NILMFormer: Non-Intrusive Load Monitoring that Accounts for Non-Stationarity

Adrien Petralia
EDF R&D
Palaiseau, France
Université Paris Cité
LIPADE
Paris, France
adrien.petralia@gmail.com

Philippe Charpentier
EDF R&D
Palaiseau, France
philippe.charpentier@edf.fr

Youssef Kadhi
EDF R&D
Palaiseau, France
youssef.kadhi@edf.fr

Themis Palpanas
Université Paris Cité
LIPADE
Paris, France
themis@mi.parisdescartes.fr

ABSTRACT

Millions of smart meters have been deployed worldwide, collecting the total power consumed by individual households. Based on these data, electricity suppliers offer their clients energy monitoring solutions to provide feedback on the consumption of their individual appliances. Historically, such estimates have relied on statistical methods that use coarse-grained total monthly consumption and static customer data, such as appliance ownership. Non-Intrusive Load Monitoring (NILM) is the problem of disaggregating a household's collected total power consumption to retrieve the consumed power for individual appliances. Current state-of-the-art (SotA) solutions for NILM are based on deep-learning (DL) and operate on subsequences of an entire household consumption reading. However, the non-stationary nature of real-world smart meter data leads to a drift in the data distribution within each segmented window, which significantly affects model performance. This paper introduces NILMFormer, a Transformer-based architecture that incorporates a new subsequence stationarization/de-stationarization scheme to mitigate the distribution drift and that uses a novel positional encoding that relies only on the subsequence's timestamp information. Experiments with 4 real-world datasets show that NILMFormer significantly outperforms the SotA approaches. Our solution has been deployed as the backbone algorithm for EDF's (Electricité De France) consumption monitoring service, delivering detailed insights to millions of customers about their individual appliances' power consumption.

CCS CONCEPTS

• **Hardware → Energy metering**; • **Computing methodologies → Neural networks**.

KEYWORDS

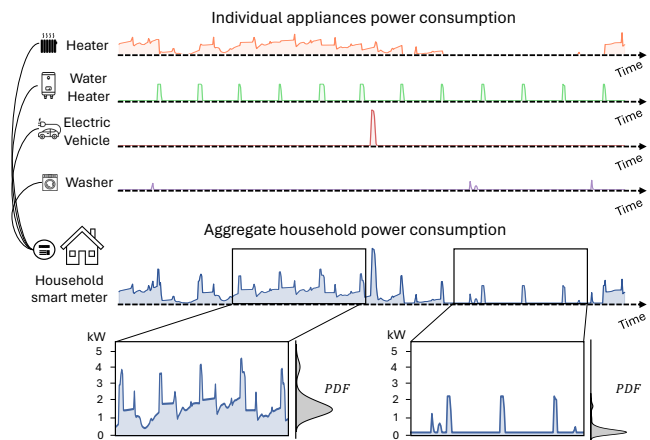Non-Intrusive Load Monitoring, Deep-Learning, Non-Stationarity

Figure 1: Illustration of a smart meter signal, composed of the addition of each individual appliance running in the household, and its non-stationary nature.

1 INTRODUCTION

Efficient energy management is emerging as a major lever for tackling climate change. Thus, enabling consumers to understand and control their own energy consumption is becoming crucial. In this regard, millions of smart meters have been deployed worldwide in individual households [8, 41], recording the total household electricity consumption at regular intervals (typically the average power consumed every 10-30min, depending on the country [64]), providing valuable data that suppliers use to forecast energy demand. Though, as Figure 1 shows, this recorded signal is the aggregated power consumed by *all* appliances operating simultaneously.

EDF (Electricité De France), one of the world's largest electricity producers and suppliers, offers to clients a consumption monitoring solution called *Mon Suivi Conso*, accessible via a web interface and a mobile application [13]. A key feature of this service is providing customers with details about individual appliance consumption.

Historically, only annual appliance-level feedback was available, and estimates relied on semi-supervised statistical methods that used the customers' static information (e.g., appliance presence) and total monthly consumption [14, 15]. However, recent user feedback has increasingly highlighted the wish to have access to fine-grained information [11], prompting EDF to develop a more detailed, accurate, and personalized solution for appliance consumption feedback to enhance customer satisfaction and retention.

A recent internal study at EDF explored Time Series Extrinsic Regression (TSER) approaches [56] to provide monthly appliance-level feedback. Using these approaches, a model is trained to predict the appliance's total monthly consumption based on the monthly smart meter signal, resulting in a marked improvement over previous semi-supervised methods, and reducing the mean absolute error (MAE) of misassigned energy by an average of 70%. Despite this progress, monthly-level feedback remains relatively coarse, limiting its practical utility. Furthermore, recent studies suggest that real-time awareness of energy-intensive appliance usage can lower electricity consumption by up to 12% [2, 4, 12], highlighting the importance of providing finer-grained details.

Non-Intrusive Load Monitoring (NILM) refers to the challenge of estimating the power consumption, pattern, or on/off state activation of individual appliances using only the aggregate smart meter household reading [29, 35, 52]. Since the introduction of Deep-Learning (DL) for solving the NILM problem [30], many different approaches have been proposed in the literature, ranging from Convolutional-based [17, 18, 38, 54, 60, 63] to recent Transformer-based architectures [1, 55, 57, 61, 66]. Nevertheless, to achieve suitable performances (in terms of cost and accuracy), all of the solutions listed above shared the same setting: the DL algorithm takes as input a *subsequence* of an entire household reading to estimate the individual appliances' power. However, as shown in Figure 1, the statistics (illustrated by the Probability Density Function) drastically vary for two subsequences extracted from the same smart meter reading. The intermittent run of electrical appliances causes this phenomenon. For instance, in Figure 1, the heating system, when active, consumes a substantial amount of energy, leading to an increase in the overall power. This change in statistical properties over time is known as the data distribution shift problem and is well known to hurt DL models' performances for time series analysis, especially in Time series forecasting (TSF) [33, 36, 43] In TSF, the solutions operate in a setting similar to the NILM problem, i.e., DL solutions are applied on subsequences of a long series. Recent studies conducted in this area [33, 36] have shown that taking into account this data distribution drift is the key to achieving accurate predictions. To the best of our knowledge, no studies have investigated this issue in the context of energy disaggregation.

In this paper, we propose NILMFormer, a sequence-to-sequence Transformer-based architecture designed to handle the data distribution drift that occurs when operating on subsequences of an entire household consumption reading. For this purpose, NILMFormer employs two simple but effective mechanisms, drawing inspiration from recent studies in TSF [33, 36]. The first one consists of making the input stationary by removing its mean and standard deviation) and passing the removed statistics information as a new token to the Transformer Block (referred to as *TokenStats*). The second one involves learning a projection of the input subsequence's statistics

(mean and std) to denormalize the output signal and predict the individual appliance power (referred to as *ProjStats*). Additionally, NILMFormer employs TimeRPE, a Positional Encoding (PE) based only on the subsequence's timestamp information, which enables the model to understand time-related appliance use. Overall, NILM-Former is significantly more accurate than current State-of-the-Art (SotA) NILM solutions, and drastically reduces the error of misassigned energy, compared to TSER methods (when used to provide per-period feedback, i.e., daily, weekly, and monthly). NILMFormer has been successfully deployed in EDF's consumption monitoring solution *Mon Suivi Conso* and currently provides millions of customers with detailed insights about their individual appliances' power consumption.

Our contributions can be summarized as follows:
• We propose NILMFormer, a sequence-to-sequence Transformer-based architecture for energy disaggregation, especially designed to handle the data distribution drift that occurs when operating on subsequences of an entire consumption series. We also introduce TimeRPE, a PE for Transformer-based model based only on the subsequence's timestamp discrete values, which significantly boosts the performance of NILMFormer compared to standard PEs.
• We evaluate NILMFormer on 4 real-world datasets against SotA NILM baselines and provide an in-depth analysis of the effectiveness of the proposed solution to handle the aforementioned data distribution drift problem.
• We provide insights about the deployment of our solution in EDF *Mon Suivi Conso* monitoring solution, as well as highlighting the value and practicality of the solution for EDF's decision-making.

The source code of NILMFormer is available online [46].

2 BACKGROUND AND RELATED WORK

2.1 Non-Intrusive Load Monitoring (NILM)

NILM [24], also called load disaggregation, relies on identifying the individual power consumption, pattern, or on/off state activation of individual appliances using only the total aggregated load curve [29, 35, 49, 52]. Early NILM solutions involved Combinatorial Optimization (CO) to estimate the proportion of total power consumption used by distinct active appliances at each time step [24]. Later, unsupervised machine learning algorithms, such as factorial hidden Markov Models (FHMM) were used [32]. NILM gained popularity in the late 2010s, following the release of smart meter datasets [21, 31, 34]. Kelly et al. [30] were the first to investigate DL approaches to tackle the NILM problem, and proposed three different architectures, including a Bidirectional Long-Short-Term Memory (BiLSTM). Zhang et al. [63] proposed a Fully Convolutional Network (FCN) and introduced the sequence-to-point framework, which estimates the power consumption of individual appliances only at the middle point of an input aggregate subsequence. However, this solution is not scalable to real-world long and large datasets currently available to suppliers, as the model needs to operate timestamp by timestamp over the entire consumption series to predict the individual appliances' power. Thus, numerous sequence-to-sequence Convolutional-based architecture were later investigated, e.g., Dilated Attention ResNet [60], Temporal Pooling [38] and UNet [18]. With the breakthrough of the Transformer architecture [58] in Natural Language Processing, BERT4NILM [61]

was proposed. Then, variants of these architectures was proposed, such as Energformer [1] that introduced a convolutional layer in the transformer block and replaced the original attention mechanism with its linear attention variant for efficiency. More recently, STNILM [57] is a variant of BERT4NILM that replaces the standard FFN layer with the Mixture-Of-Expert layer [20]. In addition, hybrid architectures were recently investigated, such as BiGRU [51] that mixes Convolution layers and Recurrent Units, and TSILNet [66].

