TransTCN: An Attention-Based TCN Framework for Sequential Modeling

Anonymous authors
Paper under double-blind review

Abstract

Among the sequential modeling issues, the ability to model the long-term dependency remains a significant issue yet to be overcome. Although, recurrent networks extract this information via a recurrent connection, the training step also considers the temporal connection, which reduces the efficiency. However, a temporal connection network (TCN) exploits the benefit of parallelization of convolution and consequently models the sequential information via causal-dilated connection of layers. Moreover, Transformer has exhibited great ability to capture long-term dependency. Thus, in this study, based on the TCN model, the attention blocks in Transformer were introduced to form a model called TransTCN. TransTCN models the sequential information considering attention modules. The model was evaluated across a wide range of the tasks in time series, which are commonly used to the benchmark of TCN and recurrent networks. To the best of our knowledge, TransTCN is the first framework to combine the attention in transformer with TCN to achieve a SOTA performance. The experimental results showed that the perplexity of the word-level prediction on PennTreebank reached only 1.33 while TCN achieved 87.90, which is 66 times of the original TCN. In addition, nearly all loss and perplexity/bpc was improved on other datasets that are commonly used in TCN, except for several datasets wherein our approach maintained performance similar to the original TCN. Furthermore, the training process of TransTCN converges faster than that of TCN.

1 Introduction

The modeling of long-term dependency information in time sequence series tasks has always been a research focus area, because the memory of a long-term relationship can better predict the next step of the time series. Sequential modeling problems have been a extensively studied and have utilized technology, such as language modeling(Peters et al., 2018), neural machine translation(Cho et al., 2014), and polyphonic music generation(Boulanger-Lewandowski et al., 2012). The sequential modeling problems must be overcome to model the long-term dependency. Traditionally, unsupervised and semi-supervised pre-training methods(Dai & Le, 2015)(Radford et al., 2018) have been employed to extract the dependency in the series. However, a neural network is used to extract the long-term dependency in sequential series such as BERT(Devlin et al., 2018), which is a giant network having several parameters, and the long-term dependency information can be memorized owing to the capability of the model. There exist neural models with fewer parameters than BERT such as RNN(Goodfellow et al., 2016) and LSTM(Greff et al., 2016a), but the training process is slow and the ability to capture the long-term dependency is limited as well.

Transformer exhibits a good ability in dealing with problems within sequential series, owing to its attention mechanism with the strong capacity to memory long-term dependency(Vaswani et al., 2017). The BERT model uses the Transformer to build a language model and achieves the SOTA performance of natural language processing(Devlin et al., 2018). However, training this model requires considerable computing resources and a multitude of datasets. Moreover, the training process converges slowly, thereby limiting its application under practical conditions for forwarding tasks. However, the attention block mechanism is a source of inspiration, as it can be used to capture long-term dependency information in other models, and thereby improve the prediction performance of time series models.

For sequential problems, the cyclic autoregressive structure of RNN can be used to model and express the structure well(Goodfellow et al., 2016). However, extraction of long-range dependence information as well training remain challenging (Pascanu et al., 2013). Moreover, there is a problem with the internal design of RNN. The network only reads and parses one word (or character) of the input text at one time, requiring the deep neural network to wait for the previous word to be processed before the next word is processed. This implies that RNN is unable to perform massive parallel processing like CNN, particularly when performing bidirectional text processing. Furthermore, this implies that RNN requires high computational power because all intermediate results must be saved before the entire task is completed. In contrast, applying the convolution to the modeling of sequential problems to build Temporal Convolutional Network (TCN) (Bai et al., 2018) achieves results equivalent to or exceeding that of RNN models. The advantage of CNN is exploited wherein the calculation is independent of the information of the previous time, and thus, they can be parallel.

However, TCN is still affected by the convolution kernel function when dealing with sequential problems, and the receptive field is limited, which makes the capturing of long-range dependent information a challenge. Therefore, as a solution, we considered integrating the attention mechanism in the Transformer into TCN, such that TCN can better capture long-range dependency information and accurately encode and utilize the sequential information by position encoding.

To the best of our knowledge, this study is the first framework to combine the attention in transformer with TCN to achieve a SOTA performance. The experimental results on the typical test dataset of TCN indicated that our method is better than traditional TCN in terms of reduced training speed of loss function and the final convergence value. The TCN achieved a perplexity greater than 87.9 for the word-level prediction on PennTreebank, and the Transformer-based model Transformer-XL reaches perplexity of 54.52(Dai et al., 2019). However, the TransTCN achieved 66 times of perplexity better than that of the original TCN. In addition, nearly all loss and perplexity/bpc was also improved on other common datasets, except for several datasets wherein our approach maintained a performance similar to the original TCN.

2 Related Work

Recurrent networks capture the information in the time dimension but are weak for long-term memory and are notoriously difficult to train (Pascanu et al., 2013). Consequently, LSTM was proposed to extract the long-term dependency for a sequence, and was enhanced in length dependence. However, capturing longer dependencies was a challenge, and it had several parameters(Greff et al., 2016a). In contrast, TCN achieved the same performance as LSTM with fewer parameters and less training cost(Bai et al., 2018). TCN is an empirical evaluation of generic convolutional and recurrent architectures to process the sequence tasks, containing dilations and residual connections of causal convolutions(Bai et al., 2018). Further, MS-TCN is a multi-stage architecture based on TCN for similar tasks such as the temporal action segmentation task(Farha & Gall, 2019). Each stage of MS-TCN considers an initial prediction from the previous stage, and the first stage is a frame-wise feature of the task. Thus, the probability output of the first stage is the input of the second stage, and MS-TCN is suitable for temporal action segmentation tasks. Furthermore, Transformer can maintain long-term dependencies and possesses the ability to tap the impact of different locations(Vaswani et al., 2017).

In addition, Transformer can deal with long-range dependencies, thus, it exhibits better performance for sequence problems. However, it requires a multitude of parameters. Moreover, the ability to capture long-range dependence requires a multitude of datasets for training, and the training time is longer. In addition to the sequential issues, Transformer can also deal with problems in the computer vision field, which was first introduced by (Dosovitskiy et al., 2020). Moreover, the object detection and 3D object detection can be solved using a Transformer (Carion et al., 2020)(Pan et al., 2021).

