The Electricity Journal 37 (2024) 107370

Contents lists available at ScienceDirect

The Electricity Journal

journal homepage: www.elsevier.com/locate/tej

A Transformer based approach to electricity load forecasting

Jun Wei Chan [a,*], Chai Kiat Yeo [a]

[a] Nanyang Technological University, School of Computer Science and Engineering, 50 Nanyang Avenue, Singapore 639798, Singapore

ARTICLE INFO

Keywords:
Computer science
Neural networks
Time series
Load forecasting

ABSTRACT

In natural language processing (NLP), transformer based models have surpassed recurrent neural networks (RNN) as state of the art, being introduced specifically to address the limitations of RNNs originating from its sequential nature. As a similar sequence modeling problem, transformer methods can be readily adapted for deep learning time series prediction. This paper proposes a sparse transformer based approach for electricity load prediction. The layers of a transformer addresses the shortcomings of RNNs and CNNs by applying the attention mechanism on the entire time series, allowing any data point in the input to influence any location in the output of the layer. This allows transformers to incorporate information from the entire sequence in a single layer. Attention computations can also be parallelized. Thus, transformers can achieve faster speeds, or trade this speed for more layers and increased complexity. In experiments on public datasets, the sparse transformer attained comparable accuracy to an RNN-based SOTA method (Liu et al., 2022) while being up to 5× faster during inference. Moreover, the proposed model is general enough to forecast the load from individual households to city levels as shown in the extensive experiments conducted.

1. Introduction

Time series forecasting is a valuable tool, particularly in the power generation industry, where the ability to predict short term load variations allows power generation companies to plan the switching of generators in advance to match load conditions (Nassif et al., 2021). Longer term forecasts would allow planning for fuel purchases, and further opportunities to minimize cost (Alvarez et al., 2021). Currently, the most established deep learning methods for time series forecasting are based on variations of recurrent neural networks (RNN) such as the long short term memory (LSTM), convolutional neural networks (CNN) and hybrid models that combine these techniques. However, the recurrent architecture of RNNs requires the series to be processed in a sequential manner. This results in computational time proportional to the length of the series, which cannot be improved by parallelization with hardware accelerators. Meanwhile, CNNs typically restrict the size of the convolution kernels to reduce the number of parameters, with the trade-off that more layers are needed to allow the output to ‘see’ all inputs, often resulting in very deep networks.

The transformer architecture was introduced in natural language processing (NLP) to address the limitations of RNNs, and to better utilize dedicated matrix multiplication hardware (Vaswani et al., 2017). Unlike RNNs where sequences are processed one element at a time, transformers process the entire input sequence at once, alternating between feed-forward layers that are shared and applied identically to all elements in the sequence, and mixing layers that use weighted combinations of input elements in different positions to form the output. Since its introduction in NLP, transformer based methods have displaced the previous generation of RNN type methods as the state of the art. Subsequently, transformers have also been adapted to tasks other than sequence modelling, such as computer vision tasks (Carion et al., 2020).

The success of transformers in NLP and computer vision has inspired its rising popularity in time series forecasting as well. However, differences between the nature of NLP tasks and time series tasks results in several challenges in adapting transformers for use with time series. Sequences encountered in time series tasks can be much longer than text encountered in NLP, especially for high frequency datasets. For example, the Europarl dataset, for which the associated translation task is evaluated sentence-wise, averages 30 words per sentence (Koehn, 2005). In contrast, a one month time series collected half-hourly, as with the London Smart Meter dataset used in the experiments presented later in this paper, contains 1344 points. The attention layer in transformers compute pairwise weights between all positions in the input sequences, with quadratic memory requirements. This makes transformers difficult to scale efficiently to the long sequences typical in time series tasks. Additionally, the nature of time series data is different from text data.

* Corresponding author.
E-mail address: s200132@e.ntu.edu.sg (J.W. Chan).

https://doi.org/10.1016/j.tej.2024.107370
Received 23 October 2023; Received in revised form 26 January 2024; Accepted 27 January 2024
Available online 7 February 2024

For example, time series data may exhibit periodicity and short-range correlations which are not usually encountered in NLP. The search for more efficient alternatives to the attention mechanism is an active topic for time series transformers.

This paper expands our previous work (Chan and Yeo, 2022) to showcase the scalability, versatility and robustness of our proposed sparse transformer based model. The initial model has been enhanced to support multivariate inputs, which allows it to incorporate additional information such as weather data. We conduct additional extensive experiments on different public datasets to prove that our model is versatile enough to forecast the load beyond individual households as reported in (Liu et al., 2022) to city levels. In addition, we show that our model works as well for short-term prediction of 1 day for the individual households to longer term prediction horizon of 1 week for both the individual households and city levels. In all the experiments, our model is able to perform well based on both long (9 weeks) and short (5 weeks) lookback period during inference without significant increase in inference time, attesting to the scalability of our model. Furthermore, we also investigate the effectiveness of augmenting the load information with weather data to improve the prediction accuracy.

2. Related work

Before the development of deep learning methods for sequence modelling, the dominant techniques for time series prediction are statistical and classical machine learning models such as multiple linear regression (Charlton and Singleton, 2014), support vector regression (Kavaklioglu, 2011; Hong, 2009), Kalman filter (Guan et al., 2013), exponential smoothing (Taylor, 2003), and ARIMA/SARIMA (Huang and Shih, 2003). Statistical models such as ARIMA and exponential smoothing predict future values as a linear sum of historical values. While their construction is simple, their linear nature limits their expressive power (Kaur et al., 2020) and they tend to be sub-par compared to machine learning models in terms of prediction accuracy (Nassif et al., 2021). Likewise, support vector regression also does not scale well to complex data. While hybrid models have been proposed to overcome these limitations, according to Nassif et al., they do not improve accuracy over the base models by any significant amount (Nassif et al., 2021). In Kaur et al. (2020) review, deep learning RNN methods consistently outperform both ARIMA and support vector regression in their benchmarks.

2.1. Convolutional networks

Convolutional networks for sequential data use convolutional layers that convolve learned kernels with the input to detect features (patterns) in the input. The size of the kernel is usually limited, as such, only a limited number of neighbouring input points can affect a corresponding point at the output of the convolutional layer. This is known as the receptive field. To ensure that a point in the final output considers all inputs, the CNN must be deep enough to ensure that the overall receptive field covers all inputs. There is a practical limit to the depth of the CNN due to the vanishing/exploding gradients problem inherent in deep neural networks. Pooling layers are often interleaved to increase the effective size of the receptive fields and reduce the depth required of the CNN. Wang et al. (2017) presented a solar power predictor using a CNN augmented by a wavelet transformation at the input. Deng et al. (2019) used additional residual connections that skip layers to alleviate vanishing/exploding gradients.

