Attentional Gated Res2Net for Multivariate Time Series Classification

Chao Yang[1*], Xianzhi Wang[1], Lina Yao[2], Guodong Long[3], Jing Jiang[3] and Guandong Xu[4]

[1*]School of Computer Science, University of Technology Sydney, Sydney, NSW 2007, Australia.
[2]School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia.
[3]Australian Artificial Intelligence Institute, University of Technology Sydney, Sydney, NSW 2007, Australia.
[4]Data Science Institute, University of Technology Sydney, Sydney, 2007, NSW, Australia.

*Corresponding author(s). E-mail(s): chao.yang@student.uts.edu.au;
Contributing authors: xianzhi.wang@uts.edu.au; lina.yao@unsw.edu.au; guodong.long@uts.edu.au; jing.jiang@uts.edu.au; guandong.xu@uts.edu.au;

Abstract

Multivariate time series classification is a critical problem in data mining with broad applications. It requires harnessing the inter-relationship of multiple variables and various ranges of temporal dependencies to assign the correct classification label of the time series. Multivariate time series may come from a wide range of sources and be used in various scenarios, bringing the classifier challenge of temporal representation learning. We propose a novel convolutional neural network architecture called Attentional Gated Res2Net for multivariate time series classification. Our model uses hierarchical residual-like connections to achieve multi-scale receptive fields and capture multi-granular temporal information. The gating mechanism enables the model to consider the relations between the feature maps extracted by receptive fields of multiple sizes for information fusion. Further, we propose two types of attention modules, channel-wise attention and block-wise attention, to better leverage the

multi-granular temporal patterns. Our experimental results on 14 benchmark multivariate time-series datasets show that our model outperforms several baselines and state-of-the-art methods by a large margin. Our model outperforms the SOTA by a large margin, the classification accuracy of our model is 10.16% better than the SOTA model. Besides, we demonstrate that our model improves the performance of existing models when used as a plugin. Further, based on our experiments and analysis, we provide practical advice on applying our model to a new problem.

Keywords: multivariate time series classification, convolutional neural network, attention module, gating mechanism

1 Introduction

Time series data grant a great potential for various prediction tasks [1], and time series classification is one of the most challenging tasks in data mining [2]. A typical time series classification task involves multiple variables, represented by multiple data streams each corresponding to a variable. This is known as *multivariate time series classification (MTSC)*—given a group of time-aligned segments of these data streams, the task is to assign the correct classification label to it. MTSC has demonstrated significance in various applications, such as activity recognition [3], disease diagnosis [4], and automatic device classification [5], etc. Multivariate time series contain the temporal information from different sources, hence, measuring the interaction of sources and learning the temporal representations are the keys to realizing accurate MTSC [6]. Different tasks have different requirements for the classifier, making building a generalized used classifier a challenge. For example, EEG signal based MTSC can be focused on many different goals such as the recognition of emotion [7, 8], decoding cognitive skills [9], recognition, investigation of sustained attention, detection of sleep disorder, decoding of cognitive tasks in brain-computer-interface, etc. In EEG classification, the performance is sensitive to many parameters together such as the number of recording channels, i.e., feature dimension, recording time length, i.e., the number of features, number of the individuals in each group, feature extraction method and, classifier's architecture.

Traditional methods for time series classification include distance-based models (e.g., k-nearest neighbors) and feature-based models (e.g., random forest [10] and support vector machine [11]). These models highly rely on manually-defined features, which are heuristic and task-dependent [12]. Also, it takes the expertise and considerable time of domain experts to design such features. Furthermore, conventional machine learning (ML) techniques have limitations in processing high-dimension data and representing complicated functions efficiently [13].

Recently, deep learning (DL) has gained popularity in computer vision, natural language processing, and data mining, thanks to its advantages in capturing complicated, nonlinear relations from massive data [14]. Deep neural networks usually stack multiple neural layers for automatic feature extraction and representation learning [15]. Many neural network architectures, such as Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), Transformer [16], Long Short-Term Memory (LSTM) [17], and Gated Recurrent Unit (GRU) [18], have been applied for time series analysis. In particular, RNN sends the prior output to the next input layer to facilitate temporal feature extraction; therefore, it takes a long training time and cannot support parallel computation. CNN can extract temporal feature and be parallelized during training to fully exploit the power of Graphics Processing Units (GPUs); however, it faces challenges in capturing long-range temporal dependencies and is, therefore, less used for time series classification. Transformer [16] has recently emerged as promising solution to multivariate time series classification. While transformer supports both parallel computing and efficient temporal feature extraction, it requires massive parameters for the multiple fully connected layers, making the training extremely time-consuming. Furthermore, transformer suffers overfitting on small datasets [19], and faces challenges in capturing short-range temporal information [20]. Besides, existing solutions to MTSC commonly require careful adjustments of architectures and parameters to deal with time series of various lengths. This is a critical yet little studied issue in existing time series classification research.

To summarize, ML methods are expertise-dependent and difficult for representing complicated non-linear functions. Among the DL methods, CNN is efficient for training and inferencing but challenging for capturing long dependencies; RNN can effectively learn the temporal representations of long temporal features, but is computationally expensive; transformer contains too many parameters, making it easy to prone to overfitting on small size datasets. We aim for accurate MTSC that can adapt to time series of various lengths to address the above deficiencies of existing studies. To this end, we propose a novel CNN architecture called Attentional Gated Res2Net (AGRes2Net) for MTSC. Our model can overcome the shortcoming of the standard CNN architecture by enabling the extraction of both global and local temporal features. It also has the capability to leverage multi-granular feature maps through channel-wise and block-wise attention mechanisms. In a nutshell, we make the following contributions in this paper:

- We propose a novel AGRes2Net architecture for accurate MTSC. Our model can capture dependencies over various ranges and exploit the inter-variable relations to achieve high performance on time series of various lengths, making it feasible for various tasks.
- We propose two attention mechanisms, namely channel-wise attention and block-wise attention, to leverage multi-granular temporal information for tasks with different data characteristics. The former has advantages on

datasets with many variables, while the latter can effectively prevent overfitting on datasets with very few variables.
- We conducted extensive experiments on 14 benchmark datasets to evaluate the model. A comparison with several baselines and state-of-the-art methods shows the superior performance of our model. Besides, plugging our model into MLSTM-FCN, a state-of-the-art CNN-RNN parallel model, demonstrates the model's capability to improve existing models' performance.

The remainder of the paper is organized as follows. Section 2 overviews the related work; Section 3 presents the proposed model and attention mechanisms; Section 4 reports our experiments and results; and finally, Section 5 gives the concluding remarks.