Further, the Transformer has a wide range of applications, and it is necessary to reduce attention module calculation further. Several studies have focused on improving the training and forwarding speed of the Transformer and reducing the corresponding parameters without compromising much on performance. Performers were employed to decompose the matrices of keys and queries into a lower-dimensional space to reduce the complexity of both space and time (Choromanski et al., 2020). Furthermore, a linear kernel attention mechanism was proposed in (Schlag et al., 2021), which utilizes the memory to store the dot product of keys and queries to reduce number of operation iterations. In addition, DeLighT uses the extension of the attention dimension and reduces the steps to replace the multi-head with a single head(Mehta et al., 2020).

As is widely known, training a deep and large network is difficult. One approach to solving this problem is to modify the structure, while another is to adjust the reasonable width and depth of the existing structure. ResNet is a useful structure for deep networks that enables fast convergence to ideal problem solution (He et al., 2016). Moreover, a high-order ResNet can further extend the performance of vanilla ResNet(Luo et al., 2021a) and can fundamentally solve this problem to find a way to deduce reasonable width, depth, and network structure in theory(Wu et al., 2019). The width of a deep neural network causes the network parameters to appear closer to Gaussian distribution, while the increasing depth can facilitate the network in constructing more complex kernel function mapping(Roberts et al., 2021). Therefore, along with depth the width needs to be carefully selected and tuned as a wider network might be more effective(Xue et al., 2021).

2.1 Dilated Causal Convolution in TCN

TCN is designed based on the principle that there must be no leakage from the future to the past. The primary architecture within the TCN is the dilated causal convolution. In addition, TCN contains a residual connection between the input and output layers.

There exist several operators in dilated causal convolution: padding, causal convolution, and dilated convolution. First, a group of padding is added at the front of the input embedding sequence. Second, a subsequence with the length of the kernel size forms the hidden state via causal convolution. These hidden states serve as a sequence, and thereafter, a dilated convolution samples this sequence to turn into the subsequent hidden sequence.

2.2 Multi Head Attention in Transformer

The Transformer exhibits powerful capabilities in both natural language processing(Devlin et al., 2018) and computer vision(Khan et al., 2021), which improves the performance of related tasks. In contrast to the classic seq2seq module with only one attention(Sutskever et al., 2014), The Transformer contains more than one attention, referred to as multi-head attention.

The inputs of an attention module are K, Q, V, which are the key, query, and value, respectively. Subsequently, three linear operators transform them into KW^K, QW^Q, VW^V, which is a single-head attention. For multi-head attention, there exist a group of the single-head attentions with respective parameters. The attention of the i-th head is denoted as

$$\text{Attention}_i(Q, K, V) = \text{Softmax}\left(\frac{QW_i^Q \cdot (KW_i^K)^{\mathrm{T}}}{\sqrt{d_k}}\right) \cdot VW_i^V,$$

where d_k is the embedding dimension as a scaling factor, making the model training stable. Thus, for multi-head attention with h heads, the outputs is denoted as

$$\text{MultiHeadAttention}(Q, K, V) = \text{Concat}_{i=1}^{h}\left\{\text{Attention}_i(Q, K, V)\right\},$$

Finally, the output attains the same size as inputs through a linear layer.

3 Method

The framework of the proposed TransTCN is shown in Figure 1. TransTCN is a serialization structure with four temporal blocks in total. The temporal inputs of timing t were embedded to a sparse space of c dimensions, and then fed to the series-connected four blocks. First, for each dilated causal convolution in vanilla TCN, a layer with two linear layers was added up to its output to extract the similarity of the output of each layer. Therefore, a ResNet Structure of the output feature for each dilated causal convolution is formed, which is a causal-dilated Resnet composed of two layers of causal dilation convolutions. In addition, the linear layers are required to position encoding information to utilize the location information of the time series. Second, the TransTCN had one new branch called Global Attention Branch, which is the global attention with position encoding of the feature extracted by a convolution layer.

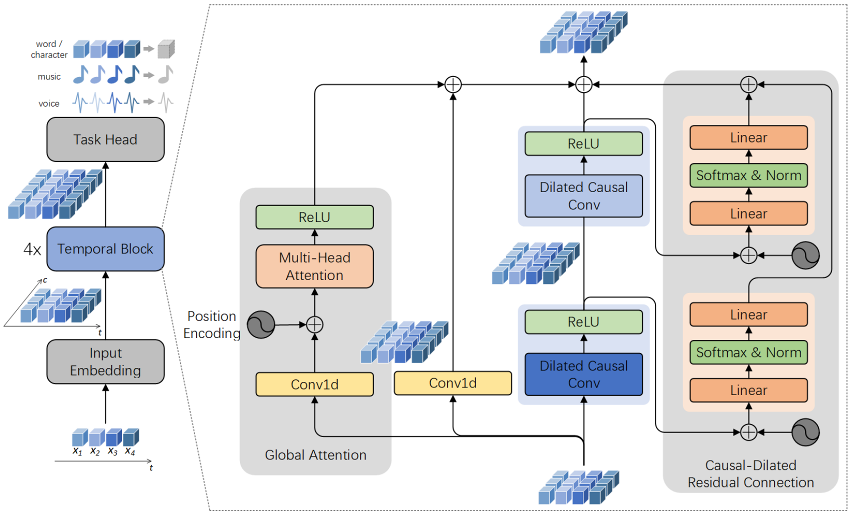


Figure 1: The framework of TransTCN. The input of TransTCN is a series of timing signals from the input embedding layer or features from the prior block. The downstream task can be the next time value of the prediction time series, such as prediction of word level or character level, or the noting prediction of a music.

3.1 Temporal Block

The temporal block is the main structure of TransTCN and is used to extract the characteristics of time series data. Each temporal block maps the timing input sequence of a certain dimension to a higher dimension such that the timing characteristics are reflected in a higher dimension. For example, a time series denoted as $\{x_1, x_2, x_3, \ldots, x_n\}$ is embedded to an embedding space of c dimensions. Therefore, the input of the first temporal block is of size (c, n), indicating that the input has n data points, and each data point is a point in c-dimensional embedding space. However, the input size parameter of the i-th temporal block was set to $c_{\mathrm{i}i}$, and the output size was set to $c_{\mathrm{o}i}$. Consequently, as the signal passed through each temporal block, the signal dimension changed from $c_{\mathrm{i}i}$ to $c_{\mathrm{o}i}$. Further, the output size of the temporal block equaled the output size of the next block. Thus, the dimension of the data passing through the temporal blocks can be denoted as $X^{c \times n} \to X^{c_{\mathrm{o}1} \times n} \to X^{c_{\mathrm{o}2} \times n} \to X^{c_{\mathrm{o}3} \times n} \to X^{c_{\mathrm{o}4} \times n} \triangleq X_{\mathrm{out}}$. Subsequently, the task head accepts X_{out} as its input to predict the next value, as X_{out} extracted by four sequential temporal blocks carries all the extracted features of time series.

