Domain Adaptation for Time series Transformers using One-step fine-tuning

Subina Khanal[1], Seshu Tirupathi[2], Giulio Zizzo[2], Ambrish Rawat[2], Torben Bach Pedersen[1]
[1]Department of Computer Science, Aalborg University
[2]IBM Research Europe
subinak@cs.aau.dk, seshutir@ie.ibm.com, giulio.zizzo2@ibm.com, ambrish.rawat@ie.ibm.com, tbp@cs.aau.dk

Abstract

The recent breakthrough of Transformers in deep learning has drawn significant attention of the time series community due to their ability to capture long-range dependencies. However, like other deep learning models, Transformers face limitations in time series prediction, including insufficient temporal understanding, generalization challenges, and data shift issues for the domains with limited data. Additionally, addressing the issue of catastrophic forgetting, where models forget previously learned information when exposed to new data, is another critical aspect that requires attention in enhancing the robustness of Transformers for time series tasks. To address these limitations, in this paper, we pre-train the time series Transformer model on a source domain with sufficient data and fine-tune it on the target domain with limited data. We introduce the One-step fine-tuning approach, adding some percentage of source domain data to the target domains, providing the model with diverse time series instances. We then fine-tune the pre-trained model using a gradual unfreezing technique. This helps enhance the model's performance in time series prediction for domains with limited data. Extensive experimental results on two real-world datasets show that our approach improves over the state-of-the-art baselines by 4.35% and 11.54% for indoor temperature and wind power prediction, respectively.

Introduction

Time series prediction has been a significant subject of academic study with applications in finance, weather, and climate change. Time series prediction approaches have evolved over the past few decades from classical statistical methodologies and machine learning (ML) techniques to deep learning-based solutions. Recently, the breakthrough of Transformers in deep learning has attracted much attention from the time series community owing to its outstanding performance in a variety of computer vision and natural language processing tasks (??). Transformers have many benefits, but one that makes time series modeling particularly well suited for them is their capacity to capture long-range dependencies and interactions. This has resulted in significant advancements in a variety of time series applications (?). However, Transformers, like other deep learning models such as Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), or Autoencoders, have several limitations on time series prediction. First, limited data availability: deep learning models require a large amount of data for training, which is not always available for some time series prediction in real-world scenarios (?). Second, lack of temporal understanding: Transformers, in particular, may not accurately capture the temporal dynamics in time series because they are primarily intended for tasks not inherently involving temporal dependencies (?). Third, the problem of data shift: deep learning models assume that the training and test data are drawn from the same distribution. However, real-world scenarios often involve changes in data distribution, termed data shift. When this assumption is violated, the performance of the model degrades significantly (?). Fourth, lack of generalization: deep learning models frequently have difficulty generalizing well to new, unseen data (?). They may perform well on the training data but fail to accurately predict unseen data. This is the scenario where the unseen data is not independent and identically distributed (non-i.i.d) w.r.t. the training data because of the data shift problem. Thus, the model becomes specialized to the training data and cannot perform well for all types of new data.

Domain adaptation (DA) is a technique used to address the above mentioned challenges by improving the model's ability to generalize in scenarios with a different data distribution (??). DA leverages knowledge from the source domain, where sufficient data is available, to the target domain, where the data is limited, for making accurate predictions even when the two domains have different data distributions. However, models, like Transformers, are usually trained on fixed data, assuming the distribution of the test data remains constant. This is impractical for real-world applications, where data can evolve, and the model might need to adapt to new information without completely retraining from scratch (?). Continual learning (?) addresses this by allowing models to learn continuously from a data stream over time.

Accepted at the Fourth Workshop of Artificial Intelligence for Time Series Analysis (AI4TS): Theory, Algorithms, and Applications, AAAI 2024, Vancouver, Canada.

The ML model can retain knowledge from previously learned tasks while learning and adapting continuously as new data becomes available. However, this causes catastrophic forgetting where performance on old tasks degrades over time as new tasks are added (?). To address catastrophic forgetting in continual learning, replay methods (?) reintroduce past data during training that enables the model to balance learning from new tasks and retaining knowledge from earlier ones. It is also seen that catastrophic forgetting can occur during DA when a model is adapted to the target domain (??). This is because of two reasons: First, when performing DA, we train the prediction model on sufficient data from the source domain. Then, we leverage the knowledge of that model to the target domain with limited data to allow the model to generalize effectively on the target domain. However, during this adaptation process, the model may prioritize the data of the target domain and adjust its parameters in a way that causes it to forget the previously learned knowledge from the source domain. As a result, the performance of the model may degrade in the source domain, leading to generalization issues. Second, there is a higher chance of forgetting if the data distributions for the source and target domains differ significantly.

This paper aims to adapt the Transformer model to the target domain while simultaneously addressing the data shift and catastrophic forgetting problems between source and target domains. We pre-train a time series Transformer model on large data of the source domain and fine-tune and adapt the pre-trained model on different target domains using our One-step fine-tuning approach. Therein, we specifically involve portion of source domain data to the target domains and fine-tune the pre-trained model using a gradual unfreezing (GU) (?) technique, which allows us to adapt and scale the pre-trained Transformer model to different target domains.

