Engineering Applications of Artificial Intelligence 123 (2023) 106296

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Time series classification based on convolutional network with a Gated Linear Units kernel

Chen Liu [a], Juntao Zhen [a], Wei Shan [b,c,*]

[a] Business School, University of Shanghai for Science and Technology, Shanghai 200093, China
[b] School of Economics and Management, Beihang University, Beijng 100191, China
[c] Key Laboratory of Complex System Analysis and Management Decision, Ministry of Education, Beijng 100191, China

ARTICLE INFO

Keywords:
Time series classification
Convolutional Gated Linear Units
Deep learning
Inception module

ABSTRACT

Time series data are ubiquitous in human society and nature, and classification is one of the most significant problems in the field of time series mining. Although it has been intensively studied, and has achieved significant results and successful applications, it is still a challenging problem, which requires capturing of multi-scale features of one-dimensional or multi-dimensional time series in variable length. In this paper, we propose a novel time series feature extraction block named Convolutional Gated Linear Units (CGLU), which is a combination of convolutional operations and Gated Linear Units for adaptively extracting local temporal features of time series. Combined with a temporal maxpooling block, it can extract global temporal features. To capture more diverse features, the Inception architecture is adopted to organize the CGLUs with different convolution kernel sizes, which result in the Convolutional GLU network. In order to evaluate the performance, we conduct extensive experiments on the UCR time series datasets (one-dimension) and UEA datasets (multi-dimension). Compared with baselines, our model obtains best results in terms of classification accuracy and training speed, which demonstrate effectiveness and efficiency of CGLUs and Conv-GLU network on time series classification tasks.

1. Introduction

Time series data is ubiquitous in a wide range of fields, including finance, environment and healthcare (Hu et al., 2016; Wang et al., 2017; Bagnall et al., 2017). Time series classification (TSC) is one of fundamental task in data analysis and machine learning, which involves predicting the class or category of a given time series (Yang and Wu, 2006; Esling and Agon, 2012). Its research and application has received attention in many fields. Examples of time series classification tasks include detecting types of vegetation by temporal optical signal from satellite (Rußwurm and Körner, 2020), identifying types of heartbeats in electrocardiogram data (Xie et al., 2021), classifying animal behaviors using data from an internet of things device (Arablouei et al., 2023), and so on.

The main challenge in time series classification is to capture the temporal dependencies and patterns in the series data. Unlike traditional classification problems, time series classification requires the model to considering the temporal order of data points in addition to their values, which make traditional machine learning algorithms such as support vector machines and decision trees are often ineffective. Therefore, dedicated methods are needed for time series classification tasks. Among them, conventional research on TSC mainly focus on distance based methods and feature based methods.

Distance based methods measure the similarity between two time series by computing a distance metric. Dynamic Time Warping (DTW) is a popular distance-based method (Berndt and Clifford, 1994; Rakthanmanon and Keogh, 2013), which find the optimal alignment between two time series under certain constraints and realize the matching between them through nonlinear warping. However, the original DTW does not consider the relative importance of the phase difference between a reference point and a testing point. As an effective improvement, Keogh and Pazzani (1999) proposed Derivative DTW which use first-order derivatives to convert original time series features into advanced features before using of classic DTW algorithm. Górecki and Łuczak (2013) further proposed a parameter approach to make a trade off between the DTW distance of original time series and distance of the time series under first derivative, and the research propose another Weighted DTW (Jeong et al., 2011), which penalizes points with higher phase difference between the reference point and the test point to prevent the minimum distance distortion caused by abnormal values. The *k*-nearest neighbor is a commonly used and effective classifier in distance-based TSC method (Hsu et al., 2011). The advantage of distance-based methods is that it can handle time series of different lengths, and they are good at capturing the local patterns in time series. Although these methods may offer certain benefits, they are also

* Corresponding author at: School of Economics and Management, Beihang University, Beijng 100191, China.
E-mail address: shanwei@buaa.edu.cn (W. Shan).

https://doi.org/10.1016/j.engappai.2023.106296
Received 20 December 2022; Received in revised form 3 April 2023; Accepted 4 April 2023
Available online 17 April 2023

associated with significant shortcomings. Specifically, they may not be capable of capturing global patterns, and they can be computationally expensive, especially when dealing with long time series.

Compare with distance based TSC methods, feature-based methods rely heavily on the features extraction, and usually consist of two steps of defining temporal features and training classifiers (Xing et al., 2010). Statistical features are commonly used for TSC. Take the work of Nanopoulos et al. (2001) as an example, they use features including mean and variance and take multilayer perceptron neural network as a classifier. This method only extracts the global features of time series while ignoring local features containing important classification information. To solve this problem, Geurts (2001) proposed a novel method which randomly extracts subsequences of time series data, and trains a threshold to judge the matching degree of the extracted subsequence with the original sequence. Then the subsequence with higher matching degree are taken as features, and a decision tree is used as classifier. This method is beneficial to improve the classification performance, but the computational complexity is increased significantly due to the extracting of local sequences. The disadvantage of feature-based methods is that choice of features can affect classification performance and it may require domain knowledge to select appropriate features. In addition, it is hard to capture all relevant information in the time series.

In recent years, deep learning models such as CNN (Heinrich et al., 2021; Cui et al., 2016), RNN (Kim et al., 2020), attention mechanisms (Zhao et al., 2023; Yang et al., 2022) and Transformers (Chen et al., 2022; Xu et al., 2022) have emerged as effective approaches for time series mining and classification. In these studies, one of the most influential and basic research is Wang et al. (2017) in which Fully Convolutional Networks (FCN) is proved to have a significant effective against the TSC task. A considerable part of research based on CNNs was affected by their study. Khan et al. (2021) propose to combine RNN and CNN for TSC, and build a model of Bidirectional Long Short-Term Memory with FCN (BiLSTM-FCN), in which attention mechanisms are also integrated. Recently, self-supervision learning has been proved to be able to effectively extract features, and has achieved success in many fields (Ericsson et al., 2022). Self-supervised learning is a type of machine learning in which the model is trained on the data without explicit labels. In the context of time series, the model can capture the underlying patterns in the data (Yue et al., 2022) and the classification can be treated as downstream tasks (Jawed et al., 2020; Xi et al., 2022). Besides these studies, deep learning methods are also combined with traditional feature based routine to focus on certain kinds of patterns, such as temporal features (Ji et al., 2022a) and shapelet features (Ji et al., 2022b). Deep learning based models can automatically extract features from the raw time series data and capture the temporal dependencies between data points. They have been successfully applied to a wide range of time series classification tasks and have achieved state-of-the-art performance on many benchmark datasets. Inspired by these studies, this paper focuses on making the model have the ability to extract local features and global features.

In this paper, we adapt Gated Linear Units (GLU), which is an effective feature extracting mechanism in sequential data (Zou et al., 2019; Dauphin et al., 2017), to a new kind of convolutional block to extract local feature in time series data called Convolutional Gated Linear Units (CGLU) block. In this block, we use GLU instead of the convolutional kernel of traditional convolutional neural network to perform convolutional computation on time series data. In order to further extract multi-scale features of time series, the architecture of Inception module (Szegedy et al., 2015) is adopted, which organizes CGLU blocks with different convolutional kernel sizes and stacks the outputs as extracted features for next layer. Besides, a time-wise max-pooling (TMaxPooling) is used to further aggregate features so that the model can capture the global characteristics of the time series. Extensive experiments on benchmark datasets show that our model can effectively capture time series features in multi-scale, and it has not only better classification performance, but also has a fast training speed and easy convergence. The contributions are summarized as follows:

- We present a novel convolutional module that uses GLU as the convolution kernel, instead of the conventional convolutional neural network. It can capture the local features of time series data effectively.
- To extract features and fuse them at different scales, we employ the Inception architecture, which combines multiple convolutional blocks with various kernel sizes in multiple layers.
- Experiments demonstrate that our model has outstanding generalization capabilities for time series classification on both one-dimensional and multi-dimensional time series data. Additionally, it has a low dependence on computing resources.

