To what extent can the incorporation of Temporal Convolution into Transformer Improve the Efficiency on Time Series Prediction?

Brian Zu

To what extent can the incorporation of Temporal Convolution into Transformer Improve the Efficiency on Time Series Prediction?

Introduction

Time series prediction is a critical task in various domains such as finance, healthcare, weather forecasting, and transportation. It uses historical data to predict future. In recent years, attention mechanism was well known after being implemented in the transformer architecture [13]. Since 2017, there have been many works around this model in many tasks such as natural language processing [2], and computer vision [4] given its performance. In the field of time series prediction, many researchers found that the standard Transformer architecture may not fully exploit the temporal relationships and dependencies present in time series data. So there many variants of transformers appeared, including informer [17], patchTST [10], that all aim to improve its performance. In this work, we propose a generalized architecture, where the input data can be partially extracted. Then, we used an example of using CNN and Transformer in this architecture, the TCformer, and experimented it's efficiency and performance.

Related work

Transformer architecture

Despite the impressing works above, the transformer architecture comes with its great size. Many other works in training large scaled transformer models appeared with up to 600B weights that cost for 2048 TPU v3 cores [6]. Recent work [7] has reduced the size with compression with the cost of precision. In fact, currently researchers have no way to get around with the bulky model without compromises. Seeing this flaw, we proposed a new hybrid architecture TCformer that both uses CNN and Transformer to achieve an improvement in precision with reduction in size for multivariate time series prediction task.

Considering the architecture of Transformer, different from convolution layers, its parameter size is inherently dependent on the size of the input. In Transformer, the scaled dot-product attention [13] computes the output as:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}}V) \tag{1}$$

Where the Q represents the query matrix, K the key matrix, and V value matrix. In time series prediction task, those matrices are 3D tensors that has a shape of:

$$Q,K,V \in \mathbb{R}^{[B\times L\times (D/H)]} \tag{2}$$

where B is the batch size, H the number of heads, L the sequence length, D the numbers of features. Thus, they are all dependent on the input size. Even though it can be addressed through a linear layer that maps the dimension lower, it's not a common approach due to the huge information loss through this process which decreases the model's precision. This attention mechanism itself already made this model large, not to mention the parameters in the FFN (Feed Forward Network).

Due to this full attention mechanism, this leads to the quadratic dependency of Transformer-based models [16], which makes the computational and memory dependent on the sequence length. With that being said, reducing the dimension while remaining the precision of attention mechanism became the key in reducing the model's size.

Flash attention has made a significant break through in reducing the space complexity to linear and using SRAM to reduce IO complexity. Different from the algorithmic improvement, we made the model more efficient by changing the architecture without changing the specific layers. These two approaches doesn't conflict, and can be used simultaneously. However, in our work we used the most basic encoder layer for simplicity.

Convolution-based architectures

Though Transformer has gained the most focus after its success in Large Language Models (LLMs) and was mostly researched on in doing various tasks including times series, convolution has also been applied on time series [5]. From the two base mechanisms, the time

series models have diverged into two groups as more models are proposed based on either CNN or Transformer. Moreover, recent work [12] has shown that CNN are more accurate than Transformer model, reaching 92% accuracy compared to the 80% accuracy of Transformer.

However, pure convolutional architectures are hard to deal with time series data. For each kernel, relatively small to the whole sequence, can only see a very limited range of data and can have a overview of a long series only when the network is deep enough and the pieces of traits converges at the top of the network. Thus its hard for those architectures expand it's capacity in processing long sequences. Moreover, CNN has much more hyperparameters on each layer, such as kernel size, stride and padding. This makes it hard to build a model that's at it's maximum performance. In our work, the CNN is used for simply extracting general features in the past data which only requires one to two layers of it, and leaves the rest of the prediction task to the Transformer encoder.

Convolution attention combined architectures

The combination of convolution layers with attention mechanism isn't a rare approach. In ConTNet, researchers have embedded the encoding block in between the convolution layers [15] for computer vision task. In time series prediction, Informer [17] has changed the encoder by adding convolution layers between the attention layers for handling longer time series sequences.

Unlike previous works, the TCformer we proposed neither embed convolutional layer in between the model nor apply it on the whole input.

Inspiration

We originally used Transformer solely on stock prediction task, and were surprised how poorly it performs in the prediction compared to other models.

(a) *LSTM*

(b) *Transformer*

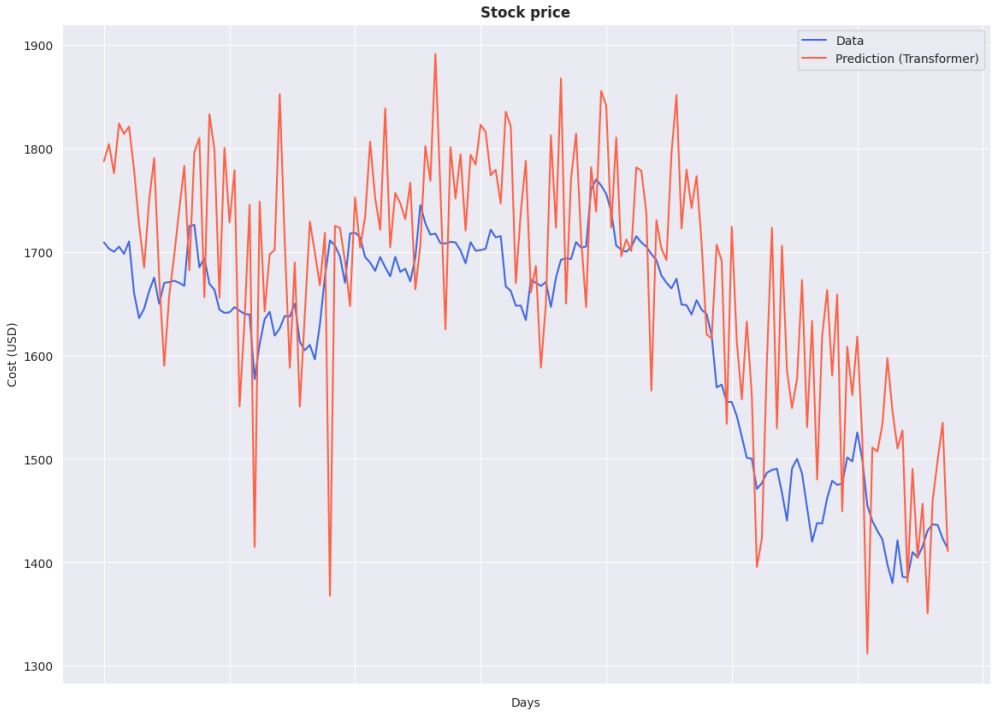

Figure 1

Stock price prediction comparison

We ran both LSTM and Transformer on a Chinese stock with code 600519. On the left of fig.1, LSTM predicted relatively closely and shows a smooth trend. However, on the right, Transformer's prediction had shown a significant variation with sharp turning points that occurred frequently. Numerically, LSTM's MSE test loss is about 91.84% lower than Transformer's. We reflected on why had the prevalent Transformer model failed in such task. After trying a range of hyperparameters, the Transformer's performance had some minor fluctuation, but the sharp fluctuation remains. By examining the prediction graph of Transformer, we found out that the general trend for Transformer is correct. Thus, the high loss attribute to the sharp peaks and troughs. Conceptually, we guessed it's the attention mechanism that led to this issue.

