References

1. D. Gielen, F. Boshell, D. Saygin, et al. The role of renewable energy in the global energy transformation. *Energy Strategy Reviews*, 24:38–50, 2019.
2. R. Schwartz, J. Dodge, N. A. Smith, et al. Green AI. *CoRR*, pages 1–12, 2019. arXiv: 1907.10597.
3. J. Schreiber and B. Sick. Emerging Relation Network and Task Embedding for Multi-Task Regression Problems. In *ICPR*, 2020.
4. S. Vogt, A. Braun, J. Dobschinski, et al. Wind Power Forecasting Based on Deep Neural Networks and Transfer Learning. In *18th Wind Integration Workshop*, page 8, 2019.
5. J. Schreiber, A. Buschin, and B. Sick. Influences in Forecast Errors for Wind and Photovoltaic Power: A Study on Machine Learning Models. In *INFORMATIK 2019*, pages 585–598. Gesellschaft für Informatik e.V., 2019.
6. M. Solas, N. Cepeda, and J. L. Viegas. Convolutional Neural Network for Short-term Wind Power Forecasting. In *Proc. of the ISGT-Europe 2019*, 2019.
7. Z. Fuzhen, Q. Zhiyuan, D. Keyu, et al. A comprehensive survey on transfer learning. *Proc. of the IEEE*, 109(1):43–76, 2021.
8. Yu Zhang and Qiang Yang. A Survey on Multi-Task Learning. *IEEE TKDE (Early Access)*, pages 1–20, 2021.
9. T. Shireen, C. Shao, H. Wang, et al. Iterative multi-task learning for time-series modeling of solar panel PV outputs. *Applied Energy*, 212:654–662, 2018.
10. S. Tasnim, A. Rahman, A. M. T. 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*, 145:15–24, 2018.
11. L. Cao, L. Wang, C. Huang, et al. A Transfer Learning Strategy for Short-term Wind Power Forecasting. In *Chinese Automation Congress*, pages 3070–3075. IEEE, 2018.
12. 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):159, 2019.
13. A. S. Qureshi and 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):1088–1112, 2019.
14. X. Liu, Z. Cao, and Z. Zhang. Short-term predictions of multiple wind turbine power outputs based on deep neural networks with transfer learning. *Energy*, 217:119356, 2021.
15. Y. Ju, J. Li, and G. Sun. Ultra-Short-Term Photovoltaic Power Prediction Based on Self-Attention Mechanism and Multi-Task Learning. *IEEE Access*, 8:44821–44829, 2020.
16. S. Zhou, L. Zhou, M. Mao, et al. Transfer learning for photovoltaic power forecasting with long short-term memory neural network. In *Proc. of the BigComp 2020*, pages 125–132, 2020.
17. H. Zang, L. Cheng, T. Ding, et al. Day-ahead photovoltaic power forecasting approach based on deep convolutional neural networks and meta learning. *Int. JEPE*, 118:105790, 2020.
18. Tomas Mikolov, Kai Chen, Greg Corrado, et al. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*, pages 3111–3119, 2013.
19. C. Guo and F. Berkhahn. Entity Embeddings of Categorical Variables. *CoRR*, pages 1–9, 2016. arXiv: 1604.06737.

20. H. I. Fawaz, G. Forestier, J. Weber, et al. Transfer learning for time series classification. In *2018 IEEE BigData*, pages 1367–1376, 2019.
21. Jining Yan, Lin Mu, Lizhe Wang, Rajiv Ranjan, and Albert Y. Zomaya. Temporal Convolutional Networks for the Advance Prediction of ENSO. *Scientific Reports*, 10(1), 2020.
22. J. Reis and G. Gonçalves. Hyper-Process Model: A Zero-Shot Learning algorithm for Regression Problems based on Shape Analysis. *JMLR*, 1:1–36, oct 2018.
23. C. Blundell, J. Cornebise, K. Kavukcuoglu, et al. Weight Uncertainty in Neural Networks. In *32nd ICML 2015*, volume 37, pages 1613–1622, 2015.
24. J. Schreiber, M. Siefert, K. Winter, et al. Prophesy: Prognoseunsicherheiten von Windenergie und Photovoltaik in zukünftigen Stromversorgungssystemen. *German National Library of Science and Technology*, page 159, 2020.
25. European centre for medium-range weather forecasts. `http://www.ecmwf.int/`, 2020. [Online; accessed 2021-03-30].
26. S. R. Sain and V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, 2006.
27. L. N. Smith. A Disciplined Approach To Neural Network Hyper-Parameters: Part 1. *CoRR*, pages 1–21, 2016. arXiv: 1803.09820.
28. S. Bai, J. Z. Kolter, and V. Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *CoRR*, pages 1–14, 2018. arXiv: 1803.01271.

A Intro

The following supplementary material provides additional insights and results for the article *"Task Embedding Temporal Convolution Networks for Transfer Learning Problems in Renewable Power Time-Series Forecast"*.

B Examples of the Data

In the following, we provide exemplary plots of the EuropeWindFarm and GermanSolarFarm datasets. In both datasets, the weather forecasts' uncertainty causes a deviation between the weather feature and the power generation.

We provide a scatter plot between one of the essential features and the historical power measurement for each dataset. We can observe the difference between the weather forecast and the actual generated power in the scatter plots. Similarly, we provide a time series plot.

Generally, PV power forecasts are considered a more straightforward problem as the relation between the solar radiation and the generated power is primarily linear. Nonetheless, features of NWP influence each other non-linearly, making it still a challenging problem. In contrast to the solar dataset, the non-linearity between the most crucial feature - wind speed - and the generated power is more non-linear, making it an even more challenging problem.

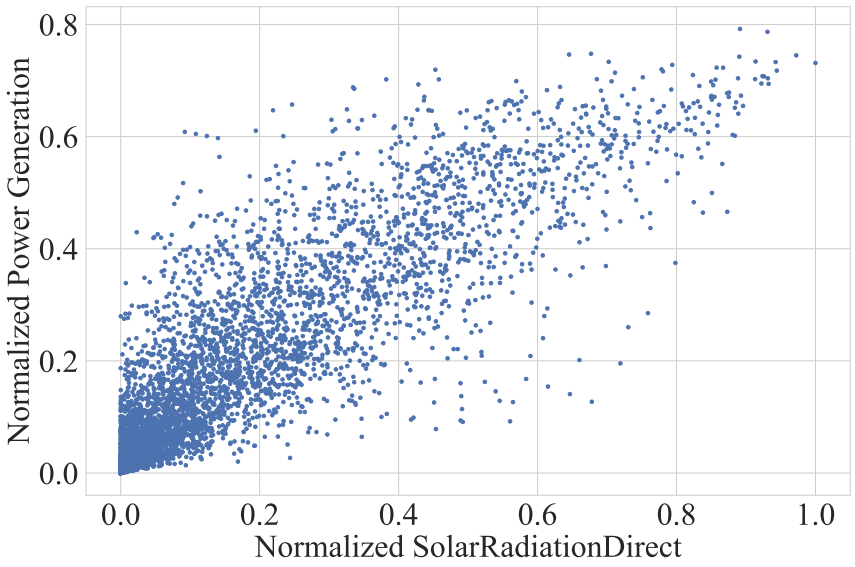

Fig. 5. Exemplary scatter plot between the most important feature - radiation - and the generated power for the GermanSolarFarm dataset.

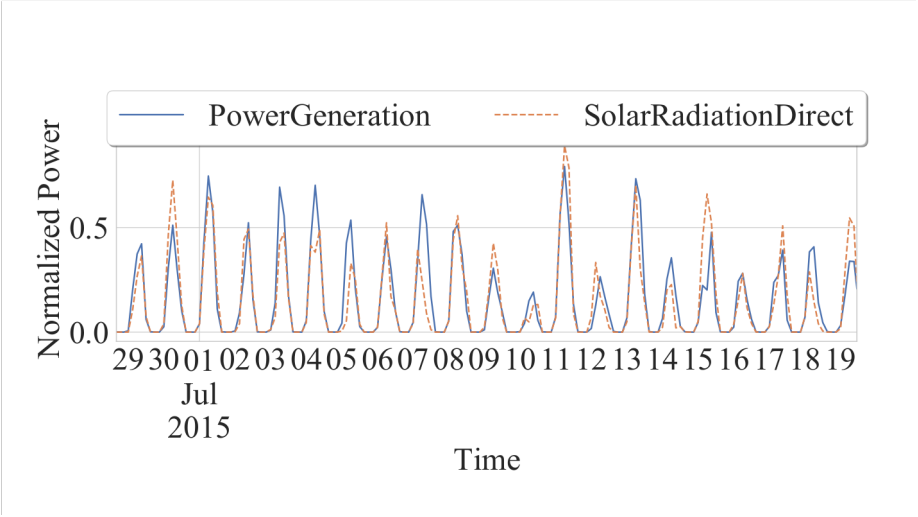

Fig. 6. Exemplary time seris plot between the most important feature - radiation - and the generated power for the GermanSolarFarm dataset.