In summary, the main contributions of the present paper are as follows. (1) We propose a One-step fine-tuning technique, where we add some percentage of source domain data to the target domains and fine-tune the pre-trained model. Thereby, we show the fine-tuned model is adaptable and performs well on new, unseen data from different target domains. (2) We mitigate the problems of data shift and catastrophic forgetting by fine-tuning the pre-trained time series Transformer model on different target domains with limited data. (3) We conduct an extensive experimental evaluation using real-world datasets, which shows that the our One-step fine-tuning approach outperforms the state-of-the-art baselines. We obtain 4.35% and 11.54% improvements over the most competitive baseline for indoor temperature and wind power prediction, respectively.

Related Work

Several Transformer-based time series forecasting techniques have gained significant attention recently, particularly for long-term time series forecasting. The authors in (?) study the long-sequence time series forecasting problem and aim to predict long sequences. The ProbSparse self-attention mechanism and the distilling operation are used to handle the challenges of quadratic time complexity and quadratic memory usage in the vanilla Transformer. Also, this method alleviates the limitation of the traditional encoder-decoder architecture by using a carefully designed generative decoder. Similarly, in (?), the authors study the problem of long-term forecasting of time series. They propose a decomposition architecture by embedding the series decomposition block as an inner operator, which can progressively aggregate the long-term trend part from intermediate prediction. This approach also designs an Auto-Correlation mechanism to conduct dependencies discovery and information aggregation at the series level, which differs significantly from the previous self-attention family. Likewise, in (?), the authors introduce two key components: 1) patching; segmentation of time series into subseries-level patches, which are served as input tokens to the Transformer; and 2) a channel-independent structure, where each channel contains a single univariate time series that shares the same embedding and Transformer weights across all the series for long-term multivariate time series forecasting and self-supervised representation learning.

Preliminaries

This section presents the core concepts of our One-step fine-tuning approach for time series prediction that will be used throughout the paper.

Time series: A univariate time series $X = (x_1, x_2, \dots, x_n)$, where $x_t \in \mathbb{R}$, is a sequence of real values that measure the same phenomenon and are chronologically ordered. Each value x_t is recorded at uniformly spaced time instants $t \in \{1, 2, \dots\}$. A time series X has length n if it has n collected samples.

Time window of a time series: Let X be a time series of length n. A time window $X_w = (x_{t-m+1}, \dots, x_{t-1}, x_t)$ consists of historical values of X recorded at time $(t-m+1)$ up to time t. Here, m is the memory, denoting the size of the time window X_w which defines how many historical values will be used for prediction.

Source domain: A source domain SD is a set of training data with a given distribution of the input data $P(X_{\mathrm{SD}})$ used to train a prediction model. Here, we consider a training dataset with enough historical data, often collected over more than a year, as the source domain.

Target domain: A target domain TD is a set of data with a given distribution of the input data $P(X_{\mathrm{TD}})$ that differs from the source domain distribution. Here, we consider target domains to have limited data.

Learning task: Both source and target domains have the same learning task T. Here, the learning task is to predict future time series using historical values.

Objective of the Source task: Let us consider X_{SD} to be the input of the model $\mathcal{M}$, that maps X_{SD} to the predicted output $\hat{Y}_{\text{SD}}$ in the source domain SD.

$$\hat{Y}_{\text{SD}} = \mathcal{M}(X_{\text{SD}}; \theta), \tag{1}$$

where θ is the learnable parameters, i.e., weights of model $\mathcal{M}$. The objective during source task training is to minimize a source task loss function, denoted as $\mathcal{L}(\hat{Y}_{\text{SD}}, Y_{\text{SD}})$, where Y_{SD} is the actual output.

$$\theta^* = \arg\min_{\theta} \frac{1}{N_{\text{SD}}} \sum_{i=1}^{N_{\text{SD}}} \mathcal{L}(\hat{Y}_i, Y_i), \tag{2}$$

where N_{SD} is the total number of samples in source domain, $\hat{Y}_i$ is the predicted output for the i^{th} sample, and Y_i is the actual output for the i^{th} sample in the source domain.

Fine-tuning Objective on the Target task: Once pre-training of the source task is complete, the knowledge gained by the source model is transferred to the target task. During fine-tuning, we define a target loss function, denoted as $\mathcal{L}(\hat{Y}, Y)$, where Y is the actual output.

$$\theta_{\text{ft}} = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\hat{Y}_i, Y_i; \theta^*), \tag{3}$$

where N is the total number of samples in target domain, $\hat{Y}_i$ is the predicted output for the i^{th} sample, and Y_i is the actual output for the i^{th} sample in the target domain. Here, the goal is to update the model parameters to minimize the target loss $\mathcal{L}(\hat{Y}, Y)$, thereby improving the performance of the model on the target task in the target domain.

Proposed Approach

In this section, we present the detailed deployment design of our One-step fine-tuning approach. We build on a time series Transformer model, which is a type of neural network architecture specifically designed to process and model sequential data (?). The detailed workflow of the model architecture is as follows:

Training Dataset and Data Pre-processing: Data collected from multiple residential buildings and wind turbines are used as the source and target training datasets. Before initiating the model training process, these training datasets are pre-processed to clean and format the data into input vectors.

Positional Encoding: The first layer of the model architecture is positional encoding, which is used to add positional information to the input sequence vectors. In Transformers, positional encoding is essential for informing the model about the relative positions of the data points in the sequence (?). For the model to comprehend the temporal relationships between data points, positional encoding is required because the timing and order of observations are critical in time series. The positional encoding is added to the input embeddings, where each input embedding corresponds to a data point in the time series. The positional encoding vector is determined based on each data point's position (timestamp) in the sequence.