3.2 Causal-Dilated Residual Connection Branch

The feature extracted from each causal-dilated convolution was passed through a residual connection with two linear layers to extract the similarity and dependency. However, the causal and dilation operators employed were original TCN layers without any change. The first causal-dilated convolution expands the number of channels for the input feature, denoted as $X^{c_{\mathrm{i}i} \times n}$ for the i-th temporal block. Therefore, the output of the first causal-dilated convolution can be denoted as $X^{c_{\mathrm{cdo1},i} \times n}$, where $c_{\mathrm{cdo}i} > c_{\mathrm{i}i}$. In contrast, the second causal-dilated convolution maintains a constant feature channel, and the output feature is denoted as $X^{c_{\mathrm{cdo2},i} \times n}$. Furthermore, the output of the first causal-dilated convolution $X^{c_{\mathrm{cdo1},i} \times n}$ was fed to a two-linear-layer residual connection. This residual connection consisted of two linear layers with a softmax activation and a normalization layer between them to approximate a function $f^{(1)}_{\text{bi-linear}}$. Moreover, the outputs of the two causal-dilated convolutions was processed by $f^{(j)}_{\text{bi-linear}}, (j = 1, 2)$. However, the first residual connection forms a second-order scheme Luo et al. (2021b), with the output adding to the output of the next block. Therefore, the output of this branch is

$$X_{\text{causal\_dilated}} = f^{(1)}_{\text{bi-linear}}\left(X^{c_{\mathrm{cdo1},i} \times n}\right) + f^{(2)}_{\text{bi-linear}}\left(X^{c_{\mathrm{cdo2},i} \times n}\right) + X^{c_{\mathrm{cdo2},i} \times n},$$

where the size of $X_{\text{causal\_dilated}}$ is $(c_{\text{cdo2,i}}, n)$. Moreover, the residual connection also considers the position encoding because the positional information in vanilla TCN is not fully utilized by causal-dilated convolution.

For causal-dilated convolution, multiple signals in adjacent times of one kernel are fused into a signal of the next layer. Following the dilation operation, the adjacent signals are fused to the next layer. This process does not consider the timing characteristics of the input sequence. Therefore, position encoding can be used to encode position information, further encode the sequence of timing signals, and subsequently extract sequence features from the perspective of signal sequence prediction. For a causal-dilated convolution with kernel size of k, the subsequence of input $\{x_{s+1}, x_{s+2}, \ldots, x_{s+k}\}$ are treated equally as the input of the next layer, and the order is not distinguished. Thus, subsequence of $\{x_{s+1}, x_{s+2}, \ldots, x_{s+k}\}$ and $\{x_{s+k}, x_{s+1}, \ldots, x_{s+2}\}$ exhibit no difference. Thus, when introducing position encoding into the input feature, each signal at different positions is treated differently, such that the value of the subsequent time can be effectively predicted according to the time series.

3.3 Global Attention Branch

The causal-dilated convolution in TCN extracts week dependency of time series because the causal-dilated convolution only focuses on the local sequence in one kernel. Finally, the partially extracted information is discarded by the TCN. As TCN can only analyze the preamble sequence, it only focuses on semantics in local. However, although the global attention branch considers the global semantic similarity information, the location information is added by it to emphasize the meaning of input entities in different locations.

The global attention branch first uses a one-dimensional convolution layer to extract features from time series, and the output is denoted as X_{conv1d} with size of $(c_{\text{cdo2,i}}, n)$. Subsequently, a position encoding for each feature is added to the output of convolution, which encodes the global position of the time series to utilize it, as the position in a time series is the key information required for predicting the next step of time series. Moreover, the same value in different positions contributes to the future prediction in varying manners.

In the next step, the feature with its position encoding is sent to a multi-head attention layer to extract the global dependency of the time series. In addition, the subsequences at key positions require greater attention, as they impact the prediction of the future values. The output of the global attention is denoted as $X_{\text{global\_attention}}$ with size of $(c_{\text{cdo2,i}}, n)$. Further, a residual connection was used to improve the training convergence and training speed of global attention, and the output is denoted as $X_{\text{global\_residual}} = X_{\text{global\_attention}} + X_{\text{conv1d}}$ with size of $(c_{\text{cdo2,i}}, n)$.

Moreover, employing the one-dimensional convolution after the global attention block was not possible according to the experiment in section A.0.2. If the convolution were to be used after the global block, the positional embedding information would also be weighted by the convolution function. Consequently, the positional information would be disrupted, deviating from the operation of global information using location information to weigh the importance of values at different locations. In addition, the phenomenon can be observed in the subsequent ablation experiment, which was used to compare the prediction effect after the exchange of the two modules.

3.4 Concatenate of All Branches

Until now, both the causal-dilated residual connection and global attention branches have been introduced to vanilla TCN, and a TransTCN was obtained. The output of these two branches are of the same size $(c_{\text{cdo2,i}}, n)$. Thus, the output of the temporal block can be considered as their sum, denoted as

$$X_{\text{temporal\_block}} = X_{\text{causal\_dilated}} + X_{\text{global\_residual}},$$

where the size of $X_{\text{temporal\_block}}$ is $(c_{\text{cdo2,i}}, n)$. This output contains both the time series value information and the positional information, and also the importance of different positions in time series.

Table 1: Tasks and datasets

Sequence Modeling Task	Dataset
Adding Problem	Conditional random generation (Hochreiter & Schmidhuber, 1997) (Zhang et al., 2016)
Word-level language modeling	PennTreebank(Marcus et al., 1993), Wikitext-103(Merity et al., 2016) , LAMBADA(Paperno et al., 2016)
Character-level language modeling	PennTreebank, text8(Mikolov et al., 2012)
Polyphonic music prediction	JSB Chorales(Allan & Williams, 2005), Nottingham(Greff et al., 2016b), Piano(Bernd, 1998), Muse(Stanford, 1984)
Digit classification	MNIST(LeCun et al., 1998)
Copy memory	Conditional random generation (Jing et al., 2017)

4 Experiments

The experiments were all the classical tasks and the datasets used in vanilla TCN were employed to deal with timing problems. TransTCN and vanilla TCN were compared in terms of prediction accuracy and the convergence state of training. The intention was to conduct a baseline for TransTCN and further discuss the improvement of TCN with two branches under different structures.