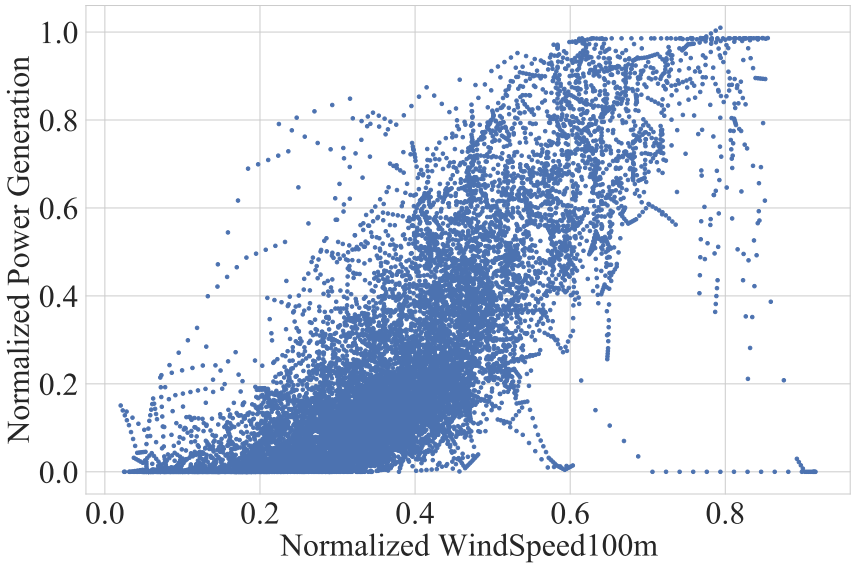

Fig. 7. Exemplary scatter plot between the most important feature - wind speed - and the generated power for the EuropeWindFarm dataset.

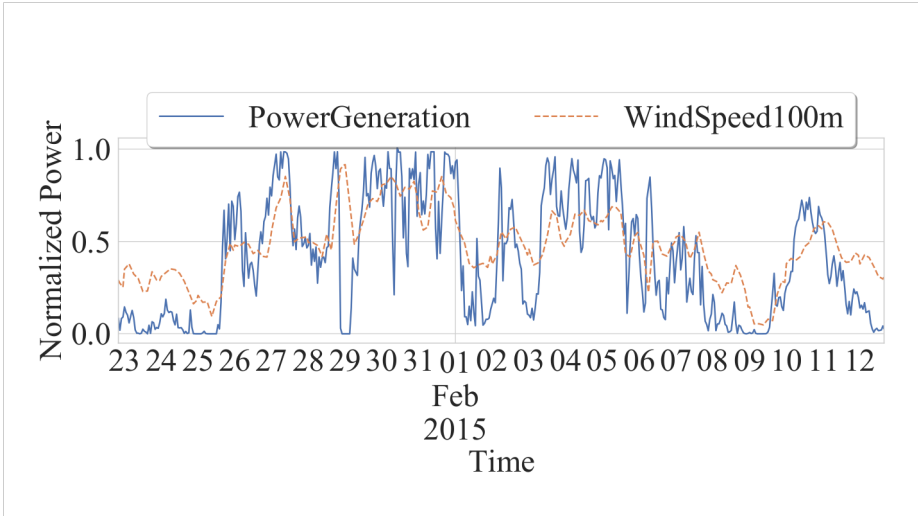

Fig. 8. Exemplary time seris plot between the most important feature - wind speed - and the generated power for the EuropeWindFarm dataset.

C Method

The following section provides additional material for Sec. 3 - Proposed Method.

C.1 Task-Embedding for MLPs

Fig. 9 depicts the task embedding MLP. Before training, each task gets a unique task ID assigned. In our case, this is an increasing number as we are interested in extending the MTL to new tasks later on. E.g., when considering 100 PV parks, we have task IDs from one to hundred. This task ID is the input to the embedding layer. For a particular task ID, this embedding space's output is concatenated with other continuous inputs such as those from the NWP. In this way, it is extendable to any network architecture, as the output of the embedding layer can be used as additional continuous input to the network structure. E.g., the network avoids the necessity of task-specific layers of an HPS architecture. However, additional assumptions need to be made; see description of task-TCN in Sec. 3.2.

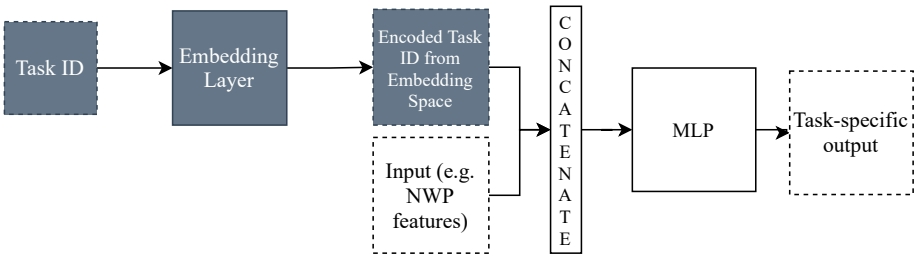

Fig. 9. Task embedding for MLP to create task-specific predictions based on a hard parameter sharing architecture without separate layers. By encoding a task ID, for each task, through an embedding layer, the MLP learns task-specific forecasts while utilizing the data from all tasks to improve predictions.

C.2 Bayesian Task-Embeddings

In our particular approach of the Bayesian task embedding, see Sec. 3.2, we utilize a standard normal distribution as variational distribution. Respectively, each embedding vector requires a mean $\boldsymbol{\mu}_{m\beta}$ and the standard deviation $\boldsymbol{\sigma}_{m\beta}$ for the task m. By applying the local reparametrization trick [23] we sample from the variational distribution to obtain:

$$\mathbf{w}_{m_i\beta} = \boldsymbol{\mu}_{m_i\beta} + \boldsymbol{\sigma}_{m_i\beta}\epsilon.$$

Afterward, each sample is concatenated with its respective numerical weather prediction $\mathbf{x}_i^m$. By adding the Kullback-Leibler Divergence (KLD), between the standard normal and the variational distribution, to the mean squared error (MSE) loss $\mathcal{L}$, we obtain the following update rules:

$$\Delta\boldsymbol{\mu}_{m_i\beta} = \frac{\partial L}{\partial \mathbf{w}_{m_i\beta}} + \lambda \boldsymbol{\mu}_{m_i\beta}$$

$$\Delta\boldsymbol{\delta}_{m_i\beta} = \frac{\partial L}{\partial \mathbf{w}_{m_i\beta}} \epsilon + \lambda \left(\boldsymbol{\delta}_{m_i\beta} - \frac{1}{\boldsymbol{\delta}_{m_i\beta}} \right)$$

C.3 Residual Block

In each residual block, the input is processed through two times the following pattern: dilated convolution, weight norm, ReLU activation, and dropout for regularization, see Fig. 2. An optional convolution to transform the original input to the output shape ensures that the initial input features match those of the residual block's output. Finally, this potentially transformed input is added to the other output.

D Experiment

D.1 Design of Experiment of Sec. 4.3 - MTL Experiment

Based on the description on Sec. 4.3, we now detail the model's hyperparameter optimization via grid search. The layers' initialization strategy is the default one of the utilized libraries (*pytorch*[1], *fastai*[2], and *blitz-bayesian-pytorch*[3]). For all models, we conducted a grid search on the validation dataset. We initially transform the model's number of features in a higher dimension through a factor $k \in \{1, 5, 10, 20\}$. Note that initially transforming the input features in a higher-dimensional space typically improves the performance [26]. Afterward, the features are reduced by 50 percent in each layer to a minimum of 11 before the final output. The final two layers of all networks have sizes 5 and 1. The learning rate of the *adam* optimizer is 10^{-4} for all experiments. To accelerate the training, we initially train for 25 epochs with a one-cycle learning rate scheduler [27] with a maximum learning rate of 10^{-3} and cosine annealing. During each training, we train for 50 or 100 epochs with a batch size of 1024, 2048, or 4096 for the MLP and 32, 64, or 128 for TCN models as hyperparameters. The initial features of all models are fed through a batch norm. This batch norm assures that input features are within the same range for all parks. Other parameters of the TCN are equal to [28] with a kernel size of three. The MLP has the following components in each layer: batch norm, linear layer, ReLU activation. In the Bayesian variants we weight the KLD by one of $\{10^{-1}, 10^{-3}, 10^{-6}\}$.

