Information Sciences 627 (2023) 97–114

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Rethinking attention mechanism in time series classification

Bowen Zhao [a,b,c], Huanlai Xing [a,b,c,*], Xinhan Wang [a,b,c], Fuhong Song [a,b,c], Zhiwen Xiao [a,b,c]

[a] School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, China
[b] Tangshan Institute of Southwest Jiaotong University, Chengdu, China
[c] Engineering Research Center of Sustainable Urban Intelligent Transportation, Ministry of Education, Chengdu, China

ARTICLE INFO

Keywords:
Deformable convolution
Knowledge distillation
Time series classification
Transformer

ABSTRACT

The potential of attention mechanisms in time series classification (TSC) is limited owing to general drawbacks, like weak local perception and quadratic complexity. To promote the performance of attention mechanisms, we present a flexible multi-head linear attention (FMLA) architecture, which enhances locality awareness through layer-wise interactions with deformable convolutional blocks and online knowledge distillation. We develop a simple but effective mask mechanism that helps reduce noise influence in time series and reduces the redundancy of FMLA by probabilistically selecting and masking the positions of each given series. We use incremental ablation studies on 85 UCR2018 datasets to evaluate the performance of the main techniques developed. Experimental results demonstrate that FMLA outperformed 11 state-of-the-art TSC algorithms, obtaining a mean accuracy of 89.37%. FMLA achieved the best performance on 29 short-medium and 7 long time-series datasets regarding accuracy. FMLA has a complexity of $O(N)$, where N is the sample length. As N increases from 100 to 1000, the floating-point operations per second grow linearly from 0.13G to 1.34G.

1. Introduction

Nowadays, a large amount of time-series data is stored and analyzed every second in areas like healthcare monitoring, intelligent manufacturing, and many other Internet of Things applications, which require effective time-series data mining. In particular, time-series classification (TSC) has drawn increasingly more attention in the past few years and effectively extracting temporal representations for the series is one of the general purposes.

In contrast to images and sentences, a time series consists of continuous and non-stationary data with shapelets [29] that may not be truncated arbitrarily. The spans of shapelets may also change even in the same application area. For instance, one can easily recognize whether an object is a dog or a cat only through partial features, like eyes or claws, wherever they are. Nevertheless, a segment of the Electrocardiograph (ECG) series is normally meaningless. Thus, many data augmentation techniques, like jigsaw puzzles and geometric transformation, are normally powerful in pretraining models in computer vision but powerless in time-series analysis.

For sequence modeling, long short-term memory (LSTM) networks have advantages in temporal feature discovery since they process sequences in chronological order. More recently processed data have more influence than those processed earlier, causing data

* Corresponding author at: School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, China.
E-mail address: hxx@home.swjtu.edu.cn (H. Xing).

https://doi.org/10.1016/j.ins.2023.01.093
Received 13 July 2022; Received in revised form 10 January 2023; Accepted 15 January 2023
Available online 21 January 2023

points to differ in their relative importance. LSTM networks were recognized as promising for TSC. However, they cannot process time-series data in parallel and easily forget the meaningful information previously extracted. There have been some sequence modeling studies based on convolutional neural networks (CNNs), like temporal convolutional networks (TCNs) [1] and multi-scale attention convolutional neural network (MACNN) [3]. CNN-like structures can achieve better parallelism and their inductive bias, that is, locality awareness, makes them powerful in handling time series. However, these structures are usually limited by their inductive bias in situations that require global awareness. Unlike structures based on and CNNs above, Transformers allow data to be processed in parallel and have excellent global receptive fields. These advantages are helpful in sequence modeling especially for analyzing time-series data, like residual attention net (RAN) [18], graph correlated attention recurrent neural network (GCAR) [13] and robust temporal feature network (RTFN) [45]. The shapelets hidden in a time series are normally of various forms and spans. Therefore, feature extractors require global and local perception. According to [11,23], the Transformer was initially designed for global pattern extraction whereas local pattern extraction is also essential for time-series analysis. Although there have been some special implementations of local attention mechanisms, like [10], the selection of attention scope is unavoidable, and it may destroy the shapelets in a sequence if not set properly. Meanwhile, it has been recognized that pure local attention is not superior to those well-designed CNNs in terms of local feature extraction [14]. In this regard, some research attention has been paid to wisely hybridizing Transformers and CNNs, e.g., loosely-coupled Transformer-CNN structures [9,30] and CNN-like Transformers [27]. Recently, enhancing local awareness of Transformers using CNN-like algorithms has become one of the most attractive research directions.

Beside the deficiency of local representation, Transformers also suffer from a complexity of $O(N^2)$ [37] mainly due to the adopted multi-head attention structure. The dot-products between queries and keys in the attention mechanism make the quadratic complexity unavoidable. This unacceptable time and memory consumption make it hard to apply Transformers to long-sequence mining. Therefore, several variants, like Softmax-free Transformer (SOFT) [28], Linformer [41], Swin Transformer [27], and Poolingformer [47], have been proposed to linearly approximate the attention maps in the past few years and their advantages and disadvantages are all summarized in [37]. There are three main directions for the complexity improvement of the Transformer: the window attention in Swin Transformer [27], the sampling technique in SOFT [28] and the low-rank assumption in Linformer [41]. However, the window-based method in Swin Transformer limits its potential to extract features with various lengths since there is no universal window size. The sampling technique in SOFT may lose meaningful information with the reduction of complexity, and the sampling process is also not stable enough. The low-rank assumption and rough approximation in Linformer break the consistency between keys and values, which results in useless attention information. The feature redundancy between multiple heads may also influence the results of classification and hinder the flexibility of these attention-based classification models. In TSC, we need to strike a balance between high-complexity computation and effective feature extraction to find a linear attention mechanism with global and local awareness.

In TSC, the notorious noise interference is encountered in many fields, such as finance, weather, and audio. It is easy for human beings to ignore noise because they do not have to analyze a given time series one digit after another as machines do. The fluctuations caused by noise may lead to deteriorated performance for local pattern recognition. Addressing noise interference is still a challenge.

We propose a new variant of the Transformer for TSC, namely flexible multi-head linear attention (FMLA). Compared with the existing Transformer-based models, our architecture has excellent local feature extraction, low complexity, and promising noise-resistance ability. Each FMLA block integrates deformable convolutional networks (DCN) [5] into a collaborative linear attention (CLA) mechanism, which ensures accurate approximation to position-wise low-rank attention maps. This helps FMLA achieve outstanding performance in local awareness. FMLA differs from SOFT, Swin Transformer and Linformer in that it approximates attention maps accurately through lower-rank matrices under the guidance of DCN blocks. In this way, FMLA achieves linear complexity with a flexible receptive field. To reduce noise interference and capture useful information, we present a plug-in mask mechanism to optimize the model at the training and testing stages. The mask mechanism with a proper frequency helps ignore irrelevant details in time series caused by noise and mine diverse shapelets.

Our contributions are summarized below:

- We propose a time-series-specific model, FMLA, which combines the advantages of Transformers and CNN-like algorithms to extract global and local features. Each CLA block in FMLA consists of collaborative multi-heads [4] responsible for analyzing local denoised features from the related DCN block. In addition, online distillation is used to circulate those deformable local features captured by the DCN blocks within the entire FMLA model. Thus, we realize bidirectional circulation in FMLA to dynamically adjust its receptive field in the forward and backward propagation processes.
- In the FMLA architecture, each CLA block compresses the length of the input series under the guidance of the hidden features from the counterpart DCN block. For keys and values in each CLA block, the space complexity can be reduced to a constant level while the positional relations remain unchanged. Compared with the vanilla Transformer, FMLA keeps a flexible receptive field with linear complexity only.
- FMLA adopts the mask mechanism to reduce noise interference by adding random and regular mask layers immediately after the end of each CLA block. We apply self-distillation to random mask layers, which stabilizes and speeds up training. Through position-wise random masks during training and frequency-based regular masks in the inference process, FMLA avoids local optimum and is more robust when addressing TSC problems.
- We compared the proposed model with 11 state-of-the-art TSC algorithms, and our model obtained the best accuracy on 36 out of the 85 UCR2018 datasets and ranked first in terms of average rank. Thanks to the linear complexity structure, FMLA achieves significantly better performance with fewer parameters compared with most variants of the the vanilla Transformer.

The rest of the paper is organized as follows. In Section 2, we review relevant studies on TSC and Transformers, in particular, how to

enhance Transformers with knowledge distillation. The proposed FMLA is detailed in Section 3. In Section 4, performance evaluation and result analysis are provided. Section 5 concludes the work.

2. Related work

First this section reviews traditional and deep learning algorithms for TSC. Second, a number of well-known variants of the Transformer and their trends are given. Thirdly, the existing work on enhancing Transformer-like structures by knowledge distillation is provided.

2.1. Traditional and deep learning TSC algorithms

Traditional Algorithms. Traditional TSC algorithms are mostly based on statistical machine learning, and some important achievements are reviewed below. Lines et al. focused on five temporal modules and introduced HIVE-COTE [25] which integrated several classifiers to hierarchically vote for classification. There were many other ensemble classifiers, such as Rocket [7], and MultiRocket [36]. Rocket used a number of random convolutional kernels to extract diverse features, all of which then went through a linear classifier to obtain final results. Based on Rocket, MultiRocket introduced four additional pooling operators and avoided overfitting by multiple combinations of transformation. Compared with its predecessor, MultiRocket achieved better processing speed and accuracy. Proximity Forest [29], like the random forest, made each decision by multiple random proximity trees. This method used parameterized distance measure to compare unclassified samples with each exemplar randomly chosen from each class so that the decision could go through the related branches of a tree. TS-CHIEF [33] was based on the proximity forest and organized many advanced classifiers in a tree structure.

Deep Learning Algorithms. With a large number of parameters, deep learning algorithms were capable of capturing detailed and various levels of information. Compared with traditional ones, these algorithms achieved better classification results by adaptively learning rich representations from each time series during training. InceptionTime [12] used several convolutional layers and pooling layers for time series analysis. However, this model was only good at capturing local patterns due to the adopted pure convolutional structure. MACNN [3] used the attention mechanism to improve the classification performance of multi-scale CNNs. ABFPN was composed of atrous convolution operators with different dilation rates and skip connections to realize feature fusion in defect detection [46]. In [43], Wu et al. applied time–frequency attention and framewise self-attention to environmental sound classification in the Internet of Things. Chen *et al.* presented DA-Net[2] which adopted dual attention modules to tackle multivariate TSC problems. Zheng *et al.* [49] pretrained the DTCRAE in an unsupervised style to reinforce the TCN for TSC. Huang *et al.* [18] proposed a dual-network-based architecture combining Transformer and ResNet [15] for TSC, with their outputs concatenated for the final classification. This architecture extracted global and local features separately. Xiao *et al.* [45] developed another dual-network architecture, RTFN, placing attention and CNN modules in parallel. Similarly, the features captured were concatenated before the classifier. Dual-network architectures were able to extract sufficient and flexible information from a given time series. However, one branch could not make full use of the hidden states of the other during feature extraction since the final classification results were simply generated by concatenating the outputs of the two branches. Most of these researchers tried to design powerful CNN-based or attention-based neural networks. However, neither could exploit the potential of local perception in CNNs and global perception in attention mechanisms sufficiently. Hence, we are motivated to try layer-wise integration of deformable neural networks and the attention mechanism to thoroughly utilize their hidden states, increasing the classification accuracy.

