenging.

Appendix B. Method

The following sections detail proposed methods and utilized approaches.

Appendix B.1. Coopetitive Soft Gating Ensemble for Model Combination

To adapt source models from a model hub for a target, it is essential that a learning strategy only adapts relevant knowledge for the target. For instance, a learning strategy should assure that a model is not adapted to seasonal specific behavior when only limited data of a specific season is available. One way to achieve this is through ensembles. The Bayesian model averaging (BMA) ensemble, for instance, allows combining source models by their marginal likelihood on the target. However, in practice, Bayesian models are often not available. While the BMA combines models through the posterior, we propose to utilize the coopetitive soft gating ensemble (CSGE), which combines ensemble members based on the error scores. Recently, the CSGE has proven successful in renewable power [48, 34, 47] and trajectory forecasts [49]. The following section details this method in the context of inductive transfer learning (ITL).

Ensemble approaches are categorized as either weighting or gating strategies to combine ensemble members. In the case of a gating strategy, a single ensemble member provides a prediction and other source models are neglected. In the case of a weighting strategy, several ensemble members are combined. The CSGE incorporates both principles through the soft gating principle [50]. The idea of the CSGE is to link the weights to the ensemble members' performance, i.e., good source models are weighted stronger than weaker ones. The overall weight of a source model for the target task is characterized using three aspects:

- The *global weight* is defined by how well a source model performs with the available training data on the target task.
- The *local weight* is defined by how well a source model performs on the target tasks for different areas in the feature space. For example, in the case of wind, one model might performs well for low wind speeds, while another source model might performs well for larger wind speeds on the target.
- The *forecast horizon-dependent weight* is defined by how well a source model performs for different lead times on the target task.

Fig. 3 provides an overview of the CSGE. In the following, we treat ensemble members equivalently to source models from a model hub. The CSGE includes M ensemble members, with $m \in \{1, ..., M\}$. Each of these ensemble members is a source model with a predictive function f_m. Each source model forecasts uni-variate estimates $\hat{y}^{(m)}_{t+k|t} \in \mathbb{R}$ for the input $\mathbf{x}_{t+k|t} \in \mathbb{R}^D$ of a target task T, we omit the subscript T for reasons of clarity and comprehensibility. Let D be the dimension of the input feature vector $\mathbf{x}_{t+k|t}$. Then, k denotes the forecast horizon, denoted by the subscript, for the forecast origin t. Furthermore, let $\hat{y}^{(m)}_{t+k|t}$ the m-th source model's prediction. For each prediction of each source model, we compute an aggregated weight $w^{(m)}_{t+k|t}$, based on the global, local, and forecast horizon-dependent weighting. Finally, the weight and the prediction of each model for each forecast horizon are aggregated as a final prediction through

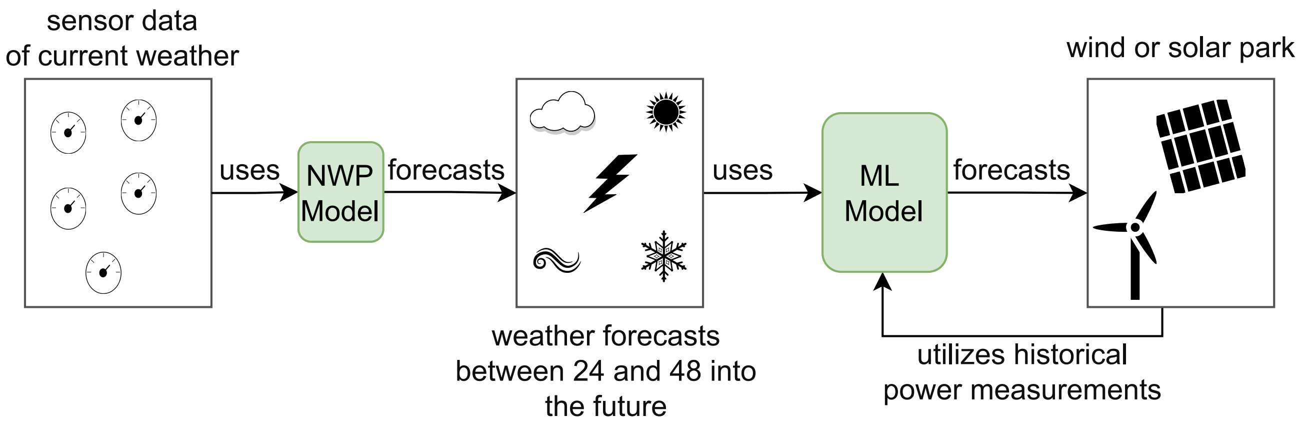


Figure Appendix A.1: Overview of renewable power forecast process.

$$\bar{\bar{y}}_{t+k|t} = \sum_{m=1}^{M} w_{t+k|t}^{(m)} \cdot \hat{y}_{t+k|t}^{(m)}, \qquad \text{(Appendix B.1)}$$

with the following constraints

$$\sum_{m=1}^{M} w_{t|t+k}^{(m)} = 1 \qquad \forall k \in \mathbb{N}_{\geq 1}. \qquad \text{(Appendix B.2)}$$

The constraints ensure that the weights of all M ensemble members, where $w_{t|t+k}^{(m)}$ denotes the weight of the m-th ensemble member's prediction with lead time k, are properly normalized. The weight incorporating the global, local, and forecast horizon-dependent weight, for a single source model and lead time, is given by

$$\bar{w}_{t+k|t}^{(m)} = w_{g}^{(m)} \cdot w_{l}^{(m,t)} \cdot w_{h}^{(m,k)}. \qquad \text{(Appendix B.3)}$$

We achieve a normalization of $\bar{w}_{t+k|t}^{(m)}$ through the division of the sum of all weights for all source models and lead times:

$$w_{t+k|t}^{(m)} = \bar{w}_{t+k|t}^{(m)} / \sum_{\tilde{m}=1}^{M} \bar{w}_{t+k|t}^{(\tilde{m})}. \qquad \text{(Appendix B.4)}$$

Appendix B.1.1. Soft Gating Principle

The soft gating principle to calculate a source model's weight has several theoretical advantages in the context of an ensemble for a target task from a model hub. These advantages are all due to the compromise between the weighting and the gating principle of the CSGE. On the one hand, we want gating, as properly many source models do not apply to a target task and should therefore not be considered. On the other hand, those applicable source models should be combined through weighting. In a sense, this is comparable to an BMA approach, where models with a low posterior are neglected (gated), while those with a larger posterior get a larger weight.