We follow the sinusoidal positional encoding technique, initially introduced in the Transformer architecture for natural language processing tasks (?) defined as:

$$\text{PE}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), \tag{4}$$

$$\text{PE}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), \tag{5}$$

where *pos* is the position of time step in a sequence, i is a index of the dimension, and d_{model} is the embedding dimension.

Encoder Layers: The encoder layers are the second layer of the model architecture, responsible for processing the input sequence using self-attention mechanisms and feed-forward networks. These layers are stacked on each other to form the encoder (?). The initial sub-layer of an encoder is the Self-Attention Mechanism, which processes the input sequence vectors and allows the interaction between the values of input vectors. The mechanism learns to weigh the importance of each vector with respect to the others, capturing longer-range temporal dependencies and relationships in the time series. During training, the output of the self-attention part is added to the original inputs for gradient flow, and then layer normalization is applied. Each point in the sequence is individually processed by a feed-forward neural network, adding non-linearity to the encoder's transformations and enabling the learning of higher-level representations of the input data at each layer.

Linear Decoder Layer: The final layer is the linear decoder layer, a fully connected layer used to map the output of the encoder layer to the final prediction (?). The input to a linear decoder layer consists of hidden representations, i.e., the output from the encoder layer. The fundamental operation of a linear decoder layer is a linear transformation represented by a fully connected (dense) neural network layer. The linear transformation projects the high-dimensional input representations into a lower-dimensional space, aligning the model's internal representation with the desired output space. Then, it is reshaped to match the target sequence length.

Next, with the model architecture defined, we discuss the two phases of the workflow in our One-step fine-tuning approach.

1) Pre-training time series Transformer model in source domain: In the first phase, we train a time series Transformer model, i.e., the source model, using the data from the source domain until it converges to a certain level of prediction accuracy. The trained source model efficiently learns the temporal dependencies and patterns from the large and diverse data of the source domain that we transfer to the target domains with limited data. As the learning task is the same between the source and target domains, we validate the patterns

learned by the source model benefits the target domains for such scenarios.

2) Fine-tuning on Target Domains: The second phase is fine-tuning the pre-trained time series Transformer model, i.e., the source model, on the target domains, allowing it to adapt to the target data. This domain adaptation process helps the model perform better in the target domains, even with limited data. For this, we apply our proposed One-step fine-tuning of the source model in the target domains to obtain a fine-tuned target model. This approach includes the following steps:

(2a) We first add some percentage of randomly sampled source domain data to the target domains. Including time series instances of the source domain in the target domains jointly addresses three fundamental issues: (i) the problem of data scarcity, (ii) data distribution mismatch, as data from both domains gets involved during fine tuning, and (iii) catastrophic forgetting, as the source model retains useful data representations through data sharing, enabling better adaptation to the target domain without completely forgetting the knowledge gained from the source domain.

(2b) We then apply gradual unfreezing (GU) technique (?), where each layer of source model is frozen at first and then gradually unfreezed during each training epoch, to obtain a fine-tuned target model. This allows keeping the source model knowledge intact, while the newly added layers of the model are made trainable, and subsequently, tailoring the model for the target domain. As such, this helps stabilize the training process and minimize catastrophic forgetting of pre-trained representations. We consider both source and target domains to have same learning task; hence, new layers are not added to the source model during fine-tuning.

The execution of GU steps for Energy Data (see details on Datasets) is as follows: (2b.1) In the first 10 epochs, the top layers, i.e., the decoder layers (output layer), are unfrozen, (2b.2) From epochs 10 to 20, we gradually unfreeze the layers closer to the input, i.e., we progressively unfreeze the subsequent layers in the Transformer encoder layer and train the model, and finally, (2b.3) after epochs 20 to 35, we unfreeze all the remaining layers of the source model and train the model. Here, we track the training loss over several epochs and implement early stopping to avoid overfitting. If the training loss increases or plateaus, we gradually unfreeze the layers. We fine-tune the source model until it converges to a certain level of prediction accuracy.

Experimental Evaluation

In this section, we present the experimental evaluation of our One-step fine-tuning approach for time series prediction in the source and target domains.

Datasets

We use the following real-world datasets to evaluate the performance of our One-step fine-tuning approach:

Energy Data: We use two public energy datasets in our experiments. The first dataset comes from the New York State Energy Research and Development Authority (NYSERDA) (?), which maintain data from residential buildings in New York State. These buildings are 100 to 600 square meters and have 50 geothermal heat pumps. The data includes observations on indoor temperature, outdoor temperature, and power consumption for about 12 months, with readings every 15 minutes.

The second dataset is collected from the Net-Zero Energy Residential Test Facility (NIST) (?). This facility tests technologies for meeting residential energy needs with renewable energy. The data, collected over a year, simulates the energy usage of a family of four. The readings are taken every minute.

A 15-minute data granularity is selected to be used across all datasets, as it is usually used in electricity and flexibility markets. Thus, the NIST dataset is downsampled by averaging power consumption readings to 15-minute intervals and taking the last indoor and outdoor temperature readings.

