AI

Article

Should We Reconsider RNNs for Time-Series Forecasting?

Vahid Naghashi [1,2], Mounir Boukadoum [1,2] and Abdoulaye Banire Diallo [1,2,*]

1 Department of Computer Science, Université du Québec à Montréal, Montreal, QC H2L 2C4, Canada; naghashi.vahid@courrier.uqam.ca (V.N.); boukadoum.mounir@uqam.ca (M.B.)
2 WELL-E : Research and Innovation Chair in Animal Welfare and Artificial Intelligence, Montreal, QC H2L 2C4, Canada
* Correspondence: diallo.abdoulaye@uqam.ca

Abstract: (1) Background: In recent years, Transformer-based models have dominated the time-series forecasting domain, overshadowing recurrent neural networks (RNNs) such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). While Transformers demonstrate superior performance, their high computational cost limits their practical application in resource-constrained settings. (2) Methods: In this paper, we reconsider RNNs—specifically the GRU architecture—as an efficient alternative to time-series forecasting by leveraging this architecture's sequential representation capability to capture cross-channel dependencies effectively. Our model also utilizes a feed-forward layer right after the GRU module to represent temporal dependencies, and aggregates it with the GRU layers to predict future values of a given time-series. (3) Results and conclusions: Our extensive experiments conducted on different real-world datasets show that our inverted GRU (iGRU) model achieves promising results in terms of error metrics and memory efficiency, challenging or surpassing state-of-the-art models on various benchmarks.

Keywords: time-series; gated recurrent units; temporal dependencies; cross-channel correlations

Academic Editor: Mohammad Valipour

Received: 2 April 2025
Revised: 23 April 2025
Accepted: 24 April 2025
Published: 25 April 2025

Citation: Naghashi, V.; Boukadoum, M.; Diallo, A.B. Should We Reconsider RNNs for Time-Series Forecasting? *AI* **2025**, *6*, 90. https://doi.org/10.3390/ai6050090

1. Introduction

Time-series forecasting is an important task in several application domains, including finance, meteorology, transportation, and energy consumption. Providing accurate forecasts helps industries and businesses save both money and time by enabling optimized and informed decision-making ahead of time, thereby avoiding unnecessary actions. Time-series forecasting involves leveraging historical (past) data from various channels or variates to predict future values of the same or related variates. As shown in Figure 1, these variables are often inter-correlated, and temporal relationships exist along the time dimension in a time-series. Time-series analysis and forecasting have garnered significant attention from researchers over the past few decades. With the rise of Artificial Intelligence (AI), deep learning-based methods have taken the lead in this field [1]. Among the deep learning models developed for time-series forecasting, Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), Multi-Layer Perceptrons (MLPs), Transformer models, and Large Language Model (LLM)-based approaches have attained remarkable performance due to their ability to capture complex long-term temporal dependencies [1–5]. A model usually demonstrates high performance for multivariate time-series forecasting when it captures the relations between the prediction variables and the temporal correlations across the historical time steps. There are two main approaches used for time-series forecasting with deep learning models. The first category of methods are known as Channel-Dependent (CD) methods, which usually project the channel dimension into a hidden space (modeling dimension). However, recent works have shown that Channel-Independent (CI) models generally achieve better results [6,7]. Recently, many Transformer-based models have been proposed using channel independence, where multiple channels are predicted independently. In addition, one of the recently invented CI-based models, iTransformer [8], directly captures the intricate interaction among multiple time-series channels by exploiting the high capacity of the multi-head self-attention module inside the Transformer, and embedding the temporal dimension within the model dimension, leading to impressive results, specifically when dealing with complex real-world datasets. However, the Transformer-based models face the obvious challenge of quadratic time complexity with respect to the input sequence length, specifically in the inference step, which gives rise to an intensive calculation when applied to a large number of variates or longer look-back windows, hindering their deployment in real-world applications.

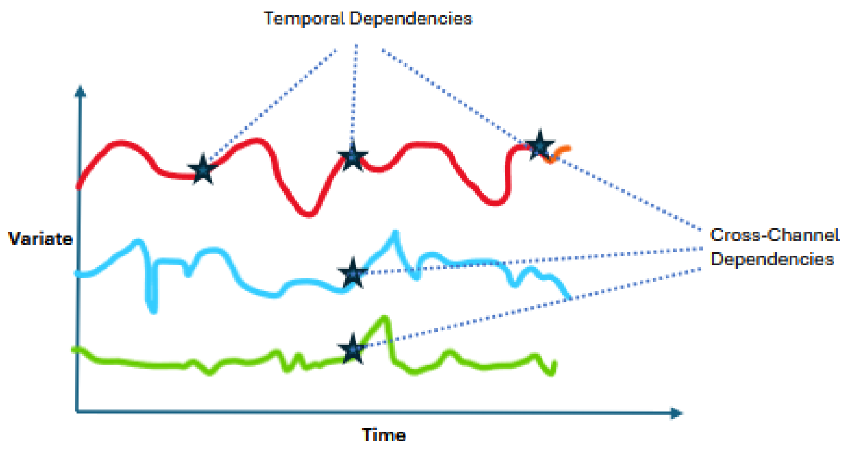

Figure 1. An example of a multivariate time-series which consists of multiple channels (variates). There are inter-variate dependencies between the channels and temporal correlations between different time-steps.

There have already been attempts to reduce the computational complexity of Transformer for time-series forecasting. For instance, Ref. [9] modified the Transformer to focus on a portion of the sequence, while other works utilized linear models to decrease the time complexity [7,10]. Although linear models can reduce time complexity, they mainly rely on linear computations and fail to exploit contextual information, resulting in sub-optimal predictions, specifically when applied to datasets involving many variates or series. Given the prevalent use of Transformer models in the time-series forecasting (TSF) domain, recurrent models such as LSTM and GRU have been neglected and their potential to capture cross-channel dependencies escaped the attention of researchers in the field. RNNs are still occasionally utilized for TSF [11] in rare cases. However, their capability for sequential modeling is underestimated compared to that of Transformers.

In this paper, we reconsider the RNNs, specifically the Gated Recurrent Unit (GRU), as an alternative to the multi-head self-attention module in the Transformer models. We exploit GRUs to extract the interactions among time-series channels and utilize them in multiple-channel forecasting. Inspired by the iTransformer [8] model, we apply the GRU to the channel dimension of the input series to capture inter-series correlations. Considering the importance of capturing cross-channel correlations in time-series analysis, we will answer the question of whether RNNs (including GRU or LSTM) should still be considered for time-series forecasting. We answer this question by conducting experiments using various public time-series datasets and analyzing the results, specifically through the ablation experiments. Recurrent Neural Networks (RNNs) represent significant advantages in inference time efficiency for time-series forecasting, although their performance some-times falls behind that of Transformer-based architectures. The proposed iGRU achieves a competitive performance with reduced computational overhead. On datasets like Solar, iGRU outperforms several Transformer models in both accuracy and efficiency, as shown in Section 3. However, for some datasets, iGRU's performance is surpassed by computation-intensive models, reflecting a trade-off between accuracy and computational cost. This work explores these trade-offs, demonstrating iGRU's potential as a lightweight and effective model for time-series forecasting task. Furthermore, in most cases, iGRU outperforms the recently proposed iTransformer [8]. The prominent reason for utilizing GRUs in our model is that RNNs provide significantly lower time and memory complexity compared to many Transformer-based models, resulting in improved efficiency for specific applications, as demonstrated in Table 1. Additionally, the clear temporal flow in RNNs provides better interpretability in understanding how information propagates through sequences [12], specifically when compared to Transformer and MLP-based architectures.

Table 1. Comparison of time and memory complexity for different models, where L represents the input sequence length (context length).

Method	Type	Time Complexity	Memory Complexity
GRU	RNN	$O(L)$	$O(L)$
DLinear	MLP	$O(L)$	$O(L)$
Crossformer	Transformer	$O(L^2)$	$O(L^2)$

Our work includes the following contributions to the time-series forecasting domain:

- We reconsider RNNs for time-series forecasting using a different approach by focusing on the inter-channel dependencies and describe the inverted GRU (iGRU), which exploits GRU blocks to capture interactions between the time-series channels and feed-forward layers to represent temporal relations.
- We extensively evaluate iGRU on eleven public datasets and report the results in terms of error metrics and memory efficiency.
- We show that our iGRU model achieves comparable results to the state-of-the-art models or outperforms them.