2.2. Recurrent neural networks

Recurrent neural networks are neural networks where the intermediate outputs of hidden layers are also fed backwards into the same or earlier layers as inputs to be included in the next computation. This structure thus enables the internal states of the neural network to be updated with each new computation recurrently. The internal state acts as memory, storing and propagating information from past iterations. Hence, the output of the RNN can be influenced by current and past inputs. This property allows RNNs to handle sequential inputs of arbitrary length, making RNNs a natural choice for time series data. However, the recurrent topology results in a deep computational graph, where gradients need to be propagated through the network for as many time steps as there are in the sequence. This leaves the basic RNN susceptible to vanishing or exploding gradients, making them difficult to train. Variants such as LSTMs and gated recurrent units (GRU) address this problem by providing a simpler path for gradients to backpropagate while avoiding major computational blocks.

As a sequence modelling method, LSTMs have been adapted to video prediction tasks, where videos are treated as sequences of images. This inspired Liu et al. (2022)'s TSRNN, which reformulates 1D time series modelling as a video prediction task by reshaping the 1D series into 2D images, which is carried out by an embedded Eidectic LSTM (Wang et al., 2019). By selecting the dimensions of the reshaped images to match known or expected periods in cyclical trends, these trends are emphasized in the resulting images. The grouping of datapoints into images also reduces the effective sequence length, helping to mitigate exploding gradients.

2.3. Transformers

Transformers have been shown to be versatile, producing competitive results in fields outside NLP, including time series modeling (Zerveas et al., 2021). Whereas RNNs process sequences element by element, transformers process the entire sequence at once, alternating between attention and feed-forward layers. Attention blocks compute an output sequence as weighted sums of transformed input elements. This is the mechanism that allows information to flow along the sequence dimension. Positions with higher weight can be described as having more attention associated with them. The feed-forward block implements non-linearity in the transformer layer, and are applied identically to all elements.

As Tsai et al. described in a footnote in Tsai et al. (2019), the feed-forward blocks are position-wise operations applied to each position independently, while the attention blocks are inter-position operations that allow horizontal information transfer.

In the original transformer, dot product attention computes pairwise weights between all elements of the sequence being processed, resulting in a quadratic memory cost to store these intermediate weights, and poor scaling to longer input sequences. Since the introduction of the transformer, various modifications have been proposed to increase its effectiveness on time series tasks. Ahmed et al. (2022) classified improvements to the transformers according to the target component or technique; positional encoding, attention, convolution, gating, and dense interpolation.

Positional encoding is the mechanism used to introduce positional information to the transformer, as otherwise the position-wise operations are replicated and do not distinguish between positions. The original transformer used cosine positional encoding where the absolute position of datapoints were represented by evaluating cosines of different frequencies at their corresponding angular position. Relative positional encodings have been introduced under the hypothesis that considering distance between two elements is more effective (Su et al., 2021; Shaw et al., 2018). Of interest to time series modeling, in particular data such as electricity consumption or consumer prices that are expected to follow seasonal trends according to calendar date and local time, are timestamp positional encoding methods that represent the date and time information (Shankaranarayana and Runje, 2021).

2.4. Sparse attention

Sparse attention addresses the quadratic cost of the original dot

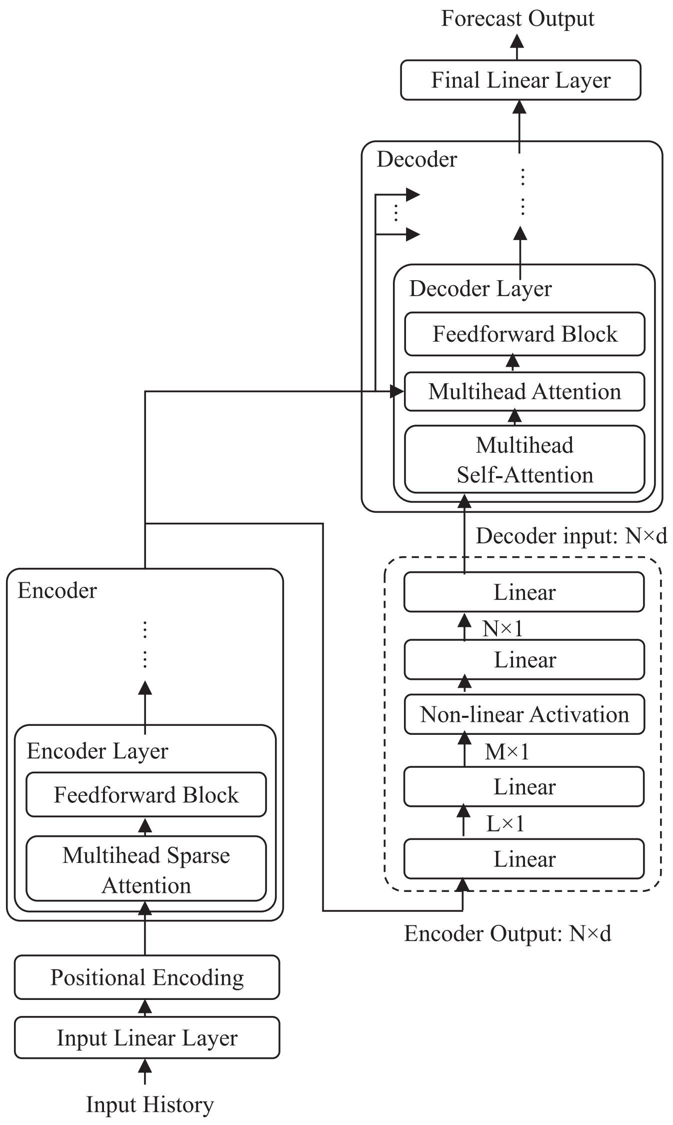


Fig. 1. Overall structure of the proposed transformer based model. L is the length of the input sequence. d is the embedding dimension of the transformer. N is the length of the output sequence. M is the internal intermediate length for the decoder initial sequence generator.

product attention by using a subset of attention weights, selected to scale at a slower rate. For example, LogSparse attention restricts attentions from the layer output to earlier elements in the layer input selected with an exponentially increasing space between them (Li et al., 2019). With N outputs connected to $\log N$ inputs, the footprint of a single layer is improved to $O(N\log(N))$. Even with the additional $\log(N)$ layers needed to restore dense connectivity equivalent to dot product attention, the overall complexity is still an improvement at $O(N\log^2(N))$.

A commonly cited downside of such sparse attention methods is the need for gather operations to collect only the relevant vectors for multiplication, which involve inefficient memory operations.

Big Bird (Zaheer et al., 2020) introduced a sparse attention pattern that does not require gather operations, instead reshaping the input matrices into blocks such that batched matrix multiplications can be used to compute only the desired attention weights. The attention pattern for Big Bird admits the first block of rows and columns, along with the main block diagonal and adjacent upper and lower diagonals. The block diagonals restrict attention to a window around each element, reflecting the locality principle that nearby nodes have greater influence. The block rows and columns of the attention map allows the first block of nodes to involve the whole sequence, and for it to influence all other nodes in the next layer.

An alternative approach to sparsity is linear attention, where kernel method based approximations to the complete dot product attention are used (Wang et al., 2020; Katharopoulos et al., 2020; Zhuoran et al., 2021). In essence, these methods do not directly compute the full attention map, but defines attention as a product in some kernel space, where the kernel feature map is of a more manageable size. However, there is no practical kernel function that exactly matches the softmax definition of attention, therefore these linear attention methods are labeled as approximations. Additionally, these methods have the caveat that a poor choice of kernel function can make training unstable and fail to converge (Tsai et al., 2019).