2.2. Transformer and its variants

Low-complexity Transformers. The vanilla Transformer [40] and its application in computer vision, ViT [8], suffered from quadratic time and space complexity [37]. ViT divided original data into patches which were projected into a low-dimensional space for attention computation. However, it was difficult for a window-based algorithm to always choose suitable window sizes for various lengths of time series. An improper window size can truncate complete shapelets, including those critical ones. This influenced the quality of the features extracted since it was hard to analyze truncated features in TSC, different from those in computer vision. Therefore, many attempts were made to deal with the high-complexity and feature-truncation problems above. Swin Transformer [27] strengthened its local feature extraction ability using a CNN-like architecture. The shift windows technique could, to some extent, relieve the feature-truncation problem. However, Swin Transformer also suffered from the problems CNNs encountered. It had to stack more Swin Transformer blocks and required an additional shift layer per block to expand the receptive field. Neither ViT nor Swin Transformer could well recognize long-span features. In [23], Li et al. used convolutional self-attention for local-feature awareness and LogSparse and restart attention mechanisms for memory reduction. Nevertheless, the LogSparse attention might lose useful information while it was hard for the restart attention to choose an appropriate range for a given time series. All the above variants belonged to local attention mechanisms. As a global attention mechanism, Linformer [41] did not encounter the above problems. It used linear projection to reduce the length of a given input sequence. It was a rough approximation to the attention map in the vanilla Transformer which led to serious information loss [28]. Noise data increased the probability that Linformer converged to a local optimum point [11,23]. SOFT [28] sampled several tokens as queries and achieved a more accurate linear attention through decomposing attention maps. However, it could not reduce the noise interference in time series and one token was usually associated with several others, causing two irrelevant tokens to be identified as related in all the attention algorithms mentioned above. Hence, it was possible that the mismatched tokens were then wrongly assigned larger weight values wrongly [11].

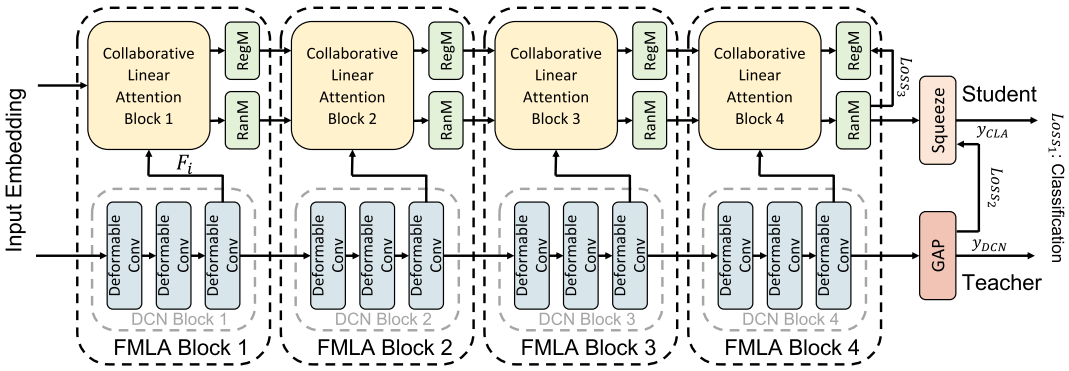


Fig. 1. Structure of our algorithm. We use 4 FMLA blocks as an example.

Transformers enhanced by CNNs. To focus more on the local information, there were several attempts to effectively integrate CNNs into Transformers. ConViT [9] used gate units in each layer, optimizing the weights of the Transformer and CNN blocks in the training process. Thus, the model could adaptively learn the relative importance of each component of different data. Peng et al. [30] found that Transformers were not good at capturing local features and constructed a two-branch structure, Transformer and CNN, where the hidden states of one branch were added to the other commutatively for feature supplementation purposes. On the other hand, traditional CNNs had fixed kernel shapes, which could not extract features with various shapes since local features even in the same class were not always the same under noisy situations. Zhu *et al.* [50] introduced deformable DERT that generated K coordinates for deformable kernels. The linear transformation of each query vector was adopted for attention computation. However, K was a fixed hyperparameter, which could not handle problems with various levels of difficulty. Xia *et al.* proposed a deformable attention module that generated offsets of key and value vectors based on a uniform grid using a lightweight network [44]. In the attention computation, different queries shared shifted keys and values. The deformable attention module was effective in extracting meaningful features from noisy data. The Transformers mentioned above either handle the quadratic complexity or feature extraction, but not both. Therefore, we are motivated to combine the advantages of the variants above and provide our FMLA model with low complexity and flexible feature extraction simultaneously.

2.3. Knowledge distillation in transformers

Knowledge distillation [17] promoted knowledge flows from teacher to student networks, achieving significant parameter compression with acceptable performance degradation. Jiao et al. [19] applied knowledge distillation to a well-known BERT model. They distilled the embedding layer initially and matched layers of the student with those of the teacher according to the ratio of the number of layers in the teacher network to those in the student network, where distillation of attention maps and hidden states was implemented between each pair of layers. DistilBERT [32] was another attempt on distilled the BERT model. The student network had the same architecture as the teacher network but only with half of the layers. DistilBERT achieved similar classification performance as the teacher network and was 60% faster than the teacher. DeiT [38] used a different way to distill from the Transformer by defining a distillation token that was optimized with all other tokens together, just like the CLS token in the BERT model. Experiments showed that it was better for Transformer-like algorithms to use CNNs as the teacher. Microsoft raised BERT-PKD [35] with two distillation strategies, PKD-Last and PKD-Skip. In BERT-PKD, students could either learn from the last K layers or every K layers.

On the other hand, self-distillation [48], where the teacher and student lie in the same model, has been increasingly useful. It can accelerate the training process, compress parameters, and even boost performance. In [26], Liu et al. proposed FastBERT, where they added the same classifier as that of the teacher's after each Transformer block and used the output of the last layer to guide the training of the students. In this way, they realized adaptive inference and reduced the time consumption without performance degradation according to the uncertainty of each level of students. Most of these studies used knowledge distillation to compress the Transformer from different perspectives. However, the capability of knowledge distillation is much more than that. Its applications have not been exploited sufficiently. In this study, we not only use online-distillation to transfer the deformable ability from DCNs to Transformers but also implement self-distillation to stabilize the mask mechanism in our FMLA.

3. Flexible multi-head linear attention (FMLA)

In this section, we first demonstrate the principle of FMLA through the integration between DCN and CLA blocks. We then introduce the mask mechanism, including the inspiration and implementation. Finally, the adopted online distillation technique adopted is detailed.The overview of FMLA is shown in Fig. 1.

3.1. Flexible attention based on deformable convolutional networks

FMLA consists of a DCN block and a CLA block in each layer to extract local and global representations with linear complexity. As

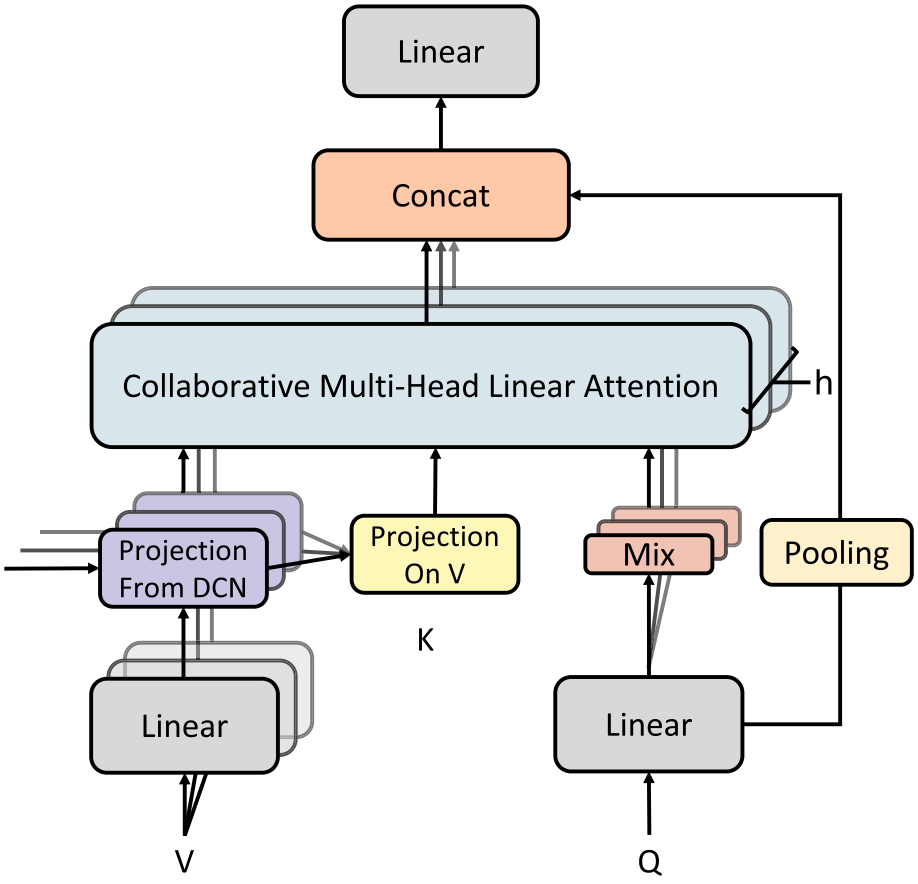


Fig. 3. Each CLA block utilizes the compressed mapping generated from the counterpart DCN block. The unique group of key vectors is generated from several groups of values from different heads of the attention mechanism. The mixing vectors originate from the collaborative attention mechanism to help reduce the redundancy in query vectors.

for a CLA block in Fig. 3, values are regenerated based on the output of the features by the deformable mechanism and keys are generated based on the new values. We assume that there are N_h heads for the attention mechanism. X and Y are the inputs for queries and keys and $X = Y$ in self attention. Eqs. (1), (2), and (3) redefine the implementation of attention computation, generation of values, V_i, and generation of keys, K_i, in head i,$\bar{H}_i$:

$$\bar{H}_i = \text{Attention}\left(XW_i^{\text{Q}}, K_i, V_i\right), \tag{1}$$

$$V_i = F_i Y W_i^{\text{V}}, \tag{2}$$

$$K_i = \text{Conv}(V_i), \tag{3}$$

where W_i^{Q} and W_i^{V} are projection matrices. The compressed mapping, F_i, is defined in Eq. (4)

Since K_i is generated based on V_i through the pointwise $\text{Conv}()$ operation in each head, FMLA avoids unnecessary pair-wise inner-product computation by directly constructing position-wise relations. Moreover, there may be several quite different sub-optimal strategies for compression in algorithms like SOFT or Linformer if we train the model on the same dataset several times, which is caused by the instability of sampling or approximation process and is referred to as local optima. Therefore, a proper mapping strategy can reduce model complexity and enable linear attention (CLA in this paper) to extract more flexible features with less interference. To this end, we project the input sequence in each CLA block based on the output of its counterpart DCN block, $h_{\text{DCN}}^{G_i}$, as shown in

$$F_i = \text{Conv}_i\left(h_{\text{DCN}}^{G_i}\right), \tag{4}$$

where F_i is the well-designed projection matrix for values in head i. $\text{Conv}_i()$ is the i-th pointwise group convolution that projects feature maps, $h_{\text{DCN}}^{G_i}$, in each DCN block to weights, F_i. G_i indexes the input channels of the i-th group convolution. $h_{DCN}^{G_i}$ is the hidden states of the group i in the counterpart DCN block. The number of channels, C, in F_i equals the length of compressed values in the CLA block,