4.1 Tasks and Datasets

The tasks and datasets are shown in Table 1. According to the vanilla TCN performance comparison baseline, a total of six tasks were performed to evaluate the performance of TransTCN.

4.2 Setups

The experiments included one comparison to the state-of-the-art TCN and three ablation experiments in the same setting for each dataset. The modules used in the ablation experiments are shown in Figure 2. All the experiments were trained and evaluated on 32 cards of NVIDIA A100-SXM4-40GB GPUs(Nvidia, 2020).

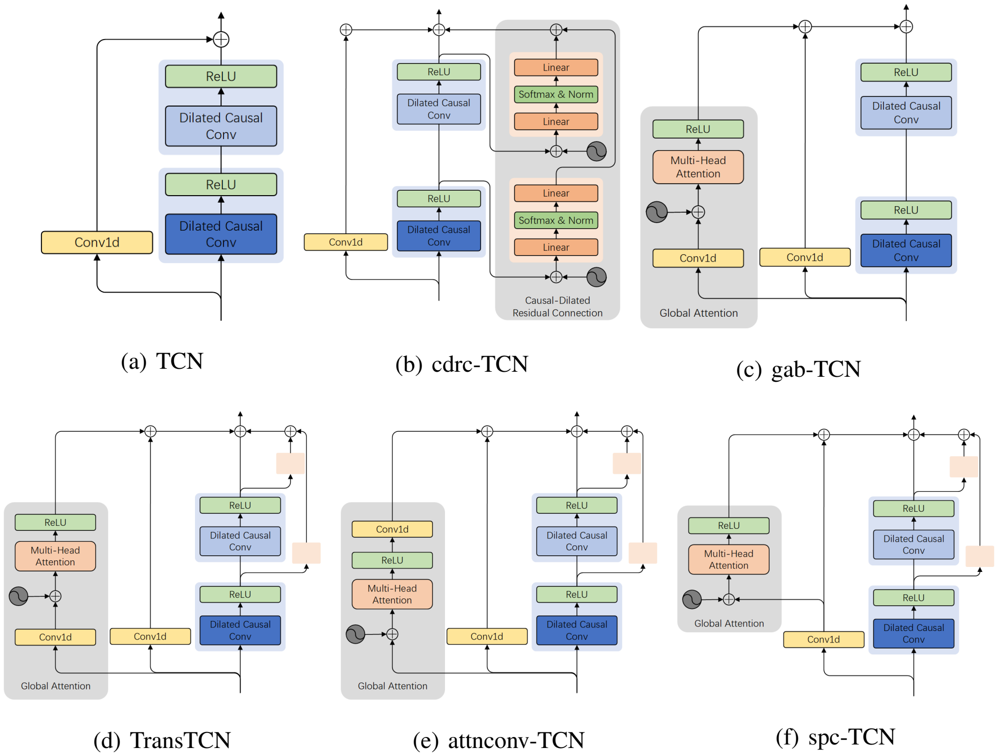
In contrast to the vanilla TCN, there were two additional branches in the temporal block of TransTCN: a causal-dilated residual connection branch and a global attention branch. Thus, the experiments focused on the addition of these two branches.

The first experiment involved comparing the results with the original TCN model comprising only one of the two branches, that is, the TCN with the causal-dilated residual connection branches(cdrc-TCN) as in Figure 2(b), and the TCN with global attention branches(gab-TCN) as in Figure 2(c). These two experiments illustrated the importance of the two branches and verified the analysis and design ideas in the previous sections.

In the global attention branch, the order of the convolution and attention layers affects the performance of the model. Thus, the second experiment was to compare the results of the order, that is, a convolution layer before the attention layer(TransTCN) as in Figure 2(d), and a convolution layer after the attention layer(attnconv-TCN) as in Figure 2(e).

In addition, for the case of a convolution layer before the attention layer, the convolution parameters could be shared with those of convolution residual connection in original TCN, leading to the third comparative experiment; that is, the shared parameters convolution(spc-TCN) as in Figure 2(f).

For the various tasks, the requirement of parameters in TransTCN was different, and the comparison with TCN is shown in Table 2. The number of parameters varied from tasks for the same module because the number of input channels were different in various tasks. Therefore, the temporal



(a) TCN

(b) cdrc-TCN

(c) gab-TCN

(d) TransTCN

(e) attnconv-TCN

(f) spc-TCN

Figure 2: The models and setups of the three ablation experiments. (a) The vanilla TCN without the two branches proposed in this paper. (b) The right branch in the dark area is the causal-dilated residual connection branch. (c) The left branch in the dark area is the global attention branch. (d) Both the causal-dilated residual connection and the global attention branch are present in the model. (e) The convolution layer is behind the attention module in the global attention branch. (f) The convolution layers in TransTCN are combined as one layer to share the parameter.

Table 2: The number of parameters for different tasks and modules.

Sequence Modeling Task	TCN	x-TCN				TransTCN
		cdrc	gab	attnconv	spc	
Adding Problem ($T = 600$)	70K	172K	147K	143K	208K	208K
Copy Memory ($T = 1000$)	16K	35K	19K	18K	39K	39K
Word-level PennTreebank	13M	18M	25M	25M	25M	25M
Character-level PennTreebank	3M	3M	5M	5M	5M	5M
Character-level text8	5M	3M	5M	5M	5M	5M
Permuted MNIST	70K	128K	103K	100K	154K	154K
JSB Chorales	300K	1217K	1507K	1448K	1660K	1660K
Nottingham	1M	1.2M	1.5M	1.4M	1.6M	1.6M
Muse	1M	1.2M	1.5M	1.4M	1.6M	1.6M
Piano	1M	1.2M	1.5M	1.4M	1.6M	1.6M

Table 3: The testing performance of TransTCN compared with TCN.