D.2 Design of Experiment of Sec. 4.5 - Inductive TL Experiment

For seven days of training, we used 20 and otherwise 10 percent of the training data to select the number of epochs $(1, 2, 5, 10, 20)$ and the weight decay $(0, 0.25, 0.5)$. Note that we used the entire first year for standardization of the target data. This approach is generally practical as the input features are forecasts for each location with a long history.

[1] `https://pytorch.org/docs/stable/index.html`, accessed 2020-04-29
[2] `https://docs.fast.ai/`, accessed 2020-04-29
[3] `https://pypi.org/project/blitz-bayesian-pytorch/`, accessed 2020-04-29

E Exploratory Analysis of Task-TCN

The task-TCN initially transforms the discrete task ID through an embedding layer. Ideally, in this **embedding space**, similar tasks should be close to one another. However, in contrast to the task embedding for MLPs, an additional transformation occurs, which results in a **transformed embedding space**. In particular, the embedding space is transformed through a 1D convolution. The results of this transformation are added to the output of the residual block. By adding the convolution results, we essentially add a bias to each channel, depending on the relationship between tasks.

For instance consider, the output of the first residual block has 100 channels of 24 timesteps. For a single sample, the dimension will be $1 \times 100 \times 24$. Also, consider each task has an embedding vector of dimension 1×10. The embedding tensor will be then $1 \times 10 \times 24$. After applying the transformation, through the 1D convolution, the transformed embedding tensor will also be of dimension $1 \times 100 \times 24$. In the following, we exemplarily visually inspect features of those spaces and compare them with each other. For those visualizations, we use the task-TCN with a non-Bayesian embedding layer which only includes the encoded task ID in the first residual (TCN) block.

E.1 Visualization of Embedding Space through t-SNE

In the following, we visualize the embedding space and the transformed embedding space through t-distributed stochastic neighbor embedding (t-SNE) in Fig. 10 and 11. Therefore, we create once the embedding for each task. In the example here, this results in a vector of dimension 36×12. The first dimension is the number of tasks and the second dimension is the resulting dimension of the embedding layer. In the transformed embedding space, we have a dimension of 36×140, where 140 is now the number of channels in the first layer of the task-TCN.

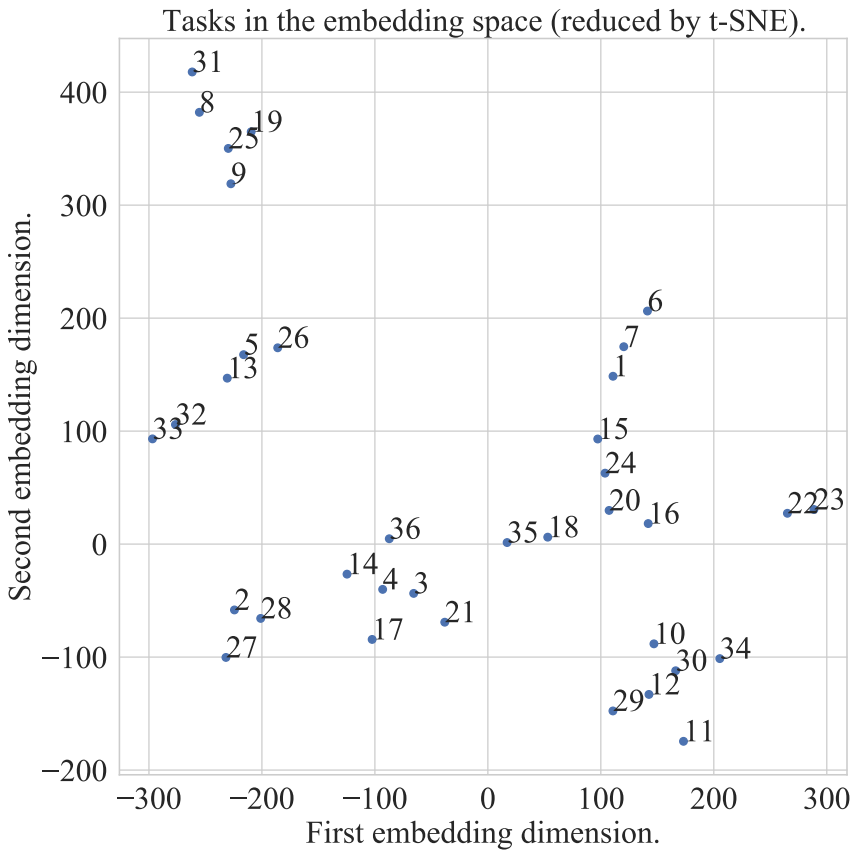

Fig. 10. Visualization of the embedding space through t-SNE. The encoded task ID, from the embedding space is calculated for each task and concatenated. The dimensions are reduced through t-SNE and visualized in the scatter plot. Each number corresponds to the respective task ID.

Due to the transformation through t-SNE, we cannot directly compare the embedding and the transformed embedding space. However, we can visually verify if specific tasks are close to one another in both plots. For instance the tasks $29, 12, 30, 10$, and 34 are always close to one another in both spaces, c.f. Fig. 10 and 11, even for multiple runs of t-SNE. Similarly, $26, 5, 13, 32$, and 33. A smaller distance is also present in various other clusters. However, some cannot be clustered easily. For instance task 11 is close to $29, 12, 30, 34, 10$ in the embedding space. In the transformed embedding space, it is closest to $22, 23, 24, 20$ and 16. This potentially indicates that task 11 is less similar to other tasks.

E.2 Correlation between Embedding Space and Transformed Embedding Space

To verify if the similarity between tasks in the embedding space is similar to the transformed embedding space, we utilize the Euclidean distance. In particular, we calculate the Euclidean distances for each task with other tasks in the embedding space and the transformed embedding space. Fig. 12 depicts the results of this calulation.

The Pearson correlation of above 0.9 indicates a substantial linear dependency that is also visually available in the scatter plot. These results let us assume that the learned similarity from the embedding space is not drastically changed through the transformation.

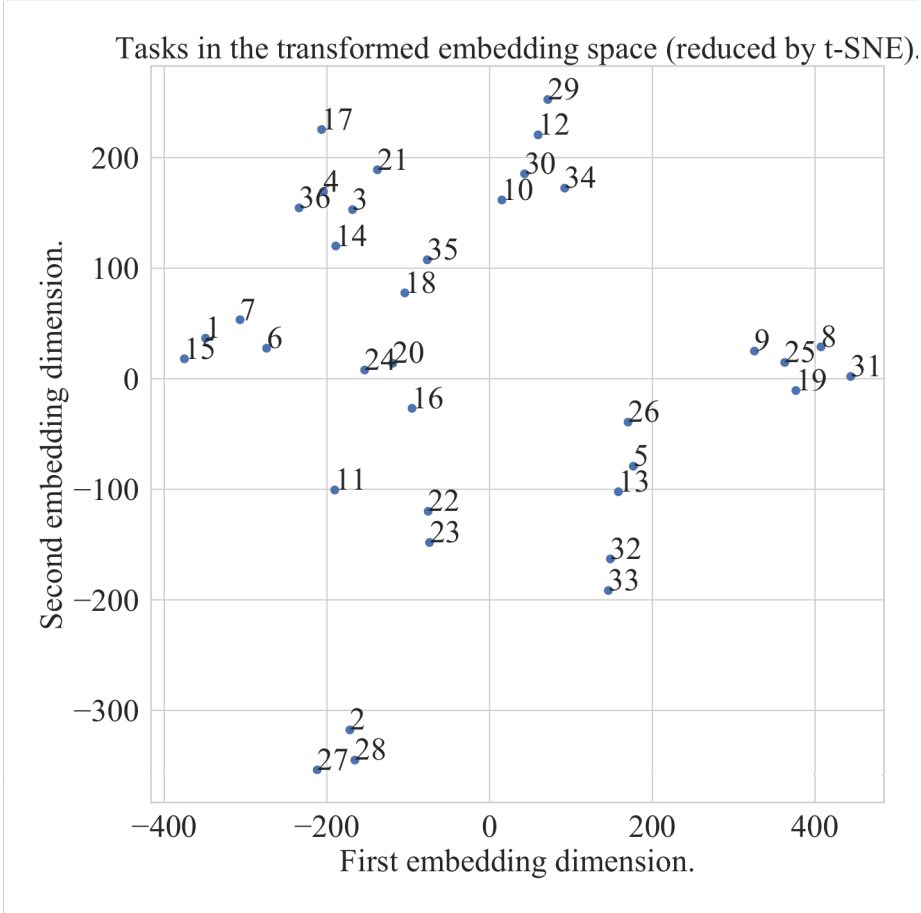

Fig. 11. Visualization of the transformed embedding space through t-SNE. The encoded task ID, from the embedding space is calculated for each task and channel. The dimensions are reduced through t-SNE and visualized in the scatter plot. Each number corresponds to the respective task ID.