2.5. *Alternatives to attention*

Li et al. (2019) noted that the original transformer and similar variants process data in a pointwise fashion, while for time series data the context of surrounding points may contain additional information such as the shape of the curve. Incorporating convolution would allow transformers to take into account such contextual features. Convolution layers alone do not have the long-range modeling capability that attention provides. Therefore, most implementations incorporate convolution layers into the transformer pipeline in addition to the existing attention blocks at various locations, either in series (Li et al., 2019) or in parallel (Peng et al., 2022). Similarly, Fourier transforms have also been substituted for attention (Lee-Thorp et al., 2022; Sevim et al., 2022), and may be more suited for time series data due to its application in signal analysis.

3. Proposed method

This paper proposes a transformer equipped with sparse attention to generate forecasts. The initial model proposed in Ref. Chan and Yeo (2022) has been enhanced to support multivariate inputs, which allows it to incorporate additional information such as weather data. The enhancement is made by modifying the input dimension of the linear layer in 1 to permit the model to adapt to multidimensional inputs. This also showcases the versatility of the proposed model design. Sparse attention transformers are better suited for electrical load forecasting compared to full dot-product attention as their reduced memory requirements allow them to handle longer inputs and make forecasts based on a longer history. Fig. 1 shows the overall model architecture.

The transformer consists of two major sections, the first is the encoder, which transforms the input sequence through repeated applications of dot-product attention within the input sequence. This step preserves the length of the sequence. Then, the decoder generates the output sequence by applying dot-product attention between the encoded input and the output sequences as it is being formed, or a placeholder sequence in the first decoder layer. Some works use only the encoder of the transformer (Zerveas et al., 2021). For the proposed model however, the full encoder-decoder structure is used as the decoder can directly generate a forecast sequence of the desired length. (Chan and Yeo, 2022).

The main components of the proposed model are detailed in the following subsections.

3.1. *Input/output*

The attention block computes attention weights as dot products between vectors. Hence, the dimension of the vectors directly correlates to the expressive power of attention. A linear layer maps the 1 dimensional electricity power consumption series to the larger transformer embedding dimension. Likewise, a final linear layer reduces the output from the embedding dimension into a 1 dimensional power forecast. This

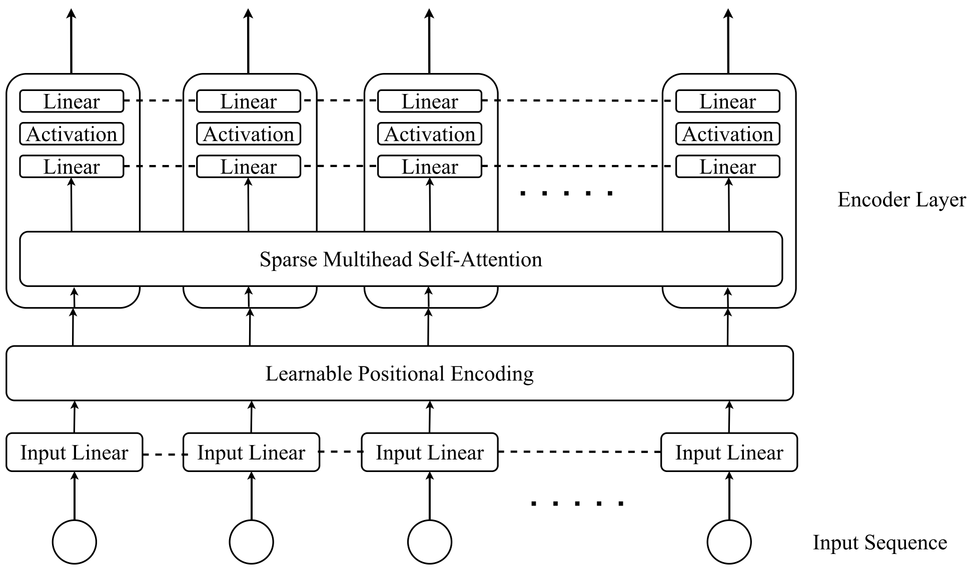


Fig. 2. Transformer structure, encoder side. Dashed lines indicate layers which are shared between inputs. Only the first encoder layer is shown, although the same structure is repeated in later layers (Chan and Yeo, 2022).

arrangement also permits a straightforward adaptation to multidimensional inputs by adjusting the input dimension of the linear layer accordingly. Therefore, we also demonstrate a multidimensional version of the model which augments input electricity data with weather data.

Since, the behaviour of the attention operation and the feed-forward blocks is identical for all sequence positions, positional information is introduced by a fully learnable positional encoding that is added after the first linear layer.

3.2. *Encoder*

The encoder is comprised of multiple encoder layers, each with a multihead self-attention block and a feed-forward block (Fig. 2). The immediate output of encoder is referred to as encoded memory, and has the same shape as the input to the transformer. The memory values are further processed by the decoder to generate the output forecast. The attention mechanism is further explained in the next subsection.

3.3. *Attention*

Dot product attention is the standard attention mechanism implemented in most transformers. As its name implies, it computes attention weights as dot products of query and key vectors.