2. Related work

2.1. Fully convolutional network

The Fully Convolutional Network is a popular neural network architecture used for various computer vision tasks such as image segmentation, object detection, and image recognition (Long et al., 2015; Liu et al., 2022). Wang et al. (2017) proposed the FCN model for time series classification, which showed impressive performance. This end-to-end model does not require complicated data preprocessing and comprises three basic blocks, each consisting of a convolutional layer, a batch normalization layer and a global average pooling layer. The model's fundamental concept is that the neural network can learn nonlinear time series characteristics by applying one-dimensional convolution to the input time series data.

Moreover, the model was expanded into a residual network for time series classification, where the neural network learned the time series features for classification by skip connection of FCN modules. However, the residual network stacked multiple FCN modules, resulting in a chain-like model that could cause slow training speed and vanishing gradients (Zou et al., 2019). In this paper, we propose a new convolutional network that takes GLU (Dauphin et al., 2017) as the convolution kernel to learn the nonlinear time series characteristics by applying one-dimensional convolution to the time series data.

2.2. Gated linear units

The Long Short-Term Memory (LSTM) network has emerged as the dominant method for processing sequential data due to its ability to handle long-term dependencies (Zhou et al., 2022). However, since the next output of the LSTM relies on the previous hidden state (Szegedy et al., 2017), it is not conducive to parallel computing along sequence items. As a result, the model's training speed can be slow when dealing with long sequences and large datasets. To address this issue, Dauphin et al. (2017) proposed using a convolutional neural network to process extensive sequential data.

In this approach, a model consisting of stacked CNNs is utilized to extract higher-level and more abstract features from long text. Additionally, a new gating mechanism called GLU is proposed, which utilizes nondeterministic gates to alleviate the vanishing gradient problem by coupling linear elements to the gates. This technique preserves the power of nonlinear learning while enabling the gradient to be propagated in linear cells without scaling.

In this study, GLU is taken as a nonlinear transformation and introduced to handling of long-term dependencies and alleviate the vanishing gradient problem in time series classification.

2.3. Inception module

Most deep convolutional neural network models were built by stacking CNN layers to improve model performance. However, this simple solution has two significant disadvantages. Firstly, a neural network that is too deep is more prone to overfitting during training, especially

when the training data set is small. Secondly, the use of computational resources significantly increases during training.

To address these issues, the Inception module was developed to identify the optimal local sparse structure in a convolutional network (Szegedy et al., 2015), reducing the number of parameters and computing resources required. The inception module employs convolution kernel sizes of $1 \times 1, 3 \times 3, 5 \times 5$, and 3×3 pooling in the same layer to extract features with varying degrees of sparsity. Several versions of the Inception module are proposed, including Inception V2, Inception V3, Inception V4, and Inception-ResNet (Szegedy et al., 2016, 2017), with each version representing an iterative improvement over the previous one.

This paper adopts the structure of the Inception module to organize CGLU blocks with different kernel sizes to abstract features with multiple timespans of the time series.

3. Methodology

In this paper, we propose a convolutional-based neural network called the Conv-GLU network to solve time series classification problems. Traditional convolutional neural networks for classification problems generally consist of two phases: feature extraction from raw input data and classification using a fully connected layer that takes extracted features as input. Feature extraction is typically accomplished using multiple convolutional and pooling layers. The Conv-GLU network is focused on constructing new convolutional and pooling layers for feature extraction from raw time series data. Specifically, we introduce a new convolutional layer called the CGLU that uses GLU as the convolution kernel and performs convolution calculations on time series data. We also developed a temporal maxpooling (TMaxPooling) block that performs pooling operations in the time dimension after CGLU blocks. To extract both global and local features of time series in multi-scale manner, we group multiple blocks of CGLU and TMaxPooling with different kernel sizes according to the structure of the Inception module to construct the Conv-GLU network.

3.1. Convolution with gated linear units kernel

The CGLU block is different from the general convolutional layer in that it uses the GLU (Dauphin et al., 2017; Kim et al., 2020) as the convolution kernel. The GLU is a gating mechanism for sequence data that has shown to perform well in processing sequence data like text (Xie et al., 2019) and speech (Zeghidour et al., 2018). The GLU convolution kernel is defined as follows:

$$GLU(\mathbf{F}) = (\mathbf{F} \cdot \mathbf{W}_{linear} + \mathbf{b}_{linear}) \bigodot \sigma(\mathbf{F} \cdot \mathbf{W}_{gate} + \mathbf{b}_{gate}). \tag{1}$$

The convolution kernel is a linear projection represented by $\mathbf{F} \cdot \mathbf{W}_{linear} + \mathbf{b}_{linear}$, which is modulated by the gates $\sigma(\mathbf{F} \cdot \mathbf{W}_{gate} + \mathbf{b}_{gate})$ to controls the transmission of sequence data. Here, $\mathbf{F} \in \mathbb{R}^{M \times L}$ represents the input features. $\mathbf{W}_{linear}$, $\mathbf{W}_{gate} \in \mathbb{R}^{L \times N}$ are the learnable parameters to convert the dimension of the input time series data's dimension from M to N. During this process, information of different dimensions is integrated and extracted from the time series.$\mathbf{b}_{linear}$, $\mathbf{b}_{gate} \in \mathbb{R}^{N}$ are the learnable biases. $\bigodot$ represents the dot product and σ is the sigmoid function.

Taking GLU as kernel, convolution operation in the CGLU block is represented as:

$$CGLU(\mathbf{X}) = \mathbf{X} * GLU_{\{\mathbf{W}_{linear}, \mathbf{W}_{gate}, \mathbf{b}_{linear}, \mathbf{b}_{gate}\}} \tag{2}$$

where $*$ is the convolution operator and $GLU_{\{\mathbf{W}_{linear}, \mathbf{W}_{gate}, \mathbf{b}_{linear}, \mathbf{b}_{gate}\}}$ is the convolution kernel. Let $\mathbf{X}_{[i]}$ represent the ith column of $\mathbf{X}$ and $\mathbf{X}_{[i:j]}$ represent the slice from the ith column to the jth column of $\mathbf{X}$. Then, the detailed calculation of the CGLU block is:

$$CGLU(\mathbf{X})_{[t]} = \sum_{i=t \times s}^{t \times s + k - 1} GLU(\mathbf{X}_{[t \times s : t \times s + k - 1]}; \mathbf{W}_{linear}, \mathbf{W}_{gate}, \mathbf{b}_{linear}, \mathbf{b}_{gate})_{[i]} \tag{3}$$

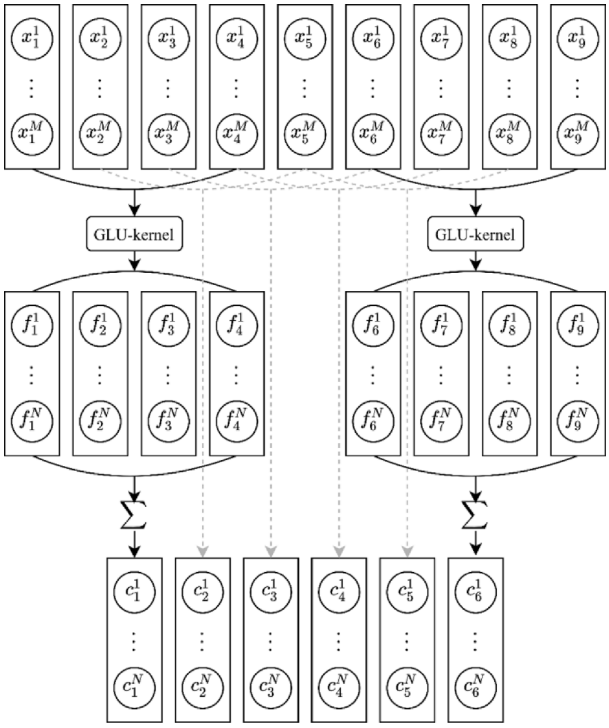

Fig. 1. The Convolutional GLU (CGLU) block.

where k is the size of the convolution kernel, and s is the convolutional stride. As shown in Fig. 1, the CGLU block takes a M-dimensional time series with length 9 as input. The kernel size is 4 and the convolutional stride is 1. The output of this CGLU block is a N-dimension sequence with length 6.