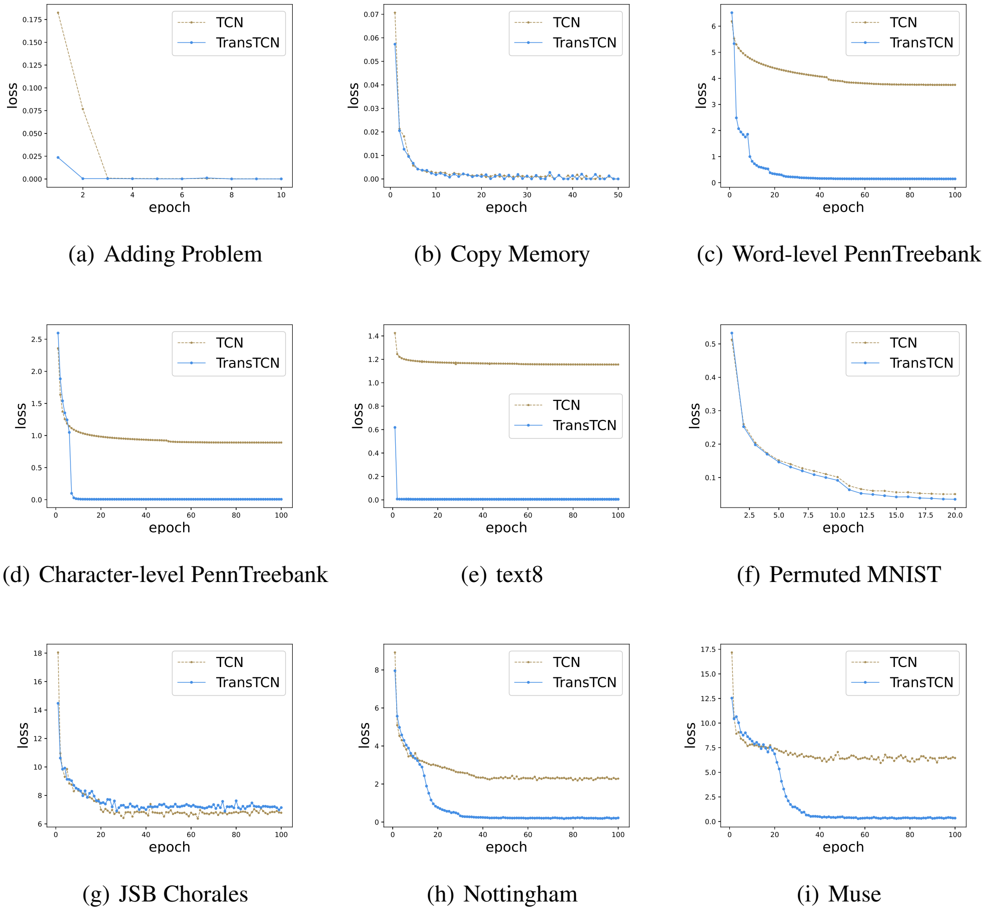
Sequence Modeling Task	Evaluation	TCN	TransTCN	Improvement($\times n$)
Adding Problem	loss	$5.8\mathrm{e}-5$	$\mathbf{1.2e-5}$	4.83
Copy Memory	loss	$3.5\mathrm{e}-5$	$\mathbf{1.1e-6}$	31.82
Word-level PennTreebank	loss	4.48	**0.29**	15.45
	perplexity	87.90	**1.33**	66.09
Character-level PennTreebank	loss	0.918	**0.017**	54.00
	bpc	1.324	**0.024**	55.17
Character-level text8	loss	1.099	**0.044**	24.98
	bpc	1.585	**0.063**	25.16
Permuted MNIST	loss	0.0851	**0.0816**	1.04
	accuracy(%)	97.62	**97.69**	1.001
JSB Chorales	loss	**8.10**	8.42	0.96
Nottingham	loss	3.07	**0.32**	9.59
Muse	loss	6.99	**0.46**	15.20
Piano	loss	**7.61**	7.95	0.96

blocks require varying number of parameters. For most of the tasks, the number of parameters in TransTCN was approximately twice as many as that in TCN, except for the adding problem and the JSB Chorales tasks, where the number of parameters in TransTCN was still within the acceptable range.

4.3 Comparison with State of the Art

For the original TCN, the number of parameters required in the model was less than RNN(Pascanu et al., 2013) and LSTM(Greff et al., 2016a), which is the main advantage of TCN. Fewer parameters for TCN facilitate easy training and the quick convergence of the training steps. However, for TransTCN, although the number of parameters required is greater, the training convergence process is faster than TCN, and simultaneously the final performance is better.

As presented in the Table 3, for most tasks except for Permuted MNIST, JSB Chorales and Piano, the TransTCN exhibited better training performance than that of the original TCN. Further, the performance of TransTCN and TCN is almost the same for the Permuted MNIST, JSB Chorales, and Piano tasks. Therefore, it is evident that the proposed method can achieve the same performance as the original TCN, and can even achieve much better performance on most of the tasks. Moreover, the training step of TransTCN converges faster than that of TCN as shown in Figure 3. The loss value in the training process was found to decrease rapidly after several starting epochs. For the final convergence value in training steps of TransTCN on the following datasets, Word/Character-level PennTreebank as in Figure 3(c)/(d), text8 as in Figure 3(e), Nottingham as in Figure 3(h), and Muse as in Figure 3(i), the training loss was observed to converge to a significantly smaller value than that of TCN. Furthermore, these experiments exhibit a significant performance improvement on the respective datasets, as shown in the last column in Table 3.



(a) Adding Problem

(b) Copy Memory

(c) Word-level PennTreebank

(d) Character-level PennTreebank

(e) text8

(f) Permuted MNIST

(g) JSB Chorales

(h) Nottingham

(i) Muse

Figure 3: The convergence process of training TransTCN and TCN on different datasets.

5 Conclusion

In this study, a TransTCN model was proposed based on the TCN model combined with the attention blocks in Transformer to capture long-term dependency information in time series. TransTCN, exploits the advantage of TCN and the attention in Transformer, and achieved improved or similar performance and training convergence speed to that of original TCN. Moreover, the TransTCN achieved SOTA perplexity on most of the commonly used datasets without introducing excessive model parameters. Further, the training process converged faster than TCN.

However, there are certain challenges yet to be overcome. The positional encoding is now simple as the sequence modeling issues may be complex in the time dimension. In addition, the causal-dilated branch used positional encoding twice, which repeatedly emphasized the importance of location information to prevent the model from weakening or even losing sequential information in the forward transmission process. For future work, the positional encoding modules can be better designed according to the location characteristics of time series problem. In addition, the performance on larger datasets can be implemented to check the performance of TransTCN.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61934003 and Grant No. 62090054, in part by the Major Scientific and Technological Program of Jilin Province under Grants No. 20200501007GX, and in part by the Program for JLU Science and Technology Innovative Research Team (JLUSTIRT, 2021TD-39). We would like to thank Editage (www.editage.cn) for English language editing.