2 Related Work

2.1 Multivariate Time Series Classification

MTSC has been a longstanding problem and solved by traditional statistic and ML methods [21–23]. A representative example is k-Nearest Neighbors (KNN), which is proven outstanding in MTSC [24]. Its combination with Dynamic Time Warping (DTW) can achieve even better performance [25, 26]. DL methods are increasingly applied to MTSC, given their capability in automatic feature extraction and learning complex relations from massive amounts of data [27–29]. Commonly used DL architectures include Recurrent Neural Networks (RNNs), Gated Recurrent Unit (GRU) [18], Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) [17], and Transformer [16]. And recent studies heavily rely on CNNs to overcome the efficiency and scalability issues with recurrent models (e.g., RNN, LSTM, and GRU) [30–32].

Traditionally, CNNs are used for computer vision tasks, such as image recognition [33], object detection [34–36], and semantic segmentation [37]. Recent studies [38–42] show 1D-CNN is promising for temporal feature extraction—the convolution computation can capture potential temporal patterns while the information fusion across channels can cope with the inter-relations among variables. Further, Inception [43] uses multiple parallel convolutional kernels of different sizes to address the challenged faced by CNNs in capturing long-range temporal dependencies [44, 45]. However, Inception's receptive field has a restricted width, which limits its ability to capture long-range dependencies.

The combination of CNN and RNN represents an effort to exploit the advantages of both [46]. Hybrid CNN-RNN architectures generally follow a parallel or cascade style to facilitate temporal feature extraction in various ranges. For example, LSTM-FCN [47] uses CNN and RNN in parallel and achieves state-of-the-art performance on several benchmark datasets. Since LSTM-FCN employs RNN as a component, it cannot fully leverage

the power of GPUs, leading to extended training time. In comparison, transformer [16] learns both temporal dependencies and inter-variable relations based on positional embedding and attention mechanism. It achieves state-of-the-art performance on several time-series datasets [48, 49] but suffers extended training time and overfitting on small datasets [19] due to its massive trainable parameters. It also finds difficulty in capturing short-range temporal information when compared with RNN.

2.2 Attention Mechanism

Attention mechanism was first used in the seq2seq model for machine translation [18]. A vanilla seq2seq model first feeds the input sequence to an encoder (which consists of multiple recurrent layers) [18] to generate hidden states and outputs. It then collects the hidden states of all the steps to represent the information of the input. An attention mechanism forces the model to learn the weights of hidden states in the decoder part during this process. Thus, the model can focus on a specific region of the input sequence, leading to a significant performance improvement.

Recent studies have designed different attention modules and applied them to various domains [50, 51]. Among them, Squeeze-and-Excitation Block (SE) [52] is widely used for various tasks thanks to its easiness of implementation. SE works in two steps. First, it uses global average pooling to obtain an information vector of feature maps from different channels. Then, it employs fully connected layers to capture the inter-relations between feature maps to learn the weights of feature maps and highlight the critical information.

3 Our Approach

We propose Attentional Gated Res2Net for accurate classification of multivariate time series of various lengths. In particular, we incorporate gating and attention mechanisms on top of Res2Net [53], where *gate*s control the information flow across the groups of convolutional filters, and the *attention* module harnesses the feature maps at different levels of granularity.

The overall architecture of AGRes2Net (shown in Figure 1) consists of two stages: Convolution and Attention. We illustrate these two stages in the following subsections, respectively.

3.1 Convolution stage

We design the convolution stage based on Res2Net [53], a CNN backbone specially designed to achieve multi-scale receptive fields based on group convolution. Group convolution first appeared in AlexNet [54] and significantly reduced the number of the parameters in that model. It has since been adopted in many lightweight networks [55, 56] to generate a large number of feature maps with a small number of parameters.

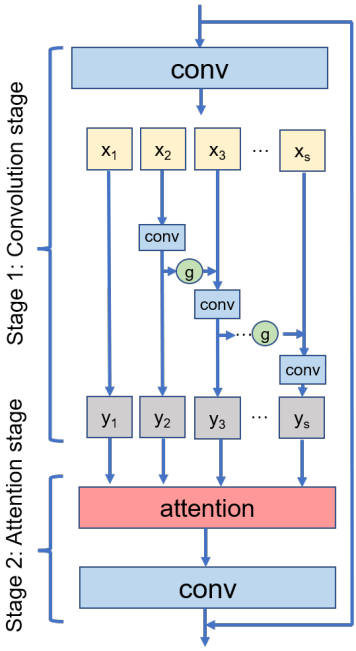

Fig. 1 The structure of Attentional Gated Res2Net. It consists of two stages: convolution and attention. The convolution stage feeds the input to a convolutional layer for channel expansion and then groups the output along the channel. Each group (except the first) conducts convolution based on its input and its precedent group's output (passed through gates). The attention stage forces the model to consider the temporal information at different levels of granularity. Finally, the network uses a convolutional layer for channel compression and information fusion.

Unlike conventional CNNs, which use a single set of filters to work on all channels, Res2Net includes multiple groups of filters and uses a separate group to handle each subset of channels. These filter groups are connected in a hierarchical, residual-like style, and they work as follows. First, a convolutional layer takes the input data and outputs a feature map for channel expansion. Then, the feature map is split into groups along the channel, generating groups of feature maps, i.e., *input feature map*s. Finally, for each input feature map, a separate group of filters extracts features and generates the corresponding output, i.e., an *output feature map*. In particular, when extracting features from an input feature map, the filter group also takes into account the output of the filter group that comes immediately before it. The whole process repeats until all input feature maps are processed.

Suppose $\mathbf{X}$ is the feature map obtained from channel expansion, and $\mathbf{X}$ is evenly divided into s groups, $\{\mathbf{x}_i\}_{i=1}^{s}$, where $\mathbf{x}_i$ denotes the ith group. Each group contains an input feature map that has the same temporal size but contains only $1/s$ of the channels in $\mathbf{X}$. Let $\mathbf{K}_i$ be the convolution operation. Then, given an input feature map $\mathbf{x}_i$, the convolution output, $\mathbf{y}_i$, is calculated

as follows:

$$\mathbf{y}_i = \begin{cases} \mathbf{x}_i & i = 1 \\ \mathbf{K}_i\left(\mathbf{x}_i\right) & i = 2 \\ \mathbf{K}_i\left(\mathbf{x}_i + \mathbf{y}_{i-1}\right) & 2 < i \leqslant s. \end{cases} \tag{1}$$

By feeding the concatenation of all the outputs to a convolutional layer, Res2Net achieves multi-scale receptive fields to facilitate multivariate time series classification. However, it has difficulty in controlling the information flow between the feature-map groups—at each step, $\mathbf{y}_i$ is always fully sent to the next group regardless of whether it avails or harms the model's performance.

Addressing this limitation is important as it enables to model to control how to weigh the precedent output feature map against the current input feature map in an input-dependent manner. This, in turn, mitigates the problem of vanishing gradients without having to take long delays. To this end, we introduce the gating mechanism [31] into Res2Net at the convolutional stage to enhance feature extraction. Specifically, in our model (shown in Figure 1), all groups of feature maps (except the first) are sent to convolutional layers for feature extraction, and a gating unit lies between each pair of adjacent feature-map groups to control how much information flows from the precedent to the current group. Given a feature-map group (or more specifically, input feature map), $\mathbf{x}_i$, the value of the corresponding gate, $\mathbf{g}_i$, is calculated as follows:

$$\mathbf{g}_i = \tanh\left(a\left(\text{concat}\left(a(\mathbf{y}_{i-1}), a\left(\mathbf{x}_i\right)\right)\right)\right). \tag{2}$$

where a can be either fully-connected or 1-D convolutional layers, *concat* is the concatenation operation, and *tanh* is the activation function commonly used for gates.