To calculate the weights $w_{t+k|t}^{(m)}$, we utilize the definition of the inverse similarity measurement $\mathcal{S}^{-1}$ from Section 3.3 and the coopetitive soft gating principle from Eq. (Appendix B.5). By calculating the weighting through the inverse $\mathcal{S}^{-1}$, here the normalized root mean squared error (nRMSE), we directly measure how well a source model performs on the target. Therefore, let us assume that $\mathbf{\Phi} \in \mathbb{R}^{J}$ contains all $J \in \mathbb{N}_{\geq 1}$ estimates based on the nRMSE and ϕ is an arbitrary element from $\mathbf{\Phi}$. Then, $\eta \geq 0$ depicts the amount of exponential weighting and the small constant $\epsilon > 0$ avoids division by zero. For greater η, the CSGE tends to work as a gating ensemble and, thereby, considering only a few source models. For smaller η result in a weighting ensemble. After calculating all weights from $\mathbf{\Phi}$ through Eq. (15), we normalize the results to sum up to one estimating the final weights.

$$\varsigma_{\eta}^{'}(\mathbf{\Phi}, \phi) = \frac{\sum_{j=1}^{J} \mathbf{\Phi}_{j}}{\phi^{\eta} + \epsilon}, \qquad \text{(Appendix B.5)}$$

Finally, to assure that calculated weights for each ensemble member are normalized, we use the function $\varsigma_{\eta} : (\mathbb{R}^{+})^{M+1} \rightarrow [0, 1]$ with

$$\varsigma_{\eta}(\mathbf{\Phi}, \phi) = \frac{\varsigma_{\eta}^{'}(\mathbf{\Phi}, \phi,)}{\sum_{m=1}^{M} \varsigma_{\eta}^{'}(\mathbf{\Phi}, \phi_{m})} \qquad \text{(Appendix B.6)}$$

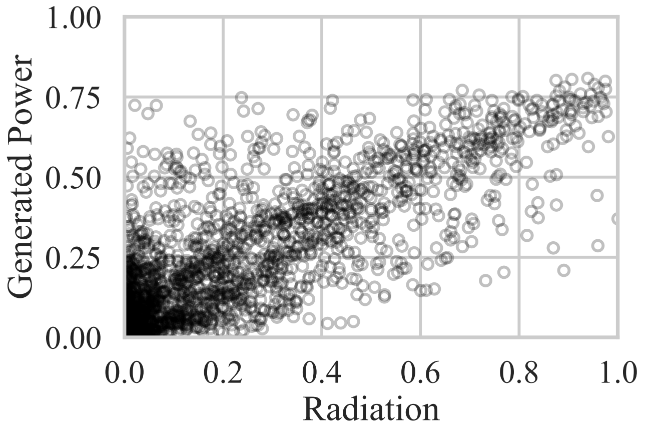
Appendix B.1.2. Global Weighting

The global weight expresses the overall performance of the different source models on the target based on the available training data. The average $r^{(m)} \in \mathbb{R}_{\geq 1}$ of a single source model is given by

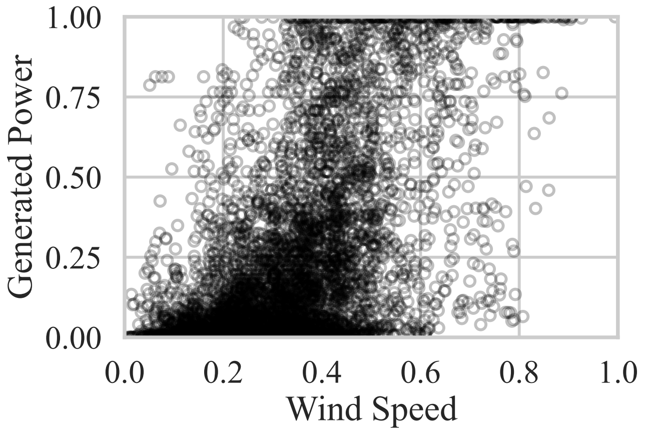
$$r^{(m)} = \frac{1}{N} \cdot \sum_{n=1}^{N} \mathcal{S}^{-1}(\hat{y}_{n}^{(m)}, o_{n}), \qquad \text{(Appendix B.7)}$$

where N refers to the number of available training data, $\mathcal{S}^{-1}$ is the inverse similarity for the prediction $\hat{y}_{n}^{(m)}$ and the respective observations o_{n}. Then, the set of all errors, from all source models, is given by $R = (r^{(1)}, \ldots, r^{(m)}, \ldots, r^{(M)})$. This set allows us to calculate the global weight by

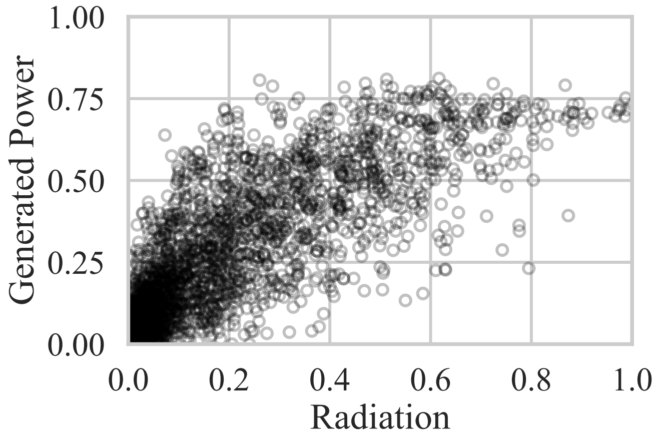
$$w_{g}^{(m)} = \varsigma_{\eta_{g}}(R, r^{(m)}), \qquad \text{(Appendix B.8)}$$



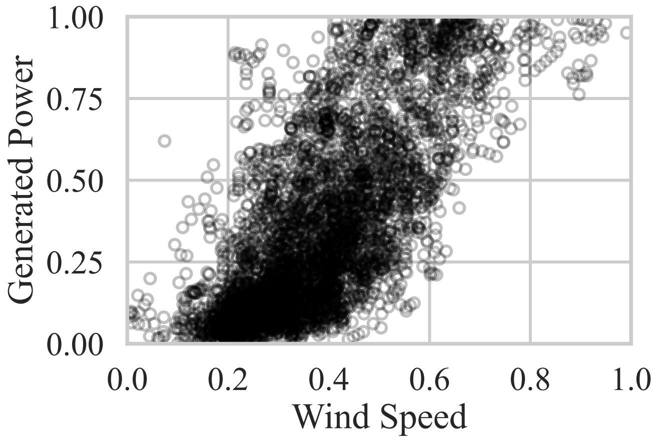
(a) PVSYN datastet.



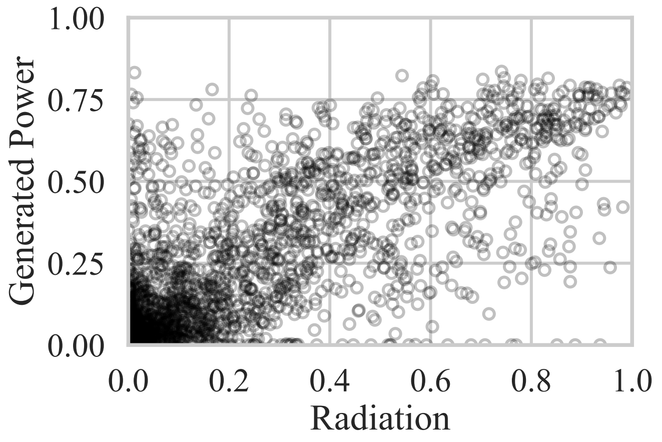
(b) WINDSYN datastet.



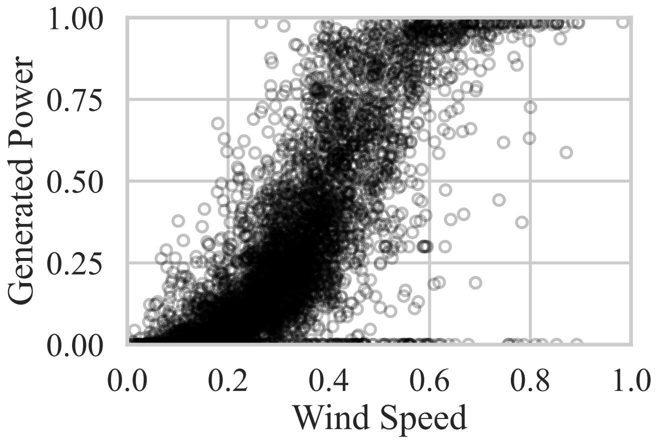
(c) PVOPEN datastet.