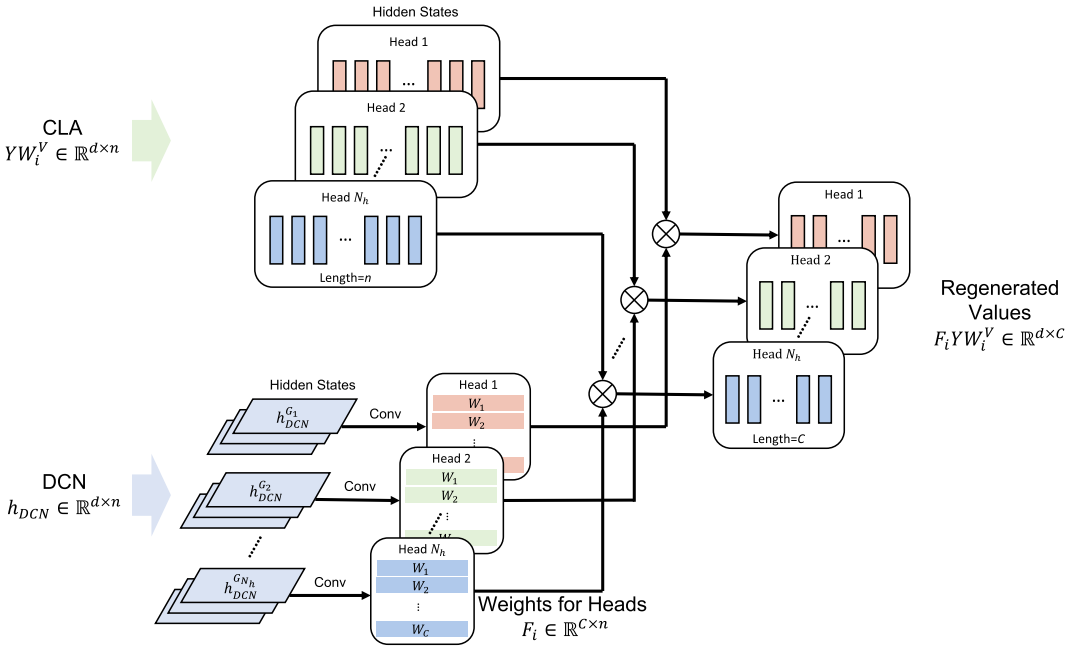


Fig. 2. Components of our flexible multi-head linear attention.

namely each channel, W_i, represents the weights to generate each vector in the regenerated values as shown in Fig. 2. In this way, each channel group guides one head of the multi-head attention mechanism in each CLA block. As a special branch of CNNs, DCNs are also considered as low-level feature extractors. Integrating DCN blocks into CLA blocks enhances their ability to mine local representations and filter noise data. The linear projections in FMLA clearly lose much information due to the large ratio of compression, which can make higher layers difficult to train. Therefore, we apply the residual pooling connection from multiscale vision transformer (MViT) [10] to FMLA. Originally, this technique contains pooling operations on the queries, keys, and values and the residual connection from queries to the output of the related attention block to reduce the complexity and facilitate the training process. To avoid losing too much information in the linear projection process of FMLA and minimize the noise influence in the queries, we only add pooled query matrices to the output of each attention block as a low-resolution complement as shown in Eq. (5), where $\bar{H}_i$ is the output of the i-th head and XW_i^{Q} is the queries in the head i. Pooling is the average pooling operation and W_O is the compressed mapping of the aggregated information from N_h heads in each FMLA block.

$$\mathrm{MultiHead}\left(X,Y\right) = \underset{i\in[N_h]}{Concat}\left[\bar{H}_i + \mathrm{Pooling}\left(XW_i^{\mathrm{Q}}\right)\right]W_O \tag{5}$$

In the canonical attention mechanism, each head extracts information individually. The output of all heads is concatenated with a squeeze operation for the same-size output, which results in redundant features between heads. To reduce the redundancy, mix vectors are used to represent the unique information of each head before the dot product calculation, as written in Eqs. (6) and (7), which is referred to as the collaborative multi-head attention [4]:

$$\bar{H}_i = \mathrm{Attention}\left(X\widetilde{W}^{\mathrm{Q}}\mathrm{diag}(m_i), Y\widetilde{W}^{\mathrm{K}}, YW_i^{\mathrm{V}}\right), \tag{6}$$

$$\mathrm{CollabHead}\left(X,Y\right) = \mathrm{Concat}_{i\in[N_h]}\left[\bar{H}_i\right]W_O, \tag{7}$$

where $\widetilde{W}^Q$ and $\widetilde{W}^K$ are shared across multiple heads. m_i is the mix vector for the unique information extracted in each head and $\mathrm{diag}()$ extends the mix vector to a square matrix. W_O in Eq. (7) is the linear projection to aggregate the outputs of different heads.

As known, it is necessary for feature extractors to distinguish noise from shapelets since each head focuses on some of the useful information and unavoidable noise. With the collaboration of multiple heads, a feature extractor can summarize the common features that appear in each head with low noise interference. Therefore, we implement the collaborative multi-head attention, Eq. (6), in FMLA as Eq. (8) where m is the mix vector for distinguishing the unique knowledge learned by different heads so that projection for each query per head is no longer needed. Since only one key is shared between multiple queries and values in the collaborative attention, we can naturally obtain one key generated by the pointwise linear projection from all values as defined in Eq. (9), retaining the position-wise relation between the keys and values. Thus, the attention maps calculated from the dot-product of queries and keys are directly used for weighting the values, YW_i^V.

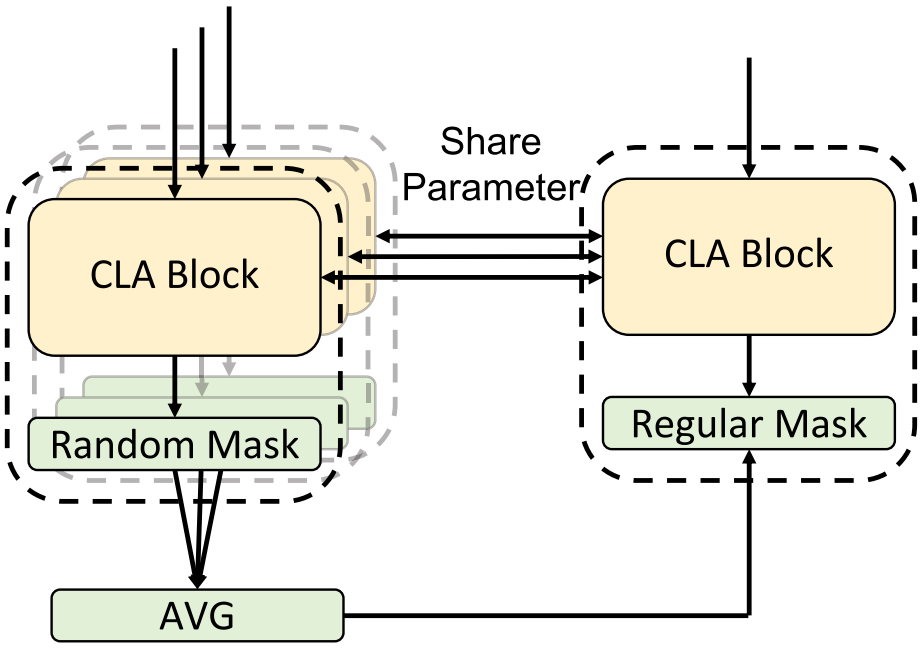


Fig. 5. Implementation of self-distillation in the mask mechanism.

$$H_i = \text{Attention}\left(X\widetilde{W}^{Q}\text{diag}\left(m_i\right), \widehat{K}, F_i Y W_i^{V}\right), \tag{8}$$

$$\widehat{K} = \text{Conv}\left(\text{Concat}\left(F_1 V_1 W_1^{V}, F_2 V_2 W_2^{V}, \ldots, F_{N_h} V_{N_h} W_{N_h}^{V}\right)\right), \tag{9}$$

where *Conv*() is a compressed mapping to aggregate the results of multiple heads and the shape of $\widehat{K}$ is the same as that of $F_1 V_1 W_1^{V}$.

3.2. Mask mechanism

Time series are generally generated with a fixed sampling interval. One can assume that the macroscopic waveforms and critical shapelets are retained as long as we properly lengthen the sampling interval and lower the sampling frequency in most cases. In TSC, there are no such necessary data points that directly affect the classification result since time series specify a continuous process. Recent research [24] shows that only a few query-key pairs contribute to each related task, which means we may reduce the influence of noise and the redundancy of models by dropping the data in multiple positions of a given time series. In each layer of FMLA, this paper uses position-wise random masks based on Bernoulli distribution in the training process and frequency-based regular masks in the testing process. To be specific, different heads use different random masks and the positions masked are resampled in each iteration. In this way, FMLA will not overfit specific local features and there is no need to worry about the permanent loss of important information. For instance, we randomly mask the data in 50% of the positions in the sequence, like $X = \{x_1, x_2, 0, 0, x_5, 0, x_7, x_8, 0, 0\}$, during training. In the inference process, the sequence is regularly masked according to the frequency adopted in the training process, that is, 50% in the instance above. We mask the data at the second (or first) position of each two consecutive positions during the testing process, like $X = \{x_1, 0, x_3, 0, x_5, 0, x_7, 0, x_9, 0\}$. Thus, low-frequency resampling is achieved. The mask technique used in this paper can also be considered as a grouped version of Dropout [34], where we group all weights according to their associated positions and drop those weights in the chosen groups together. That's why the mask mechanism has the ability to resist overfitting (like the original Dropout). Actually, our experimental results show the mask mechanism can work well with Dropout. Details can be found in SubSection 4.2.

Clearly, our mask mechanism has randomness due to the predefined frequency, which may prevent our model from convergence. Self-distillation is invoked to stabilize and speed the training as illustrated in Fig. 5. The input time series is fed to the model with random mask layers a predefined number of times. We calculate the average output value and distill the related knowledge to the model above with regular mask layers for the consistency of training and testing as shown in Eq. (10). In this way, we not only stabilize the training and make use of the advantages of Bagging in ensemble learning, that is, variance reduction and random fluctuation stabilization.

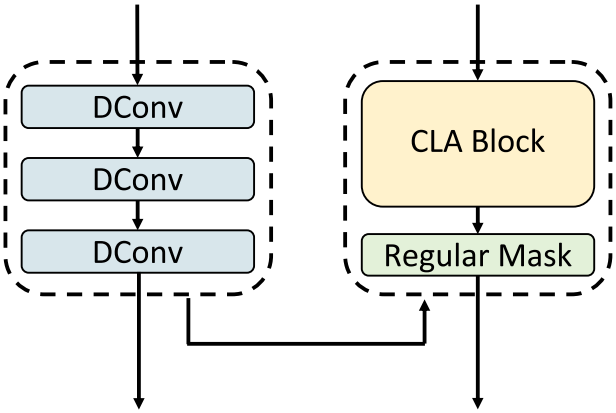


Fig. 4. Implementation of online knowledge distillation at the last layer of FMLA.

$$Loss_1 = \mathrm{D_{KL}}\left(\frac{1}{N}\sum_{t=1}^{N}\mathrm{RandomMask}\left(\mathrm{FMLA}\left(x\right)\right),\right.$$
$$\left.\mathrm{RegularMask}(\mathrm{FMLA}(x))\right), \tag{10}$$

where functions $\mathrm{RandomMask}()$ and $\mathrm{RegularMask}()$ indicate whether random or regular masks are implemented in FMLA. N represents a predefined number of times each input time series are fed to the model with random mask layers.