Note that, we only use the precedent output feature map $\mathbf{y}_{i-1}$ and the current input feature map $\mathbf{x}_i$ to calculate the gate—this is different from the gating mechanism in [31]. More specifically, we omit the undivided feature map $\mathbf{X}$ as it contains redundant information and does not significantly improve the performance. Eventually, after the convolution stage, we obtain $\mathbf{y}_i$ as follows:

$$\mathbf{y}_i = \begin{cases} \mathbf{x}_i & i = 1 \\ \mathbf{K}_i\left(\mathbf{x}_i\right) & i = 2 \\ \mathbf{K}_i\left(\mathbf{x}_i + \mathbf{g}_i \cdot \mathbf{y}_{i-1}\right) & 2 < i \leqslant s. \end{cases} \tag{3}$$

3.2 Attention stage

The convolution stage only considers the information flow between adjacent feature-map groups. As such, it limits the model's ability to capture the dependencies between groups that have long distances in-between. In this regard, we design an attention stage to attend to a certain part when processing output feature maps. In particular, we propose two types of attention modules, namely channel-wise attention module and block-wise attention module, to harness multi-granular temporal patterns effectively.

3.2.1 Channel-wise attention

Channel-wise attention captures the relations between channels of the convolution stage's output, i.e., output feature maps, $\{\mathbf{y}_i\}_{i=1}^{s}$, where s is the number of feature-map groups in the convolution stage.

Suppose every $\mathbf{y}_i$ contains the same number of channels, say J channels—this is reasonable as we divide the original feature map $\mathbf{X}$ evenly along the channel. Let $\mathbf{h}_{i,j}$ be the feature map for the jth channel of $\mathbf{y}_i$. We use three fully-connected layers to learn the query, key, and value of $\mathbf{h}_{i,j}$ (denoted by $\mathbf{q}_{i,j}$, $\mathbf{k}_{i,j}$, and $\mathbf{v}_{i,j}$, respectively). Similarly, we denote by $\mathbf{q}_{m,n}$, $\mathbf{k}_{m,n}$, and $\mathbf{v}_{m,n}$ the query, key, and value of $\mathbf{h}_{m,n}$, and the feature map for the nth channel of $\mathbf{y}_m$. Given two different feature maps, $\mathbf{h}_{i,j}$ and $\mathbf{h}_{m,n}$, we calculate the channel-wise attention as follows:

$$\textbf{attention}\left(\mathbf{q}_{i,j}, \mathbf{k}_{m,n}\right) = \frac{\mathbf{q}_{i,j}\mathbf{k}_{m,n}^{T}}{\sqrt{J}} \tag{4}$$

Once computed, we can update the feature map of every channel according to its relations with all the other feature maps. As the feature maps contain temporal information within various ranges, channel-wise attention can capture temporal dependencies at multiple levels of granularity. Based on the above, the updated feature map $\tilde{\mathbf{h}}_{i,j}$ can be calculated as follows:

$$\tilde{\mathbf{h}}_{i,j} = \sum_{s}\sum_{J} \text{Softmax}\left(\frac{\textbf{attention}\left(\mathbf{q}_{i,j}, \mathbf{k}_{m,n}\right)}{\sum_{s}\sum_{J}\textbf{attention}\left(\mathbf{q}_{i,j}, \mathbf{k}_{m,n}\right)}\right)\mathbf{v}_{m,n} \tag{5}$$

Given s output feature maps each having J channels with k dimensions, the total number of feature maps for channel-wise attention is $s \times J$, resulting in the computational complexity of $\mathcal{O}\left((s \times J)^2 k\right)$.

3.2.2 Block-wise attention

Block-wise attention regards each $\mathbf{y}_i$ as an individual block that contains temporal information at a certain granularity. Instead of calculating attention values along the channel, block-wise attention directly feeds $\mathbf{y}_i$ to the fully-connected layers to calculate the corresponding query, key, and value. Block-wise attention has advantages in mitigating *overfitting* as it considers sparse relations when computing the attention.

Suppose $\mathbf{y}_i$ and $\mathbf{y}_m$ are two output feature maps. We denote by $\mathbf{q}_i$, $\mathbf{k}_i$ and $\mathbf{v}_i$ the query, key and value of $\mathbf{y}_i$; similarly, we denote by $\mathbf{q}_m$, $\mathbf{k}_m$ and $\mathbf{v}_m$ the query, key and value of $\mathbf{y}_m$. Then, we calculate the block-wise attention as follows:

$$\textbf{attention}\left(\mathbf{q}_i, \mathbf{k}_m\right) = \frac{\mathbf{q}_i\mathbf{k}_m^{T}}{\sqrt{s}} \tag{6}$$

Once computed, we can update the feature map of every block according to their relations with all the other feature maps. And the updated feature

Table 1 A list of our experimental datasets.

Dataset	Task	#Classes	#Variables	Length	Train-test ratio	SOTA
Action 3D [57]	Action Recognition	20	570	100	48:52	MALSTM-FCN [47]
Ozone[1]	Weather Forecasting	2	72	291	50:50	MLSTM-FCN[47]
AREM[1]	Activity Recognition	7	7	480	50:50	MALSTM-FCN [47]
LP5[1]	Failure Detection	5	6	15	39:61	MUSE [58]
EEG[1]	EEG Classification	2	13	117	50:50	MLSTM-FCN [47]
Gesture Phase[1]	Gesture Recognition	5	18	214	50:50	MLSTM-FCN [47]
ECG[2]	ECG Classification	2	2	147	50:50	MUSE [58]
FingerMovements[3] [59]	Movement Classification	2	28	50	76:24	InceptionTime [60]
DuckDuckGeese[3] [59]	Audio Classification	5	1345	270	50:50	InceptionTime [60]
HeartBeat[3] [59]	Audio Classification	2	61	405	49:51	Canonical Interval Forest [61]
LSST[3] [59]	Signal Classification	14	36	6	50:50	MUSE [58]
MotorImagery[3] [59]	EEG Classification	2	3000	64	74:26	Time Series Forest [21]
SelfRegulationSCP2[3] [59]	EEG Classification	2	1152	7	53:47	DTW [62]
StandWalkJump[3] [59]	Activity Recognition	3	2500	4	45:55	ROCKET [63]

map for each block, $\tilde{\mathbf{y}}_i$, can be calculated as follows:

$$\tilde{\mathbf{y}}_i = \sum_s \text{Softmax}\left(\frac{\textbf{attention}\left(\mathbf{q}_i, \mathbf{k}_j\right)}{\sum_s \textbf{attention}\left(\mathbf{q}_i, \mathbf{k}_j\right)}\right)\mathbf{v}_j \tag{7}$$

Given s feature maps, each having J channels with k dimensions, the computational complexity of block-wise attention is $\mathcal{O}\left(s^2Jk\right)$.