(d) WINDOPEN datastet.

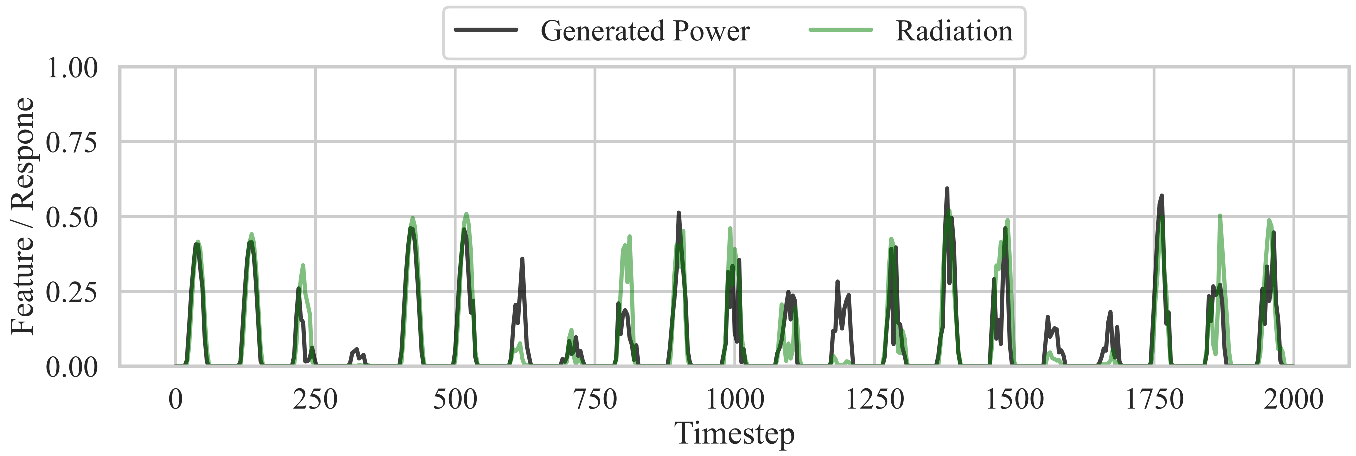


(e) PVREAL datastet.

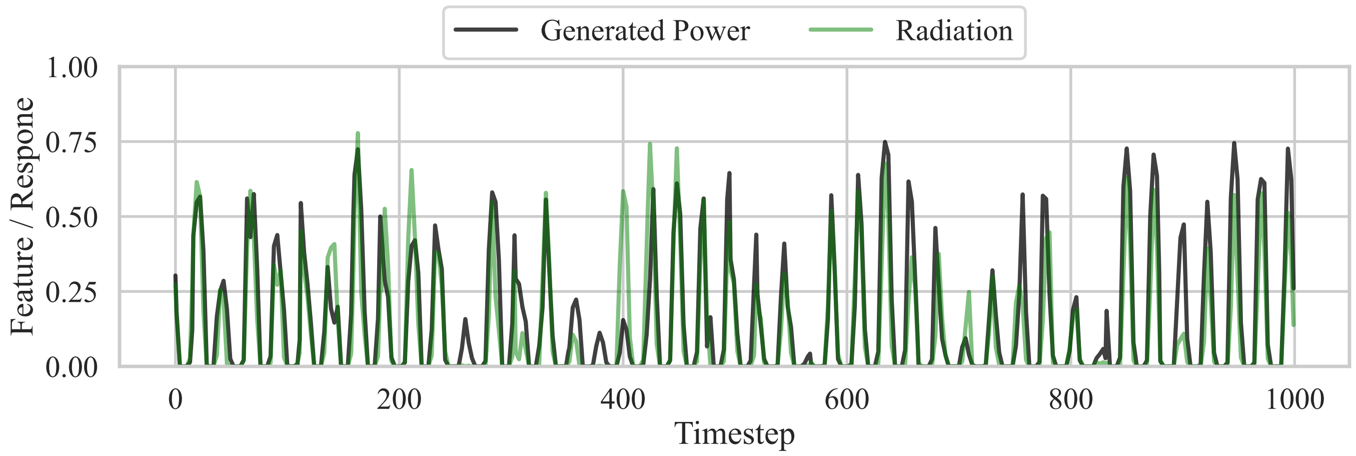


(f) WINDREAL datastet.

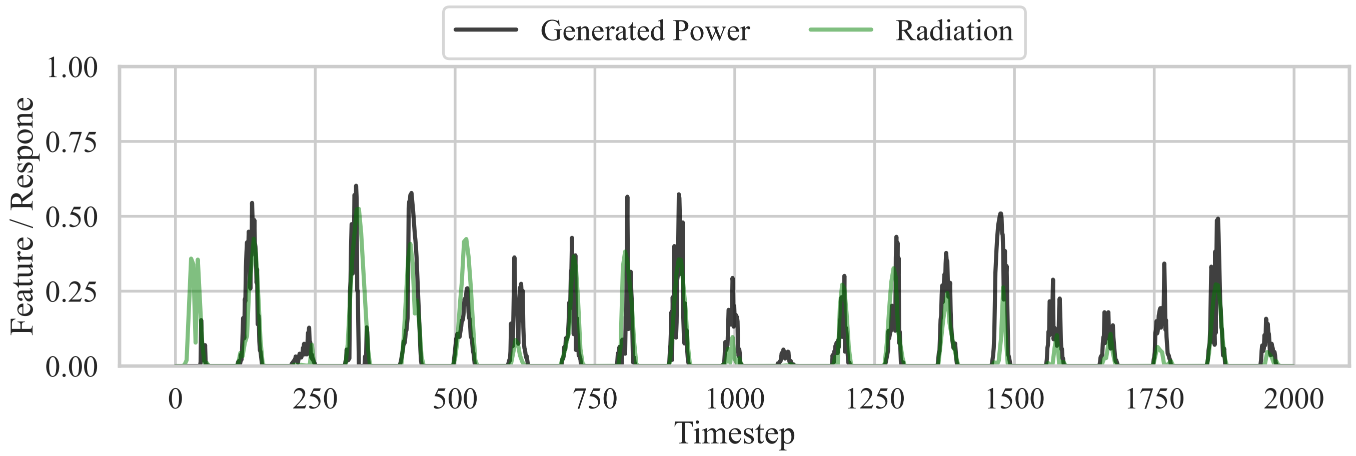
Figure Appendix A.2: Scatter plots of most relevant features for power forecasts from day-ahead weather forecasts and the historical power measurements for an exemplary park from all datasets.



(a) PVSYN dataset.