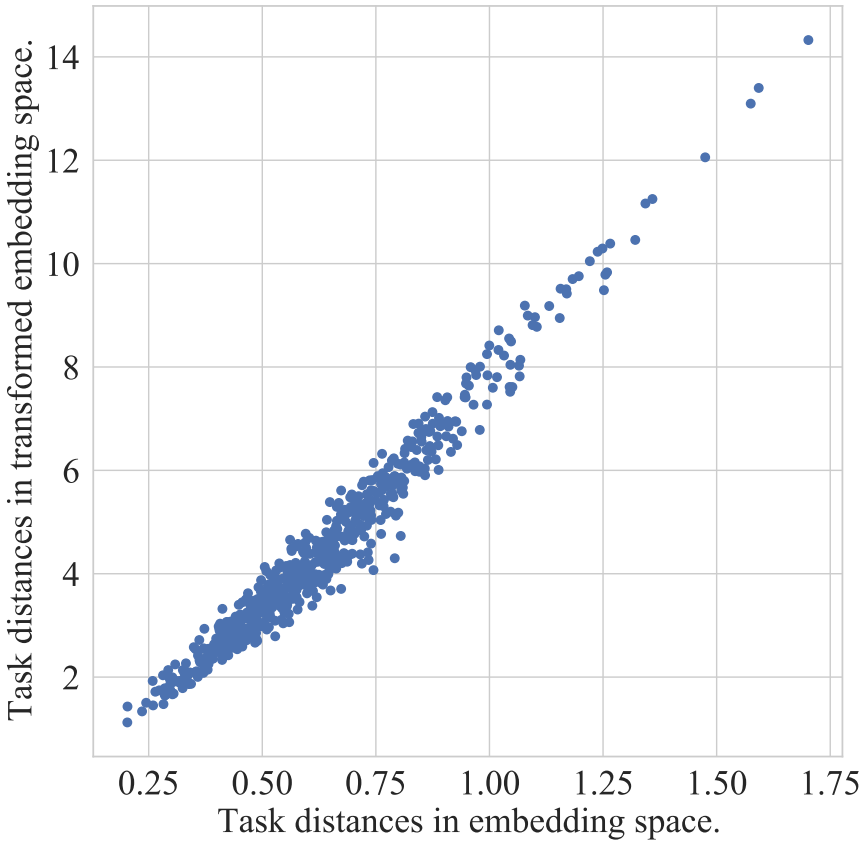

Fig. 12. Euclidean distance between task in the embedding space and the transformed embedding space.

E.3 Distance between Tasks for each Channel

The following exemplary heatmaps visualize the distances between tasks for each channel. Therefore, we calculate the Euclidean distance for each task within a channel. So, we essentially measure the distance of the bias per channel as visualized in Fig. 13, 14, and 15. Interestingly, the similarities or rather dissimilarities between channels differ strongly. For instance, there is a substantial difference of task 30 to all other tasks in channel two. Comparing channel two with channel three, we observe that this pattern is not visible. Instead, there is a substantial dissimilarity between 20 and 21. Other channels such as $9, 10$ and 11 have a check-board-like pattern, where we can observe groups of similarities and dissimilarities.

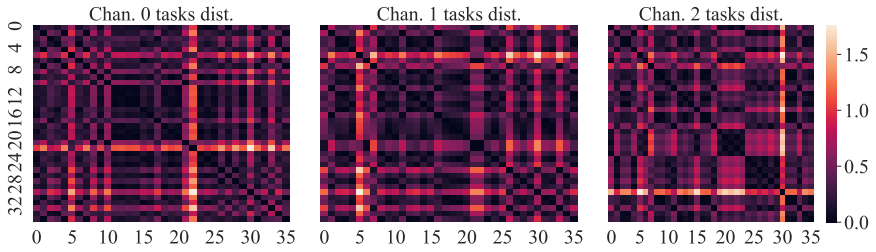

Fig. 13. Euclidean distance between task in the transformed embedding space for exemplary channels. Values closer to zero indicate a more substantial similarity.

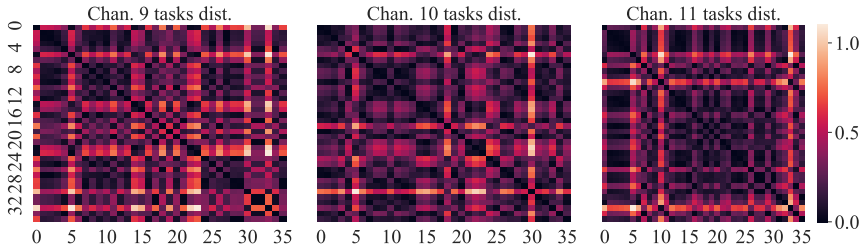

Fig. 14. Euclidean distance between task in the transformed embedding space for exemplary channels. Values closer to zero indicate a more substantial similarity.

These findings let us assume that the additional transformation is beneficial in learning complex similarities structures. We can expect that specific channels are responsible for different kinds of weather situations and terrain. Finally, we can conclude that similarities are complex ones that need to be modeled in a high-dimensional space. In this way, the additional transformation is beneficial so that the model can learn

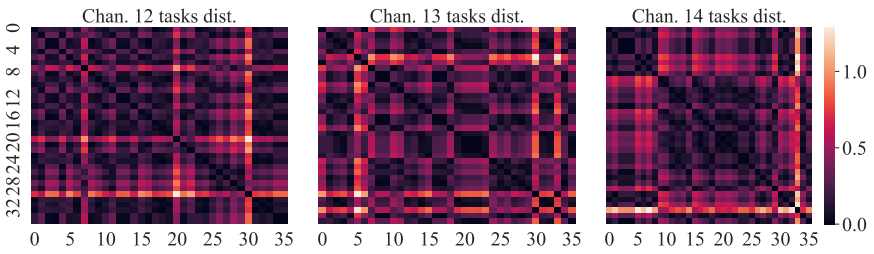

Fig. 15. Euclidean distance between task in the transformed embedding space for exemplary channels. Values closer to zero indicate a more substantial similarity.

diverse features for different weather situations while taking similarities of tasks into account.

F Exemplary Forecasts of all Models of Sec. 4.5 - Inductive TL Experiment

The following section provides samples plot for both datasets, all six models, and the different initialization strategies for the embedding layer for new parks All models are trained with 90 days from spring.

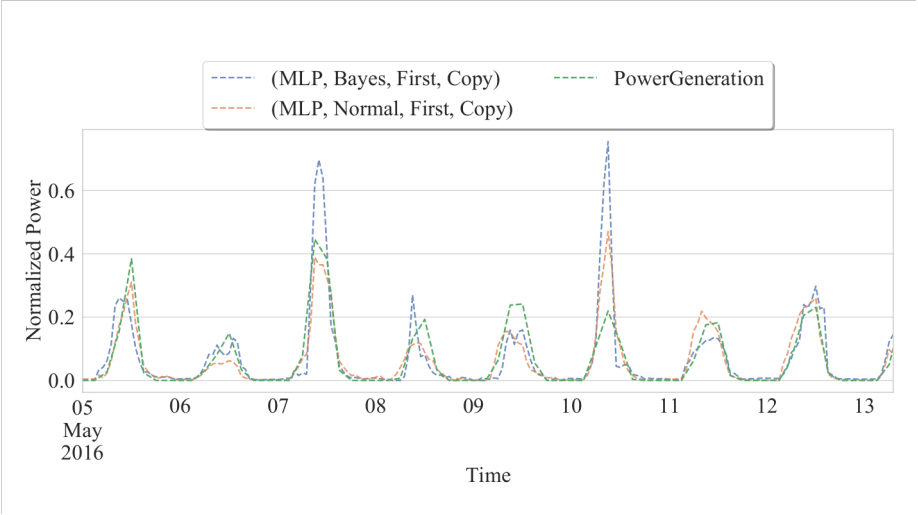

Fig. 16. Sample forecast of the task embedding MLP for the GermanSolarFarm dataset. The embedding initialization is copied from the most similar park.

