Gazi University
Journal of Science
PART A: ENGINEERING AND INNOVATION
http://dergipark.org.tr/gujsa

IoT-Based Energy Consumption Prediction Using Transformers

Abdul Amir ALIOGHLI[1*] Feyza YILDIRIM OKAY[2]

[1] Gazi University, Department of Computer Science, Ankara, Türkiye
[2] Gazi University, Department of Computer Engineering, Ankara, Türkiye

Keywords
Transformers
Time-Series
Prediction
IoT

Abstract
With the advancement of various IoT-based systems, the amount of data is steadily increasing. The increase of data on a daily basis is essential for decision-makers to assess current situations and formulate future policies. Among the various types of data, time-series data presents a challenging relationship between current and future dependencies. Time-series prediction aims to forecast future values of target variables by leveraging insights gained from past data points. Recent advancements in deep learning-based algorithms have surpassed traditional machine learning-based algorithms for time-series in IoT systems. In this study, we employ Enc & Dec Transformer, the latest advancements in neural networks for time-series prediction problems. The obtained results were compared with Encoder-only and Decoder-only Transformer blocks as well as well-known recurrent based algorithms, including 1D-CNN, RNN, LSTM, and GRU. To validate our approach, we utilize three different univariate time-series datasets collected on an hourly basis, focusing on energy consumption within IoT systems. Our results demonstrate that our proposed Transformer model outperforms its counterparts, achieving a minimum Mean Squared Error (MSE) of 0.020 on small, 0.008 on medium, and 0.006 on large-sized datasets.

Cite
Alioghli, A. A., & Yildrim Okay, F. (2024). IoT-Based Energy Consumption Prediction Using Transformers. *GU J Sci, Part A*, *11*(2), 304-323. doi:10.54287/gujsa.1438011

Author ID (ORCID Number)

0009-0001-7273-3292	Abdul Amir ALIOGHLI
0000-0002-6239-3722	Feyza YILDIRIM OKAY

Article Process

Submission Date	15.02.2024
Revision Date	19.03.2024
Accepted Date	03.04.2024
Published Date	05.06.2024

1. INTRODUCTION

The latest Internet of Things (IoT), Analytics 'State of IoT—Spring 2023' report indicates a significant surge in global IoT connections, marking an 18% increase in 2022, with 14.3 billion active IoT endpoints. Projections by IoT Analytics suggest a further 16% growth in 2023, pushing the global number of connected IoT devices to an expected 16.7 billion active endpoints (IoT Analytics, 2023). This continual rise indicates an ongoing expansion of IoT device connections, with these devices increasingly found in homes, businesses, factories, and hospitals (Pashamokhtari, 2020). With the rapid proliferation of the IoT, there is an exponential generation of extensive time-series data (Hu et al., 2023). This data, characterized by repeated observations of various variables over time, is a product of IoT, significantly contributing to the vastness and velocity of big data applications (Adhikari & Agrawal, 2013). It encompasses diverse predictions such as energy consumption, temperature fluctuations, light intensity measurements, among other factors.

Mathematically, a time-series is defined as $\{x_1, x_2, x_3 \dots x_T\}$, where $t = 1, 2, \dots, T$ denotes time, ranging from 1 to T, and x_T signifies a vector of random variables (Cochrane, 1997). The significance of time-series data derived from IoT in scientific and technological research cannot be overstated. Analysis of this data reveals hidden patterns and laws, enabling its application in environmental contexts. This analysis further aids in uncovering correlations and periodicities between events, fostering a deeper understanding of their nature and mechanisms. The insights derived provide robust support across various disciplines engaged in related research

*Corresponding Author, e-mail: aamir.oghli@gazi.edu.tr

(Hu et al., 2023). With the vast array of data generated by IoT, accurate analysis and prediction of energy-related data have emerged as critical areas. The role of such analysis is pivotal in curbing energy wastage (Shapi et al., 2021). According to the international energy agency, global energy demand is declining by 3.8% to 6% each year in developed countries, while it is increasing in developing countries due to their escalating energy usage, ranging between 4% and 7% (Raheem et al., 2022). This imbalance in energy consumption necessitates an accurate forecast more than ever before.

Time-series prediction expects the future distribution of target variables by analyzing past observations within the time-series. This approach has been applied to various temporal inferencing problems, such as filtering, smoothing, and predicting unobserved past events or alternative histories (Russell & Norvig., 2020). The models utilized for energy consumption included time-series prediction, spanning from the past to the present, can be categorized into three distinct groups based on their abilities and advancements (Shi et al., 2022): traditional models, classical machine-learning, and deep learning models. Models such as ARIMA and SARIMA (Hipel & McLeod, 1994; Box et al., 2015) were initially employed as traditional for linear and non-linear time-series analysis, addressing stationary and non-stationary data, respectively. As machine learning algorithms progressed, researchers ventured into implementing standard algorithms like SVM (Cao, 2003) in this domain. Additionally, hybrid techniques combining ARIMA and SVM have been employed specifically for short-term time-series prediction (Nie et al., 2012). neural networks and artificial neural networks have demonstrated superior performance compared to classical machine learning and traditional techniques in energy consumption IoT-based generated data (Nor et al., 2017) (Masum & Chiverton, 2018). Among the widely used deep learning algorithms for time-series prediction are 1D-CNN (Markova, 2022), RNN (Coulibaly & Baldwin, 2005), LSTM (Sahoo et al., 2019), and GRU (Afanasieva & Platov, 2019).

The literature review on deep learning for time-series prediction (Tealab, 2018; Torres et al., 2021), indicating that all previous recurrent neural network-based algorithms have problems detecting relationships between input features of time-series data, as well as being unable to capture long sequences. A recent significant advancement in deep learning is the multi-head attention-based approach, which has shown remarkable results in Natural Language Processing (NLP), introduced (Vaswani et al., 2017), this technique has found successful applications in various domains, including computer vision and speech recognition (Carion et al., 2020), (Zeyer et al., 2019). Various variations of Transformers have been implemented based on a survey on time-series for transformers, as conducted (Wen et al., 2023). The transformer network comprises a sequence of encoder and decoder blocks, with each block featuring a residual connection employed in the time-series data. Additionally, each layer of transformers is utilized independently in various areas of time-series forecasting problems. To the best of our knowledge, there are currently no studies incorporating the dataset we utilized and analyzing it with Enc&Dec Transformer and its variants, including Encoder-only and Decoder-only Transformer models. Our dataset contains smart home energy data, and the precise predictions made by these efficient algorithms aid consumers in balancing their consumption and determining appropriate pricing strategies. Furthermore, there is a lack of a defined methodology for assessing the performance of transformer-based neural networks across varying volumes of time-series data, spanning low, medium, and high quantities, particularly concerning multi-step predictions.

Therefore, in this paper, our approach involves using vanilla transformers to predict univariate multi-step time series energy consumption data generated by IoT devices, as well as employing encoder-only and decoder-only transformer layers separately. In particular, our aim is to demonstrate the superior success of our proposed model, the Enc&Dec transformer, in analyzing IoT time series data compared to leading deep learning methods in other literature. We evaluate our models on three dataset sizes in the energy domain: small, medium, and large.

Motivated by our contributions to the literature, our study aims to achieve the following objectives:

- Implementation of the Enc & Dec Transformer in its fundamental form, specifically targeting univariate time-series prediction within the energy domain data.
- implementing the Encoder-only and Decoder-only blocks of Transformers separately for the time-series prediction problem.
- Comparison of the obtained results with prevalent deep learning-based algorithms commonly utilized in time-series prediction across datasets of varying sizes, low, medium and high data volumes.

The remaining sections of the paper are organized as follows: In Section 2, the existing studies in the literature are discussed. Section 3 gives background information about recurrent-based neural network algorithms. Section 4 presents and explains our developed model. Section 5 covers experimental analysis, including dataset, hyperparameter optimization, experimental setup, and evaluation metrics, all of which are detailed. In Section 6, the experimental results of models are discussed. Finally, in Section 7, the study is summarized, and key points are emphasized. Furthermore, the detailed acronyms and their abbreviations used throughout the entire study are listed in Table 1.

Table 1. *List of Abbreviations and Acronyms*

Abbreviation	Definition	Abbreviation	Definition
1D-CNN	One-Dimensional Convolutional Neural Network	**MIMIC-II**	Multiparameter Intelligent Monitoring in Intensive Care II
ARIMA	Autoregressive Integrated Moving Average	**MLP**	Multi-Layer Perceptron
ATM	Automated Teller Machine	**MMC**	MovieLens
C-MAPSS	Commercial Modular Aero-Propulsion System Simulation	**MSE**	Multi-Layer Perceptron
DUQ	Duquesne Light Company	**MW**	Megawatts
EKPC	East Kentucky Power Cooperative	**NLP**	Natural Language Processing
FD001	Flight Degradation Simulation 001,002,003,004	**ReLU**	Rectified Linear Unit
GPU	Graphics Processing Unit	**RMSE**	Root Mean Squared Error
GRU	Gated Recurrent Unit	**RNN**	Megawatts
IoT	Internet of Things	**RT**	Reuters-21578 Text Categorization Collection
KDD	Knowledge Discovery in Databases	**RUL**	Remaining Useful Life
LSTM	Long Short-Term Memory	**SARIMA**	Seasonal Autoregressive Integrated Moving Average
MAE	Mean Absolute Error	**SOF**	Stack Overflow
MHA	Multi-head Attention	**SVM**	Support Vector Machine

2. RELATED WORK

The original Transformer architecture is a multifaceted model comprising an encoder and a decoder. However, drawing inspiration from NLP, within the expansive domain since its introduction for time-series data, literature has found value in either utilizing both or just one of these components. According to Table 2, summarized related works of transformers for time-series prediction are from modifications of structure-based categorized into three parts including: Enc & Dec Transformer, Encoder-only Transformer, and Decoder-only Transformer in different applications.

