To what extent can the incorporation of Temporal Convolution into Transformer improve

efficiency on long time series prediction?

Brian Zu

To what extent can the incorporation of Temporal Convolution into Transformer improve efficiency on long 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 [9]. Since 2017, there have been many works around this model in many tasks such as natural language processing [2], and computer vision [3] 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 [13], patchTST [7], that all aim to improve its 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 [5]. Recent work [6] 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 [9] 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 [12], 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 [4]. 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 [8] 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 [11] for computer vision task. In time series prediction, Informer [13] 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.

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 $X_1, ..., X_n$.
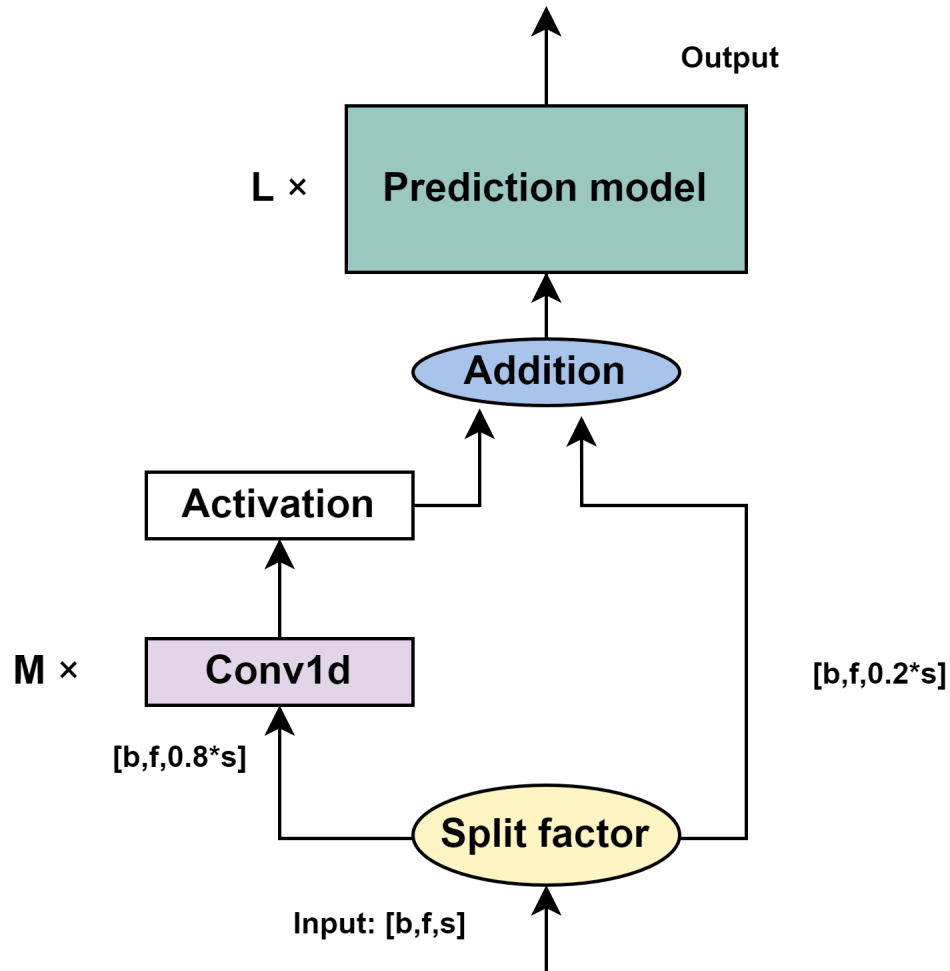
Structure Overview

Figure 1

Sample TCformer 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 **??**, 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.**??**, 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 2