MORE Data: We use wind park data with 18 months of data from 11 wind turbines (WT2 to WT11) in a wind park provided by Engie as part of the MORE H2020 project (?). The dataset consists of SCADA data from the sensors on the wind turbines and weather data. Data included in this dataset are:
- Power output: average, minimum, maximum, standard deviation over 10min.
- Ambient temperature: average, minimum, maximum, standard deviation over 10min.
- Wind speed: average, minimum, maximum, standard deviation over 10min.

There are over 840,000 samples from all the wind turbines with 1 hour data granularity. The target variable is minimum power output forecast for a 4 hour horizon at 1 hour intervals.

Table 1: Parameters used in source model fine-tuning.

Parameters	Values	
	Energy Data	MORE Data
No. of Input features	1	18
Target Variable	Indoor Temperature	Minimum Wind Power
Batch Size	8	24
Training Epoch	35	20
Historical values	96 (i.e. 24 hours)	24 (i.e. 24 hours)
Prediction horizon	4 (i.e. 1 hour)	4 (i.e. 4 hours)
Learning rate	0.001	0.00001
Training Set: Test Set	70%:30%	
Optimizer	Adam	
Model Loss Function	Mean Absolute Error (MAE)	

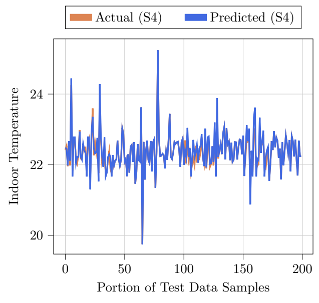
Implementation Details

We use PyTorch framework for the model implementation and evaluation. The implementation starts by selecting the source and target domains as follows.

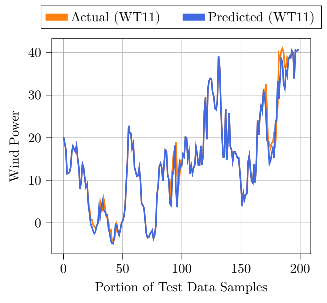
Selection of source domain for pre-training: The source domain should have adequate training data with

Table 2: Prediction error of source model compared to the model baselines.

Methods	Energy Data Prediction Error in S4		MORE Data Prediction Error in WT11	
	RMSE	MAE	RMSE	MAE
Informer	0.420	0.293	5.389	3.251
Autoformer	0.402	0.307	4.830	3.198
PatchTST	0.383	0.296	4.417	2.334
LSTM	0.243	0.185	0.136	0.109
Linear Regression	0.264	0.208	0.139	0.116
Source Model	0.236	0.180	0.118	0.075



(a)



(b)

Figure 1: Performance of Source Model: Actual v.s. Predicted values of Source Domains: (a) S4, and (b) WT11.

seasonal variations. Therefore, for Energy data, out of 50 geothermal heat pump sites from the NYSERDA dataset, we have selected deployment site S4 as the source domain. Similarly, for MORE data, out of 11 wind turbines, we have selected WT11 as the source domain.

Selection of target domains for fine-tuning: The target domains should have the higher data dispersion compared to the selected source domain, which leads to better generalization capability of a model, as it can capture the specific patterns and characteristics unique to the target domain. To compute the marginal distribution disparity between the source and target domains, we adopt a widely used loss function, Maximum Mean Discrepancy (MMD) (??). MMD measures the non-parametric distances between the source and target domains by converting them into a Reproducing Kernel Hilbert Space (RKHS), computed as

$$\begin{aligned}\mathrm{MMD}[P(X_{\mathrm{SD}}), P(X_{\mathrm{TD}})] &= \left\|\mathbb{E}[\varphi(X_{\mathrm{SD}})] - \mathbb{E}[\varphi(X_{\mathrm{TD}})]\right\|^2_{\mathcal{H}} \\ &= \left\|\frac{1}{N_{\mathrm{SD}}}\sum\nolimits_{p=1}^{N_{\mathrm{SD}}}\varphi(x_p) - \frac{1}{N}\sum\nolimits_{q=1}^{N}\varphi(x_q)\right\|^2_{\mathcal{H}},\end{aligned} \tag{6}$$

where $P(X_{\mathrm{SD}})$ and $P(X_{\mathrm{TD}})$ are the marginal data distributions of the source and target domains, respectively, N_{SD} and N are the number of data samples in the source and target domains, $\varphi(\cdot)$ is the mapping function from the original feature space to the RKHS, and $\mathcal{H}$ is the RHKS space. The value of MMD starts from 0, signifying that the compared domains are the same.

The target domains are selected based on MMD values to cover the distribution range of all the target domains. For Energy data, we used the Net-Zero dataset (NIST) and four different deployment sites (S5, S8, S15, and S49) from NYSERDA as the target domains to fine-tune the source model, with MMD values ranging from 0.089 to 0.133. Likewise, for MORE data, we used WT4, WT6, WT7, WT8, and WT10 as the target domains to fine-tune the source model, with MMD values ranging from 0.095 to 0.100.

Source model fine-tuning parameters: Table 1 shows the summary of parameters used during source model fine-tuning in the selected target domains for Energy and MORE data. The hyper-parameters are obtained through grid search.

Baselines

1. Model Baselines: We have selected five different model baselines to evaluate our source model; including Informer (?), Autoformer (?), PatchTST (?), LSTM (?), and Linear Regression (??).