Self-attention, used in both the encoder and decoder, is described by

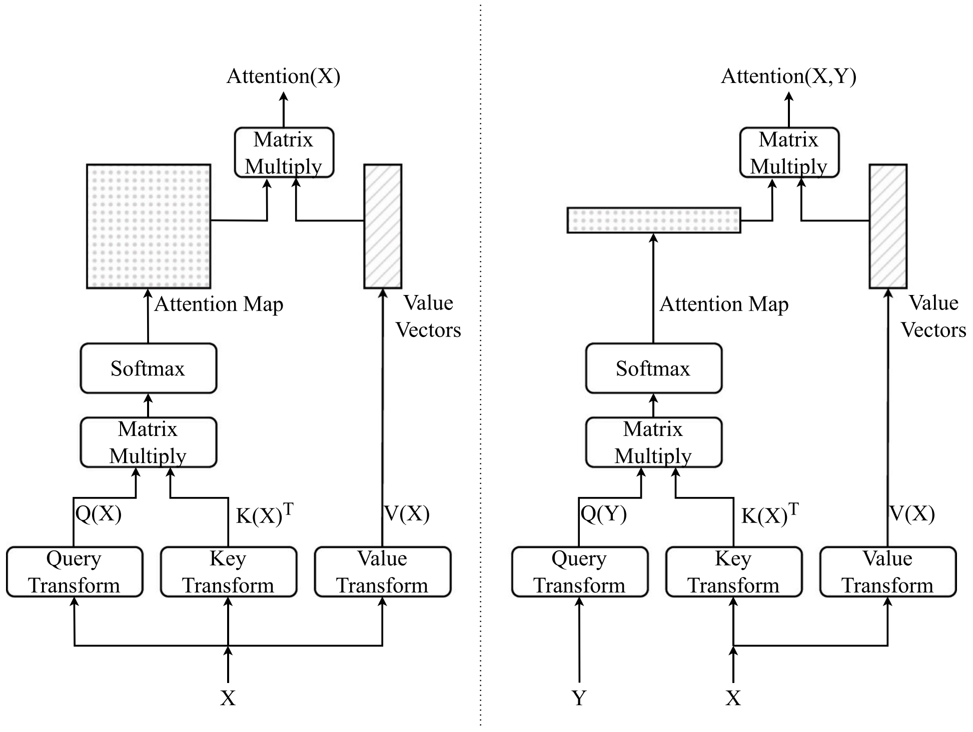
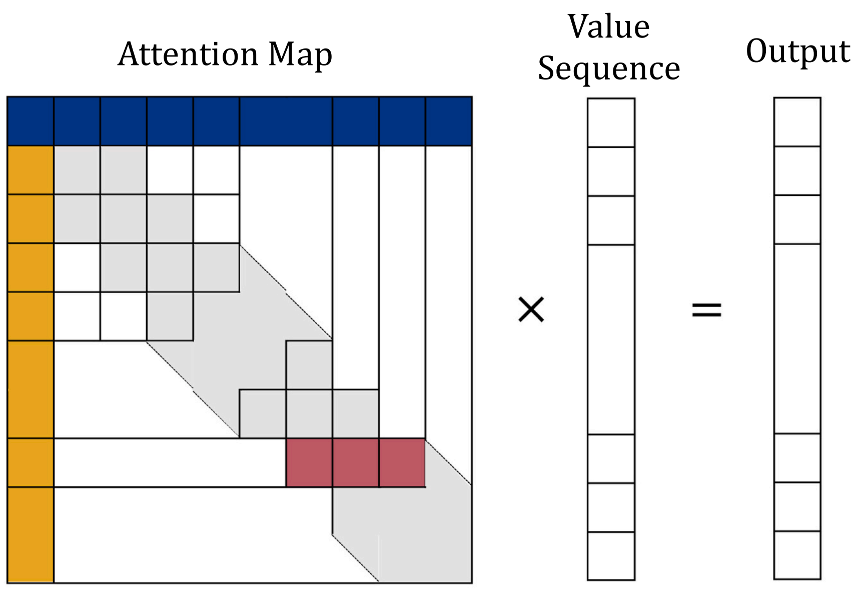


Fig. 3. Visual representation of the attention function as used in the transformer. Left side shows the self-attention as used in the encoder. Right side shows attention as used in the decoder. Variable labels correspond to the variables in equations (1) and (2). (Chan and Yeo, 2022).



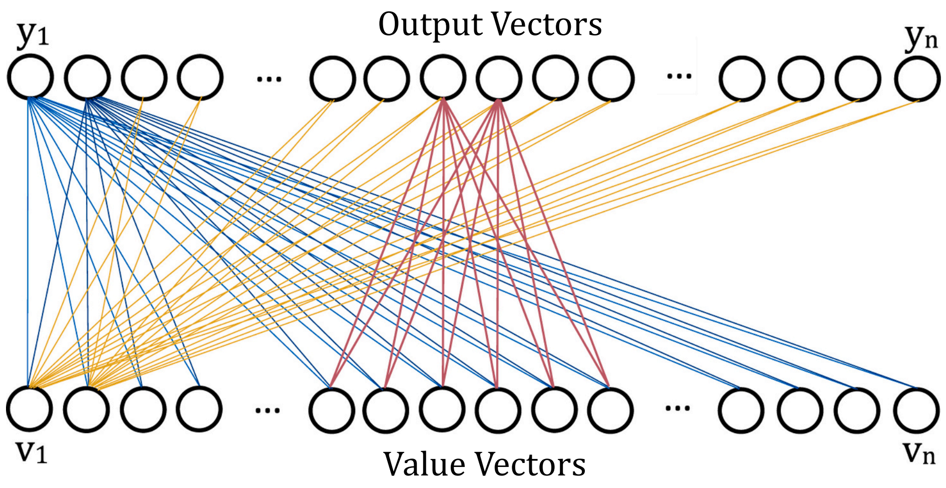


Fig. 4. Big Bird sparse attention pattern. This diagram depicts a block size of 2. The connections between the value vectors on the input side and the output vectors are shown, with their corresponding blocks in the attention map colored accordingly. Gray block connections are omitted for clarity, with red connections providing an example for 1 row of the attention map. The first block of elements are fully connected to all others (blue and yellow). (Zaheer et al., 2020).

the following equation:

$$\text{Attention}_{\text{self}}(X) = \sigma\left(\frac{Q(X)(K(X)))^T}{\sqrt{d}}\right)V(X) \tag{1}$$

where $X \in \mathbb{R}^{n\times d}$ is the n length sequence of d dimensional input vectors, d being the number of embedding dimensions used in the transformer. For example, the experiments on the London dataset presented in this paper uses $n = 1632$ corresponding to 5 weeks of half-hourly sampled data minus 48 samples of forecast, with the internal embedding dimension of the transformer $d = 24$. Q, K, and V are the query, key and value linear transformations, possibly including bias vectors. σ is the softmax function.

This is called self-attention as the query and key vectors are both derived from the input sequence. In the decoder attention block, the query vectors are derived from the output sequence, while the key vectors are derived from the encoder output, as follows:

$$\text{Attention}(X, Y) = \sigma\left(\frac{Q(Y)(K(X)))^T}{\sqrt{d}}\right)V(X) \tag{2}$$

where $Y \in \mathbb{R}^{n'\times d}$ is the sequence of output vectors of some other length n'. In this case, n' would be the desired forecast horizon (Chan and Yeo, 2022). The matrix multiplication $Q(X)(K(X))^T$ $(Q(X)(K(Y))^T$ for decoder attention) computes the pairwise dot products of the query and key vectors, as represented as row vectors. Considering a simplified case where bias values are not used in the query and key transforms, this multiplication can be re-expressed as $\mathbf{x}QK^T\mathbf{y}$, where Q and K are matrices representing their respective transformations. This demonstrates that attention is somewhat analogous to an inner product between the input vectors $\mathbf{x}$ and $\mathbf{y}$. The softmax function normalizes the attention matrix into weight values that sum to 1. Fig. 3 contains a graphical summary of the attention functions as used in the encoder and decoder.

In most implementations, the d embedding dimensions are further split into groups, with each group having its own attention map separate from other groups. This is known as multi-head attention and allows multiple attention patterns to be applied in one layer without the additional cost from using multiple full attention layers side-by-side.

The multiplication of the normlized attention map with the transformed value vectors constructs the output from weighted linear combinations of value vectors. Hence, information in any value vector can be included in any output vector regardless of distance. This is how attention can model long range dependencies in a single layer. At this point, the attention map can be masked if necessary to restrict which value vectors are allowed contribute to the output, as is often done with triangular masks to keep the decoder attention from ‘looking into future values’.