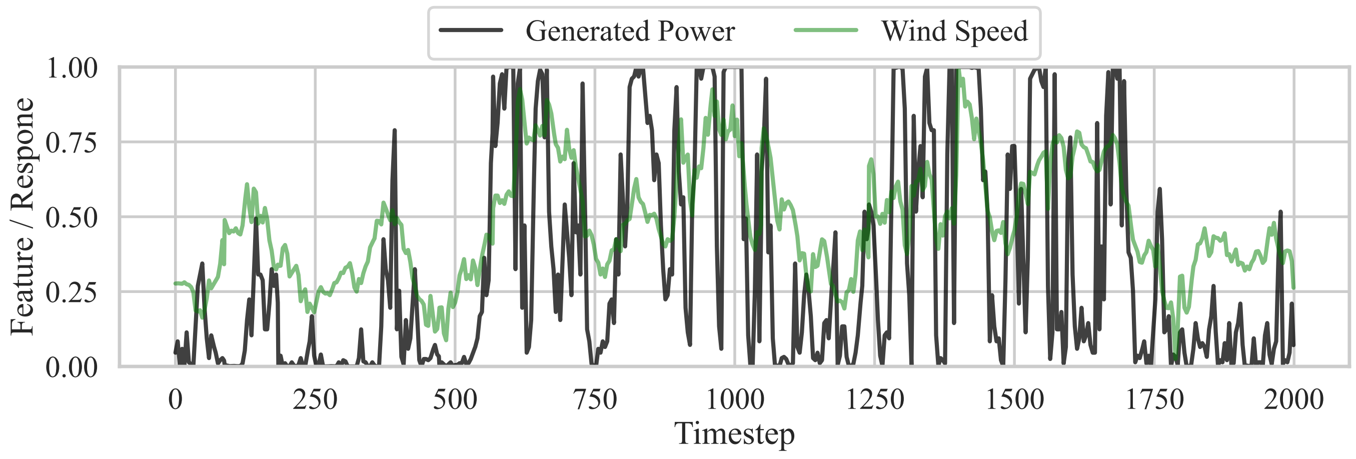


(b) PVOPEN dataset.

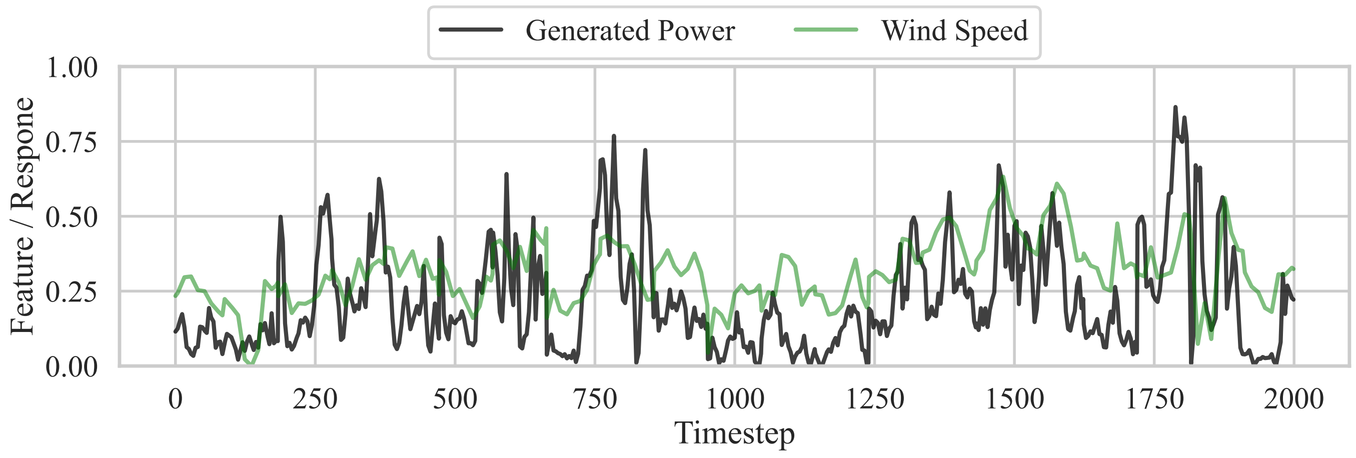


(c) PVREAL dataset.

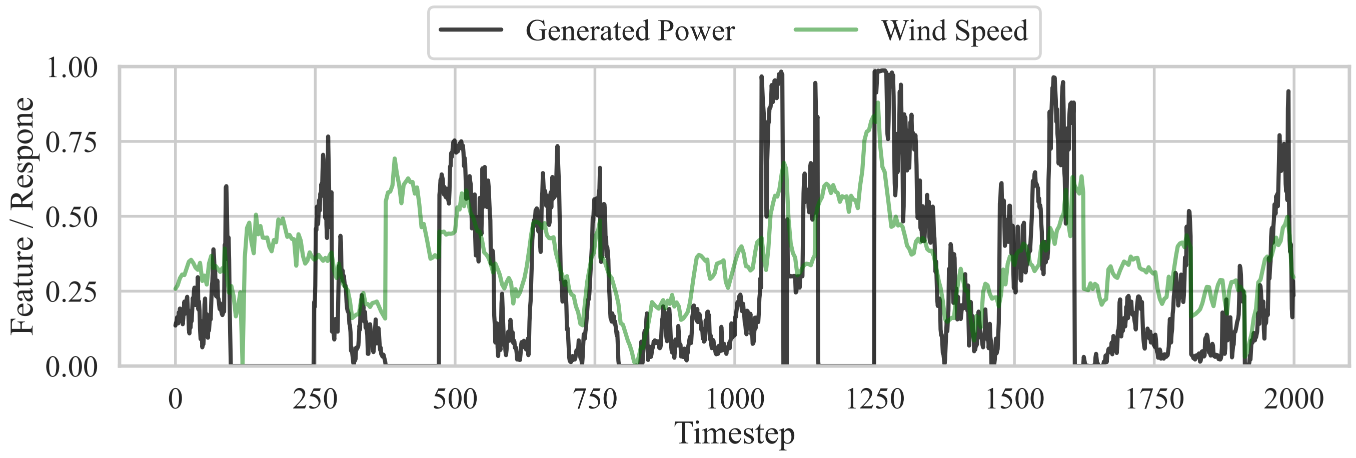
Figure Appendix A.3: Timeseries plots of PV datasets.



(a) WINDSYN dataset.



(b) WINDOPEN dataset.



(c) WINDREAL dataset.

Figure Appendix A.4: Timeseries plots of wind datasets.

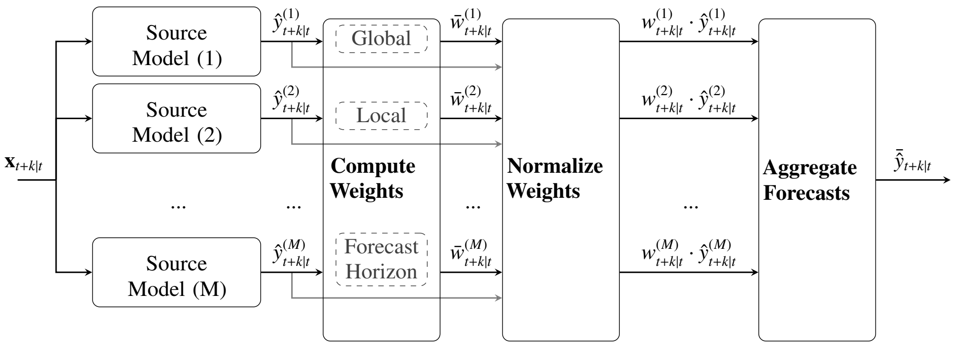


Figure Appendix B.1: The architecture of the CSGE. The source models' predictions $\hat{y}^{(m)}_{t+k|t}$ for the input **x** are passed to the CSGE. The ensemble member's weights are given by the aggregation of the respective global-, local- and forecast horizon-dependent weights. The weights are normalized. In the final step, the source models' predictions are weighted and aggregated.

with the hyperparameter η_g for the soft gating formula.