All the studies mentioned above apply their solution to the subsequences of an entire household reading. However, as pointed out in Section 1, this leads to a large distribution drift in each subsequence, hurting model performance. Despite the large number of studies and different proposed architectures, none of them has addressed this issue in the context of energy disaggregation. Moreover, and mainly due to the lack of large publicly available datasets, none of these studies provide insights into the performance of the proposed baselines in real-world scenarios involving high-consuming appliances, such as a Heating system, Water Heater, or Electric Vehicle.

2.2 Non-Stationarity in Time Series Analysis

DL is an established solution for tackling time series analysis tasks [22, 42, 62], including electricity consumption series [37, 47, 48, 50]. However, recent studies conducted in Time Series Forecasting (TSF) pointed out that the non-stationary aspect of real-world time series (e.g., the change in statistics over time) hurts model performance [33, 36, 43]. Indeed, similar to the NILM setting, DL methods for TSF operate on subsequences (called look-back windows); this leads to data distribution drifts over the subsequences. To mitigate this problem, RevIN [33], a stationarization/destationarization scheme, is a solution adopted by most SotA TSF architectures [27, 42]. RevIN applies a per-subsequence normalization to the input, making each series inside the network follow a similar distribution, and restores the removed statistics in the network's output. However, this process can lead to a so-called over-stationarization, i.e., a loss of information inside the network. The Non-Stationary Transformer framework [36] mitigates this by combining RevIN with a handcrafted attention mechanism that feeds the removed subsequence statistical information in the network. Despite their success in TSF, applying such solutions directly to the NILM problem is not possible. Specifically, per-subsequence scaling like RevIN is not ideal for NILM since power values are closely related to specific appliances. Similar patterns (shapes) may be common to multiple appliances, but the maximum power differentiates them. Moreover, restoring input statistics in the output does not align with NILM objectives, as disaggregation inherently involves a change in statistical values: an appliance's individual consumption is always lower than the aggregate input signal. Thus, while RevIN and similar methods are effective in TSF, they are not directly applicable to NILM.

[Data Drift Consideration in NILM] Mitigating the data drift when solving the NILM problem has been studied in the past. In [23], the authors propose a framework to mitigate data drift in high-frequency measurements of electricity consumption. Their method uses subsequences of real, apparent, and reactive power recorded at much higher resolution than standard household smart meters, which typically measure only average real power. Moreover, their

approach is based on classification based methods that are only able to detect whether a subsequence (window of 6sec) corresponds to an appliance's usage (more akin to signature recognition) and cannot be used to predict and estimate appliance-specific power consumption. Meanwhile, Chang et al. [5] present a tree-based method to adapt a trained model from one domain (eg, region or household) to another—a transfer learning perspective. Thus, none of these solutions tackle the intrinsic distribution drift observed in subsequences of standard smart meter data during the training phase.

2.3 Time Series Extrinsic Regression (TSER)

TSER solves a regression problem by learning the relationship between a time series and a continuous scalar variable, a task closely related to Time Series classification (TSC) [56]. A recent study [56] compared the current SotA approaches for TSC applied to TSER, and showed that CNN-based neural networks (ConvNet [44], ResNet [25], InceptionTime [19]), random convolutional based (Rocket [9]) and XGBoost [6] are the current best approaches for this task.

Note that predicting the total individual appliance power consumed by an appliance over time (e.g., per day, week, or month) is a straightforward application case of TSER.

3 PROBLEM DEFINITION

A smart meter signal is a univariate time series $x = (x_1, ..., x_T)$ of T timestamped power consumption readings. The meter reading is defined as the time difference $\Delta_t = t_i - t_{i-1}$ between two consecutive timestamps t_i. Each element x_t, typically measured in Watts or Watt-hours, represents either the actual power at time t or the average power over the interval Δ_t.

[Energy Disaggregation] The aggregate power consumption is defined as the sum of the N individual appliance power signal $a_1(t), a_2(t), \ldots, a_N(t)$ that run simultaneously plus some noise $\epsilon(t)$, accounting for measurement errors. Formally, it is defined as:

$$x(t) = \sum_{i=0}^{N} a_i(t) + \epsilon(t) \tag{1}$$

where $x(t)$ is the total power consumption measured by the main meter at timestep t; N is the total number of appliances connected to the smart meter; and $\epsilon(t)$ is defined as the noise or the measurement error at timestep t. The NILM challenge relies on accurately decomposing $x(t)$ to retrieve the $a_i(t)$ components.

4 PROPOSED APPROACH

The Transformer architecture has demonstrated good performance when applied to energy disaggregation [1, 57, 61], but current approaches do not consider the non-stationary aspect of real-world smart meter data. We propose NILMFormer as a sequence-to-sequence Transformer-based architecture designed to handle this phenomenon. NILMFormer operates by first stationarizing the input subsequence by subtracting its mean and standard deviation. While the normalized subsequence is passed through a robust convolutional block that serves as a feature extractor, the removed statistics are linearly projected in a vector (referred to as *TokenStats*), and the proposed TimeRPE module uses the timestamps to compute a positional encoding matrix. These features are concatenated and fed
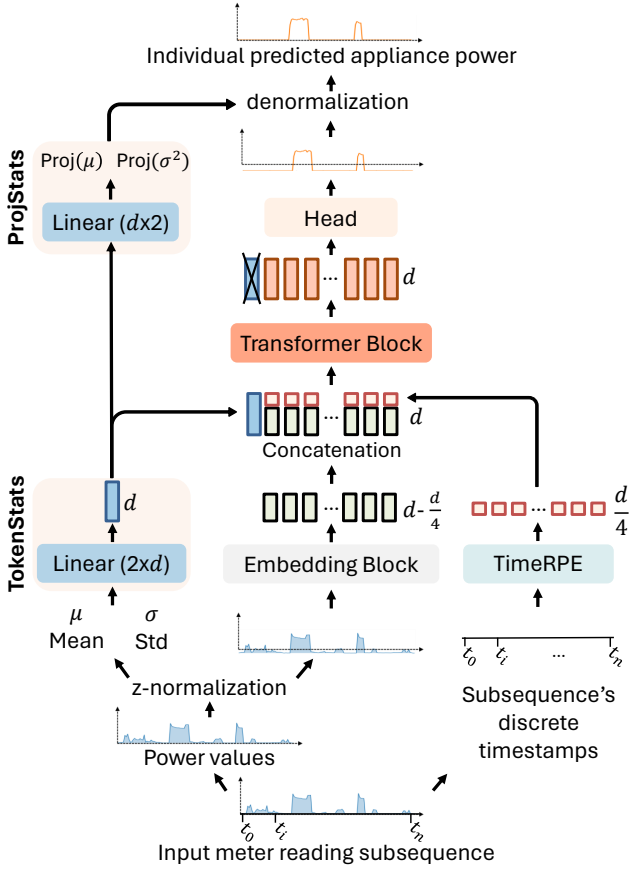
Figure 2: Overview of the NILMFormer architecture.

into the Transformer block, followed by a simple Head to obtain a 1D sequence of values. The final step consists of linearly projecting back the *TokenStats* (referred to as *ProjStats*) to 2 scalar values that are then used to denormalize the output, providing the final individual appliance consumption.

Overall, NILMFormer first splits and encodes separately the shape, the temporal information, and the intrinsic statistics of the subsequences, which are then mixed back together in the Transformer block. In addition, the output prediction is refined through the linear transformation of the input series statistics, accounting for the loss of power when disaggregating the signal.

4.1 NILMFormer Architecture

As depicted in Figure 2, NILMFormer results in an encoder that takes as input a subsequence of an entire household smart meter reading $x = (x_1, ..., x_n)$ and outputs the individual consumption $a = (a_1, ..., a_n)$ for an appliance. The workflow unfolds in the following steps.

Step 1: Input subsequence stationarization. The input power subsequence $x^{1 \times n}$ is first z-normalized. More specifically, we first extract the mean and standard deviation of the sequence as $\mu = \frac{1}{n} \sum_{i=0}^{n} x_i$ and $\sigma = \sqrt{\frac{1}{n} \sum_{i=0}^{n} (x_i - \mu)^2}$, respectively. Then, we remove the extracted mean to each value x_i and divide them by the

standard deviation as $\tilde{x} = \frac{x - \mu}{\sigma + \epsilon}$, such that the mean of the subsequences become 0 and the standard deviation 1.

Step 2: Tokenization. The mean μ and standard deviation σ values are projected using a learnable linear layer in a d-dimensional vector, referred to as *TokenStats*. In addition, the z-normalized subsequence $\tilde{x}^{1 \times n}$ is passed through the embedding block, used to extract local features. This block is composed of several convolutional layers using $d - \frac{d}{4}$ filters (see Section 4.1.1 for details), which output a features map $z^{d - \frac{d}{4} \times n}$. In parallel, the positional encoding matrix $PE^{\frac{d}{4} \times n}$ is computed according to the subsequence's discrete timestamp information using the TimeRPE module (detailed in Section 4.1.3).

Step 3: Features mix. The *TokenStats* and the positional encoding matrix are concatenated to the extracted subsequence's features maps, resulting in a new features map $\hat{z}^{(n+1) \times d}$. More specifically, the positional encoding matrix is concatenated along the inner dimension d, and the *TokenStats* is concatenated along the time dimension n (it can be viewed as adding a new token). Then, the obtained matrix $\hat{z}$ is passed through the Transformer Block (see Section 4.1.2 for details) that is used to mix the different information and learn long-range dependencies. Then, the first token of the output representation $\hat{z}^{(n+1) \times d}$, corresponding to the *TokenStats*, is removed to obtain a feature map matching the input subsequence length, $z^{n \times d}$. Finally, $z^{n \times d}$ is processed through the output Head, consisting of a 1D convolutional layer, which maps the latent representation back to a 1D series representation, $\tilde{a}^{1 \times n}$.

Step 4: Output de-stationarization. The *TokenStats* is projected back using a learnable linear layer ($d \times 2$) that provides two scalar values $Proj(\mu)$ and $Proj(\sigma)$, referred to as *ProjStats*. These values are used as a new mean and standard deviation to denormalize the output $\tilde{a}^{1 \times n}$ and obtain the final prediction as: $a = \tilde{a} * Proj(\sigma) + Proj(\mu)$. The subsequence a is the individual appliance power.