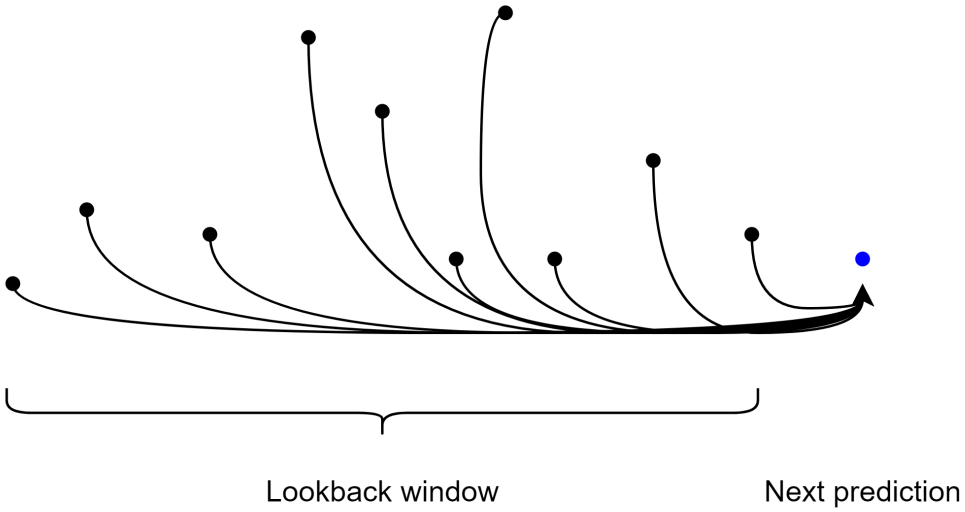

Figure 2

Attention in time series

In fig.2, it illustrates the attention mechanism in time series. Every point in the lookback window (black) will have a weighted contribution to the probability of next prediction (blue). While this might be successful when the data points are word tokens in Natural Language Processing (NLP), this use of information is too detailed and specific in time series task. The model will need consider every single data points which might fluctuates, leading to the the increased fluctuation in prediction. As the series increase, there will be more irrelevant

fluctuations captured by Transformer while failing to capture the general trend.

In order to address this problem, we used CNN to "summarize" and extract the input features. This can compress a range k data points into a general one, where k is the CNN kernel size while leaving the recent data (last part of data) uncompressed.

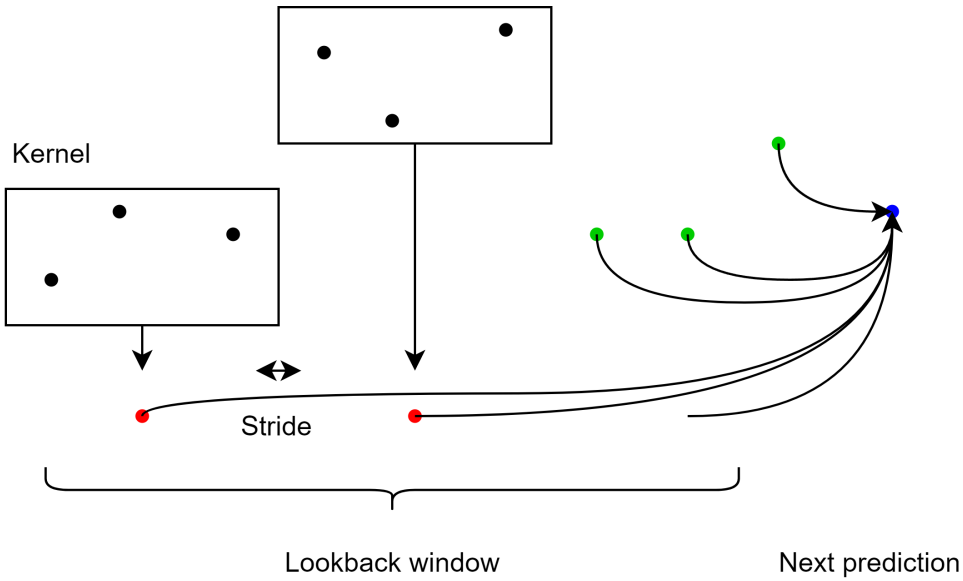

Figure 3

TCformer attention for time series

In fig.3, the kernel of shape 3 is extracting the feature of 3 data points and mapping it into one (red). The distance kernel moves forward is dependent on the stride.

The extracted data points (red) along with the raw (green) is then applied attention in Transformer. This not only reduces the sequence length, but also captures the trend in previous data.

However, this design is based on an important assumption: more recent data in the dataset needs to have be more important. This allows us split the data. This applies to many different datasets, such as traffic, solar, electricity, stock and so on. Conversely, in language models where the input might refer to words in a very early context would not be expected to have an improved precision.

TCformer

The architecture we propose is not a specific one but a general idea on how current models can preprocess the input tensor for better efficiency. In this work, we used a simple Encoder for the base model, and used CNN for preprocessing the input. For the most common single-step forecasting task, where the model predicts next timestamp X_{n+1} by observing the historical data $X_1, ..., X_n$.