References

Moray Allan and Christopher KI Williams. Harmonising chorales by probabilistic inference. *Advances in neural information processing systems*, 17:25–32, 2005.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Krueger Bernd. Classical piano midi page. [EB/OL], 1998. `http://www.piano-midi.de`.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pp. 213–229. Springer, 2020.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. *Advances in neural information processing systems*, 28:3079–3087, 2015.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Yazan Abu Farha and Jurgen Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3575–3584, 2019.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10): 2222–2232, 2016a.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10): 2222–2232, 2016b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Li Jing, Yichen Shen, Tena Dubcek, John Peurifoy, Scott Skirlo, Yann LeCun, Max Tegmark, and Marin Soljačić. Tunable efficient unitary neural networks (eunn) and their application to rnns. In *International Conference on Machine Learning*, pp. 1733–1741. PMLR, 2017.

Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *arXiv preprint arXiv:2101.01169*, 2021.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Zhengbo Luo, Zitang Sun, Weilian Zhou, and Sei-ichiro Kamata. Rethinking resnets: Improved stacking strategies with high order schemes. *arXiv preprint arXiv:2103.15244*, 2021a.

Zhengbo Luo, Zitang Sun, Weilian Zhou, and Sei-ichiro Kamata. Rethinking resnets: Improved stacking strategies with high order schemes. *arXiv preprint arXiv:2103.15244*, 2021b.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

Sachin Mehta, Marjan Ghazvininejad, Srinivasan Iyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. Delight: Deep and light-weight transformer. *arXiv preprint arXiv:2008.00623*, 2020.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. Subword language modeling with neural networks. *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)*, 8:67, 2012.

Nvidia. Nvidia a100 tensor core gpu datasheet. [EB/OL], September 2020. `https://images.nvidia.com/content/Solutions/data-center/a100/pdf/nvidia-a100-datasheet-us-partner-1758950-r4-zhCN.pdf`.

Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7463–7472, 2021.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318. PMLR, 2013.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

Daniel A Roberts, Sho Yaida, and Boris Hanin. The principles of deep learning theory. *arXiv preprint arXiv:2106.10165*, 2021.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight memory systems. *arXiv preprint arXiv:2102.11174*, 2021.

Table 4: The performance of TransTCN compared with cdrc-TCN and gab-TCN.

Sequence Modeling Task	Evaluation	cdrc-TCN	gab-TCN	TransTCN
Adding Problem	loss	2.8e − 5	4.4e − 2	**1.2e − 5**
Copy Memory	loss	4.9e − 5	1.9e − 5	**1.1e − 6**
Word-level PennTreebank	loss	**0.29**	6.46	**0.29**
	perplexity	1.34	639.18	**1.33**
Character-level PennTreebank	loss	0.018	0.023	**0.017**
	bpc	0.026	0.033	**0.024**
Character-level text8	loss	**0.040**	2.859	0.044
	bpc	**0.058**	4.125	0.063
Permuted MNIST	loss	0.0882	0.0842	**0.0816**
	accuracy(%)	97.37	97.50	**97.69**
JSB Chorales	loss	8.44	8.59	**8.42**
Nottingham	loss	3.64	3.11	**0.32**
Muse	loss	–	8.04	**0.46**
Piano	loss	8.39	8.18	**7.95**

Stanford. Center for computer assisted research in the humanities. [EB/OL], 1984. http://musedata.stanford.edu.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Zifeng Wu, Chunhua Shen, and Anton Van Den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90:119–133, 2019.

Fuzhao Xue, Ziji Shi, Yuxuan Lou, Yong Liu, and Yang You. Go wider instead of deeper. *arXiv preprint arXiv:2107.11817*, 2021.

Saizheng Zhang, Yuhuai Wu, Tong Che, Zhouhan Lin, Roland Memisevic, Russ R Salakhutdinov, and Yoshua Bengio. Architectural complexity measures of recurrent neural networks. *Advances in neural information processing systems*, 29:1822–1830, 2016.

A Appendix: Ablation Experiments

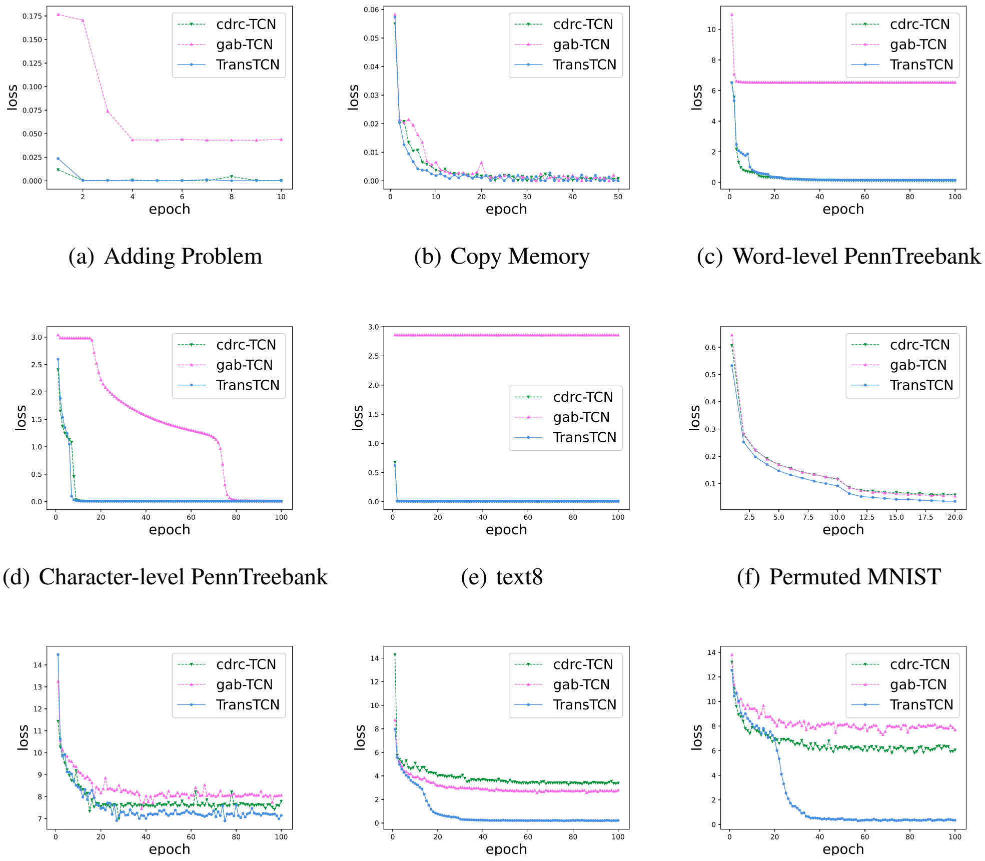
After removing the two branches, namely, the global attention and the causal-dilated residual connection branches, the ablation experiments examined the performance of the model and compared it with the original model to illustrate the importance of branches from an experimental perspective. For the global attention branch, the position of the convolution layer was also switched with the attention layer to analyze the performance of the model further.