4.1.1 Embedding Block. As depicted in Figure 3(b), the Embedding Block results in 4 stacked convolutional Residual Units (ResUnit), each one composed of a convolutional layer, a GeLU activation function [26], and a BatchNormalization layer [28]. Not that a residual connection is used between each ResUnit, and a stride parameter of 1 is employed in each convolutional filter to keep the time dimension unchanged. For each Residual Unit $i = 1, ..., 4$, a dilation parameter $d = 2^i$ that exponentially increases according to the ResUnit's depth is employed. We motivate the choice of using this feature extractor in Section 5.4.

4.1.2 Transformer Block. The core of NILMFormer relies on a block composed of N stacked Transformer layers. Each Transformer layer is made of the following elements (cf. Figure 3(a)): a normalization layer, a Multi-Head Diagonally Masked Self-Attention mechanism (Multi-Head DMSA), a second normalization layer, and a Positional Feed-Forward Network [58] (PFFN). We use a Multi-Head DMSA instead of the original Attention Mechanism, as it is more effective when applied to electricity consumption series analysis [50]. In addition, we introduce residual connections after the Multi-Head DMSA and the PFFN, and the use of a Dropout parameter.

4.1.3 Timestamp Related Positional Encoding (TimeRPE). The Transformer architecture does not inherently understand sequence order
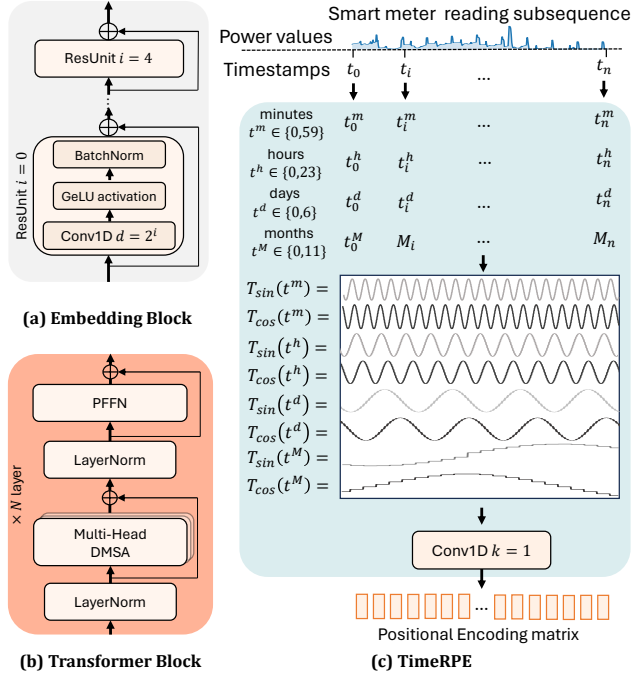
(a) Embedding Block

(b) Transformer Block

(c) TimeRPE

Figure 3: NILMFormer's architecture parts detail: (a) Embedding Block; (b) Transformer Block; (c) TimeRPE module.

Table 1: Dataset characteristics and parameters

Datasets	UKDALE	REFIT	EDF 1	EDF 2
Nb. Houses	5	20	369	24
Avg. recording time	223days	150days	3.7years	1year
Sampling rate	1min	1min	10min	30min
Max. power limit (τ_{max})	6000	10000	24000	13000
Max. ffill	3min	3min	30min	1h30
Appliances	Dishwasher Washing Machine Microwave Kettle Fridge	Dishwasher Washing Machine Microwave Kettle	Central Heater Heatpump Water Heater WhiteUsages	Electric Vehicle

$\cos\left(\frac{2\pi t_i^j}{p^j}\right)$, with $j \in \{m, h, d, M\}$ and $\{p^m = 59, p^h = 23, p^d = 6, p^M = 11\}$ corresponding to the set of max possible discrete timestamp variable. Afterward, the obtained representation is projected in a higher-dimensional space using a 1D convolution layer with a kernel of size 1. We evaluate the effectiveness of TimeRPE against various standard PE used in time series Transformers in Section 5.4.

5 EXPERIMENTAL EVALUATION

All experiments are performed on a cluster with 2 Intel Xeon Platinum 8260 CPUs, 384GB RAM, and 4 NVidia Tesla V100 GPUs with 32GB RAM. The source code [46] is in Python v3.10, and the core of NILMFormer in PyTorch v2.5.1 [45].

5.1 Datasets

We use 4 different datasets (see Table 1) that provide the total power consumed by the house, recorded by a smart meter, and the individual load power measurement for a set of appliances.

5.1.1 Public Datasets. UKDALE [31] and REFIT [21] are two well-known datasets used in many research papers to assess the performance of NILM approaches [30, 54, 61, 63, 66]. The two datasets contain high-frequency sampled data collected from small groups of houses in the UK and focus on small appliances.

5.1.2 EDF Datasets. We also use 2 private (EDF) datasets that include modern appliances not included in the public datasets, such as Heater systems, Water Heaters and Electric Vehicle chargers.
[EDF Dataset 1] It contains data from 358 houses in France between 2010 and 2014. Houses were recorded for an average of 1357 days (shortest: 501 days; longest: 1504 days). Houses were monitored with smart meters that recorded the aggregate main power, as well as the consumption of individual appliances at 10min intervals. Diverse systems, including the central heating system, the heat pump, and the water heater, have been monitored directly through the electrical panel. In addition, other appliances, such as the dishwasher, washing machine, and dryer, were monitored using meter sockets. The signals collected from these appliances have been grouped in one channel called "White Appliances".
[EDF Dataset 2] It contains data from 24 houses in France from July 2022 to February 2024. Data were recorded for an average of 397 days (shortest: 175 days; longest: 587 days). Houses were monitored with individual smart meters that recorded the main aggregate power of the house and a clamp meter that recorded the

due to its self-attention mechanisms, which are permutation invariant. Therefore, Positional Encoding (PE) is mandatory to provide this context, allowing the model to consider the position of each token in a sequence [58]. Fixed sinusoidal or fully learnable PEs are commonly used in most current Transformer-based architectures for time series analysis [42], including those proposed for energy disaggregation [7, 61]. This kind of PE consists of adding a matrix of fixed or learnable weight on the extracted features before the Transformer block. However, these PEs only help the model understand local context information (i.e., the given order of the tokens in the sequence) and do not provide any information about the global context when operating on subsequences of a longer series. In the context of NILM, appliance use is often related to specific periods (e.g., dishwashers running after mealtimes, electric vehicles charging at night, or on weekends). Moreover, detailed timestamp information is always available in real-world NILM applications. Thus, using a PE based on timestamp information can help the model better understand the recurrent use of appliances. Timestamp-based PEs have been briefly investigated for time series forecasting [65] but were always combined with a fixed or learnable PE and directly added to the extracted features.

Therefore, we proposed the Timestamps Related Positional Encoding (TimeRPE), a Positional Encoding based only on the discrete timestamp values extracted from the input subsequences. The TimeRPE module, depicted in Figure 3 (c), takes as input the timestamps information t from the input subsequences, decomposes it such as minutes t^m, hours t^h, days t^d, and months t^M, and project them in a sinusoidal basis, as: $T_{\sin}(t_i) = \sin\left(\frac{2\pi t_i^j}{p^j}\right)$ and $T_{\cos}(t_i) =$

Table 2: Overall results for the different disaggregation cases, datasets, and subsequences window length. Each reported result is the average score obtained for 3 runs. The best score is shown in bold, and the second best is underlined.