Structure Overview

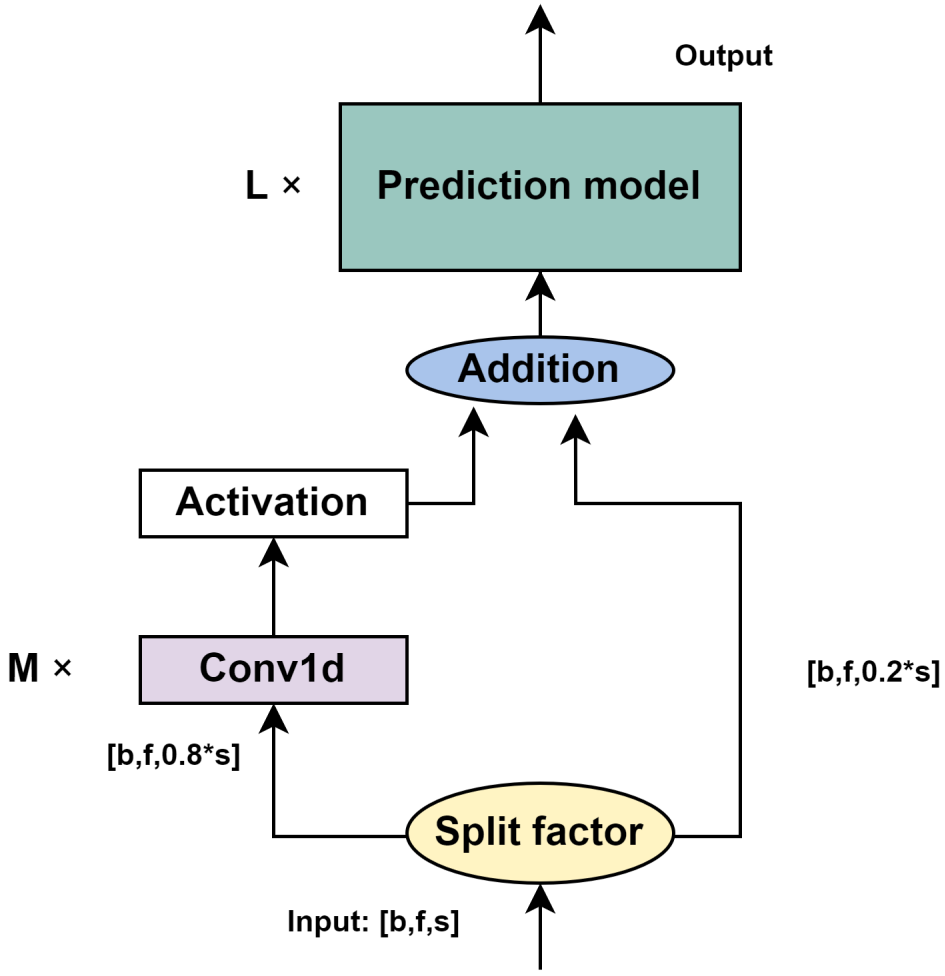

Figure 4

TCformer sample structure

The TCformer has a simple and flexible structure. In general, we use CNN for extracting the first $p\%$ of the input data, leaving the rest raw, and feeds it to the prediction model. On fig 4, the split factor is a scalar p that splits the dataset into to on the dimension of sequence length. Note that the split factor is a hyperparameter that needs to be manually assigned. For example, a scalar $p = 0.8$ would split the input of shape $X \in \mathbb{R}^{b,f,s}$ into $X \in \mathbb{R}^{b,f,0.8\cdot s}$ and $X \in \mathbb{R}^{b,f,0.2\cdot s}$ where:

1. b is the batch size
2. f is the dimension of feature
3. s is the sequence length

On fig.4, there can be M Conv1d layer. They are 1-Dimensional convolution layers that extract traits from the $p\%$ of the input and reduces the sequence length, where p is the split factor and it's set to 0.8 in this illustration. There could be multiple layers added flexibly. It can be replaced by a Conv2d (needs to ensure that the dimension of the model doesn't reduce). The right branch leaves the raw data that will be joined back with the left branch in the concat component. In concat component the two branches' tensors will be merged on the third dimension s. Lastly, the concatenated tensor will be the input for the prediction model.

Convolution extraction

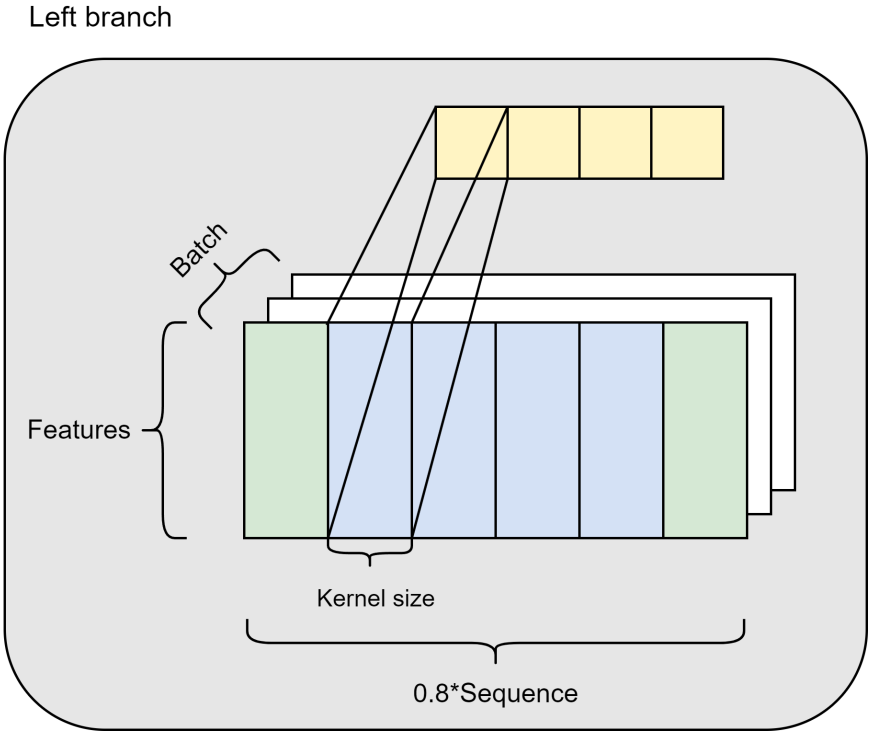

Figure 5

Convolutional sequence summary on left branch

In the left branch illustrated in 5, Conv1d layer is outputting summaries for the first 80% of earlier data, leaving the rest 20% raw for the Transformer encoder to process. This will cause the input X to change it's third dimension's shape s from:

$$s_1 = 0.8 \cdot s \tag{3}$$

$$s_2 = \lfloor \frac{0.8 \cdot s - k}{r} \rfloor + 1 \tag{4}$$

1. s_1 is the sequence length before convolution layer
2. k is kernel size
3. s_2 is the sequence length after convolution layer

4. r is the stride

On the right branch, the raw data is not processed. Given that the Transformer has already have a good performance on raw data, applying convolution is not needed and might reduce the information perceived in the encoder block.

Sequence length reduction

After the addition, the input for the encoder block will have a sequence length of:

$$s_3 = s_2 + 0.2 \cdot s \tag{5}$$

$$= \lfloor \frac{0.8 \cdot s - k}{r} \rfloor + 1 + 0.2 \cdot s \tag{6}$$

The reduction of sequence length s_3 compared to s can be calculated:

$$l = s - s_3 \tag{7}$$

$$= s - \lfloor \frac{0.8 \cdot s - k}{r} \rfloor - 1 - 0.2 \cdot s \tag{8}$$

Notice l is positively related with k and r, changing the stride and kernel size can give a flexible control on the model size. In the attention block, a reduction of l will reduce:

$$l \times b \times f \tag{9}$$

in each of the Q, K, V matrices, largely reducing it's parameters.

Experiments

We experimented the model on both single step prediction and sequence to sequence (seq2seq) task.

Experiments for single-step prediction

We had conducted experiments on 3 real world datasets in the experiment, including weather used in Autoformer [14] and AMD stock data.

Dataset

The weather dataset included 52696 rows of data with 21 covariates.

The AMD stock dataset is accessed through Baostock API [1]. It includes 10995 rows of data with 5 covariates. The data spans from 2nd of January 1981 to 14th of August 2024.

Experiment setup

Transformer model has a hidden dimension of 8, 2 layers, 2 heads for attention. For all models the optimizer uses Adam, with lr=1e-3, wd=1e-4, dropout=0.1, training in 5 epochs with batch size of 512 with train test 9 : 1 splitting. The TCformer uses split factor of 0.8 with the CNN using kernel size 6 with a stride of 6. The criteria uses MSE (Mean Squared Error).

Results

Table 1

AMD stock data test MSE loss

TCformer1d	transformer	seq_len	pct_improve(%)
0.018490	0.073744	16	74.93
0.015125	0.061805	32	75.53
0.012830	0.053022	64	75.80
0.017049	0.069303	128	75.40
0.016919	0.070958	256	76.16
0.017332	0.072824	512	76.20

In table 1, for AMD stock data, the TCformer with 1d in average has improved performance by about 75%, while having reduced size of model. During experiment, the TCformer was successfully trained on dataset when sequence length is 1024 however the Transformer model failed due to GPU memory overflow, so the sequence length for the experiment is limited up to 512.