3.3. Deformable attention by online distillation

According to [38], CNN-like algorithms can be excellent teachers for supervising Transformers, which inspired us to distill knowledge from the output of the DCN and transfer it to the entire FMLA model, that is, $Loss_2$ in Eq. (11). Compared with CNNs, DCNs have better representation ability and can reduce the influence of noise, help improve the locality awareness of FMLA and concentrate more on deformable features. Thus, the DCN in FMLA is regarded as the teacher. We add the outputs of the last CLA and DCN blocks for classification, which helps retain the DCN independence. By using DCN blocks to guide the compressed mapping in the forward propagation and adopting the online distillation loss to optimize the backward propagation, we realize a bidirectional circulation of knowledge mined by the DCN, which enhances the vanilla attention mechanism in terms of locality awareness, and noise and complexity reduction.

$$Loss_2 = \alpha \mathrm{D_{KL}}(y_{\mathrm{DCN}}, y_{\mathrm{CLA}}), \tag{11}$$

$$\mathrm{D_{KL}}\left(p, q\right) = \sum_{x\in X} p\left(x\right)\log\frac{p(x)}{q(x)}, \tag{12}$$

where y_{DCN} and y_{CLA} are the outputs of DCN and CLA branches, respectively. $\mathrm{D_{KL}}$ is the Kullback–Leibler divergence implemented by Eq. (12) for two probability distributions coming from the outputs of two neural networks, $p(x)$ and $q(x)$. This is widely used as the loss in knowledge distillation to measure the difference between two distributions, i.e., using $q(x)$ to fit $p(x)$.

As the loss decreases during training, the FMLA model is likely to pay more attention to the deformable local features captured by DCN blocks. Hyperparameter α ($0 \leqslant \alpha \leqslant 1$) in Eq. (11) is used to adjust how much knowledge CLA blocks learn from the DCN. In other words, α indicates to what extent FMLA focuses on local patterns. A smaller α tends to emphasize global trends while a larger one is more helpful for extracting local features. In this way, our model achieves a promising perception of global and local patterns. The skeleton of our online distillation method is shown in Fig. 4. At the last layers of FMLA, the knowledge from the DCN is distilled to teach regular-mask enhanced CLA blocks.

$Loss_3$ in Eq. (13) is the cross entropy for classification and implemented as Eq. (14) for two probability distributions coming from the outputs of two neural networks, respectively. The final loss function consisting of $Loss_1, Loss_2$, and $Loss_3$, for optimizing the algorithm is shown in Eq. (15).

$$Loss_3 = \mathrm{H}(\widehat{y}, y) \tag{13}$$

$$\mathrm{H}\left(p, q\right) = \sum_{x\in X} p\left(x\right)\log\left(q\left(x\right)\right) \tag{14}$$

$$Loss = Loss_1 + Loss_2 + Loss_3 \tag{15}$$

Table 1
The top-1 accuracy results of the five models on 85 UCR2018 datasets.

Dataset	ResNet-Transformer	CLAwDCN	CLAwDCN-M	CLAwDCN-M-SD	FMLA
Adiac	0.849105	**0.810742**	0.808184	0.836317	**0.854220**
ArrowHead	**0.891429**	**0.771429**	0.834286	0.834286	0.857143
Beef	**0.866667**	0.633333	0.800000	0.700000	0.833333
BeetleFly	**1.000000**	**0.800000**	**0.850000**	**1.000000**	**1.000000**
BirdChicken	0.950000	0.850000	**0.950000**	**1.000000**	**1.000000**
Car	0.866667	**0.883333**	0.900000	0.900000	**0.916667**
CBF	**1.000000**	0.987778	**0.997778**	0.998889	**1.000000**
ChlorineCon.	0.409375	0.782813	0.812500	**0.829948**	0.815365
CinCECGTorso	**0.890580**	**0.902899**	0.782609	0.805797	0.832609
Coffee	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**
CricketX	**0.810256**	0.751282	0.771795	0.738462	0.787179
CricketY	**0.825641**	0.769231	0.771795	0.776923	0.794872
CricketZ	0.128205	0.764103	0.761538	0.761538	**0.800000**
DiatomSizeRe.	0.379085	0.630719	0.934641	**0.937908**	**0.996732**
DistalPhalanxO.A.G	**0.467626**	0.769784	0.798561	0.798561	**0.805755**
DistalPhalanxO.C.	0.822464	0.797101	0.793478	0.822464	**0.826087**
Earthquakes	0.762590	**0.798561**	0.776978	**0.805755**	0.798561
ECG200	0.940000	0.920000	0.930000	0.930000	**0.960000**
ECG5000	0.944222	0.942000	0.939778	**0.945333**	0.944222
ECGFiveDays	**1.000000**	**1.000000**	**0.930314**	**0.974448**	**0.977933**
FaceFour	**0.977273**	0.806818	0.943182	0.795455	**0.977273**
FacesUCR	0.926829	0.949268	0.952683	0.953659	**0.954146**
FordA	0.517424	**0.940152**	0.935606	0.935606	0.939394
FordB	**0.838272**	0.823457	0.828395	0.822222	0.827160
GunPoint	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**
Ham	0.619048	0.809524	**0.938095**	0.809524	0.838095
HandOutlines	0.835135	0.948649	0.943243	0.951351	**0.959459**
Haptics	**0.600649**	0.519481	0.535714	0.564935	**0.564935**
Herring	0.656250	**0.703125**	0.703125	**0.750000**	0.718750
InsectWingbeatS.	0.535859	0.636869	0.461616	0.532323	**0.835341**
ItalyPowerDemand	0.962099	0.967930	0.969874	0.969874	**0.971817**
Lightning2	0.754098	0.885246	0.836066	0.836066	**0.901639**
Lightning7	0.383562	**0.821918**	0.780822	**0.849315**	**0.849315**
Mallat	0.934328	**0.882729**	0.945416	0.957783	**0.975693**
Meat	**1.000000**	0.916667	**0.983333**	**0.983333**	**1.000000**
MedicalImages	0.759211	**0.767105**	0.761842	0.756579	0.756579
MiddlePhalanxO.A.G.	**0.623377**	0.662338	0.655844	**0.675325**	0.668831
MiddlePhalanxO.C.	0.848797	0.862543	0.848797	**0.872852**	0.869416
MiddlePhalanxTW	0.551948	0.580909	0.590909	0.610390	**0.623377**
MoteStrain	**0.937700**	0.861821	0.901757	0.881789	0.930511
OliveOil	0.933333	0.933333	**0.933333**	0.933333	**0.966667**
Plane	**0.371492**	**1.000000**	**1.000000**	**1.000000**	**1.000000**
ProximalPhalanxO.A.G.	0.882927	**0.897561**	0.887805	0.892683	**0.897561**
ProximalPhalanxO.C.	0.683849	0.924399	0.931271	0.927835	**0.938144**
ProximalPhalanxTW	0.819512	0.848780	0.848780	**0.853659**	**0.853659**
ShapeletSim	**0.888889**	**0.922222**	**0.955556**	**1.000000**	**1.000000**
ShapesAll	0.921667	0.928333	0.930000	0.921667	**0.933333**
SonyAIBORobotSur.1	0.708819	0.960067	**0.956739**	0.976705	**0.985025**
SonyAIBORobotSur.2	**0.984260**	0.940189	0.951731	0.965373	**0.965373**
Strawberry	0.986486	0.986486	0.983784	0.986486	**0.989189**
SwedishLeaf	**0.969600**	**0.966400**	0.958400	0.963200	0.968000
Symbols	0.976884	**0.849246**	0.935678	0.955779	**0.982915**
SyntheticControl	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**
ToeSegmentation1	0.978070	**0.850877**	0.872807	0.921053	**0.982456**
ToeSegmentation2	**0.953846**	0.892308	**0.907692**	0.938462	0.938462
Trace	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**
TwoLeadECG	**1.000000**	**0.979807**	**1.000000**	**1.000000**	**1.000000**
TwoPatterns	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**
UWaveGestureL.All	**0.939978**	0.892797	0.919598	0.904802	0.893076
UWaveGestureL.X	**0.810999**	0.789782	0.800391	0.805974	0.805974
UWaveGestureL.Y	0.671413	0.687046	0.689280	**0.698492**	**0.698492**
UWaveGestureL.Z	**0.760469**	**0.758515**	0.735064	0.719314	0.738693
Wafer	**0.998540**	0.997080	0.993835	0.993511	0.995620
Wine	0.870370	0.851852	0.907407	0.814815	**0.962963**
WordSynonyms	**0.636364**	0.564263	0.559561	0.540752	0.592476
ACSF1	**0.930000**	0.910000	**0.910000**	0.900000	**0.930000**
BME	**1.000000**	0.973333	**1.000000**	**0.986667**	**1.000000**
Chinatown	0.985507	0.976676	0.982507	**0.988338**	**0.988338**

(*continued on next page*)

Table 1 (*continued*)

Dataset	ResNet-Transformer	CLAwDCN	CLAwDCN-M	CLAwDCN-M-SD	FMLA
Crop	**0.746012**	0.728690	0.725238	0.705952	0.725119
DodgerLoopDay	0.462500	**0.575000**	0.575000	0.575000	**0.612500**
DodgerLoopGame	0.550725	0.818841	**0.920290**	0.913043	0.905797
DodgerLoopWeekend	0.949275	0.971014	0.971014	0.978261	**0.985507**
GunPointAgeSpan	**1.000000**	0.993671	0.996835	**1.000000**	**1.000000**
GunPointMaleV.F.	0.996835	**1.000000**	**1.000000**	**1.000000**	**1.000000**
GunPointOldV.Y.	**1.000000**	**1.000000**	**1.000000**	**0.993651**	**1.000000**
InsectEPGRegularT.	**1.000000**	0.911647	**0.947791**	**1.000000**	**1.000000**
InsectEPGSmallT.	**0.971888**	0.714859	0.823293	0.763052	0.919679
MelbournePedestrian	**0.904898**	0.899549	0.893399	0.895449	0.889299
PowerCons	0.927778	**0.938889**	0.961111	**0.977778**	**0.977778**
Rock	0.820000	**0.880000**	0.800000	**0.780000**	0.780000
SemgHandG.Ch2	0.848333	0.836667	0.818333	0.845000	**0.870000**
SemgHandM.Ch2	0.391111	0.511111	0.431111	0.511111	**0.533333**
SemgHandS.Ch2	0.666667	**0.831111**	0.713333	**0.831111**	0.800000
SmoothSubspace	0.993333	0.986667	**0.986667**	**1.000000**	**1.000000**
UMD	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**
MeanAcc	0.823040	0.856361	0.867943	0.873686	0.893739

Table 2
Performance comparison of 5 algorithms on 85 UCR2018 datasets in terms of 'Win'/'Tie'/'Lose' metric.

Algorithm 1 vs Algorithm 2	Win	Tie	Lose	Best
ResNet-Transformer vs CLAwDCN	36	10	39	46
CLAwDCN vs CLAwDCN-M	40	16	29	56
CLAwDCN-M vs CLAwDCN-M-SD	44	20	21	64
CLAwDCN-M-SD vs FMLA	48	26	11	74

Note: 'win'/'tie'/'lose' indicate on how many datasets Alg. 2 performs better than, equivalent to and worse than Alg. 1, respectively.

4. Experiments and analysis

We first introduce the experimental setup and ablate the three critical components of FMLA, including the DCN-enhanced multi-head attention, mask mechanism, and online distillation. Then, the complexity of our model is analyzed and its comparison with several algorithms is discussed.