Model / Dataset and case		Win	NILMFormer		BERT4NILM		STNILM		BiGRU		Energformer		FCN		UNet NILM		TSILNet		BiLSTM		DiffNILM		DAResNet	
			MAE	MR	MAE	MR	MAE	MR	MAE	MR	MAE	MR	MAE	MR	MAE	MR	MAE	MR	MAE	MR	MAE	MR	MAE	MR
UKDALE	Dishwasher	128	**22.9**	**0.534**	32.2	0.27	33.3	0.257	<u>30.8</u>	<u>0.326</u>	41.4	0.096	39.8	0.171	36.4	0.28	38.7	0.267	36.0	0.319	59.2	0.018	48.0	0.293
		256	**16.7**	**0.626**	34.4	0.239	31.6	0.287	<u>30.9</u>	<u>0.359</u>	40.4	0.099	43.2	0.145	40.6	0.268	48.6	0.203	44.3	0.273	87.6	0.022	66.0	0.209
		512	**24.1**	**0.442**	34.4	0.265	31.2	0.314	<u>27.2</u>	<u>0.408</u>	45.2	0.097	45.3	0.171	47.8	0.205	57.4	0.255			95.3	0.036	126.5	0.085
	Fridge	128	31.6	0.364	25.6	0.473	36.9	0.364	36.7	0.363	26.5	0.471	25.8	0.493	**23.3**	**0.536**	<u>24.6</u>	0.528	<u>24.6</u>	<u>0.533</u>	28.8	0.494	92.9	0.266
		256	34.4	0.291	<u>27.0</u>	0.447	36.9	0.364	36.8	0.365	32.3	0.384	27.8	<u>0.458</u>	**26.0**	**0.485**	33.3	0.412	36.9	0.366	72.6	0.256	33.8	0.444
		512	33.3	0.315	<u>26.6</u>	<u>0.452</u>	36.9	0.364	36.7	0.364	33.3	0.427	<u>28.0</u>	**0.478**	29.8	0.425	37.0	0.365	37.0	0.366	83.9	0.225	122.0	0.159
	Kettle	128	8.7	**0.705**	11.0	0.642	<u>9.8</u>	0.665	<u>9.8</u>	<u>0.67</u>	13.5	0.595	17.4	0.515	16.2	0.535	18.9	0.496	19.0	0.512	15.3	0.573	94.4	0.265
		256	11.0	0.635	12.6	0.608	**9.8**	**0.664**	<u>10.0</u>	<u>0.664</u>	10.1	0.663	22.0	0.438	25.8	0.406	23.9	0.437	32.4	0.364	63.1	0.046	38.8	0.278
		512	**9.8**	**0.666**	13.6	0.583	<u>10.8</u>	<u>0.632</u>	11.6	0.618	12.3	0.602	29.2	0.353	43.7	0.223	34.6	0.314	60.4	0.126	88.3	0.037	113.8	0.068
	Microwave	128	10.1	0.144	**8.7**	**0.22**	<u>8.9</u>	0.185	12.8	0.088	9.0	<u>0.201</u>	14.6	0.116	13.8	0.062	15.9	0.051	15.6	0.045	16.7	0.011	23.5	0.044
		256	<u>10.1</u>	0.13	**9.1**	**0.17**	11.2	0.099	14.3	0.043	11.0	<u>0.149</u>	15.2	0.075	14.3	0.04	16.3	0.032	16.1	0.028	37.3	0.01	29.1	0.036
		512	**8.3**	**0.187**	<u>10.4</u>	0.111	10.9	0.066	12.4	0.04	10.5	<u>0.129</u>	17.0	0.027	14.7	0.018	16.0	0.022	15.9	0.012	56.0	0.009	80.6	0.013
	Washer	128	**12.4**	**0.247**	<u>16.4</u>	<u>0.137</u>	21.4	0.114	20.6	0.109	29.4	0.136	24.9	0.114	22.7	0.109	28.5	0.06	28.6	0.068	34.8	0.021	35.9	0.075
		256	**8.5**	**0.358**	26.3	0.084	26.6	0.081	<u>19.0</u>	0.093	32.0	<u>0.116</u>	32.9	0.089	25.4	0.093	34.5	0.051	32.9	0.055	50.1	0.014	49.2	0.042
		512	**10.8**	**0.268**	31.5	0.09	26.5	<u>0.098</u>	<u>21.1</u>	0.086	31.4	0.086	38.3	0.058	29.2	0.066	34.8	0.052	39.3	0.044	66.1	0.013	90.5	0.019
REFIT	Dishwasher	128	**29.1**	**0.332**	49.3	0.081	41.1	0.136	44.5	0.169	<u>35.8</u>	0.082	39.3	<u>0.218</u>	49.6	0.166	47.0	0.134	70.0	0.091	69.0	0.018	242.2	0.044
		256	45.5	**0.312**	<u>44.2</u>	0.145	**40.5**	<u>0.151</u>	70.4	0.069	54.9	0.084	50.2	0.151	59.2	0.093	72.6	0.046	72.7	0.057	74.3	0.02	125.7	0.049
		512	**42.7**	**0.205**	56.0	0.117	<u>46.3</u>	0.121	75.9	0.072	64.6	0.112	52.8	<u>0.134</u>	59.1	0.086	72.6	0.046	90.2	0.037	83.1	0.022	231.9	0.042
	Kettle	128	<u>9.3</u>	**0.522**	16.9	0.437	**8.8**	**0.529**	9.4	0.508	12.4	0.426	15.5	0.387	15.7	0.368	17.6	0.348	18.0	0.35	33.8	0.009	53.4	0.121
		256	12.5	0.454	**9.2**	**0.504**	<u>9.8</u>	<u>0.503</u>	10.0	0.496	16.0	0.359	18.2	0.332	19.1	0.31	18.9	0.312	25.4	0.249	34.1	0.141	54.9	0.123
		512	**8.6**	**0.526**	<u>10.6</u>	<u>0.462</u>	11.0	0.439	19.3	0.329	17.7	0.306	22.5	0.259	28.3	0.205	26.2	0.233	42.5	0.054	43.2	0.014	119.9	0.051
	Microwave	128	**5.7**	**0.11**	10.7	<u>0.074</u>	11.9	0.04	<u>9.1</u>	0.04	10.0	0.046	12.9	0.05	12.9	0.043	10.1	0.042	10.2	0.023	22.2	0.006	43.4	0.014
		256	9.7	<u>0.05</u>	10.7	**0.056**	**9.7**	0.029	10.0	0.023	10.2	0.032	10.3	0.028	13.8	0.023	9.8	0.019	11.1	0.017	21.7	0.007	77.6	0.009
		512	**6.4**	**0.082**	10.7	<u>0.029</u>	<u>8.0</u>	0.024	9.7	0.015	9.9	0.027	11.4	0.015	11.3	0.013	9.5	0.014	9.2	0.007	31.0	0.006	54.0	0.007
	Washer	128	**18.9**	**0.25**	33.6	<u>0.172</u>	33.0	0.126	<u>30.6</u>	0.115	36.9	0.115	35.5	0.112	42.7	0.1	35.8	0.109	40.9	0.097	48.4	0.023	101.4	0.048
		256	22.3	**0.172**	34.8	0.109	30.9	<u>0.125</u>	34.0	0.1	<u>30.7</u>	0.109	42.0	0.098	42.4	0.07	42.8	0.077	40.0	0.076	57.7	0.026	166.5	0.029
		512	**20.6**	**0.168**	31.0	0.09	<u>29.0</u>	<u>0.11</u>	37.4	0.075	32.8	0.097	40.1	0.094	35.7	0.086	40.5	0.069	43.0	0.063	50.3	0.027	188.4	0.029
EDF1	Heater	128	**283.4**	**0.52**	312.1	<u>0.457</u>	<u>310.3</u>	0.44	342.7	0.435	354.0	0.445	328.0	0.428	352.8	0.437	374.6	0.416	345.0	0.425	351.6	0.434	529.7	0.305
		256	**263.6**	**0.525**	297.2	<u>0.458</u>	<u>290.8</u>	0.456	323.0	0.425	371.9	0.414	316.6	0.437	359.0	0.402	337.7	0.405	330.5	0.415	333.5	0.442	533.6	0.296
		512	**255.8**	**0.521**	283.3	<u>0.456</u>	<u>275.3</u>	<u>0.456</u>	321.1	0.411	368.7	0.386	307.9	0.424	323.1	0.403	314.6	0.396	330.0	0.395	389.2	0.336	497.2	0.26
	Heatpump	128	**270.5**	**0.548**	288.5	0.504	283.3	0.523	282.0	0.526	<u>274.9</u>	<u>0.53</u>	328.1	0.475	362.9	0.441	362.6	0.462	337.4	0.446	344.0	0.452	507.4	0.356
		256	**276.2**	**0.545**	292.0	0.504	282.2	0.501	288.1	0.507	<u>280.4</u>	<u>0.51</u>	353.1	0.43	364.0	0.422	345.6	0.451	370.9	0.412	304.9	0.486	542.6	0.272
		512	**258.1**	**0.533**	295.9	0.475	<u>290.3</u>	<u>0.489</u>	303.0	0.457	299.2	0.48	392.7	0.374	349.9	0.394	363.3	0.388	377.5	0.375	362.9	0.397	579.7	0.272
	WaterHeater	128	**88.2**	**0.686**	113.1	0.613	134.9	0.566	<u>112.5</u>	<u>0.619</u>	135.6	0.551	185.2	0.46	224.9	0.364	169.8	0.494	178.9	0.478	131.0	0.55	268.8	0.361
		256	**91.8**	**0.676**	126.3	0.591	136.4	0.564	114.2	0.614	146.5	0.521	207.6	0.428	238.9	0.337	203.6	0.443	223.7	0.4	<u>104.8</u>	<u>0.639</u>	342.2	0.255
		512	**92.4**	**0.668**	111.2	<u>0.615</u>	138.1	0.55	118.5	0.601	181.0	0.464	223.7	0.392	236.9	0.372	274.8	0.319			126.7	0.587	356.0	0.256
	White Appl.	128	**90.8**	<u>0.209</u>	105.2	0.203	111.6	0.171	<u>99.3</u>	**0.209**	109.8	0.179	120.9	0.158	116.9	0.139	127.5	0.123	128.3	0.145	134.1	0.083	164.5	0.105
		256	**86.1**	**0.231**	104.2	0.189	<u>99.3</u>	<u>0.191</u>	106.0	0.176	114.8	0.133	130.4	0.143	119.8	0.144	131.3	0.107	131.1	0.119	126.5	0.069	206.8	0.098
		512	**79.5**	**0.215**	94.2	<u>0.178</u>	<u>89.7</u>	<u>0.178</u>	102.8	0.128	107.9	0.123	114.8	0.103	115.2	0.099	116.8	0.102	124.0	0.088	203.2	0.06	192.2	0.071
EDF2	EV	128	**102.5**	**0.617**	169.2	0.472	<u>109.9</u>	<u>0.57</u>	210.4	0.402	134.8	0.507	189.5	0.428	245.7	0.384	242.5	0.372	238.4	0.371	286.3	0.322	434.5	0.233
		256	**111.8**	**0.582**	168.6	0.456	<u>119.1</u>	<u>0.553</u>	227.8	0.36	180.1	0.396	232.2	0.381	262.5	0.31	299.4	0.303	301.2	0.313	213.8	0.404	932.8	0.097
		512	**114.3**	**0.616**	152.5	0.544	<u>130.6</u>	<u>0.597</u>	327.1	0.304	147.3	0.556	323.2	0.326	398.4	0.205	357.7	0.206	481.4	0.122	295.3	0.354	1897.1	0.046
Avg. Score Per Metrics			**70.7**	**0.4**	84.5	<u>0.328</u>	<u>81.2</u>	0.3263	94.5	0.3042	93.2	0.2914	107.8	0.2611	116.0	0.2412	114.5	0.2373	122.5	0.2216	122.2	0.1838	261.3	0.1462
Avg. Rank Per Metrics			**1.762**	**1.905**	3.667	<u>3.476</u>	<u>3.31</u>	3.762	4.262	4.667	4.881	4.714	6.286	5.714	6.857	6.81	7.262	7.69	8.143	8.143	8.833	9.238	10.738	9.881
Avg. Rank			**1.833**		<u>3.536</u>		3.571		4.464		4.798		6.0		6.833		7.476		8.143		9.036		10.31	

power consumed by an electric vehicle's charger. Aggregate main power and electric vehicle recharge were sampled 30min intervals.

5.1.3 Data processing. According to the parameters reported in Table 1, we resampled and readjusted recorded values to round timestamps by averaging the power consumed during the interval Δ_t, we forward-filled the missing values, and we clipped the consumption values between 0 and a maximum power threshold.

