ReCycle: Fast and Efficient Long Time Series Forecasting with Residual Cyclic Transformers

Arvid Weyrauch
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Gemany
arvid.weyrauch@kit.edu

Thomas Steens
German Aerospace Center (DLR)
Oldenburg, Germany
t.steens@hotmail.com

Oskar Taubert
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Gemany
oskar.taubert@kit.edu

Benedikt Hanke
German Aerospace Center (DLR)
Oldenburg, Germany
benedikt.hanke@dlr.de

Aslan Eqbal
INENSUS GmbH
Goslar, Germany
a.eqbal@inensus.com

Ewa Götz
Siemens AG, Digital Industries
Karlsruhe, Germany
ewa.goetz@siemens.com

Achim Streit
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Gemany
achim.streit@kit.edu

Markus Götz
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Gemany
markus.goetz@kit.edu

Charlotte Debus
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Gemany
charlotte.debus@kit.edu

***Abstract*—Transformers have recently gained prominence in long time series forecasting by elevating accuracies in a variety of use cases. Regrettably, in the race for better predictive performance the overhead of model architectures has grown onerous, leading to models with computational demand infeasible for most practical applications. To bridge the gap between high method complexity and realistic computational resources, we introduce the Residual Cyclic Transformer, ReCycle. ReCycle utilizes primary cycle compression to address the computational complexity of the attention mechanism in long time series. By learning residuals from refined smoothing average techniques, ReCycle surpasses state-of-the-art accuracy in a variety of application use cases. The reliable and explainable fallback behavior ensured by simple, yet robust, smoothing average techniques additionally lowers the barrier for user acceptance. At the same time, our approach reduces the run time and energy consumption by more than an order of magnitude, making both training and inference feasible on low-performance, low-power and edge computing devices. Code is available at https://github.com/Helmholtz-AI-Energy/ReCycle**

***Index Terms*—time-series, neural networks, transformer, energy efficiency**

I. Introduction

Among the different applications of machine learning (ML) methods, time series forecasting is one of the most widely encountered and most complex tasks. While for a long time, traditional statistical methods, such as smoothing averages or auto-regressive moving averages, dominated the algorithmic landscape for analysis and prediction of temporal behavior, recent advances in deep learning (DL), and natural language processing (NLP) in particular, have paved the way for neural network-based approaches [1]. The introduction of the Transformer architecture [2] has preluded significant breakthroughs in sequence processing, making it a natural candidate for state-of-the-art (SOTA) time series forecasting. However, adaptation to time series applications poses a mayor challenge: The calculation and memory complexity of the attention mechanism for a sequence of length L is $\mathcal{O}(L^2)$, making the forecasting of long time series exceptionally compute intensive. For real-time deployment on hardware with limited memory and computation capabilities, this computational footprint is prohibitive. Moreover, dot-product attention was originally designed for multi-feature tokens, complicating the method transfer to univariate time-series.

Several authors have put forward approaches for general Transformer-based time series forecasting problems that attempt to tackle the computational complexity of the attention mechanism [3]–[5]. However, while they theoretically reduce the complexity to logarithmic or even linear, practical implementations do not yield significant computational performance increase, as shown by our experiments. The increased computational overheads and complex model structures ultimately yield longer run times and consequently, higher energy consumption. This renders them infeasible for typical deployment hardware, such as on-board systems, embedded devices, or low-power hardware in sensors. But more importantly, it contributes to the steadily increasing environmental footprint of AI methods that has accompanied the drive for more accurate models, a trend which has been termed Red AI [6].

This work is supported by the Helmholtz Association Initiative and Networking Fund under the Helmholtz AI platform grant and the HAICORE@KIT partition and by the German Federal Ministry of Education and Research under the 01IS22068 - EQUIPE grant. The authors gratefully acknowledge the computing time made available to them on the high-performance computer HoreKa at the NHR Center KIT. This center is jointly supported by the Federal Ministry of Education and Research and the state governments participating in the NHR (www.nhr-verein.de/unsere-partner).

979-8-3503-5409-6/24/$31.00 ©2024 IEEE
DOI 10.1109/CAI59869.2024.00212

In this work, we present ReCycle: Residual Cyclic Transformers, a method for fast and energy efficient time series forecasting. Our contributions include a rigorous mathematical discussion of the attention mechanism for single-feature sequences, leading to something that we term *scalar breakdown of dot-product attention*. The corresponding incapability of the attention mechanism to capture relational properties between single-featured tokens has direct implications for univariate time series forecasting, which adds to our motivation for this work.

We propose two conceptual changes in representing temporally resolved data, to tackle the growing computational demand in long-time series forecasting with Transformer-based architectures:

1) Primary Cycle Compression (PCC): Real-world time series applications often exhibit distinct patterns and characteristics, such as daily, weekly, and seasonal cycles. We use this fact to our advantage, converting univariate time series into multivariate time series over these cycles. PCC addresses the scalar breakdown of dot-product attention in an easy and intuitive way. Furthermore, it naturally bypasses the memory and computation bottleneck, leading to simpler, yet computationally faster and more energy efficient time series forecasting.
2) *Residual Learning*: We leverage prior knowledge on periodicity and underlying temporal profiles, and accounts for temporal locality such as concept drift through *recent historic profiles*, a new form of smoothing average naive forecast. By learning only differences to these RHP, i.e., residuals, we achieve improved prediction accuracy as the model can focus specifically on these modulations instead of the often predominant but already known periodicity.