2. Fine-tuning Baselines: We use the following fine-tuning baselines to evaluate our One-step fine-tuning approach. Gradual unfreezing (GU) (?): This technique gradually unfreezes the layers during fine-tuning rather than unfreezing all the layers at once, without including data from the source domain. Elastic Weight Consolidation (EWC) (??): EWC is a regularization technique used to mitigate catastrophic forgetting in neural networks, particularly in the context of continual learning. This technique adds penalty terms or constraints to the model training process. It modifies the model loss/objective function by adding a regularization term.

3. Model Training Baselines: We also compare the performance of our One-step fine-tuning approach with other model training baselines as follows. Exclusive Training: Here, we train the time series Transformer model individually in different target domains and evaluate the individual target models on a test set of the target domain data. Before fine-tuning: Here, we directly assess the source model in the target domains without fine-tuning.

Time Series Prediction Evaluation

Evaluation Metrics: We use well-known metrics, Root Mean Square Error (RMSE) and Mean Absolute Error (MAE), for the quantitative assessment of source and final global model performance on test target domains. These metrics are calculated as

$$RMSE(Y_i, \hat{Y}_i) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}, \tag{7}$$

$$MAE(Y_i, \hat{Y}_i) = \frac{1}{N}\sum_{i=1}^{N}\left|(Y_i - \hat{Y}_i)\right|, \tag{8}$$

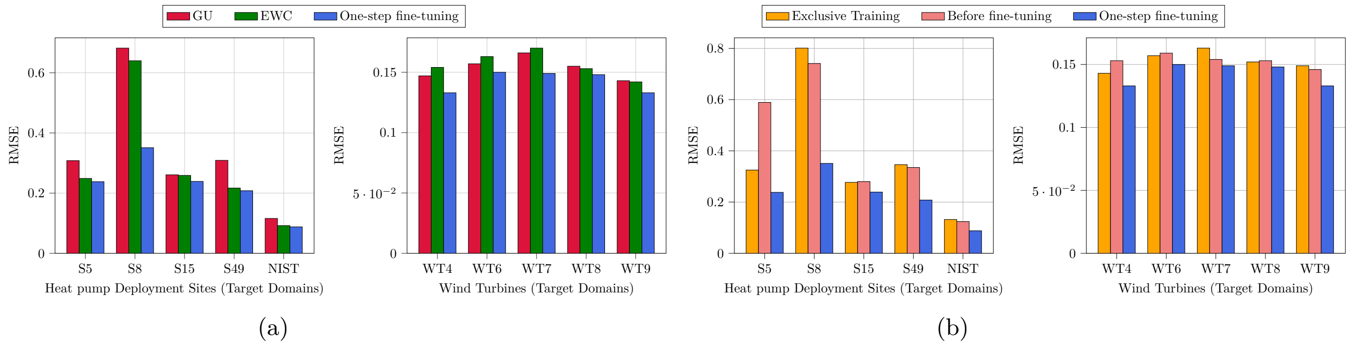


Figure 2: Prediction error of One-step fine-tuning compared to (a) fine-tuning baselines, (b) model training baselines.

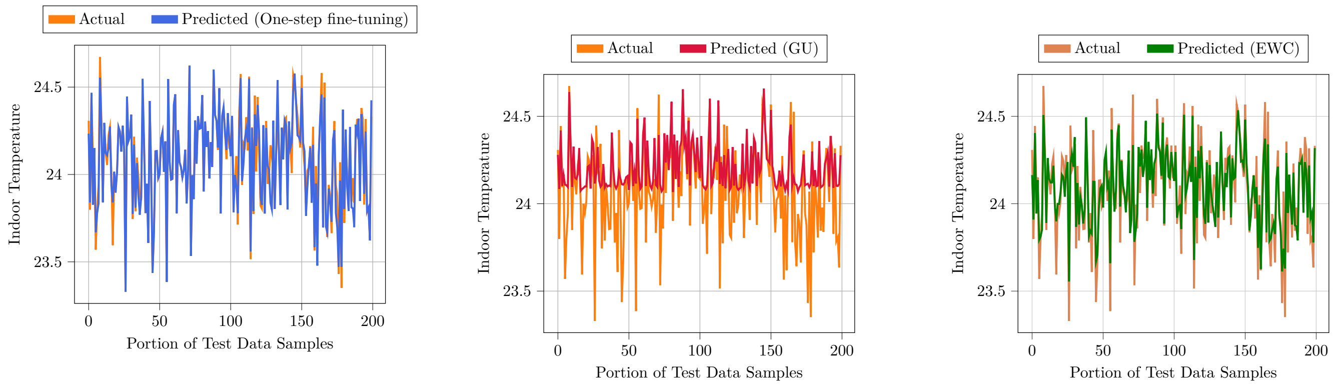


Figure 3: Performance comparison between One-step fine-tuning and fine-tuning baselines in terms of Actual v.s. Predicted indoor temperature values of target domain NIST.