Convolutional sequence summary on left branch
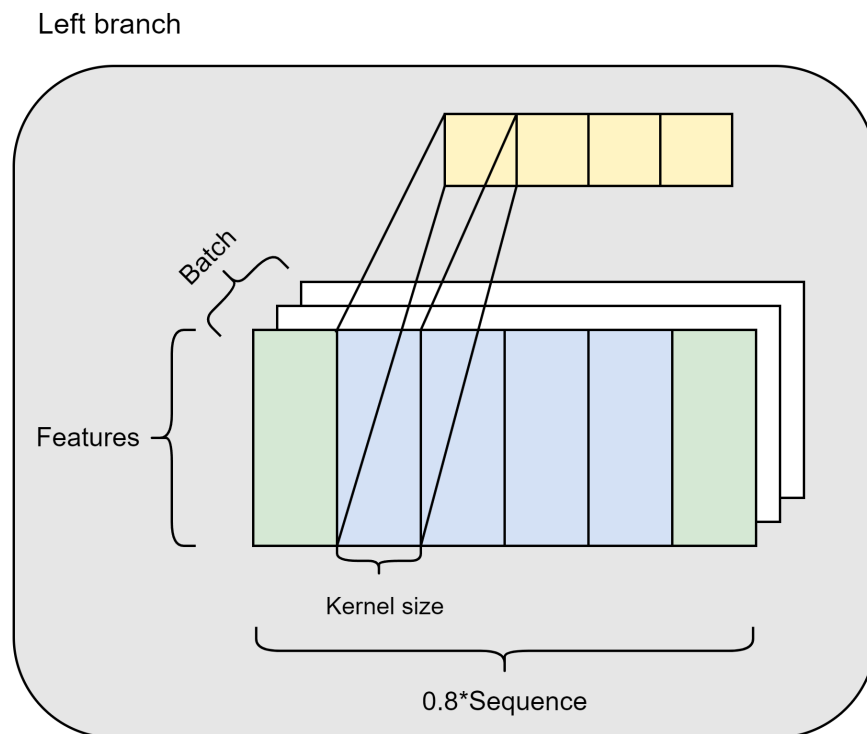
In the left branch, Conv1d layer is outputting summaries for the first 80% of earlier data, leaving the rest 20% raw for the Transformer encoder to process. This will cause the input X to

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.

Parameter 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 had conducted experiments on 3 real world datasets in the experiment, including weather used in Autoformer [10] 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 **??**, 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.
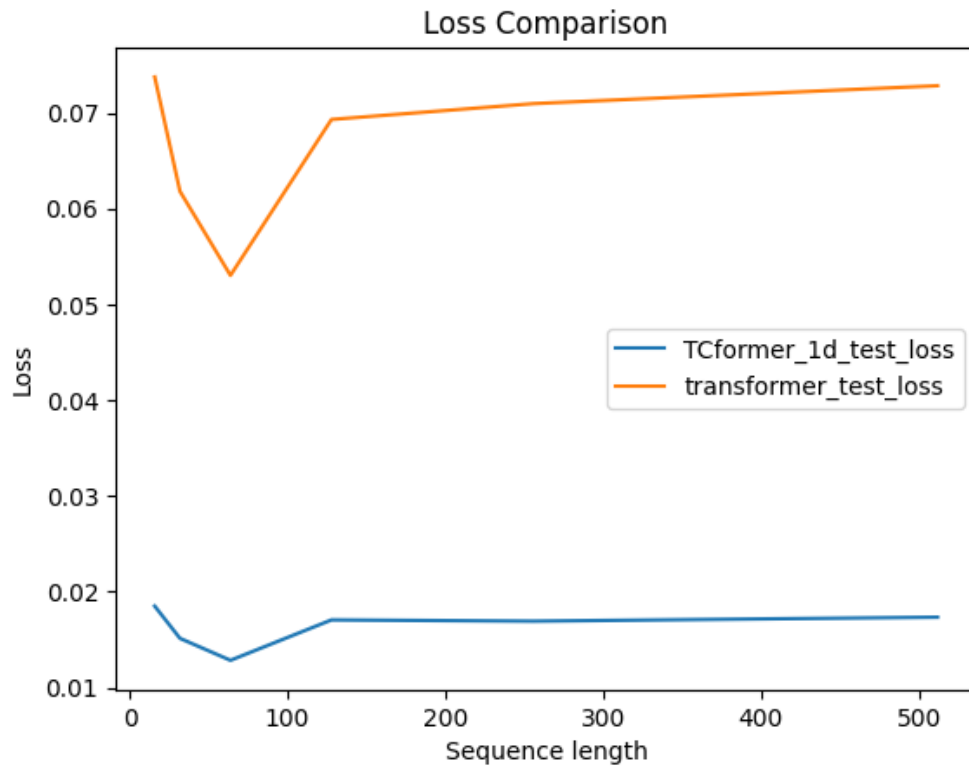
Figure 3