4 Experiments

This section reports our extensive experiments to evaluate our proposed approach, including comparisons against baselines, ablation studies, and parameter studies on several public time-series datasets. We demonstrate that our approach can be used as a plugin to improve the performance of state-of-the-art methods and provide practical advice on how to adapt our approach to a specific problem.

4.1 Datasets

We conducted experiments on 14 public multivariate time series datasets (summarized in Table 1). These datasets cover various tasks from different application domains, such as activity recognition, EEG classification, and weather forecasting. They contain time series of various lengths with different numbers of variables. We have carefully selected these datasets to reflect applications in various domains and ensure that they are diverse enough in the length and variable number of time series to reflect different difficulty levels in real-world multivariate time-series classification problems.

4.2 Baseline Methods

We selected several competitive baselines and state-of-the-art (SOTA) methods to compare with our approach.

[1] https://archive.ics.uci.edu/ml/index.php
[2] http://www.cs.cmu.edu/∼bobski
[3] https://www.cs.ucr.edu/∼eamonn/time_series_data_2018

- **Res2Net** [53]: this is a CNN backbone that uses group convolution and hierarchical residual-like connections between convolutional filter groups to achieve multi-scale receptive fields.
- **GRes2Net** [31]: this work incorporates gates in Res2Net, where the gates' values are calculated based on a different method from ours—it additionally takes into account the original feature map before it is divided into groups when calculating gates' values.
- **Res2Net+SE**: this work combines Res2Net with a Squeeze-and-Excitation Block (SE) [52] to leverage the effectiveness of attention modules.
- **GRes2Net+SE**: this work combines GRes2Net with SE to leverage the effectiveness of attention modules.

We briefly introduce the SOTA methods for the experimental datasets below. A full list of SOTA methods is given in Table 1.

- **MLSTM-FCN** [47]: a multivariate LSTM fully convolutional network that concatenates the outputs of two parallel blocks: a fully convolutional block (embedded with SEs) and an LSTM block. It is a variant of LSTM-FCN.
- **MALSTM-FCN** [47]: a multivariate attention LSTM fully convolutional network, which resembles MLSTM-FCN but replaces LSTM cells with attention LSTM cells.
- **MUSE** [58]: a model that extracts and filters multivariate features by encoding context information into each feature.
- **InceptionTime** [60]: a CNN-based model transferred from computer vision to time series classification, which stacks multiple parallel convolutional filters for temporal feature extraction.
- **Time Series Forest** [21]: an ensemble tree-based method that employs a combination of entropy gain and a distance measure to evaluate the differences between time-series sequences.
- **Canonical Interval Forest** [61]: a model that refines *Time Series Forest* by upgrading the interval-based component.
- **Dynamic Time Warping (DTW)**[62]: a traditional distance-based machine learning method for time series analysis.
- **Random Convolutional Kernel Transform (ROCKET)** [63]: a CNN-based model that uses random convolutional kernels to extract multi-granular temporal features.

4.3 Model Configuration and Evaluation Metric

We followed the methods as illustrated in the SOTA methods to preprocess the datasets. In particular, we normalized each dataset to zero mean and unit standard deviation. We also applied zero paddings to cope with sequences with various lengths in the same training set. The experimental results of each method were obtained under the optimal or suggested settings as provided in the original paper.

Table 2 Experiment configuration settings

Dataset	Number of Layers	Number of Groups	Dropout Rate
FingerMovements	4	4	0.2
DuckDuckGeese	4	64	0.3
HeartBeat	2	64	0.2
LSST	2	8	0.25
MotorImagery	6	64	0.2
SelfRegulationSCP2	4	8	0.5
StandWalkJump	4	64	0.2
Action 3D	4	8	0.3
Ozone	4	8	0.25
AREM	6	64	0.25
LP5	4	2	0.4
EEG	4	4	0.5
Gesture Phase	5	8	0.4
ECG	4	8	0.5

To ensure a fair comparison, we set all the models based on Res2Net, GRes2Net, and our approach contained the same number of feature-map groups and used identical filters for each group.

We used our model as the backbone for feature extraction and trained our model for 500 training epochs using Adam [64] optimizer. The learning rate was set to 0.001 and adjusted to 1/10 of itself after every 100 epochs. The dropout rate was set to 0.4 to avoid possible overfitting. We repeated the training and test processes five times and took the average of multiple runs as the final results; this mitigates the impact of randomized parameter initialization. The details including the number of layers, the number of convolutional groups, and the dropout rate settings can be found in Table 2.

We used *accuracy*, which is currently used by all the SOTA methods on the experimental datasets, as the metric for evaluating the methods. However, accuracy is not comprehensive enough to measuring the performance of the classifier. Although the vast majority of the related work uses accuracy as the only evaluation metric, we additionally use *precision*, *recall*, and *F-score* in our parameter and ablation studies to gain further insights into how our model performs.

4.4 Comparison of Different Methods

Table 3 shows a performance comparison of all the methods on the experimental datasets. Our proposed model, using either channel-wise or block-wise attention, consistently outperformed all the other compared methods on all the 14 datasets, demonstrating our model's superiority in solving MTSC in diverse contexts regardless of the lengths of time-series sequences.

Channel-wise attention favors longer time-series sequences, as it beats block-wise attention on all the top-8 datasets with the longest sequences. The results conform to our intuition that channel-wise attention may have an edge on capturing multi-granular temporal information.

Block-wise attention tends to excel on datasets that contain fewer variables. Among the top-4 datasets with the least variables, it beats channel-wise

Table 3 Accuracy of different models on 14 benchmark datasets. AGRes2Net+CA and AGRes2Net+BA represent our Attentional Gated Res2Net model incorporated with channel-wise attention and block-wise attention, respectively. The improvement is the the comparison between SOTA and the proposed model.