The encoder self-attention contributes the most to the overall computational cost of the transformer, as it is where self-attention is applied on the long look-back sequence. In the decoder layers, self-attention is applied to the much shorter forecast sequence, while attention is between the forecast sequence and the processed look-back sequence. Therefore, only the self-attention blocks in the encoder were substituted with sparse attention. We used Big Bird’s (Zaheer et al., 2020) sparse attention implementation as it efficiently implements sparsity using reshape instead of gather operations. Fig. 4 provides a graphical depiction of Big Bird’s sparsity pattern.

3.4. Decoder

The structure of the decoder is similar to that of the encoder, except with decoder layers that apply attention twice. First, as self-attention on the output sequence being generated with itself, followed by attention between the output and the encoder memory sequence.

The decoder requires an initial sequence of the desired output length as an input. When generating the output sequence, the decoder is typically used in one of two ways. In an autoregressive manner, starting from a single initial element at the start of the sequence, subsequent predicted elements are fed back to the input of the decoder one at a time to form the initial sequence. However, this requires the decoder to be repeatedly run over the length of the output. Instead, we generate the output in a one-shot approach, where the decoder transforms an initial sequence into the output in a single pass without output feedback.

We generate the initial sequence from the encoder memory by applying a 4-layer feed-forward multilayer perceptron (MLP) block. The first two layers transforms the memory vectors from the encoding dimensions to 1 dimension, and expands the sequence in the temporal axis to a larger intermediate size. This is followed by an activation function, and two more layers that maps the sequence to the desired prediction length and back to the transformer encoding dimension.

Finally, the output of the decoder is in the transformer embedding space, and a linear layer is used to map the output back a 1 dimensional series.

4. Experiments and results

We conducted experiments on three datasets and compared the accuracy of our model to the SOTA TSRNN, as well as other baseline methods measured in Ref. Liu et al. (2022) such as Seasonal Auto Regressive Integrated Moving Average (SARIMA, (Durbin and Koopman, 2012)), Exponential Smoothing (Hyndman et al., 2008) and long short-term memory (LSTM).

4.1. Datasets

We conducted separate experiments on three datasets; London Smart Meter (Daignan, 2017), American Electric Power (AEP) (Mulla, 2019), and Red Eléctrica de España (REE) (Spain). This section provides details on the datasets as well as the preprocessing steps.

4.1.1. London Smart Meter

The London Smart Meter dataset provides the energy consumption readings of 5567 households recorded between November 2011 and February 2014, organized into 112 blocks according to household income. These series are collected on a individual basis per household, and are sampled half-hourly. As these series do not have a common starting time, each series is preprocessed separately into non-overlapping subsequences consisting of the lookback period and prediction horizon. Min-max normalization was also performed separately for each household. Series with more than 80 % missing values, and those with insufficient length to make up at least 1 input - ground truth pair, are excluded. The dataset is split randomly according to household ids to maintain a similar distribution of income levels between the training, validation and test sets. Missing values are treated as NaNs, which are replaced with 0 s when they occur in the input to the model. The loss computation ignores positions where NaNs occur in the ground truth.

4.1.2. AEP and REE

Unlike the London dataset which collected data from individual households, the AEP and Spain datasets were obtained from their respective transmission service operators, Red Eléctrica de España (REE) for Spain (Jhana, 2020), and American Electric Power, and contain the aggregated power consumption at a regional level, or at least over consumers connected to the transmission operators. These datasets are significantly smaller than the London dataset. To compensate for the reduced size, instead of chunking the dataset into non-overlapping segments as with the London dataset, the series is split into overlapping segments. Min-max normalization is performed over the entire dataset. Missing values are processed the same way as in the London dataset.

4.2. Hardware

Training and accuracy tests were performed on 2 Nvidia RTX 3090 GPUs. For the timing tests, only 1 GPU was used.

4.3. Setup

The PyTorch framework, version 1.12, was used for these experiments.

For experiments on the AEP dataset, the dataset series is split into 80 % train, 10 % validation, and 10 % test splits, with the first 80 % of values being the training segment, and the last 10 % being the test segment. We used a stride of 29 h between dataset elements to stagger start times so that all hours of the day are included.

For the London Smart Meter dataset, households are randomly distributed into 60 % train, 20 % validation, and 20 % test splits. The energy consumption of each household is stored as a single time series, which is split into sub-series in preprocessing. All sub-series associated with a given household id are added to the dataset split assigned to the household id.

A look-back period of 5 weeks, corresponding to 1632 or 1512 samples after subtracting the forecast horizons, is used in the proposed model. Prediction horizons of 48 samples (1 day) and 168 samples (3.5 days) are tested.

These values are as per those in Ref. Liu et al. (2022). Note that TSRNN’s design requires the total duration of the look-back and the forecast periods to be an integer number of weeks. Batch size is set to the largest multiple of 16 that does not exceed 90 % of available memory.

We sub-classed our transformer model from the transformer class provided with PyTorch, redefining relevant modules and functions accordingly.

The code for the Big Bird sparse attention module was adapted from the Huggingface Transformers library (Wolf et al., 2020).

We tested the transformer with embedding dimension 24, 4 heads for multihead attention, 4 encoder layers, and 4 decoder layers. For the feedforward decoder initial sequence generator block, we used an intermediate sequence length of 2560. The GELU function is used for the activation in the decoder initial sqeuence generator, all other activations use the default function. As recommended in Ref. Zerveas et al. (2021), the normalization layers within the transformer layers were

Table 1
Accuracy scores for look-back period of 5 weeks, and forecast horizons of 2 days, and 1 week on the PJM-AEP dataset. The results of all other methods in this table were obtained from Liu et al. (2022).

	AEP - 2 days (48 samples)			AEP - 1 week (168 samples)		
Model (Liu et al., 2022)	RMSE	MAE	sMAPE	RMSE	MAE	sMAPE
SARIMA	0.057	0.047	0.146	0.113	0.094	0.311
Exponential Smoothing	0.095	0.077	0.226	0.126	0.101	0.322
CatBoost	0.056	0.046	0.139	0.089	0.073	0.259
Pooled Regression	0.057	0.048	0.146	0.090	0.074	0.259
Theta	0.111	0.096	0.312	0.134	0.113	0.386
TBATS	0.096	0.080	0.263	0.194	0.161	0.491
DHR	0.210	0.179	0.618	0.223	0.186	0.618
MLP	0.122	0.108	0.322	0.127	0.107	0.367
DeepAR	0.086	0.069	0.220	0.092	0.076	0.262
NBEATS	0.055	0.046	0.140	0.088	0.072	0.257
Transformer	0.073	0.061	0.180	0.138	0.120	0.503
WaveNet	**0.053**	**0.044**	**0.134**	0.088	0.073	0.257
LSTM	0.114	0.100	0.303	0.121	0.102	0.352
tsRNN	0.060	0.051	0.157	**0.083**	**0.069**	0.246
Proposed sparse transformer	0.060	0.051	0.179	0.084	0.070	0.236
Proposed fixed variant	0.060	0.051	0.177	0.084	0.070	**0.235**

Table 2
Accuracy scores for look-back period of 5 weeks, and forecast horizons of 1 day (48 samples), and 3.5 days (168 samples), on the London dataset.