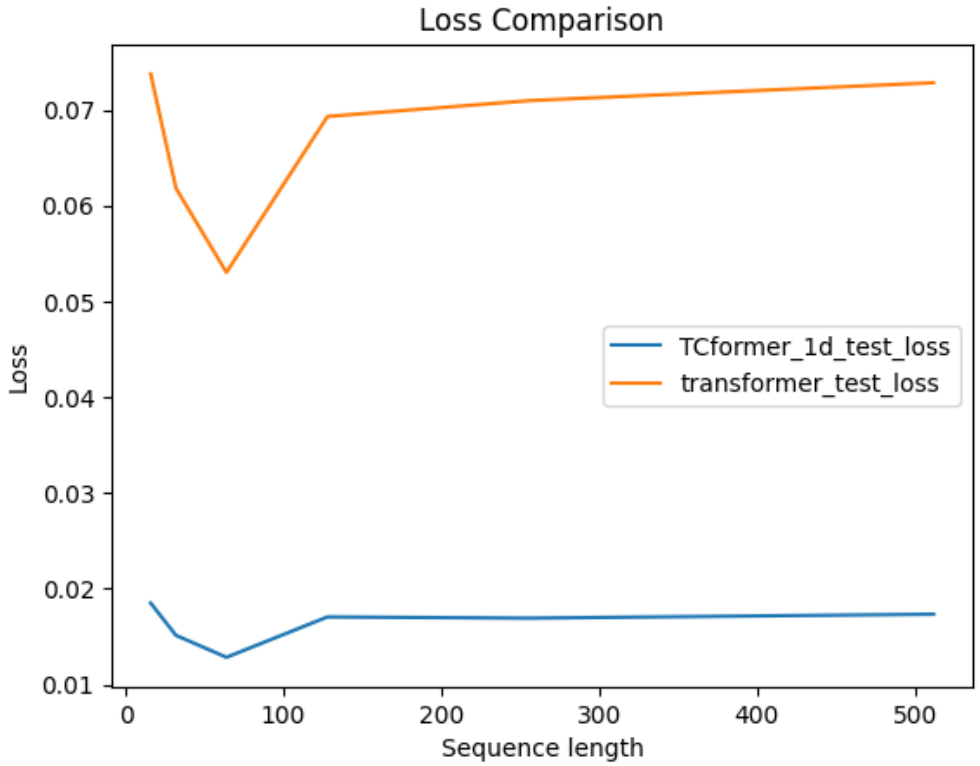

Figure 6

AMD loss curve

On fig.6, the test loss for both models both has it's minimum at a sequence length of 64. For every sequence length, TCformer has a much lower lost compared to Transformer. The Transformer loss shows an increase as sequence length increases from 128 to 512 while this is relatively not as significant in TCformer's loss. This shows that the sequence reduction has enabled TCformer to handle longer sequence while remaining similar precision.

Table 2

Weather data test MSE loss

TCformer1d	transformer	seq_len	pct_improve (%)
0.026144	0.104435	16	74.97
0.026144	0.104253	32	74.92
0.026188	0.103928	64	74.80
0.025542	0.101652	128	74.87
0.028019	0.107837	256	74.02
0.026371	0.104923	512	74.87

Like the results in table 1, the percentage improved is around 74%. Given that Transformer has a poor performance, this great improvement is within expectation.

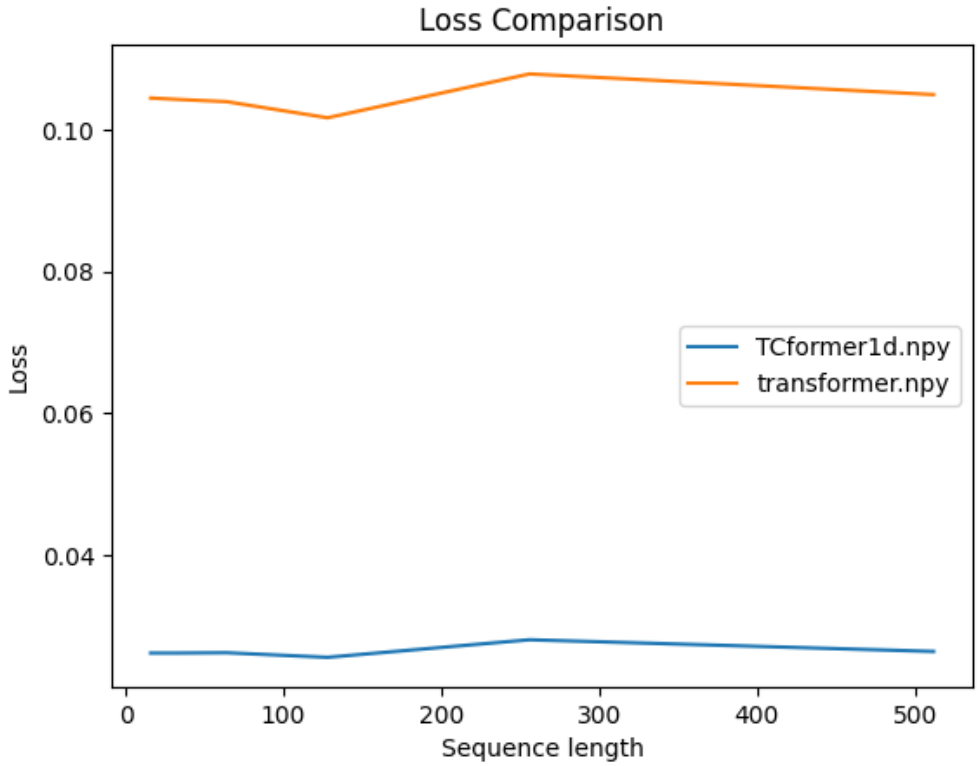

Figure 7

Weather loss curve

Experiments for sequence to sequence

Dataset

For sequence to sequence, we used 7 datasets (ECL,ETT subset 1,Exchange, Traffic, Weather, and Solar energy), the same with the experiment for iTransformer [8]. For each dataset, we ran Transformer, iTransformer, and TCiTransformer (iTransformer with CNN). Since iTransformer currently has the best accuracy on different datasets, we used it as our prediction model, simply replacing the TCformer's Transformer Encoder with iTransformer Encoder.