Appendix B.1.3. Local Weighting

The local weighting expresses how well a source model performs for different areas in the feature space. For instance, one source model might perform well on the target for cloudy days while another performs well for sunny days. To calculate an expectation of the error under certain conditions in the input features **x**, we train a regression model for each source model based on the available training data. Once the regression model is trained, this function with $L^{(m)} : \mathbb{R}^D \rightarrow \mathbb{R}^+$ lets us predict the expected error $\hat{q}^{(m)}_{t+k|t}$ for each ensemble member and each k with

$$\hat{q}^{(m)}_{t+k|t} = L^{(m)}(\mathbf{x}_{t+k|t}). \qquad \text{(Appendix B.9)}$$

Similar, to the global weighting, the set of all local errors

$$Q = (\hat{q}^{(1)}_{t+k_{\min}|t}, \ldots, \hat{q}^{(m)}_{t+k|t}, \ldots, \hat{q}^{(M)}_{t+k_{\max}|t}),$$

$k_{\min}$ refers to the smallest forecast horizon and $k_{\max}$ refers to the maximum forecast horizon, lets us calculate the local weight $w^{(j)}_l$ with

$$w^{(t,k)}_l = \varsigma_{\eta_l}(Q, \hat{q}^{(m)}_{t+k|t}), \qquad \text{(Appendix B.10)}$$

where η_l is the hyperparameter of the soft gating formula for the local weighting.

Appendix B.1.4. Forecast horizon-Dependent Weighting

The time-dependent weighting considers that ensemble members may perform differently for different forecast horizons k, e.g., some source models might be better in the short-term while others are better in the long term. Similar to global weighting, we calculate this weighting during training. But instead of averaging across all training samples, we calculate the mean with

$$p^{(m)}_k = \frac{1}{N_k} \cdot \sum_{n_k=1}^{N_k} \mathcal{S}^{-1}(\hat{y}^{(m)}_{n_k}, o_{n_k}) \qquad \text{(Appendix B.11)}$$

to obtain the forecast horizon-dependent error $p^{(m,k)}$, where N_k refers to the number of samples for a forecast horizon k. All errors of all ensemble member are the combined through the set $P_k = (p^{(1)}_k, \ldots, p^{(m)}_k, \ldots, p^{(M)}_k)$ to calculate the time-dependent weight through

$$w^{(m,k)}_{lt} = \varsigma_{\eta_{lt}}(P_k, p^{(m)}_k). \qquad \text{(Appendix B.12)}$$

The forecast horizon-dependent hyperparameter is here denoted with η_{lt}.

Appendix C. Experiment

In the following, we detail our hyper-parameters for model training, compare them to a physical model for the REAL datasets, provide additional evaluations and example forecasts for the most promising methods.

Appendix C.1. Hyperparameter Tuning of Multi-Layer Perceptron and Temporal Convolution Network Source Models

As source models for the model hub, we utilize an multi-layer perceptron (MLP) and a temporal convolution network (TCN). The layers' initialization strategy is the default one of the utilized libraries (*pytorch*[1], *fastai*[2]). For all source models, we conducted a hyperparameter search through an tree-structured Parzen sampler for 200 samples through the *optuna*[3] library. The best hyperparameters are selected based on the smallest nRMSE on the validation dataset. We initially transform the model's number of input features in a higher dimension through a factor $k \in \{1, \ldots, 20\}$. Note that initially transforming the input features in a higher-dimensional space typically improves the performance [51]. Afterward, the features are reduced by 50 percent in each layer to a minimum of 11 before the final output.

[1]`https://pytorch.org/docs/stable/index.html`, accessed 2021-02-28
[2]`https://docs.fast.ai/`, accessed 2021-02-28
[3]`https://optuna.org/`, accessed 2021-02-28

The final two layers of all networks have sizes 3 and 1. The learning rate of the *adam* optimizer is sampled from a logarithmic uniform distribution $\mathcal{U}(\ln(10^{-6}), \ln(10^{-1}))$. For each model, we sample the number of training epochs from the set $\{10, \ldots, 100\}$. The dropout is sampled from $\mathcal{U}(0, 0.9)$ and the weight decay from $\{0.0, 0.1, 0.2, 0.4\}$. For the MLP the batch size is sampled from the set $\{1024, 2048, 4096\}$. In the case of the TCN network architecture, these batch sizes are divided by the number of samples per day, 24 for the PVOPEN dataset and 96 for the remaining datasets.

Appendix C.2. Hyperparameter Tuning Bayesian Extreme Learning Machine Source Models

The Bayesian extreme learning machine (BELM) is optimized by sampling 200 times through the tree-structured Parzen sampler and the best parameters are selected by the largest evidence on the validation data. The number of hidden neurons is from $\mathcal{U}(10, 1000)$ to generate the random features. As an activation function for the random features, one of Rectified Linear Unit (ReLU), sigmoid, ReLU and sigmoid is sampled. In the latter case, the results of the two activation functions are concatenated to generate a maximum of 2000 random features. Additionally to these random features, we give the hyperparameter tuning the option to include the original NWP features. Alpha and beta of the Bayesian linear regression are optimized through empirical Bayes [44].

Appendix C.3. Gradient Boosting Regression Tree Target Model

To have a strong baseline that generalizes well with the limited amount of data and is known to mitigate the effects of overfitting, we train a gradient boosting regression tree (GBRT) for each target model. We optimize the GBRT for each park, season, and number of training data through a grid search by three-fold cross-validation. Thereby, the learning rate is evaluated at $[10^{-6}, 3.1 \times 10^{-6}, \ldots, 3.1 \times 10^{-1}, 1]$. The number of estimators is 300 and the maximum depth is one of $[2, 4, 6, 8]$. Other values are the default ones of scikit-learn[4].

Appendix C.4. Target Training Extreme Learning Machine

: For the adaptation of the BELM model, we assume that given an appropriate model selection strategy, the prior α and β of the Bayesian linear regression is also valid for the target. In principle, this is a relatively strong assumption. While the aleatoric uncertainty might be the same between a source and a target, the knowledge of the source model for the target is low and therefore, the epistemic uncertainty should be high. We argue that the problem is reduced as we update the mean vector and the precision matrix, given the available target data through Eq. (2), where the previous posterior, from the source model, acts as prior for the target. Another interesting consideration is that the insufficient evidence of the source model on the target requires that an update of the source model needs a large amount of target data which contracts to the training from the source model. Therefore, it reduces the risk of catastrophic forgetting.

We first update all source models by the given target data to select a source model. We train and select the source model with all the available data for the evidence selection strategy. For the nRMSE selection, we use 70% of the data for training and the remaining data for the selection.

Appendix C.5. Target training of Multi-Layer Perceptron and Temporal Convolution Network

As pointed out earlier, we deploy different model selection and adaptation strategies, as summarized in Table 1. In case we fine-tune a source model, e.g., through weight decay, we conduct a grid search for different hyperparameters and select the best one based on the lowest nRMSE through 30% of the training data as a validation dataset. The learning rate and the amount of penalty λ are optimized for all adaptation strategies, as detailed earlier in this section. For all optimizations, we use the stochastic gradient descent algorithm without momentum as suggested in [11] for transfer learning (TL). For the Bayesian tuning, we select the ten most relevant models selected through the evidence for regularization. In all cases, the batch size is selected to have about ten iterations within an epoch and we train for a single epoch to limit the amount of training time.

For the direct adaptation, we utilize the source model without any adaptation. In the case of the direct linear adaptation, we replace the final layer through a Bayesian linear regression (BLR) model(s). These BLRs are trained through empirical Bayes. For the TCN source model, we train one BLR for each forecast horizon based on the extracted features from the source model from a specific forecast horizon and its response.