Time-series forecasting methods are generally categorized into statistical models, such as Auto-ARIMA [13], and modern (deep learning) approaches, including Transformer, linear and convolutional neural network models. The Transformer architecture was initially designed to process and generate token sequences, mainly for natural language processing applications, especially Large Language Models (LLMs). However, its excellent potential motivated the TSF research community to deploy and adapt it for time-series tasks. For instance, LogTrans [14] uses convolutional attention in the LogSparse design to capture local information and reduce time complexity. The Informer [15] exploits the ProbSparse self-attention with distillation to emphasize prominent keys. In the Autoformer [16], the idea of time-series decomposition and auto-correlation calculation is proposed to extract temporal correlations. The FEDformer [17] is designed based on Fourier-based architecture and achieves a linear time complexity. In another work, the Pyraformer [18] utilizes pyramidal attention to capture inter-scale and intra-scale relations with a linear complexity. Recently, PatchTST [19] was proposed based on a Transformer architecture, which utilizes patched time-series with channel independence to capture temporal correlations for each channel, separately. In a different design, the Crossformer [20] exploits an encoder–decoder structure with hierarchical attention modules to leverage cross-channel dependencies. However, some Linear models have emerged recently to outperform Transformers in benchmark experiments [7,21]. On the other hand, these linear models fall short in representing non-linear dependencies between the input series and future time steps [7]. Recently, CNN-based models achieved promising results in time-series analysis tasks. As a prominent example, TimesNet [22] utilizes two-dimensional convolutions to capture inter-period and intra-period relations in a time-series with multiple period lengths, obtaining promising results. Over the past few years, Transformers and CNNs have overshadowed Recurrent Neural Networks (RNNs) in the time-series forecasting domain. For instance, the iTransformer [8] model is proposed based on the vanilla Transformer model and embedding the channels of the input series. This model attained impressive results in many benchmark datasets, pinpointing the importance of modeling cross-channel interactions in the forecasting tasks. The more recent model, TimeXer [23], incorporates the external information to enhance the forecasting accuracy, which strengthens the canonical Transformer to harmonize endogenous and exogenous information by using patch-wise self-attention and cross-variate attention, simultaneously. In this paper, we reconsider the potential of RNNs for capturing sequential dependencies and introduce the inverted GRU (iGRU), which leverages GRUs to capture dependencies across time-series channels while utilizing feed-forward layers to extract temporal features.

2. Materials and Methods

Our iGRU architecture is illustrated in Figure 2 and the corresponding forecasting procedure is shown in Algorithm 1. Given $x_t \in \mathbb{R}^C$, the observation of a time-series with C channels at time t, we aim to forecast its future H time-steps $x_{t+1}, ..., x_{t+H}$ using a history or context window w_t of length L (i.e., $w_t = (x_{t-L+1}, ..., x_t)$).

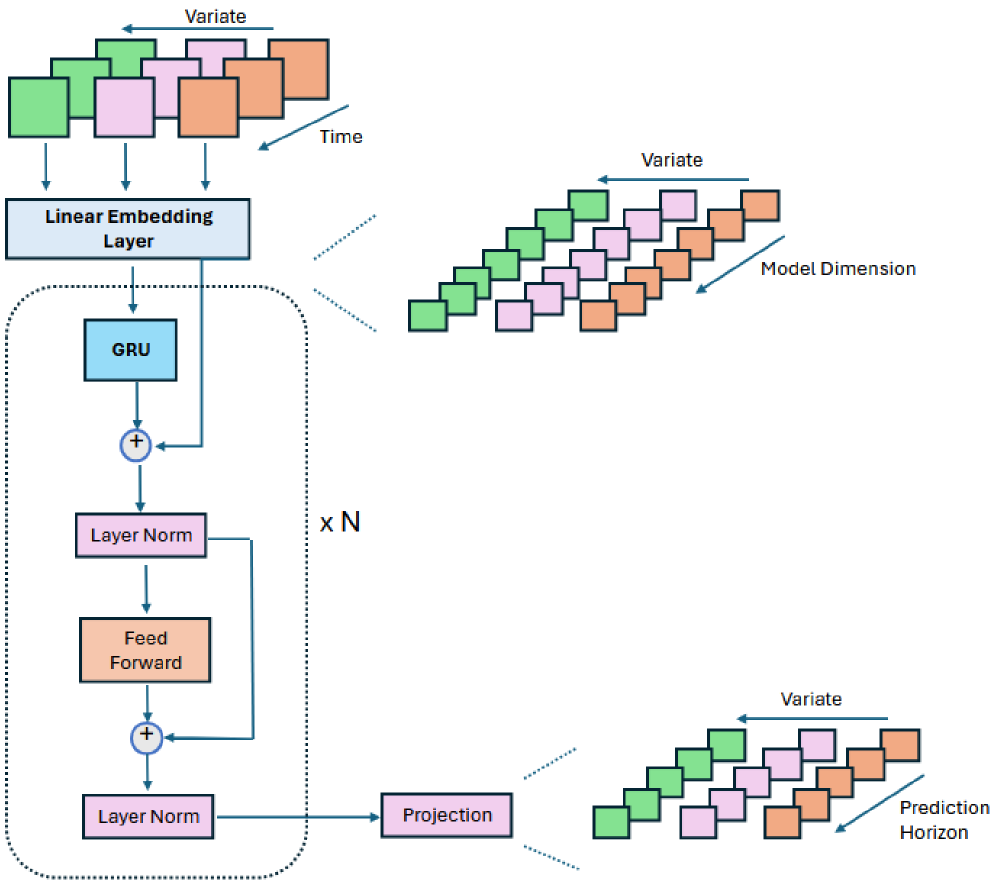

Figure 2. Illustration of iGRU architecture.

Algorithm 1 The forecasting procedure of iGRU

Input: Batch$(X) = [x_1, x_2, \ldots, x_L] : (B, L, C)$
Output: Batch$(Y) = [y_1, y_2, \ldots, y_H] : (B, H, C)$

```
1:  $X_T : (B, C, L) \leftarrow \text{Transpose}(\text{Batch}(X))$
2:  $X_{embedded} : (B, C, D) \leftarrow \text{Embedding}(X_T)$
3:  for each layer $l$ in iGRU layers do
4:      $X_{CC} \leftarrow \overrightarrow{\text{GRU}}(X_{embedded})$
5:      $X_{CCR} \leftarrow X_{CC} + X_{embedded}$
6:      $X_{CCR} \leftarrow \text{LayerNorm}(X_{CCR})$
7:      $X_{FF} \leftarrow \text{Feed-Forward}(X_{CCR})$
8:      $X_{FF} \leftarrow X_{FF} + X_{CCR}$
9:      $X_F \leftarrow \text{LayerNorm}(X_{FF})$
10:     $X_{embedded} \leftarrow X_F$
11: end for
12: $X_{out} : (B, C, H) \leftarrow \text{Projection}(X_F)$
13: $\text{Batch}(Y) : (B, H, C) \leftarrow \text{Transpose}(X_{out})$
```

2.1. Preliminaries

2.1.1. RNN and GRU

Recurrent Neural Networks (RNNs) are a category of neural networks designed for sequential data processing across multiple time steps. The Gated Recurring Unit [24], introduced in 2014, is a specific type of RNN with a gating mechanism to input or forget information along a sequence of timesteps, which employs update and forget units to process sequential data mapped to a hidden space. The GRU operation is defined by the following equations:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{1}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{2}$$

$$\hat{h}_t = \phi(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{3}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \tag{4}$$

where $\odot$ represents element-wise multiplication. x_t and h_t indicate input and hidden state vectors at time t, respectively. z_t, r_t, $\hat{h}_t$ represent the update vector, reset vector and candidate hidden state at time step t, respectively. In addition, W_h, W_z, W_r, U_h, U_z, U_r, b_h, b_z and b_r are the relevant weights and biases which are learned during the model training phase. This type of RNN showed effective sequential modeling in various applications [24–26]. In most applications, GRUs and LSTMs are commonly used to capture temporal dependencies. However, their efficiency in modeling cross-channel correlations within time-series data has often been overlooked. To leverage RNNs for capturing inter-series dependencies, the input multivariate time-series must first be projected into a hidden space using a simple linear embedding layer [8]. We selected GRU over LSTM due to its comparatively lower parameter count while achieving similar performance, which aligned with our goal of maintaining model efficiency. Vanilla RNNs, on the other hand, are not studied in our approach as they lack memory gates, which limits their performance relative to GRUs.

2.1.2. Temporal Embedding of Time-Series

Similarly to [8], the input multivariate series is embedded into a higher-dimensional space through a linear layer. The input to the temporal embedding has the shape X (Batch, Channel, Time Length), which is an inverted version of the input time-series obtained by swapping the temporal and channel dimensions. Then, X is projected into the model space along its temporal dimension by

$$X_{embedded} = Linear(X) \tag{5}$$

Here, *Linear* refers to a fully connected layer, where the input dimension corresponds to the series length, and the output dimension corresponds to the model dimensionality.