An initial application of the self-attention concept in time-series forecasting is Cross-Dimensional Self-Attention, which is tailored for multivariate, geo-tagged time-series imputation (Ma et al., 2019). The proposed novel approach aims to jointly capture self-attention across multiple dimensions, such as time, location, and sensor measurements, while keeping computational complexity low. It processes each dimension sequentially, yet in a manner that is independent of order, as presented in the study. Extensive experiments conducted on four real-world datasets show that this method surpasses state-of-the-art imputation and forecasting techniques in performance. In another extensive investigation, an encoder-decoder Transformer architecture is employed for time-series forecasting, with a focus on influenza-like illness prediction (Wu et al., 2020). The study introduces a vanilla Transformer architecture for both univariate and multivariate time-series predictions. The findings demonstrate that the forecasting outcomes achieved by their method are notably comparable to the current state-of-the-art approaches. Subsequently, an upgraded iteration of the Transformer was introduced (Li et al., 2019), incorporating causal convolution within the self-attention module to enhance the model's responsiveness to local context. Additionally, adjustments were made to mitigate the memory overhead of Transformers, rendering them more adept at handling lengthy time series. A pioneering Temporal Fusion

Transformer emerged (Lim et al., 2020), combining recurrent and attention layers to discern temporal dependencies across various scales within numerous real-world datasets. At this time, the transformer-based neural networks predominantly featured an encoder-decoder architecture.

***Table 2**. Summary of Related Works of Transformers for Time-Series Prediction*

Model	Reference Title	Year	Dataset	Data Type
Enc&Dec Transformer	CDSA: Cross-Dimensional Self-Attention for Multivariate, Geo-tagged Time Series Imputation	2019	Traffic, KDD cup 2015&2018	Multivariate
	Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case	2020	ILI reports from Centers for Disease Control and Prevention	Univariate & Multivariate
	Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting	2020	Real-worlds (Traffic and Electricity) and Synthetic dataset	Univariate & Multivariate
	Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting	2020	Real-world datasets (Traffic, Electricity, Volatility, and Favorita)	Multivariate
Encoder-only Transformer	Self-Attentive Hawkes Process	2020	Real-world datasets (RT, SOF, and MMC) and Synthetic dataset	Multivariate
	Transformer Hawkes Process	2021	Retweets, MemeTrack, Finanial, MIMIC-II, SOF, 911-Calls, Earthquake datasets	Univariate & Multivariate
	Remaining useful life estimation via transformer encoder enhanced by a gated convolutional unit	2021	C-MAPSS datasets (FD001, FD002, FD003, and FD004)	Multivariate
	A Transformer-based Framework for Multi-variate Time Series: A Remaining Useful Life Prediction Use Case	2023	C-MAPSS datasets (FD001, FD002, FD003, and FD004)	Multivariate
Decoder-only Transformer	Evaluation of the Transformer Architecture for Univariate Time Series Forecasting	2021	Traffic, Tourism, Financial (Exchange rate, Daily ATM's cash, and artificially generated) M4, M3, and Solar Energy datasets	Univariate
	Persistence Initialization: a novel adaptation of the Transformer architecture for time series forecasting	2022	M4 dataset	Univariate & Multivariate
	A Decoder-Only Foundation Model for Time-Series Forecasting	2024	Google Trends, Wiki Pageviews, M4	Univariate & Multivariate

Despite outperforming recurrent-based and convolutional-based neural networks on time-series data, they have problems including the requirement for high computational resources, time-consuming inference, and complex architectures that are challenging to employ in real-world scenarios. Therefore, to overcome these issues more effectively, some studies utilize the Encoder layer (Encoder-only Transformer), as well as the Decoder layer (Decoder-only Transformer), to sought solutions to various time-series prediction problems.

Self-Attentive Hawkes Processes and Transformer Hawkes Process models are proposed as Encoder-only Transformers for event forecasting problem (Zhang et al., 2019; and Zuo et al., 2020). Both methods utilize a Transformer encoder architecture to capture the impact of past events and calculate the intensity function for event prediction, representing a modification at the architecture level to an Encoder-only structure. They adjust the positional encoding by converting time intervals into sinusoidal functions, enabling the utilization of event intervals. A hybrid approach has been proposed for the prediction of Remaining Useful Life (RUL), employing a transformer encoder architecture combined with a gated convolutional unit. This approach utilizes the gated convolution to extract local features and employs the encoder transformer to capture global features (Mo et al., 2021). Furthermore, in the domain of RUL prediction, a recent comprehensive method has been introduced. This method is based on an encoder-transformer architecture for multivariate time series prediction (Ogunfowora & Najjaran, 2023). The study employs a basic transformer encoder block, which comprises four main sub-modules: multi-head attention, positional encoding, layer normalization, skip connections, and feed-forward neural network layers.

A Decoder-only Transformer structure-based method, geared towards univariate time-series forecasting, is introduced (Lara-Benítez et al., 2021). In this investigation, the conventional Transformer Decoder blocks were adhered to, wherein each decoder block comprises a masked self-attention module followed by multi-head attention and a feed-forward block. Additionally, all sub-modules incorporate a residual connection, followed by dropout and batch normalization layers, to enhance the network's generalization capacity. Similarly, the Persistence Initialization framework is outlined, consisting of four components: normalization, linear projections, a decoder-only Transformer incorporating Rotary positional encodings and ReZero normalization, and Persistence Initialization (Haugsdal et al., 2023). Finally, in a recent work, a decoder-only pretrained attention-based model for time-series forecasting is proposed (Das et al., 2024). It serves as a foundation for decoder-only pretrained applications for time-series forecasting, consisting of an input layer, stacked transformer decoder blocks, and an output layer.

3. BACKGROUND INFORMATION

3.1. Recurrent neural network (RNN)

RNN is particularly well-suited for sequential data and is commonly employed in time-series analysis (Javaid, 2019). RNN's employ recurrent neural architectures to grasp the functional relationships between input characteristics from the immediate past and a future target variable (Coulibaly & Baldwin, 2005). The computational procedure defining each hidden state (hidden unit or hidden cell) can be mathematically define, as demonstrated in Equation (1) within our framework.

$$\mathrm{S_t = tanh(W_{xs} \cdot (x_t \oplus S_{t-1}) + b_s)\ \&\ y_t = \sigma(W_y \cdot S_t + b_y)} \tag{1}$$

where $\mathrm{x_t} \in \mathrm{R^m}$ represents input vector comprising m input features at time t; $\mathrm{W_{xs}} \in \mathrm{R^{n\times(m+n)}}$ and $\mathrm{W_y} \in \mathrm{R^{n\times m}}$are parameter matrices; n denotes number of neurons in the RNN; $\mathrm{b_s} \in \mathrm{R^n}$ and $\mathrm{b_y} \in \mathrm{R^n}$ are bias vectors for internal state and output, respectively; σ denotes the sigmoid activation; $\mathrm{S_t}$ represents the internal (hidden) state; and $\mathrm{x_t} \oplus \mathrm{S_t}$ shows concatenation of vectors $\mathrm{x_t}$ and $\mathrm{S_{t-1}}$.
One of the significant drawbacks of RNNs is their susceptibility to the gradient vanishing problem, stemming from repeated multiplication of the recurrent weight matrix. This issue leads to diminishing gradients over time, causing the RNN to retain information effectively only for shorter durations (Sahoo et al., 2019).

3.2. Long short-term model (LSTM)

LSTM network is a variation of RNN, offer partial mitigation of the vanishing gradient problem (Sahoo et al., 2019). and excel in capturing longer-term dependencies within time-series data. LSTMs utilize gates—forget gate, input gate, addition gate, and output gate—to manage the removal, multiplication, addition, and filtering of information. The computational procedures implementing these functions, in our context $f_t, i_t, \hat{\mathrm{C}}_t$ and O_t , respectively are described in Eq. (2).

$$\begin{gathered}
f_t = \sigma\big(W_f \cdot (x_t \otimes S_{t-1}) + b_f\big); \\
i_t = \sigma(W_i \cdot (x_t \otimes S_{t-1}) + b_i) \\
\hat{\mathrm{C}}_t = tanh(W_c \cdot (x_t \otimes S_{t-1}) + b_c); \\
C_t = f_t \cdot C_{t-1} \cdot i_t \cdot \hat{\mathrm{C}}_t; \\
O_t = \sigma(W_o \cdot (x_t \otimes S_{t-1}) + b_o); \\
S_t = tanh(C_t) \cdot O_t\ \ \&\ y_t = \sigma(W_y \cdot S_t + b_y)
\end{gathered} \tag{2}$$

where $x_t \in R^m$ represents input vector comprising m input features at time t; $W_f, W_i, W_c, W_o \in R^{n\times(m+n)}$ and $W_y \in R^{n\times m}$ are parameter matrices; n denotes number of neurons in LSTM layer; $b_f, b_i, b_c, b_o \in R^n$ are bias vectors; σ denotes the sigmoid activation; and S_t represents the internal (hidden) state. The functions $f_t, i_t, \hat{C}_t$ and O_t correspond to the forget gate, input gate, addition gate, and output gate, respectively.