	London - 1 day			London - 3.5 days		
Model	RMSE	MAE	sMAPE	RMSE	MAE	sMAPE
SARIMA (Liu et al., 2022)	0.126	0.089	0.806	0.126	0.085	0.835
Exponential Smoothing (Liu et al., 2022)	0.124	0.087	0.844	0.131	0.093	0.900
LSTM (Liu et al., 2022)	0.124	0.087	0.844	0.132	0.088	0.819
TSRNN (Liu et al., 2022)	**0.094**	**0.063**	0.711	**0.102**	**0.067**	0.718
Proposed transformer based model	0.106	0.071	0.692	0.114	0.076	0.724
Proposed variant model (fixed)	0.107	0.072	**0.686**	0.114	0.075	**0.714**

implemented as batch normalization.

4.3.1. Static attention maps

From empirical results of early exploratory experiments, we found that in some cases, the parameters of the query and key transformations converged to 0. Even so, the forecasts generated by these models reflected the daily power consumption trend. When the query and key parameters are 0, all transformed query and key vectors, along with their dot products, will be 0 s as well, regardless of the input. This is equivalent to a static attention map where for each position at the output, all inputs are weighted equally (subject to attention masks and sparsity rules). Since the attention matrix is fixed, this presents another avenue for speedup as the attention computation can be substituted by a constant matrix. Therefore, we also test a fixed variant of the proposed model, using the same sparsity pattern. This was implemented as a modification to the attention function, by skipping the query–key matrix multiplications and substituting the results by appropriately scaled ones matrices equivalent to the end result of the softmax operation.

4.4. Training

For all experiments, AdamW was used as the optimizer, with initial learning rate 1e −4, and the default weight decay value. The loss function used during training was the mean squared error loss. A multi-

Table 3
Inference times for look-back period of 5 weeks (1632 samples as in the London dataset). (Chan and Yeo, 2022).

Model	Inference Time (ms)
TSRNN	25.8
Proposed transformer based model	8.80
Proposed variant model (fixed)	4.70

Table 4
Inference times for look-back period of 9 weeks (2928 samples as in the London dataset). (Chan and Yeo, 2022).

Model	Inference Time (ms)
TSRNN	56.5
Proposed transformer based model	9.70
Proposed variant model (fixed)	5.40

step learning rate schedule was used with steps at half, and $\frac{3}{4}$ of the total epochs, with multipliers of 0.4 and 0.2 of the initial learning rate respectively. For experiments involving the AEP and REE datasets, the total epochs is 1000. For experiments on the London dataset, the total epochs is 200. The model is validated after every training epoch, and the weights corresponding to the best validation loss are restored at the end of training.

Accelerated multi-precision with BFloat16 was used for training only.

4.5. Results

4.5.1. Accuracy

Root mean squared error (RMSE), mean absolute error (MAE), and symmetrical mean abolute percentage error (sMAPE) results are computed as the average of 3 runs. The accuracy results are reported in Tables 1 and 2 for AEP and London datasets respectively. For the AEP dataset, the accuracy results for the existing methods such as SARIMA and exponential smoothing are cited directly from Liu et al. (2022).

The proposed model performs comparably to TSRNN for both forecast horizons.

4.5.2. Inference Time

The models are run on 10 random inputs and the inference time measured and averaged. We have previously reported these results (Chan and Yeo, 2022). Look-back periods of 5 weeks and 9 weeks, corresponding to 1632 and 2928 samples as on the London dataset, were tested. For even greater speedup, the fixed attention transformer for these test is implemented by substituting the attention function with a matrix multiplication by a hard coded matrix.

The proposed model and its fixed attention variant are significantly faster than TSRNN. The fixed attention weight variant is almost twice as fast as the proposed original model (Table 3) and without any noticeable decrease in accuracy. Furthermore, tests on 9 weeks of look-back (Table 4) (Chan and Yeo, 2022) also demonstrate that our model scales better to longer (or higher sampling rate) inputs, with its speed advantage over TSRNN increasing further to 5 $\times$ (10 $\times$ for the fixed attention variant).

5. Additional experiments with weather data

In this section, additional weather data are introduced to the proposed model to investigate if it would benefit from the additional information. These experiments are conducted on the London dataset and the Spain dataset (Jhana, 2020) with 5 weeks of lookback period. The same sparse attention transformer from the previous experiment was used. The only modification to the transformer was to increase the input dimensions of the input linear layer to accommodate the additional

Table 5
London Dataset comparison with and without weather data. Accuracy scores for look-back period of 5 weeks and forecast horizon of 1 day.

Data augmentation	RMSE	MAE	sMAPE
Without weather	**0.1062**	**0.0715**	0.6919
With weather	0.1073	0.0718	**0.6840**

Table 6
Spain Dataset comparison with and without weather data. Accuracy scores for look-back period of 5 weeks and forecast horizon of 2 days.

Data augmentation	RMSE	MAE	sMAPE
Without weather	**0.09119**	**0.0756**	**0.1879**
With weather	0.0976	0.0817	0.2003

weather data.

5.1. Weather data

The weather data for the London dataset, obtained from the Darksky API, was already included in (Daignan, 2017) and contains the following; visibility (km), wind direction, temperature (°C), dew point (°C), pressure (hPa), apparent temperature (°C), wind speed, precipitation type (rain or snow), relative humidity, and icon and text summaries of cloud cover and wind level. The weather data was collected at an hourly frequency, while the London power data was at a half-hourly frequency. Therefore linear interpolation was used to substitute for data at half hour points.

The weather data for Spain contains the temperature (K), pressure (hPa), relative humidity (%), wind speed, wind direction, rain amount, and icon and text summary. However, we do not use the rain, icon nor text description. Furthermore, the weather data covers multiple cites, so only the data for the capital, Madrid, which is near the center of REE's coverage area is used to represent the weather of the whole transmission

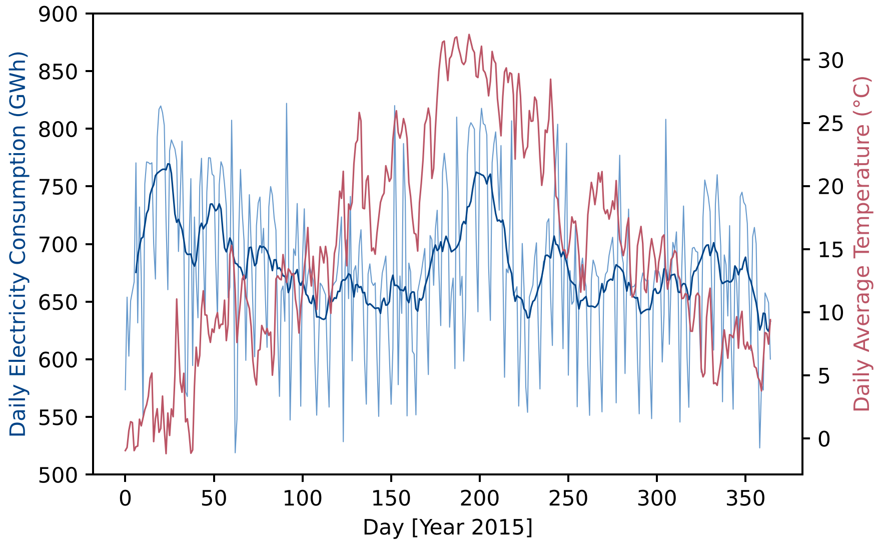


Fig. 5. Plot of both daily average temperature and daily energy consumption of the Spain dataset. The light blue curve depicts daily energy consumption, with a dark blue trendline for clarity, while the red curve depicts the daily average temperature. The plot starts from January 1 and lasts for 1 year.