3.2. TMaxPooling block

TMaxPooling is essentially a one-dimensional convolution along the time dimension. The purpose of the TMaxPooling block is to reduce the feature size and the number of parameters and improve the calculation efficiency. More importantly, TMaxPooling introduces the invariance of the spatial displacement, which makes the training more stable.

The TMaxPooling block is calculated as:

$$TMaxPool(\mathbf{X}; s, w)_{[t]} = \max_{dim=1} (\mathbf{X}_{[t \times s \,:\, t \times s + w - 1]}) \tag{4}$$

where $\max_{dim=1}$ means taking the maximum value of a sliding window, which is the kernel, along the time dimension, w is the size of the pooling kernel, and s is the stride. Fig. 2 shows an example of TMaxPooling for N-dimensional time series data with length 6. When the stride is 3 and the kernel size is 3, the output is the N-dimension features with length 2.

3.3. Conv-GLU network

In deep learning models, the most direct approach to enhance convolutional neural networks is to stack multiple convolutional layers. This method is simple and effective. However, a neural network that is too deep can result in hard model training due to the abundance of parameters. The Inception module offers a solution to this issue by concatenating multiple convolutional blocks in parallel, rather than simply stacking them. This technique reduces the number of model parameters and overcomes the previously mentioned problems. In this paper, we propose the Conv-GLU network which consist of several

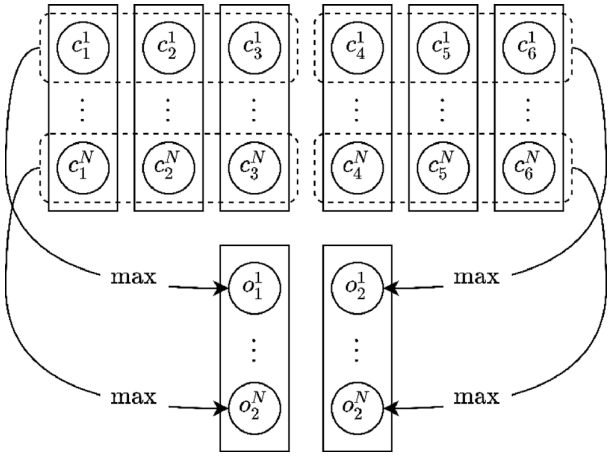

Fig. 2. The TMaxPooling block.

Conv-GLU blocks. The Conv-GLU network uses the structure of Inception module to organize multiple CGLU blocks and TMaxpooling blocks. Specifically, a Conv-GLU block consists of several parallel sub-blocks of CGLU and TMaxPooling. Each sub-block comprises one or two CGLU blocks stacked with a TMaxPooling block.

As shown in Fig. 3, the Conv-GLU network consists of one Conv-GLU block, which in turn includes four sub-blocks. The kernel size of the first layer of these sub-blocks are {8, 2, 14, 5}, respectively. For the last three sub-blocks, the kernel size of the second layer are {18, 6, 2}, respectively. Each sub-block is followed by a TMaxPooling block. Finally, the outputs of these four sub-blocks are concatenated and forwarded into fully connected layers. The last layer of Conv-GLU network is the softmax layer for classification. Additionally, dropout can be used to prevent overfitting.

4. Experiments

4.1. Datasets

In this section, we used two datasets to evaluate performance of the proposed model. The first dataset is the UCR dataset (Dau et al., 2019), which contains 44 different one-dimensional time series datasets from various fields, such as motion, sensors, electrocardiograms, stimulations, and spectra. The datasets were normalized, and all the data were split into training and test sets. The length of the time series ranged from 24 to 1882, the size of the training sets ranged from 16 to 1800, the size of the test sets ranged from 20 to 6164, and the number of categories ranged from 2 to 50. Table 1 shows the details of some of the datasets.

The second dataset used in this study is the UEA dataset (Bagnall et al., 2018), which contains 30 multidimensional time series datasets from different fields. The time series in the dataset range from 2 to 1345 dimensions, and all data are standardized and divided into training and test sets. Table 2 presents additional details on the dataset.

4.2. Experimental settings

The model is implemented using the TensorFlow framework. In this model, the time series data is inputted in an CGLU block with kernel size 2 followed by a TMaxpooling block. Then, it followed by two layers of Conv-GLU blocks. The first layer is composed of four parallel CGLU blocks, and the second layer is composed of three parallel-computed CGLU blocks. After conducting a series of experiments across various datasets, we obtain the hyperparameters that achieve bset model performance in most datasets. The optimal sizes of the GLU convolution kernels are {8, 2, 14, 5} and {18, 6, 2}, respectively. Each lastly GLU block foolwed by a TMaxpooling. The Adam optimizer is used in the model (Kingma and Ba, 2014), with $\alpha = 0.01$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon =$ 1e-8. The cross-entropy loss function is utilized in training the model. In the Conv-GLU network model, the parameters are initialized randomly, and these parameters are continually trained during the backpropagation process.

Table 1
Characteristics of time series datasets in the UCR datasets.

Datasets	Class	Length	Training count	Test count
Adiac	37	176	390	391
Beef	5	470	30	30
CBF	3	128	30	900
Coffee	2	286	28	28
CricketX	12	300	390	390
ECGFiveDays	2	136	23	861
FaceAll	14	131	560	1690
FaceFour	4	350	24	88
FacesUCR	14	131	200	2050
FiftyWords	50	270	450	455
Fish	7	463	175	175
GunPoint	2	150	50	150
OliveOil	4	570	30	30
OSULeaf	6	427	200	242
SwedishLeaf	15	128	500	625
Symbols	6	398	25	995
Trace	4	275	100	100
TwoLeadECG	2	82	23	1139
TwoPatterns	4	128	1000	4000
Wafer	2	152	1000	6164
WordSynonyms	25	270	267	638
Yoga	2	426	300	3000

Table 2
Characteristics of time series datasets in the UEA datasets.

Dataset	Classes	Training count	Test count	Dimensions
AtrialFibrillation	3	15	15	2
BasicMotions	4	40	40	6
Cricket	12	108	72	6
DuckDuckGeese	5	60	40	1345
EigenWorms	5	128	131	6
Epilepsy	4	137	138	3
ERing	6	30	30	4
FaceDetection	2	5890	3524	144
Handwriting	26	150	850	3
Heartbeat	2	204	205	61
JapaneseVowels	9	270	370	12
Libras	15	180	180	2
LSST	14	2459	2466	6
NATOPS	6	180	180	24
PenDigits	10	7494	3498	2
PEMS-SF	7	267	173	936
Phoneme	39	3315	3353	11
RacketSports	4	151	152	6
SelfRegulationSCP1	2	268	293	6
StandWalkJump	3	12	15	4

4.3. Evaluation metrics

In this paper, we adopt different evaluation measure for the two datasets. The Mean Per-Class Error (MPCE) is used to evaluate the model's classification performance for one-dimensional time series datasets. The Per-Class Error (PCE) of a model on the kth datasets is:

$$PCE_k = \frac{e_k}{c_k} \tag{5}$$

where c_k is the number of categories in the kth data set and e_k is the classification error. Then MPCE of a model on all datasets is:

$$MPCE = \frac{1}{D}\sum_{k=1}^{D} PCE_k \tag{6}$$

where D is the total number of datasets.