Appendix C.6. Model Selection and Adaptation

Previously in Section 4, we discussed the best model selection and adaptation strategies through the mean rank. Table Appendix C.1 and Appendix C.2 lists those combination, from the overview in Table 1, previously not discussed. We additionally include the *TCN-EV-WD* model, as it allows for a better comparison across all evaluated fine-tuning techniques. We calculate the mean rank separate from the best models to better quantify the quality within the worse models. Again, we test for a significant improvement compared to the GBRT baseline . We evaluate the influence of available training data on the different strategies.

For all datasets, we observe that TCN based models have a smaller rank than MLP based models and adaptations. For both datasets, only one model is included that is not considering fine-tuning, the *MLP-EV-DI* model, which is not adapted to the target. This model is the best for PV datasets with limited data for up to 14 days. Additional results for the PV datasets can be summarized as follows:

- The MLP model has the best results when the weight decay source adaptation strategy is utilized for PV datasets. A weight decay concerning the origin has a larger rank in most cases.

[4]`https://scikit-learn.org/`, accessed 2022-02-28

- For the PVOPEN dataset, models are not significantly better than the baseline with more than 14 days of training data.
- For PVSYN and PVREAL , the TCN model with the evidence selection strategy and weight decay source adaptation leads to the best significant improvements.
- The Bayesian tuning is among the best for the TCN model but leads to one of the worst results for the MLP model.

The results for the wind datasets can be summarized as follows

- For the MLP and the selection through nRMSE there is no clear winner of the adaptation strategies.
- For the MLP and the selection through nRMSE with limited data, the weight decay source is preferable; with more data, the weight decay leads to better results.
- For the MLP and the adaptation through Bayesian tuning, results are worse than those from other selection and adaptation strategies.
- For the TCN, the Bayesian tuning has the best results.
- The second best technique for the TCN is selecting through the evidence and a weight decay with respect to the origin.
- No model, selection, and adaptation is superior to the baseline with 90 days of training data.

Generally, we can observe that for the MLP model, a selection through the nRMSE is preferable over the evidence, regardless of the fine-tuning technique. The opposite is the case for the TCN model. Here, we observe that in most cases, it is preferable to select a source model based on the evidence instead of the nRMSE This observation suggests that the nRMSE correlates better with the transferability of the final layer than for the evidence for the MLP. For the TCN, the final layer is a residual block and the evidence better captures this complexity.

Table Appendix C.1: Rank summary for different source models, selections, and adaptation strategies on the PV datasets. Only those that are *not* within the top ranks for a dataset are included. GBRT is the baseline and all models are tested if the forecasts error is significantly better ($\vee$), worse ($\wedge$), or not significantly different ($\diamond$). EV indicates the model selection by the evidence, RM the one by the nRMSE. DI shows that a source model is directly applied to the target without any adaptation. DILI indicates the adaptation though the BLR, WD indicates fine-tuning regularized by weight decay w.r.t. the origin, WDS indicates fine-tuning regularized by weight decay w.r.t. the source models parameter, and BT indicates Bayesian tuning.

Data Type	#Days	**Baseline**	**MLP-EV-BT**	**MLP-EV-WD**	**MLP-EV-WDS**	**MLP-RM-WD**	**MLP-RM-WDS**	**MLP-EV-BT**	**TCN-EV-WDS**	**TCN-RM-WDS**
PVOPEN	7	4.464	5.857$_\wedge$	5.679$_\wedge$	5.964$_\wedge$	5.5$_\wedge$	5.417$_\wedge$	**3.714**$_\vee$	4.071$_\diamond$	4.131$_\diamond$
PVREAL	7	7.179	5.298$_\vee$	5.649$_\vee$	5.22$_\vee$	5.804$_\vee$	5.637$_\vee$	**3.196**$_\vee$	3.405$_\vee$	3.274$_\vee$
PVSYN	7	6.75	5.958$_\vee$	5.726$_\vee$	5.7$_\vee$	5.11$_\vee$	5.151$_\vee$	**3.224**$_\vee$	3.559$_\vee$	3.45$_\vee$
PVOPEN	14	4.571	5.262$_\diamond$	4.94$_\diamond$	4.845$_\diamond$	4.31$_\diamond$	**4.131**$_\diamond$	5.524$_\wedge$	5.488$_\diamond$	5.929$_\wedge$
PVREAL	14	6.994	4.798$_\vee$	4.78$_\vee$	4.476$_\vee$	5.613$_\vee$	5.464$_\vee$	4.292$_\vee$	4.268$_\vee$	**4.149**$_\vee$
PVSYN	14	6.004	5.647$_\diamond$	5.414$_\vee$	5.279$_\vee$	4.748$_\vee$	4.673$_\vee$	**4.171**$_\vee$	4.529$_\vee$	4.406$_\vee$
PVOPEN	30	**3.881**	5.774$_\wedge$	5.429$_\wedge$	5.536$_\wedge$	4.667$_\diamond$	5.036$_\wedge$	4.821$_\diamond$	4.714$_\diamond$	5.143$_\wedge$
PVREAL	30	6.643	4.839$_\vee$	5.065$_\vee$	4.935$_\vee$	5.881$_\vee$	5.744$_\vee$	**3.726**$_\vee$	4.113$_\vee$	4.03$_\vee$
PVSYN	30	5.371	6.048$_\wedge$	5.939$_\wedge$	5.711$_\diamond$	5.114$_\diamond$	5.02$_\diamond$	**3.748**$_\vee$	4.072$_\vee$	3.963$_\vee$
PVOPEN	60	**4.036**	5.024$_\wedge$	5.5$_\wedge$	5.393$_\wedge$	4.905$_\wedge$	4.952$_\wedge$	4.69$_\diamond$	5.333$_\wedge$	5.083$_\diamond$
PVREAL	60	6.482	4.946$_\vee$	5.411$_\vee$	5.202$_\vee$	6.03$_\diamond$	5.857$_\vee$	**3.548**$_\vee$	3.613$_\vee$	3.911$_\vee$
PVSYN	60	4.933	6.291$_\wedge$	6.217$_\wedge$	5.953$_\wedge$	5.2$_\diamond$	5.047$_\diamond$	**3.502**$_\vee$	3.93$_\vee$	3.922$_\vee$
PVOPEN	90	**3.417**	5.583$_\wedge$	5.917$_\wedge$	5.619$_\wedge$	5.107$_\wedge$	4.726$_\wedge$	5.357$_\wedge$	4.214$_\diamond$	5.06$_\wedge$
PVREAL	90	6.351	5.071$_\vee$	5.262$_\vee$	5.321$_\vee$	5.804$_\diamond$	5.857$_\diamond$	**3.649**$_\vee$	3.851$_\vee$	3.833$_\vee$
PVSYN	90	4.722	6.278$_\wedge$	6.135$_\wedge$	6.2$_\wedge$	5.176$_\wedge$	5.138$_\diamond$	**3.705**$_\vee$	3.87$_\vee$	3.776$_\vee$

Table Appendix C.2: Rank summary for the wind datasets. Compare Table Appendix C.1.