A.1 The Validity of Two Branches

On removing the global attention branch of TransTCN, the cdrc-TCN was obtained, and similarly, the gab-TCN was obtained on removing the other branch. The results are shown in Table 4. The TransTCN performs best among the three modules except for the character-level prediction task on dataset text8, although the performance gap is minimal. Thus, the performance improvement achieved with the two branches working alone is poor as the global attention and causal-dilated residual connection branches extract various features from the sequence.

The convergence states of the cdrc-TCN, gab-TCN and TransTCN for different datasets are shown in Figure 4. According to the experimental results of the datasets Word-level PennTreebank as in

Figure 4(c) and text8 as in Figure 4(e), the gab-TCN converges to a much larger value than that of cdrc-TCN or TransTCN. In addition, the training step converges slowly on Character-level PennTreebank as in Figure 4(d). This indicates that a TransTCN with only global attention without local causal attention cannot effectively extract long-term dependency. The causal-dilated convolution branch of TransTCN combined the information of every sequence with the positional encoding, and the long-term dependency was also stored in the weights of the two linear layers of each residual connection block. However, cdrc-TCN exhibited a significantly different behavior from TransTCN on the datasets of JSB Chorales as in Figure 4(g), Nottingham as in Figure 4(h), and Muse as in Figure 4(i). Thus, for the prediction on the datasets in the field of music, the global attention branch in TransTCN extracted the global information such as the style of music, and the prediction that incorporates this information would be more accurate.



(a) Adding Problem

(b) Copy Memory

(c) Word-level PennTreebank

(d) Character-level PennTreebank

(e) text8

(f) Permuted MNIST

(g) JSB Chorales

(h) Nottingham

(i) Muse

Figure 4: The convergence process of training TransTCN and TransTCN without two respective branches on different datasets.

A.2 Place Global Attention Branch Ahead of Convolution

The other issue is to discuss the location of the convolution and the attention layers. We need to determine whether the performance would improve or decrease, when the order of these two layers is switched. The results of switching the convolution layer order are shown in Table 5. It was found that when the convolution layer was placed behind the attention, the corresponding positional encoding information was also processed through one-dimensional convolution to combine the feature in the size of the kernel window so as to fuse the position encoding information, This destroyed the position information introduced into the global attention. In contrast, placing the convolution layer before

Table 5: The performance of TransTCN compared with attnconv-TCN.

Sequence Modeling Task	Evaluation	attnconv-TCN	TransTCN
Adding Problem	loss	–	**1.2e − 5**
Copy Memory	loss	1.6e − 6	**1.1e − 6**
Word-level PennTreebank	loss	0.30	**0.29**
	perplexity	1.34	**1.33**
Character-level PennTreebank	loss	0.907	**0.017**
	bpc	1.308	**0.024**
Character-level text8	loss	**0.042**	0.044
	bpc	**0.016**	0.063
Permuted MNIST	loss	0.0866	**0.0816**
	accuracy(%)	97.58	**97.69**
JSB Chorales	loss	8.51	**8.42**
Nottingham	loss	2.96	**0.32**
Muse	loss	7.76	**0.46**
Piano	loss	7.86	**7.95**

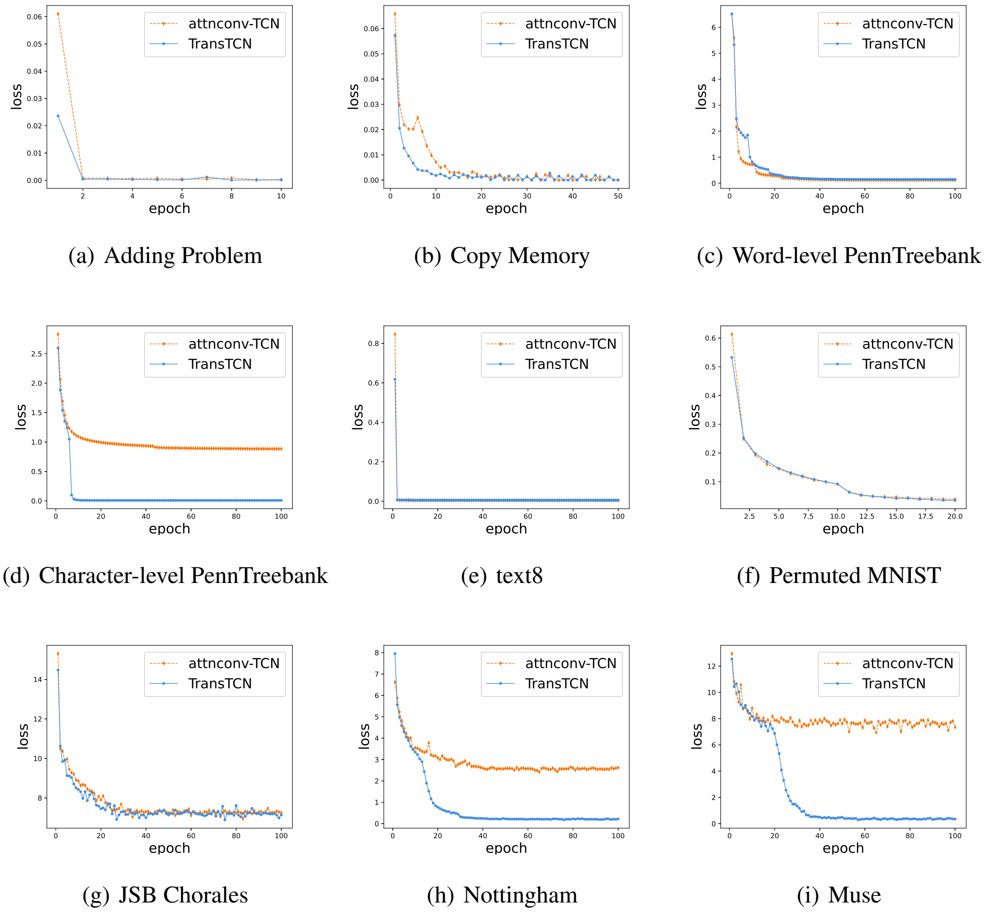
the attention ensured that the positional information is encoded after convolution, and consequently, the positional information to the attention was be damaged.