The dataset covers a range of sample and feature size:

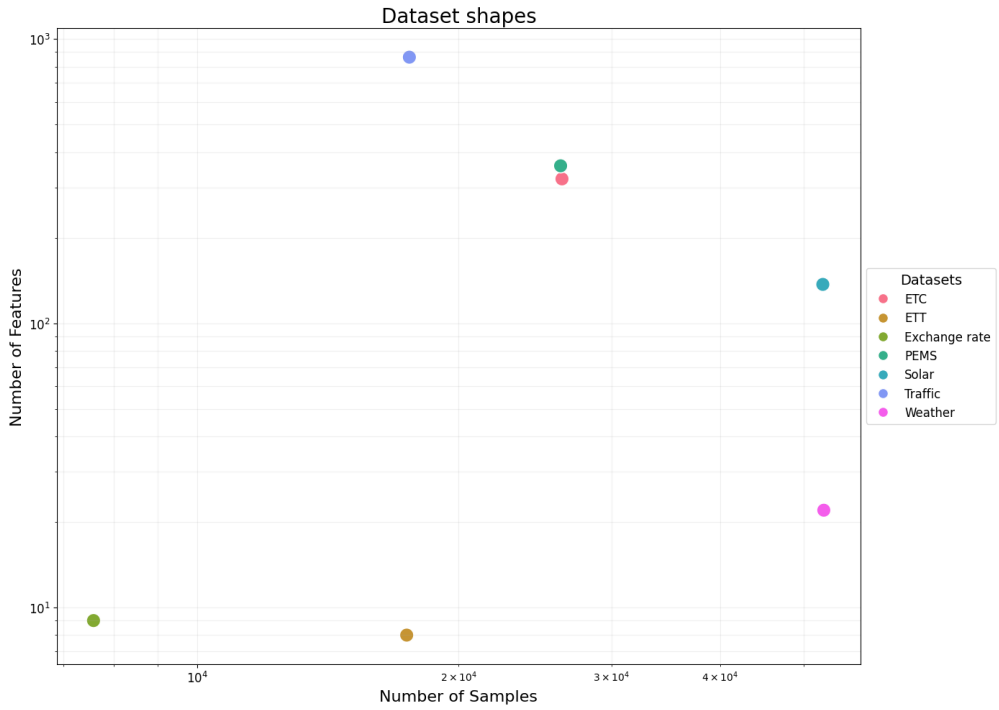

Figure 8

Dataset shapes

Experiment setup

For each model, we set the prediction length to $P = \{96, 192, 336\}$ and lookback length to 96. Since PEMS is too large, we set the lookback 48 and prediction length to $P = \{12, 24, 48\}$. We set two layers of CNN, and split factor to 0.8 for the iTransformer with CNN. The kernel of first layer is 5, stride 2 and second layer is 3, stride 1. We had experimented the model with different hyperparameters such as the CNN kernel size, stride, the split factors and so on. The impact on the performance isn't obvious, so we set two layers to have kernel capturing 9 timestamps at each time. This ensures that in most of the dataset, it doesn't cover a whole period but extracts the trends that exists in a period and then giving it to the prediction model. The kernel isn't set too small to ensure that it can reduce the sequence length.

Results

Dataset	Model	MSE	MAE	Input length	Output length
ECL	Transformer	0.249665	0.349949	96	96
ECL	iTransformer	0.167462	0.256914	96	96
ECL	TCiTransformer	0.180147	0.280937	96	96
ETTh1	Transformer	0.796016	0.702380	96	96
ETTh1	iTransformer	0.387790	0.406386	96	96
ETTh1	TCiTransformer	0.393983	0.415327	96	96
Exchange	Transformer	0.677474	0.637843	96	96
Exchange	iTransformer	0.086314	0.206543	96	96
Exchange	TCiTransformer	0.086183	0.207386	96	96
PEMS03	Transformer	0.105133	0.204023	48	12
PEMS03	iTransformer	0.073994	0.179749	48	12
PEMS03	TCiTransformer	0.066601	0.170845	48	12
solar	Transformer	0.198320	0.232168	96	192
solar	iTransformer	0.212609	0.245238	96	96
solar	TCiTransformer	0.224425	0.245268	96	96
traffic	Transformer	0.665396	0.365834	96	96
traffic	iTransformer	0.465135	0.323030	96	96
traffic	TCiTransformer	0.546155	0.343953	96	192
weather	Transformer	0.221558	0.310034	96	96
weather	iTransformer	0.183019	0.224638	96	96
weather	TCiTransformer	0.167566	0.213238	96	96

Table 3

Summary table for loss (selecting lowest MSE)

For each model on each dataset, we selected it's the best performing results from the varying output sequence length. In table 3, iTransformer1dSplit is the iTransformer with CNN. For seq2seq task, the TCiTransformer didn't have much increase compared to iTransformer in most datasets. In ECL, ETTh1, solar and traffic, the model's precision is lower than iTransformer. However, in PEMS03, Exchange and weather, TCiTransformer had a higher precision.

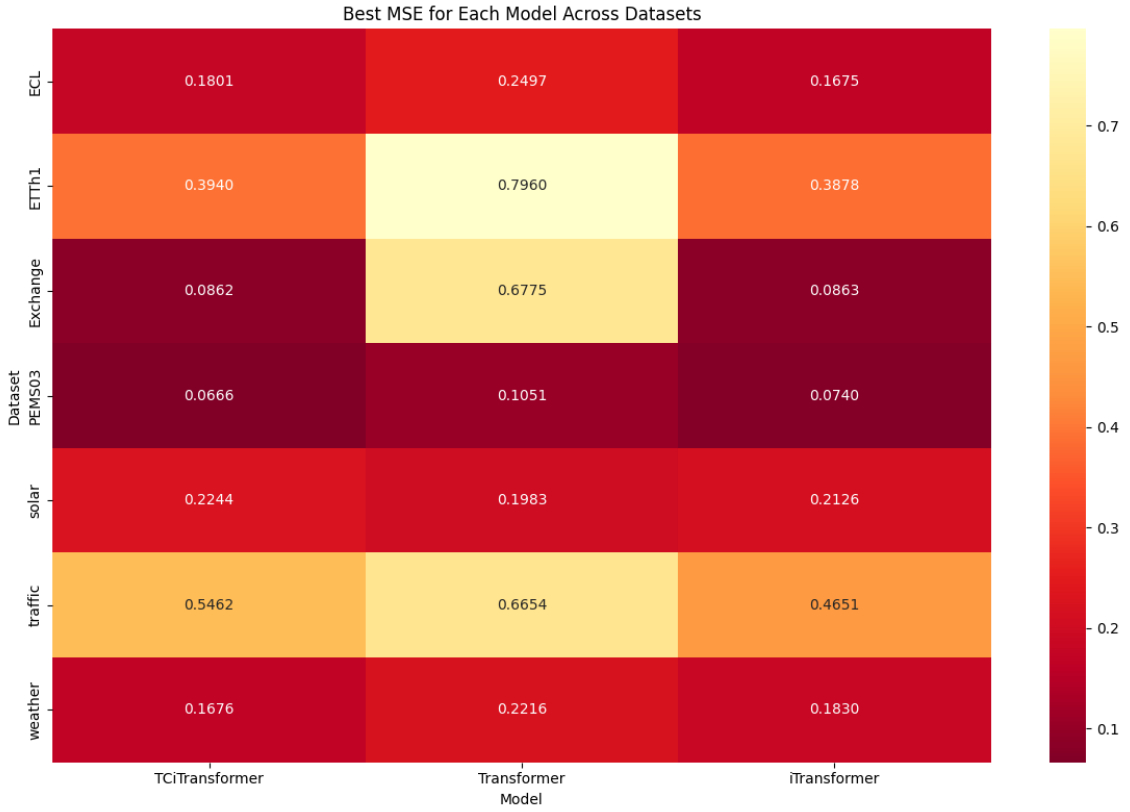

Figure 9

MSE heatmap

This is illustrated clearer in fig.9. The darker means the better the model is.