In this study, we evaluate the model's performance based on real-world scenarios using unseen data from different houses within the same dataset [52]. Distinct houses were used for training and evaluation to ensure robust performance assessment. For the UKDALE dataset, we utilized houses 1, 3, 4, and 5 for training and house 2 for testing. This selection was made because only houses 1 and 2 contained all the appliances. Note that for UKDALE, 80% of the data from house 1 was used for training, while 20% was used as a validation set to prevent overfitting.

For the REFIT and EDF1 datasets, which contain more houses, 2 houses were reserved for testing, 1 house for validation, and the remaining houses were used for training. For EDF2, which contains more than 350 houses, 70% of the houses were used for training, 10% for validation, and the remaining 20% for evaluation.

For training and evaluating the model, the entire recorded consumption data of the different houses was sliced into subsequences using a non-overlapping window of length w. Subsequences containing any remaining missing values were discarded. To ensure training stability, we scaled the data to a range between 0 and 1 by dividing the consumption values (both aggregate and individual appliance power) by the maximum power threshold τ_{max} for each dataset as reported in Table 1. Consequently, before evaluating the models, we denormalized the data by multiplying it by τ_{max}.

5.2 Evaluation Pipeline

We compare our solution against several SotA sequence-to-sequence NILM solutions. We include two recurrent-based architectures, BiLSTM [30] and BiGRU [51] that combine Convolution and Recurrent layers; three convolutional-based baselines, FCN [63], UNet-NILM [53] and DAResNet [60]; as well as a recent diffusion-based proposed approach, DiffNILM [54]. In addition, we include 4 Tranformer-based baselines, BERT4NILM [61], Energformer [1], STNILM [57], nd TSILNet [66], which integrates both Transformer and recurrent layers. We used the default parameters provided by the authors and trained all the models using the Mean Squared Error loss.

5.2.1 Window length sensitivity. We evaluate each baseline using different subsequences windows length w to assess the sensitivity of the results on this parameter. We experimented with windows length $w = \{128, 256, 512\}$ to standardize results across datasets. We also considered using window lengths corresponding to specific periods based on the sampling rate but observed no significant changes in the results.
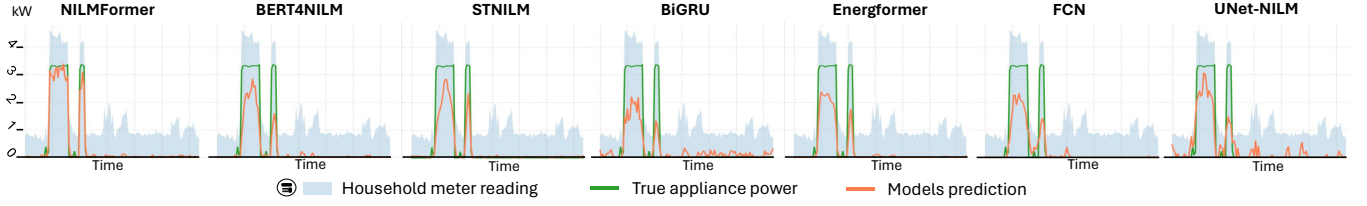
Figure 4: Qualitative disaggregation results for Electric Vehicle (EDF1) on a sample example for the 7 best baselines. (w = 128).

5.2.2 Evaluation metrics. We assess the energy disaggregation quality performance of the different baselines using 2 metrics. For each metric, T represents the total number of intervals while y_t is the true and $\hat{y}_t$ is the predicted power usage of an appliance. The first metric is the standard Mean Absolute Error (MAE): $MAE = \frac{1}{T} \sum_{t=1}^{T} |\hat{y}_t - y_t|$. The second metric is the Matching Ratio (MR), based on the overlapping rate of true and estimated prediction, and stated to be the best overall indicator performance [39]:

$$MR = \frac{\sum_{t=1}^{N} min(\hat{y}_t, y_t)}{\sum_{t=1}^{N} max(\hat{y}_t, y_t)}.$$

5.3 Results

Table 2 lists the results for the 4 datasets and each different appliance energy disaggregation case and subsequence window length. First, we note that NILMFormer outperforms other solutions overall for the 2 metrics (avg. rank per metric and avg. total Rank). More specifically, we note an improvement of over 15% in terms of MAE and 22% in MR (on average across all the datasets and cases) compared to the second-best model for each metric, STNILM and BERT4NILM, respectively. NILMFormer is outperformed only on the Fridge disaggregation case on the UKDALE dataset by UNet-NILM. This is because the fridge is always ON and present in all houses. As a result, the scheme adopted to address the non-stationary nature of the data caused the model to struggle when isolating the constant, baseline consumption that the fridge represents. We also provide in Figure 4 an example of the disaggregation quality of the 7 best baselines on the Electric Vehicle case (EDF1).

5.4 Impact of Design Choices on Performance

We perform a detailed evaluation to assess the performance of different key parts of NILMFormer. We first study the effectiveness of the two mechanisms proposed to mitigate the non-stationary aspect of the data (i.e., the *TokenStats* and the *ProjStats*). Then, we assess the proposed TimeRPE's performance against other PEs usually used in Transformers for time series analysis.

We utilize a Critical Difference Diagram (CD-Diagram) to compare the performance, computed based on the rank of the different variants. This method, proposed in [3], involves performing a Friedman test followed by a post-hoc Wilcoxon test on the calculated ranks, according to significance level $\alpha = 0.1$. The rank is given by computing an overall score according to the two disaggregation metrics (MAE and MR) averaged across all the datasets, cases, and window lengths, previously normalized to the same range (between 0 and 1). In addition, we note that bold lines in CD-diagrams indicate insignificant differences between the connected methods.

5.4.1 Non-stationarity consideration. We assess the influence of the two mechanisms adopted to mitigate the non-stationary aspect of the data. More precisely, we compared the following options: (**None**): not considering any mechanism to handle the non-stationary aspect of the data (as the previously proposed NILM approach); (**RevIN**): applying the RevIN [33] per subsequences normalization/denormalization scheme to the input/output without propagating any information inside the network, i.e., the approach currently adopted in most of TSF architecture [27, 42]; (**w/ TokenStats**): propagating only the *TokenStats* inside the Transformer block, i.e., the extracted input statistics μ and σ are re-used to denormalize the output, (**w/ ProjStats**): using only the learnable projection of the extracted input statistics μ and σ to denormalize the output without propagating information in the Transformer Block; and, (**w/ TokenStats and w/ ProjStats**); the final proposed mechanism include in NILMFormer. The results (cf. Figure 5(a)) demonstrate that applying only the RevIN scheme leads to worse performance than using the proposed architecture without any mitigation of the non-stationary effect and, thus, confirms that level information is crucial for NILM. In addition, omitting either *TokenStats* or *ProjStats* results in a performance drop, confirming the essential role of both.

5.4.2 Positional Encoding. We evaluate the effectiveness of the proposed TimeRPE against usual standard PE methods used in the NILM literature. More specifically, we investigate the following options: (1) removing the PE (**NoPE**); (2) concatenating or adding the TimeRPE's PE matrix to the embedding features; (3) replacing TimeRPE by (**Fixed**) the original PE proposed for Transformer in [58]); (**tAPE**) a fixed PE proposed recently, especially for time series Transformer [22]; (**Learnable**) a fully learnable one, as proposed for time series analysis in [62], and used in the NILM literature [55, 59, 61]. The critical diagram in Figure 5(b) shows the significant superiority of TimeRPE over other PE options. Moreover, we noticed that concatenating the PE information with the features extracted from the Embedding Block instead of adding it lead to significantly better performance. We assume that concatenating the PE instead of adding lead helps the model to differentiate the two pieces of information.

5.4.3 Embedding Block. We evaluate the impact of different embedding blocks for extracting features from the aggregate power signal. More specifically, we investigate replacing the proposed Dilated Residual Embedding Block by: (**Linear**) a simple linear embedding layer that maps each time step of the input sequence model; (**Patchify**) a patching embedding, an approach used by numerous Transformer for time series analysis [42, 59], that involves

(a) NILMFormer with Different Non-stationary Mechanisms

(b) NILMFormer with Different Positional Encoding

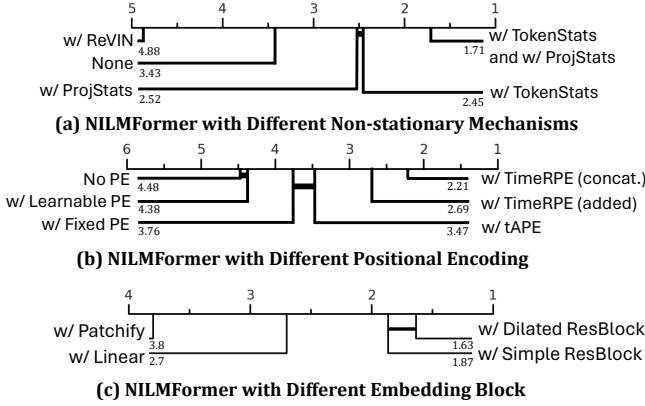**(c) NILMFormer with Different Embedding Block**

Figure 5: CD-diagram of the average rank (avg. of the MAE and MR across all datasets, cases, and window lengths) to study the impact of: (a) the proposed mechanisms to mitigate the non-stationary aspect of the data, (b) different PE, and (c) different embedding block.
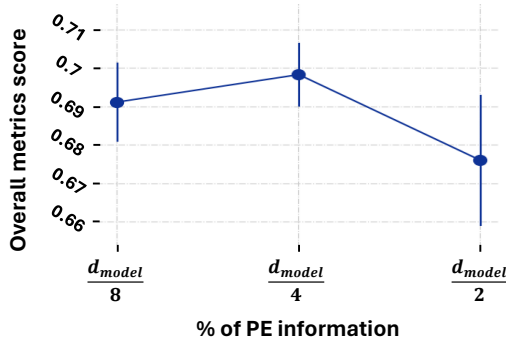

Figure 6: Overall metrics score (avg. of the MAE and MR for $w = 256$, over all the datasets and cases) by varying the ratio of positional encoded information (given by TimeRPE) according to the inner model dimension ($d_{model} = 128$).

dividing the input series in patches (i.e., subsequences); using a convolutional layer with stride and kernel of the patch length; and (**ResBlock**) a simple Residual Convolution Block without dilation (with k=3). The results (cf. Figure 5(c)) demonstrates the necessity of the convolution layers to extract localized feature patterns. Using the proposed Embedding Block leads to a slight (but not significant) increase compared to the simple ConvBlock. Note that using the patch embedding leads to the worst results, suggesting that this type of embedding does not suit sequence-to-sequence NILM solutions.