2.2. Proposed iGRU Model

As illustrated in Figure 2, the input multivariate series is first passed to the linear embedding layer, which is applied series-wise to map each channel into a hidden space. Before embedding, instance normalization [27] is utilized to reduce non-stationarity and distribution shifts between the training and test sets. The embedded series is then sent to the GRU module, which acts like the multi-head self-attention of Transformers to capture the intricate interactions among multiple time-series channels. Here, the GRU cells represent those dependencies in one direction starting from the first channel to the last one, similar to temporal sequence modeling:

$$X_{CC} = \overrightarrow{GRU}(X_{embedded}) \tag{6}$$

The channel or variate correlated output, X_{CC}, encoded by the GRU layer, is connected with its input to form the output of this layer, facilitating gradient flow and training stability:

$$X_{CCR} = X_{CC} + X_{embedded} \tag{7}$$

After adding the GRU output and the skip connection, layer normalization is applied to normalize the activations within each layer to obtain a mean of zero and a variance of one, thereby stabilizing training through the reduction of varying feature scales. Then, a feed-forward layer (FFN) is applied to the series representations to capture temporal correlations along each channel. The feed-forward network (FFN) consists of two fully connected layers mapping the series representation to a higher dimensional space (two times the model dimension) and then to the model dimension, with a Gaussian Error Linear Unit (GELU) activation used between the layers. It is worth noting that FFN implicitly represents temporal dependencies along the model dimension. Finally, the FFN module output is added to its input using a skip connection and a normalization layer is employed afterwards to adjust the obtained series representation. The final prediction is obtained by applying a projection layer to the output of the feed-forward network. The projection layer is a fully connected layer which projects the model dimension to the forecasting horizon (prediction length), generating the predictions of multiple channels.

3. Results

The Proposed iGRU is thoroughly and carefully evaluated on eleven public datasets. The Traffic dataset [6] is a collection of road occupancy data from the California Department of Transportation. It was gathered from 862 sensors between 2015 and 2016. PEMS [6,28] is a complex spatiotemporal dataset related to public traffic networks in California consisting of four different datasets (PEMS03, PEMS04, PEMS07, and PEMS08). The ETT (Electricity Transformer Temperature) dataset [16] includes data related to the load and oil temperature of electricity transformers collected from July 2016 to July 2018. This collection contains different datasets captured hourly or in minutes granularity, including ETTm1 and ETTm2, all consisting of seven variates. The Weather dataset [16] consists of 21 meteorological variates recorded every 10 min from the Max Planck Bio-geochemistry institute. The Electricity dataset [16] contains hourly electricity consumption of 321 costumers. The Solar-

Energy dataset [16], which was sampled every 10 min, collected solar power records in 2006 from 137 PV plants in the US state of Alabama. The Exchange dataset [16] is collected based on panel data of daily exchange rates corresponding to eight countries from 1990 to 2016. More information regarding the datasets are reported in Table 2.

Table 2. Dataset information: Dim represents the number of variates and Dataset Size denotes the total number of time points in (Training, Validation, Testing) split of each dataset, respectively. Prediction Length indicates the future time points to be predicted and four prediction lengths settings are specified in each dataset. Frequency denotes the sampling interval of time points.

Dataset	Dim	Prediction Length	Dataset Size	Frequency	Information
ETTm1, ETTm2	7	{96, 192, 336, 720}	(34,465, 11,521, 11,521)	15 min	Electricity
Exchange	8	{96, 192, 336, 720}	(5120, 665, 1422)	Daily	Economy
Weather	21	{96, 192, 336, 720}	(36,792, 5271, 10,540)	10 min	Weather
Electricity	321	{96, 192, 336, 720}	(18,317, 2633, 5261)	Hourly	Electricity
Traffic	862	{96, 192, 336, 720}	(12,185, 1757, 3509)	Hourly	Transportation
Solar-Energy	137	{96, 192, 336, 720}	(36,601, 5161, 10,417)	10 min	Energy
PEMS03	358	{12, 24, 48, 96}	(15,617, 5135, 5135)	5 min	Transportation
PEMS04	307	{12, 24, 48, 96}	(10,172, 3375, 3375)	5 min	Transportation
PEMS07	883	{12, 24, 48, 96}	(16,911, 5622, 5622)	5 min	Transportation
PEMS08	170	{12, 24, 48, 96}	(10,690, 3548, 3548)	5 min	Transportation

We compared the proposed iGRU model with several state-of-the-art models, including TimeXer [23], iTransformer [8], PatchTST [19], Crossformer [20], TiDE [21], DLinear [7], FEDformer [17], Autoformer [16] and TimesNet [22]. We implemented our proposed model in Pytorch [29] and executed our experiments using a single A100-40G NVIDIA GPU. All models were trained for 10 epochs with early stopping patience of three steps based on the validation loss change. The Mean Square Error (MSE) loss function is utilized with the Adam [30] optimizer to train all models. We set the learning rate to 0.001 for the Traffic, Electricity and PEMS datasets and lower values (0.0005 or 0.0001) are used for the other datasets. This choice is an attempt to mitigate overfitting and skipping of sub-optimal results, since these datasets had a limited number of training instances. For each hyper-parameter, we tried a range of possible values, and the value yielding the best result was picked. The reported results represent an average of five runs. In our experiments, the batch size is uniformly selected as 16 or 32, and the number of iGRU blocks are set from $\{1,2,3,4\}$. Additionally, the model dimension is chosen from $\{128,256,512\}$ according to each dataset. The selected hyper-parameters for each dataset are shown in Table 3. The results of time-series forecasting in terms of MSE and MAE (Mean Absolute Error) are reported in Tables 4 and 5.

Table 3. Selected hyper-parameters for training the iGRU model on different benchmarks.

Dataset	Model Dimension	Feed-Forward Dimension	iGRU Blocks	Learning Rate	Batch Size	Dropout
ETTm1	256	512	2	0.0001	32	0.1
ETTm2	256	512	2	0.0001	32	0.1
Weather	512	512	3	0.0001	32	0.1
Exchange	256	256	2	0.00005	32	0.1
Electricity	512	512	3	0.001	16	0.1
Traffic	512	512	4	0.001	16	0.1
PEMS03	512	512	4	0.001	16	0.1
PEMS04	1024	1024	4	0.001	16	0.1
PEMS07	512	512	3 or 4	0.001	16	0.1
PEMS08	512	512	3 or 4	0.001	16	0.1

Table 4. Multivariate time-series forecasting results of iGRU and the baseline models on Traffic and PEMS datasets. The input length (lookback window) is set to 96 and the prediction length is in {12, 24, 48, 96} for PEMS datasets and in {96, 192, 336, 720} for Traffic dataset. The best results are shown in bold, and the second-best results are shown in italics.