We selected the prediction of our model on traffic, which it's performing worse, and weather which it performs better:

(a) *traffic* (b) *weather*

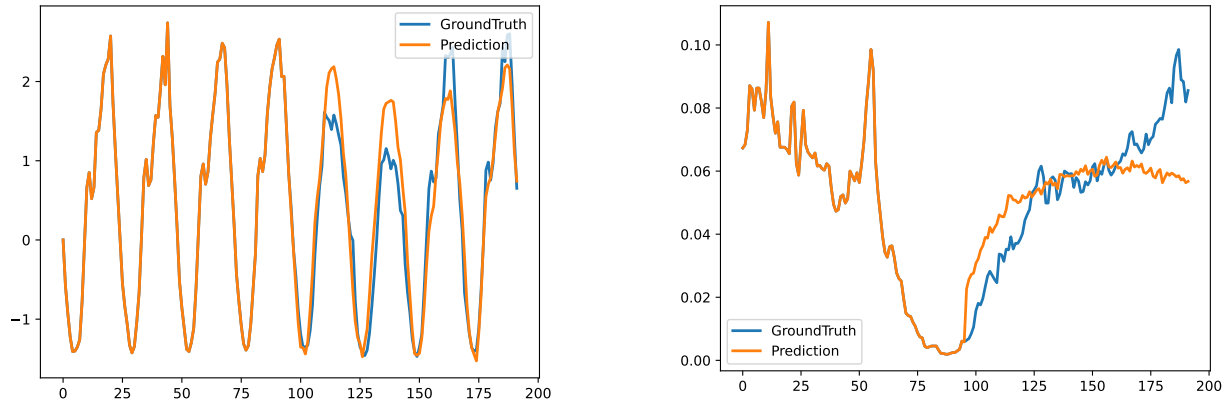

Figure 10

Prediction on traffic and Weather for both input and output length is 96

The traffic data is much more periodic with having a generally regular pattern, while the weather data is more irregular. By observing other datasets, we found out that TCiTransformer is better in non-periodic time series forecasting. This might be because the CNN cannot be trained to capture the increase and decrease over a small range of time, while it gives the general trend in datasets such as weather.

Efficiency

Although the sequence length can be reduced, as mentioned in section , the CNN requires time for operation. Depending on the kernel size and stride, the time for TCformer architecture can vary. In the experiments above, we recorded the training time for traffic data, using the same kernel size and stride:

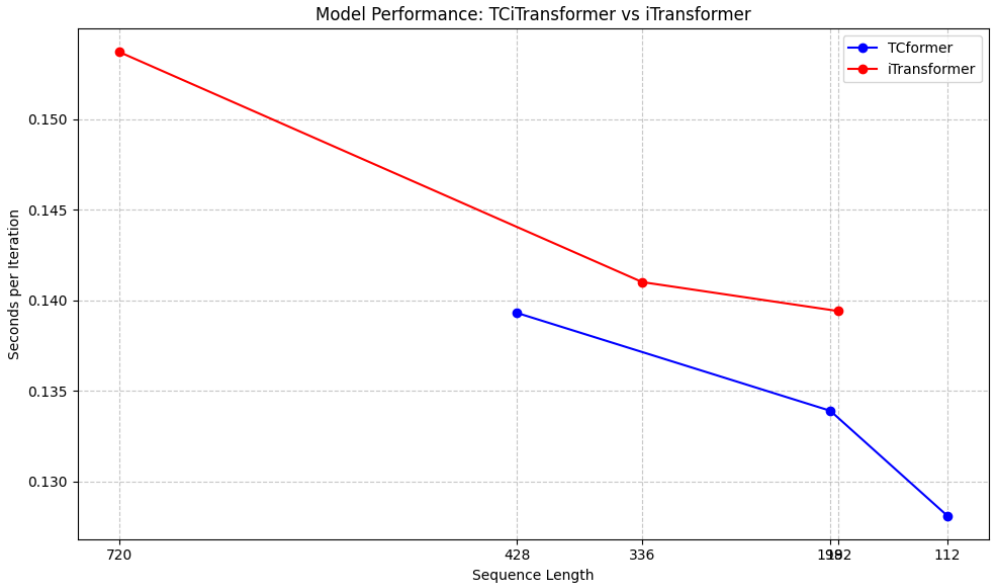

Figure 11

Speed comparison on different sequence length scale

Fig.11 shows the speed of TCiTransformer against iTransformer. Because TCformer architecture reduces the sequence length, so there is a right shift of the blue line on the sequence length. However, even at a same sequence length did TCiTransformer outperform iTransformer.

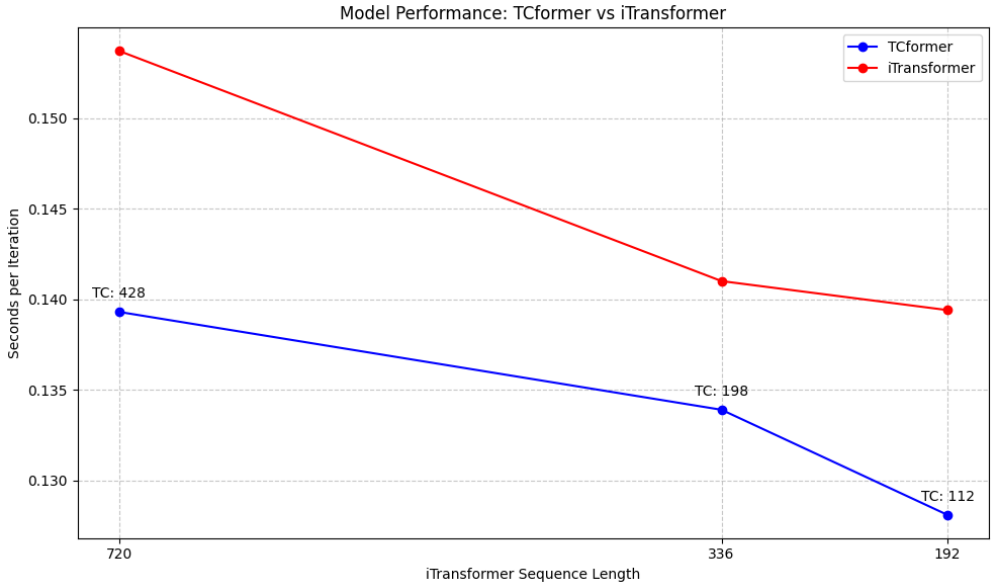

Figure 12

Speed comparison on same sequence length scale

Fig.12 plots the speed of model on a same sequence length scale. It shows at the same input sequence length, the reduced encoder block input will lead to an increase in speed, despite the cost of CNN calculation.

Conclusion

The TCformer architecture overall is successful in addressing the limitations of both pure convolutional and transformer models. By integrating temporal convolution with transformer-based attention, TCformer achieves superior predictive performance in single-step prediction task while maintaining computational efficiency. The model's ability to handle longer sequences and its consistent performance improvements across different datasets suggest its potential for wide applicability in various time series prediction tasks. As datasets continue to grow in size and complexity, current models need to increase their model dimension to increase handle longer sequence length. However, with TCformer, the prediction model receives a reduced sequence length. Lastly, the model offers a flexibility in determining the split factor, layers of CNN and the prediction model, allowing for researchers to use this general architecture in various forms.