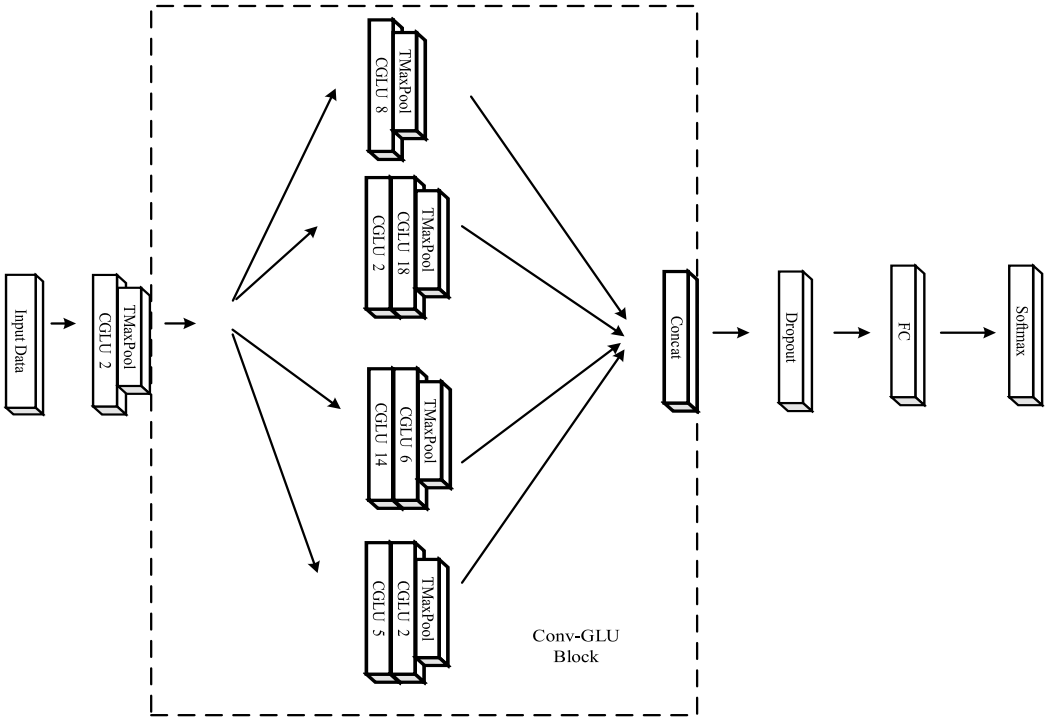

Fig. 3. The Structure of the Conv-GLU network.

For the multidimensional time series datasets, we used the accuracy to evaluate the classification performance of the models.

$$\mathrm{Acc} = \frac{1}{M}\sum_{i=1}^{M}\mathbb{I}\left(f(x)^{(i)} = y^{(i)}\right) \tag{7}$$

where M is the number of samples in a dataset, $\mathbb{I}(\cdot)$ is the indicator function, $f(x)^{(i)}$ represent the prediction of the ith sample, and $y^{(i)}$ is its ground truth.

4.4. Comparison with state-of-the-art methods

The comprehensive evaluation of our proposed model and other TSC models is presented in Tables 3 and 4. The Conv-GLU network model performs exceptionally well in all evaluation indicators, irrespective of whether on one-dimensional or multidimensional datasets. Furthermore, the FCN (Wang et al., 2017), ResNet (Wang et al., 2017), BOSS (Schäfer, 2015), and MCNN (Cui et al., 2016), also have strong competitiveness. Among them, the FCN is considered a benchmark method, and it can be extended to a ResNet structure for time series classification by stacking multiple FCN modules with skip connections. The MCNN addresses the limitation of feature extraction only on a single time scale by using different downsampling rates for the original time series data. The BOSS model extracts substructures from time series, performs noise reduction of the extracted substructures using Symbolic Fourier Approximation (SFA), and compares the differences between two time series using string matching algorithm. The BOSS VS model (Schäfer, 2016) combines the BOSS model and the vector space model for time series classification. This model represents time series as spatial vectors, uses tf-idf to count the frequency of occurrence of each category to obtain the tf-idf vector, and finally uses cosine similarity to compare the similarity of time series data.

Table 3
Comparing the classification error rate of 11 methods on 44 UCR datasets.

Dataset	DTW	MCNN	BOSSVS	BOSS	MLP	FCN	ResNet	Conv-GLU
Adiac	0.418	0.351	0.412	**0.220**	0.356	0.243	0.255	0.250
Beef	0.416	0.412	0.356	0.200	0.276	0.320	0.256	**0.006**
CBF	0.010	0.035	0.021	**0**	0.250	**0**	0.009	**0**
ChlorineCon	0.385	0.301	0.455	0.340	0.195	0.210	**0.180**	0.197
CinCECGTorso	0.459	0.078	0.090	0.125	0.160	0.195	0.320	**0.013**
Coffee	**0**	0.047	0.066	**0**	**0**	**0**	**0**	**0**
CricketX	0.356	**0.212**	0.456	0.259	0.391	0.215	0.200	0.31
CricketY	0.256	**0.191**	0.321	0.208	0.395	0.308	0.195	0.28
CricketZ	0.276	0.242	0.413	0.246	0.428	**0.197**	0.210	0.23
DiatomSizeR	0.103	0.053	0.066	0.046	0.044	0.090	0.080	**0.020**
ECGFiveDays	0.321	**0**	**0**	**0**	0.031	0.015	0.045	**0**
FaceAll	0.237	0.305	0.301	0.210	**0.095**	0.071	0.186	0.197
FaceFour	0.170	**0**	0.037	**0**	0.207	0.079	0.088	0.0682
FacesUCR	0.125	0.073	0.213	0.042	0.163	0.072	**0.032**	0.08
FiftyWords	0.273	**0.200**	0.380	0.301	0.310	0.331	0.363	0.244
Fish	0.177	0.087	0.031	**0.011**	0.156	0.030	**0.011**	0.020
GunPoint	0.133	**0**	**0**	**0**	0.079	**0**	0.015	0.020
Haptics	0.523	0.533	0.614	0.536	0.540	0.489	0.485	**0.470**

(*continued on next page*)

Table 3 (*continued*).

Dataset	DTW	MCNN	BOSSVS	BOSS	MLP	FCN	ResNet	Conv-GLU
InlineSkate	0.536	0.591	0.610	0.511	0.689	0.599	0.646	**0.491**
ItalyPower	0.050	0.045	0.098	0.053	0.056	0.052	0.051	**0.024**
Lightning2	**0.129**	0.186	0.312	0.148	0.306	0.200	0.257	0.146
Lightning7	0.264	0.221	0.293	0.342	0.410	0.156	0.187	**0.130**
Mallat	0.085	0.063	0.054	0.058	0.054	0.019	0.030	**0.020**
MedicalImages	0.303	**0.226**	0.435	0.288	0.310	0.208	0.281	0.284
MoteStrain	0.205	0.012	**0.123**	0.073	0.160	0.090	0.133	0.132
NonInvThorax1	0.210	0.066	0.212	0.161	0.068	**0.041**	0.055	0.059
NonInvThorax2	0.155	0.067	0.120	0.101	0.060	**0.045**	0.052	0.058
OliveOil	0.327	0.129	0.153	0.100	0.589	0.205	0.156	**0.100**
OSULeaf	0.649	0.286	0.075	**0.012**	0.503	**0.012**	0.030	0.346
SonyAIBORobot1	0.346	0.250	0.310	0.321	0.311	0.050	0.032	**0**
SonyAIBORobot2	0.179	0.060	0.208	0.098	0.159	0.035	0.027	**0**
StarLightCurves	**0.010**	0.120	0.126	0.021	0.039	0.035	0.033	0.026
SwedishLeaf	0.198	**0.011**	0.141	0.072	0.152	0.036	0.042	0.027
Symbols	0.050	0.038	**0.031**	0.032	0.174	0.037	0.178	0.104
SyntheticControl	0.007	0.002	0.040	0.030	0.005	0.030	**0**	**0**
Trace	**0**	**0**	**0**	**0**	0.179	**0**	**0**	**0**
TwoLeadECG	**0**	0.005	0.009	0.004	0.174	**0**	**0**	**0**
TwoPatterns	0.076	0.002	0.001	0.016	0.152	0.213	**0**	**0**
UWaveX	0.301	0.201	0.250	0.241	0.243	0.250	0.253	**0.198**
UWaveY	0.359	0.277	0.344	0.313	0.297	0.288	0.397	**0.271**
UWaveZ	0.342	**0.222**	0.326	0.312	0.285	0.280	0.255	0.269
Wafer	0.020	0.003	**0.001**	**0.001**	0.007	0.003	0.004	0.002
WordSynonyms	0.345	**0.199**	0.439	0.345	0.406	0.420	0.370	0.356
Yoga	0.214	0.143	0.171	**0.081**	0.1454	0.210	0.222	0.195
WIN	5	11	6	11	2	9	8	**20**
MPCE	0.045	0.027	0.037	0.025	0.044	0.025	0.026	**0.020**

Although these methods perform well in time series classification, as shown in Tables 3 and 4, the Conv-GLU network outperforms them in terms of the evaluation metrics, demonstrating the effectiveness of the model.