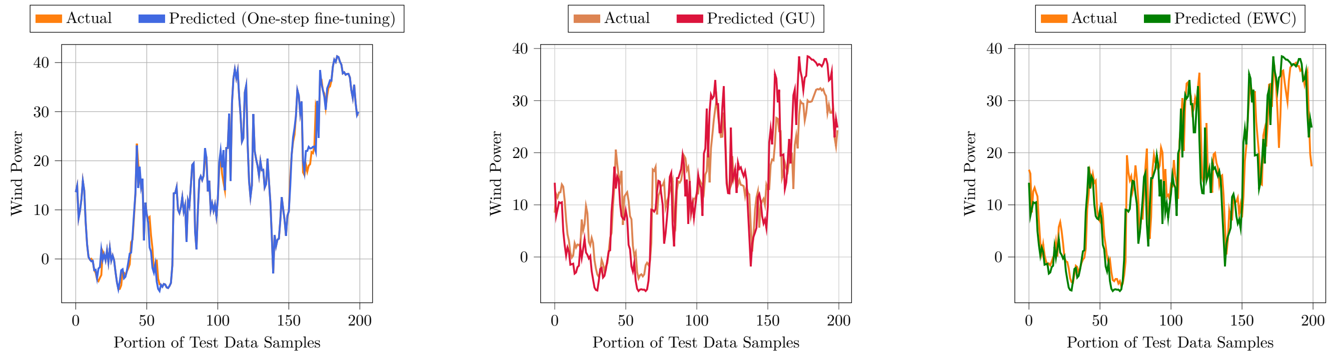


Figure 4: Performance comparison between One-step fine-tuning and fine-tuning baselines in terms of Actual v.s. Predicted wind power values of target domain WT8.

where Y_i and $\hat{Y}_i$ are the actual and predicted output, and N is the total number of samples in the target domains.

Prediction Accuracy Evaluation: Table 2 shows the quantitative performance of the source model evaluated on the source domains S4 and WT11 based on RMSE and MAE metrics respectively. Here, we observe a significant improvement in the source model's performance against the baselines for S4 and WT11, with up to 44% and 97.8% RMSE error reduction, respectively. Likewise, in Fig 1, we have the prediction performance of the source model on the source domains S4 and WT11, respectively. We can see that the source model obtains improved prediction of the indoor temperature and wind power for both source domains.

Next, Fig 2a shows the performance of the fine-tuned target model. Our One-step fine-tuning approach performs better than the baselines GU and EWC on all target domains with limited target data, with a significant improvement of up to 48.5% after adding a

Table 3: Data Shift check on other target domains.

	Domains	Prediction Error on Other Target Domains							
		S3		S7		S19		S50	
		RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
Energy Data	Source Domain (S4)	0.378	0.320	1.09	0.892	0.320	0.260	0.594	0.495
	Target Domains								
	S5	0.222	0.161	0.345	0.276	0.249	0.188	0.383	0.294
	S8	0.264	0.202	0.403	0.328	0.227	0.169	0.397	0.302
	S15	0.221	0.165	0.334	0.259	0.251	0.186	0.365	0.279
	S49	0.247	0.197	0.349	0.265	0.233	0.178	0.442	0.339
	NIST	0.315	0.239	0.376	0.282	0.232	0.171	0.409	0.321
	Domains	Prediction Error on Other Target Domains							
		WT2		WT3		WT5		WT9	
		RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
MORE Data	Source Domain (WT11)	0.133	0.096	0.160	0.118	0.151	0.113	0.152	0.112
	Target Domains								
	WT4	0.121	0.080	0.132	0.087	0.115	0.076	0.121	0.078
	WT6	0.120	0.080	0.132	0.085	0.114	0.075	0.120	0.078
	WT7	0.118	0.078	0.128	0.084	0.116	0.077	0.119	0.077
	WT8	0.124	0.081	0.138	0.089	0.119	0.076	0.126	0.081
	WT10	0.131	0.087	0.130	0.086	0.118	0.077	0.122	0.078

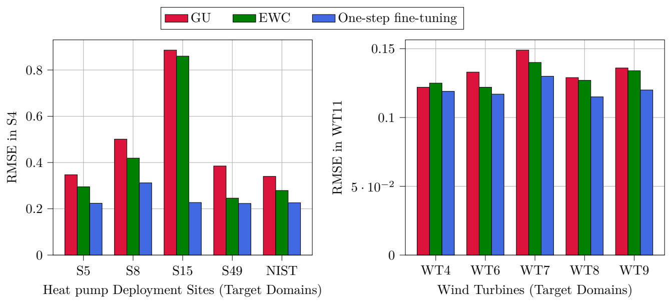


Figure 5: Catastrophic forgetting check on Source Domains S4 and WT11 using fine-tuned target models.

portion of source domain data prior to fine-tuning the source model. For Energy data, we add 5% (2731 samples) of source domain (S4) data on the data of S5, S49 and NIST as the MMD values between these target domains with S4 are higher. We add 20% (10927 samples) of S4 data on S8 and S15 as the MMD values between these target domains with S4 are lower (see details on Appendix-Ablation Study). Similarly, for MORE data, we add 5% (6796 samples) of source domain (WT11) data on the data of all the target domains. In Fig 2b, we show the comparison between different model training baselines: Exclusive Training and Before fine-tuning, with our One-step fine-tuning approach. Here, we observe that the fine-tuned target model performs up to 52.6% better than the baselines. This is because the fine-tuned target model can generalize well to new, unseen data through the learned general features and patterns from the source model, mitigating the problem of data shift. The baselines do not consider the data shift problem and perform well only if the test data follows the distribution of the training data.