We evaluate our proposed method on two representatives SOTA Transformer-based architectures for a number of different datasets, demonstrating that ReCycle can be easily integrated with existing Transformer-based architectures, leading to superior forecasting while requiring a fraction of their run time and energy consumption. We further show, that with usage of ReCycle, Transformer-based architectures clearly outperform current non-Transformer models.

II. Related Work

Neural methods for long time series forecasting have undergone massive development since the publication of the Transformer architecture [2]. A comprehensive review can be found in [7]. LogTrans [8] was the first notable adaptation of the Transformer specifically for time series forecasting. To address the issue of local context insensitivity of the self-attention mechanism, the authors substituted the point-wise dot-product with causal convolutions. A sparse bias in form of a LogSparse mask was used to reduce computational complexity to $\mathcal{O}(L \log L)$. Informer [3] focuses on dimensionality reduction via random subsampling of attention queries. A metadata input representation related to positional encoding is used to transform univariate time-series into multivariate ones. Autoformer [4] introduced a local mean-based decomposition method and replaces the dot product attention with an auto-correlation mechanism based on Fourier transforms for lower complexity. FEDformer [5] combines these ideas by selecting a fixed number of Fourier modes for auto-correlation and introducing a decomposition scheme using multiple filter lengths. Pyraformer [9] proposed the pyramidal attention module that uses inter-scale tree structures and intra-scale neighboring connections to leverage multi-resolution representations of time series. The recently introduced Crossformer [10] focuses on multivariate time-series, leveraging cross-dimension dependencies through dimension-segment-wise embedding and a two-stage attention layer.

Recent work on Transformer-based architectures for long time series forecasting has investigated approaches that are similar to individual components of ReCycle. The ETS-Former [11] exploits exponential smoothing attention and frequency attention to replace the self-attention mechanism in vanilla Transformers, thus improving both accuracy and efficiency. PatchTST [12] proposes an approach not too different from our PCC, by segmenting the time series into subseries-level patches which are used as input tokens to Transformer. Furthermore, PatchTST promotes channel-independence for multivariate time-series, attributing the same embedding and Transformer-weights to each channel.

While these models address many of the inherent challenges for the application of Transformers to time series, their approaches are designed for generic time series and thus do not leverage any problem-specific features or properties. Moreover, their typically highly complex architectures, used to make the model generalize better, introduce significant practical implementation overhead and higher demand for computational resources. A recent study questions the feasibility and efficiency of Transformers for time series forecasting, indicating that linear models might yield better predictive performances [13]. In response to the excessive demand in computational resources, researchers have turned to model approaches based on multi-layer perceptrons (MLP), such as NBEATS [14] or NHiTS [15], showing on-par and even improved accuracy compared to Transformer-based approaches.

III. Notation

Let $x(t)$ be a time-dependent, continuous variable with f features at a point in time t. A time series $\mathbf{X}$ is a sequence of N measurements of x over a time span T, taken at times $t_0, t_1, ..., t_N$ with a temporal resolution of $\Delta t = T/N$. We consider the time series forecasting problem as finding a mapping $\mathcal{M}$, such that

$$\mathcal{M}\left(x(t_{i-H}), \ldots, x(t_{i-1}), x(t_i)\right) \rightarrow x(t_{i+1}, \ldots, t_{i+F})$$

for every $x(t_i) \in \mathbf{X}$. For simplicity, we will abbreviate $x(t_i) := x_i$ moving forward. H is the *historic window length* and F is the *forecast window length*.

IV. Scalar Breakdown of Dot-Product Attention

The success of the canonical Transformer is founded in the capabilities to relate individual sequence elements with

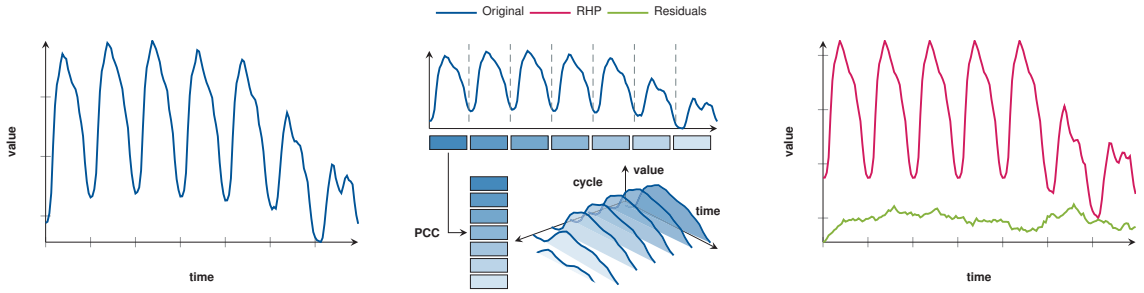

Fig. 1. The concepts of *primary cycle compression* (PCC) and *learning residuals*. First, the original univariate time series (left) is rearranged according to its primary cycles, yielding a 2D data matrix (middle). Due to the similarity in primary cycles, we can compute recent history profiles (RHP) and subtract them from the original data, resulting in residuals that the model is trained to learn (right).

one another through the dot-product self-attention mechanism spanning the attention matrix. However, we conjecture that this does not hold for single-valued sequence elements, i.e. univariate time series. Thus, for single-feature sequences, the attention matrix does not contain meaningful similarity information. We refer to this as the *scalar breakdown of dot-product attention*.