The comparison for the decline rate of testing step between attnconv-TCN and TransTCN is shown in Figure 5. The decline rate of the loss function is similar, but the final convergence value of attncon-TCN is larger than that of TransTCN, particularly for the datasets of Character-level PennTreebank as in Figure 5(d), Nottingham as in Figure5(h), and Muse as in Figure 5(i). Further, although the parameters and the functions of the blocks in attncon-TCN and TransTCN are the same, the order of the blocks is not. The convolutional layer disrupted the sequential information, thus, the performance reduced and the convergence effect of loss function became worse.

A.3 Shared Parameters in Convolution Layers

Finally, we determined that the reason for the performance loss in attnconv-TCN was the positional information embedding misleading the learning steps or the increase of network parameters in convolution layers. Thus, combining the convolution layers of the global attention branch and the original is one approach to reduce the parameters in the TransTCN. The results are shown in Table 6. However, parameter reduction by merging convolution layers does not improve the performance of the model because the two branches contain their respective convolution parameters and have different functions. The convolution layer in the global attention branch encodes the input feature of the temporal block to extract global sequential information, while the convolution layer in original TCN resized the input to implement a residual link to the output.

The convergence states of training spc-TCN is shown in Figure 6. The number of parameters in spc-TCN is slightly lesser than that of TransTCN but there is a significant decrease in the performance. This indicates that the parameters of the two convolution layers cannot be shared. The convolution layer in the global attention branch extracted the sequential information in global, while the one in raw TCN resized the input channel to that of the output channel. Moreover, there may be two groups of parameters in the two branches with respective functions.



(a) Adding Problem

(b) Copy Memory

(c) Word-level PennTreebank

(d) Character-level PennTreebank

(e) text8

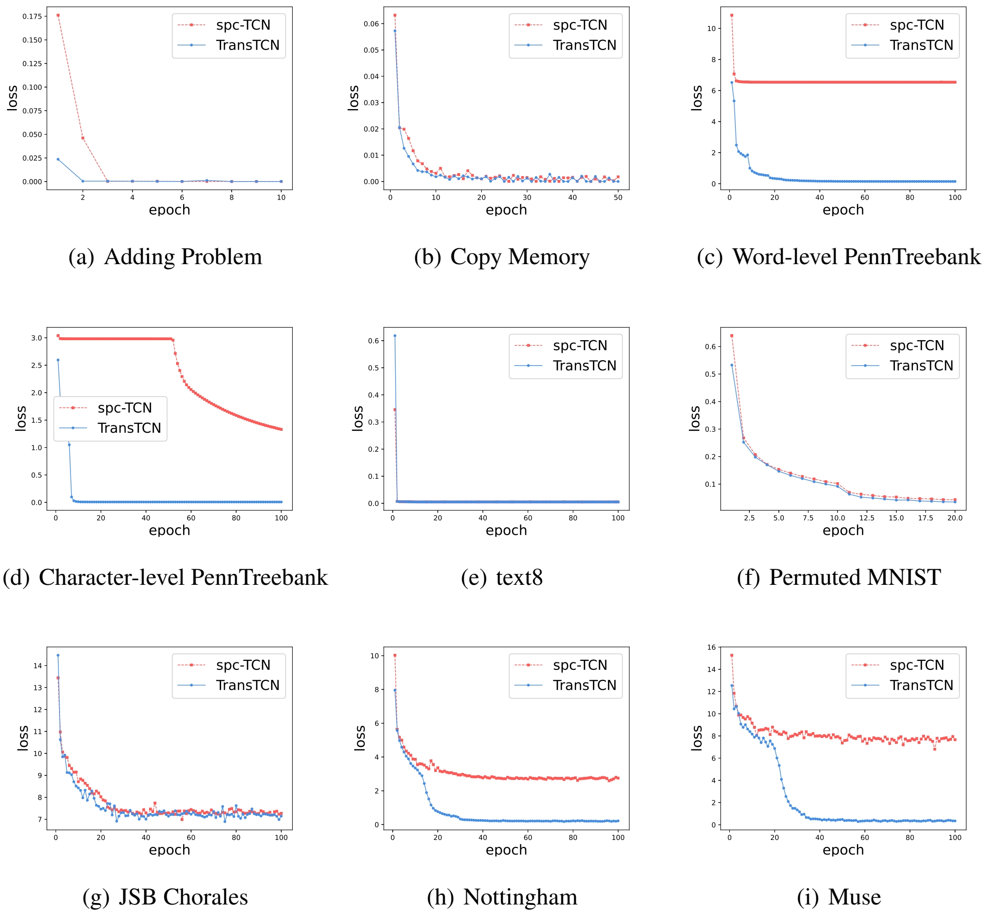
(f) Permuted MNIST

(g) JSB Chorales

(h) Nottingham

(i) Muse

Figure 5: The convergence process of training TransTCN and TransTCN with a global attention branch of convolution after attention.



(a) Adding Problem

(b) Copy Memory

(c) Word-level PennTreebank

(d) Character-level PennTreebank

(e) text8

(f) Permuted MNIST

(g) JSB Chorales

(h) Nottingham

(i) Muse

Figure 6: The convergence process of training TransTCN and TransTCN with a global attention branch of one convolution layer.

Table 6: Performance of TransTCN compared with spc-TCN.

Sequence Modeling Task	Evaluation	spc-TCN	TransTCN
Adding Problem	loss	3.0e − 5	**1.2e − 5**
Copy Memory	loss	2.1e − 4	**1.1e − 6**
Word-level PennTreebank	loss	6.46	**0.29**
	perplexity	639.18	**1.33**
Character-level PennTreebank	loss	1.230	**0.017**
	bpc	1.774	**0.024**
Character-level text8	loss	**0.030**	0.044
	bpc	**0.044**	0.063
Permuted MNIST	loss	**0.0808**	0.0816
	accuracy(%)	97.66	**97.69**
JSB Chorales	loss	8.59	**8.42**
Nottingham	loss	3.10	**0.32**
Muse	loss	7.94	**0.46**
Piano	loss	8.29	**7.95**