Dataset	Res2Net	Res2Net+SE	GRes2Net	GRes2Net+SE	SOTA	AGRes2Net+BA	AGRes2Net+CA	Improvement (%)
FingerMovements	0.5240	0.5280	0.5340	0.5480	0.5613	**0.6240**	0.5820	11.17
DuckDuckGeese	0.6360	0.6680	0.6560	0.6800	0.6347	0.6880	**0.7080**	11.55
HeartBeat	0.6463	0.7415	0.7512	0.7561	0.7652	0.7853	**0.8663**	13.21
LSST	0.5268	0.5447	0.5341	0.5799	0.6362	0.5843	**0.6671**	4.86
MotorImagery	0.5220	0.5340	0.5380	0.5740	0.5380	0.6240	**0.6280**	16.73
SelfRegulationSCP2	0.5367	0.5522	0.5444	0.5555	0.5369	0.5711	**0.6210**	15.66
StandWalkJump	0.3333	0.4000	0.3333	0.3333	0.4556	**0.4667**	0.3333	2.44
Action 3D	0.7457	0.7182	0.7301	0.8037	0.7542	0.8350	**0.8617**	14.25
Ozone	0.7989	0.8264	0.8034	0.8390	0.8150	0.8494	**0.8620**	5.77
AREM	0.7692	0.8462	0.8205	0.8717	0.8462	**0.9231**	0.8974	9.09
LP5	0.5642	0.5799	0.684	0.6328	0.7100	**0.7396**	0.7326	4.17
EEG	0.5781	0.5469	0.6094	0.6406	0.6563	**0.6719**	**0.6719**	2.38
Gesture Phase	0.5859	0.5898	0.6601	0.6641	0.5353	0.6445	**0.6953**	29.89
ECG	0.7200	0.8000	0.8400	0.8300	0.9300	0.8500	**0.9400**	1.08

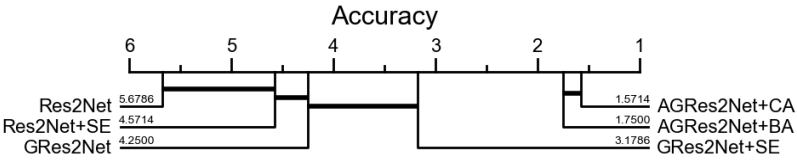

Fig. 2 Critical difference diagram of the arithmetic means of the ranks on all datasets.

attention on 3 of them (AREM, LP5, and EEG); this is also consistent with our intuition that block-wise attention may have advantages in preventing overfitting thanks to the sparse relations considered in its attention calculation.

An exception occurs on the ECG dataset, which has as few as two variables; this reason lies in that this dataset contains abundant sequences that allow for the channel-wise attention to fully exploit the training data without causing overfitting.

Figure 2 shows the result of the Wilcoxon signed-rank test on the baseline methods' performance. It shows that, overall, our model achieves similar classification performance when using channel-wise attention and block-wise attention. Either way, our model performs significantly better than the baselines. This result demonstrates the effectiveness of harnessing interdependencies between variables and multi-granular feature maps (as our model does use gates, attention, and group convolution) in improving classification performance on sequences of various lengths.

4.5 Impact of Depth and Width of Model

In this experiment, we study how the depth and width of our model impact the classification performance. Generally, a deeper and wider model has a stronger capability to capture complex relations from data. Our model becomes more complex as we increase its depth (by stacking more layers), width (by expanding the number of feature-map groups), or both.

Table 4 Training and test results under varying widths and depths.

Dataset	Configuration		Train				Test			
	Width	Depth	Accuracy	Recall	Precision	F-score	Accuracy	Recall	Precision	F-score
HeartBeat	8	8	0.9482	0.8933	0.9108	0.9019	0.7622	0.6848	0.6953	0.6901
	16	4	0.9712	0.9221	0.9423	0.9321	0.8443	0.6859	0.6879	0.6869
	32	2	**0.9742**	**0.9298**	**0.9491**	**0.9394**	**0.8570**	**0.8602**	**0.8551**	**0.8576**
Action 3D	8	8	0.8525	0.8225	0.8213	0.8219	0.7490	0.7767	0.7458	0.7609
	16	4	0.9003	0.8864	0.8848	0.8856	0.8434	0.8171	0.8091	0.8130
	32	2	**0.9068**	**0.8908**	**0.8908**	**0.8908**	**0.8515**	**0.8701**	**0.8462**	**0.8579**

We trained our model under different width and depth settings and applied different types of attention for the experiment. Considering the many experimental datasets, we only show the results on two representative datasets, *Action 3D* and *Heartbeat*. The former has medium-length sequences and a large number of variables; in contrast, the latter has long sequences but a medium number of variables, making them ideal for exemplifying the experimental results. In particular, we show the results of our model after applying channel-wise attention and block-wise attention on *Heartbeat* and *Action 3D* datasets, respectively.

Our results (Table 4) show that wider models beat deeper models in both the training and test phases. While stacking multiple layers leads to large receptive fields that can capture dependencies in a larger range, a wider model can achieve receptive fields with multiple sizes and fuse the feature maps from different convolution filters to learn multi-granular temporal patterns. In comparison, a wider model leverages the temporal features of time-series sequences more effectively, making it generally a better choice. Several studies [65, 66] in the computer vision field draw similar conclusions.

,

4.6 Impact of Group Number

In this experiment, we further explore the impact of the hyperparameter s, which determines the number of feature-map groups (as well as the number of filter groups) in our model. Intuitively, a larger s gives a wider model that can fuse more temporal features extracted by convolutional filters with multiple sizes of receptive fields, thus facilitating capturing long-range dependencies.

We kept all other settings (e.g., number of layers, epochs, learning rate, dropout rate) unchanged while varying the value of s to explore its influence on classification results. Similar to the precedent experiment, we show the experimental results on four datasets that have significantly different lengths of sequences (namely *LP5*, *AREM*, *Ozone*, and *Action 3D*) to avoid information overload. We used block-wise attention on the first two datasets and channel-wise attention on the last two.

Our results (Table 5) show our model consistently achieved better performance during training as s increased. And we can easily tune our model towards capturing a broader range of temporal information by allowing for more groups with a greater s. However, greater values of s bring the risk of overfitting, demonstrated by decreased performance in the test phase, e.g., in

Table 5 Training and test results under different s. We set greater s values for the AREM dataset as it has much longer sequences than the others do. We set 6 layers for Ozone, 6 layers for AREM, 4 layers for Action 3D, and 4 layers for LP5.

Dataset	s	Train				Test			
		Accuracy	Recall	Precision	F-score	Accuracy	Recall	Precision	F-score
Ozone	2	0.9425	0.9434	0.9437	0.9436	0.8436	0.8299	0.8281	0.8289
	4	0.9529	0.9501	0.9428	0.9464	0.8563	0.8609	0.8598	0.8603
	8	0.9635	0.9699	0.9640	0.9669	**0.8570**	**0.8602**	**0.8551**	**0.8576**
	16	0.9792	0.9771	0.9774	0.9772	0.8257	0.8346	0.8243	0.8294
	32	**0.9844**	**0.9827**	**0.9830**	**0.9829**	0.8257	0.8302	0.8372	0.8337
AREM	4	0.7907	0.7798	0.7534	0.7664	0.7949	0.7007	00.7171	0.7088
	8	0.8139	0.8452	0.8250	0.8350	0.7435	0.6871	0.8268	0.7505
	16	0.8605	0.8870	0.8397	0.8627	0.8205	0.7756	0.8648	0.8178
	32	0.8837	0.9048	0.8939	0.0.8993	0.8718	0.8163	0.9056	0.8586
	64	**0.9767**	**0.9821**	**0.9841**	**0.9831**	**0.8692**	**0.9619**	**0.8452**	**0.8998**
Action 3D	2	0.8138	0.7968	0.8426	0.8191	0.7088	0.7212	0.7352	0.7281
	4	0.8219	0.8149	0.8595	0.8366	0.7207	0.7255	0.7391	0.7322
	8	0.8232	0.8109	0.8592	0.8344	**0.8617**	**0.8764**	**0.8711**	**0.8737**
	16	0.8263	0.8201	0.8638	0.8414	0.8318	0.8359	0.8131	0.8243
	32	**0.8350**	**0.8296**	**0.8681**	**0.8484**	0.8252	0.8329	0.7991	0.8156
LP5	2	0.7813	0.7964	0.8125	0.8043	**0.7396**	**0.7818**	**0.7214**	**0.7504**
	4	0.8125	0.8511	0.8398	0.8313	0.7309	0.7014	0.7568	0.7158
	8	0.8594	0.8750	0.8593	0.8671	0.7014	0.7052	0.7665	0.7033
	16	0.8906	0.9142	0.8809	0.9022	0.6771	0.7022	0.7237	0.6894
	32	**0.9531**	**0.9714**	**0.9418**	**0.9622**	0.6215	0.6187	0.6454	0.6201