Let $\mathbf{X}$ be a sequences with feature dimension $f = 1$, σ be an activation function, and $\mathbf{Q}$ and $\mathbf{K}$ be the matrices used to map to query and key respectively. Then, the attention matrix element a_{ij} associated with two arbitrary sequence elements $x_i, x_j \in \mathbf{X}$, before the softmax, is calculated as

$$a_{ij} = \frac{\sigma\left(\mathbf{Q}x_i\right)\sigma\left(\mathbf{K}x_j\right)^T}{\sqrt{d_k}} \quad , \tag{1}$$

where d_k is the dimension of the query and key vectors. Since d_k is only introduced to keep the results in a value range where the gradient of softmax is sufficient for effective backpropagation, it can be disregarded. Yet, it also indicates that the values of query and key will generally be small. In this range most common activation functions, excluding ReLU, can be approximated as linear, and since x_i, x_j are scalar we obtain

$$\begin{aligned} a_{ij} &\propto \sigma\left(\mathbf{Q}x_i\right)\sigma\left(\mathbf{K}x_j\right)^T & &\approx x_i x_j \sigma\left(\mathbf{Q}\right)\sigma\left(\mathbf{K}\right)^T \\ &\propto x_i x_j \left(\mathbf{Q}^*\mathbf{K}^{*T}\right) & &\propto x_i x_j \quad . \end{aligned} \tag{2}$$

Hence, we argue that the product of two scalars does not contain meaningful information about their similarity as intended by the approach.

As mentioned, the ReLU activation function cannot be approximated linearly around zero. However, it is explicitly linear on $\mathbb{R}^+$ and zero on $\mathbb{R}^-$. Therefore, the above calculation can be performed similarly by considering the four possible sign combinations of x_i and x_j, resulting in four different proportionality factors but leading to the same conclusion.

In the argument above the weight-biases $\mathbf{b}_Q, \mathbf{b}_K$ were neglected for clarity. However, they can be easily reinserted into Eq. 2:

$$\begin{aligned} a_{ij} \propto& \sigma\left(\mathbf{Q}x_i + \mathbf{b}_Q\right)\sigma\left(\mathbf{K}x_j + \mathbf{b}_K\right)^T \\ \approx& x_i x_j \sigma\left(\mathbf{Q}\right)\sigma\left(\mathbf{K}^T\right) + x_i \sigma\left(\mathbf{Q}\right)\sigma\left(\mathbf{b}_K\right) \\ &+ x_j \sigma\left(\mathbf{K}\right)\sigma\left(\mathbf{b}_Q\right) + \sigma\left(\mathbf{b}_Q\right)\sigma\left(\mathbf{b}_Q\right) \quad . \end{aligned} \tag{3}$$

None of the three additional terms contain both x_i and x_j and therefore cannot contribute to the similarity measure.

We hypothesize, that the authors of previous Transformer architectures for time series forecasting were implicitly aware of problems in applying dot-product attention to univariate time series, hence enriching the data through feature enhancement and positional encoding [3], [4], altering the attention mechanism e.g. convolutional attention [8] or simply focusing on multivariate time series. However, to the best of our knowledge, the deduction above is the first explicit reasoning for the scalar breakdown of dot-product attention, and it motivates our introducing the primary cycle compression.

V. Methodology

To tackle the scalar break-down of dot-product attention for univariate time-series and to bring computational demand, runtime and energy consumption of Transformer-based architectures for long-term forecasting to a feasible level while improving forecast accuracy, we propose ReCycle, which is based on two novel concepts.

A. Primary Cycle Compression (PCC)

Transformer architectures for time series forecasting usually employ a sliding window that is applied to every time step. While this creates a flexible model with respect to the starting point of the forecast, it also introduces two problems. Firstly, it causes data redundancy: depending on the length of historic and forecast windows a data point will be used tens or hundreds of times in each model training pass, i.e. epoch. Secondly, Transformers work by determining the similarity of sequence elements by comparing elements primarily based on their absolute magnitude. In time series forecasting, it is more relevant to know whether a sequence element is located in an ascending or descending slope, a peak, or a minimum.

We address both these shortcomings using what we coin *primary cycle compression* (PCC). The reason behind this is as simple as it is intuitive: in many time series one can observe periodicity in the data, a pronounced and stable *primary cycle* (see Fig. 1). This cycle is often, but not necessarily, the day-night cycle. For example, road traffic or demand for resource, such as electricity or water, align with daily

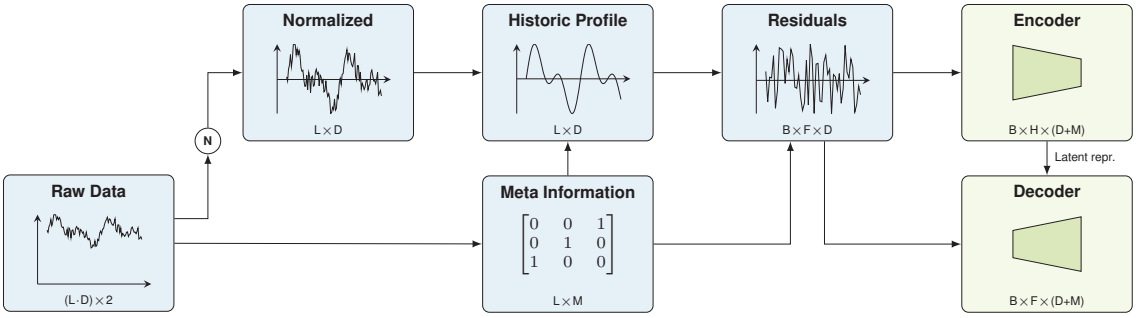

Fig. 2. Schematic overview of the data flow in ReCycle. Boxes represent building blocks, edges information flow, and tensor shapes are denoted at the bottom of each box.