4.1. Experimental setup

Datasets. We conduct extensive experiments to evaluate the main components and overall performance of FMLA on 85 out of the 128 UCR2018 datasets [6]. The 85 datasets are composed of 65 'short' or 'medium' and 20 'long' time series datasets. 'Existing SOTA', 'TS-CHIEF' and 'MACNN' only have publicly available benchmark results on 65 datasets for performance comparison.

Performance metrics. We used 'Win'/'Tie'/'Lose', MeanAcc, and Avg_rank, as the performance metrics to compare FMLA with 11 state-of-the-art TSC algorithms. These metrics are all based on top-1 accuracy. 'Win' indicates the number of datasets on which an algorithm achieves the unique best result among all algorithms; 'Tie' means the number of datasets on which the algorithm achieves the same best result as other algorithms; 'Lose' represents the number of datasets on which the algorithm does not achieve the best result. For an arbitrary algorithm, its 'Best' value is the summation of the corresponding 'Win' and 'Tie' values. 'MeanAcc' is the average accuracy of the algorithm on all datasets and 'Avg_rank' is the average rank value based on the Friedman test.

Parameter settings. There were four FMLA blocks in our experiments and each block had four heads. We compressed the length of each input sequence to 16 using projection matrices generated by each DCN block in most cases. For the mask mechanism, we chose the frequency 50% for each dataset initially and masked the data at 50% positions for computation in each CLA block. As for activation functions, we used the rectified linear unit (ReLU) function in DCN blocks and the Gaussian error linear unit (GELU) [16] function in each CLA block. We used 128 kernels in the first two DCN blocks and 64 in the last two as [18] suggested, and we set the kernel size to 3 for the feature extraction and position generation layers. Note that the parameter settings above were used in the ablation study and overall performance comparison. There may be small adjustments to the hyperparameter setting for a few datasets, e.g., for those with complex and long input sequences, more heads may be used in the attention mechanism and more positions of data may be masked in the mask mechanism.

4.2. Ablation study

To validate the effectiveness of the techniques proposed in this paper, we conducted an incremental ablation study with 85

UCR2018 datasets. The details are shown in Table 1 and a summary is provided in Table 2. The incremental models are listed below.

- ResNet-Transformer: a well-known dual-network model that consists of a ResNet branch and a vanilla Transformer branch, i.e., vanilla Transformer-based model [18].
- CLAwDCN: the collaborative linear attention with the DCN integrated, in SubSection 3.1.
- CLAwDCN-M: CLAwDCN with mask mechanism in SubSection 3.2.
- CLAwDCN-M-SD: CLAwDCN-M with self-distillation in SubSection 3.2.
- FMLA: CLAwDCN-M-SD with online distillation in SubSection 3.3, i.e., the proposed model in this paper.

The performance comparison between algorithms regarding top-1 accuracy is summarized in Table 2. First, we compared CLAwDCN and ResNet-Transformer [18]. The two models have the same structure. CLAwDCN and ResNet-Transformer are close runners with 85 datasets considered. CLAwDCN improves the mean accuracy to about 0.856163 as shown in Table 1. CLAwDCN won in 36 cases and lost in 39 cases as shown in Table 2, showing that CLAwDCN does not lead to performance deterioration compared with the vanilla Transformer-based model. ResNet-Transformer extracts local and global features separately, which is an advanced voting algorithm based on the two kinds of features. Compared to CNNs, DCNs can emphasize local features with various shapes, where the compressed matrices for each head are generated by different channel groups in the associated DCN block. Thus, CLAwDCN can calculate the similarity between useful data points based on the inputs filtered by the DCN blocks, which helps avoid the influence of fluctuation and capture more accurate information rather than the noise in time series.

We evaluated the effectiveness of the mask mechanism without self-distillation by comparing CLAwDCN and CLAwDCN-M. Different mask frequency values can be used in the shallow and deep layers of CLAwDCN-M. For simplicity purposes, we fixed the frequency to 50% in all layers in the experiment. The mean accuracy was improved by CLAwDCN-M to 0.873686 as shown in Table 1. The performance comparison between CLAwDCN and CLAwDCN-M is shown in Table 2. CLAwDCN-M won 40 cases and lost 29 ones. If we only use the dropout [34] technique which randomly drops the weights of neural networks, the output of each position the weights associated with still exists and hence the relevant influence also remains. In the attention mechanism, the classification-related data generally obtains higher weights after the dot-product operation. It means that even if data in some specific positions are dropped, they may also remain in other positions in the form of a weighted summation.

To study the performance of the mask mechanism with different frequencies of masking, we randomly selected 30 UCR2018 datasets: 20 ‘short-medium’ and 10 ‘long’ datasets. The top-1 accuracy results of CLAwDCN-M with different mask frequencies are provided in Table 3. It is clear that CLAwDCN-M with a mask frequency of 50% performed the best. With 50% of data points lost, the

Table 3
Top-1 accuracy results of CLAwDCN-M with different mask frequencies.

Datasets	Length	Mask 10%	Mask 30%	Mask 50%	Mask 70%	Mask 90%
Adiac	176	0.831202	0.790281	**0.83376**	0.820972	0.828645
ArrowHead	251	0.72	**0.8**	0.731429	0.628571	0.782857
Beef	470	0.7	0.633333	**0.833333**	0.666667	0.8
CBF	128	0.998889	**1**	**1**	0.997778	**1**
ChlorineConcentration	166	0.821615	**0.846875**	0.823958	0.826562	0.817708
CricketY	300	0.774359	**0.815385**	0.782051	0.802564	0.776923
DistalPhalanxOutlineCorrect	80	**0.822464**	0.804348	0.807971	0.804348	0.800725
FaceFour	350	0.943182	0.920455	**0.965909**	0.897727	0.931818
GunPoint	150	0.993333	0.993333	**1**	**1**	**1**
InsectWingbeatSound	256	0.521212	**0.536869**	0.534848	0.521717	0.513636
MiddlePhalanxOutlineAgeGroup	80	0.662338	0.675325	0.668831	0.662338	**0.681818**
ProximalPhalanxOutlineAgeGroup	80	0.878049	0.882927	0.882927	**0.892683**	0.887805
Strawberry	235	**0.989189**	0.981081	0.983784	0.983784	0.983784
Symbols	398	0.921608	0.907538	0.98794	0.948744	**0.99196**
SyntheticControl	60	**1**	**1**	**1**	**1**	**1**
TwoLeadECG	82	**1**	0.998244	**1**	0.999122	0.999122
Wine	234	0.888889	0.777778	0.87037	0.796296	**0.944444**
BME	128	**1**	**1**	**1**	**1**	**1**
DodgerLoopDay	288	**0.575**	0.55	**0.575**	0.5	0.5125
DodgerLoopGame	288	0.92029	0.833333	**0.927536**	0.681159	0.92029
BeetleFly	512	0.75	0.8	**0.85**	0.65	0.8
BirdChicken	512	0.85	**0.95**	**0.95**	0.9	0.9
Car	577	0.833333	0.916667	**0.95**	0.883333	0.933333
CinCECGTorso	1639	0.87971	0.886232	0.832609	0.778986	**0.942029**
Earthquakes	512	0.791367	**0.805755**	0.798561	**0.805755**	0.791367
FordA	500	0.934091	0.934848	0.935606	**0.941667**	0.939394
FordB	500	**0.832099**	0.818519	0.828395	0.806173	0.819753
Haptics	1092	0.529221	0.516234	0.538961	**0.558442**	0.545455
Herring	512	**0.71875**	0.65625	**0.71875**	0.6875	0.671875
Lightning2	637	**0.901639**	0.885246	0.885246	0.885246	**0.901639**
UWaveGestureLibraryAll	945	**0.928523**	0.912619	0.914015	0.911223	0.924065
MeanAcc		0.83581781	0.83320887	**0.85199323**	0.814172806	0.84977242

Table 4
The top-1 accuracy results of the 12 models on 85 UCR2018 datasets.