5.4.4 Impact of Positional Encoding Ratio on Model Performance.
As detailed in Section 4, NILMFormer concatenates the positional-encoding matrix produced by TimeRPE with the feature vectors generated by the Embedding Block along the inner model dimension d. To quantify how much representational budget should be allocated to positional cues, we train NILMFormer while varying the share of channels reserved for TimeRPE (using window size $w = 256$

and $d = 128$). Specifically, we examine three ratios, $\left\{\frac{d}{8}, \frac{d}{4}, \frac{d}{2}\right\}$, corresponding to dedicating 12.5%, 25%, and 50% of the channels to positional information.

The results in Figure 6 reveal that allocating one quarter of the channels to positional encoding ($\frac{d}{4}$) offers the best trade-off, delivering the highest disaggregation accuracy. Increasing the share to $\frac{d}{2}$ brings no further gains, while reducing it to $\frac{d}{8}$ only slightly affects performance.

6 DEPLOYED SOLUTION

Since 2015, EDF has offered to its clients a consumption monitoring service called *Mon Suivi Conso* [13], enabling clients to track their consumption. The service is accessible via the web and a mobile app. A new feature was released in 2018 to provide the clients with an estimation of their annual individual appliance consumption. The backbone algorithm relied on semi-supervised statistical methods that used the customers' static information [14, 15]. However, recent user feedback indicated a growing demand for more granular and personalized insights, consumption estimates down to daily frequency, and percentage-based cost breakdowns per appliance [11]. In response, EDF recently explored the use of TSER [56] to infer monthly appliance-level consumption. These approaches yielded an improvement over the original semi-supervised methods employed in *Mon Suivi Conso*, reducing the mean absolute error (MAE) of monthly misassigned consumption across various appliances (e.g., heaters, water heaters, and electric vehicles). Despite these advances, monthly-level feedback remains relatively coarse, limiting its practical value for end-users. Consequently, NILM-based algorithms, which can disaggregate consumption at a per-timestamp level, offer a promising alternative for replacing and enhancing the current individual appliance consumption feedback feature.

[NILMFormer Applied to Daily and Monthly Feedback] Employing NILMFormer for per-period feedback is straightforward, as the smart meters in France collect data at half-hour intervals. Thus, to provide the individual appliance consumption over a period of time, we adapted NILMFormer as follows:
1. The electricity consumption series of a client is sliced into subsequences using a tumbling (non-overlapping) window of size w.
2. Each subsequence is then passed to a NILMFormer instance trained to disaggregate the signal for a specific appliance a.
3. The predictions are then concatenated.
4. Finally, the appliance predicted power consumption is returned, summing over the period of interest (day, week, or month).

[Comparison to TSER Approaches] We experimentally evaluate the performance of our approach against the 5 best SotA TSER baselines reported in [56], including XGBoost [6], Rocket [9] (a random convolutional-based regressor), and 3 convolutional-based DL regressors (ConvNet, ResNet, and InceptionTime). We consider two settings: predicting the per-day and per-month appliance consumption. To train the TSER baselines, we preprocessed the datasets (EDF1 and EDF2) by slicing the entire consumption into subsequences (day or month), and by summing the total consumption of the appliance over that period to obtain the label. Note that all baselines were initialized using default parameters, and DL-based methods were trained using the Mean Squared Error Loss.
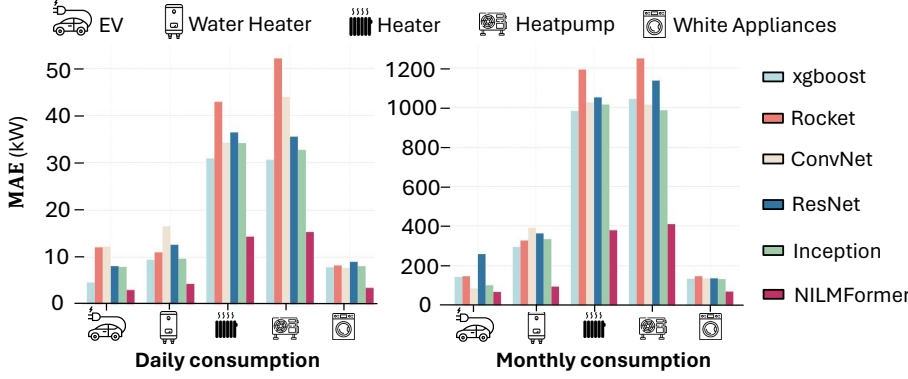
(a) Mean Average Error (MAE) in kW for different approaches applied to daily (left) and monthly (right) consumption restitution of five different appliances.
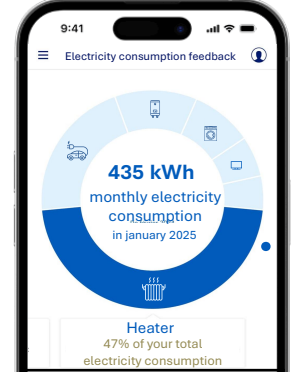
(b) User interface of EDF's Mon Suivi Conso mobile application.

Figure 7: (a) Results comparison for per-period energy estimation; (b) Example of feedback available to a client through the user interface of EDF's *Mon Suivi Conso* mobile application.

To evaluate our approach, we reused the different instances of NILMFormer trained on timestamp energy disaggregation in Section 5.3 and applied the framework described above.

Results. The results (cf. Figure 7(a)) demonstrate the superiority of NILMFormer, which significantly outperforms TSER baselines applied to daily and monthly appliance consumption prediction. On average, across the five appliances evaluated, NILMFormer achieves a 51% reduction in MAE for daily consumption compared to the second-best baseline (XGBoost), and a 151% reduction in MAE for monthly consumption compared to the second-best baseline (Inception). These findings highlight the effectiveness of our approach for deployment in *Mon Suivi Conso*, providing substantially more accurate feedback to clients than previous approaches.

6.1 Deployment Insights and Performance

DATANUMIA, an EDF subsidiary, oversees the deployment and integration of NILMFormer into *Mon Suivi Conso*, gradually replacing the existing solution for individual appliance consumption feedback. The final delivered algorithm uses a separate model for each appliance, enabling the solution to predict consumption for each appliance independently. These individual predictions are then aggregated to produce a final output showing the percentage of each appliance's consumption relative to the total (see Figure 7(b)). Users can also track their electricity usage according to different time intervals: daily, monthly, and annual periods, both in kilowatt-hours and euros (based on their contracted rates).
[Infrastructure and Performance] The deployed solution is hosted on Amazon Elastic Compute Cloud (EC2) using 6 *m6i.8xlarge* instances, each providing 32 vCPUs. The individual appliance consumption estimation runs weekly on the entire customer base that has consented to data analysis—around 3.6 million customers. Post-launch performance evaluations shows that the system currently handles 100 customer requests per second, processing the entire dataset in approximately 11 hours.
[User Engagement] During the last quarter of 2024, operational statistics indicate 7 million visits to the consumption monitoring

solution that uses our approach, with 2.2 million visits to the individual appliance consumption feedback through the web interface. For the mobile application, out of 7 million visits to this application, 6.2 million visits focused on the per-appliance consumption feature, demonstrating considerable interest in the information that our solution provides.
[Decision Making] EDF recently applied this solution to a subset of consenting consumers on La Réunion [16], focusing on electric vehicle (EV) charging habits, and the impact of different off-peak charging systems (e.g., smart plugs versus dedicated charging stations) on overall consumption. Through NILMFormer, it was possible to pinpoint exact EV charging times and calculate the associated total consumption. The results showed notable benefits for customers using controlled charging stations, offering improved insights into peak/off-peak ratios, and demonstrating the value of NILMFormer in enhancing data-driven decision-making for EDF.

7 CONCLUSIONS

We proposed NILMFormer, a DL sequence-to-sequence Transformer for energy disaggregation, designed to address the data distribution drift problem that occurs when operating on subsequences of consumption series. NILMFormer employs a stationarization/de-stationarization scheme tailored to the NILM problem and uses TimeRPE, a novel PE based only on the subsequence's timestamp information. The results show that NILMFormer significantly outperforms current SotA NILM solutions on 4 different datasets. NILMFormer outperforms the previous method for individual appliance per period feedback. Our solution has been successfully deployed as the backbone algorithm for EDF's consumption monitoring service, delivering detailed insights to millions of customers.

REFERENCES

[1] Georgios F. Angelis, Christos Timplalexis, Athanasios I. Salamanis, Stelios Krinidis, Dimosthenis Ioannidis, Dionysios Kehagias, and Dimitrios Tzovaras. 2023. Energformer: A New Transformer Model for Energy Disaggregation. *IEEE Transactions on Consumer Electronics* 69, 3 (2023), 308–320. https://doi.org/10.1109/TCE.2023.3237862

[2] K. Carrie Armel, Abhay Gupta, Gireesh Shrimali, and Adrian Albert. 2013. Is disaggregation the holy grail of energy efficiency? The case of electricity. *Energy Policy* 52 (2013), 213–234. https://doi.org/10.1016/j.enpol.2012.08.062

[3] Alessio Benavoli, Giorgio Corani, and Francesca Mangili. 2016. Should We Really Use Post-Hoc Tests Based on Mean-Ranks? *Journal of Machine Learning Research* 17, 5 (2016), 1–10. http://jmlr.org/papers/v17/benavoli16a.html

[4] Laurent Bozzi and Philippe Charpentier. 2018. Évaluation d'Impact sur la Consommation Électrique de la Solution Digitale e.quilibre d'EDF. In *Journées de Statistique (JdS)*. Société Française de Statistique (SFdS), France.