3.3. Gated recurrent unit (GRU)

GRU represent a variation of LSTM network aimed at further mitigating the vanishing gradient problem (Afanasieva & Platov, 2019). In the computational process described by Eq. (3), the distinctive aspect of this approach lies in utilization of gets including: update, reset, and a third get, which implement the functions $z_t, r_t$and $\check{S}_t$, respectively. Each gate serves a distinct role in regulating how prior information is filtered, utilized, and amalgamated. The initial element in the formula for the next state, expressed as $(1 - z_t) \cdot S_{t-1}$, determines the information retained from the past, while $z_t \cdot \check{S}_t$ decides the content to be included from the current memory.

$$\begin{gathered} r_t = \sigma\left(W_r \cdot \left(x_t \otimes S_{t-1}\right) + b_r\right) \\ z_t = \sigma\left(W_z \cdot \left(x_t \otimes S_{t-1}\right) + b_z\right) \\ \check{S}_t = tanh\left(W_s \cdot \left(x_t \otimes S_{t-1} \cdot r_t\right) + b_s\right) \\ S_{t-1} = (1 - z_t) \cdot S_{t-1} + z_t \cdot \check{S}_t \quad \& \quad y_t = \sigma(W_y \cdot S_t + b_y) \end{gathered} \tag{3}$$

where $x_t \in R^m$ represents the input vector with m input features at time t; $W_r, W_z, W_S, \in R^{n\times(m+n)}$and $W_y \in R^{n\times m}$ are parameter matrices; n denotes the number of neurons in the GRU layer; $b_r, b_z, b_s \in R^n$ are bias vectors; σ represents the sigmoid activation function; and S_t signifies the internal (hidden) state. The functions z_t, r_t, and $\check{S}_t$ correspond gets to the update, reset, and third gate, respectively.

3.4. One-dimensional convolution neural network (1D-CNN)

A CNN falls within the class of deep neural networks capable of automatically extracting features and generating informative representations from time-series data, eliminating the need for manual feature engineering (Markova, 2022). It serves as a modification of the 2D-CNN architecture. The 1D-CNN architecture introduces two distinct layer types:

1. 'CNN-layers' where 1D convolutions, activation functions, and sub-sampling (pooling) operations occur.
2. Fully-connected (dense) layers, akin to those found in a standard Multi-Layer Perceptron (MLP), hence referred to as 'MLP-layers.'

In our case, the computational process of a 1D-CNN can be represented as demonstrated in Eq. (4).

$$\begin{gathered} Z = Conv1D(X, Y) + b \\ Z_i = \sum_{j=0}^{F-1}(x_{i+j} \cdot w_j) + b \\ A_i = activation(Z_i) \\ MaxPooling(A_i, i) = max\ x_0 \leq w\ A_i, i + j \times s \end{gathered} \tag{4}$$

The input sequence, denoted as X is defined with dimensions $N \times C \times L$ where N is the batch size, C is represent the number of channels (features) and L is the number of sequence; In 1D convolution, a filter slides over the input sequence, performing convolution at each position i, yielding the output Z_i is the value of the output at position i, x_{i+j} represents the input values at positions $i + j$ (for j from 0 to $F - 1$), W_j represents the weights of the filter at position j, b_i represents the bias term for the output at position i; Following it is applied an activation function element-wise to the output of the convolution operation. $Max\ Pooling$ operation with a window size W and stride S in a 1D setting is represented in above x_i, i represents the input value at position i in channel c.

4. DEVELOPED MODEL: ENC&DEC TRANSFORMER

The Env&Dec Transformer follows the recent competitive advancement in deep learning introduced by google researchers (Vaswani et al., 2017), aligns with the majority of cutting-edge neural sequence models, featuring an encoder-decoder architecture. Illustrated in Figure 1, both the encoder and decoder comprise numerous identical blocks. In detail, each encoder block includes a multi-head self-attention module and a position-wise feed-forward network, whereas each decoder block integrates cross-attention mechanisms between the multi-head self-attention module and the position-wise feed-forward network. Within the input, w denotes the dimension of the look-back window, while k represents the count of future prediction steps. The decoder segment employs a masked attention module, and within the decoder, a cross attention mechanism is employed to choose output of encoder layer, which serve as the feature vector.

In Enc&Dec, attention is a mechanism that assigns weights to each word in a sentence based on its importance. In this context, we have a feature vector that includes query, key, and value, which conceptually are used to perform this operation. The query represents the sought-after information, while the key signifies the context or reference, and the value denotes the content under scrutiny. Multiplying the query and key yields attention scores, which are subsequently utilized to compute the weighted sum of the values (Haugsdal et al., 2023). This weighted sum, in turn, is employed to compute the model's output.

The model's recursive definition can be established by denoting X_i as the output of the i-th block, as depicted in Eq. (5-7).

$$\begin{gathered} X_i(X_i-1)=FF_i(SA_i(X_i-1)) \\ SA_i(X)=LayerNorm(X+SelfAttention_i(X)) \\ FF_i(X)=LayerNorm(X+FeedForward_i(X)) \end{gathered} \tag{5}$$

The matrix X_i has dimensions $L \times d_{model}$, where L represents the sequence or time dimension and d_{model} represents the feature dimension. X_0 acts as the base case of the recursion, representing the initial input to the model. The hyperparameter N, determines the number of blocks, and subsequently the final output of the model denoted as, X_n. Before delving into self-attention, it's crucial to establish multi-head attention.

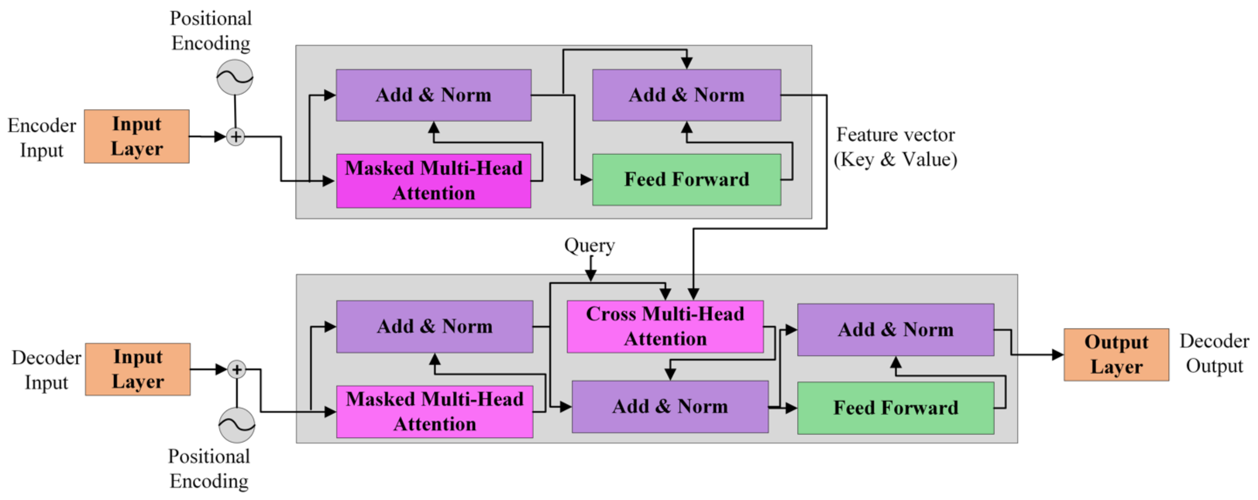

Figure 1. *Structure of Enc & Dec Transformer-based prediction model*

Multi-head attention aggregates multiple attention heads, each equipped with its unique set of learnable weights, enabling them to perform simultaneous computations.

Self-attention arises as a particular case of multi-head attention, wherein the keys, queries, and values are identical.

$$\begin{gathered} SelfAttention(X)=MHA(X,X,X) \\ MHA(Q,K,V)=Concat(head_1,\dots,head_h)W_o \\ head_j=Attention(QW_Q^{(j)},KW_K^{(j)},VW_V^{(j)}) \\ Attention(Q,K,V)=softmax(\frac{QK^T}{\sqrt{d_{model}}}+M)V \end{gathered} \tag{6}$$

where, h represents the number of attention heads, and d_{heads} is calculated as d_{model} divided by the number of heads ($d_{head} = \frac{d_{model}}{h}$). The learnable weight matrices $W_Q^{(j)}, W_K^{(j)}$and $W_V^{(j)}$ have a shape of $d_{model} \times d_{head}$, while W_o is a learnable weight matrix with a shape of $d_{model} \times d_{model}$. W_o is responsible for combining the outputs from each attention head. M denotes an upper triangular masking matrix that prevents the model from attending to future time steps.

The feed-forward layer functions in a point-wise manner, concentrating exclusively on information from the present time step, akin to a 1-D convolution. It consists of two linear transformations interspersed with a ReLU activation function, representing non-linearity.

$$FeedForward(X) = ReLU(XW_1 + b_1)W_2 + b_2 \tag{7}$$

where W_1 and W_2 denote learnable weight matrices with dimensions $d_{model} \times d_{ff}$ and $d_{ff} \times d_{model}$ respectively. Similarly, b_1 and b_2 represent learnable bias vectors with dimensions d_{ff} and d_{model}.

In Figure 1, we can observe the composition of an encoder and decoder block. During training, the decoder receives a feature vector (key and $value$) from the encoder output and utilizes it to predict the future point. The $query$ is involved in the calculation of the masked attention.

Unlike other recurrent neural network that process sequence tokens sequentially, self-attention favors parallel computation over sequential operations. It is important to note that self-attention itself doesn't maintain sequence order. To retain token order information, the prevalent approach involves providing the model with additional inputs known as positional encodings. These encodings, associated with each token, can be either learned or pre-defined. In this study, we utilized a straightforward approach: fixed positional encodings utilizing sine and cosine functions (Vaswani et al., 2017), as per Eq. (8).

$$PE_{(pos\,,2i)} = \sin\left(\frac{pos}{1000^{\frac{2i}{d_{model}}}}\right)$$
$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{1000^{\frac{2i}{d_{model}}}}\right) \tag{8}$$

4.1. Encoder-only transformer

The encoder-only is the encoder block of the Enc&Dec Transformer. According to Figure 1, it has a multi-head module without masking operation, a residual connection with normalization, and a feed-forward layer. To serve the prediction purpose by the encoder layer, we feed the output of the encoder layer to an output layer. The mathematical computations and positional encoding are the same as Eq. (5-8), respectively.