Table 3 shows the classification error rate of 8 models across 44 UCR data sets. The errors mean ranks and the MPCE score are shown in the last two rows. From the results, it can be observed that traditional time series classification methods, such as DTW, BOSSVS, BOSS has 5, 6 and 11 best results respectively. Deep learning based models such as MLP, FCN, ResNet has 2, 9 and 8 best results, respectively. In comparison, our Conv-GLU model has 20 best results, making it the most competitive among the other models. Besides, our proposed method yields a mean error rate of 0.020, which is highly competitive with other methods.

Table 4 illustrates the classification accuracy of 5 models on 23 UEA datasets. The accuracy mean ranks are shown in the last row. From the results, it is evident that DTW, FCN, ResNet and MLP models have obtained the optimal classification accuracy on 5, 4, 4 and 0 best results, respectively. In contrast, our proposed method achieves the best accuracy on 10 data sets out of 23, which is the largest number of best results among the other models.

When evaluating the performance of a neural network, it is important to consider not only the classification accuracy but also the convergence speed and training speed of the model. In this study, different neural network models are selected and compared to comprehensively evaluate their efficiency by training under exactly the same hardware.

To compare the training speeds of different models, we select the following deep learning models that demonstrated excellent performance on the evaluation indicators: FCN, ResNet, and Conv-GLU. We train models using the five datasets (Adiac, Beef, CricketX, TwoLeadECG, and Wafer) and compare their training speeds. As differences in the number of categories, training set size, and test set size, these datasets can fully reflect the model's performance in processing differently sized datasets. The batch size was set at 50 during training, and the time required to train an epoch was recorded. The results are presented in Fig. 4. The training speed of the model is significantly different across the five datasets. ResNet has a slower training speed, while the Conv-GLU model has the fastest training speed. This is because the ResNet achieves good performance by stacking, leading to a deep network structure that affects the training speed. In contrast, the Conv-GLU model can process with different scales in parallel to conduct training.

Table 4
Comparing the classification accuracy of 5 methods on 23 UEA datasets.

Dataset	DTW	FCN	ResNet	MLP	Conv-GLU
Articulary	**0.987**	0.953	0.853	0.35	0.9733
AtrialFibrillation	0.2	0.2667	0.20	0.40	**0.4667**
BasicMotions	0.975	1	1	0.275	**1**
CharacterTrajectories	**0.989**	0.974	0.993	0.61	0.966
Cricket	**1**	0.989	0.934	0.43	0.943
DuckDuckGeese	**0.6**	0.57	0.581	0.256	0.54
Epilepsy	0.964	**1**	0.971	0.263	0.978
ERing	0.133	0.46	**0.874**	0.185	0.40
FaceDetection	0.529	**0.66**	0.632	0.312	0.64
FingerMovements	0.53	0.60	0.58	0.53	**0.62**
HandMovement	0.231	0.2072	**0.4054**	0.283	0.3378
Handwriting	0.286	0.302	**0.9556**	0.186	0.4506
JapaneseVowels	0.949	0.9622	0.9865	0.253	**0.9911**
Libras	0.87	0.8611	**0.944**	0.325	0.9056
MotorImagery	0.50	0.510	0.510	0.258	**0.512**
NATOPS	0.883	0.938	0.88	0.276	0.916
PenDigits	**0.977**	0.938	0.975	0.352	0.972
Phoneme	0.151	0.22	0.201	0.097	**0.215**
RacketSports	0.803	**0.8421**	0.835	0.356	0.7763
SelfRegulationSCP1	0.775	0.62	0.756	0.432	**0.843**
SelfRegulationSCP2	0.539	0.50	0.554	0.138	**0.566**
StandWalkJump	0.2	0.33	0.33	0.120	**0.60**
UWaveGestureLibrary	0.903	0.55	0.837	0.53	**0.9125**
WIN	5	4	4	0	**10**

In addition, we compared the convergence speeds of the three neural network models (FCN, ResNet, and Conv-GLU) on five datasets of different dimensions (UWLibrary, AtrialFibril, Basic, SelfSCP1, and Japanese) from the UEA dataset. We recorded the number of epochs required for the models to converge on each dataset when they achieved optimal classification results. The results are shown in Fig. 5. The convergence of the three models differed depending on the dataset's dimensions. The difference in the convergence speeds of the different models is not significant for the UWLibrary and AtrialFibril datasets due to their low data dimension. However, for datasets with higher dimensions, such as Basic and Japanese, the Conv-GLU model always had the fastest convergence speed compared to the other models.

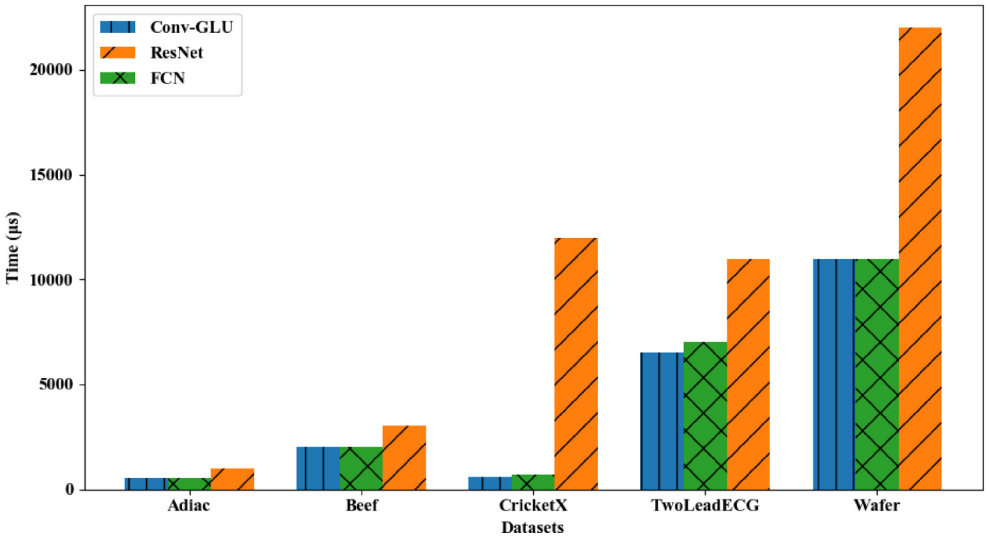

Fig. 4. Time required for different models to train one epoch on five datasets.

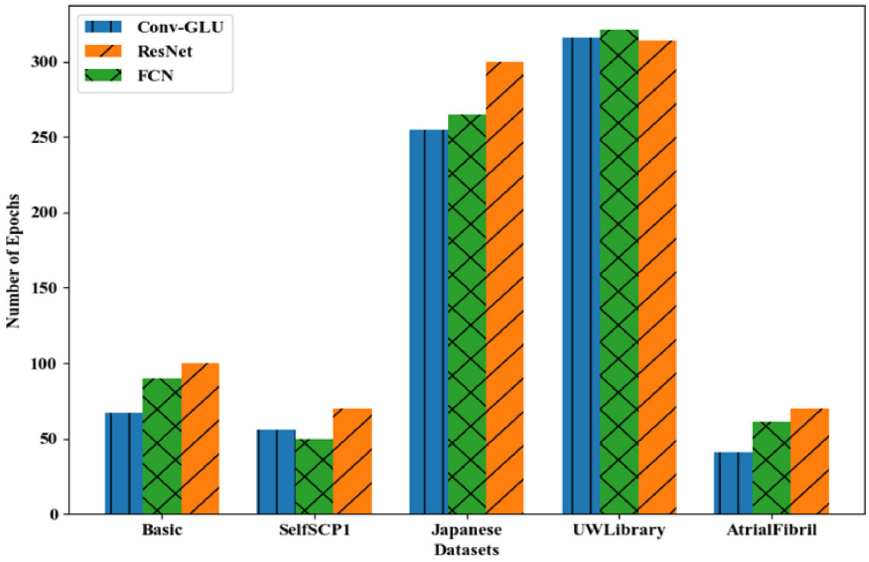

Fig. 5. Number of training epochs when different models converge.