activity of the population and industry, and hence exhibit a clear $1\,\mathrm{h}$ cycle. Similar daily profiles can be observed in certain climate or weather data, like temperature or solar irradiation. Our approach uses this primary cycle as sequence granularity, expressing each step in the form of a vector with D entries, where D is the primary cycle length. PCC enforces a fixed cycle start, however in practical applications long term forecasts are typically not performed every time step, but at the scale of the primary cycle. Furthermore, it allows the dot product attention to compare the overall shapes of daily profiles, providing more relevant similarity measures. Additionally, PCC reduces both sequence length, N, and number of samples by a factor of D while still utilizing the entire data content, thus mitigating many practical limitations originating from the $\mathcal{O}(N^2)$ memory and computational complexity of the attention mechanism. Finally, PCC allows for natural inclusion of metadata M available only on a whole-day basis, e.g. holiday vs. non-holiday; daily minimums, averages, or maximums; weather forecasts; etc.

B. Recent Historic Profiles and Residuals

PCC allows us to incorporate knowledge about predominating patterns in the data, focusing the model on learning more difficult to capture temporal dependencies beyond the trivial periodic patterns, as illustrated in Figure 1.

Recurring patterns may be captured by averaging several past instances of the data, resulting in so-called *historic profiles* (HP). There are different ways for averaging historic data and typically each application use-case necessitates a custom combination of measurements. The simplest form of a historic profile is *persistence*, where data from the last time step i is used to forecast the next one, $i + 1$. Following empirical evaluation of different averaging techniques, we identify *recent history profiles* (RHP) as an innovative and precise technique for a wide range of use cases. However, all further model derivations can be easily translated to other averaging techniques. For calculation of RHP we distinguish between K different types, which enabled incorporation of prior knowledge. A typical example for different categories would be workdays, saturdays and sun-/holidays, since many datasets exhibit distinct patterns in these categories. Instead of considering all past data we only consider the last k days in the appropriate category. That way, the profiles naturally account for current circumstances (like weather), seasonality, behavior change and concept drift.

Historic profiles capture the general time course of the primary cycle rather well, but smooth out faster, more erratic components. To let the model focus on these hard-to-predict modulations rather than the already known predominant general time course, we propose a residual forecasting approach, were input and target of the network are the difference between the RHP and the actual time curve, i.e. the *residual* (see Fig. 1). Since the overall magnitude of the time series influences the forecast of residuals, we provide the RHP as decoder input. In addition to providing a baseline, this also means the decoder input is shorter than in previous approaches [3], resulting in a trivial reduction of computations. With the residual approach, the network can easily learn to return zero if it is unable to find further patterns on top of the RHP, ensuring a robust fallback behavior of the prediction.

C. ReCycle

Our approach ReCycle combines PCC and residual forecasting based on RHP into one stand-alone preprocessing step that can be applied to any kind of Transformer-based model architecture, and in fact any time-series forecasting model. ReCycle works as follows: Time-series data is normalized to the interval $[0, 1]$. PCC is then performed on the whole dataset, i.e. a univariate time series with N time steps is rearranged into a 2D data matrix of size $L \times D$, with $L = N/D$ and D being the length of the primary cycle. D is an application-specific parameter. Metadata, such as weekday and holiday information extracted from time stamps of each sample, is then concatenated as additional features. RHPs are calculated for each sample in the dataset, regarding the last k days of category K. Residuals are then determined by subtracting the respective RHP from the original data for each time step.

The resulting dataset is then fed to the actual model, which can be any kind of encoder-decoder architecture for sequence-to-sequence modeling. We will elucidate the process on the example of the vanilla Transformer architecture, depicted in Fig. 2: The model takes the sequence of historic residuals of

size $H \times (D + M)$ as input, where H is the length of the historic window. Residuals are encoded through one multi-head attention layer into the hidden state and passed to the decoder. The decoder takes the hidden state and the forecast RHP of size $F \times (D + M)$ as input and outputs forecast residuals.

Since the output length of the prediction sequence is known a-priori in time series forecasting, we use *single pass decoding* (SPD) in both training and inference, i.e. predicting all sequence elements at once instead of one sequence element after another, as is common for Transformers in NLP. This approach has been widely adopted since its introduction in [3]. SPD comes with two advantages: For one, it reduces computational complexity, since it allows leveraging the natural vector processing capabilities of the attention mechanism. It further grants better information utilization [4], [5] and provides a significant contribution to performance improvements [13]. We also use it to forego masking, allowing the model to use the known RHP predictions of future time steps.

VI. Experimental Evaluation

We evaluate ReCycle as an extension to three current SOTA Transformer-based models on a number of time series forecasting problems. Results were compared in terms of prediction accuracy, training time and energy consumption for training and inference.