4.2. Decoder-only transformer

The decoder-only block is a component of the Enc&Dec Transformer, distinct from the decoder in the standard transformer. In contrast to the transformer decoder, which obtains the key and value from the encoder block, the decoder-only block generates a feature vector through its masked multi-head attention layer. Illustrated in Figure 1, the decoder comprises two primary elements: initially, windowed data is fed into the model's input layer, followed by the application of fixed positional embedding. Next, the raw positional encoded matrix is fed into masked multi-head attention mechanisms. After applying self-attention (according to Eq. (6)) on each split matrix, the data is further separated into three matrices: Query, Key, and Value. Layer normalization is then implemented, followed by feeding the output of the initial masked multi-head attention, which comprises the three matrices, into the subsequent non-masked multi-head attention. The mathematical calculations and positional encoding remain consistent with Eq. (5-8), respectively.

5. EXPERIMENTAL EVALUATION

5.1. Dataset

PJM Interconnection LLC (PJM) functions as a regional transmission organization within the United States, operating as a component of the Eastern Interconnection grid and managing an electric transmission system that serves multiple cities across the country. It produces their power sensor devices generated hourly energy consumption data as benchmark datasets (Mulla, 2019). We downloaded three univariate publicly available datasets from PJM's website, including PJM, DUQ, and EKPC, with large, medium and small size. Their sizes are 145,392; 119,088; and 45,336 observations respectively, and they are collected hourly in megawatts (MW).

5.2. Hyperparameter optimization

Hyperparameter tuning stands as one of the most arduous tasks within machine learning projects. As the complexity of deep learning methods continues to surge in popularity, the demand for an efficient automatic hyperparameter tuning framework has escalated significantly (Akiba et al., 2019). Numerous techniques exist to optimize hyperparameters in deep learning. While previous literature emphasizes grid search and random search as primary methods for machine learning models, recently Optuna has become very popular hyperparameter tuning technique in machine learning problems. Grid search, though widely used, can become time-consuming and computationally expensive, particularly with a large number of hyperparameters or potential values. Optuna distinguishes itself as a more advanced approach, leveraging Bayesian optimization to efficiently explore and pinpoint the best set of hyperparameters (Shekhar et al., 2021).

In this study, we employed the Optuna to identify optimal parameters suitable for our dataset and model complexity. Specifically, we configured Optuna to conduct 100 trials with 20 internal epochs. Parameters were set within defined ranges: the number of layers varied from 1 to 10, hidden layers spanned from 16 to 2048, drop probabilities ranged between 0 and 1, and learning rates fell within the range of 0 to 1, consistent across all five models. However, certain models necessitated additional hyperparameters; for instance, in the case of the 1D-CNN, we specified the $kernel\ size$ between $3-7$ and $padding$ as $(kernel\_size\ -\ 1)\ //\ 2$. Similarly, for the transformers, the required number of $heads$ set from 1 to 8, and the number of dimensions from 32 to 1024. Additionally, we manually set the activation function as linear, the number of $epochs$, $batch\ size$ and $optimizer$ in Table 3.

As seen in Table 3, we summarized the optimal result for transformer-based networks, including Enc&Dec, Encoder-only, as well as Decoder-only architectures, due to their similarity in data flow within modules and the mechanism of assigning weight and bias values, we utilize the same parameters for training for our proposed models. The algorithm recommended the best range of values for each parameter based on the outcomes from Optuna trials.

Table 3. *Hyperparameter settings through Optuna*

Model	Hyperparameters								
	# of layers	# of hidden layers	Drop rate	Learning rate	Activation function	Kernel size	Padding	# of heads	# of dimension
RNN	6	1710	0.0055353	0.0001863781	tanh				
LSTM	3	2048	0.2075485	0.0002399395	tanh				
GRU	8	74	0.0206139	8.00205811e-05	tanh				
1D-CNN		1995	0.1614444	1.64786113e-05	ReLU	5	2		
Transformers	1	1998	0.0009843	1.04834920e-05				6	501
Optimizer		Adam		Batch size			32		

5.3. Experimental setup

Deep learning models utilize past data to identify a functional connection between input characteristics and the forthcoming values of the target variable. These trained models offer forecasts for the target variable in subsequent time periods. In a time-series $\{x_1,\ x_2,\ x_3 \dots x_T\}$, where x_t denotes a set of m input features recorded at time t, the aim is to construct a model that predicts a target variable y_{t+k} at a future time point $t+k$, utilizing insights derived from historical data up to time $t-1$ ($\{\dots,\ x_{t-2},\ x_{t-1}\}$). To ensure consistency in the model's input, we adopted a fixed-length sliding time window of magnitude w, as depicted in Figure 2. In our case, the look-back size w is set to 168 hours, equivalent to one week in hourly collected data, while the look-ahead size k is set to 48 hours to forecast two days ahead based on the given w size.

The mathematical representation of the functional relationship learned by machine learning models can be illustrated through Eq. (9), as shown.

$$\hat{y}_{t+k} = f_k(x_{t-w}, \dots, x_{t-1}, y_{t-w}, \dots, y_{t-1}) \tag{9}$$

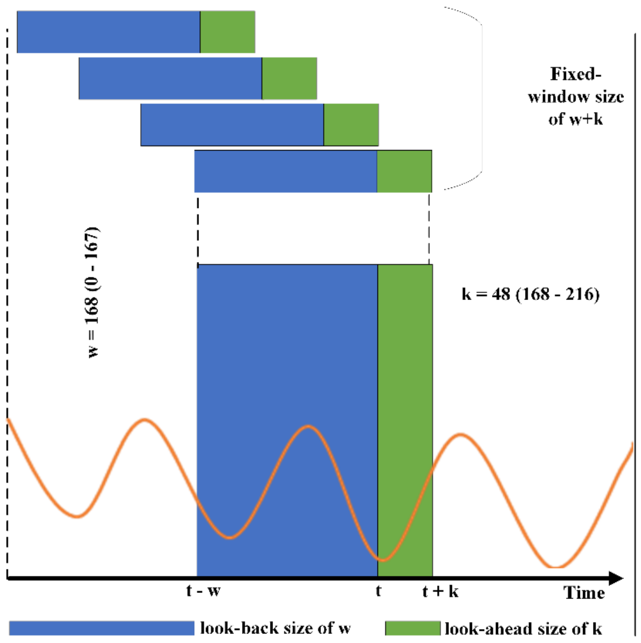

Figure 2. *Sliding window forecasting next 48 observation based on 168 data points*

where $\hat{y}_{t+k}$ represents the forecasted target variable for time $t + k$; k represents the future time duration for which the target variable is predicted; $y_{t-w}, \dots, y_{t-1}$ represent the observed target values spanning from time $t - w$ to $t - 1$; $x_{t-w}, \dots, x_{t-1}$represent the vector of m observed input features observed from time $t - w$ to $t - 1$. The function f_k denotes the learned function by deep learning models with a step size of 1. In our context, m denotes the number of input features, defaulted to 1 as it is univariate, and w indicates the size of the window utilized as input. As well as we utilized the StandardScaler to transform our data, normalize it within the range of 0 to 1, as illustrated in Eq. (10).

$$z = \frac{x-\mu}{\sigma} \tag{10}$$

Leveraging the GPU for accelerated computing with high-dimensional matrices, we implemented our data and models using PyTorch. For coding, we utilized Python 3.12, ensuring compatibility with various essential libraries such as NumPy, PyTorch, Matplotlib, among others. To maintain a robust evaluation process, we partitioned our dataset into three segments. We trained all our models on the initial 80% of the dataset, validated their performance on the subsequent 10%, and finally tested each model on the last 10%—ensuring that the test data remained unseen during training and validation. To optimize computational efficiency, we organized data input to the models in batches according to specifications outlined in Table 3 hyperparameter setting. Moreover, in order to prevent overfitting and underfitting, we implemented regularization techniques. Specifically, we set the patience parameter to 3 and applied early stopping if the validation metric did not improve for 20 or more epochs.

5.4. Evaluation metrics

To evaluate the performance of proposed models, we utilized the MAE (Mean Absolute Error), MSE (Mean Squared Error), and RMSE (Root Mean Squared Error) metrics according to Eq. (11-13), respectively.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \tilde{y}_i| \tag{11}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \tilde{y}_i)^2 \tag{12}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2}{N-P}} \tag{13}$$

where, y_i indicate individual observed values in the dataset, $\tilde{y}_i$ predicted values corresponding to y_i and μ the mean of the observed values.

6. EXPERIMENTAL RESULTS AND DISCUSSION

The training and testing losses were calculated based on the epochs to identify potential overfitting, we utilize the MSE as our chosen loss function computation for both training and testing. In Table 4, a summary of our model results is presented and organized based on high, medium and low volume datasets. Notably, the MSE values are observed to be lower than both MAE and RMSE, influencing our decision-making process.

This table highlights that RNN consistently demonstrates a high MSE score across all evaluated univariate datasets. LSTM, a variant of RNN tailored to tackle the vanishing gradient problem, exhibits a lower MSE score compared to RNN. Moreover, the GRU, addressing issues inherent in both RNN and LSTM, achieves the best results with a minimum MSE of 0.24 when compared with the MSE of 0.36 for RNN and 0.30 for LSTM. While 1D-CNN, leveraging automatic feature extraction, has a minimum MSE of 0.15 in univariate time-series prediction, it outperforms recurrent-based models, including RNN, LSTM, and GRU, as well as Transformer-based models, including Encoder-only and Decoder-only models.