the case of the Qzone and Action 3D datasets. The results suggest the necessity of tuning this hyperparameter s given a specific dataset to gain the best performance.

Beyond the above results, we may consider our model as *recurrent* because each group's output feature map is sent to the subsequent group. Following this idea, we may regard group number s as the number of steps that the model takes during its recurrent computation. While traditional convolutional neural networks obtain larger receptive fields by stacking multiple layers or employing dilation convolution layers, they are not as flexible or effective as our model in capturing multi-granular temporal information.

4.7 Impact of Attention Modules

The superiority of our attention modules over SE is indicated by our model outperforming those baselines that incorporate SE [52] (see Table 3). Specifically, the SE module uses *global average pooling*, which generates a scalar to represent the feature map of each channel. In comparison, our attention mechanisms (channel-wise and block-wise attention) avoid using global average pooling, thus preventing the information loss caused by the pooling operation.

Table 6 further shows our model's performance when using the two attention modules during training and test. We choose to show the results on three datasets, which cover a large range of variable numbers (7 for AREM, 72 for Ozone, and 570 for Action 3D). The results (Table 6) are consistent with our findings in Section 4.4 that channel-wise attention generally beats block-wise attention except for small datasets with very few variables.

Table 6 Training and test results of our model with different attention modules.

Dataset	Attention	Train				Test			
		Accuracy	Recall	Precision	F-score	Accuracy	Recall	Precision	F-score
Ozone	Block-wise	0.7088	0.7212	0.7352	0.7281	0.6793	0.6644	0.6685	0.6664
	Channel-wise	**0.8232**	**0.8109**	**0.8592**	**0.8344**	**0.8604**	**0.8669**	**0.8437**	**0.8551**
AREM	Block-wise	0.8837	0.9048	0.8939	0.8993	**0.8718**	**0.8163**	**0.9056**	**0.8586**
	Channel-wise	**0.9767**	**0.9821**	**0.9841**	**0.9831**	0.8205	0.7483	0.8772	0.8076
Action 3D	Block-wise	0.9091	0.8908	0.9091	0.8996	0.8181	0.8306	0.8150	0.8227
	Channel-wise	**0.9635**	**0.9699**	**0.9640**	**0.9669**	**0.8570**	**0.8602**	**0.8551**	**0.8576**

Table 7 Ablation test for our model.

Dataset	Model	Accuracy	Recall	Precision	F-score
EEG	Res2Net	0.5781	0.5713	0.5882	0.5796
	Res2Net + Gates	0.5938	0.5943	0.5943	0.5943
	Res2Net + channel-wise attention	0.6094	0.6105	0.6417	0.6257
	Res2Net + Gates + channel-wise attention	**0.6719**	**0.6750**	**0.6833**	**0.6791**
AREM	Res2Net	0.7692	0.7469	0.7639	0.7530
	Res2Net + Gates	0.8205	0.7551	0.8762	0.8112
	Res2Net + block-wise attention	0.8718	0.8163	0.8929	0.8529
	Res2Net + Gates + block-wise attention	**0.9231**	**0.8762**	**0.9571**	**0.9149**

As for this experiment, both Ozone and Action 3D datasets contain many variables (72 and 570) and sufficient sequences during training for channel-wise attention to perform well. In contrast, AREM contains only 43 sequences that cover as many as seven classes. The number of sequences is extremely limited for each class, making channel-wise attention easily lead to overfitting.

4.8 Ablation Study

We conducted ablation studies to explore the effectiveness of gates and our attention modules. The model without gates and attention module is the same as vanilla Res2Net. We separately incorporate gates, attention, and both attention and gates in Res2Net and compare the results.

Again, we only present the results on *EEG* and *AREM* datasets to avoid information overload. For each dataset, we tested the attention mechanism that led to inferior performance to the other, i.e., channel-wise attention on the EEG dataset and block-wise attention on the AREM dataset, to make the comparisons more evident.

Our results (Table 7) show the attention modules contribute slights more than gates on improving the performance of Res2Net, but every component contributes significantly to the improved performance.

4.9 Time Consumption of Attention Modules

We conducted experiments to analyze the extra time consumption of the attention modules. We select two datasets, *MotorImagery* and *DuckDuckGeese*, because their length and variable number are significantly large. We trained the models on i7-8700K CPU instead of GPU because GPUs are too powerful that can alleviate the impact. We stacked 4 layers and used 64 groups of convolutional filters at each layer. We trained the model with channel-wise

Table 8 Time consumption comparison with attention modules and without attention modules on DuckDuckGeese, CA means channel-wise attention and BA means block-wise attention. The data in the brackets is the standard deviation.

	Without Attention	With CA	With BA
Training time consumption (s)	1.3991 (0.1627)	3.0911 (0.2772)	2.8689 (0.2391)
Test time consumption (s)	0.7675 (0.0977)	1.1226 (0.0737)	0.980 (0.0860)

Table 9 Time consumption comparison with attention modules and without attention modules on MotorImagery, CA means channel-wise attention and BA means block-wise attention. The data in the brackets is the standard deviation.

	Without Attention	With CA	With BA
Training time consumption (s)	37.7661 (1.7530)	186.5941 (9.0698)	75.9330 (2.808)
Test time consumption (s)	7.1772 (0.3879)	22.1444 (3.5978)	8.0369 (0.4131)

attention, with block-wise attention, and without attention module 300 epochs separately, and recorded the training time and test time per epoch. We calculate and give the average time consumption and the standard deviation. The results are shown in Table 8 and Table 9.

According to the results, we can see that the time consumption significantly increases when using the attention module. Among the two attention modules, channel-wise attention is more computationally expensive. Compared with the model without any attention module, the time consumption of channel-wise attention for training is about 2.2 times on DuckDuckGeese and is 4.9 times on MotorImagery. While the time consumption of block-wise attention for training is 2.1 times on DuckDuckGeese and is 2 times on MotorImagery. Although attention modules improve the performance (shown in section 4.8), they also make the model less efficient, which brings challenges for employing the model on devices with limited computing resources.