The improvement in performance, in general, didn't exist in seq2seq task. The mediocre performance suggests that using CNN to extract features might not be helpful for predicting periodic datasets, as mentioned in section . However, TCformer could still be applied for better efficiency.

Future works

Because TCformer is an architectural change, rather than a change in components, it's compatible with the whole family of Transformer models, such as Autoformer [14], informer [17] and so on. Future works can be done to investigate TCformer's impact on these models. Not only transformer, but other family of models such as RNN, LSTM, Mamba [3] can incorporate this idea for data processing.

In this work, we applied 1D CNN layers for feature extraction. 2D CNN layers could be used in substitution to allow blocks of data, like patches in PatchTST [11] and extract the input

data across different features.

We believe this idea of "compressing" the inputs partially can become a generalized approach in time series task for better model efficiency while improving or maintaining the precision. There could be other better approaches to "compress" the data partially, other than CNN, to improve the performance and efficiency of current models.

Since TCformer had a great result in single-step prediction task, changing the encoder only architecture of iTransformer and use decoders to make the model autoregressive might bring the improved performance for TCformer in seq2seq tasks.

Moreover, Kolomogorov-Arnold Networks (KAN) can be more expressive in each neurons, so using it in TCformer feed forward network might be also a way to reduce the parameters [9].

Rather than changing the architecture, future works can also explore how to utilize the idea at component level, such as incorporating it in attention mechanism.

Appendix A

Full result

The full result of seq2seq task:

Table A1

Performance comparison of different models across various datasets

Dataset	Model	Input_length	Output_length	MSE	MAE
ECL	Transformer	96	96	0.311263	0.398829
ECL	Transformer	96	192	0.344802	0.426795
ECL	Transformer	96	336	0.348473	0.427140
ECL	Transformer	720	336	0.360726	0.429443
ECL	Transformer	336	336	0.354477	0.429831
ECL	Transformer	192	336	0.356725	0.433641
ECL	iTransformer	96	96	0.189476	0.275656
ECL	iTransformer	96	192	0.200119	0.287542
ECL	iTransformer	96	336	0.220469	0.307598
ECL	iTransformer	96	336	0.220469	0.307598
ECL	iTransformer	336	336	0.191621	0.291078
ECL	iTransformer1dSplit	96	96	0.195807	0.295794
ECL	iTransformer1dSplit	96	192	0.209703	0.306293
ECL	iTransformer1dSplit	96	336	0.229005	0.324030
ECL	iTransformer1dSplit	720	336	0.214763	0.322888
ECL	iTransformer1dSplit	336	336	0.218014	0.321793
ECL	iTransformer1dSplit	192	336	0.209419	0.312572
ETTh1	Transformer	96	96	0.903719	0.770211
ETTh1	Transformer	96	192	0.892692	0.757590

Continued on next page

Table A1 – *Continued from previous page*

Dataset	Model	Input_length	Output_length	MSE	MAE
ETTh1	Transformer	96	336	0.982860	0.801562
ETTh1	iTransformer	96	96	0.389884	0.406386
ETTh1	iTransformer	96	192	0.446103	0.440019
ETTh1	iTransformer	96	336	0.487570	0.462899
ETTh1	iTransformer1dSplit	96	96	0.393983	0.415327
ETTh1	iTransformer1dSplit	96	192	0.446689	0.444733
ETTh1	iTransformer1dSplit	96	336	0.488159	0.467200
Exchange	Transformer	96	96	0.677474	0.665115
Exchange	Transformer	96	192	1.226818	0.894968
Exchange	Transformer	96	336	1.227765	0.949669
Exchange	iTransformer	96	96	0.086843	0.206637
Exchange	iTransformer	96	192	0.182796	0.305740
Exchange	iTransformer	96	336	0.334583	0.420060
Exchange	iTransformer1dSplit	96	96	0.086183	0.207386
Exchange	iTransformer1dSplit	96	192	0.177637	0.300582
Exchange	iTransformer1dSplit	96	336	0.364597	0.432012
PEMS03	Transformer	48	12	0.116023	0.216098
PEMS03	Transformer	48	24	0.134372	0.237528
PEMS03	Transformer	48	48	0.141914	0.245927
PEMS03	iTransformer	48	12	0.074810	0.181082
PEMS03	iTransformer	48	24	0.115063	0.225679
PEMS03	iTransformer	48	48	0.211413	0.308824
solar	Transformer	96	96	0.217859	0.232168

Continued on next page

Table A1 – *Continued from previous page*

Dataset	Model	Input_length	Output_length	MSE	MAE
solar	Transformer	96	192	0.198320	0.241451
solar	Transformer	96	336	0.216907	0.252444
solar	iTransformer	96	96	0.213345	0.254571
solar	iTransformer	96	192	0.247524	0.283152
solar	iTransformer	96	336	0.263141	0.294588
solar	iTransformer1dSplit	96	96	0.224425	0.245268
traffic	Transformer	96	96	0.729950	0.415999
traffic	Transformer	96	192	0.748270	0.420306
traffic	iTransformer	96	96	0.552741	0.375079
traffic	iTransformer	96	192	0.571267	0.385067
traffic	iTransformer1dSplit	96	96	0.554344	0.359170
traffic	iTransformer1dSplit	96	192	0.636189	0.405080
weather	Transformer	96	96	0.454535	0.469195
weather	Transformer	96	192	0.417705	0.463784
weather	Transformer	96	336	0.544155	0.548513
weather	Transformer	720	336	0.483711	0.508564
weather	Transformer	336	336	0.322639	0.396713
weather	Transformer	192	336	0.363218	0.425497
weather	iTransformer	96	96	0.191416	0.231146
weather	iTransformer	96	192	0.236874	0.268680
weather	iTransformer	96	336	0.290634	0.306938
weather	iTransformer	720	336	0.251563	0.288584
weather	iTransformer	336	336	0.255852	0.289504

Continued on next page

Table A1 – *Continued from previous page*

Dataset	Model	Input_length	Output_length	MSE	MAE
weather	iTransformer	192	336	0.270618	0.297156
weather	iTransformer1dSplit	96	96	0.173825	0.219965
weather	iTransformer1dSplit	96	192	0.222881	0.262845
weather	iTransformer1dSplit	96	336	0.281121	0.304419
weather	iTransformer1dSplit	720	336	0.246249	0.287646
weather	iTransformer1dSplit	336	336	0.255297	0.290994
weather	iTransformer1dSplit	192	336	0.262164	0.293423
ECL	Transformer	96	96	0.249665	0.349949
ECL	Transformer	96	192	0.259071	0.358531
ECL	Transformer	96	336	0.284718	0.378518
ECL	Transformer	720	336	0.302568	0.390787
ECL	Transformer	336	336	0.309325	0.401690
ECL	Transformer	192	336	0.277550	0.372583
ECL	iTransformer	96	96	0.167462	0.256914
ECL	iTransformer	96	192	0.178564	0.266712
ECL	iTransformer	96	336	0.196589	0.284522
ECL	iTransformer	96	336	0.196589	0.284522
ECL	iTransformer	336	336	0.169321	0.266442
ECL	iTransformer1dSplit	96	96	0.180147	0.280937
ECL	iTransformer1dSplit	96	192	0.186350	0.284769
ECL	iTransformer1dSplit	96	336	0.201602	0.300528
ECL	iTransformer1dSplit	720	336	0.196832	0.303797
ECL	iTransformer1dSplit	336	336	0.207599	0.311820