In Fig 3 and Fig 4, we illustrate the prediction performance of the fine-tuned target models on the target domains NIST and WT8, respectively. Here, we compare our One-step fine-tuning approach with other fine-tuning baselines, where we can see that the fine-tuned target models obtains improved prediction of the indoor temperature and wind power for both target domains as compared to the baselines.

In Table 3, we show evaluation results for our One-step fine-tuning approach under data shift scenario. This is done by directly applying both the pre-trained (source) and fine-tuned target models on other new target domains, i.e., without exclusive training and fine-tuning, respectively. We observe that the performance of the fine-tuned target model is better than the source model for all the other target domains, even though having different MMD values (see Table 7 and Table 8 in Appendix) that indicates data distribution scenario. This validates our claim to efficiently handle data shift problem; the fine-tuned target model performs better on unseen data by improving generalization, leading to better domain adaptation.

Next, in Fig 5, we show the performance of the fine-tuned target models directly evaluated on the source domains S4 and WT11 for catastrophic forgetting check. Here, we observe the fine-tuned target models using our One-step fine-tuning approach outperforms on both S4 and WT11 than the fine-tuning baselines GU and EWC. The One-step fine-tuning method achieves RMSE that is better or closer to the performance of the source model on S4, i.e., 0.236. We obtain similar results for WT11, where the RMSE was 0.118. This shows the fine-tuned model retains its knowledge learned during pre-training while adapting better to the new target domain data; hence, mitigating the problem of catastrophic forgetting by adding some percentage of data samples of S4 and WT11 to the respective target domains during model training.

Conclusion

Transformers improve time series prediction by effectively capturing long-term patterns and dependencies, improving the model's capability to understand and predict complex temporal relationships. However, their performance suffers severely due to limitations like insufficient temporal understanding, generalization challenges, data shift issues, and the critical concern of catastrophic forgetting. Therefore, in this paper, we propose a new One-step fine-tuning approach where we pre-train the time series Transformer model on a source domain with sufficient data and fine-tune it on the target domain with limited data. Our approach builds on a gradual unfreezing technique, however, with addition of source domain data in the target domains to fine-tune the pre-trained model. This helps enhance the model's performance in time series prediction for domains with limited data. In the future, we will explore privacy aspects of our One-step fine-tuning when sharing source domain data.

Acknowledgments

This work has been partially supported by the MORE project (grant agreement 957345), funded by the EU Horizon 2020 program.

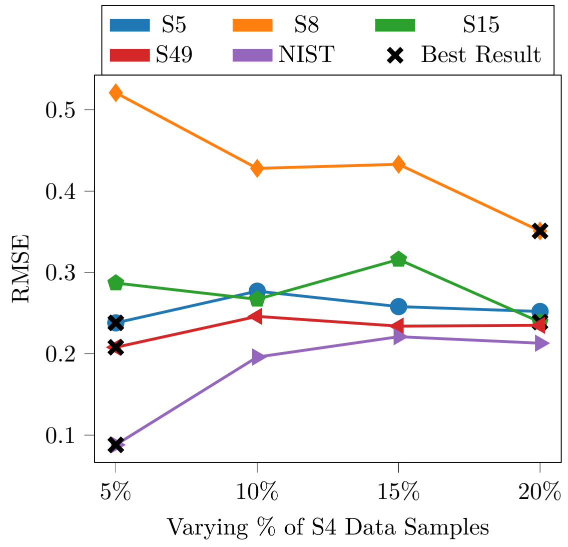


Figure 6: Performance of One-step fine-tuning when varying the percentage of data samples from source domain S4.

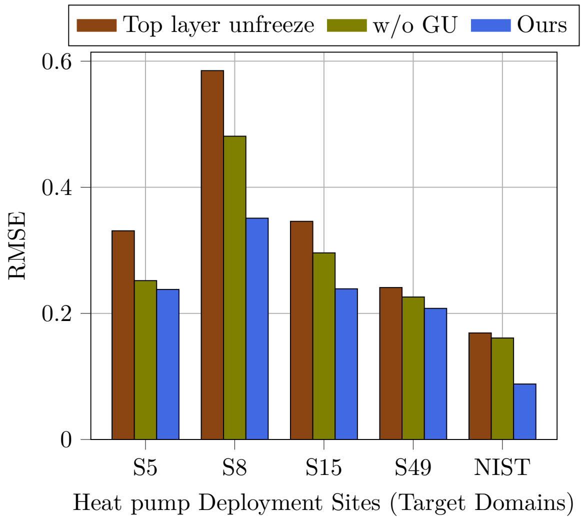


Figure 7: Performance of One-step fine-tuning compared with different freezing techniques.

Appendix

Dataset Overview

This section provides an overview of the training and evaluation datasets used during the pre-training and fine-tuning phases for Energy and MORE data. Table 4 shows the amount of data used for training and testing the source model and fine-tuned target models. Here, we can observe that the source domains S4 and WT11 have sufficient model training data compared to the target domains, even after we add the certain amount of data from source domain to the target domains.

Hyper-parameters used during pre-training

Table 5 shows the summary of parameters used during pre-training the source model in the selected source domains S4 and WT11 for Energy and MORE data, respectively. The hyper-parameters are obtained through grid search.