A. Benchmarks

The evaluation task is defined as predicting values of the next $F = 7$ days (168 hours), based on data from the last $H = 21$ days (504 hours). Three representative Transformer-based architectures were chosen: the vanilla Transformer, FEDformer and PatchTST. While originally developed for NLP sequence prediction, the *Vanilla Transformer* as described by [2] can be used to directly encode historic data and forecast future temporal behavior. *FEDformer* [5] and *PatchTST* mark current state-of-the-art Transformer variants for long time series forecasting. For Transformer and FEDformer we use the implementations provided by [5][1]. For PatchTST we use the original implementation provided by the authors[2]. All three Transformer-based models were trained with and without ReCycle.

We further include the NHiTS model [15] into our experimental evaluation. NHiTS is a non-Transformer approach that utilizes hierarchical interpolation and multi-rate data sampling to improve prediction accuracy and reduce training time. We use the implementation provided by the authors [3] and train it with the standard setup. For reference, we report predictive performance of the RHP we use to calculate residuals.

a) Hyperparameters: Hyperparameters of the FEDformer, PatchTST and NHiTS were set according to the original publications. Since Transformer was originally designed for sequence processing, we conducted a hyperparameter-search using Propulate [16]. Training with and without ReCycle was conducted using the same set of hyperparameters for each model.

b) Setup: For application of ReCycle, PCC is performed as described above. We consider the primary cycle to be the daily one, i.e. we fix $D = 24\,\mathrm{h}$. Recent historic profiles are extracted by categorizing each sample into one of $K = 3$ categories: Weekday; Saturday; Sun- or Holiday. The RHP for each sample is then calculated by averaging the last $k = 3$ samples prior to the sample date in a given category. This implies that at least three weeks of data have to be available before a prediction can be made. Furthermore we use weekday and holiday information, extracted from time stamps of each as metadata features, such that weekday information is one-hot encoded and holiday information for the current and following day is binary encoded. Thus, there is a total of $M = 9$ metadata features. Models are then trained using the Adam optimizer, without an additional learning rate schedule. Since ReCycle is trained to predict residuals, which intrinsically are more noisy compared to the original data, we optimize for mean absolute error (MAE) loss, which provides higher resilience to outliers than mean square error (MSE) loss.

B. Datasets

Five representative time series from different datasets were chosen for experimental evaluation. ***Electricity Load Diagrams***[4] (ELD) contains electricity consumption of 370 household consumers in Portugal, taken between January 2011 and December 2014 at a temporal resolution of $15\,\mathrm{min}$. Following the approach first used in [3] for univariate forecasting we only use the column 'MT_320'. ***Western European Power Consumption***[5] (ENTSO-E) contains time-resolved measurements of total electricity consumption of Germany, collected by the ENTSO-E, taken between January 2015 and August 2020 at a temporal resolution of $15\,\mathrm{min}$. ***Water Demand***[6] (Water) contains time-resolved measurements of water consumption in a water supply network, taken between January 2016 and March 2021 at a temporal resolution of $1\,\mathrm{h}$. Measurements of water consumption were collected at a pumping station in a regional water supply network in Germany for both household and industrial consumers. ***Traffic***[7] (Traffic) contains time-resolved measurements of water consumption in a water supply network, taken between January 2016 and March 2021 at a temporal resolution of $1\,\mathrm{h}$. It can be found on https://pems.dot.ca.gov under Freeways $\rightarrow$ I280-N $\rightarrow$ Performance $\rightarrow$ Aggregates $\rightarrow$ Time Series. ***Electricity Transformer Temperature*** (ETT)[8] contains measurements of power load features and oil temperature of two electricity transformers in

[1] https://github.com/MAZiqing/FEDformer
[2] https://github.com/yuqinie98/PatchTST
[3] https://github.com/cchallu/n-hits

[4] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[5] https://www.kaggle.com/datasets/francoisraucent/western-europe-power-consumption
[6] This dataset was provided by Siemens AG and can be found at https://doi.org/10.5281/zenodo.11045013
[7] https://pems.dot.ca.gov/?dnode=Freeway&content=loops&tab=det_timeseries&fwy=280&dir=N
[8] https://github.com/zhouhaoyi/ETDataset

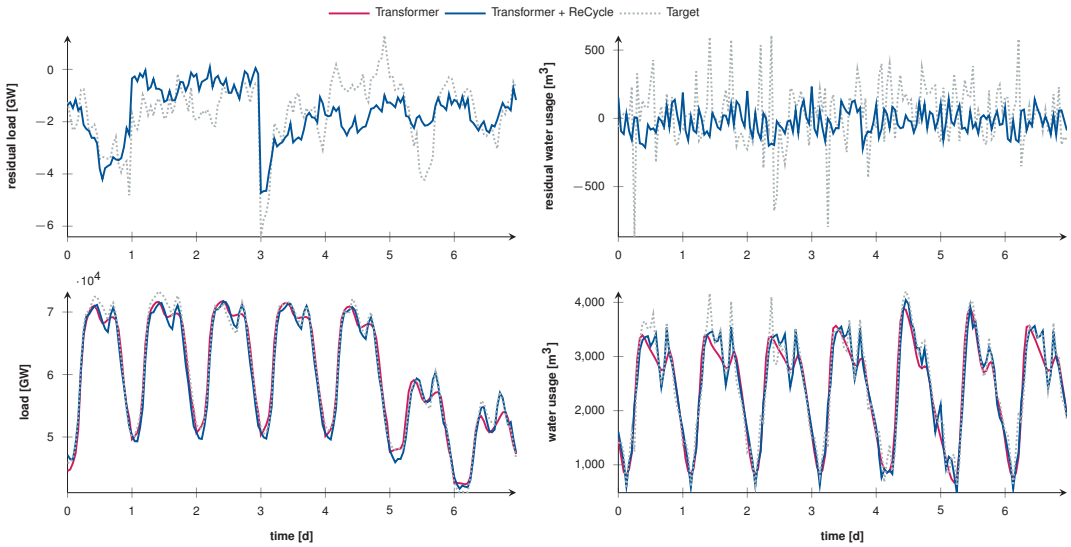

Fig. 3. Exemplary plots of target and predicted residuals (top) and full sample (bottom), for the two datasets ENTSO-E (left) and Water (right).