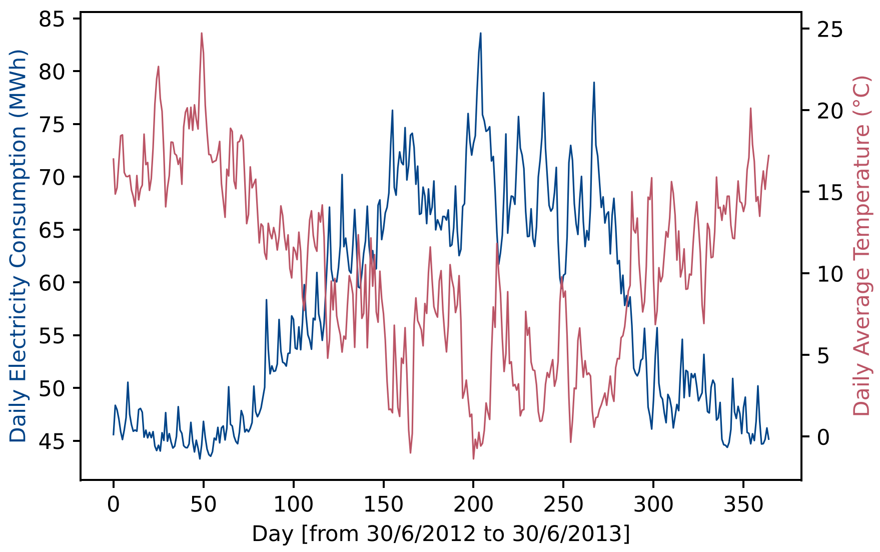


Fig. 6. Plot of aggregated daily energy consumption of households in the London dataset. The blue curve depicts energy consumption, while the red curve depicts power consumption. The plot starts from 30 June and lasts for 1 year.

network.

The weather data for the London dataset introduces 13 additional dimensions, while the weather data for Spain introduces 6. Two dimensions are used for the cosine and sine of the wind direction to avoid discontinuities when angles wrap around 2π. Hence, the input dimensions are 14 for the London dataset and 7 for the Spain dataset. The internal embedding dimension of the transformer is kept unchanged with $d = 24$.

5.2. Training

The same training setup as the previous set of experiments is used.

5.3. Results and discussion

The accuracy results from both datasets are presented in Tables 5 and 6. The inclusion of the weather data does not result in an improvement to the accuracy. This could be due to the fact that weather conditions do not correlate to power consumption strongly enough to influence prediction accuracy. For example, appliances and machinery with fixed power usage independent of weather conditions such as lighting could make up most of the power use and reduce the effect of weather on overall power consumption. Looking at plots of daily average temperature and daily electricity consumption in the Spain dataset in Fig. 5, there are high frequency variations in the daily power consumptions with somewhat constant amplitudes of approximately 100 GW h regardless of temperature. The peaks and troughs of this high frequency cycle matches with weekdays and weekends respectively. This likely reflects the difference in power grid loads between weekdays and weekends, when presumably offices, factories and similar establishments are closed during weekends. This trend follows the days of the week and is independent of weather.

It is also possible that the weather data do not provide any additional information not already present in the electricity data. Again, looking at the power curves in Fig. 5, there is a slower oscillatory trend with a period of around 6 months with peaks around January/February and June/July. This would reflect the need for heating in the winter months and air conditioning in the summer months. The relationship between temperature and power consumption is direct, and is already reflected in the power data. Hence, adding additional weather data does not improve predictions in this case.

Aggregating the power consumption data for all the households in the London dataset and plotting against the daily average temperature shows a similar relationship, with elevated power consumption during colder months. (Fig. 6)

6. Conclusion

This paper presented the application of a sparse attention transformer to the prediction of household and city level electrical loads, with competitive accuracy and speed. Further speedup can be attained by eliminating all attention computations, equivalent to a transformer with a static, input independent attention pattern, with minimal impact on accuracy. This suggests that the full expressive power of the attention mechanism may be unnecessary, at least for the task of electrical load forecasting. We also found that the addition of weather information did not appear to improve accuracy, which we attribute to the relationship between weather and energy consumption being simple and direct, and hence, not contributing any new information.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supporting information

Supplementary data associated with this article can be found in the online version at doi:10.1016/j.tej.2024.107370.