To further analyze the convergence of the three models mentioned above, we specifically selected two datasets, namely, Japanese and SelfSCP2 from UEA, to demonstrate the changes in accuracy during the training process. We chose these two datasets because they are representative in terms of dimensions, sample size, and number of categories, and can comprehensively reflect the model's performance. The results are presented in Figs. 6 and 7, respectively.

Fig. 6 demonstrates that the Conv-GLU model achieves the highest classification accuracy when classifying the higher-dimensional Japanese dataset, outperforming other models. Moreover, the model converges at 350 epochs, which is a faster convergence rate compared to the ResNet model. Although the FCN model also converges at 350 epochs, its classification accuracy is slightly inferior to that of the Conv-GLU model. In addition, Fig. 7 shows that when classifying the lower-dimensional SelfSCP2 dataset, the Conv-GLU model achieves the highest classification accuracy compared to other models. Furthermore, the model converges during training at 250 epochs.

4.5. Sensitivity analysis

In this section, we present an analysis on several key hyperparameters in our approach.

(1) Learning rate

This study employ the Adm optimizer to train the models. Among parameters of the Adam optimizer, the learning rate plays a critical role in determining its effectiveness across various datasets. We vary the learning rate value within a reasonable range $\alpha \in [0.1, 0.05, 0.01, 0.005, 0.001]$, and investigate its influence on convergence speed and optimization performance of the loss function over 350 epochs on CinCECGTors and UWaveGestureLibrary datasets.

As shown in Figs. 8 and 9, when learning ratio is too small (0.005, 0.001), the convergence speed is too slow. However, when learning ratio is too large $(0.1, 0.05)$, the loss converges quickly, but the optimization performance is not optimal. On the contrary, the loss increases with

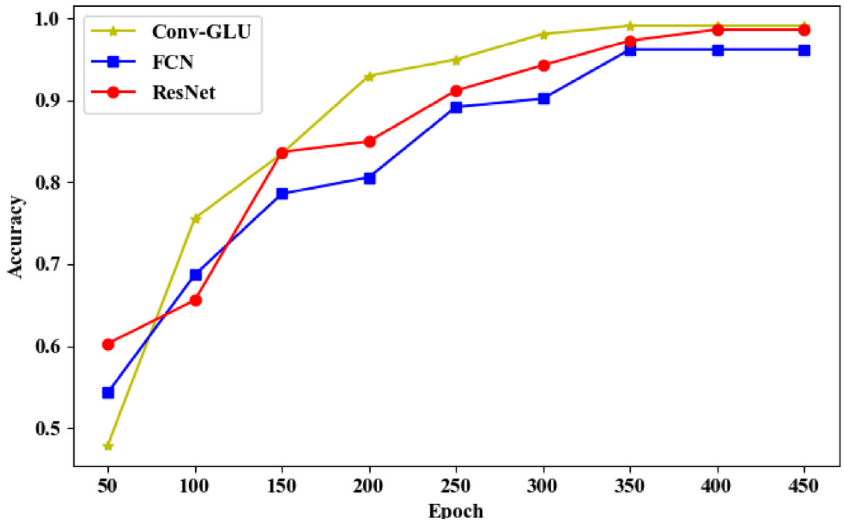

Fig. 6. Classification accuracies and convergence rates of the different models on the Japanese dataset.

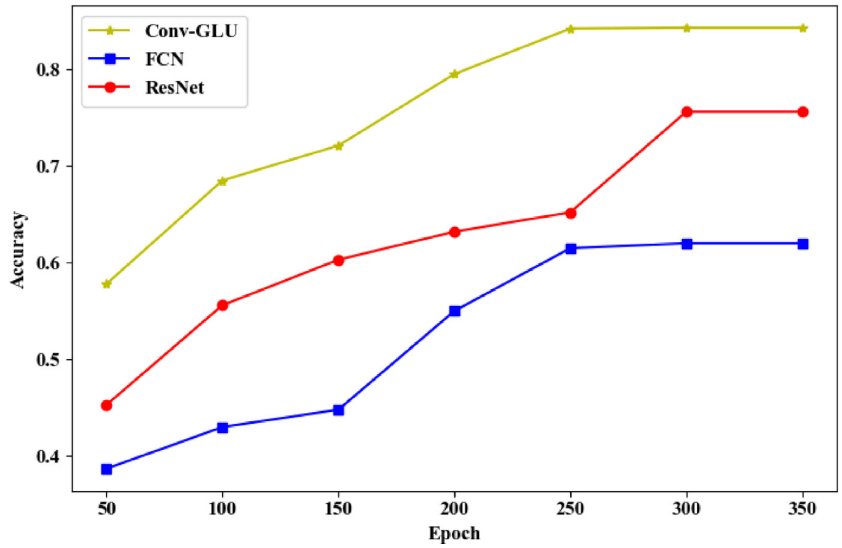

Fig. 7. Classification accuracies and convergence rates of different models on the SelfSCP2 dataset.

epochs. We find that when learning ratio $\alpha = 0.01$, the model converge quick enough and the optimization performance is optimal in the two datasets. By conducting a comprehensive analysis on other datasets, we set learning ratio hyperparameter to 0.01, which achieves the best trade-off between convergence speed and optimization performance on most datasets.

(2) Convolutional kernels

The size of the convolutional kernel is closely related to the model's ability to extract features from time series data. A smaller kernel has a smaller receptive field and focuses on local, subtle changes in the time series, while a larger kernel focuses on larger scale changes. In our experiments, we observed the following pattern: in the Conv-GLU network, the convolutional kernels in the same layer should have diverse sizes, including both smaller and larger kernels. To illustrate this observation, we showed the results of four sets of kernel sizes on the CinCECGTors and UWaveGestureLibrary datasets. Two sets have smaller values, and the size of the kernels in the first layer did not change much, with values of ({6,1,8,4},{6,2,1}) and ({6,2,8,4},{12,4,2}), respectively. Another set has larger values, but each layer lacked smaller kernels, with a size of ({10,6,16,8},{22,8,6}). The last set is the optimal result we obtained, with values of ({8,2,14,5}, {18,6,2}). In this set of parameters, the convolutional kernels in both layers had stronger diversity, which enabled the model to extract features from time series data at multiple scales. The results of these four sets of convolution kernels on the two datasets are shown in Fig. 10.

4.6. Local and global feature extraction capabilities

To test the model's ability to extract both local and global features, we selected four datasets from the UCR Dataset as shown in Fig. 11. Figs. 11(a) and 11(b) are bi-class TSC tasks, while Figs. 11(c) and 11(d) are multi-class TSC tasks with 4 and 8 categories, respectively. In Figs. 11(a) and 11(c), the global shapes of each time series are very similar, so the model is required to distinguish the differences in local details between different categories of sequences, i.e., local feature

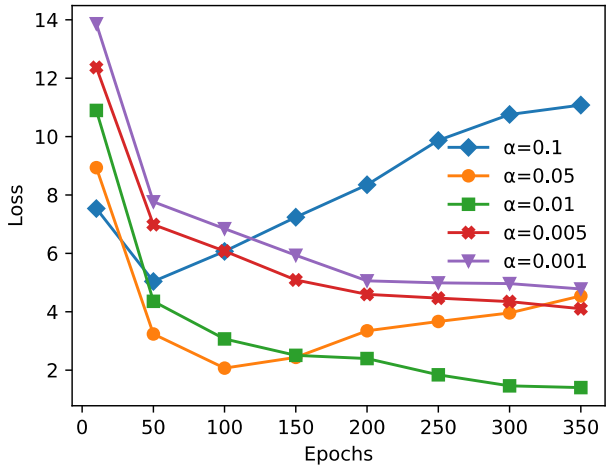

Fig. 8. Influence of learning ratio of Adm optimizer on CinCECGTorso dataset.