4.10 Impact of Feature Dimension Reduction

As discussed in section 4.9, we find that our model is less efficient when the time series contains too many variables. So we conducted experiments to explore the impact of combining feature dimension reduction algorithms with the AGRes2Net. We select *SelfRegulationSCP2* dataset as it contains 1152 variables. We used Principal Component Analysis (PCA) to reduce the number of variables from 1152 to 28. we stacked 4 layers, and each layer has 8 groups of convoltuional filters. We use the same dropout rate and experiment settings that are described in the section 4.3. We trained the model on i7-8700K CPU. We recorded the performance including accuracy, precision, recall, F1score, and time consumption of both training phase and test phase. The results are given in Table 10.

According to the results, all the performances go poorer, but the time consumption is significantly reduced. Specifically, the accuracy after using PCA decreases 9.59%, but the test speed of the model is about 33 times faster. So dimension reduction algorithms (such as PCA) are practicable for dropping

Table 10 Performance Comparison between the data with PCA and without PCA on SelfRegulationSCP2. The data in the brackets is the standard deviation.

	Training					Test				
	Accuracy	Precision	Recall	F1Score	Time per epoch	Accuracy	Precision	Recall	F1Score	Time per epoch (s)
Without PCA	**0.9386**	**0.9300**	**0.9297**	**0.9298**	18.3621 (1.0645)	**0.6210**	**0.6167**	**0.6132**	**0.6149**	6.0603 (0.2679)
With PCA	0.8958	0.8799	0.8788	0.8793	**0.6664 (0.8050)**	0.5667	0.5611	0.5556	0.5583	**0.1981 (0.0235)**

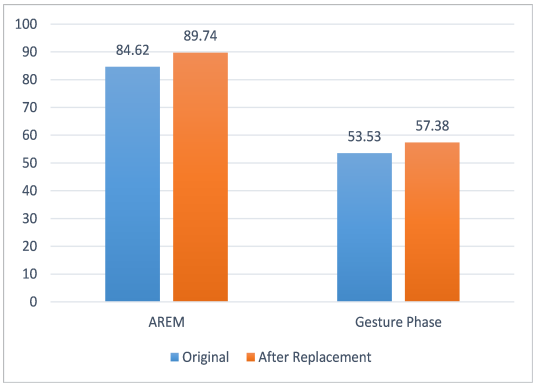

Fig. 3 Accuracy comparison between the vanilla MLSTM-FCN (blue bar) and the MLSTM-FCN where our model replaces the convolutional modules (orange bar). Block-wise attention and channel-wise attention are applied to the AREM dataset and the Gesture Phase dataset, respectively.

some features if we want to make the model more efficient in facing the time series that contain too many variables.

4.11 Effectiveness of Our Model as a Plugin

We use MLSTM-FCN, the SOTA architecture on most datasets (as shown in Table 1), to demonstrate the effectiveness of our model as a plugin. The original MLSTM-FCN follows a CNN-LSTM parallel architecture. The input goes through multiple LSTMs and CNNs, and the outputs are concatenated and go through a fully connected layer for information fusion. We conducted this experiment by replacing the original convolutional modules of MLSTM-FCN with our model while preserving the architecture and all the other parts in MLSTM-FCN.

We show the comparison results on two datasets, *AREM* and *Gesture Phase*, to demonstrate the impact of our model on the overall performance of MLSTM-FCN. Specifically, we adopted block-wise attention on the AREM dataset and channel-wise attention on the Gesture Phase dataset without particular reasons. We omit to show the results on other datasets as they draw similar conclusions.

The results (Figure 3) show a significant improvement in the classification accuracy of MLSTM-FCN on both datasets after the replacement, demonstrating the positive effect of our model on the performance of existing multivariate time series classification models when used as a plugin.

Table 11 Performance comparison based on the different variable numbers on LSST

	Block-wise Attention				Channel-wise Attention			
Variable Number	Accuracy	Precision	Recall	F1Score	Accuracy	Precision	Recall	F1Score
3	0.2173	0.1328	0.1221	0.1272	0.1473	0.1316	0.1154	0.1229
4	0.5799	0.5795	0.5722	0.5758	0.6553	0.6375	0.6358	0.6367

Table 12 Performance comparison based on the different variable numbers on HeartBeat

	Block-wise Attention				Channel-wise Attention			
Variable Number	Accuracy	Precision	Recall	F1Score	Accuracy	Precision	Recall	F1Score
2	0.7456	0.6207	0.6321	0.6264	0.6451	0.5560	0.5496	0.5528
3	0.6369	0.5970	0.5975	0.5972	0.7130	0.6072	0.6156	0.6114

4.12 Exploring the Threshold for Choosing Channel-wise Attention and Block-wise Attention

As discussed in the previous section, channel-wise attention performs better and vice versa. This section further explores whether a standard threshold exists for choosing the proper attention module. We select two datasets, *LSST* and *HeartBeat*, for experiments, because they contain many variables and channel-wise attention performs better than block-wise attention, and we can use dimension reduction methods to tune the variable numbers to find when the block-wise attention performs better. We use PCA to gradually control the variable numbers. The results can be seen in Table 11 and Table 12.

According to the results, we can see the thresholds of the two datasets are different (3 for LSST and 2 for HeartBeat). Besides, when we reduce the variable number to 3 on LSST, the performance significantly decreases, making the results less convincing. According to the results, we can see the thresholds of the two datasets are different (3 for LSST and 2 for HeartBeat). Besides, when reducing the variable number to 3 on LSST, the performance of both attention modules is significantly decreased, making the results less convincing. Besides, from the results given in Table 3, we can see on the FingerMovements dataset, the block-wise attention performs better, while on the ECG dataset, the channel-wise attention outperforms block-wise attention. However, FingerMovements contains 28 variables, while ECG contains only 2 variables. To summarize, the threshold is case-by-case, and the standard threshold does not exist. Although we can follow a rule that using channel-wise attention is preferable in facing a dataset that has lots of variables (such as SelfRegulationSCP2, Action 3D, DuckDuckGeese, etc.), we still need to do empirical studies on each dataset to choose the proper attention module.

4.13 Practical Advice

We offer several suggestions on applying our model to broader scenarios based on the above experimental results and our analysis:

- *Avoid very deep models*: a wider model is generally more capable than a deeper model of addressing a general multivariate time series classification.

We should prioritize constructing wider models rather than stacking more layers when faced with a new problem.

- *Focus on tuning the hyperparameter s*: setting a larger s increases the number of convolutional-filter groups, leading to multiple receptive fields that capture temporal patterns in various ranges. Tuning the hyperparameter s is especially important for long time-series sequences to achieve the best possible performance. It is generally worthwhile to tune s ahead of investigating the optimal settings of other parameters.
- *Choose attention module based on variable number*: the number of variables is, by far, the most useful single criterion for deciding which attention module to choose for our model, based on our experiments. As discussed, block-wise attention is preferred for sequences with a small number of variables, and channel-wise attention is more suitable for sequences with massive variables. More criteria include the number of sequences available for training, the number of classes, and the length of sequences, which must be figured out case by case.