AMD loss curve

On fig.**??**, 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 Trasfomer 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

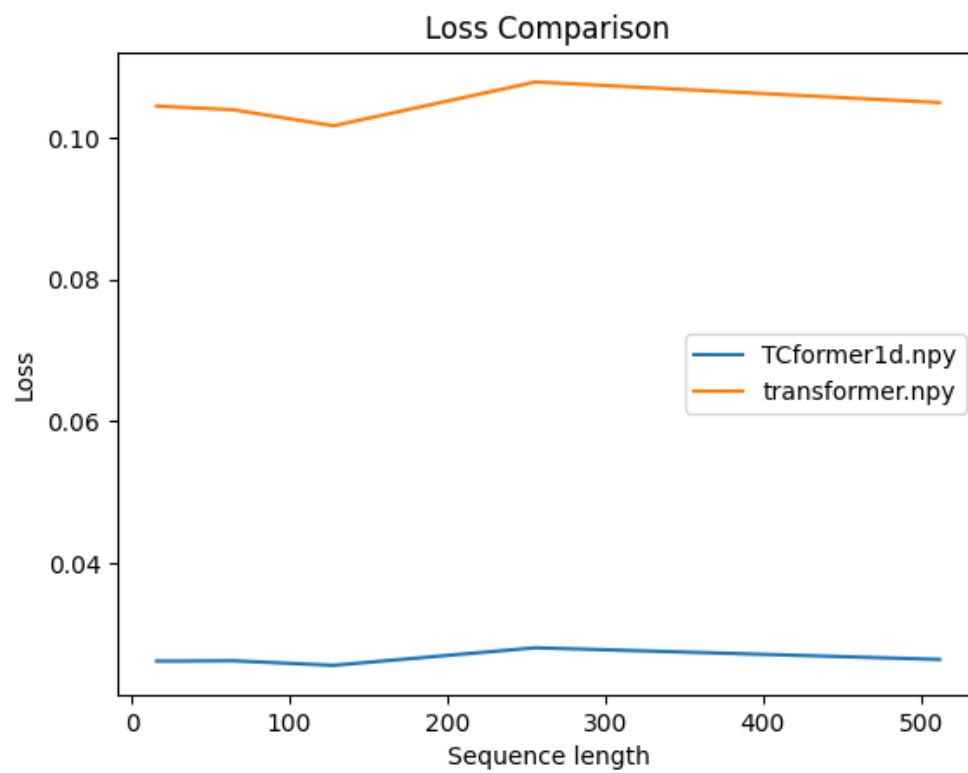| TCformer1d	transformer	seq_len	pct_improve (%)
0.026144	0.104435	16	74.97
0.026144	0.104253	32	74.92
0.026188	0.103928	64	74.80
0.025542	0.101652	128	74.87
0.028019	0.107837	256	74.02
0.026371	0.104923	512	74.87

Like the results in **??**

Figure 4

Weather loss curve

Similar to the results from

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 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. Future work could explore the model's performance on a broader range of datasets and investigate further optimizations to enhance its efficiency and scalability.

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

[4] 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.

[5] 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`.

[6] 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`.

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

[8] 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.

[9] 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`.

[10] 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`.

[11] 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`.

[12] 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`.

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