Models		**Ours**		**TimeXer**		**iTransformer**		**PatchTST**		**DLinear**		**Crossformer**		**TimesNet**		**TiDE**		**FEDformer**		**Autoformer**	
	Metric	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
Traffic	96	**0.393**	**0.268**	0.428	0.271	0.395	0.268	0.462	0.295	0.650	0.396	0.522	0.290	0.593	0.321	0.805	0.493	0.587	0.366	0.613	0.388
	192	**0.417**	*0.277*	0.448	0.282	**0.417**	**0.276**	0.466	0.296	0.598	0.370	0.530	0.293	0.617	0.336	0.756	0.474	0.604	0.373	0.616	0.382
	336	**0.431**	**0.283**	0.473	0.289	0.433	**0.283**	0.482	0.304	0.605	0.373	0.558	0.305	0.629	0.336	0.762	0.477	0.621	0.383	0.622	0.337
	720	**0.463**	**0.301**	0.516	0.307	0.467	0.302	0.514	0.322	0.645	0.394	0.589	0.238	0.640	0.350	0.719	0.449	0.626	0.382	0.660	0.408
	Avg	**0.426**	**0.282**	0.466	0.287	*0.428*	**0.282**	0.481	*0.304*	0.625	0.383	0.550	0.304	0.620	0.336	0.760	0.473	0.610	0.376	0.628	0.379
PEMS03	12	**0.069**	**0.172**	0.072	0.184	*0.071*	*0.174*	0.099	0.216	0.122	0.243	0.090	0.203	0.085	0.192	0.178	0.305	0.126	0.251	0.272	0.385
	24	**0.087**	**0.195**	*0.088*	0.202	0.093	*0.201*	0.142	0.259	0.201	0.317	0.121	0.240	*0.118*	0.223	0.257	0.371	0.149	0.275	0.334	0.440
	48	**0.119**	**0.230**	0.127	0.242	*0.125*	*0.236*	0.211	0.319	0.333	0.425	0.202	0.317	0.155	*0.260*	0.379	0.463	0.227	0.348	1.032	0.782
	96	**0.151**	**0.264**	0.177	0.284	*0.164*	*0.275*	0.269	0.370	0.457	0.515	0.262	0.367	*0.228*	*0.317*	0.490	0.539	0.348	0.434	1.031	0.796
	Avg	**0.107**	**0.215**	0.116	0.228	*0.113*	*0.221*	0.180	0.291	0.278	0.375	0.169	0.281	0.147	0.248	0.326	0.419	0.213	0.327	0.667	0.601
PEMS04	12	**0.078**	*0.185*	0.082	0.197	**0.078**	**0.183**	0.105	0.224	0.148	0.272	0.098	0.218	0.087	0.195	0.219	0.340	0.138	0.262	0.424	0.491
	24	**0.091**	**0.204**	0.094	0.212	*0.095*	*0.205*	0.153	0.275	0.224	0.340	0.131	0.256	0.103	0.215	0.292	0.398	0.177	0.293	0.459	0.509
	48	**0.114**	**0.230**	0.119	0.237	*0.120*	*0.233*	0.229	0.339	0.355	0.437	0.205	0.326	0.136	0.250	0.409	0.478	0.270	0.368	0.646	0.610
	96	**0.141**	**0.254**	0.162	0.275	*0.150*	*0.262*	0.291	0.389	*0.452*	*0.504*	0.402	0.457	0.190	0.303	0.492	0.532	0.341	0.427	0.912	0.748
	Avg	**0.106**	**0.218**	0.114	0.230	*0.111*	*0.221*	0.195	0.307	0.295	0.388	0.209	0.314	0.129	0.241	0.353	0.437	0.231	0.337	0.610	0.590
PEMS07	12	*0.065*	**0.163**	**0.063**	0.171	0.067	*0.165*	0.095	0.207	0.115	0.242	0.094	0.200	0.082	0.181	0.173	0.304	0.109	0.225	0.199	0.336
	24	*0.084*	*0.188*	**0.079**	**0.187**	0.088	0.190	0.150	0.262	0.210	0.329	0.139	0.247	0.101	0.204	0.271	0.383	0.125	0.244	0.323	0.420
	48	*0.103*	*0.210*	**0.100**	**0.203**	0.110	0.215	0.253	0.340	0.398	0.458	0.311	0.369	0.134	0.238	0.446	0.495	0.165	0.288	0.390	0.470
	96	**0.128**	*0.235*	*0.131*	**0.233**	0.139	0.245	0.346	0.404	0.594	0.553	0.396	0.442	0.181	0.279	0.628	0.577	0.262	0.376	0.554	0.578
	Avg	*0.095*	**0.199**	**0.093**	**0.199**	0.101	0.204	0.211	0.303	0.329	0.395	0.235	0.315	0.193	0.271	0.380	0.440	0.165	0.283	0.367	0.451
PEMS08	12	**0.077**	**0.179**	0.091	0.206	*0.079*	*0.182*	0.168	0.232	0.154	0.276	0.165	0.214	0.112	0.212	0.227	0.343	0.173	0.273	0.436	0.485
	24	**0.109**	**0.212**	0.133	0.253	*0.115*	*0.219*	0.224	0.281	0.248	0.353	0.215	0.260	0.141	0.238	0.318	0.409	0.210	0.301	0.467	0.502
	48	**0.177**	**0.232**	0.209	0.249	*0.186*	*0.235*	0.321	0.354	0.440	0.470	0.315	0.355	0.198	0.283	0.497	0.510	0.320	0.394	0.966	0.733
	96	**0.213**	**0.262**	0.492	0.467	*0.221*	*0.267*	0.408	0.417	0.674	0.565	0.377	0.397	0.320	0.351	0.721	0.592	0.442	0.465	1.385	0.915
	Avg	**0.144**	**0.221**	0.231	0.294	*0.150*	*0.226*	0.280	0.321	0.379	0.416	0.268	0.307	0.193	0.271	0.441	0.464	0.286	0.358	0.814	0.659

Table 5. Multivariate time-series forecasting results of iGRU and the baseline models on Weather, Electricity, Exchange, Solar-energy, ETTm1 and ETTm2 datasets. The input length (look-back window) is set to 96 and the prediction length is {96, 192, 336, 720}. The best results are shown in bold, and the second-best results are shown in italics.

Models		**Ours**		**TimeXer**		**iTransformer**		**PatchTST**		**DLinear**		**Crossformer**		**TimesNet**		**TiDE**		**FEDformer**		**Autoformer**	
	Metric	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
Weather	96	*0.167*	0.210	**0.157**	**0.205**	0.174	*0.214*	0.177	0.218	0.196	0.255	*0.158*	0.230	0.172	0.220	0.202	0.261	0.217	0.296	0.266	0.336
	192	0.215	*0.254*	**0.204**	**0.247**	0.221	*0.254*	0.225	0.259	0.237	0.296	*0.206*	0.277	0.219	0.261	0.242	0.298	0.276	0.336	0.307	0.367
	336	0.273	*0.297*	**0.261**	**0.290**	0.278	*0.296*	0.278	0.297	0.283	0.335	*0.272*	0.335	0.280	0.306	0.287	0.335	0.339	0.380	0.359	0.395
	720	0.354	0.349	**0.340**	**0.341**	0.358	*0.347*	0.354	0.348	*0.345*	0.381	0.398	0.418	0.365	0.359	0.351	0.386	0.403	0.428	0.419	0.428
	Avg	*0.253*	*0.278*	**0.241**	**0.271**	0.258	*0.278*	0.259	0.281	0.265	0.317	0.259	0.315	0.259	0.287	0.271	0.320	0.309	0.360	0.338	0.382
Electricity	96	*0.142*	**0.238**	**0.140**	0.242	0.148	*0.240*	0.181	0.270	0.197	0.282	0.219	0.314	0.168	0.272	0.237	0.329	0.193	0.308	0.201	0.317
	192	*0.160*	*0.255*	**0.157**	0.256	0.162	**0.253**	0.188	0.274	0.196	0.285	0.231	0.322	*0.184*	0.289	0.236	0.330	0.201	0.315	0.222	0.334
	336	**0.176**	*0.272*	**0.176**	0.275	*0.178*	**0.269**	0.204	0.293	0.209	0.301	0.246	0.337	0.198	*0.300*	0.249	0.344	0.214	0.329	0.231	0.338
	720	**0.210**	**0.301**	*0.211*	*0.306*	0.225	0.317	0.246	0.324	0.245	0.333	0.280	0.363	*0.220*	*0.320*	0.284	0.373	0.246	0.355	0.254	0.361
	Avg	*0.172*	**0.267**	**0.171**	0.270	0.178	*0.270*	0.205	0.290	0.212	0.300	0.244	0.334	0.192	0.295	0.251	0.344	0.214	0.327	0.227	0.338
Solar	96	*0.194*	*0.243*	**0.187**	0.250	0.203	**0.237**	0.234	0.286	0.290	0.378	0.310	0.331	0.250	0.292	0.312	0.399	0.242	0.342	0.884	0.711
	192	**0.208**	**0.255**	*0.202*	0.271	0.233	*0.261*	0.267	0.310	0.320	0.398	0.734	0.725	0.296	0.318	0.339	0.416	0.285	0.380	0.834	0.692
	336	**0.214**	**0.271**	*0.215*	0.284	0.248	*0.273*	0.290	0.315	0.353	0.415	0.750	0.735	0.319	0.330	0.368	0.430	0.282	0.376	0.941	0.723
	720	**0.214**	**0.264**	*0.220*	0.293	0.249	*0.275*	0.289	0.317	*0.356*	*0.413*	0.769	0.765	0.338	0.337	0.370	0.425	0.357	0.427	0.882	0.717
	Avg	**0.208**	**0.258**	*0.229*	0.274	0.233	*0.262*	0.270	0.307	0.330	0.401	0.641	0.639	0.301	0.319	0.347	0.417	0.291	0.381	0.885	0.711
Exchange	96	**0.086**	0.207	**0.086**	*0.206*	**0.086**	*0.206*	*0.088*	**0.205**	*0.088*	0.218	0.256	0.367	0.107	0.234	0.094	0.218	0.148	0.278	0.197	0.323
	192	*0.181*	*0.304*	0.188	0.308	0.177	**0.299**	**0.176**	**0.299**	**0.176**	0.315	0.470	0.509	0.226	0.344	0.184	0.307	0.271	0.315	0.300	0.369
	336	*0.331*	*0.417*	0.342	0.421	0.331	*0.417*	**0.301**	**0.397**	0.313	0.427	1.268	0.883	0.367	0.448	0.349	0.431	0.460	0.427	0.509	0.524
	720	0.857	0.702	0.870	0.702	*0.847*	**0.691**	0.901	0.714	**0.839**	*0.695*	1.767	1.068	0.964	0.746	0.852	0.698	1.195	0.695	1.447	0.941
	Avg	0.364	0.408	0.372	0.409	*0.360*	**0.403**	0.367	*0.404*	**0.354**	0.414	0.940	0.707	0.416	0.443	0.370	0.413	0.519	0.429	0.613	0.539
ETTm1	96	*0.321*	*0.358*	**0.318**	**0.356**	0.334	0.368	0.329	0.367	0.345	0.372	0.404	0.426	0.338	0.375	0.364	0.387	0.379	0.419	0.505	0.475
	192	*0.364*	**0.382**	**0.362**	*0.383*	0.377	0.391	0.367	0.385	0.380	0.389	0.450	0.451	0.374	0.387	0.398	0.404	0.426	0.441	0.553	0.496
	336	*0.399*	**0.406**	**0.395**	*0.407*	0.426	0.420	*0.399*	0.410	0.413	0.413	0.532	0.515	0.410	0.411	0.428	0.425	0.445	0.459	0.621	0.537
	720	0.470	0.445	**0.452**	*0.441*	0.491	0.459	*0.454*	**0.439**	0.474	0.453	0.666	0.589	0.478	0.450	487	0.461	0.543	0.490	0.671	0.561
	Avg	0.389	*0.398*	**0.382**	**0.397**	0.407	0.410	*0.387*	0.400	0.403	0.407	0.513	0.496	0.400	0.406	0.419	0.419	0.448	0.452	0.588	0.517
ETTm2	96	0.177	0.260	**0.171**	**0.256**	0.180	0.264	*0.175*	*0.259*	0.193	0.292	0.287	0.366	0.187	0.267	0.207	0.305	0.203	0.287	0.255	0.339
	192	0.242	0.304	**0.237**	**0.299**	0.250	0.309	*0.241*	*0.302*	0.284	0.362	0.414	0.492	0.249	0.304	0.290	0.364	0.269	0.328	0.281	0.340
	336	0.306	*0.343*	**0.296**	**0.338**	0.311	0.348	*0.305*	*0.343*	0.369	0.427	0.597	0.542	0.321	0.351	0.377	0.422	0.325	0.366	0.339	0.372
	720	0.408	*0.406*	**0.392**	**0.394**	0.412	0.407	*0.402*	0.412	0.554	0.522	1.730	1.042	0.408	0.403	0.558	0.524	0.421	0.415	0.433	0.432
	Avg	0.283	0.328	**0.274**	**0.322**	0.288	0.332	*0.281*	*0.326*	0.350	0.401	0.757	0.610	0.290	0.333	0.358	0.404	0.305	0.349	0.327	0.371