Table 4. *The performance metrics for experimented models across three different sizes of datasets*

Dataset	**Data size**	**Model**	**Evaluation Metrics**		
			MAE	MSE	RMSE
PJME_hourly	Large	RNN	0.49	0.40	0.63
		LSTM	0.43	0.31	0.56
		GRU	0.36	0.24	0.47
		1D-CNN	0.27	0.14	0.38
		Encoder-only Transformer	0.40	0.27	0.52
		Decoder-only Transformer	0.33	0.19	0.44
		Enc&Dec Transformer	0.06	0.006	0.08
DUQ_hourly	Medium	RNN	0.46	0.36	0.60
		LSTM	0.41	0.30	0.55
		GRU	0.38	0.24	0.49
		1D-CNN	0.28	0.15	0.39
		Encoder-only Transformer	0.38	0.26	0.51
		Decoder-only Transformer	0.31	0.19	0.43
		Enc&Dec Transformer	0.07	0.008	0.09
EKPC_hourly	Small	RNN	0.55	0.61	0.78
		LSTM	0.52	0.54	0.73
		GRU	0.53	0.56	0.75
		1D-CNN	0.40	0.34	0.58
		Encoder-only Transformer	0.54	0.58	0.76
		Decoder-only Transformer	0.47	0.45	0.67
		Enc&Dec Transformer	0.11	0.020	0.14

Utilizing the Encoder-only and Decoder-only models with minimum MSE of 0.26 and 0.19 respectively in energy prediction outperforms RNN, LSTM, and GRU. However, when comparing these results with those in Table 4, the performance falls short compared to 1D-CNN and Enc & Dec Transformer models, which is not satisfactory. Decoder-only outperforms Encoder-only due to its ability to perform masking operations for handling data and employing more attention layers. In another aspect, the Enc & Dec Transformer model, composed of both encoder and decoder, outperforms all models evaluated in this study for univariate time-series by achieving a minimum MSE of 0.006 for small, 0.008 for medium, and 0.020 for large datasets in predicting energy consumption.

Figure 3 illustrates the performance of RNN, LSTM, GRU, 1D-CNN, Encoder-only, Decoder-only, and Enc & Dec Transformer across three large, medium and small size datasets. The results indicate a clear trend: as our dataset size increases, the corresponding loss decreases. The size difference between the DUQ and PJME datasets is not substantial. However, the variance between the EKPC (small) and PJME (large) dataset is extremely high. This leads to more noticeable fluctuations in results across all models. This observation underscores the conventional wisdom that larger datasets often lead to enhanced model generalization. Across all dataset sizes, the Enc & Dec Transformer consistently demonstrates superior performance, yielding minimal loss values. This trend suggests its remarkable capability to capture long-range dependencies and complex patterns within the data. Consequently, Enc & Dec Transformers exhibit superior performance compared to recurrent and convolutional architectures, especially when dealing with increased data size. Among the recurrent models (RNNs, LSTMs, and GRUs), there's comparable performance observed, even though with slight variations depending on the dataset. This consistency indicates their stable capacity to model sequential relationships. However, 1D-CNNs exhibit a noticeable performance increase with low MAE, MSE and RMSE scores, particularly on the smallest dataset (EKPC hourly). This suggests that these models might require a larger dataset to effectively learn and harness meaningful convolutional filters. In essence, 1D-CNNs might face challenges and show diminished performance with smaller datasets, similar to other deep learning algorithms. The performance of Encoder-only remains the same in both large and medium-size datasets, but the loss is going high in the small dataset. However, the Decoder-only outperforms in the medium-size dataset. Overall, our results suggest that for training such deep learning algorithms, we must have enough data to reach the best result.

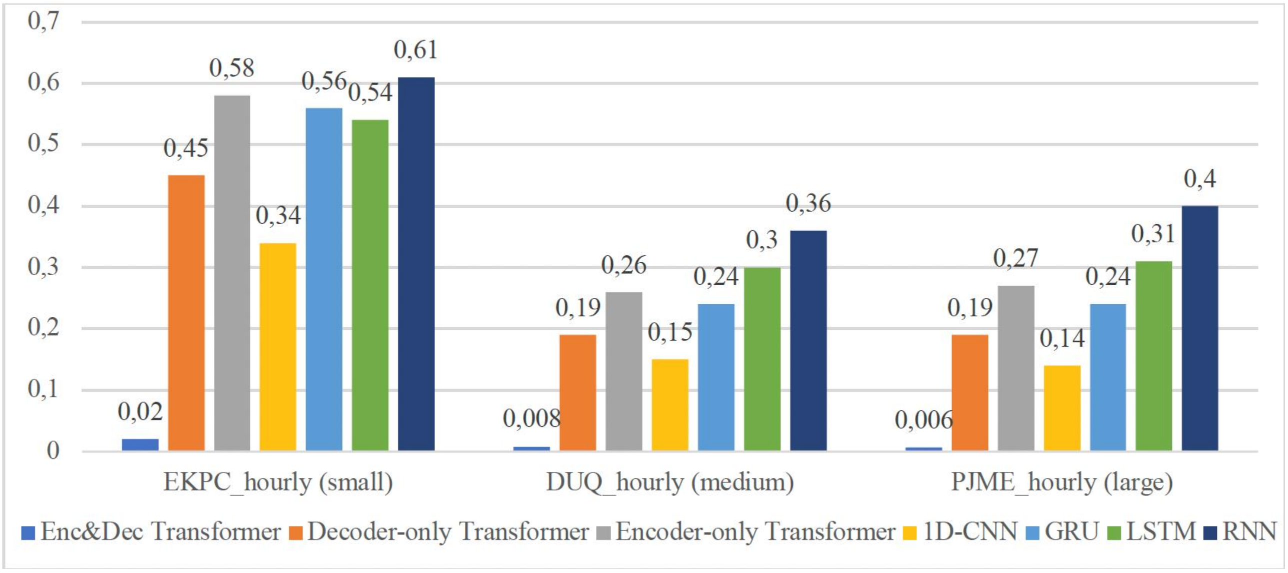

Figure 3. *Model performance in terms of MSE scores with varying sizes of datasets*

The existing literature studies we summarized in Table 2 categorized the employed Enc&Dec Transformer models in energy consumption and other domains (Mat et al. 2019, Wu et al. 2020, Li et al. 2019, Lim et al. 2020). In the studies, researchers also compared their findings with those of state-of-the-art machine learning and deep learning models. For instance, the findings of Li et al. (2019) showed that the proposed Enc&Dec Transformer models outperformed ARIMA, ETS, TRMF, DeepAR, DeepState, and even LSTM. The efficiency of the proposed Enc&Dec Transformer models by Lim et al. (2020) was demonstrated by comparing their proposed methods with state-of-the-art DL methods as well as hybrid techniques, including ConvTrans and Seq2Seq, and finding them to be superior. Since the dataset in this study differs from the studies, we could not directly compare our results with those in the literature. Nevertheless, our study, consistent with these studies, demonstrates that our Enc&Dec Transformer outperforms DL methods in predicting energy consumption. Furthermore, we distinguish our study by comparing the results of the Enc&Dec Transformer with those of Encoder-only and Decoder-only Transformer models.

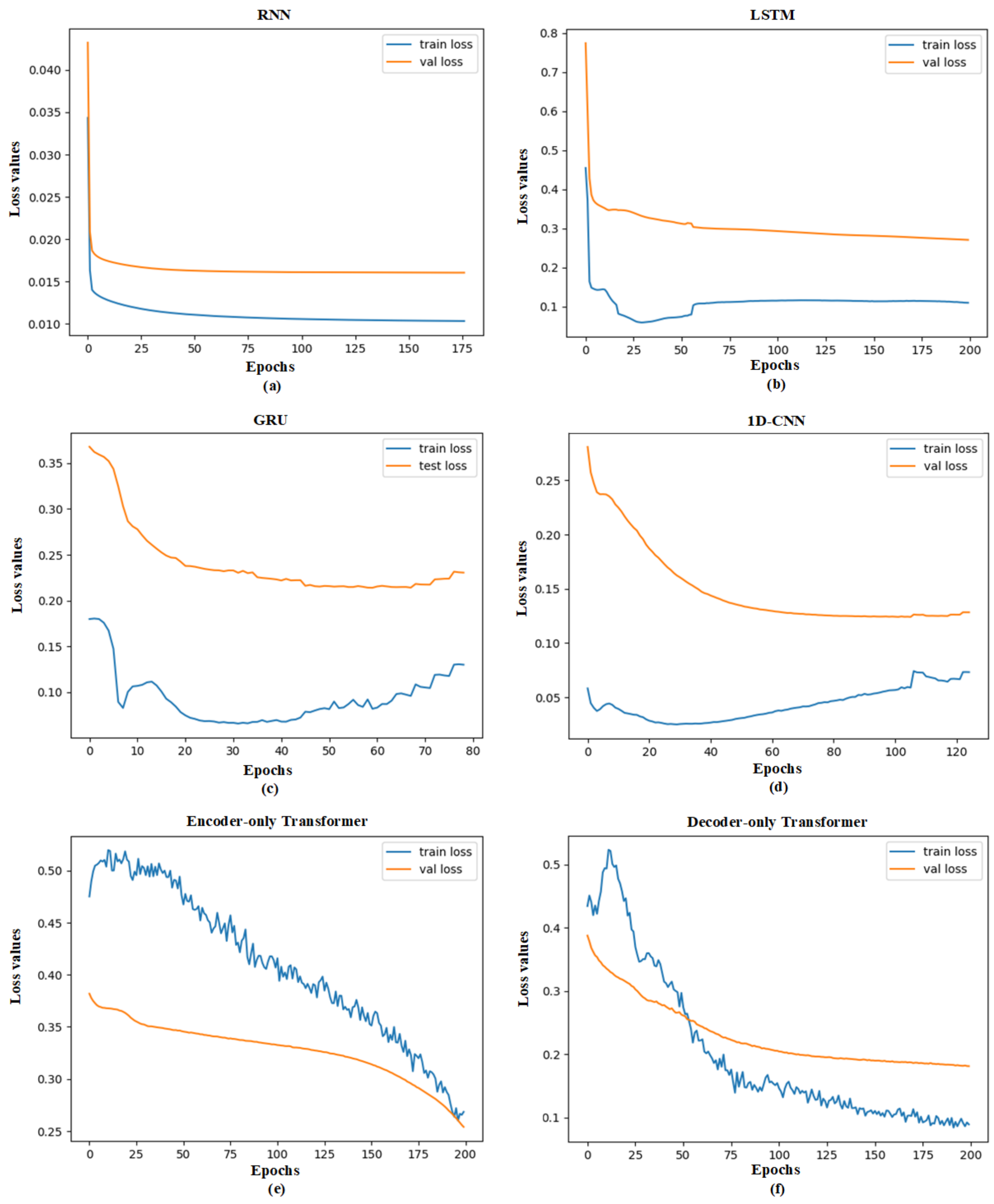

Figure 4. *Training and validation losses for **(a)** RNN, **(b)** LSTM, **(c)** GRU, **(d)** 1D-CNN, **(e)** Encoder-only Transformer, and **(f)** Decoder-only Transformer models using the medium (DUQ) dataset in terms of MSE score*