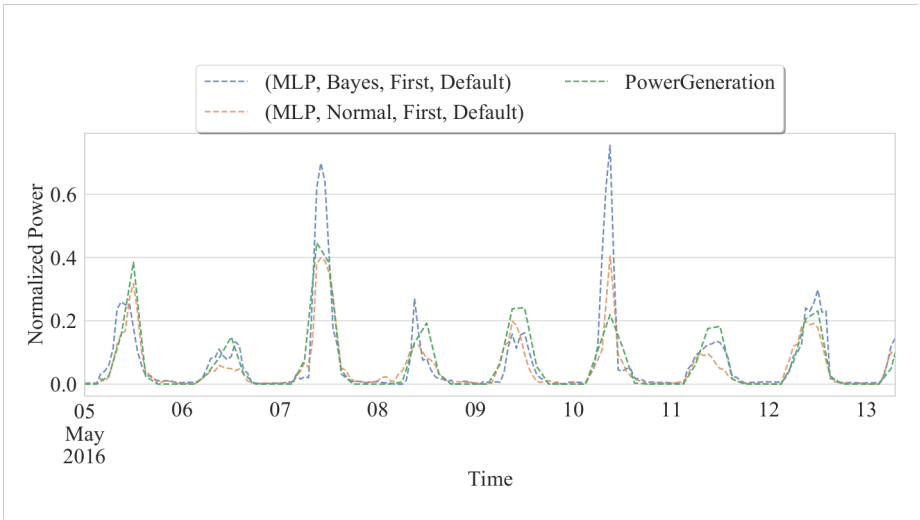

Fig. 17. Sample forecast of the task embedding MLP for the GermanSolarFarm dataset. The embedding initialization is the default one.

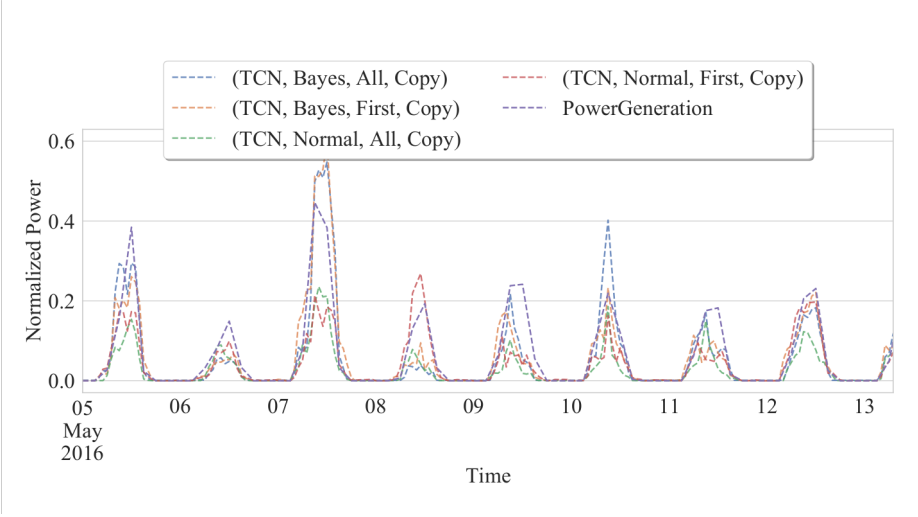

Fig. 18. Sample forecast of the task-TCN for the GermanSolarFarm dataset. The embedding initialization is copied from the most similar park.

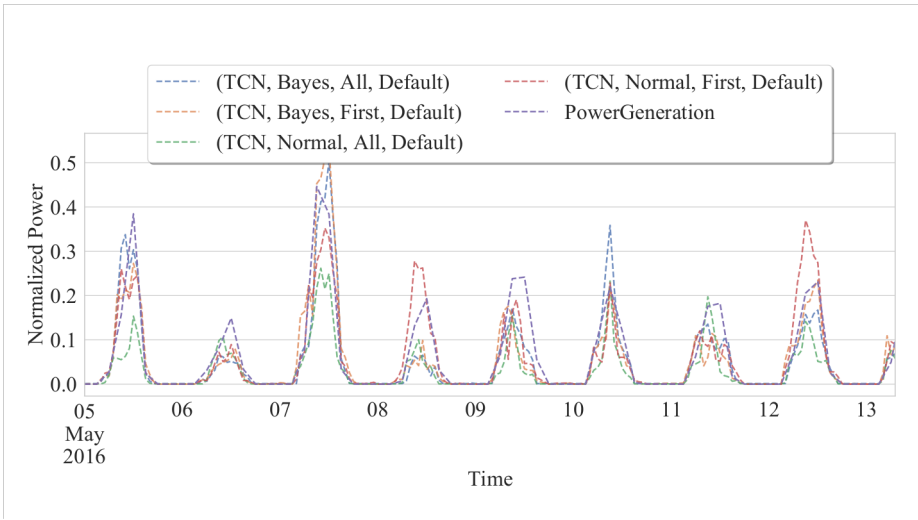

Fig. 19. Sample forecast of the task-TCN for the GermanSolarFarm dataset. The embedding initialization is the default one.

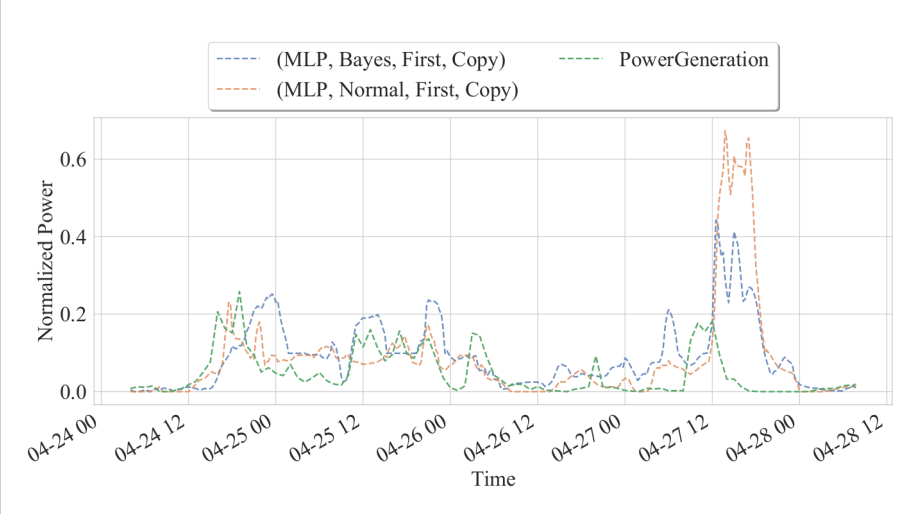

Fig. 20. Sample forecast of the task embedding MLP for the EuropeWindFarm dataset. The embedding initialization is copied from the most similar park.

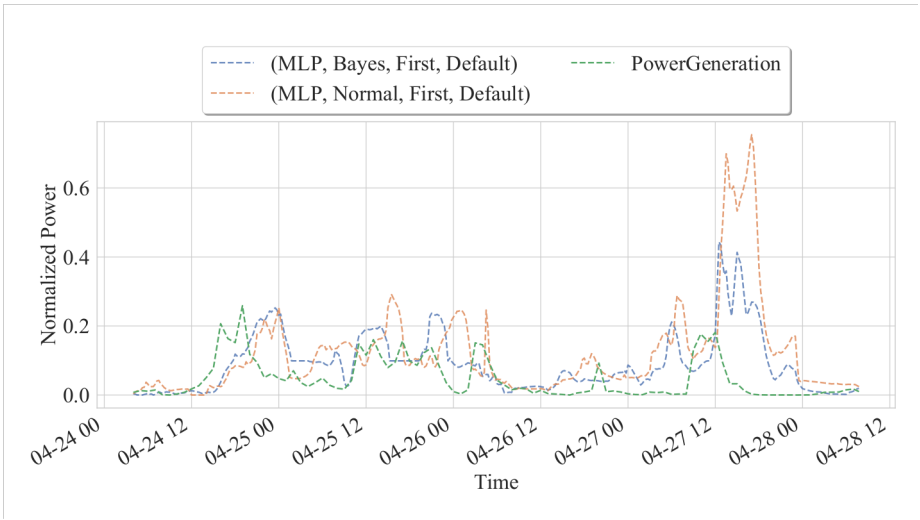

Fig. 21. Sample forecast of the task embedding MLP for the EuropeWindFarm dataset. The embedding initialization is the default one.