4. Discussion

Our iGRU model outperforms most baseline models, notably iTransformer, by capturing inter-series relations with GRU modules. Despite the relatively low number of variates in the ETT datasets (seven variates), iGRU achieves a better performance compared to iTransformer and PatchTST models. For example, on the ETTm1 dataset, our iGRU model outperforms iTransformer by more than 4% on average in terms of the MSE metric, highlighting its efficiency in capturing relations between variates and temporal time steps. Our model efficiency is also verified by its performance in datasets with numerous periodic variations, including datasets from traffic, electricity, and PEMS. As observed in Table 4, the iGRU model exhibits noticeably higher MSE and MAE values on the Traffic dataset compared to the PEMSn datasets. This difference can be attributed to the inherent complexity and variability of the Traffic dataset, which includes diverse traffic patterns influenced by external factors such as weather, events, or road conditions, making it less predictable than the PEMSn datasets. In addition, the Traffic dataset's hourly sampling frequency records broader trends, amplifying variability and reducing short-term pattern consistency, whereas the PEMSn datasets' 5 min sampling pattern provides finer patterns, facilitating more predictable temporal dependencies. This trend of elevated errors is consistent across other models evaluated on the traffic dataset, suggesting that the dataset's characteristics pose a general challenge. The results associated with the Traffic, PEMS03, PEMS04, PEMS07 and PEMS08 datasets underscore the capability of iGRU in handling inter-series dependencies more efficiently compared to other baseline models. According to Table 4, iGRU reduces the MSE error (averaged over four prediction lengths) by nearly 6% compared to the second-best baseline (iTransformer) in the PEMS07 dataset, which comprises 883 variates and represents a spatiotemporal type of time-series. iGRU also achieves prediction accuracy comparable to TimeXer while consuming less GPU memory and training time, especially when the prediction length is set to 96. In the traffic dataset, iGRU reduces the MSE error by more than 8.5% on average, relative to TimeXer. When averaged across four prediction horizons, the MSE error also improved by more than 21% in the PEMS08 dataset compared to TimeXer. Additionaly, iGRU performs better than TimeXer in predicting solar power along different forecasting lengths, by more than 9% on average MSE. This highlights the capability of the iGRU model to capture complex cross-variate dependencies. To demonstrate the robustness of our iGRU model, we trained it five times with various random seeds and reported the mean and standard deviation of the results in Table 6. The low standard deviations of the MSE and MAE errors in the test sets associated with different datasets confirm the stability and robustness of the iGRU model.

To illustrate the performance of iGRU on different datasets intuitively, we present a visual comparison of its predictions against the ground truth target series. These visualizations provide a clear and interpretable assessment of the model forecasting accuracy. In the provided plots (Figure 3), the orange line indicates the predictions related to a model and the blue line demonstrates the actual selected sequence. Figure 3 indicates that predictions corresponding to the iGRU are well aligned with the ground truth series on different datasets compared to the predictions generated by the iTransformer.

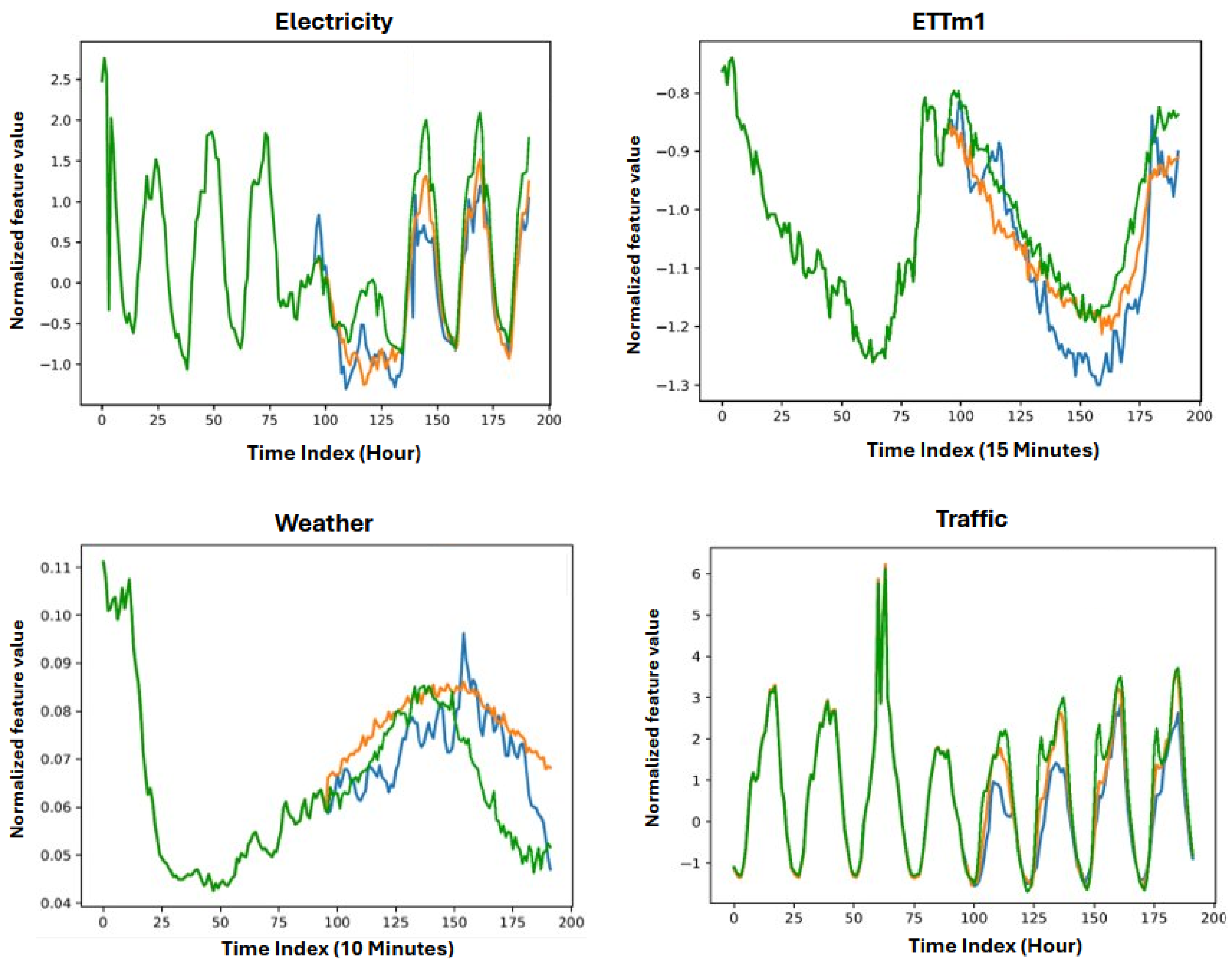

Figure 3. Visual comparison of iGRU and iTransformer on different datasets. The orange line indicates predictions for iGRU and green line corresponds to predictions for iTransformer, with the blue line indicating the ground truth. The look-back window and forecast window lengths are set to 96 for all datasets.