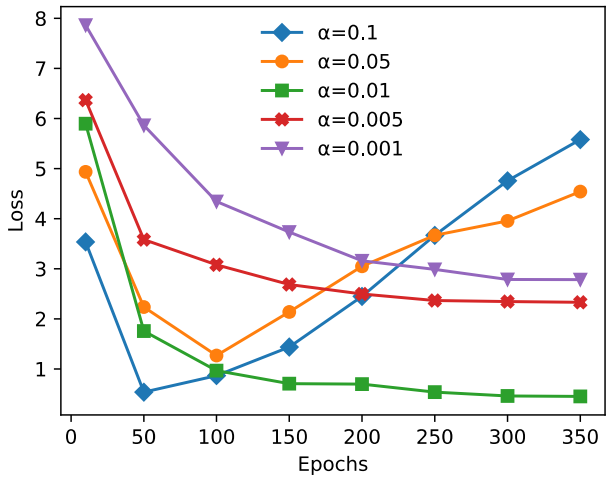

Fig. 9. Influence of learning ratio of Adm optimizer on UWaveGestureLibrary dataset.

extraction ability. In Figs. 11(b) and 11(d), the different categories of time series are more reflected in the global level differences, so the model is required to have good global feature extraction ability.

The classification error rates of our model and baselines on these datasets are shown in Table 3, and our model achieved the best results on all of them, indicating that the Conv-GLU network has good local and global feature extraction capabilities.

In classification problems, the larger the difference in the model's predicted results for different categories, the stronger its confidence in distinguishing between different category samples, which implying stronger generalization ability. This further suggests that the model has better feature extraction ability. Since the predicted results of classification problems are probability distributions over categories, we use the KL divergence of the average predicted results of the model on different categories to measure the difference in the model's ability. Specifically, the prediction discrepancy of the model between categories c_1 and c_2 is expressed as:

$$\mathrm{disc}\left(c_1, c_2\right) = D_{KL}\left(\overline{p}\left(c1\right) \parallel \overline{p}\left(c_2\right)\right) + D_{KL}\left(\overline{p}\left(c_2\right) \parallel \overline{p}\left(c1\right)\right) \tag{8}$$

where $D_{KL}\left(p \parallel q\right) = \sum_{i=1}^{N} p_i \log \frac{p_i}{q_i}$ is the KL divergence of probability distributions p and q, $\overline{p}\left(c_1\right) = \frac{1}{N_{c_1}} \sum_{i=1}^{N_{c_1}} p\left(x_i\right)$ is the average of the model's predictions for all samples of category c_1, $p\left(x_i\right)$ represents the

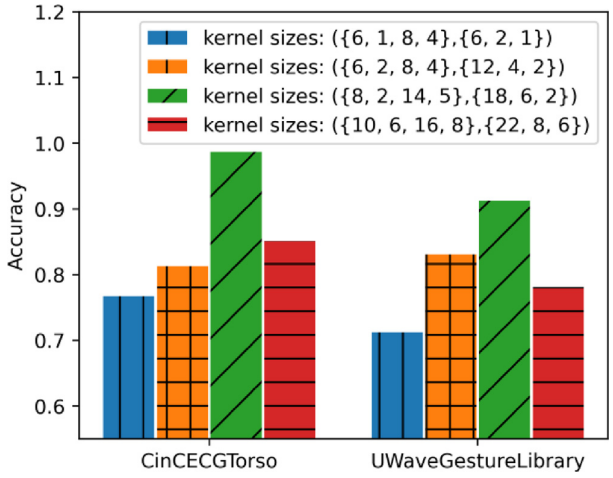

Fig. 10. Influence of Convolutional kernels on model accuracy on different dataset.

Table 5
Average prediction discrepancy between categories.

Dataset	Category number	MLP	ResNet	FCN	Conv-GLU
ECGFiveDays	2	1.7375	2.2656	2.5997	**3.3142**
SonyAIBORobot2	2	1.2579	1.5841	1.2978	**1.7492**
DiatomSizeR	4	4.2343	3.6450	3.3368	**6.9951**
UWaveY	8	2.2682	2.3265	2.4318	**2.5290**

prediction of the model for sample x_i, and N_{c_1} is the number of samples in category c_1. In the case of multiple classification:

$$\mathrm{disc}_{\mathrm{multi}} = \frac{1}{N_{c_i \neq c_j}} \sum_{c_i \neq c_j} \mathrm{dist}\left(c_i, c_j\right) \tag{9}$$

The experimental results are shown in Table 5. When making predictions on the four datasets, the Conv-GLU network is able to differentiate between different categories of samples as much as possible. Given that the four datasets have different numbers of categories and features, the experimental results demonstrate that our model has good local and global feature extraction capabilities for both two-class and multi-class problems.

5. Conclusions

We proposed the Conv-GLU network for time series classification tasks. Our model utilizes a special convolutional neural network with Gated Linear Units as the convolution kernel and uses the Inception module to organize the convolutional blocks with different kernel sizes. This ensures that the features learned by the model are more comprehensive and in a multi-scale manner. Additionally, the Inception architecture avoids overly deep models and improves the training efficiency. Our proposed model is an end-to-end solution that does not require extensive preprocessing of the original data. The model has strong generalization ability and can handle both one-dimensional and multidimensional time series data. We conducted comprehensive experiments on the UCR datasets and UEA datasets and compared our results with various time series classification methods. Results show that the Conv-GLU network model has excellent classification accuracy on the datasets. We have also chosen representative datasets from the UCR datasets and UEA datasets, trained the Conv-GLU network and other models on these datasets. We then conducted a comprehensive comparison of the models' convergence speeds and classification accuracies. The experimental results demonstrate the feasibility and effectiveness of employing the Gated Linear Units as a convolution kernel for time series classification.

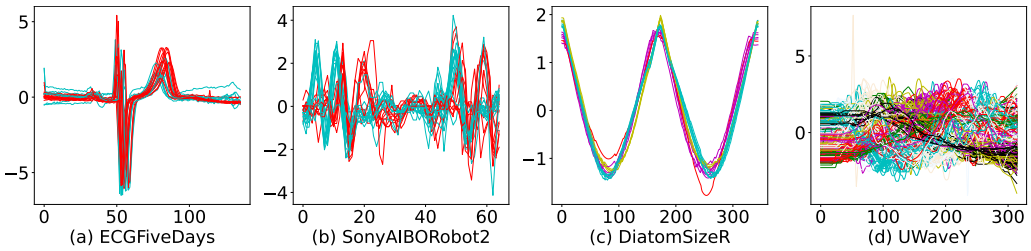

Fig. 11. Times series datasets with local feature discrepancy (a and c) and global feature discrepancy (b and d).

In future research, we will explore the integration of the Conv-GLU approach with the Transformer and attention mechanisms to enhance the multi-scale feature extraction capability of time series. Additionally, Conv-GLU can be combined with self-supervised learning for efficient time series representation learning. Finally, the application of this approach to specific problems is also an important direction to pursue.

CRediT authorship contribution statement

Chen Liu: Formal analysis, Validation, Data curation, Visualization, Writing – original draft, Supervision. **Juntao Zhen:** Conceptualization, Methodology, Formal analysis, Investigation, Writing – review & editing. **Wei Shan:** Methodology, Investigation, Resources, Supervision, Funding acquisition.

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.

Acknowledgments

The authors would like to thank the editor and anonymous reviewers for their valuable comments and helpful suggestions. Furthermore, this work was supported by the National Natural Science Foundation of China (No. 71971008), the Beijing Natural Science Foundation, China (No. 9222020), the Shanghai Philosophy and Social Science Planning Project, China (No. 2021BTQ003), the National Social Science Foundation of China (No. 22CGL050), and the Teaching Reform Project of Beihang University, China.

References

Arablouei, R., Wang, L., Currie, L., Yates, J., Alvarenga, F.A., Bishop-Hurley, G.J., 2023. Animal behavior classification via deep learning on embedded systems. Comput. Electron. Agric. 207, 107707.