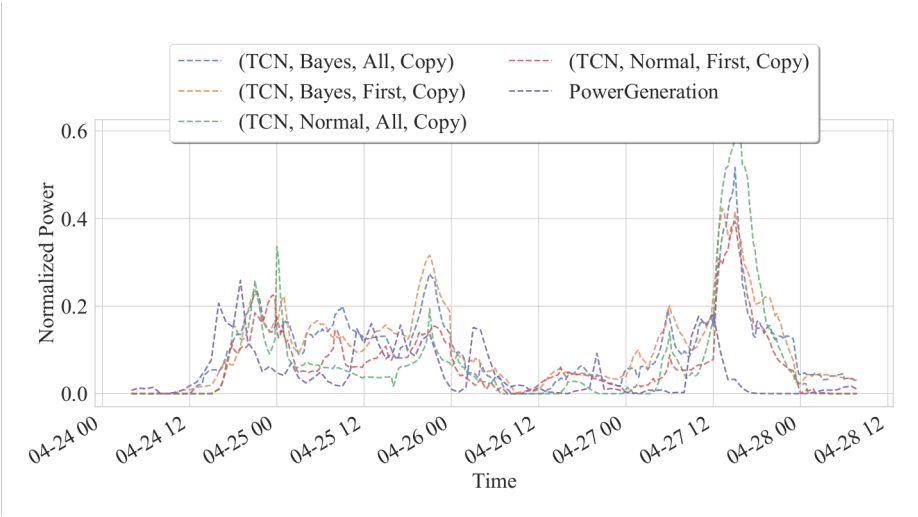

Fig. 22. Sample forecast of the task TCN for the EuropeWindFarm dataset. The embedding initialization is copied from the most similar park.

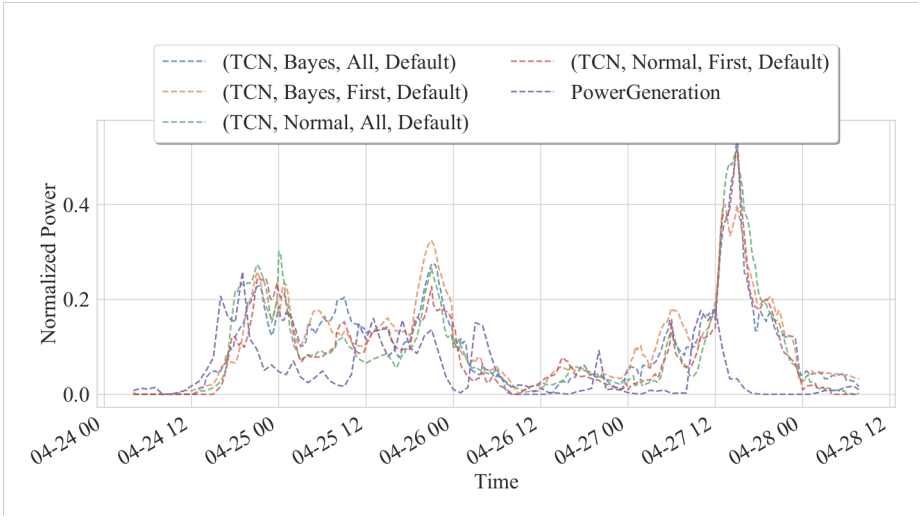

Fig. 23. Sample forecast of the task TCN for the EuropeWindFarm dataset. The embedding initialization is the default one.

G Detailed Results of Sec. 4.5 - Inductive TL Experiment

G.1 EuropeWindFarm Results

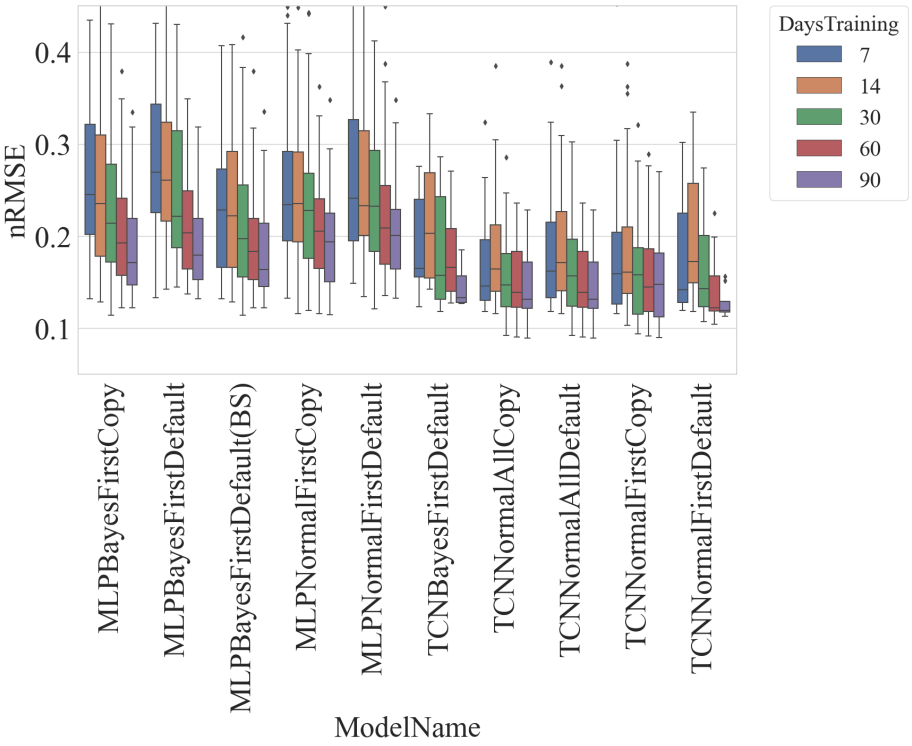

Fig. 24. Results for inductive TL experiment for winter of the wind dataset. BS marks the baseline.

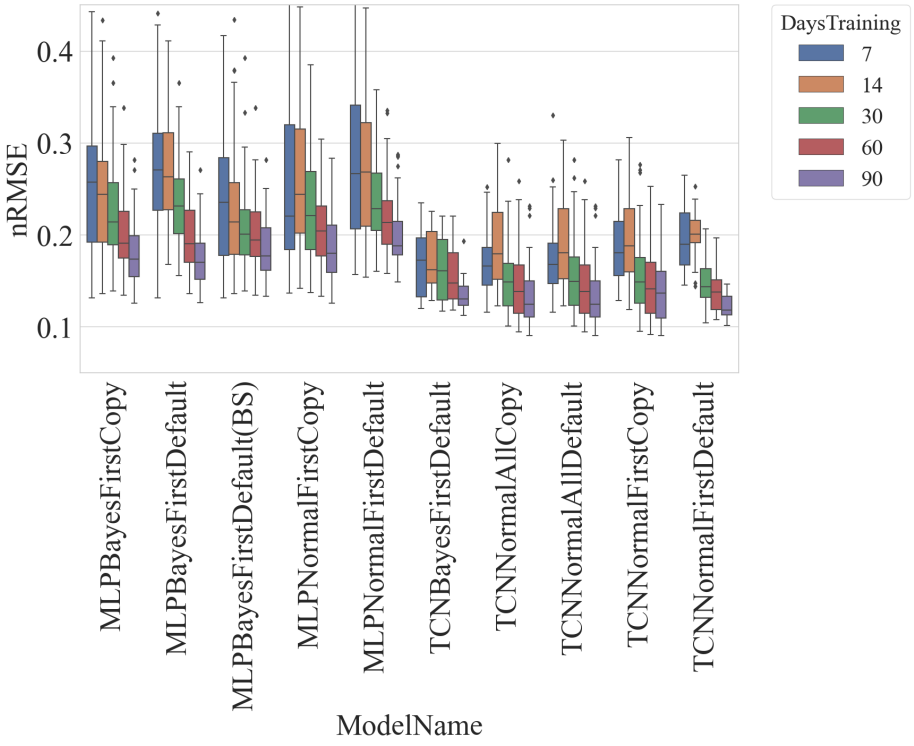

Fig. 25. Results for inductive TL experiment for spring of the wind dataset. BS marks the baseline.

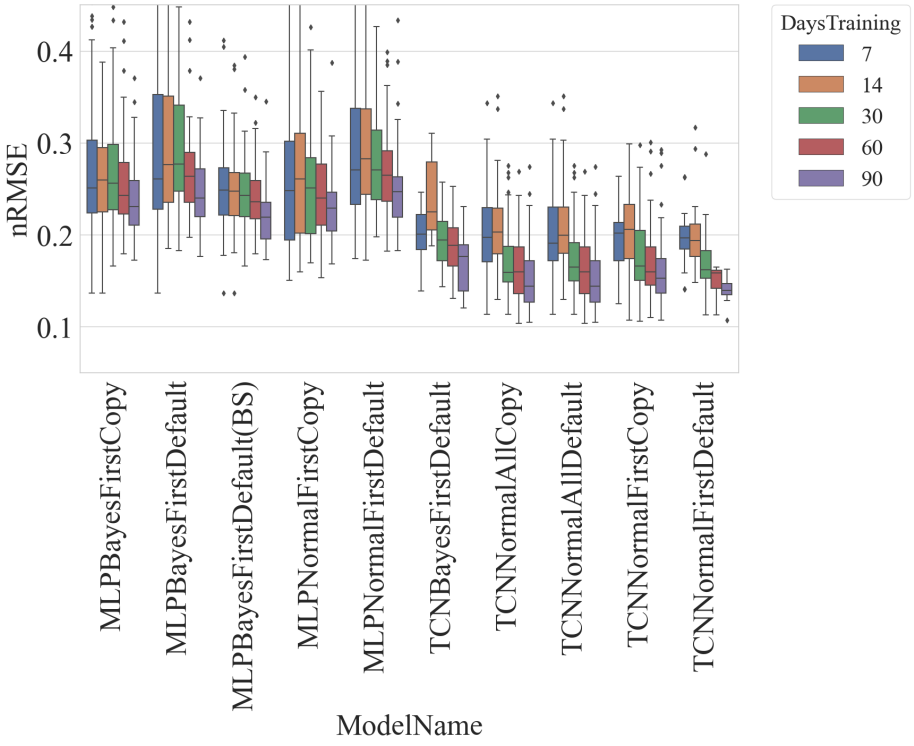

Fig. 26. Results for inductive TL experiment for summer of the wind dataset. BS marks the baseline.