China, taken between July 2016 and June 2018. We use the dataset of the second transformer with a temporal resolution of 1 h (ETTh2). Where necessary datasets are downsampled to 1 h. Missing values are filled in with the average of the two neighboring values since this trivially generalizes to any resolution. Each dataset is separated into training, validation and test set in the ratio 6:2:2 along the temporal axis.

C. Compute Infrastructure

All models were run on a single node of a supercomputing cluster, equipped with two Intel Xeon "Ice Lake" processor cores and 4 NVIDIA A100 Tensor Core GPUs. The system allows for measurements of whole-node energy consumption via sensors of Lenovo's XClarity Controller (XCC), which can be read via IPMI and SLURM. Models were implemented in Python 3.9.2 using the PyTorch framework [17] versioned 1.13.1+cu116 with CUDA version 11.6. Runs were performed using a single NVIDIA A100-40 GPU for each model.

D. Results

Table I summarizes the predictive performance of training runs of all studied approaches.

We report mean square error (MSE) as well as mean absolute percentage Error (MAPE), which is the mean absolute error (MAE) normalized to the original value. In addition to prediction accuracy metrics, we measure run time as well as total energy consumption of model training for model architectures, as they are known to be computationally complex and are thus energy-hungry. Towards this end, each model is evaluated three times and the average time provided, while energy consumption is measured as the total consumption of all three evaluations.

Throughout all datasets, the naive approach of using plain RHP yields predictive performance competitive with the evaluated DL-models. Nonetheless, training a Transformer-based model architecture on RHP residuals always yields improved forecasting, demonstrating the benefit of these models of simple statistical methods.

Results show, that no clearly superior model can be identified and that in fact, depending on the datasets, different approaches yield the best prediction accuracy. Furthermore, the model yielding lowest MSE differs from the model yielding lowest MAPE in almost all cases. For the ENTSO-E and Traffic datasets, adding ReCycle improves both MSE and MAPE for all Transformer-based approaches. For Water and ELD datasets, plain FEDformer yields lowest MSE values, but not lowest MAPE values. On those two datasets, adding ReCycle to the vanilla Transformer and PatchTST also improves MSE and MAPE. Further investigation showed, that these two datasets consist of very noisy time series. Hence, only few patterns remain in the residuals and the model detects almost exclusively noise, a illustrated in Fig. 3. In that case, ReCycle provides a reliable fallback behavior, where the model returns the RHP prediction, if it cannot identify recurring patterns. Thus, the simple nature of the RHP provides a worst case scenario that is both explainable and robust. The ETTh2 datasets constitutes a special case, as application of ReCycle clearly improves MAPE values, but not MSE values. This is likely caused by the fact that models with ReCycle were trained on optimizing MAE, whereas the vanilla versions are trained on MSE loss. Table II shows ablation with respect to forecast window length on two datasets, which demonstrates consistent behavior across for the different forecast window lengths.

The true benefit of using ReCycle becomes evident when

TABLE I
FORECASTING PERFORMANCE OF TRANSFORMER-BASED MODELS WITH AND WITHOUT USAGE OF RECYCLE ON FIVE DATASETS FOR A FORECAST WINDOW OF ONE WEEK. BEST RESULTS ARE HIGHLIGHTED IN ITALICS. MAPE IS PROVIDED IN PERCENT. MSE IS RESCALED PER DATASET FOR READABILITY, THE RESPECTIVE ORDERS OF MAGNITUDE ARE NOTED IN SQUARE BRACKETS NEXT TO THE DATASET.