4.1. Model Efficiency

To assess our model's computational efficiency, its memory consumption and training time are compared against those of the other baselines on the Traffic and PEMS07 datasets. Here, by efficiency, we mean training time and GPU memory consumption. Independent runs are conducted using a single A100-40G GPU with the batch size fixed to 16. Our model's efficiency is illustrated in Figure 4, where bubble charts show a visual comparison of efficiency metrics. The vertical axis indicates the prediction MSE, and the horizontal axis depicts the duration of one training iteration (milliseconds/iteration). The bubble size indicates the related memory footprint in Gigabytes (total memory consumption in one epoch). As Figure 4 illustrates, the iGRU model attains the most accurate results, while requiring less or equal training time and memory usage compared to other baselines, except the linear models. DLinear consumes minimal memory and time resources than the other models, while delivering the least accurate forecasts. To further support the claim regarding the computational efficiency of iGRU compared to Transformer-based models, we conducted additional experiments on the Electricity dataset with varying input lengths (96, 336, and 720), while keeping the prediction length fixed at 96. For each configuration, we measured the training time (ms/iteration), peak GPU memory usage, and related model performance (MSE). Figure 5 illustrates how model cost varies across different input lengths. Notably, iGRU consistently maintains a short training time and low memory footprint, while delivering competitive or better accuracy compared to Transformer-based alternatives. These results confirm that iGRU offers a stable trade-off between efficiency and performance, specifically when the optimal model is sensitive to the length of input window.

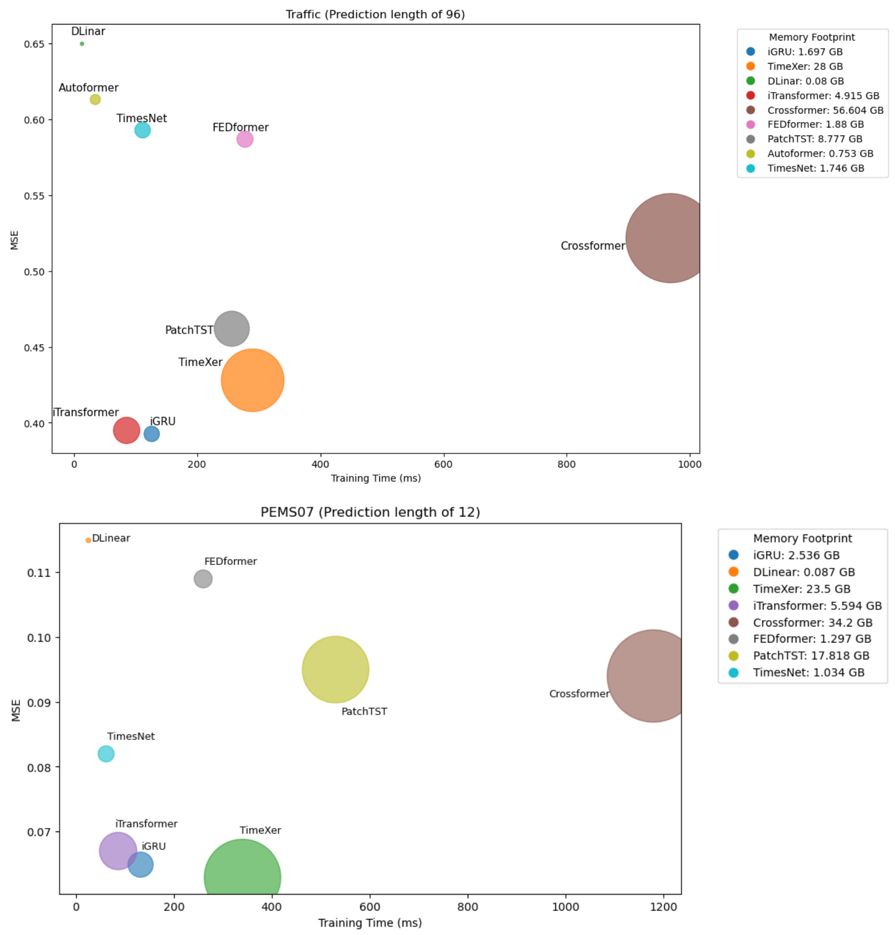

Figure 4. Comparison of iGRU with other baselines in terms of MSE, training time and GPU memory usage on Traffic and PEMS07 datasets. The look-back window is set to 96 and the prediction length is set to 96 and 12 for the Traffic and PEMS07 datasets, respectively.