Ablation Study

We also performed additional experiments on Energy data for an ablation study of our design decisions. In Fig 6, we show how adding a certain amount of data samples from the source domain affects the performance of target domains during source model fine-tuning. As can be seen, for the target domains S5, S49, and NIST, adding 5% of data from the source domain performs better compared to other percentages of added data, as the MMD values between these target domains with S4 are higher (see Table 6). Likewise, for the target domains S8 and S15, adding 20% of data from the source domain works best compared to other percentages of added data, as the MMD values between these target domains with S4 are lower (see Table 6).

Next, in Fig 7, we experimented with various techniques for freezing pre-trained layers. We compared our One-step fine-tuning approach with two different methods: Top layer unfreeze and w/o GU. For the Top Layer Unfreeze approach, we only unfreeze the top layer-specifically, the decoder layer of the source model. We then fine-tune the model using the GU technique. In the w/o GU approach, we unfreeze all the layers at once at the initial stage of fine-tuning rather than gradually unfreezing the source model layers. Our One-step fine-tuning approach, where we gradually unfreeze all the pre-trained layers after adding some percentage of source domain data to the target domain, performs better than the other two approaches. Here, we can see an improvement of up to 40%.

Distribution Alignment Analysis

In this section, we delve into a thorough analysis of the alignment between the data distributions of the source and different target domains. Table 6, Table 7, and Table 8 present the MMD values, which serve as a crucial metric for quantifying the dissimilarity or similarity between these domains.

Table 6 shows the calculation of MMD values between the source and target domains for Energy data. These target domains were selected for source model fine-tuning. Here, the varying MMD values among the target domains, ranging from 0.089 to 0.133, show the discrepancies in the alignment between the source and individual target domains. Notably, the lower MMD values, such as 0.089, indicate a more effective alignment, indicating a closer resemblance in the feature distributions between the source and those specific target domains. This potentially signifies more robust adaptability of the model to those domains. On the other hand, higher MMD values, like 0.133, imply a more sig-

Table 4: Overview of dataset used during pre-training and One-step fine-tuning.

Energy Data			MORE Data		
Domain	Training Data	Test Data	Domain	Training Data	Test Data
Source Domain (S4)	38,245	16,390	Source Domain (WT11)	95,148	40,777
S5	4097	1755	WT4	10,850	4650
S8	10,070	4316	WT6	10,794	2587
S15	8830	3784	WT7	9994	2569
S49	5700	2442	WT8	8128	2626
NIST	5439	2331	WT9	9014	2577

Table 5: Parameters used in source model pre-training.

Parameters	Values	
	Energy Data	MORE Data
No. of Input features	1	18
Target Variable	Indoor Temperature	Minimum Wind Power
Batch Size	8	64
Training Epoch	50	100
Historical values	96 (i.e. 24 hours)	24 (i.e. 24 hours)
Prediction horizon	4 (i.e. 1 hour)	4 (i.e. 4 hours)
Learning rate	0.001	0.0001
Training Set: Test Set	70%:30%	
Optimizer	Adam	
Model Loss Function	Mean Absolute Error (MAE)	

Table 6: Calculation of MMD values between source and target domains for Energy data.

Source Domain	Target Domain	MMD values
S4	S5	0.114
	S8	0.097
	S15	0.089
	S49	0.104
	NIST	0.133

nificant dissimilarity, highlighting challenges in aligning certain target domains with the source.

Next, Table 7 and Table 8 shows the calculation of MMD values between the source and various target domains for Energy and MORE data, respectively. Here, the other target domains were not involved during source model fine-tuning. The tables not only provide the perspective of the alignment between the source and individual target domains but also of the disparities among the various target domains themselves. These MMD values help provide insight into examining data shifts between the source and different target domains in the domain adaptation process. This analysis allows us to see how well the domain adaptation strategy works and how the fine-tuned target model works well across different domains.

Table 7: Calculation of MMD values between source and target domains for Energy data.

Source Domain	Other Target Domains	MMD values	Target Domain	Other Target Domains	MMD values
S4	S3	0.127	S15	S3	0.058
	S7	0.153		S7	0.157
	S19	0.131		S19	0.107
	S50	0.167		S50	0.100
Target Domain	Other Target Domains	MMD values	Target Domain	Other Target Domains	MMD values
S5	S3	0.110	S49	S3	0.117
	S7	0.164		S7	0.216
	S19	0.125		S19	0.170
	S50	0.150		S50	0.158
Target Domain	Other Target Domains	MMD values	Target Domain	Other Target Domains	MMD values
S8	S3	0.230	NIST	S3	0.069
	S7	0.083		S7	0.170
	S19	0.088		S19	0.123
	S50	0.240		S50	0.110

Table 8: Calculation of MMD values between source and target domains for MORE data.

Source Domain	Other Target Domains	MMD values	Target Domain	Other Target Domains	MMD values
WT11	WT2	0.079	WT4	WT2	0.077
	WT3	0.084		WT3	0.084
	WT5	0.095		WT5	0.092
	WT9	0.095		WT9	0.093
Target Domain	Other Target Domains	MMD values	Target Domain	Other Target Domains	MMD values
WT6	WT2	0.080	WT7	WT2	0.079
	WT3	0.081		WT3	0.093
	WT5	0.095		WT5	0.097
	WT9	0.100		WT9	0.095
Target Domain	Other Target Domains	MMD values	Target Domain	Other Target Domains	MMD values
WT8	WT2	0.085	WT9	WT2	0.080
	WT3	0.085		WT3	0.083
	WT5	0.100		WT5	0.098
	WT9	0.099		WT9	0.100