		MSE	MAPE
ENTSO-E [10^6]	Transformer	8.53±0.31	3.59±0.14
	+ ReCycle	**5.80±0.58**	**3.19±0.17**
	FEDformer	11.5±0.14	4.03±0.03
	+ ReCycle	6.36±0.22	3.25±0.05
	PatchTST	14.5±0.13	4.54±0.02
	+ ReCycle	7.97±0.10	3.49±0.02
	NHiTS	7.90±0.01	**3.19±0.01**
	RHP	8.85	3.89
ELD [10^1]	Transformer	11.30±0.27	11.30±0.44
	+ ReCycle	9.60±0.29	6.25±0.27
	FEDformer	**8.34±0.01**	9.14±0.03
	+ ReCycle	11.70±1.85	7.37±0.84
	PatchTST	7940.0±152.0	**5.52±0.05**
	+ ReCycle	8.88±0.04	5.71±0.01
	NHiTS	9.86±0.23	6.66±0.16
	RHP	9.44	6.52
Water [10^4]	Transformer	16.3±0.2	15.1±0.1
	+ ReCycle	17.4±0.6	14.3±0.1
	FEDformer	**14.3±0.1**	15.6±0.1
	+ ReCycle	17.6±0.3	15.0±0.2
	PatchTST	27.9±1.4	19.8±0.4
	+ ReCycle	14.9±0.1	**13.8±0.1**
	NHiTS	163.9±0.3	61.5±0.1
	RHP	14.5	13.9
Traffic [10^8]	Transformer	2.77±0.14	16.1±2.1
	+ ReCycle	1.43±0.04	10.7±1.4
	FEDformer	2.41±0.03	15.2±0.1
	+ ReCycle	1.28±0.08	9.0±0.9
	PatchTST	2.17±0.04	12.8±0.5
	+ ReCycle	**1.23±0.01**	**7.3±0.1**
	NHiTS	2.46±0.01	9.3±0.2
	RHP	2.02	9.7
ETTh2 [10^1]	Transformer	4.00±0.25	37.1±1.1
	+ ReCycle	3.64±0.18	17.7±0.3
	FEDformer	4.11±0.04	28.4±1.5
	+ ReCycle	4.37±0.17	18.7±0.4
	PatchTST	**2.15±0.01**	139.0±0.6
	+ ReCycle	4.33±0.01	18.9±0.1
	NHiTS	4.39±0.05	**17.3±0.1**
	RHP	5.03	20.99

considering training time and energy consumption, which is shown in Table III. Results clearly demonstrate, that ReCycle helps to drastically reduce both compute time and energy consumption for training Transformer-based approaches. Notably, the reduction in energy consumption does not stem solely from shorter training times. For the vanilla Transformer, using ReCycle achieves a training speed-up of three to five, while reducing energy consumption by a factor of up to $\approx$ 20 (Water dataset). The biggest improvements can be observed for FEDFormer, where usage of ReCycle yields up to $\approx$ 18 times faster training (ETTh2 dataset) and up to

TABLE II
FORECAST WINDOW LENGTH ABLATION STUDY OF TRANSFORMER-BASED MODELS WITH AND WITHOUT USAGE OF RECYCLE ON TWO DATASETS.

		Seq. Length = 96		Seq. Length = 336	
		MSE	MAPE	MSE	MAPE
ENTSO-E [10^6]	Transformer	7.86±0.44	3.32±0.11	8.92±0.07	3.63±0.04
	+ ReCycle	5.69±0.23	3.17±0.09	5.86±0.33	3.19±0.03
	FEDformer	9.85±0.11	6.69±0.01	13.2±0.67	4.44±0.18
	+ ReCycle	6.23±0.46	3.29±0.11	6.12±0.23	3.26±0.04
	PatchTST	13.6±0.46	4.34±0.06	13.9±.043	4.65±0.09
	+ ReCycle	7.97±0.13	3.50±0.02	8.03±0.08	3.51±0.02
	NHiTS	7.41±0.07	2.93±0.02	6.92±0.40	3.14±0.07
ETTh2 [10^1]	Transformer	38.8±0.9	43.3±3.8	40.8±0.4	31.8±0.4
	+ ReCycle	40.5±3.2	18.3±0.1	38.4±3.3	18.0±0.5
	FEDformer	35.1±0.1	30.8±0.9	46.2±1.1	26.9±1.1
	+ ReCycle	44.7±0.6	19.1±0.1	46.7±2.5	19.4±0.5
	PatchTST	17.2±0.1	119.0±3.13	24.9±0.2	168.0±2.2
	+ ReCycle	43.5±0.1	18.8±0.1	43.4±0.4	18.8±0.1
	NHiTS	37.9±0.3	16.9±0.2	31.9±0.2	12.4±0.1

$\approx$ 35 times lower energy consumption (ELD dataset). In those cases, where FEDformer yields the best forecasting accuracy in terms of MSE, the decrease of $\approx$ 30% (ELD) and $\approx$ 14% (Water) in MSE is to be gauged against a factor of $\approx 13-15$ longer training time and $\approx 22-35$ higher energy consumption. The same holds for PatchTST in the ETTh2 dataset, where conceding a two times higher MSE can reduce training time by a factor of $\approx$ 10 and energy consumption by a factor of $\approx$ 16. NHiTS yields low training times and energy consumption throughout all datasets, however, Transformer-based approaches with ReCycle provide even lower values. For the ENTSO-E and the ETTh2 datasets, the difference between the best MAPE achieved by NHiTS and the runner up Transformer + ReCycle is almost negligible, while the latter runs at $\approx$ 50-75% shorter training and corresponding lower energy.

VII. CONCLUSION

Attributing to its central role in many real world applications, time series forecasting has received a lot of attention in recent years. A plethora of approaches have been published, many of them relying of the Transformer architecture. However, in the thrive for ever better prediction accuracy, the demand in computational resources of current models has far outgrown anything feasible for practical deployment. The steadily increasing energy consumption required to (re)train and run these models not only limits their application e.g. on embedded or edge devices, it also constitutes a further step along the unsustainable path that AI research is currently on. In response to that, we present ReCycle, a method for reducing runtime and energy consumption in long time series forecasting with Transformer-based architectures. ReCycle introduces the concepts of primary cycle compression (PCC) and learning residuals derived from simple smoothing average techniques. The former addresses the issue of scalar breakdown of dot-product attention and bypasses the high computational complexity of Transformer-based architectures, while the latter helps to incorporate prior knowledge and thus

TABLE III
AVERAGE RUN TIME, ENERGY CONSUMPTION AND REDUCTION FACTORS ACHIEVED THROUGH RECYCLE OF TRANSFORMER-BASED MODELS WITH AND WITHOUT USAGE OF RECYCLE ON FIVE DATASETS FOR A FORECAST WINDOW OF ONE WEEK. LOWEST ABSOLUTE VALUES ARE HIGHLIGHTED IN BOLD.