Continued on next page

Table A1 – *Continued from previous page*

Dataset	Model	Input_length	Output_length	MSE	MAE
ECL	iTransformer1dSplit	192	336	0.194282	0.294770
ETTh1	Transformer	96	96	0.796016	0.702380
ETTh1	Transformer	96	192	1.266312	0.898232
ETTh1	Transformer	96	336	1.298750	0.932393
ETTh1	iTransformer	96	96	0.387790	0.407319
ETTh1	iTransformer	96	192	0.439360	0.435550
ETTh1	iTransformer	96	336	0.490981	0.462162
ETTh1	iTransformer1dSplit	96	96	0.410717	0.422466
ETTh1	iTransformer1dSplit	96	192	0.450122	0.446277
ETTh1	iTransformer1dSplit	96	336	0.498669	0.467550
Exchange	Transformer	96	96	0.681726	0.637843
Exchange	Transformer	96	192	1.183335	0.833934
Exchange	Transformer	96	336	1.767208	1.057534
Exchange	iTransformer	96	96	0.086314	0.206543
Exchange	iTransformer	96	192	0.180865	0.304338
Exchange	iTransformer	96	336	0.343557	0.426245
Exchange	iTransformer1dSplit	96	96	0.087456	0.209841
Exchange	iTransformer1dSplit	96	192	0.179929	0.306283
Exchange	iTransformer1dSplit	96	336	0.367684	0.437366
PEMS03	Transformer	48	12	0.105133	0.204023
PEMS03	Transformer	48	24	0.118884	0.222696
PEMS03	Transformer	48	48	0.147279	0.249651
PEMS03	iTransformer	48	12	0.073994	0.179749

Continued on next page

Table A1 – *Continued from previous page*

Dataset	Model	Input_length	Output_length	MSE	MAE
PEMS03	iTransformer	48	24	0.172277	0.270824
PEMS03	iTransformer	48	48	0.792518	0.641669
PEMS03	iTransformer1dSplit	48	12	0.066601	0.170845
solar	Transformer	96	96	0.214818	0.235886
solar	Transformer	96	192	0.216327	0.240304
solar	Transformer	96	336	0.224272	0.257986
solar	iTransformer	96	96	0.212609	0.245238
solar	iTransformer	96	192	0.240658	0.269337
solar	iTransformer	96	336	0.251896	0.277082
traffic	Transformer	96	96	0.665396	0.368861
traffic	Transformer	96	192	0.666415	0.365834
traffic	iTransformer	96	96	0.465135	0.323030
traffic	iTransformer	96	192	0.480384	0.326609
traffic	iTransformer1dSplit	96	96	0.554344	0.359170
traffic	iTransformer1dSplit	96	192	0.546155	0.343953
weather	Transformer	96	96	0.221558	0.310034
weather	Transformer	96	192	0.350565	0.414695
weather	Transformer	96	336	0.483071	0.495829
weather	Transformer	720	336	0.428837	0.453114
weather	Transformer	336	336	0.362688	0.417207
weather	Transformer	192	336	0.333627	0.398888
weather	iTransformer	96	96	0.183019	0.224638
weather	iTransformer	96	192	0.229880	0.262471

Continued on next page

Table A1 – *Continued from previous page*

Dataset	Model	Input_length	Output_length	MSE	MAE
weather	iTransformer	96	336	0.283890	0.300866
weather	iTransformer	720	336	0.269172	0.303571
weather	iTransformer	336	336	0.254031	0.289478
weather	iTransformer	192	336	0.266545	0.294338
weather	iTransformer1dSplit	96	96	0.167566	0.213238
weather	iTransformer1dSplit	96	192	0.213299	0.255012
weather	iTransformer1dSplit	96	336	0.272819	0.297549
weather	iTransformer1dSplit	720	336	0.246176	0.288838
weather	iTransformer1dSplit	336	336	0.254880	0.294645
weather	iTransformer1dSplit	192	336	0.259389	0.290187
traffic	iTransformer1dSplit	720	336	0.493737	0.354944
traffic	iTransformer1dSplit	336	336	0.538614	0.366240
traffic	iTransformer1dSplit	192	336	0.592530	0.384399

Appendix B

Code

We constructed the code using PyTorch 2 and python 3.8. The seq2seq task is based on the code base for iTransformer [8]. The full code is available on GitHub:

`https://github.com/Friedforks/EE-v2`.

Appendix C

*

References

[1] BaoStock. *BaoStock*. BaoStock.com, 2024. URL: http://baostock.com/baostock/index.php/%E9%A6%96%E9%A1%B5.

[2] Tom Brown et al. "Language Models are Few-Shot Learners." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[3] Tri Dao and Albert Gu. *Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality*. 2024. arXiv: 2405.21060 [cs.LG]. URL: https://arxiv.org/abs/2405.21060.

[4] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2020. eprint: arXiv:2010.11929.

[5] Pradeep Hewage et al. "Temporal convolutional neural (TCN) network for an effective weather forecasting using time-series data from the local weather station." In: *Soft Computing* 24 (2020), pp. 16453–16482.

[6] Dmitry Lepikhin et al. *GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding*. 2020. arXiv: 2006.16668 [cs.CL]. URL: https://arxiv.org/abs/2006.16668.

[7] Zhuohan Li et al. "Train Big, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers." In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 5958–5968. URL: https://proceedings.mlr.press/v119/li20m.html.

[8] Yong Liu et al. *iTransformer: Inverted Transformers Are Effective for Time Series Forecasting*. 2023. arXiv: 2310.06625 [cs.LG].

[9] Ziming Liu et al. *KAN 2.0: Kolmogorov-Arnold Networks Meet Science*. 2024. arXiv: 2408.10205 [cs.LG]. URL: https://arxiv.org/abs/2408.10205.

[10] Yuqi Nie et al. *A Time Series is Worth 64 Words: Long-term Forecasting with Transformers*. 2022. eprint: arXiv:2211.14730.

[11] Yuqi Nie et al. *A Time Series is Worth 64 Words: Long-term Forecasting with Transformers*. 2023. arXiv: 2211.14730 [cs.LG]. URL: https://arxiv.org/abs/2211.14730.

[12] Ilvico Sonata and Yaya Heryadi. "Transformer and CNN Comparison for Time Series Classification Model." In: *2023 15th International Congress on Advanced Applied Informatics Winter (IIAI-AAI-Winter)*. IEEE. 2023, pp. 160–164.

[13] Ashish Vaswani et al. "Attention is All you Need." In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[14] Haixu Wu et al. "Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting." In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 22419–22430. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/bcc0d400288793e8bdcd7c19a8ac0c2b-Paper.pdf.

[15] Haotian Yan et al. *ConTNet: Why not use convolution and transformer at the same time?* 2021. arXiv: 2104.13497 [cs.CV]. URL: https://arxiv.org/abs/2104.13497.

[16] Manzil Zaheer et al. "Big Bird: Transformers for Longer Sequences." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 17283–17297. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf.

[17] Haoyi Zhou et al. *Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting*. 2020. eprint: arXiv:2012.07436.