Data Type	#Days	**Baseline**	**MLP-EV-BT**	**MLP-EV-WD**	**MLP-EV-WDS**	**MLP-RM-WD**	**MLP-RM-WDS**	**MLP-EV-BT**	**TCN-EV-WDS**	**TCN-RM-WDS**
WINDOPEN	7	6.012	5.532$_\diamond$	5.237$_\vee$	5.272$_\vee$	5.202$_\vee$	5.006$_\vee$	**3.971**$_\vee$	4.358$_\vee$	4.162$_\vee$
WINDREAL	7	7.18	5.62$_\vee$	5.377$_\vee$	5.426$_\vee$	4.94$_\vee$	4.927$_\vee$	3.797$_\vee$	**3.736**$_\vee$	3.788$_\vee$
WINDSYN	7	6.615	5.114$_\vee$	4.751$_\vee$	4.745$_\vee$	5.041$_\vee$	5.0$_\vee$	**4.391**$_\vee$	4.5$_\vee$	4.751$_\vee$
WINDOPEN	14	5.22	5.497$_\diamond$	5.416$_\diamond$	5.191$_\diamond$	5.156$_\diamond$	5.289$_\diamond$	**4.254**$_\vee$	4.653$_\diamond$	4.266$_\vee$
WINDREAL	14	6.188	5.402$_\vee$	5.341$_\vee$	5.301$_\vee$	4.685$_\vee$	4.691$_\vee$	**4.327**$_\vee$	4.535$_\vee$	4.504$_\vee$
WINDSYN	14	6.092	4.882$_\vee$	4.673$_\vee$	**4.63**$_\vee$	4.952$_\vee$	4.935$_\vee$	4.857$_\vee$	4.879$_\vee$	5.099$_\vee$
WINDOPEN	30	**3.892**	5.97$_\wedge$	5.443$_\wedge$	5.479$_\wedge$	5.671$_\wedge$	5.599$_\wedge$	4.072$_\diamond$	4.353$_\diamond$	4.479$_\diamond$
WINDREAL	30	5.03	5.657$_\wedge$	5.661$_\wedge$	5.753$_\wedge$	5.36$_\wedge$	5.359$_\wedge$	**3.915**$_\vee$	4.161$_\vee$	4.106$_\vee$
WINDSYN	30	5.063	5.076$_\diamond$	4.963$_\diamond$	5.074$_\diamond$	5.327$_\diamond$	5.348$_\diamond$	4.731$_\diamond$	**4.652**$_\vee$	4.766$_\vee$
WINDOPEN	60	**3.316**	6.02$_\wedge$	5.895$_\wedge$	5.849$_\wedge$	5.836$_\wedge$	5.789$_\wedge$	4.033$_\wedge$	4.197$_\wedge$	4.066$_\wedge$
WINDREAL	60	4.087	5.774$_\wedge$	5.848$_\wedge$	6.004$_\wedge$	5.401$_\wedge$	5.33$_\wedge$	**4.068**$_\diamond$	4.268$_\diamond$	4.219$_\diamond$
WINDSYN	60	4.475	5.427$_\wedge$	5.359$_\wedge$	5.519$_\wedge$	5.295$_\wedge$	5.371$_\wedge$	4.569$_\diamond$	**4.375**$_\diamond$	4.61$_\diamond$
WINDOPEN	90	**2.914**	6.367$_\wedge$	6.328$_\wedge$	6.188$_\wedge$	5.617$_\wedge$	5.445$_\wedge$	3.727$_\wedge$	4.141$_\wedge$	4.273$_\wedge$
WINDREAL	90	**3.577**	5.936$_\wedge$	6.249$_\wedge$	6.452$_\wedge$	5.556$_\wedge$	5.45$_\wedge$	3.878$_\wedge$	4.033$_\wedge$	3.869$_\diamond$
WINDSYN	90	**3.75**	5.644$_\wedge$	5.568$_\wedge$	5.76$_\wedge$	5.289$_\wedge$	5.333$_\wedge$	4.703$_\wedge$	4.339$_\wedge$	4.614$_\wedge$

References

[1] J. Schreiber, 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, in: Organic Computing - Doctoral Dissertation Colloquium, kassel university press GmbH, 2019, pp. 75–87.

[2] R. Schwartz, J. Dodge, N. A. Smith, et al., Green AI, CoRR (2019) 1–12ArXiv: 1907.10597.

[3] J. Schreiber, S. Vogt, B. Sick, Task Embedding Temporal Convolution Networks for Transfer Learning Problems in Renewable Power Time-Series Forecast, in: ECML, 2021, pp. 1–16. doi:10.1007/978-3-030-86514-6_8.

[4] K. You, Y. Liu, J. Wang, et al., LogME: Practical Assessment of Pre-trained Models for Transfer Learning, in: ICML, 2021, pp. 12133–12143. arXiv:2102.11005.

[5] K. You, Y. Liu, J. Wang, et al., Ranking and Tuning Pre-trained Models: A New Paradigm of Exploiting Model Hubs, CoRR arXiv:2110.10545 (2021) 1–45. arXiv:2110.10545, doi:10.48550/arXiv.2110.10545.

[6] G. Alkhayat, R. Mehmood, A review and taxonomy of wind and solar energy forecasting methods based on deep learning, Energy and AI 4 (2021) 100060. doi:10.1016/j.egyai.2021.100060.

[7] S. Vogt, J. Schreiber, Synthetic Photovoltaic and Wind Power Forecasting Data, CoRR arXiv:2204.00411 (2022).

[8] X.-W. Zheng, H.-N. Li, P. Gardoni, Hybrid bayesian-copula-based risk assessment for tall buildings subject to wind loads considering various uncertainties, Reliability Engineering and System Safety 233 (2023) 109100. doi:https://doi.org/10.1016/j.ress.2023.109100.

[9] M. Alruqi, P. Sharma, B. Deepanraj, F. Shaik, Renewable energy approach towards powering the ci engine with ternary blends of algal biodiesel-diesel-diethyl ether: Bayesian optimized gaussian process regression for modeling-optimization, Fuel 334 (2023) 126827. doi:https://doi.org/10.1016/j.fuel.2022.126827.

[10] Z. Said, P. Sharma, L. Syam Sundar, V. G. Nguyen, V. D. Tran, V. V. Le, Using bayesian optimization and ensemble boosted regression trees for optimizing thermal performance of solar flat plate collector under thermosyphon condition employing mwcnt-fe3o4/water hybrid nanofluids, Sustainable Energy Technologies and Assessments 53 (2022) 102708. doi:https://doi.org/10.1016/j.seta.2022.102708.

[11] H. Li, P. Chaudhari, H. Yang, et al., Rethinking the Hyperparameters for Fine-tuning, in: ICLR, 2020, pp. 1–20. arXiv:2002.11770.
URL http://arxiv.org/abs/2002.11770

[12] X. Li, Y. Grandvalet, F. Davoine, Explicit inductive bias for transfer learning with convolutional networks, in: ICML, Vol. 6, 2018, pp. 4408–4419. arXiv:1802.01483.

[13] A. S. Qureshi, A. Khan, Adaptive transfer learning in deep neural networks: Wind power prediction using knowledge transfer from region to region and between different task domains, Computational Intelligence 35 (4) (2019) 1088–1112. doi:10.1111/coin.12236.

[14] X. Liu, Z. Cao, Z. Zhang, Short-term predictions of multiple wind turbine power outputs based on deep neural networks with transfer learning, Energy 217 (2021) 119356. doi:10.1016/j.energy.2020.119356.

[15] Y. Ju, J. Li, G. Sun, Ultra-Short-Term Photovoltaic Power Prediction Based on Self-Attention Mechanism and Multi-Task Learning, IEEE Access 8 (2020) 44821–44829. doi:10.1109/access.2020.2978635.