		Time [s]		Energy [W h]	
ENTSO-E	Transformer	161.09	×2.8	111.0	×9.0
	+ ReCycle	**56.96**		**12.3**	
	FEDformer	1242.81	×10.1	723.0	×23.8
	+ ReCycle	123.61		30.4	
	PatchTST	315.62	×3.7	164.0	×10.1
	+ ReCycle	85.71		16.2	
	NHiTS	86.71		19.0	
ELD	Transformer	105.99	×4.5	74.2	×12.5
	+ ReCycle	**23.62**		**5.92**	
	FEDformer	977.55	×15.4	560.0	×34.4
	+ ReCycle	63.30		16.3	
	PatchTST	221.05	×4.4	115.0	×9.8
	+ ReCycle	50.64		11.7	
	NHiTS	81.98		17.9	
Water	Transformer	250.97	×5.6	169.0	×17.8
	+ ReCycle	44.71		**9.5**	
	FEDformer	1243.55	×13.3	712.0	×32.2
	+ ReCycle	93.34		22.1	
	PatchTST	331.55	×7.5	173.0	×16.0
	+ ReCycle	**44.23**		10.8	
	NHiTS	85.8		20.4	
Traffic	Transformer	111.54	×2.9	79.0	×6.4
	+ ReCycle	**38.91**		12.3	
	FEDformer	1108.96	×10.3	646.0	×24.6
	+ ReCycle	107.56		26.3	
	PatchTST	319.91	×7.9	164.0	×19.8
	+ ReCycle	40.33		**8.28**	
	NHiTS	86.61		18.7	
ETTh2	Transformer	73.11	×5.0	50.4	×12.3
	+ ReCycle	**14.54**		**4.1**	
	FEDformer	369.74	×17.9	216.0	×28.8
	+ ReCycle	20.71		7.5	
	PatchTST	152.41	×9.6	80.6	×16.4
	+ ReCycle	15.87		4.9	
	NHiTS	77.65		16.1	

improves prediction accuracy. ReCycle allows models to naturally adapt to concept drift and provides robust and explainable fallback behavior to statistical methods, both of which are highly desirable characteristics for real-world application in critical infrastructure systems.

ReCycle can substantially improve current state-of-the-art Transformer-based models for time series forecasting. We perform extensive evaluation of our approach on three representative model architectures on a variety of of datasets. While our experiments focus mainly on Transformer-based architectures, ReCycle in principle be easily incorporated as add-on into any kind of deep-learning based models.

In our results, we cannot confirm the superiority of neither one of the current SOTA Transformer-based architectures, nor of MLP-based approaches such as NHiTS, that were designed to overcome the drawbacks in computational demand of Transformers. We hypothesize that published results showing a clear advantage of any of these approaches originate from extensive hyperparameter tuning specific to the respective datasets. Contrary to that, when used in practice, the choice of best model varies. Our results demonstrate that regardless of the choice of model architecture, ReCycle can significantly improve forecasting accuracy. But more importantly, it drastically reduces demand in compute resources with respect to training time and energy consumption, thus providing a viable and useful approach to bringing state-of-the-art time series forecasting from theoretical studies into practical application.

REFERENCES

[1] P. Lara-Benitez, M. Carranza-Garcia, and J. C. Riquelme, "An experimental review on deep learning architectures for time series forecasting," *International Journal of Neural Systems*, vol. 31, no. 03, p. 2130001, 2021.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *arXiv:1706.03762 [cs]*, Dec. 2017. arXiv: 1706.03762.

[3] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting,"

[4] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *Advances in Neural Information Processing Systems*, vol. 34, pp. 22419–22430, 2021.

[5] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting," no. arXiv:2201.12740. version: 3.

[6] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[7] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in time series: A survey," *arXiv preprint arXiv:2202.07125*, 2022.

[8] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," *Advances in neural information processing systems*, vol. 32, 2019.

[9] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar, "Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting," in *International Conference on Learning Representations*, 2021.

[10] Y. Zhang and J. Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *The Eleventh International Conference on Learning Representations*, 2022.

[11] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. Hoi, "Etsformer: Exponential smoothing transformers for time-series forecasting," *arXiv preprint arXiv:2202.01381*, 2022.

[12] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," *arXiv preprint arXiv:2211.14730*, 2022.

[13] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are Transformers Effective for Time Series Forecasting?," Aug. 2022. arXiv:2205.13504 [cs].

[14] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," *arXiv preprint arXiv:1905.10437*, 2019.

[15] C. Challu, K. G. Olivares, B. N. Oreshkin, F. G. Ramirez, M. M. Canseco, and A. Dubrawski, "Nhits: Neural hierarchical interpolation for time series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 6989–6997, 2023.

[16] O. Taubert, M. Weiel, D. Coquelin, A. Farshian, C. Debus, A. Schug, A. Streit, and M. Götz, "Massively parallel genetic optimization through asynchronous propagation of populations," 2023.

[17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.