Bagnall, A., Dau, H.A., Lines, J., Flynn, M., Large, J., Bostrom, A., Keogh, E., 2018. The UEA multivariate time series classification archive. arXiv preprint, arXiv:1811.00075.

Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E., 2017. The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances. Data Min. Knowl. Discov. 31 (3), 606–660.

Berndt, D.J., Clifford, J., 1994. Using dynamic time warping to find patterns in time series. In: KDD Workshop, Vol. 10, no. 16. pp. 359–370.

Chen, R., Yan, X., Wang, S., Xiao, G., 2022. DA-Net: Dual-attention network for multivariate time series classification. Inform. Sci. 610, 472–487.

Cui, Z., Chen, W., Chen, Y., 2016. Multi-scale convolutional neural networks for time series classification. arXiv preprint arXiv:1603.06995.

Dau, H.A., Bagnall, A., Kamgar, K., Yeh, C.C.M., Zhu, Y., Gharghabi, S., Keogh, E., 2019. The UCR time series archive. IEEE/CAA J. Autom. Sin. 6 (6), 1293–1305.

Dauphin, Y.N., Fan, A., Auli, M., Grangier, D., 2017. Language modeling with gated convolutional networks. In: International Conference on Machine Learning. PMLR, pp. 933–941.

Ericsson, L., Gouk, H., Loy, C.C., Hospedales, T.M., 2022. Self-supervised representation learning: Introduction, advances, and challenges. IEEE Signal Process. Mag. 39 (3), 42–62.

Esling, P., Agon, C., 2012. Time-series data mining. ACM Comput. Surv. 45 (1), 1–34.

Geurts, P., 2001. Pattern extraction for time series classification. In: European Conference on Principles of Data Mining and Knowledge Discovery. Springer, Berlin, Heidelberg.

Górecki, T., Łuczak, M., 2013. Using derivatives in time series classification. Data Min. Knowl. Discov. 26 (2), 310–331.

Heinrich, K., Zschech, P., Janiesch, C., Bonin, M., 2021. Process data properties matter: Introducing gated convolutional neural networks (GCNN) and key–value-predict attention networks (KVP) for next event prediction with deep learning. Decis. Support Syst. 143, 113494.

Hsu, H.H., Yang, A.C., Lu, M.D., 2011. KNN-DTW based missing value imputation for microarray time series data. J. Comput. 6 (3), 418–425.

Hu, B., Chen, Y., Keogh, E., 2016. Classification of streaming time series under more realistic assumptions. Data Min. Knowl. Discov. 30 (2), 403–437.

Jawed, S., Grabocka, J., Schmidt-Thieme, L., 2020. Self-supervised learning for semi-supervised time series classification. In: Advances in Knowledge Discovery and Data Mining: 24th Pacific-Asia Conference, Vol. 12084. PAKDD 2020, pp. 499–511.

Jeong, Y.S., Jeong, M.K., Omitaomu, O.A., 2011. Weighted dynamic time warping for time series classification. Pattern Recognit. 44 (9), 2231–2240.

Ji, C., Du, M., Hu, Y., Liu, S., Pan, L., Zheng, X., 2022a. Time series classification based on temporal features. Appl. Soft Comput. 128, 109494.

Ji, C., Hu, Y., Liu, S., Pan, L., Li, B., Zheng, X., 2022b. Fully convolutional networks with shapelet features for time series classification. Inform. Sci. 612, 835–847.

Keogh, E.J., Pazzani, M.J., 1999. Scaling up dynamic time warping to massive datasets. In: Principles of Data Mining and Knowledge Discovery: Third European Conference. PKDD'99, pp. 1–11.

Khan, M., Wang, H., Riaz, A., Elfatyany, A., Karim, S., 2021. Bidirectional LSTM-RNN-based hybrid deep learning frameworks for univariate time series classification. J. Supercomput. 77 (7), 7021–7045.

Kim, B., Park, J., Suh, J., 2020. Transparency and accountability in AI decision support: Explaining and visualizing convolutional neural networks for text information. Decis. Support Syst. 134, 113302.

Kingma, D.P, Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Liu, C., Cao, T., Zhou, L., 2022. Learning to rank complex network node based on the self-supervised graph convolution model. Knowl.-Based Syst. 251, 109220.

Long, J., Shelhamer, E., Darrell, T., 2015. Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3431–3440.

Nanopoulos, A., Alcock, Yannis, M., 2001. Feature-based classification of time-series data. Int. J. Comput. Res. 10 (3), 49–61.

Rakthanmanon, T., Keogh, E.J., 2013. Data mining a trillion time series subsequences under dynamic time warping. In: Twenty-Third International Joint Conference on Artificial Intelligence. pp. 3047–3051.

Rußwurm, M., Körner, M., 2020. Self-attention for raw optical satellite time series classification. ISPRS J. Photogramm. Remote Sens. 169, 421–435.

Schäfer, P., 2015. The BOSS is concerned with time series classification in the presence of noise. Data Min. Knowl. Discov. 29 (6), 1505–1530.

Schäfer, P., 2016. Scalable time series classification. Data Min. Knowl. Discov. 30 (5), 1273–1298.

Szegedy, C., Ioffe, S., Vanhoucke, S., 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In: Thirty-First AAAI Conference on Artificial Intelligence, Vol. 31, no. 1. pp. 4278–4284.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Rabinovich, A., 2015. Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2818–2826.

Wang, Z., Yan, W., Oates, T., 2017. Time series classification from scratch with deep neural networks: A strong baseline. In: International Joint Conference on Neural Networks. IJCNN, pp. 1578–1585.

Xi, L., Yun, Z., Liu, H., Wang, R., Huang, X., Fan, H., 2022. Semi-supervised time series classification model with self-supervised learning. Eng. Appl. Artif. Intell. 116, 105331.

Xie, H., Fang, S., Zha, Z.J., Yang, Y., Li, Y., Zhang, Y., 2019. Convolutional attention networks for scene text recognition. ACM Trans. Multimed. Compu. Commun. Appl. (TOMM) 15 (1), 1–17.

Xie, X., Liu, H., Shu, M., Zhu, Q., Huang, A., Kong, X., Wang, Y., 2021. A multi-stage denoising framework for ambulatory ECG signal based on domain knowledge and motion artifact detection. Future Gener. Comput. Syst. 116, 103–116.

Xing, Z., Pei, J., Keogh, E., 2010. A brief survey on sequence classification. ACM Sigkdd Explor. Newslett. 12 (1), 40–48.

Xu, J., Wu, H., Wang, J., Long, M., 2022. Anomaly transformer: Time series anomaly detection with association discrepancy. In: International Conference on Learning Representations. arXiv preprint arXiv:2110.02642.

Yang, C., Wang, X., Yao, L., Long, G., Jiang, J., Xu, G., 2022. Attentional gated Res2Net for multivariate time series classification. Neural Process. Lett. 1–25.

Yang, Q., Wu, X., 2006. 10 Challenging problems in data mining research. Int. J. Inf. Technol. Decis. Mak. 05 (04), 597–604.

Yue, Z., Wang, Y., Duan, J., Yang, T., Huang, C., Tong, Y., Xu, B., 2022. Ts2vec: Towards universal representation of time series. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 36, no. 8. pp. 8980–8987.

Zeghidour, N., Xu, Q., Liptchinsky, V., Usunier, N., Synnaeve, G., Collobert, 2018. Fully convolutional speech recognition. arXiv preprint arXiv:1812.06864.

Zhao, B., Xing, H., Wang, X., Song, F., Xiao, Z., 2023. Rethinking attention mechanism in time series classification. Inform. Sci. 627, 97–114.

Zhou, L., Zhang, Z., Zhao, L., Yang, P., 2022. Attention-based BiLSTM models for personality recognition from user-generated content. Inform. Sci. 596, 460–471.

Zou, X., Wang, Z., Li, Q., Sheng, W., 2019. Integration of residual network and convolutional neural network along with various activation functions and global pooling for time series classification. Neurocomputing 367, 39–45.