[16] J. Henze, J. Schreiber, B. Sick, Representation Learning in Power Time Series Forecasting, in: Deep Learning: Algorithms and Applications, Springer, Cham, 2020, pp. 67–101. doi:10.1007/978-3-030-31760-7_3.

[17] L. Cao, L. Wang, C. Huang, X. Luo, J.-H. Wang, A Transfer Learning Strategy for Short-term Wind Power Forecasting, in: Chinese Automation Congress, 2018, pp. 3070–3075. doi:10.1016/j.renene.2015.06.034.

[18] L. Cai, J. Gu, J. Ma, et al., Probabilistic wind power forecasting approach via instance-based transfer learning embedded gradient boosting decision trees, Energies 12 (1) (2019) 159. doi:10.3390/en12010159.

[19] Y. Liu, J. Wang, Transfer learning based multi-layer extreme learning machine for probabilistic wind power forecasting, Applied Energy 312 (2022) 118729. doi:10.1016/J.APENERGY.2022.118729.

[20] J. Chen, Q. Zhu, H. Li, L. Zhu, D. Shi, Y. Li, X. Duan, Y. Liu, Learning Heterogeneous Features Jointly: A Deep End-to-End Framework for Multi-Step Short-Term Wind Power Prediction, IEEE Transactions on Sustainable Energy 11 (3) (2020) 1761–1772. doi:10.1109/TSTE.2019.2940590.

[21] H. Sheng, B. Ray, J. Shao, D. Lasantha, N. Das, Generalization of solar power yield modelling using knowledge transfer, Expert Systems with Applications (2022) 116992doi:10.1016/J.ESWA.2022.116992.

[22] H. Yin, Z. Ou, J. Fu, Y. Cai, S. Chen, A. Meng, A novel transfer learning approach for wind power prediction based on a serio-parallel deep learning architecture, Energy 234 (2021) 121271. doi:10.1016/J.ENERGY.2021.121271.

[23] M. Khan, M. R. Naeem, E. A. Al-Ammar, W. Ko, H. Vettikalladi, I. Ahmad, Power Forecasting of Regional Wind Farms via Variational Auto-Encoder and Deep Hybrid Transfer Learning, Electronics 11 (2) (2022) 206. doi:10.3390/ELECTRONICS11020206.

[24] G. Almonacid-Olleros, G. Almonacid, D. Gil, J. Medina-Quero, Evaluation of Transfer Learning and Fine-Tuning to Nowcast Energy Generation of Photovoltaic Systems in Different Climates, Sustainability 14 (5) (2022) 3092. doi:10.3390/SU14053092.

[25] C. Yan, Y. Pan, C. L. Archer, A general method to estimate wind farm power using artificial neural networks, Wind Energy 22 (11) (2019) 1421–1432. doi:10.1002/WE.2379.

[26] J. Schreiber, A. Buschin, B. Sick, Influences in Forecast Errors for Wind and Photovoltaic Power: A Study on Machine Learning Models, in: INFORMATIK 2019, GI e.V., 2019, pp. 585–598. doi:10.18420/inf2019_74.

[27] S. Zhou, L. Zhou, M. Mao, et al., Transfer learning for photovoltaic power forecasting with long short-term memory neural network, in: International Conference on Big Data and Smart Computing (BigComp), 2020, pp. 125–132. doi:10.1109/BigComp48618.2020.00-87.

[28] M. Ceci, R. Corizzo, F. Fumarola, et al., Predictive Modeling of PV Energy Production: How to Set Up the Learning Task for a Better Prediction?, IEEE TIL 13 (3) (2017) 956–966. doi:10.1109/TII.2016.2604758.

[29] T. Shireen, C. Shao, H. Wang, et al., Iterative multi-task learning for time-series modeling of solar panel PV outputs, Applied Energy 212 (2018) 654–662. doi:10.1016/j.apenergy.2017.12.058.

[30] S. Tasnim, A. Rahman, A. Oo, et al., Wind power prediction in new stations based on knowledge of existing Stations: A cluster based multi source domain adaptation approach, Knowledge-Based Systems 145 (2018) 15–24. doi:10.1016/j.knosys.2017.12.036.

[31] S. Vogt, A. Braun, J. Dobschinski, et al., Wind Power Forecasting Based on Deep Neural Networks and Transfer Learning, in: Wind Integration Workshop, Vol. 18, 2019, p. 8.

[32] J. Schreiber, B. Sick, Emerging Relation Network and Task Embedding for Multi-Task Regression Problems, in: ICPR, 2020, pp. 2663–2670. doi:10.1109/ICPR48806.2021.9412476.

[33] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, 2000.

[34] A. Gensler, Wind power ensemble forecasting, Ph.D. thesis, University of Kassel (2018).

[35] C. Zhang, J. Bin, Z. Liu, Wind turbine ice assessment through inductive transfer learning, in: International Instrumentation and Measurement Technology Conference, IEEE, 2018, pp. 1–6. doi:10.1109/I2MTC.2018.8409794.

[36] G. Guariso, G. Nunnari, M. Sangiorgio, Multi-step solar irradiance forecasting and domain adaptation of deep neural networks, Energies 13 (15) (2020) 1–18. doi:10.3390/en13153987.

[37] M. Thill, W. Konen, T. Bäck, Time Series Encodings with Temporal Convolutional Networks, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 12438 LNCS (2020) 161–173. doi:10.1007/978-3-030-63710-1_13/COVER/.

[38] R. Zhu, W. Liao, Y. Wang, Short-term prediction for wind power based on temporal convolutional network, Energy Reports 6 (2020) 424–429. doi:10.1016/j.egyr.2020.11.219.

[39] J. Yan, L. Mu, L. Wang, R. Ranjan, A. Y. Zomaya, Temporal Convolutional Networks for the Advance Prediction of ENSO, Scientific Reports 10 (1) (2020). doi:10.1038/s41598-020-65070-5.

[40] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.

[41] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, C. Liu, A survey on deep transfer learning, Lecture Notes in Computer Science 11141 (2018) 270–279. doi:10.1007/978-3-030-01424-7_27/COVER.

[42] H. I. Fawaz, G. Forestier, J. Weber, et al., Transfer learning for time

series classification, in: 2018 IEEE BigData, 2019, pp. 1367–1376. doi:10.1109/BigData.2018.8621990.

[43] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 1–112. doi:10.48550/arXiv.1512.03385.

[44] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[45] M. Borthwick, Math for Machine Learning, Cambridge University, 2019. doi:10.1515/9781400843909-001.

[46] J. A. Hoeting, D. Madigan, A. E. Raftery, et al., Bayesian model averaging: a tutorial, MathSciNet 14 (4) (1999) 382–417. doi:10.1214/SS/1009212519.

[47] A. Gensler, B. Sick, A multi-scheme ensemble using coopetitive soft gating with application to power forecasting for renewable energy generation, CoRR arXiv:1803.06344 (2018).

[48] S. Deist, M. Bieshaar, J. Schreiber, A. Gensler, B. Sick, Coopetitive soft gating ensemble, in: Workshop on Self-Improving System Integration (SISSY), Trento, Italy, 2018, pp. 190–197.

[49] M. Bieshaar, Cooperative intention detection using machine learning. Advanced cyclist protection in the context of automated driving, kassel university press, 2021.

[50] A. Gensler, B. Sick, Forecasting wind power - an ensemble technique with gradual coopetitive weighting based on weather situation, in: IJCNN, Vancouver, BC, 2016, pp. 4976–4984.

[51] S. R. Sain, V. N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag New York, 2006. doi:10.1007/978-1-4757-3264-1.