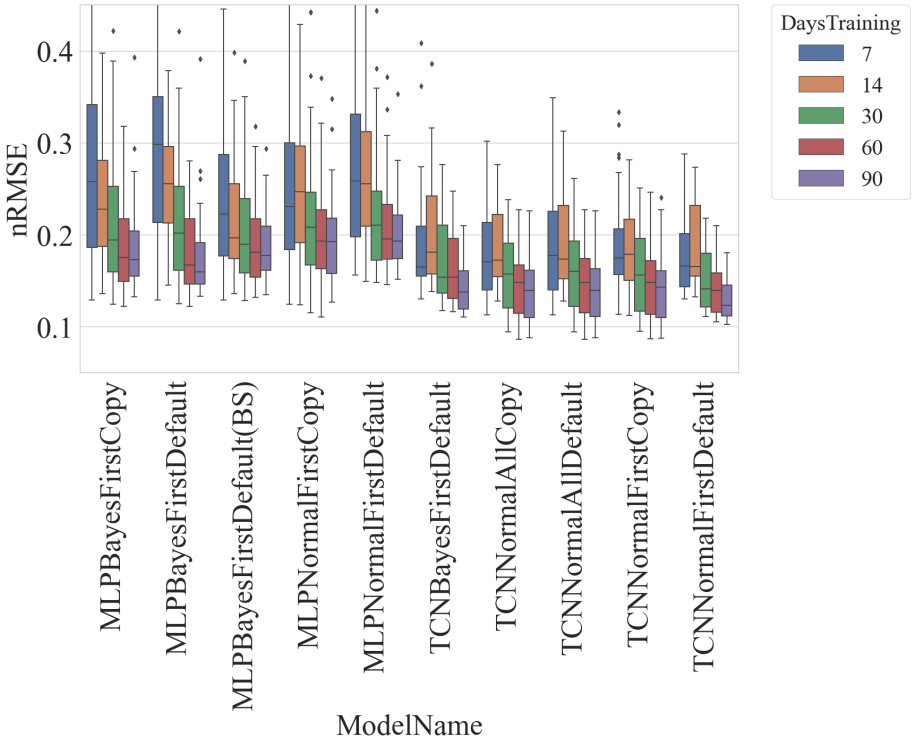

Fig. 27. Results for inductive TL experiment for autumn of the wind dataset. BS marks the baseline.

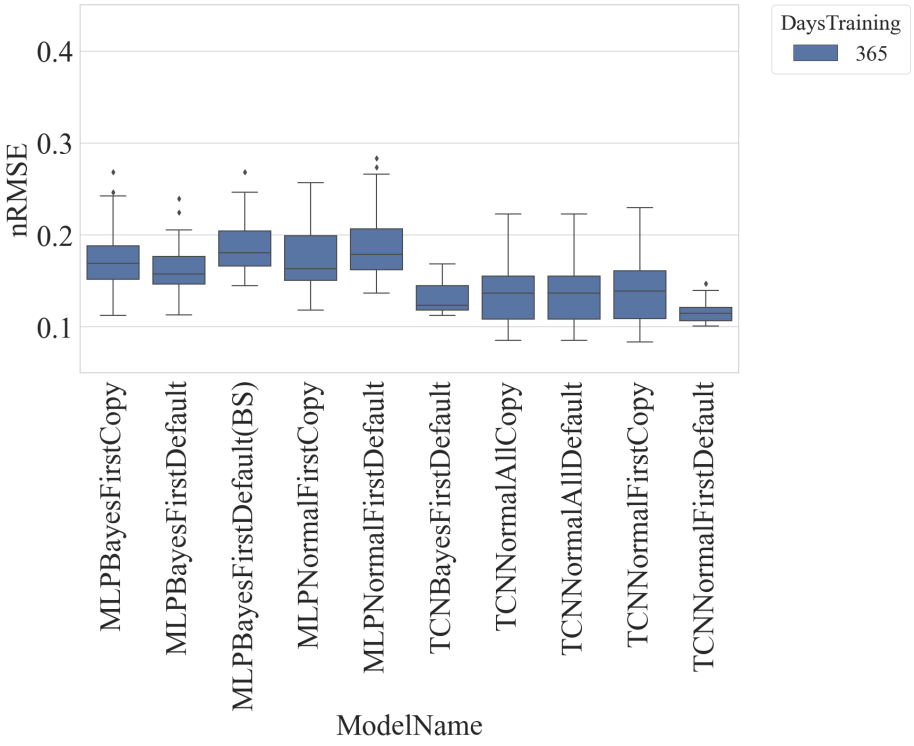

Fig. 28. Results for inductive TL experiment for complete year of the wind dataset. BS marks the baseline.

G.2 GermanSolarFarm Results

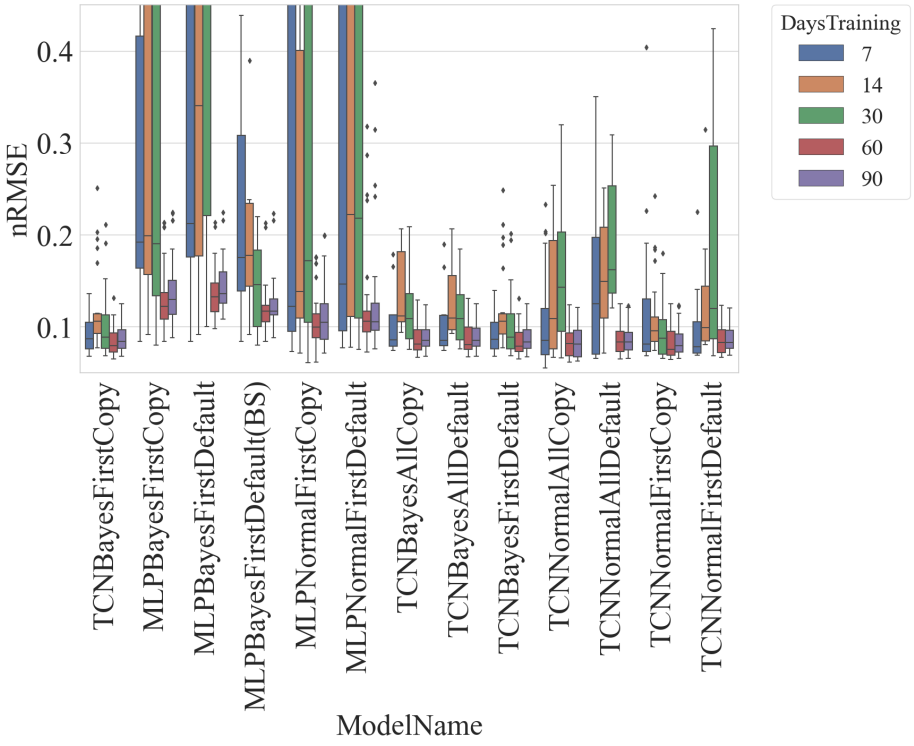

Fig. 29. Results for inductive TL experiment for winter of the solar dataset. BS marks the baseline.