Dataset	Existing SOTA	Best:lstm-fcn	Vanilla: Transformer	ResNet-Trans1	ResNet-Trans2	ResNet-Trans3	TS-CHIEF	ResNet50 SC	Inception	ROCKET	MACNN	FMLA
Adiac	0.857000	**0.869565**	0.843990	0.849105	0.849105	0.849105	0.798000	0.844000	0.841432	0.783376	0.824000	0.854220
ArrowHead	0.880000	**0.925714**	0.891429	0.891429	0.891429	0.897143	0.832700	0.885700	0.845714	0.814286	0.863000	0.857143
Beef	0.900000	0.900000	0.866667	0.866667	0.866667	0.866667	0.706100	0.733300	0.700000	0.833333	**0.933000**	0.833333
BeetleFly	0.950000	**1.000000**	**1.000000**	0.950000	**1.000000**	0.700000	0.913600	0.900000	0.800000	0.900000	**1.000000**	**1.000000**
BirdChicken	0.950000	0.950000	**1.000000**	0.900000	0.950000	**1.000000**	0.909100	0.900000	0.950000	0.900000	**1.000000**	**1.000000**
Car	0.933000	**0.966667**	0.950000	0.883333	0.866667	0.300000	0.854500	0.883300	0.883333	0.846667	0.917000	0.916667
CBF	**1.000000**	0.996667	**1.000000**	0.997778	**1.000000**	**1.000000**	0.997900	0.904400	0.998889	**1.000000**	**1.000000**	**1.000000**
ChlorineCon.	0.872000	0.816146	0.849479	0.863281	0.409375	0.861719	0.716700	0.784400	0.876563	0.814531	**0.888000**	0.815365
CinCECGTorso	**0.994900**	0.904348	0.871739	0.656522	0.890580	0.310870	0.983200	0.891300	0.853623	0.836159	0.886000	0.832609
Coffee	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**
CricketX	0.821000	0.792308	0.838462	0.800000	0.810256	0.800000	0.813800	0.735900	0.853846	0.819487	**0.862000**	0.787179
CricketY	0.825600	0.802564	0.838462	0.820513	0.825641	0.807692	0.801900	0.735900	0.851282	0.852308	**0.869000**	0.794872
CricketZ	0.815400	0.807692	0.820513	0.805128	0.128205	0.100000	0.834000	0.728200	**0.861538**	0.855897	0.121000	0.800000
DiatomSizeRe.	0.967000	0.970588	0.993464	**0.996732**	0.379085	**0.996732**	0.973000	0.937900	0.934641	0.969935	0.977000	**0.996732**
DistalPhalanxO.A.G	**0.835000**	0.791367	0.812950	0.776978	0.467626	0.776978	0.746200	0.784200	0.733813	0.758993	0.768000	0.805755
DistalPhalanxO.C.	0.820000	0.791367	0.822464	0.822464	0.822464	0.793478	0.782300	0.815200	0.782609	0.769565	0.786000	**0.826087**
Earthquakes	0.801000	**0.812950**	0.755396	0.755396	0.762590	0.755396	0.748200	0.777000	0.741007	0.748201	0.755000	0.798561
ECG200	0.920000	0.910000	0.940000	0.950000	0.940000	0.930000	0.861800	0.870000	0.930000	0.906000	0.920000	**0.960000**
ECG5000	0.948200	0.948222	0.941556	0.943556	0.944222	0.940444	0.945400	0.945800	0.940889	0.947156	**0.949000**	0.944222
ECGFiveDays	**1.000000**	0.987224	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	0.816500	**1.000000**	**1.000000**	**1.000000**	0.977933
FaceFour	**1.000000**	0.943182	0.954545	0.965909	0.977273	0.215909	**1.000000**	0.727300	0.954545	0.977273	0.966000	0.977273
FacesUCR	0.958000	0.941463	0.957561	0.947805	0.926829	0.951220	0.966300	0.777100	0.971220	0.961415	**0.980000**	0.954146
FordA	0.972700	**0.976515**	0.948485	0.946212	0.517424	0.940909	0.941000	0.938600	0.961364	0.944394	0.955000	0.939394
FordB	**0.917300**	0.792593	0.838272	0.830864	0.838272	0.823457	0.829600	0.822200	0.861728	0.805062	0.874000	0.827160
GunPoint	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**
Ham	0.781000	0.809524	0.761905	0.780952	0.619048	0.514286	0.715200	0.790500	0.714286	0.725714	0.829000	**0.838095**
HandOutlines	0.948700	0.954054	0.937838	0.948649	0.835135	0.945946	0.932200	0.951400	0.954054	0.942432	0.951000	**0.959459**
Haptics	0.551000	0.558442	0.564935	0.545455	**0.600649**	0.194805	0.516800	0.516200	0.548701	0.524026	0.542000	0.564935
Herring	0.703000	**0.750000**	0.703125	0.734375	0.656250	0.703125	0.588100	0.687500	0.671875	0.692188	0.688000	0.718750
InsectWingbeatS.	0.652500	0.668687	0.522222	0.642424	0.535859	0.536364	0.642900	0.617700	0.638889	0.656818	0.647000	**0.835341**
ItalyPowerDemand	0.970000	0.963071	0.965015	0.969874	0.962099	0.971817	0.970300	0.960200	0.965015	0.969582	**0.972000**	0.971817
Lightning2	0.885300	0.819672	0.852459	0.852459	0.754098	0.868852	0.748100	0.852500	0.770492	0.759016	0.820000	**0.901639**
Lightning7	0.863000	**0.863014**	0.821918	0.849315	0.383562	0.835616	0.763400	0.849300	0.835616	0.823288	0.863000	0.849315
Mallat	0.980000	**0.980810**	0.977399	0.975267	0.934328	0.979104	0.975000	0.932600	0.955224	0.955949	0.980000	0.975693
Meat	**1.000000**	0.883333	**1.000000**	**1.000000**	**1.000000**	**1.000000**	0.887900	**1.000000**	0.933333	0.948333	**1.000000**	**1.000000**
MedicalImages	0.792000	0.798684	0.780263	0.765789	0.759211	0.789474	0.795800	0.786800	0.794737	**0.799474**	0.783000	0.756579
MiddlePhalanxO.A.G.	**0.814400**	0.668831	0.655844	0.662338	0.623377	0.662338	0.583200	0.636400	0.551948	0.590260	0.617000	0.668831
MiddlePhalanxO.C.	0.807600	0.841924	0.848797	0.848797	0.848797	0.835052	0.853500	0.852200	0.817869	0.838488	0.838000	**0.862543**
MiddlePhalanxTW	0.612000	0.603896	0.564935	0.577922	0.551948	0.623377	0.550200	**0.623400**	0.512987	0.560390	0.591000	0.623377
MoteStrain	**0.950000**	0.938498	0.940895	0.916933	0.937700	0.200000	0.947500	0.840300	0.886581	0.914617	0.082000	0.930511
OliveOil	0.933300	0.766667	**0.966667**	0.900000	0.933333	0.900000	0.887900	0.733300	0.833333	0.916667	0.900000	**0.966667**
Plane	**1.000000**	**1.000000**	**1.000000**	**1.000000**	0.371492	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**
ProximalPhalanxO.A.G.	0.883200	0.887805	0.887805	0.892683	0.882927	0.892683	0.849700	0.887800	0.848780	0.855610	0.854000	**0.897561**
ProximalPhalanxO.C.	0.918000	0.931271	0.931271	0.931271	0.683849	0.924399	0.888200	0.924400	0.931271	0.898969	0.924000	**0.938144**
ProximalPhalanxTW	0.815000	0.843902	0.819512	0.814634	0.819512	0.819512	0.818600	0.804900	0.775610	0.816585	0.792000	**0.853659**
ShapeletSim	**1.000000**	**1.000000**	**1.000000**	0.911111	0.888889	0.977778	**1.000000**	0.555600	0.955556	**1.000000**	**1.000000**	**1.000000**
ShapesAll	0.918300	0.905000	0.923333	0.876667	0.921667	0.933333	0.930000	0.885000	0.928333	0.906833	**0.940000**	0.933333

(*continued on next page*)

Table 4 (*continued*)

Dataset	Existing SOTA	Best:lstm-fcn	Vanilla: Transformer	ResNet-Trans1	ResNet-Trans2	ResNet-Trans3	TS-CHIEF	ResNet50 SC	Inception	ROCKET	MACNN	FMLA
SonyAIBORobotSur.1	0.985000	0.980525	**0.988353**	0.978369	0.708819	0.985025	0.826400	0.880200	0.868552	0.922463	0.985000	0.985025
SonyAIBORobotSur.2	0.962000	0.972718	0.976915	0.974816	**0.984260**	0.976915	0.924800	0.814300	0.946485	0.912592	0.962000	0.965373
Strawberry	0.976000	0.986486	0.986486	0.986486	0.986486	0.986486	0.966300	0.981100	0.983784	0.981351	0.976000	**0.989189**
SwedishLeaf	0.966400	**0.979200**	0.977200	0.972800	0.969600	0.966400	0.965500	0.968000	0.974400	0.964000	0.963000	0.968000
Symbols	0.966800	**0.987940**	0.979900	0.970854	0.976884	0.252261	0.976600	0.971900	0.980905	0.974271	0.980000	0.982915
SyntheticControl	**1.000000**	0.993333	**1.000000**	0.996667	**1.000000**	**1.000000**	0.997900	0.713300	0.996667	0.999667	**1.000000**	**1.000000**
ToeSegmentation1	0.973700	**0.991228**	0.969298	0.969298	0.978070	**0.991228**	0.965300	0.916700	0.964912	0.968421	0.974000	0.982456
ToeSegmentation2	0.961500	0.930769	**0.976923**	0.953846	0.953846	**0.976923**	0.955300	0.938500	0.938462	0.923846	0.946000	0.938462
Trace	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**
TwoLeadECG	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	0.994600	0.989500	0.995610	0.999122	**1.000000**	**1.000000**
TwoPatterns	**1.000000**	0.996750	**1.000000**	**1.000000**	**1.000000**	**1.000000**	**1.000000**	0.515700	**1.000000**	**1.000000**	**1.000000**	**1.000000**
UWaveGestureL.All	0.968500	0.961195	0.856784	0.933277	0.939978	0.879118	0.968900	0.937500	0.951982	**0.975377**	0.960000	0.893076
UWaveGestureL.X	0.830800	0.843663	0.780849	0.814629	0.810999	0.808766	0.841100	0.707400	0.824958	**0.854746**	0.840000	0.805974
UWaveGestureL.Y	0.758500	0.765215	0.664992	0.716360	0.671413	0.678950	0.772300	0.751300	0.767169	0.773981	**0.781000**	0.698492
UWaveGestureL.Z	0.772500	**0.795924**	0.756002	0.761027	0.760469	0.762144	0.784400	0.728900	0.764098	0.791904	0.788000	0.738693
Wafer	**1.000000**	0.998378	0.998540	0.998215	0.998540	0.999027	0.999100	0.997700	0.998540	0.998232	**1.000000**	0.995620
Wine	0.889000	0.833333	0.851852	0.870370	0.870370	0.907407	0.890600	0.666700	0.611111	0.812963	0.870000	**0.962963**
WordSynonyms	0.779000	0.680251	0.661442	0.650470	0.636364	0.678683	**0.787400**	0.685000	0.733542	0.753448	0.766000	0.592476
ACSF1	–	0.900000	**0.960000**	0.910000	0.930000	0.170000	–	0.780000	0.920000	0.886000	–	0.930000
BME	–	0.993333	**1.000000**	**1.000000**	**1.000000**	**1.000000**	–	**1.000000**	0.993333	**1.000000**	–	**1.000000**
Chinatown	–	0.982609	0.985507	0.985507	0.985507	0.985507	–	0.724600	0.985423	0.982507	–	**0.988338**
Crop	–	0.744940	0.743869	0.742738	0.746012	0.740476	–	0.755900	**0.772202**	0.751345	–	0.725119
DodgerLoopDay	–	**0.637500**	0.537500	0.550000	0.462500	0.500000	–	0.487500	0.150000	0.572500	–	0.612500
DodgerLoopGame	–	0.898551	0.876812	0.891304	0.550725	**0.905797**	–	0.681200	0.855072	0.873188	–	**0.905797**
DodgerLoopWeekend	–	0.978261	0.963768	0.978261	0.949275	0.963768	–	0.942000	0.971014	0.974638	–	**0.985507**
GunPointAgeSpan	–	0.996835	0.996835	0.996835	**1.000000**	0.848101	–	0.987300	0.987342	0.996835	–	**1.000000**
GunPointMaleV.F.	–	**1.000000**	**1.000000**	**1.000000**	0.996835	0.996835	–	0.993700	0.993671	0.998418	–	**1.000000**
GunPointOldV.Y.	–	0.993651	**1.000000**	**1.000000**	**1.000000**	0.990476	–	0.981000	0.965079	0.991111	–	**1.000000**
InsectEPGRegularT.	–	0.995984	**1.000000**	**1.000000**	**1.000000**	**1.000000**	–	0.971900	**1.000000**	**1.000000**	–	**1.000000**
InsectEPGSmallT.	–	0.935743	0.955823	0.927711	0.971888	0.477912	–	0.943800	0.943775	**0.979116**	–	0.919679
MelbournePedestrian	–	0.913061	0.912245	0.911837	0.904898	0.901633	–	0.360400	**0.913899**	0.904387	–	0.889299
PowerCons	–	**0.994444**	0.933333	0.944444	0.927778	0.927778	–	0.938900	0.944444	0.940000	–	0.977778
Rock	–	**0.920000**	0.780000	**0.920000**	0.820000	0.760000	–	0.780000	0.800000	0.900000	–	0.780000
SemgHandG.Ch2	–	0.910000	0.866667	0.916667	0.848333	0.651667	–	0.786700	0.816667	**0.926833**	–	0.870000
SemgHandM.Ch2	–	0.560000	0.513333	0.504444	0.391111	0.468889	–	0.524400	0.482222	**0.645111**	–	0.533333
SemgHandS.Ch2	–	0.873333	0.746667	0.740000	0.666667	0.788889	–	0.664400	0.824444	**0.881111**	–	0.800000
SmoothSubspace	–	0.980000	**1.000000**	**1.000000**	0.993333	**1.000000**	–	0.993300	0.993333	0.978667	–	**1.000000**
UMD	–	0.986111	**1.000000**	**1.000000**	**1.000000**	**1.000000**	–	0.826400	0.986111	0.992361	–	**1.000000**

Table 5
Performance summary of 12 algorithms on 85 UCR2018 datasets.