In Figure 4, the graphs depict the train losses vs. validation losses of RNN, LSTM, GRU, 1D-CNN, Encoder-only, and Decoder-only, respectively evaluated only for medium (DUQ_hourly) dataset. The figures show the models' generalizing performance, indicating overfitting, underfitting, and the fitting capability of each model. Figures 4 (a), (b) and (c) depict the train vs. validation losses of RNN, LSTM and GRU networks. The training loss remains high throughout the training process, never dipping below the validation loss. This suggests that the model is not effectively learning from the training data. Additionally, the gap between the training and validation loss remains significant throughout the training process, indicating that the model is not generalizing well to unseen data. Neither the training nor the validation loss shows a significant decrease over time, suggesting that the model is not making progress during training. These are all indicators of underfitting, implying that the models are not complex enough to capture the underlying patterns in the data. As a result, the model is unable to make accurate predictions on new data. In Figure 4(d) graph displays the 1D-CNN training vs. validation loss. This graph shows that the training loss starts off much higher than the validation loss, but then steadily decreases and crosses over the validation loss around epoch 60. This is a positive sign, suggesting that the model is learning from the training data and generalizing well to unseen data. However, after the crossover, the training loss continues to decrease while the validation loss starts to increase slightly. In Figures 4(e) and Figure 4(f) illustrate the training losses and validation losses of Decoder-only and Encoder-only, attention-based neural networks. The graphs show that the training loss is initially higher than the validation loss, but it quickly decreases and crosses over the validation loss around epoch 50. This is a positive sign, suggesting that the model is learning from the training data and generalizing well to unseen data. The validation loss also starts to decrease after epoch 50, but it remains slightly higher than the training loss throughout the rest of the training process. This could be an indication of underfitting. The Encoder-only graph displays the training loss and validation loss over the course of training epochs. The validation loss decreases initially, indicating that the model is generalizing well to unseen data. There is no indication of overfitting in this graph, as the validation loss never increases after decreasing, and the gap between the training and validation loss remains relatively constant throughout the training process. This could be an indication of overfitting.

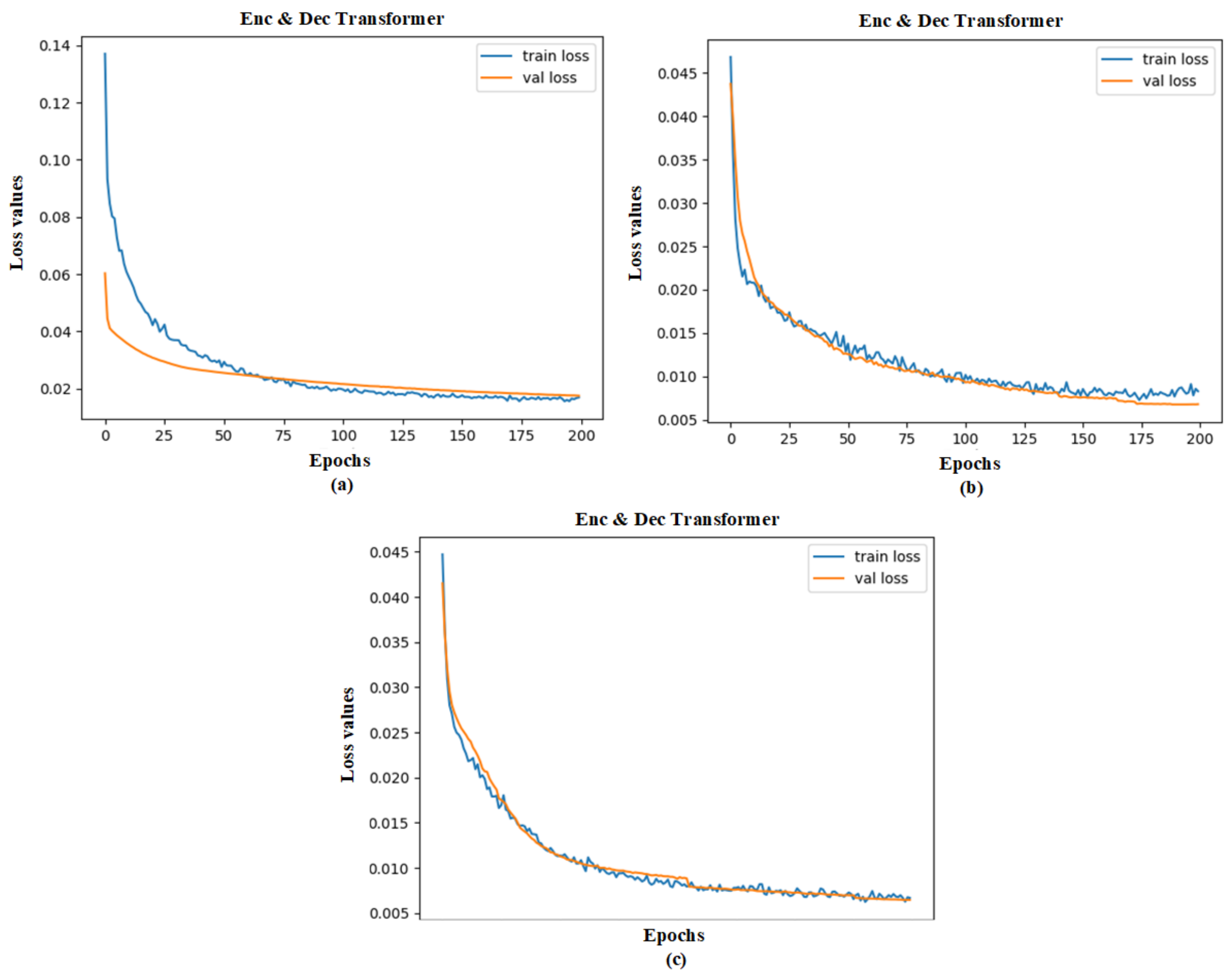

Figure 5. *Training and validation losses of Enc&Dec Transformer model in terms on MSE score using **(a)** small (EKPC), **(b)** medium (DUQ) and **(c)** large (PJME) datasets*

Figure 5 shows the training and validation loss curves for a time series prediction model trained with an Enc&Dec Transformer on small, medium, and large datasets. In Figure 5(a), for the small dataset, the training loss is considerably higher before epoch 50. However, after epoch 60, the validation loss decreases while the training loss increases; overall, the difference is not very significant. In Figure 5(b), for the medium dataset, the training loss curve consistently remains lower than the validation loss curve, indicating a positive sign. This suggests that the model effectively learns the training data without overfitting. The validation loss curve initially decreases rapidly but then plateaus around epoch 25. Figure 5(c) demonstrates the performance of the Enc&Dec Transformer model on a large dataset. Initially, the validation loss is lower than the training loss, but this trend is not consistent throughout. Again, after epoch 100, the validation loss starts to decrease, indicating that the model is beginning to generalize well to unseen data.