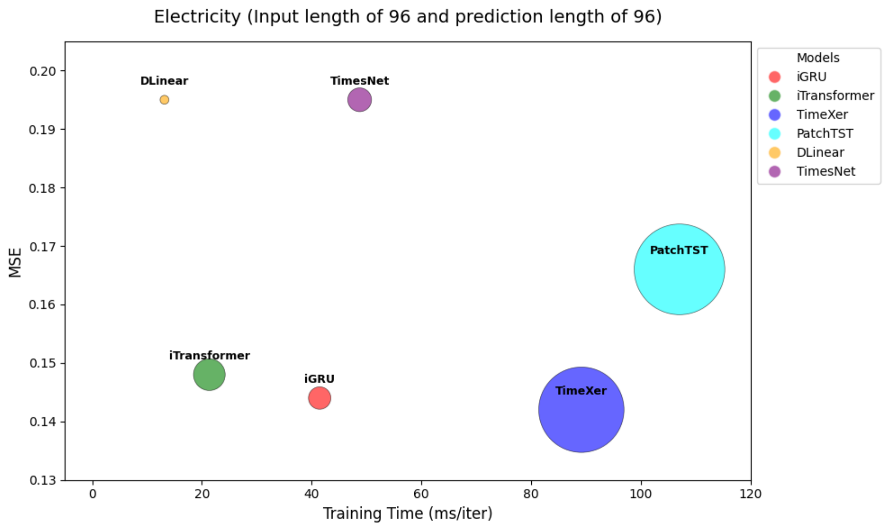

Figure 5. *Cont.*

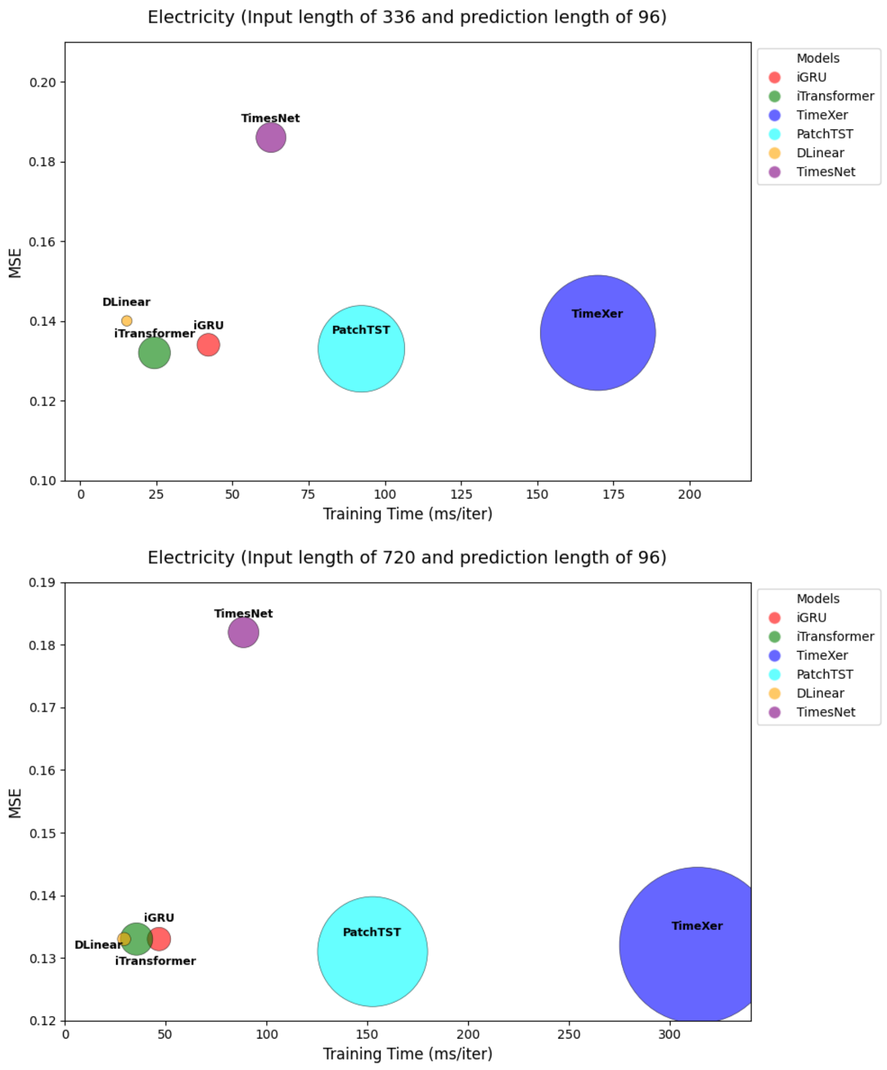

Figure 5. Comparison of training time (ms/iter), model performance (MSE), and memory usage (bubble size) across different input lengths on the Electricity dataset. iGRU consistently demonstrates a low training cost and memory usage while maintaining competitive or better performance compared to Transformer-based models.

4.2. Ablation Study

We demonstrate the importance of the RNN (GRU) and feed-forward layers in our model through conducting experiments with and without the GRU, feed-forward layer and skip connections. The results are reported in Table 7, showcasing the impact of these components on the iGRU performance. The impact of the above components is mostly evident on the Traffic dataset, which is a complex dataset with many time-series variates. The skip connections added to the outputs of the GRU and feed-forward layers contribute to the gradient flow and training efficiency, which helps to attain optimal results. The significant contribution of GRU blocks in capturing cross-variate correlations becomes evident when comparing the results of the model without GRU blocks (W/O GRU) to those of the iGRU model, which incorporates these blocks. This comparison underscores the potential of GRUs to enhance time-series forecasting performance and prompts a raised discussion on whether RNNs, particularly GRUs, should be reconsidered as a powerful tool in time-series analysis, specifically for modeling cross-channel dependencies.

Table 6. Robustness of iGRU model. Five independent runs are conducted with different random seeds.

Dataset	**Electricity**		**Traffic**		**Weather**	
Horizon	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
96	0.142 ± 0.000	0.238 ± 0.000	0.393 ± 0.001	0.267 ± 0.001	0.167 ± 0.002	0.210 ± 0.001
192	0.160 ± 0.000	0.254 ± 0.000	0.416 ± 0.000	0.277 ± 0.001	0.215 ± 0.000	0.254 ± 0.001
336	0.176 ± 0.000	0.272 ± 0.000	0.431 ± 0.000	0.283 ± 0.000	0.273 ± 0.000	0.297 ± 0.000
720	0.210 ± 0.003	0.301 ± 0.003	0.463 ± 0.000	0.301 ± 0.000	0.354 ± 0.001	0.349 ± 0.001
Dataset	**ETTm1**		**ETTm2**		**Exchange**	
Horizon	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
96	0.320 ± 0.001	0.358 ± 0.001	0.178 ± 0.000	0.260 ± 0.000	0.086 ± 0.000	0.207 ± 0.000
192	0.364 ± 0.000	0.382 ± 0.000	0.244 ± 0.001	0.304 ± 0.001	0.181 ± 0.000	0.304 ± 0.000
336	0.398 ± 0.000	0.405 ± 0.000	0.304 ± 0.001	0.343 ± 0.000	0.331 ± 0.000	0.417 ± 0.000
720	0.469 ± 0.001	0.445 ± 0.000	0.408 ± 0.001	0.403 ± 0.001	0.857 ± 0.000	0.702 ± 0.000

Table 7. Ablation of iGRU model without GRU, skip or residual connections and feed-forward layer. The best results are shown in bold.

Design		**W/O F.F**		**W/O F.F + W/O Skip Connection**		**W/O Skip Connection**		**W/O GRU**		**iGRU**	
Metric		**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
ETTm1	96	0.324	0.361	0.325	0.364	0.327	0.364	0.324	0.362	**0.321**	**0.358**
	192	0.366	0.383	0.368	0.387	0.373	0.390	0.367	0.384	**0.364**	**0.382**
	336	**0.399**	**0.405**	0.403	0.410	0.415	0.418	0.400	0.406	**0.399**	0.406
	720	**0.467**	**0.443**	0.475	0.450	0.482	0.455	**0.467**	0.444	0.470	0.445
Traffic	96	0.407	0.281	0.414	0.287	0.502	0.366	0.437	0.282	**0.393**	**0.268**
	192	0.428	0.289	0.437	0.296	0.535	0.372	0.450	0.287	**0.417**	**0.277**
	336	0.445	0.296	0.452	0.303	0.537	0.371	0.464	0.294	**0.431**	**0.283**
	720	0.476	0.313	0.487	0.323	0.572	0.389	0.495	0.312	**0.463**	**0.301**
Weather	96	0.170	0.214	**0.166**	**0.211**	0.169	0.213	0.194	0.232	0.168	**0.211**
	192	0.217	0.256	**0.214**	0.255	0.217	0.257	0.239	0.269	0.215	**0.254**
	336	0.274	0.297	**0.271**	**0.296**	0.277	0.300	0.291	0.307	0.274	0.297
	720	**0.353**	0.349	**0.353**	0.349	0.357	0.352	0.364	0.354	0.354	**0.348**

4.3. Increasing Look-Back Length

Some of the previous Transformer-based works [7,19] have shown that increasing the length of look-back window does not necessarily improve the forecasting results, which can be caused by distracted attention on the increasing (prolonged) input. As the structure of our iGRU model is different from the Transformer-based models, we evaluate the performance of iGRU and its main competitor models, e.g., TimeXer, iTransformer, DLinear and PatchTST in Figure 6 using various input lengths. The results pinpoint the performance promotion of iGRU for longer input windows and its capability to leverage the information over the extended temporal context.

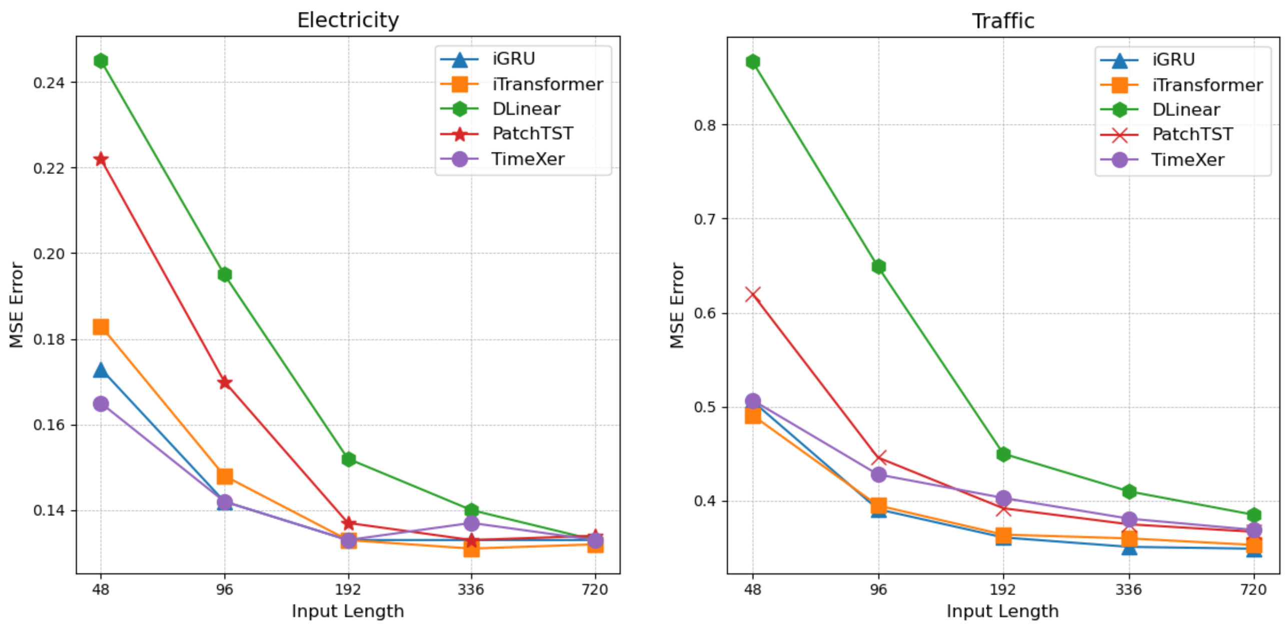

Figure 6. Forecasting with look-back length in {48, 96, 192, 336, 720} and prediction length of 96 on the Electricity and Traffic datasets. The proposed iGRU model exploits enlarged and shortened input lengths and delivers accurate results.