Performance Metrics	Existing SOTA	Best:LSTM-FCN	Vanilla:ResNet-Trans	ResNet-Trans 1	ResNet-Trans 2	ResNet-Trans 3
Total	65	85	85	85	85	85
Win	5	13	2	0	2	0
Tie	13	10	21	16	15	19
Lose	47	62	62	69	68	66
Best	18	23	23	16	17	19
MeanAcc	0.9001	0.8932	0.8866	0.8833	0.8230	0.8077
Avg_rank	6.1588	5.1823	5.3176	5.9588	6.8294	6.5058

Performance Metrics	TS-CHIEF	ROCKET	InceptionTime	MACNN	ResNet-50 SC	FMLA
Total	65	85	85	65	85	85
Win	1	7	3	9	1	14
Tie	8	10	7	14	6	22
Lose	56	68	75	42	78	49
Best	9	17	10	23	7	36
MeanAcc	0.8679	0.8814	0.8653	0.8692	0.8249	0.8937
Avg_rank	8.1470	6.6941	7.1117	5.4461	8.2882	4.8176

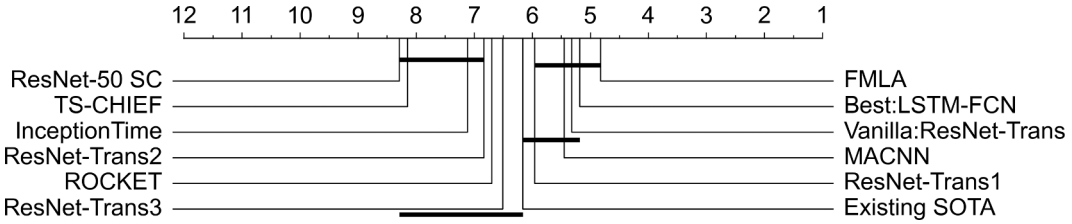


Fig. 6. Results of Avg_rank of 12 algorithms on 85 univariate datasets.

data loss in CLAwDCN-M does not degrade local features in most cases and the relation between local features, that is, the global trend, remains. Therefore, we consider that 50% is an appropriate frequency and can be used as an initial setting. On the other hand, CLAwDCN-M with a mask frequency of 90% took the second position, mainly because of its outstanding performance in a small number of datasets, like 'Symbols', 'Wine', and 'CinCECGTorso'.

As mentioned in SubSection 3.2, self-distillation is applied to the mask mechanism to stabilize the training process. We evaluated the effectiveness of self-distillation by comparing CLAwDCN-M and CLAwDCN-M-SD with respect to accuracy. We fed the input time series to CLAwDCN-M with random mask layers 3 times. The averaged output was used to teach the same model with regular mask layers, i.e., CLAwDCN-M-SD. The mean accuracy of CLAwDCN-M-SD was boosted to 0.873686 by self-distillation as shown in Table 1. According to Table 2, CLAwDCN-M-SD won CLAwDCN-M in 44 cases and lost in 21 cases. The mask mechanism can reduce the influence of noise, and in general, the masked positions do not harm the whole waveform of the time series. However, mask operations may appear at different positions in different iterations, which could enhance the model's robustness but makes it hard to train. The self-distillation technique used here is for accelerating the training process under randomness.

Finally, we evaluate the effectiveness of the online distillation by comparing CLAwDCN-M-SD and FMLA. FMLA overweighed CLAwDCN-M-SD in terms of 'Win'/'Tie'/'Lose' as shown in Table 2. With the mean accuracy reaching 0.893739, FMLA was the winner of 48 cases and the loser of 11 cases. Online distillation helps realize the forward–backward bidirectional circulation in our FMLA. Therefore, the locality awareness of attention in FMLA is further strengthened. This also supports the proposition in [38] that CNN-like architectures are good teachers for Transformer-like algorithms in knowledge distillation.

4.3. Performance evaluation

To evaluate the performance of FMLA, we compared it with 11 state-of-the-art TSC algorithms against 'Win'/'Tie'/'Lose', MeanAcc, and Avg_rank based on top-1 accuracy, with the experimental and statistical results on 85 UCR2018 datasets shown in Tables 4 and 5. All the models for comparison are listed below. We discuss their performance in the order they rank in the Avg_rank.

- Existing SOTA: a well-known benchmark consisting of the highest accuracy on each dataset obtained by STC [31], HC [22], gRSF [21] and mv-ARF [39].
- Best:LSTM-FCN [20]: a dual-network algorithm consisting of an attention-based LSTM branch and an FCN branch.
- Vanilla: ResNet-Transformer [18]: a dual-network algorithm consisting of a ResNet branch and a vanilla Transformer branch.
- ResNet-Transformer 1, 2, 3 [18]: three dual-network algorithms consisting of a ResNet branch and a Transformer branch with different numbers of building blocks.

Table 6
'Best' results achieved by 12 algorithms on short-medium and long datasets.

Dataset Type	Existing SOTA	Best:lstm-fcn	Vanilla:Transformer	ResNet-Trans1	ResNet-Trans2	ResNet-Trans3
short-medium	15	15	17	14	14	17
long	3	8	6	2	3	2

Dataset Type	TS-CHIEF	ROCKET	Inception	MACNN	ResNet50 SC	OUR
short-medium	8	10	9	19	7	29
long	1	7	1	4	0	7

- TS-CHIEF [33]: a tree classifier based on the Proximity Tree algorithm.
- ROCKET [7]: an ensemble algorithm consisting of a large number of random convolutional kernels.
- InceptionTime [12]: a CNN-based network adapted to TSC.
- MACNN [3]: an attention-based multi-scale CNN.
- ResNet-50 SC [42]: a ResNet-50 architecture with different classifiers on different datasets.
- FMLA: the proposed model in this paper.

As shown in Table 5, the top five algorithms are all based on the attention mechanism, which, to some extent, demonstrates the effectiveness of attention. The proposed FMLA outperformed all comparison algorithms; it obtained the best top-1 accuracy results on 36 out of the 85 datasets and won 22 cases. FMLA ranked first regarding the average ranking shown in Fig. 6 and we may consider this as the manifestation of the flexible perception of the model. Table 6 summarizes the performance of 12 algorithms on the short-medium and long time series datasets. It is clear that FMLA is the best algorithm for classifying short-medium time series. FMLA achieved the best results seven times on long time series and was the second best of all compared algorithms. The Avg_rank results show that FMLA is capable of capturing shapelets of different lengths based on the same structure. That is because FMLA wisely hybridizes Transformers and CNN-like algorithms, hence benefiting from the advantages of both. The performance of LSTM-FCN was slightly lower than that of our algorithm. The attention-enhanced LSTM, namely ALSTM, makes LSTM-FCN receive similar advantages as Transformer-based algorithms. However, its intrinsic disadvantage, namely being biased on the latest information, is still unavoidable. This drawback easily leads to the loss of information previously extracted in long time series, which weakens the TSC performance. As the feature fusion happens at the end of the model, it is just a classification based on the CNN and LSTM branches separately. However, such a model cannot exploit the advantages of CNNs and Transformers sufficiently. Moreover, FMLA outperformed the four Transformer-based models in [18], where 'Vanilla: ResNet-Transformer' achieved third place in the average ranking. These four models performed similarly in terms of 'Best' but achieved different mean accuracy values. This means they have similar feature extraction abilities but adapt to datasets with different sizes based on their Params. The four models fuse features in the same way as LSTM-FCN, namely, at the end of the feature extraction. The fourth-best algorithm in average ranking, MACNN, applies the attention block after each multi-scale CNN block to aggregate the multi-scale information obtained. The relative importance of the features extracted by the CNN blocks can be distinguished. In this way, more useful features contribute more to the classification result. The average rank of MACNN was slightly lower than the 'Vanilla:ResNet-Transformer' model but outperformed the other three Transformer-based models.

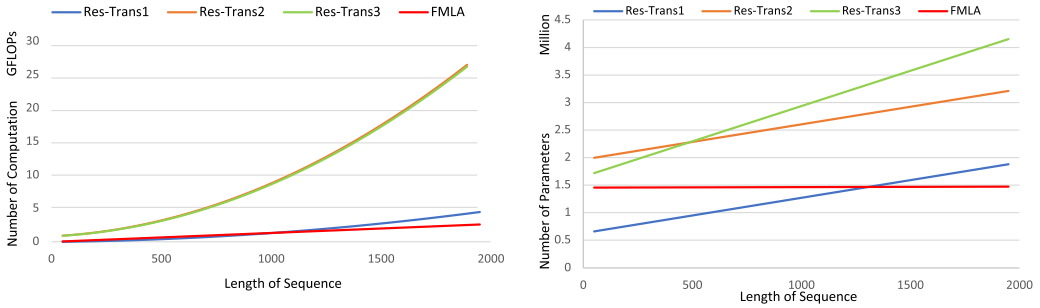
Both Rocket and InceptionTime use multiple kernels for multi-scale feature extraction. Nevertheless, both cannot well analyze the importance of each feature extracted and the shapelets of various lengths without the attention mechanism. 'TS-CHIEF' and 'ResNet-50 SC' are the worst two algorithms since they use regular methods to extract monotonous features. It is thus difficult for them to extract flexible shapelets in TSC data with noise.

To sum up, FMLA achieved the best performance in terms of 'Win'/'Tie'/'Lose', MeanAcc, and Avg_rank. On the one hand, FMLA inferences more efficiently than those of the vanilla attention-based and LSTM-based algorithms thanks to its higher parallelism and lower complexity. On the other hand, FMLA outperformed the other attention-irrelevant algorithms as it provides a better combination of local and global features through the integration of DCN and CLA blocks.

4.4. Parallelism and complexity analysis

LSTM-FCN [20] is one of the pioneering studies on TSC. It processes a sequence token by token to obtain global representations. However, the weak parallelism causes its training time to be highly dependent on the length of a given series and it needs more parameters to remember more previous information. To overcome these problems, attention-based models with better parallelism have been proposed, like ResNet-Transformer [18] and MACNN [3]. Attention treats information in sequences as query-key pairs, which leads to quadratic complexity. Assuming that d is the dimension of input vectors and n is the length of the input sequence, there are mainly four processes directly related to the complexity of the vanilla attention mechanism in [18,3], including the projection of queries, keys, and values with $O(3nd^2)$, dot-product calculation of similarity with $O(n^2d)$, SoftMax calculation of similarity with $O(n^2)$ and weighted sum calculation of values with $O(n^2d)$.

To reduce the complexity of vanilla attention, we compress the length of values based on the low-level features extracted by DCN blocks and the keys are generated from the compressed values. The projections of keys in the vanilla attention are omitted so the



(a) FLOPs (b) Params

Fig. 7. Complexity comparison in terms of FLOPs and Params.

complexity in this stage is reduced to $O(2nd^2)$. Otherwise, FMLA has a linear complexity of $O(Cnd)$, in the dot-product, SoftMax, and weighted sum calculation processes. The mask mechanism is a broadcast operation with a complexity of $O(nd)$ and the knowledge distillation techniques are non-parameter operations that do not directly influence the complexity. All in all, the complexity of our model, $O(n)$, is much lower than that of the vanilla Transformer-based model, $O(n^2)$, and our parallelism is better than that of the vanilla attention-based models.

The floating-point operations per second (FLOPs) and the number of parameters (Params) of the FMLA model were compared with those of three Transformer-based models, as shown in Fig. 7. ResNet-Transformer2 and ResNet-Transformer3 have the same number of blocks as ours, but their FLOPs increased quadratically. ResNet-Transformer1 has fewer FLOPs in the early stage because it only uses one Transformer block. However, as the length of the given sequence grows, the FLOPs demonstrate the advantages of FMLA in efficiency. The Params of all four algorithm grows linearly, and the growth mainly happens in the final classification layers. In FMLA, we squeeze the outputs of two branches and add to the results before classification without a specific classifier. Therefore, the number of parameters we used grows slower than the others with the growth of the sequence length.