References

Ahmed, S., Nielsen, I.E., Tripathi, A., Siddiqui, S., Rasool, G., Ramachandran, R.P., 2022. Transformers in time-series analysis: a tutorial: arXiv preprint arXiv:2205.01138.
Alvarez, V., Mazuelas, S., Lozano, J.A., 2021. Probabilistic load forecasting based on adaptive online learning.IEEE Transactions on Power Systems. ⟨https://doi.org/10.1109/TPWRS.2021.3050837⟩.
Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S., 2020. End-to-end object detection with transformers. Springe Int. Publ. 213–229. https://doi.org/10.1007/978-3-030-58452-8_13.
Chan, J.W., Yeo, C.K., 2022. Electrical power consumption forecasting with transformers, In: 2022 IEEE Electrical Power and Energy Conference (EPEC), IEEE. pp. 255–260. ⟨https://doi.org/10.1109/EPEC56903.2022.10000228⟩.
Charlton, N., Singleton, C., 2014. A refined parametric model for short term load forecasting. Int. J. Forecast. 30, 364–368. https://doi.org/10.1016/j.ijforecast.2013.07.003.
Daignan, J.M., 2017.Smart meters in london — kaggle. ⟨https://www.kaggle.com/datasets/jeanmidev/smart-meters-in-london?datasetId=4021⟩.
Deng, Z., Wang, B., Xu, Y., Xu, T., Liu, C., Zhu, Z., 2019. Multi-scale convolutional neural network with time-cognition for multi-step short-term load forecasting. IEEE Access 7, 88058–88071. https://doi.org/10.1109/ACCESS.2019.2926137.
Durbin, J., Koopman, S.J., 2012. Time Series Analysis by State Space Methods, 2nd ed. Oxford University Press ⟨https://academic.oup.com/book/16563⟩. 10.1093/acprof:oso/9780199641178.001.0001.
Guan, C., Luh, P.B., Michel, L.D., Chi, Z., 2013. Hybrid kalman filters for very short-term load forecasting and prediction interval estimation. IEEE Trans. Power Syst. 28, 3806–3817. https://doi.org/10.1109/TPWRS.2013.2264488.
Hong, W.C., 2009. Electric load forecasting by support vector model. Appl. Math. Model. 33, 2444–2454. https://doi.org/10.1016/j.apm.2008.07.010.
Huang, S.J., Shih, K.R., 2003. Short-term load forecasting via arma model identification including non-gaussian process considerations. IEEE Trans. Power Syst. 18, 673–679. https://doi.org/10.1109/TPWRS.2003.811010.
Hyndman, R., Koehler, A., Ord, K., Snyder, R., 2008. Forecasting with Exponential Smoothing. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-71918-2.
Jhana, N., 2020. Hourly energy demand generation and weather — kaggle.⟨https://www.kaggle.com/datasets/nicholasjhana/energy-consumption-generation-prices-and-weather⟩.
Katharopoulos, A., Vyas, A., Pappas, N., Fleuret, F., 2020. Transformers are rnns: fast autoregressive transformers with linear attention. In: III, H.D., Singh, A. (Eds.), Proceedings of the 37th International Conference on Machine Learning, pp. 5156–5165 (PMLR). ⟨https://proceedings.mlr.press/v119/katharopoulos20a.html⟩ (PMLR).
Kaur, D., Islam, S.N., Mahmud, M.A., Haque, M.E., Dong, Z., 2020.Energy forecasting in smart grid systems: A review of the state-of-the-art techniques.arXiv preprint arXiv:2011.12598.
Kavaklioglu, K., 2011. Modeling and prediction of turkey's electricity consumption using support vector regression. Appl. Energy 88, 368–375. https://doi.org/10.1016/j.apenergy.2010.07.021.
Koehn, P., 2005.Europarl: A parallel corpus for statistical machine translation, In: Proceedings of Machine Translation Summit X: Papers, 79–86. ⟨https://aclanthology.org/2005.mtsummit-papers.11⟩.
Lee-Thorp, J., Ainslie, J., Eckstein, I., Ontanon, S., 2022.Fnet: Mixing tokens with fourier transforms, In: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics. pp. 4296–4313. ⟨https://doi.org/10.18653/v1/2022.naacl-main.319⟩.
Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.X., Yan, X., 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In: Wallach, H., Larochelle, H., Beygelzimer, A., d Alché-Buc, F., Fox, E., Garnett, R. (Eds.), Advances in Neural Information Processing Systems. Curran Associates, Inc.. In: ⟨https://proceedings.neurips.cc/paper/2019/file/6775a0635c302542da2c32aa19d86be0-Paper.pdf⟩
Liu, Y., Dutta, S., Kong, A.W.K., Yeo, C.K., 2022. An image inpainting approach to short-term load forecasting. IEEE Trans. Power Syst. 1. https://doi.org/10.1109/TPWRS.2022.3159493.
Mulla, R., 2019. Hourly energy consumption — kaggle.⟨https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption/data⟩.
Nassif, A.B., Soudan, B., Azzeh, M., Attilli, I., Almulla, O., 2021. Artificial intelligence and statistical techniques in short-term load forecasting: a review. Int. Rev. Model. Simul. (IREMOS) 14, 408. https://doi.org/10.15866/iremos.v14i6.21328.
Peng, Y., Dalmia, S., Lane, I., Watanabe, S., 2022. Branchformer: Parallel mlp-attention architectures to capture local and global context for speech recognition and understanding, In: Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S., (Eds.), Proceedings of the 39th International Conference on Machine Learning, PMLR.17627–17643.⟨https://proceedings.mlr.press/v162/peng22a.html⟩.
Sevim, N., Özyedek, E.O., Şahinuç, F., Koç, A., 2022.Fast-fnet: Accelerating transformer encoder models via efficient fourier layers.arXiv preprint arXiv:2209.12816.

Shankaranarayana, S.M., Runje, D., 2021. Attention augmented convolutional transformer for tabular time-series. In: Proceedings of the 2021 International Conference on Data Mining Workshops (ICDMW), IEEE. pp. 537–541. ⟨https://doi.org/10.1109/ICDMW53433.2021.00071⟩.

Shaw, P., Uszkoreit, J., Vaswani, A., 2018. Self-attention with relative position representations, In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), Association for Computational Linguistics. pp. 464–468. ⟨https://doi.org/10.18653/v1/N18-2074⟩.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., Liu, Y., 2021. Roformer: Enhanced transformer with rotary position embedding.arXiv preprint arXiv:2104.09864.

Taylor, J.W., 2003. Short-term electricity demand forecasting using double seasonal exponential smoothing. J. Oper. Res. Soc. 54, 799–805. https://doi.org/10.1057/palgrave.jors.2601589.

Tsai, Y.H.H., Bai, S., Yamada, M., Morency, L.P., Salakhutdinov, R., 2019.Transformer dissection: an unified understanding for transformer's attention via the lens of kernel, In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, pp. 4343–4352. ⟨https://doi.org/10.18653/v1/D19-1443⟩.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Łukasz Kaiser, Polosukhin, I., 2017. Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems. Curran Associates, Inc. In: ⟨https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf⟩.

Wang, H., Yi, H., Peng, J., Wang, G., Liu, Y., Jiang, H., Liu, W., 2017. Deterministic and probabilistic forecasting of photovoltaic power based on deep convolutional neural network. Energy Convers. Manag. 153, 409–422. https://doi.org/10.1016/j.enconman.2017.10.008.

Wang, S., Li, B.Z., Khabsa, M., Fang, H., Ma, H., 2020. Linformer: Self-attention with linear complexity.arXiv preprint arXiv:2006.04768.

Wang, Y., Jiang, L., Yang, M.H., Li, L.J., Long, M., Fei-Fei, L., 2019. Eidetic 3d lstm: A model for video prediction and beyond, in: Proceedings of the International Conference on Learning Representations. ⟨https://openreview.net/forum?id=B1lKS2AqtX⟩.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M., 2020. Transformers: State-of-the-art natural language processing, In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics. pp. 38–45. ⟨https://www.aclweb.org/anthology/2020.emnlp-demos.6⟩.

Zaheer, M., Guruganesh, G., Dubey, K.A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., Ahmed, A., 2020. Big bird: transformers for longer sequences. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (Eds.), Advances in Neural Information Processing Systems. Curran Associates, Inc.. In: ⟨https://proceedings.neurips.cc/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf⟩

Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A., Eickhoff, C., 2021.A transformer-based framework for multivariate time series representation learning, In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, ACM. pp. 2114–2124. ⟨https://doi.org/10.1145/3447548.3467401⟩.

Zhuoran, S., Mingyuan, Z., Haiyu, Z., Shuai, Y., Hongsheng, L., 2021. Efficient attention: Attention with linear complexities, In: 2021 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, pp. 3530–3538. ⟨https://doi.org/10.1109/WACV48630.2021.00357⟩.

Jun Wei Chan is a Ph.D candidate with the School of Computer Science and Engineering, Nanyang Technological University. His current research interests are in time series forecasting and related applications.

Chai Kiat Yeo is an Associate Professor with School of Computer Science and Engineering, Nanyang Technological University. She is also the Deputy Director and Programme Lead of Singtel Cognitive and Artificial Intelligence Lab for Enterprises@NTU. Her research interests include anomaly detection, machine learning, artificial intelligence, predictive operational analytics, ad hoc and mobile networks.