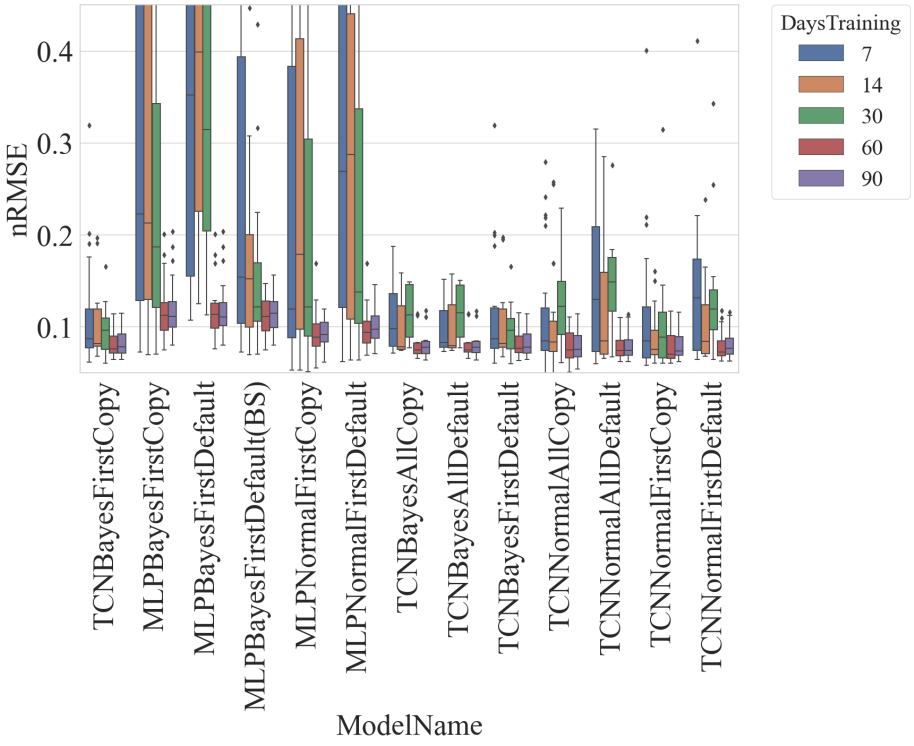

Fig. 30. Results for inductive TL experiment for spring of the solar dataset. BS marks the baseline.

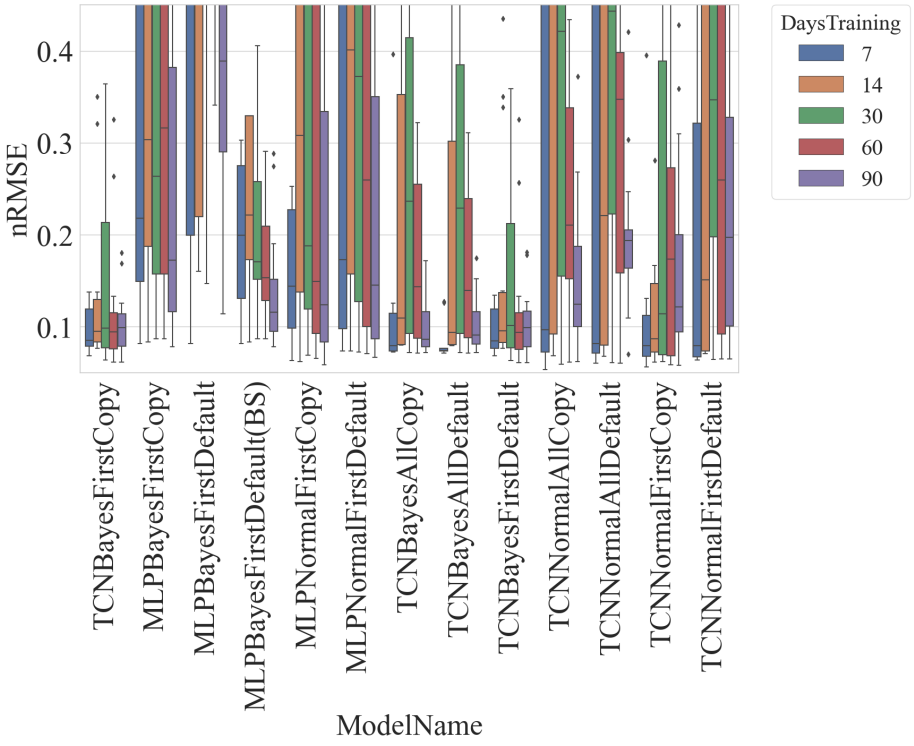

Fig. 31. Results for inductive TL experiment for summer of the solar dataset. BS marks the baseline.

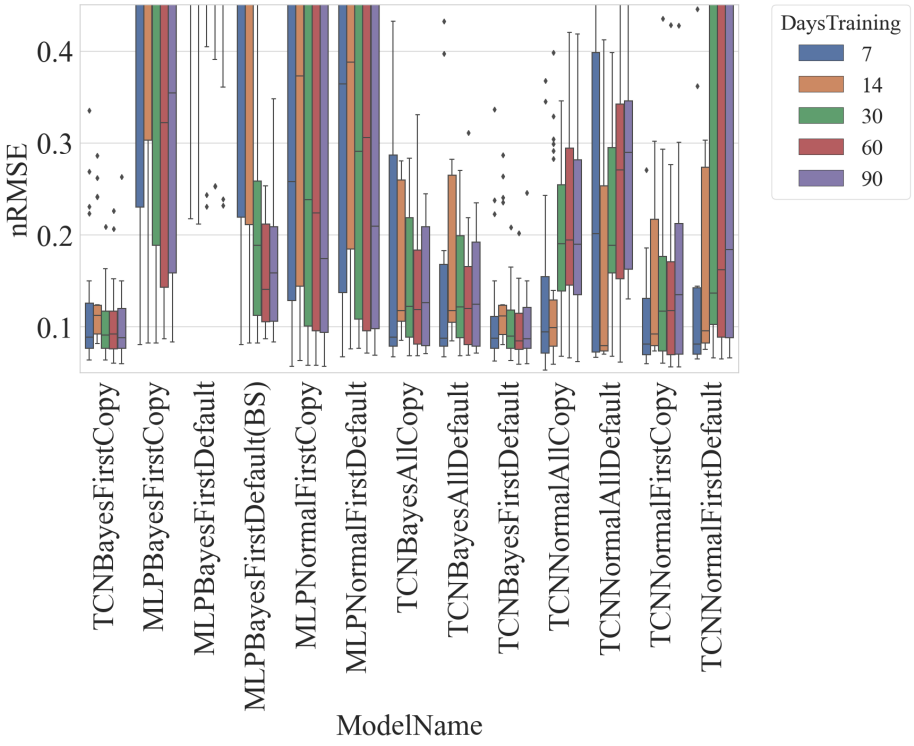

Fig. 32. Results for inductive TL experiment for autumn of the solar dataset. BS marks the baseline.

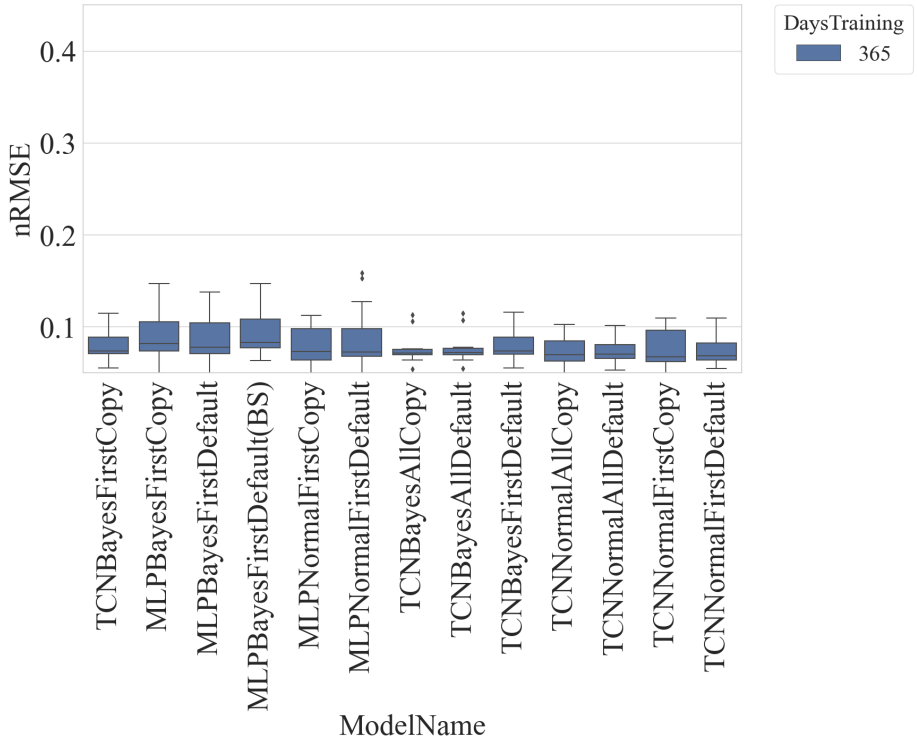

Fig. 33. Results for inductive TL experiment for complete year of the solar dataset. BS marks the baseline.