5. Conclusion and future work

In this paper, we propose a global–local attention model, namely FMLA, for TSC. FMLA systematically hybridizes DCN, collaborative attention mechanism, online knowledge distillation, and mask mechanism. FMLA realizes global and local feature awareness, model redundancy reduction, and low noise interference. Extensive experiments are conducted on 85 UCR2018 datasets. FMLA achieved first place in the average ranking between 12 state-of-the-art algorithms. It won on 14 datasets and performed the best on 36 datasets including 29 short-medium and 7 long time series datasets in terms of 'Win'/'Tie'/'Lose' measure. FMLA achieves a mean accuracy of 89.37%, the second-best of all algorithms. Experimental results show that the proposed FMLA can extract abundant features with lower complexity for TSC compared with other state-of-the-art algorithms.

However, there are some limitations to FMLA. First, the adopted self-distillation averages outputs from several forward propagations. Parallelizing this process requires additional memory, however, the parameters in each propagation are the same. We are going to conduct more studies on memory resource reduction in the training process. Second, the linear classification layer can perform well on traditional single-network-based structures. However, it is not sufficient for dual-network architectures. It is necessary to design a specific classification layer for dual-network models in the future.

CRediT authorship contribution statement

Bowen Zhao: Methodology, Conceptualization, Writing - original draft. **Huanlai Xing:** Methodology, Writing - review & editing, Supervision. **Xinhan Wang:** Investigation, Writing - review & editing. **Fuhong Song:** Software, Validation. **Zhiwen Xiao:** Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work was supported in part by the Natural Science Foundation of Hebei Province (No. F2022105027), the Natural Science Foundation of Sichuan Province (No. 2022NSFSC0568), and the Fundamental Research Funds for the Central Universities, P. R. China.

References

[1] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling., 2018.doi: 10.48550/arXiv.1803.01271.
[2] Rongjun Chen, Xuanhui Yan, Shiping Wang, Guobao Xiao, Da-net: Dual-attention network for multivariate time series classification, Inf. Sci. 610 (2022) 472–487.
[3] Wei Chen, Ke Shi, Multi-scale attention convolutional neural network for time series classification, Neural Networks 136 (2021) 126–140, https://doi.org/10.1016/j.neunet.2021.01.001.
[4] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. Multi-head attention: Collaborate instead of concatenate. arXiv:2006.16362, 2021. doi: 10.48550/arXiv.2006.16362.
[5] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In Proceedings of the IEEE international Conference on Computer Vision, pages 764–773, 2017. doi: 10.1109/ICCV.2017.89.
[6] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The UCR time series archive. IEEE/CAA Journal of Automatica Sinica, 6(6), 1293–1305, 2019. doi: 10.1109/JAS.2019.1911747.
[7] Angus Dempster, François Petitjean, Geoffrey I. Webb, ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels, Data Min. Knowl. Disc. 34 (5) (2020) 1454–1495, https://doi.org/10.1007/s10618-020-00701-z.
[8] 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 16×16 words: Transformers for image recognition at scale. In International Conference on Learning Representations, 2020.
[9] Stéphane d'Ascoli, Hugo Touvron, Matthew L. Leavitt, Ari S. Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision Transformers with soft convolutional inductive biases. In International Conference on Machine Learning, pages 2286–2296. PMLR, 2021.
[10] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 6824–6835, 2021. doi: 10.1109/ICCV48922.2021.00675.
[11] Zhihao Fan, Yeyun Gong, Dayiheng Liu, Zhongyu Wei, Siyuan Wang, Jian Jiao, Nan Duan, Ruofei Zhang, and Xuan-Jing Huang. Mask attention networks: Rethinking and strengthen Transformer. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1692–1701, 2021. doi: 10.18653/v1/2021.naacl-main.135.
[12] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F. Schmidt, Jonathan Weber, Geoffrey I. Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. InceptionTime: Finding AlexNet for time series classification. Data Mining and Knowledge Discovery, 34(6), 1936–1962, 2020. doi: 10.1007/s10618-020-00710-y.
[13] Xiulin Geng, Xiaoyu He, Xu. Lingyu, Yu. Jie, Graph correlated attention recurrent neural network for multivariate time series forecasting, Inf. Sci. 606 (2022) 126–142.
[14] Qi Han, Zejia Fan, Qi Dai, Lei Sun, Ming-Ming Cheng, Jiaying Liu, and Jingdong Wang. On the connection between local attention and dynamic depth-wise convolution. In International Conference on Learning Representations, pages 1–25, 2021.
[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Identity mappings in deep residual networks, in: European Conference on Computer Vision, Springer, 2016, pp. 630–645, https://doi.org/10.1007/978-3-319-46493-0_38.
[16] Dan Hendrycks, Kevin Gimpel, Gaussian error linear units (GELUs) (2020), https://doi.org/10.48550/arXiv.1606.08415.
[17] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. arXiv:1503.02531, 2(7), 2015. doi: 10.48550/arXiv.1503.02531.
[18] Seth H. Huang, Xu Lingjie, and Jiang Congwei. Residual attention net for superior cross-domain time sequence modeling. arXiv:2001.04077, 2020. doi: 10.48550/arXiv.2001.04077.
[19] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 4163–4174, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.372.
[20] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. LSTM fully convolutional networks for time series classification. IEEE Access, 6: 1662–1669, 2018. doi: 10.1109/ACCESS.2017.2779939.
[21] Isak Karlsson, Panagiotis Papapetrou, Henrik Boström, Generalized random shapelet forests, Data Min. Knowl. Disc. 30 (5) (2016) 1053–1085, https://doi.org/10.1007/s10618-016-0473-y.
[22] James Large, Jason Lines, Anthony Bagnall, A probabilistic classifier ensemble weighting scheme based on cross-validated accuracy estimates, Data Min. Knowl. Disc. 33 (6) (2019) 1674–1709, https://doi.org/10.1007/s10618-019-00638-y.
[23] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of Transformer on time series forecasting. Adv. Neural Inform. Process. Syst., 32, 2019.
[24] Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. In The Tenth International Conference on Learning Representations, pages 1–20, 2022.
[25] Jason Lines, Sarah Taylor, and Anthony Bagnall. HIVE-COTE: The hierarchical vote collective of transformation-based ensembles for time series classification. In 2016 IEEE 16th International Conference on Data Mining (ICDM), pages 1041–1046, Barcelona, Spain, 2016. IEEE. doi: 10.1109/ICDM.2016.0133.
[26] Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. FastBERT: A self-distilling BERT with adaptive inference time. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 6035–6044, 2020. doi: 10.18653/v1/2020.acl-main.537.
[27] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 10012–10022, 2021. doi: 10.1109/ICCV48922.2021.00986.
[28] Lu. Jiachen, Jinghan Yao, Junge Zhang, Xiatian Zhu, Xu. Hang, Weiguo Gao, Xu. Chunjing, Tao Xiang, Li Zhang, SOFT: Softmax-free transformer with linear complexity, Adv. Neural Inform. Process. Syst. 34 (2021) 21297–21309.
[29] Benjamin Lucas, Ahmed Shifaz, Charlotte Pelletier, Lachlan O'Neill, Nayyar Zaidi, Bart Goethals, François Petitjean, Geoffrey I. Webb, Proximity Forest: An effective and scalable distance-based classifier for time series, Data Min. Knowl. Disc. 33 (3) (2019) 607–635, https://doi.org/10.1007/s10618-019-00617-3.
[30] Zhiliang Peng, Wei Huang, Shanzhi Gu, Lingxi Xie, Yaowei Wang, Jianbin Jiao, and Qixiang Ye. Conformer: Local features coupling global representations for visual recognition. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 367–376, 2021. doi: 10.1109/ICCV48922.2021.00042.
[31] Alejandro Pasos Ruiz, Michael Flynn, James Large, Matthew Middlehurst, and Anthony Bagnall. The great multivariate time series classification bake off: A review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery, 35(2), 401–449, 2021. doi: 10.1007/s10618-020-00727-3.
[32] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. arXiv: 1910.01108, 2019. doi: 10.48550/arXiv.1910.01108.
[33] Ahmed Shifaz, Charlotte Pelletier, Francois Petitjean, Geoffrey I. Webb, TS-CHIEF: A scalable and accurate forest algorithm for time series classification, Data Min. Knowl. Disc. 34 (3) (2020) 742–775, https://doi.org/10.1007/s10618-020-00679-8.

[34] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958, https://doi.org/10.5555/2627435.2670313.
[35] Yu. Siqi Sun, Zhe Gan Cheng, Jingjing Liu, Patient knowledge distillation for BERT model compression, in: Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, 2019, https://doi.org/10.18653/v1/D19-1441.
[36] Chang Wei Tan, Angus Dempster, Christoph Bergmeir, Geoffrey I Webb, Multirocket: Multiple pooling operators and Transformations for fast and effective time series classification, Data Min. Knowl. Disc. (2022) 1–24, https://doi.org/10.1007/s10618-022-00844-1.
[37] Yi Tay, Mostafa Dehghani, Dara Bahri, Donald Metzler, Efficient Transformers: A survey, ACM Computing Surveys (CSUR) (2022), https://doi.org/10.1145/3530811.
[38] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image Transformers & distillation through attention. In International Conference on Machine Learning, pages 10347–10357. PMLR, 2021.
[39] Kerem Sinan Tuncel, Mustafa Gokce Baydogan, Autoregressive forests for multivariate time series modeling, Pattern Recogn. 73 (2018) 202–215, https://doi.org/10.1007/s10618-016-0473-y.
[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in Neural Information Processing Systems, 30, 2017.
[41] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. arXiv:2006.04768, 2020. doi: 10.48550/arXiv.2006.04768.
[42] Marc Wenninger, Sebastian P. Bayerl, Jochen Schmidt, Korbinian Riedhammer, Timage – a robust time series classification pipeline, in: Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series, volume 11730, Springer International Publishing, Cham, 2019, pp. 450–461, https://doi.org/10.1007/978-3-030-30490-4_36.
[43] Wu. Bo, Xiao-Ping Zhang, Environmental sound classification via time–frequency attention and framewise self-attention-based deep neural networks, IEEE Internet Things J. 9 (5) (2021) 3416–3428, https://doi.org/10.1109/JIOT.2021.3098464.
[44] Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, Gao Huang, Vision Transformer with deformable attention, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 4794–4803.
[45] Zhiwen Xiao, Xu. Xin, Huanlai Xing, Shouxi Luo, Penglin Dai, Dawei Zhan, RTFN: A robust temporal feature network for time series classification, Inf. Sci. 571 (2021) 65–86, https://doi.org/10.1016/j.ins.2021.04.053.
[46] Nianyin Zeng, Wu. Peishu, Zidong Wang, Han Li, Weibo Liu, Xiaohui Liu, A small-sized object detection oriented multi-scale feature fusion approach with application to defect detection, IEEE Trans. Instrum. Meas. 71 (2022) 1–14, https://doi.org/10.1109/TIM.2022.3153997.
[47] Hang Zhang, Yeyun Gong, Yelong Shen, Weisheng Li, Jiancheng Lv, Nan Duan, and Weizhu Chen. Poolingformer: Long document modeling with pooling attention. In International Conference on Machine Learning, pages 12437–12446. PMLR, 2021.
[48] Linfeng Zhang, Chenglong Bao, and Kaisheng Ma. Self-distillation: Towards efficient and compact neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 4388–4403, 2021. doi: 10.1109/TPAMI.2021.3067100.
[49] Zhong Zheng, Zijun Zhang, Long Wang, Xiong Luo, Denoising temporal convolutional recurrent autoencoders for time series classification, Inf. Sci. 588 (2022) 159–173.
[50] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable Transformers for end-to-end object detection. In International Conference on Learning Representations, 2020.