[5] Xiaomin Chang, Wei Li, Chunqiu Xia, Qiang Yang, Jin Ma, Ting Yang, and Albert Y. Zomaya. 2022. Transferable Tree-Based Ensemble Model for Non-Intrusive Load Monitoring. *IEEE Transactions on Sustainable Computing* 7, 4 (2022), 970–981. https://doi.org/10.1109/TSUSC.2022.3175941

[6] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. ACM, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[7] Xu Cheng, Meng Zhao, Jianhua Zhang, Jinghao Wang, Xueping Pan, and Xiufeng Liu. 2022. TransNILM: A Transformer-based Deep Learning Model for Non-intrusive Load Monitoring. In *Proceedings of the 2022 International Conference on High Performance Big Data and Intelligent Systems (HDIS)*. 13–20. https://doi.org/10.1109/HDIS56859.2022.9991439

[8] Stanislav Chren, Bruno Rossi, and Tomáš Pitner. 2016. Smart grids deployments within EU projects: The role of smart meters. In *2016 Smart Cities Symposium Prague (SCSP)*. 1–5. https://doi.org/10.1109/SCSP.2016.7501033

[9] Angus Dempster, François Petitjean, and Geoffrey I. Webb. 2019. ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. *CoRR* abs/1910.13051 (2019). arXiv:1910.13051 http://arxiv.org/abs/1910.13051

[10] DMLC. 2024. eXtreme Gradient Boosting. https://github.com/dmlc/xgboost

[11] EDF. 2024. Private communication with DATANUMIA Team Manager.

[12] EDF. 2025. The digital revolution driving energy efficiency. https://www.edf.fr/en/the-edf-group/taking-action-as-a-responsible-company/corporate-social-responsibility/the-digital-revolution-driving-energy-efficiency

[13] EDF. 2025. Solution suivi conso EDF. https://particulier.edf.fr/fr/accueil/bilan-consommation/solution-suivi-conso.html

[14] Gregory Yard EDF, Laurent Bozzi. French Patent FR1451531, 2014. ESTIMATION DE LA CONSOMMATION ELECTRIQUE D'UN EQUIPEMENT DONNE PARMI UN ENSEMBLE D'EQUIPEMENTS ELECTRIQUES. https://data.inpi.fr/brevets/FR1451531

[15] Melanie Cazes EDF, Laurent Bozzi. French Patent FR3017975, 2016. ESTIMATION FINE DE CONSOMMATION ELECTRIQUE POUR DES BESOINS DE CHAUFFAGE/CLIMATISATION D'UN LOCAL D'HABITATION. https://data.inpi.fr/brevets/FR3017975

[16] EDF à la Réunion. 2024. Sarz la Kaz. https://reunion.edf.fr/edf-a-la-reunion/actualites-a-la-reunion/sarz-la-kaz. Accessed: 2025-02-07.

[17] Anthony Faustine and Lucas Pereira. 2020. Multi-Label Learning for Appliance Recognition in NILM Using Fryze-Current Decomposition and Convolutional Neural Network. *Energies* 13, 16 (2020). https://doi.org/10.3390/en13164154

[18] Anthony Faustine, Lucas Pereira, Hafsa Bousbiat, and Shridhar Kulkarni. 2020. UNet-NILM: A Deep Neural Network for Multi-tasks Appliances State Detection and Power Estimation in NILM. In *Proceedings of the 5th International Workshop on Non-Intrusive Load Monitoring* (Virtual Event, Japan) *(NILM'20)*. Association for Computing Machinery, New York, NY, USA, 84–88. https://doi.org/10.1145/3427771.3427859

[19] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F. Schmidt, Jonathan Weber, Geoffrey I. Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. 2020. InceptionTime: Finding AlexNet for time series classification. *Data Mining and Knowledge Discovery* 34, 6 (sep 2020), 1936–1962. https://doi.org/10.1007/s10618-020-00710-y

[20] William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.* 23, 1, Article 120 (Jan. 2022), 39 pages.

[21] Steven Firth, Tom Kane, Vanda Dimitriou, Tarek Hassan, Farid Fouchal, Michael Coleman, and Lynda Webb. 2017. REFIT Smart Home dataset. (6 2017). https://doi.org/10.17028/rd.lboro.2070091.v1

[22] Navid Mohammadi Foumani, Chang Wei Tan, Geoffrey I Webb, and Mahsa Salehi. 2024. Improving position encoding of transformers for multivariate time series classification. *Data Mining and Knowledge Discovery* 38, 1 (2024), 22–48.

[23] Daisy H. Green, Aaron W. Langham, Rebecca A. Agustin, Devin W. Quinn, and Steven B. Leeb. 2023. Adaptation for Automated Drift Detection in Electromechanical Machine Monitoring. *IEEE Transactions on Neural Networks and Learning Systems* 34, 10 (2023), 6768–6782. https://doi.org/10.1109/TNNLS.2022.3184011

[24] G.W. Hart. 1992. Nonintrusive appliance load monitoring. *Proc. IEEE* 80, 12 (1992), 1870–1891. https://doi.org/10.1109/5.192069

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. https://doi.org/10.48550/ARXIV.1512.03385

[26] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian Error Linear Units (GELUs). https://doi.org/10.48550/ARXIV.1606.08415

[27] Romain Ilbert, Ambroise Odonnat, Vasilii Feofanov, Aladin Virmaux, Giuseppe Paolo, Themis Palpanas, and Ievgen Redko. 2024. SAMformer: Unlocking the Potential of Transformers in Time Series Forecasting with Sharpness-Aware Minimization and Channel-Wise Attention. arXiv:2402.10198 [cs.LG]

[28] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. https://doi.org/10.48550/ARXIV.1502.03167

[29] Maria Kaselimi, Eftychios Protopapadakis, Athanasios Voulodimos, Nikolaos Doulamis, and Anastasios Doulamis. 2022. Towards Trustworthy Energy Disaggregation: A Review of Challenges, Methods, and Perspectives for Non-Intrusive Load Monitoring. *Sensors* 22 (08 2022), 5872. https://doi.org/10.3390/s22155872

[30] Jack Kelly and William Knottenbelt. 2015. Neural NILM. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*. ACM. https://doi.org/10.1145/2821650.2821672

[31] Jack Kelly and William Knottenbelt. 2015. The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes. *Scientific Data* 2 (03 2015). https://doi.org/10.1038/sdata.2015.7

[32] Hyungsul Kim, Manish Marwah, Martin F. Arlitt, Geoff Lyon, and Jiawei Han. 2011. Unsupervised Disaggregation of Low Frequency Power Measurements. In *SDM*. https://api.semanticscholar.org/CorpusID:18447017

[33] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. 2021. Reversible Instance Normalization for Accurate Time-Series Forecasting against Distribution Shift. In *International Conference on Learning Representations*. https://openreview.net/forum?id=cGDAkQo1C0p

[34] J. Zico Kolter. 2011. REDD : A Public Data Set for Energy Disaggregation Research.

[35] Pauline Laviron, Xueqi Dai, Bérénice Huquet, and Themis Palpanas. 2021. Electricity Demand Activation Extraction: From Known to Unknown Signatures, Using Similarity Search. In *e-Energy '21: The Twelfth ACM International Conference on Future Energy Systems, Virtual Event, Torino, Italy, 28 June - 2 July, 2021*, Herman de Meer and Michela Meo (Eds.). ACM, 148–159. https://doi.org/10.1145/3447555.3464865

[36] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2022. Non-stationary Transformers: Exploring the Stationarity in Time Series Forecasting. In *Neural Information Processing Systems*. https://api.semanticscholar.org/CorpusID:252968420

[37] Prajowal Manandhar, Hasan Rafiq, Edwin Rodriguez-Ubinas, and Themis Palpanas. 2024. New Forecasting Metrics Evaluated in Prophet, Random Forest, and Long Short-Term Memory Models for Load Forecasting. *Energies* 17, 23 (2024). https://doi.org/10.3390/en17236131

[38] Luca Massidda, Marino Marrocu, and Simone Manca. 2020. Non-Intrusive Load Disaggregation by Convolutional Neural Network and Multilabel Classification. *Applied Sciences* 10, 4 (2020). https://doi.org/10.3390/app10041454

[39] Ebony Mayhorn, Greg Sullivan, Joseph M. Petersen, Ryan Butner, and Erica M. Johnson. 2016. Load Disaggregation Technologies: Real World and Laboratory Performance. https://api.semanticscholar.org/CorpusID:115779193

[40] Matthew Middlehurst, Ali Ismail-Fawaz, Antoine Guillaume, Christopher Holder, David Guijo Rubio, Guzal Bulatova, Leonidas Tsaprounis, Lukasz Mentel, Martin Walter, Patrick Schäfer, and Anthony Bagnall. 2024. aeon: a Python toolkit for learning from time series. arXiv:2406.14231 [cs.LG] https://arxiv.org/abs/2406.14231

[41] Megan Milam and G. Kumar Venayagamoorthy. 2014. Smart meter deployment: US initiatives. In *ISGT 2014*. 1–5. https://doi.org/10.1109/ISGT.2014.6816507

[42] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *International Conference on Learning Representations*.