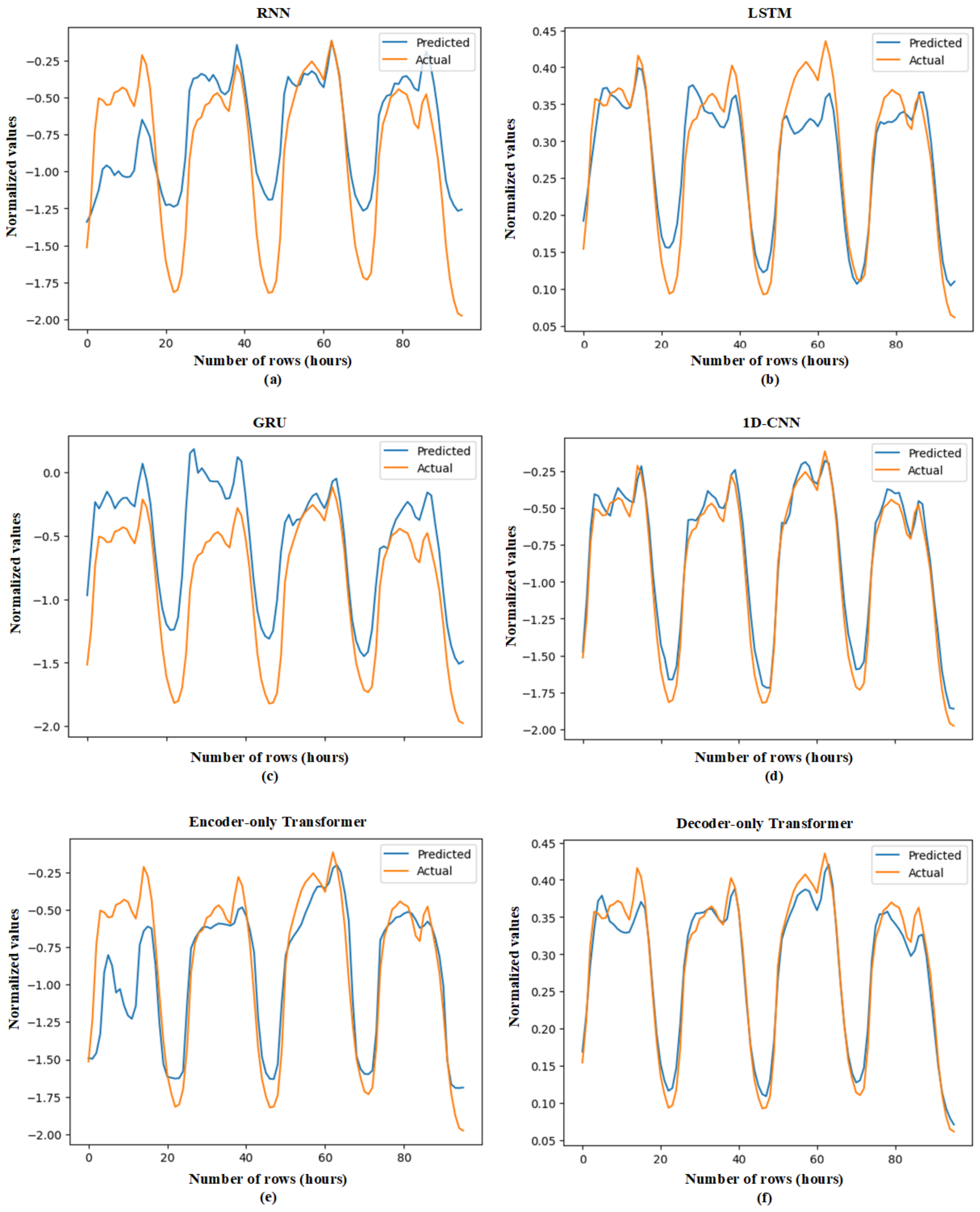

Figure 6. *Predicted and actual values of for **(a)** RNN, **(b)** LSTM, **(c)** GRU, **(d)** 1D-CNN, **(e)** Encoder-only Transformer, and **(f)** Decoder-only Transformer models using medium (DUQ) dataset in terms of MSE score*

Figure 6 illustrates the ground truth data for RNN, LSTM, GRU, 1D-CNN, Encoder-only, and Decoder-only models, plotting the deviation between predicted and actual values, specifically for the medium dataset (DUQ_hourly), with the aim of comprehensively assessing model performance. Across all graphs, the blue line corresponds to predicted values, whereas the orange line represents actual values. For prediction, a fixed-length window size is employed. As demonstrated in Figure 2, a fixed look-back window size of 168 hours is utilized to forecast a look-ahead size of 48 hours during training. To validate our models, we conducted testing on the last 10% of unseen data, predicting the subsequent four days ahead. This testing involved targeting and comparing 96 hours of data for predictions and the evaluation of the model. Each graph is plotted based on the MSE loss scores, and the comparison is specifically conducted on the medium dataset. The result of Figure 6(a) confirms that the RNN model exhibits a significant difference between predicted and actual values, primarily due to the vanishing gradient problem, impacting its ability to memorize previous data effectively. Comparatively, LSTM and GRU in Figures 6(b) and Figure 6(c), respectively, demonstrate better predictive capabilities than RNN. Particularly in our proposed domain data, GRU outperforms both LSTM and RNN. Additionally, both the 1D-CNN and Decoder-only in Figures 6(d) and Figure 6(f) surpass the performance of recurrent-based models. They predict values very close to the actual ones, but there are instances, particularly between 58 and 60, where the maximum wavelength incurs higher losses. Beyond point 60, the predicted values align more closely with the actual ones. However, in the case of 1D-CNN, the minimum point of the wavelength exhibits better performance. In Figure 6(e), where the Encoder-only is plotted, the model seems to be learning well and generalizing decently, but there might be room for improvement.

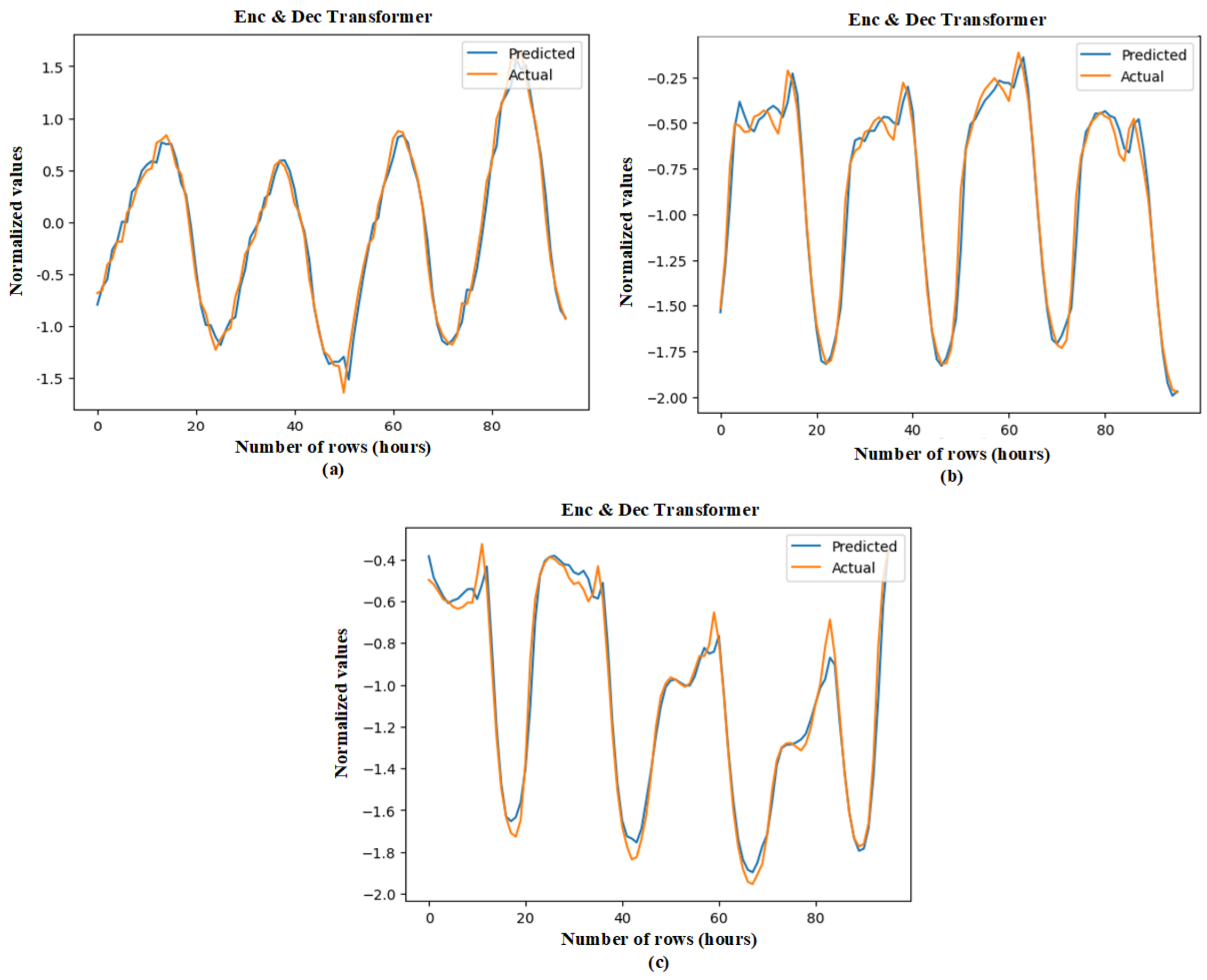

Figure 7. *Predicted and actual values of Enc&Dec Transformer model in terms on MSE score using* ***(a)*** *small (EKPC),* ***(b)*** *medium (DUQ,), and* ***(c)*** *large (PJME) datasets*

Figure 7 depicts the predicted and actual values of the first 96 hours ahead for small, medium, and large datasets, utilizing our proposed approach, Enc&Dec Transformer. In Figure 7(a), the difference between predicted values and actual values for the small size of data indicates in the initial stage a lesser height. In Figure 7(b), the points for the medium size of data in the scatter plot are generally close to the blue line, indicating that the model is performing well in predicting the actual values. Many points cluster around the ideal line, suggesting that the model is generally accurate. There are very few outliers, as the model points out. This implies that the model's predictions were very close to the actual values for the majority of the Enc&Dec Transformer. In Figure 7(c), it shows the growth trend for the large size of data, in which the difference between actual and predicted values is very close; at sometimes, they are the same. There is no trend of underestimation at high values; in fact, there is a slight trend of overestimation at high values, with the red circles positioned slightly above the blue line on the higher values of the x-axis. The graph exhibits that in the case of high volume of data Enc&Dec Transformer is good at predicting the actual value.

7. CONCLUSION

In this study, we propose utilizing the Enc & Dec Transformer having encoder-decoder architecture to forecast univariate time-series in IoT device-generated energy consumption data. The performance of our proposed models is compared with recurrent-based models, including RNN, LSTM, GRU, convolutional-based 1D-CNN, as well as self-attention-based Encoder-only and Decoder-only. To determine optimal hyperparameters, we utilized Optuna, optimizing network complexity efficiently with minimal resource and memory usage. We evaluated the performance of our proposed model alongside other deep learning models on datasets of varying sizes: small, medium, and large. Our experimental results indicate that all models perform admirably well on large time-series datasets. Among the recurrent-based models, GRU demonstrated superior performance compared to RNN and LSTM, particularly in understanding the relationships between features within the univariate energy domain data. The 1D-CNN model, leveraging automatic feature engineering properties, outperformed all models in the case of medium-sized data. For the comparison of self-attention-based models, Decoder-only outperforms Encoder-only as well as recurrent-based algorithms. Our proposed Enc & Dec Transformer model showcased superior performance on small, medium, and very large datasets. As dataset sizes increased, the performance gaps widened, highlighting the transformer's advantage in handling large-scale data. However, it is essential to note that while transformers achieve higher scores, they also incur higher computational expenses during training. Depending on specific applications and available computational resources, separated Transformer models like Encoder-only and Decoder-only transformers or simpler model like RNNs or 1D-CNNs might be more viable, especially for smaller datasets. Our results show that attention mechanisms, especially those composed of Enc & Dec Transformer which composed of encoder and decoder layers, represent the state-of-the-art in time-series prediction. This interpretation method is inspired by human recognition and allows neural networks to focus on how various inputs influence outputs at each step of inference in the model development process, providing a quantitative explanation of these influences. One of the important components that can improve the efficiency of Encoder-only Transformer, Decoder-only Transformer, and Enc & Dec Transformer models is positional encoding. Attention modules may not understand the exact location of each data point, which is why positional encodings are utilized to represent the location of each time-dependent data point in a precise manner. In this study, we trained models using one of the traditional positional encodings (absolute). Therefore, it is suggested to evaluate the performance of transformers by employing different types of positional encoding to demonstrate efficiency in energy consumption prediction.