5. Conclusions

In this work, recurrent neural network architecture has been brought back to the time-series forecasting field by using it in a different manner and integrating it with a feed-forward layer. Experimentally, the proposed iGRU achieves results that can compete with those of the other state-of-the-art models, while consuming less training time and memory. In future work, we will investigate large-scale foundation models using the iGRU and perform a more in-depth exploration of time-series analysis.

Author Contributions: Conceptualization, V.N. and A.B.D.; methodology, V.N.; software, V.N.; validation, V.N., A.B.D. and M.B.; formal analysis, A.B.D. and V.N.; investigation, V.N., A.B.D. and M.B.; resources, A.B.D. and V.N.; writing—original draft preparation, V.N.; writing—review and editing, V.N., A.B.D. and M.B.; visualization, V.N.; supervision, A.B.D. and M.B.; project administration, A.B.D.; funding acquisition, A.B.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original data presented in the study are openly available at URL: https://drive.google.com/drive/folders/1ZOYpTUa82_jCcxIdTmyr0LXQfvaM9vIy and https://drive.google.com/drive/folders/1Gv1MXjLo5bLGep4bsqDyaNMI2oQC9GH2 (accessed on 22 April 2025).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Chen, Z.; Ma, M.; Li, T.; Wang, H.; Li, C. Long sequence time-series forecasting with deep learning: A survey. *Inf. Fusion* **2023**, *97*, 101819. [CrossRef]
2. Challu, C.; Olivares, K.G.; Oreshkin, B.N.; Ramirez, F.G.; Canseco, M.M.; Dubrawski, A. Nhits: Neural hierarchical interpolation for time-series forecasting. In Proceedings of the 37th AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2023; Volume 37, pp. 6989–6997.

3. Chen, S.-A.; Li, C.-L.; Yoder, N.; Arik, S.O.; Pfister, T. Tsmixer: An all-mlp architecture for time-series forecasting. *arXiv* **2023**, arXiv:2303.06053.
4. Zhou, T.; Niu, P.; Sun, L.; Jin, R. One fits all: Power general time-series analysis by pretrained LM. In Proceedings of the 37th International Conference on Neural Information Processing Systems, New Orleans, LA, USA, 10–16 December 2023; Advances in Neural Information Processing Systems 36 (NeurIPS 2023); Neural Information Processing Systems Foundation, Inc. (NeurIPS): San Diego, CA, USA, 2023; pp. 43322–43355.
5. Luo, D.; Wang, X. Moderntcn: A modern pure convolution structure for general time-series analysis. In Proceedings of the Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024; pp. 1–43.
6. Liu, M.; Zeng, A.; Xu, Q.; Zhang, L.; Chen, M.; Xu, Q. Scinet: Time-series modeling and forecasting with sample convolution and interaction. In Proceedings of the 36th International Conference on Neural Information Processing Systems, New Orleans, LA, USA, 28 November–9 December 2022; Advances in Neural Information Processing Systems 35 (NeurIPS 2022); Neural Information Processing Systems Foundation, Inc. (NeurIPS): San Diego, CA, USA, 2022; pp. 5816–5828.
7. Zeng, A.; Chen, M.; Zhang, L.; Xu, Q. Are transformers effective for time-series forecasting? In Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23), Washington, DC, USA, 7–14 February 2023; Volume 37, pp. 11121–11128.
8. Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; Long, M. Itransformer: Inverted transformers are effective for time-series forecasting. *arXiv* **2023**, arXiv:2310.06625.
9. Kitaev, N.; Kaiser, L.; Levskaya, A. Reformer: The efficient transformer. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
10. Li, Z.; Qi, S.; Li, Y.; Xu, Z. Revisiting long-term time-series forecasting: An investigation on linear mapping. *arXiv* **2023**, arXiv:2305.10721.
11. Lin, S.; Lin, W.; Wu, W.; Zhao, F.; Mo, R.; Zhang, H. Segrnn: Segment recurrent neural network for long-term time-series forecasting. *arXiv* **2023**, arXiv:2308.11200.
12. Hou, B.-J.; Zhou, Z.-H. Learning with interpretable structure from gated RNN. *IEEE Trans. Neural Networks Learn. Syst.* **2020**, *31*, 2267–2279. [CrossRef] [PubMed]
13. Elsaraiti, M.; Ali, G.; Musbah, H.; Merabet, A.; Little, T. time-series analysis of electricity consumption forecasting using ARIMA model. In Proceedings of the 2021 IEEE Green Technologies Conference (GreenTech), Denver, CO, USA, 7–9 April 2021; pp. 259–262.
14. Zhou, X.; Chen, W.; Wang, Y.X.; Yan, X. Enhancing the locality and breaking the memory bottleneck of transformer on time-series forecasting. In Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019; Advances in Neural Information Processing Systems 32 (NeurIPS 2019); Neural Information Processing Systems Foundation, Inc. (NeurIPS): San Diego, CA, USA, 2019.
15. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21), Virtually, 2–9 February 2021; Volume 35, pp. 11106–11115.
16. Wu, H.; Xu, J.; Wang, J.; Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Online, 6–14 December 2021; Advances in Neural Information Processing Systems 34 (NeurIPS 2021); Neural Information Processing Systems Foundation, Inc. (NeurIPS): San Diego, CA, USA, 2021; pp. 22419–22430.
17. Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; Jin, R. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In Proceedings of the 39th International Conference on Machine Learning PMLR 2022, Baltimore, MD, USA, 17–23 July 2022; pp. 27268–27286.
18. Liu, S.; Yu, H.; Liao, C.; Li, J.; Lin, W.; Liu, A.X.; Dustdar, S. Pyraformer: Low-complexity pyramidal attention for long-range time-series modeling and forecasting. In Proceedings of the Tenth International Conference on Learning Representations (ICLR 2022), Virtual, 25–29 April 2022.
19. Nie, Y.; Nguyen, N.H.; Sinthong, P.; Kalagnanam, J. A time-series is worth 64 words: Long-term forecasting with transformers. *arXiv* **2022**, arXiv:2211.14730.
20. Zhang, Y.; Yan, J. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time-series forecasting. In Proceedings of the Eleventh International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.
21. Das, A.; Kong, W.; Leach, A.; Mathur, S.; Sen, R.; Yu, R. Long-term forecasting with tide: Time-series dense encoder. *arXiv* **2023**, arXiv:2304.08424.
22. Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; Long, M. Timesnet: Temporal 2d-variation modeling for general time-series analysis. *arXiv* **2022**, arXiv:2210.02186.

23. Wang, Y.; Zhang, L.; Chen, M.; Xu, Q.; Zeng, A. Timexer: Empowering transformers for time-series forecasting with exogenous variables. In Proceedings of the Thirty-Eighth Annual Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 9–15 December 2024; Advances in Neural Information Processing Systems 37 (NeurIPS 2024); Neural Information Processing Systems Foundation, Inc. (NeurIPS): San Diego, CA, USA, 2024.
24. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.
25. Wang, C.; Liu, Z.; Wei, H.; Chen, L.; Zhang, H. Hybrid deep learning model for short-term wind speed forecasting based on time-series decomposition and gated recurrent unit. *Complex Syst. Model. Simul.* **2021**, *1*, 308–321. [CrossRef]
26. Lawi, A.; Mesra, H.; Amir, S. Implementation of long short-term memory and gated recurrent units on grouped time-series data to predict stock prices accurately. *J. Big Data* **2022**, *9*, 89. [CrossRef]
27. Kim, T.; Kim, J.; Tae, Y.; Park, C.; Choi, J.-H.; Choo, J. Reversible instance normalization for accurate time-series forecasting against distribution shift. In Proceedings of the International Conference on Learning Representations, Vienna, Austria, 4 May 2021.
28. Chen, C.; Petty, K.; Skabardonis, A.; Varaiya, P.; Jia, Z. Freeway performance measurement system: Mining loop detector data. *Transp. Res. Rec.* **2001**, *1748*, 96–102. [CrossRef]
29. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Advances in Neural Information Processing Systems 32 (NeurIPS 2019); Neural Information Processing Systems Foundation, Inc. (NeurIPS): San Diego, CA, USA, 2019.
30. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.