[43] Eduardo Ogasawara, Leonardo C. Martinez, Daniel de Oliveira, Geraldo Zimbrão, Gisele Lobo Pappa, and Marta Mattoso. 2010. Adaptive Normalization: A novel data normalization approach for non-stationary time series. *The 2010 International Joint Conference on Neural Networks (IJCNN)* (2010), 1–8. https://api.semanticscholar.org/CorpusID:5757527

[44] Keiron O'Shea and Ryan Nash. 2015. An Introduction to Convolutional Neural Networks. *CoRR* abs/1511.08458 (2015). arXiv:1511.08458 http://arxiv.org/abs/1511.08458

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., Red Hook, NY, USA.
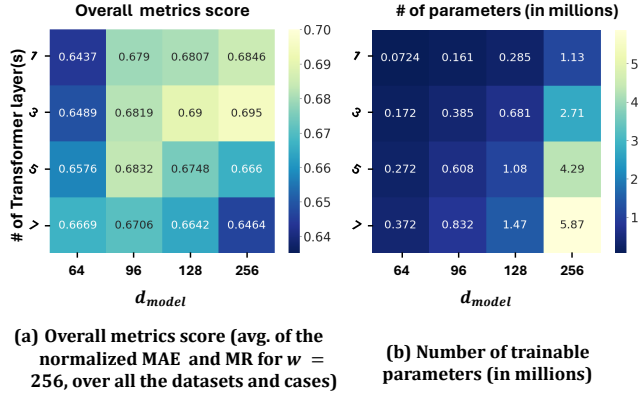
[46] Adrien Petralia. 2025. Source code of NILMFormer experiments. https://github.com/adrienpetralia/NILMFormer

[47] Adrien Petralia, Paul Boniol, Philippe Charpentier, and Themis Palpanas. 2025. DeviceScope: An Interactive App to Detect and Localize Appliance Patterns in Electricity Consumption Time Series . In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. 4552–4555. https://doi.org/10.1109/ICDE65448.2025.00350

[48] Adrien Petralia, Paul Boniol, Philippe Charpentier, and Themis Palpanas. 2025. Few Labels are All You Need: A Weakly Supervised Framework for Appliance Localization in Smart-Meter Series . In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. 4386–4399. https://doi.org/10.1109/ICDE65448.2025.00329

[49] Adrien Petralia, Philippe Charpentier, Paul Boniol, and Themis Palpanas. 2023. Appliance Detection Using Very Low-Frequency Smart Meter Time Series. In *Proceedings of the 14th ACM International Conference on Future Energy Systems* (Orlando, FL, USA) *(e-Energy '23)*. Association for Computing Machinery, New York, NY, USA, 214–225. https://doi.org/10.1145/3575813.3595198

[50] Adrien Petralia, Philippe Charpentier, and Themis Palpanas. 2023. ADF & TransApp: A Transformer-Based Framework for Appliance Detection Using Smart Meter Consumption Series. *Proc. VLDB Endow.* 17, 3 (nov 2023), 553–562. https://doi.org/10.14778/3632093.3632115

[51] Daniel Precioso Garcelán and David Gomez-Ullate. 2023. Thresholding methods in non-intrusive load monitoring. *The Journal of Supercomputing* 79 (04 2023), 1–24. https://doi.org/10.1007/s11227-023-05149-8

[52] Hasan Rafiq, Prajowal Manandhar, Edwin Rodriguez-Ubinas, Omer Ahmed Qureshi, and Themis Palpanas. 2024. A review of current methods and challenges of advanced deep learning-based non-intrusive load monitoring (NILM) in residential context. *Energy and Buildings* 305 (2024), 113890. https://doi.org/10.1016/j.enbuild.2024.113890

[53] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi (Eds.). Springer International Publishing, Cham, 234–241.

[54] Ruichen Sun, Kun Dong, and Jianfeng Zhao. 2023. DiffNILM: A Novel Framework for Non-Intrusive Load Monitoring Based on the Conditional Diffusion Model. *Sensors* 23, 7 (2023). https://doi.org/10.3390/s23073540

[55] Stavros Sykiotis, Maria Kaselimi, Anastasios Doulamis, and Nikolaos Doulamis. 2022. ELECTRIcity: An Efficient Transformer for Non-Intrusive Load Monitoring. *Sensors* 22, 8 (2022). https://doi.org/10.3390/s22082926

[56] Chang Wei Tan, Christoph Bergmeir, François Petitjean, and Geoffrey I. Webb. 2021. Time series extrinsic regression. *Data Mining and Knowledge Discovery* 35, 3 (5 2021), 1032–1060. https://doi.org/10.1007/s10618-021-00745-9

[57] L.N. Sastry Varanasi and Sri Phani Krishna Karri. 2024. STNILM: Switch Transformer based Non-Intrusive Load Monitoring for short and long duration appliances. *Sustainable Energy, Grids and Networks* 37 (2024), 101246. https://doi.org/10.1016/j.segan.2023.101246

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR* abs/1706.03762 (2017). arXiv:1706.03762 http://arxiv.org/abs/1706.03762

[59] Lingxiao Wang, Shiwen Mao, and R. Mark Nelms. 2022. Transformer for Non-intrusive Load Monitoring: Complexity Reduction and Transferability. *IEEE Internet of Things Journal* 9, 19 (2022), 18987–18997. https://doi.org/10.1109/JIOT.2022.3163347

[60] Min Xia, Wan'an Liu, Yiqing Xu, Ke Wang, and Xu Zhang. 2019. Dilated residual attention network for load disaggregation. *Neural Computing and Applications* 31, 12 (12 2019), 8931–8953. https://doi.org/10.1007/s00521-019-04414-3

[61] Zhenrui Yue, Camilo Requena Witzig, Daniel Jorde, and Hans-Arno Jacobsen. 2020. BERT4NILM: A Bidirectional Transformer Model for Non-Intrusive Load Monitoring. In *Proceedings of the 5th International Workshop on Non-Intrusive Load Monitoring* (Virtual Event, Japan) *(NILM'20)*. Association for Computing Machinery, New York, NY, USA, 89–93. https://doi.org/10.1145/3427771.3429390

[62] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. 2021. A Transformer-based Framework for Multivariate Time Series Representation Learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 2114–2124. https://doi.org/10.1145/3447548.3467401

[63] Chaoyun Zhang, Mingjun Zhong, Zongzuo Wang, Nigel Goddard, and Charles Sutton. 2018. Sequence-to-point learning with neural networks for non-intrusive load monitoring. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence* (New Orleans, Louisiana, USA) *(AAAI'18/IAAI'18/EAAI'18)*. AAAI Press, Article 318, 8 pages.

[64] Bochao Zhao, Minxiang Ye, Lina Stankovic, and Vladimir Stankovic. 2020. Non-intrusive load disaggregation solutions for very low-rate smart meter data. *Applied Energy* 268 (2020), 114949. https://doi.org/10.1016/j.apenergy.2020.114949

[65] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wan Zhang. 2020. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *AAAI Conference on Artificial Intelligence*. https://api.semanticscholar.org/CorpusID:229156802

[66] Ziwei Zhu, Mengran Zhou, Feng Hu, Kun Wang, Guangyao Zhou, Weile Kong, Yijie Hu, and Enhan Cui. 2024. TSILNet: A novel hybrid model for energy disaggregation based on two-stage improved TCN combined with IECA-LSTM. *Building Simulation* 17, 11 (2024), 2083–2095. https://doi.org/10.1007/s12273-024-1175-9

Table 4: Number of learnable parameters (in millions) according to input sequence window length.

Models	Input subsequences w length		
	$w = 128$	$w = 256$	$w = 512$
NILMFormer	0.385	0.385	0.385
BERT4NILM	1.893	1.91	1.943
STNILM	11.359	11.375	11.408
BiGRU	0.423	0.423	0.423
Energgformer	0.567	0.567	0.567
FCN	0.169	0.3	0.562
UNet-NILM	2.414	2.676	3.202
TSILNet	17.402	34.245	67.931
BiLSTM	4.517	8.728	17.15
DiffNILM	9.221	9.221	9.221
DAResNet	0.331	0.462	0.724

Table 5: Details of training hyperparameters

Hyperparameters	
Init. learning rate	1e-4
Scheduler	ReduceLROnPlateau
Patience reduce lr	5
Batch size	64
Max. number of epochs	50
Early stopping epochs	10

(a) Overall metrics score (avg. of the normalized MAE and MR for $w = 256$, over all the datasets and cases)

(b) Number of trainable parameters (in millions)

Figure 8: Influence of the number of Transformer layer(s) according to the inner model dimension (d_{model}) in NILM-Former on (a) the overall metrics disaggregation score; (b) the number of learnable parameters.

Table 3: NILMFormer hyperparameters

Hyperparameters		
Embedding block	♯ ResBlock	4
	♯ filters	72
	kernel size	3
	Dilation rate	$\{1, 2, 3, 4\}$
Transformer block	♯ Transformer Layers	3
	d_model	96
	♯ heads	8
	PFFN ratio	4
	PFFN activation	GeLU
	Dropout	0.2
Head	♯ filters	128
	kernel size	3

A NILMFORMER ARCHITECTURE DETAILS

This section provides additional details and insights about the NILM-Former architecture.

A.1 Hyperparameters

We report in Table 3 the list of hyperparameters used in NILM-Former for our experiments.

A.2 Hyperparameter Impact

We experimentally evaluate the impact of two main NILMFormer hyperparameters (the inner dimension d and the number of Transformer layers) on the disaggregation performances and the complexity (in terms of the number of trainable parameters). More specifically, we trained and evaluated NILMFormer in the setting described in Section 5 (for all the datasets, cases, and with $w = 256$) for the following inner dimension: $d = \{64, 96, 128, 256\}$, and the following number of Transformer layers $n_l = \{1, 3, 5, 7\}$.

We reported the results using the overall metrics score, computed by averaging the 2 metrics (MAE and MR, averaged to the same range) and averaging the scores across the datasets and cases. The heatmap in Figure 8 (a) indicates that combining 3 layers with an inner dimension of $d = 256$ yields the best disaggregation performance. However, as reported in Figure 8 (b), this combination induces a higher number of trainable parameters (3.55 million). Therefore, due to the real-world deployment of our solution, we opted for a more efficient configuration: 3 layers with an inner dimension $d = 96$. This combination offers a good balance between accuracy and a reduced number of parameters.

B MODELS COMPLEXITY

We examine the number of trainable parameters to compare the complexity of the different NILM baselines used in our experiments. Since this number is influenced by the subsequence window length for certain baselines, we present the number of parameters according to the window length (with $w = \{128, 256, 512\}$). The results, reported in Table 4, indicate that FCN and BiGRU are the smallest models in terms of trainable parameters. However, despite the use of the Transformer architecture, NILMFormer's number of trainable parameters is kept small compared to the other baselines. Specifically, the second and third-best baselines, BERT4NILM and STNILM, contain over 1 million and 11 million parameters, respectively.

C REPRODUCIBILITY

All reported scores are averaged over three household-disjoint train/test splits generated with seeds $\{0, 1, 2\}$.

[Deep-learning Baselines] All DL baselines were re-implemented in PyTorch 2.5.1, except BERT4NILM [61], for which we used the authors' code. Training hyper-parameters appear in Table 5. Models were optimised with Adam, a decaying learning-rate scheduler, and early stopping.

[Other Baselines] ROCKET leverages the Aeon implementation [40], while XGBoost uses the official Python package [10].