AUTHOR CONTRIBUTIONS

Methodology and writing-reviewing, A.A. and F.Y.O; editing, F.Y.O.; conceptualization and software, A.A. All authors have read and legally accepted the final version of the article published in the journal.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

REFERENCES

Adhikari, R., & Agrawal, R.K. (2013). An Introductory Study on Time Series Modeling and Forecasting. *ArXiv, abs/1302.6613*. https://doi.org/10.48550/arXiv.1302.6613

Afanasieva, T., & Platov, P. (2019). The Study of Recurrent Neuron Networks based on GRU and LSTM in Time Series Forecasting. *In ITISE 2019. Proceedings of papers. Vol 1 (pp. 12). Granada, Spain: International Conference on Time Series and Forecasting.* https://itise.ugr.es/ITISE2019_Vol1.pdf

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. *In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 2623–2631). *Association for Computing Machinery*. https://doi.org/10.1145/3292500.3330701

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). Time Series Analysis: Forecasting and Control (5th ed.). *Hoboken, NJ: John Wiley & Sons Inc*. https://doi.org/10.1111/jtsa.12194

Cao, L. (2003). Support vector machines experts for time series forecasting. *Neurocomputing*, 51, 321-339. https://doi.org/10.1016/S0925-2312(02)00577-5

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-End Object Detection with Transformers. *CoRR, abs/2005.12872*. https://doi.org/10.48550/arXiv.2005.12872

Cochrane, J. H. (1997). Time Series for Macroeconomics and Finance. *Graduate School of Business, University of Chicago.* Retrieved from http://www.fsb.miamioh.edu/lij14/672_notes_Cochrane

Coulibaly, P., & Baldwin, C. K. (2005). Nonstationary hydrological time series forecasting using nonlinear dynamic methods. *Journal of Hydrology*, 307(1–4), 164-174. https://doi.org/10.1016/j.jhydrol.2004.10.008

Das, A., Kong, W., Sen, R., & Zhou, Y. (2024). A decoder-only foundation model for time-series forecasting. *ICML*. https://doi.org/10.48550/arXiv.2310.10688

Haugsdal, E., Aune, E., & Ruocco, M. (2023). Persistence Initialization: a novel adaptation of the Transformer architecture for time-series prediction. *Applied Intelligence*, 53, 26781–26796. https://doi.org/10.1007/s10489-023-04927-4

Hipel, K. W., & McLeod, I. (1994). Time series modelling of water resources and environmental systems. *In Proceedings of the International Conference on Systems, Man and Cybernetics* (pp. 1-6). https://doi.org/10.1016/s0167-5648(08)x7026-1

Hu, C., Sun, Z., Li, C., Zhang, Y., & Xing, C. (2023). Survey of time-series data generation in IoT. *Sensors*, 23(15), 6976. https://doi.org/10.3390/s23156976

IoT Analytics (2023). state of IoT 2023: number of connected IoT devices growing 16% to 16.7 billion *globally*. https://iot-analytics.com/number-connected-iot

Javaid N., Jul 12, 2019. Implementing an RNN from scratch in Python: towards data science. https://towardsdatascience.com/recurrent-neural-networks-rnns-3f06d7653a85

Lara-Benítez, P., Gallego-Ledesma, L., Carranza-García, M., & Luna-Romera, J. M. (2021). Evaluation of the Transformer Architecture for Univariate Time Series Forecasting. *In E. Alba et al. (Eds.), Advances in Artificial Intelligence. CAEPIA 2021.* Lecture Notes in Computer Science (Vol. 12882). *Springer*, Cham. https://doi.org/10.1007/978-3-030-85713-4_11

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. *CoRR, abs/1907.00235*. https://doi.org/10.48550/arXiv.1907.00235

Lim, B., Arık, S. Ö., Loeff, N., & Pfister, T. (2021). Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4), 1748–1764. https://doi.org/10.1016/j.ijforecast.2021.03.012

Ma, J., Shou, Z., Zareian, A., Mansour, H., Vetro, A., & Chang, S. (2019). CDSA: Cross-Dimensional Self-Attention for Multivariate, Geo-tagged Time Series Imputation. *ArXiv, abs/1905.09904*. https://doi.org/10.48550/arXiv.1905.09904

Markova, M. (2022). Convolutional neural networks for forex time series forecasting. *AIP Conference Proceedings*, 2459(1), 030024. https://doi.org/10.1063/5.0083533

Masum, S., Liu, Y., & Chiverton, J. (2018). Multi-step Time Series Forecasting of Electric Load Using Machine Learning Models. In L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, & J. M. Zurada (Eds.), *Artificial Intelligence and Soft Computing* (pp. 148-159). Springer International Publishing. https://doi.org/10.1007/978-3-319-91253-0_15

Mo, Y., Wu, Q., Li, X., et al. (2021). Remaining useful life estimation via transformer encoder enhanced by a gated convolutional unit. *Journal of Intelligent Manufacturing*, 32, 1997–2006. https://doi.org/10.1007/s10845-021-01750-x

Mulla, R. (2019). Hourly Energy Consumption. PJM Interconnection LLC in *Kaggle*.

Nie, H., Liu, G., Liu, X., & Wang, Y. (2012). Hybrid of ARIMA and SVMs for Short-Term Load Forecasting. *Energy Procedia*, 16, 1455-1460. https://doi.org/10.1016/j.egypro.2012.01.229

Nor, M. E., Mohd Safuan, H., Md Shab, N. F., Asrul, M., Abdullah, A., Mohamad, N. A. I., & Lee, M. H. (2017). Neural network versus classical time series forecasting models. *AIP Conference Proceedings*, 1842(1), 030027. https://doi.org/10.1063/1.4982865

Ogunfowora, O., & Najjaran, H. (2023). A Transformer-based Framework for Multi-variate Time Series: A Remaining Useful Life Prediction Use Case. https://doi.org/10.48550/arXiv.2308.09884

Pashamokhtari, A. (2020). Dynamic inference on IoT network traffic using programmable telemetry and machine learning. In Proceedings of the 2020 19th *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)* (pp. 371–372). https://doi.org/10.1109/IPSN48710.2020.00006

Raheem, I., Mubarak, N. M., Karri, R. R., et al. (2022). Forecasting of energy consumption by G20 countries using an adjacent accumulation grey model. *Scientific Reports*, 12, 13417. https://doi.org/10.1038/s41598-022-17505-4

Russell, S. J., & Norvig, P. (2020). (4th ed.). *Artificial Intelligence: A Modern Approach*. Prentice Hall Publishing.

Sahoo, B. B., Jha, R., Singh, A., et al. (2019). Long short-term memory (LSTM) recurrent neural network for low-flow hydrological time series forecasting. *Acta Geophysica*, 67, 1471–1481. https://doi.org/10.1007/s11600-019-00330-1

Shapi, M. K. M., Ramli, N. A., & Awalin, L. J. (2021). Energy consumption prediction by using machine learning for smart building: Case study in Malaysia. *Developments in the Built Environment*, 5, 100037. https://doi.org/10.1016/j.dibe.2020.100037

Shekhar, S., Bansode, A., & Salim, A. (2021). A Comparative study of Hyper-Parameter Optimization Tools. *In 2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)* (pp. 1-6). Brisbane, Australia. https://doi.org/10.1109/CSDE53843.2021.9718485

Shi, J., Jain, M., & Narasimhan, G. (2022). Time Series Forecasting (TSF) Using Various Deep Learning Models. *arXiv, 2204.11115*. https://doi.org/10.48550/arXiv.2204.11115

Tealab, A. (2018). Time series forecasting using artificial neural networks methodologies: A systematic review. *Future Computing and Informatics Journal*, 3(2), 334-340. https://doi.org/10.1016/j.fcij.2018.10.003

Torres, J. F., Hadjout, D., Sebaa, A., Martínez-Álvarez, F., & Troncoso, A. (2021). Deep Learning for Time Series Forecasting: A Survey. *Big Data*, 9(1), 3-21. https://doi.org/10.1089/big.2020.0159

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. *CoRR, abs/1706.03762*. https://doi.org/10.48550/arXiv.1706.03762

Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., & Sun, L. (2023). Transformers in time series: A survey. *In Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence* (pp. 759). Macao, P.R. China: International Joint Conferences on Artificial Intelligence. https://doi.org/10.24963/ijcai.2023/759

Wu, N., Green, B., Ben, X., & O'Banion, S. (2020). Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case. *CoRR, abs/2001.08317*. https://doi.org/10.48550/arXiv.2001.08317

Zeyer, A., Bahar, P., Irie, K., Schlüter, R., & Ney, H. (2019). A Comparison of Transformer and LSTM Encoder Decoder Models for ASR. *In 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)* (pp. 8-15). Singapore. https://doi.org/10.1109/ASRU46091.2019.9004025

Zhang, Q., Lipani, A., Kirnap, Ö., & Yilmaz, E. (2019). Self-Attentive Hawkes Processes. *CoRR, abs/1907.07561*. https://doi.org/10.48550/arXiv.1907.07561

Zuo, S., Jiang, H., Li, Z., Zhao, T., & Zha, H. (2020). Transformer Hawkes Process. *CoRR, abs/2002.09291*. https://doi.org/10.48550/arXiv.2002.09291