5 Conclusion and Future Work

In this paper, we propose a novel deep learning architecture called *Attentional Gated Res2Net* for accurate multivariate time series classification. Our model comprehensively incorporates gates and two types of attention modules to capture multi-granular temporal information. We evaluate the model on diverse datasets that contain sequences of various lengths with a wide range of variable numbers. Our experiments show the model outperforms several baselines and state-of-the-art methods by a large margin. We thoroughly investigate the effect of different components and settings on the model's performance and provide hands-on advice on applying our model to a new problem. Our test on plugging the model into a state-of-the-art architecture, MLSTM-FCN, demonstrates the potential for using our model as a plugin to improve the performance of existing models.

However, our attention modules increase the training, and inference is time-consuming facing the time series with many variables. Although some dimension reduction algorithms can alleviate the time consumption, it negatively influences classification accuracy. In the future, we aim to explore a pluggable feature selection module to select essential variables hence accelerating the training and inference process. Besides, our model still rely on manual fine-tuning for various datasets. We wish to make our model dynamic instead of static to ensure automatic adaptability based on the certain dataset.

6 Funding

This work was supported by the Australian Research Council (Grant numbers DE180100251, DP220103717 and LP180100654).

7 Conflict of Interest

The authors have no conflict of interest to declare that are relevant to the content of this article.

8 Data Availability Statement

The datasets generated and/or analysed during the current study are available from the corresponding author on reasonable request.

References

[1] Spiegel, S., Gaebler, J., Lommatzsch, A., De Luca, E., Albayrak, S.: Pattern recognition and classification for multivariate time series. In: Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data, pp. 34–42 (2011)

[2] Esling, P., Agon, C.: Time-series data mining. ACM Computing Surveys (CSUR) **45**(1), 1–34 (2012)

[3] Yu, Z., Lee, M.: Real-time human action classification using a dynamic neural model. Neural Networks **69**, 29–43 (2015)

[4] Chitra, R., Seenivasagam, V.: Heart disease prediction system using supervised learning classifier. Bonfring International Journal of Software Engineering and Soft Computing **3**(1), 01–07 (2013)

[5] Bai, L., Yao, L., Kanhere, S.S., Wang, X., Yang, Z.: Automatic device classification from network traffic streams of internet of things. In: 2018 IEEE 43rd Conference on Local Computer Networks (LCN), pp. 1–9 (2018). IEEE

[6] Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence **35**(8), 1798–1828 (2013)

[7] Aydın, S.: Deep learning classification of neuro-emotional phase domain complexity levels induced by affective video film clips. IEEE Journal of Biomedical and Health Informatics **24**(6), 1695–1702 (2019)

[8] Kılıç, B., Aydın, S.: Classification of contrasting discrete emotional states indicated by eeg based graph theoretical network measures. Neuroinformatics, 1–15 (2022)

[9] Aydın, S.: Cross-validated adaboost classification of emotion regulation strategies identified by spectral coherence in resting-state. Neuroinformatics, 1–13 (2021)

[10] Baydogan, M.G., Runger, G., Tuv, E.: A bag-of-features framework to classify time series. IEEE transactions on pattern analysis and machine intelligence **35**(11), 2796–2802 (2013)

[11] Kampouraki, A., Manis, G., Nikou, C.: Heartbeat time series classification with support vector machines. IEEE Transactions on Information Technology in Biomedicine **13**(4), 512–518 (2008)

[12] Bai, L., Yao, L., Wang, X., Kanhere, S.S., Xiao, Y.: Prototype similarity learning for activity recognition. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 649–661 (2020). Springer

[13] Bengio, Y., LeCun, Y., *et al.*: Scaling learning algorithms towards ai. Large-scale kernel machines **34**(5), 1–41 (2007)

[14] LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)

[15] Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural networks **4**(2), 251–257 (1991)

[16] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)

[17] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

[18] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)

[19] Xu, P., Kumar, D., Yang, W., Zi, W., Tang, K., Huang, C., Cheung, J.C.K., Prince, S.J., Cao, Y.: Optimizing deeper transformers on small datasets. In: ACL/IJCNLP (1) (2021)

[20] Di Gangi, M.A., Negri, M., Cattoni, R., Dessi, R., Turchi, M.: Enhancing transformer for end-to-end speech-to-text translation. In: Proceedings of Machine Translation Summit XVII: Research Track, pp. 21–31 (2019)

[21] Deng, H., Runger, G., Tuv, E., Vladimir, M.: A time series forest for classification and feature extraction. Information Sciences **239**, 142–153 (2013)

[22] Jović, A., Brkić, K., Bogunović, N.: Decision tree ensembles in biomedical time-series classification. In: Joint DAGM (German Association

for Pattern Recognition) and OAGM Symposium, pp. 408–417 (2012). Springer

[23] Zhang, D., Zuo, W., Zhang, D., Zhang, H.: Time series classification using support vector machine with gaussian elastic metric kernel. In: 2010 20th International Conference on Pattern Recognition, pp. 29–32 (2010). IEEE

[24] Lee, Y.-H., Wei, C.-P., Cheng, T.-H., Yang, C.-T.: Nearest-neighbor-based approach to time-series classification. Decision Support Systems **53**(1), 207–217 (2012)

[25] Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery **31**(3), 606–660 (2017)

[26] Seto, S., Zhang, W., Zhou, Y.: Multivariate time series classification using dynamic time warping template selection for human activity recognition. In: 2015 IEEE Symposium Series on Computational Intelligence, pp. 1399–1406 (2015). IEEE

[27] Tang, Y., Xu, J., Matsumoto, K., Ono, C.: Sequence-to-sequence model with attention for time series classification. In: 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), pp. 503–510 (2016). IEEE

[28] Tan, H.X., Aung, N.N., Tian, J., Chua, M.C.H., Yang, Y.O.: Time series classification using a modified lstm approach from accelerometer-based data: A comparative study for gait cycle detection. Gait & posture **74**, 128–134 (2019)

[29] Elsayed, N., Maida, A.S., Bayoumi, M.: Deep gated recurrent and convolutional network hybrid model for univariate time series classification. arXiv preprint arXiv:1812.07683 (2018)

[30] Zhao, B., Lu, H., Chen, S., Liu, J., Wu, D.: Convolutional neural networks for time series classification. Journal of Systems Engineering and Electronics **28**(1), 162–169 (2017)

[31] Yang, C., Jiang, M., Guo, Z., Liu, Y.: Gated res2net for multivariate time series analysis. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–7 (2020). IEEE

[32] Tang, W., Long, G., Liu, L., Zhou, T., Jiang, J., Blumenstein, M.: Rethinking 1d-cnn for time series classification: A stronger baseline. arXiv preprint arXiv:2002.10061 (2020)

[33] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015)

[34] Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1440–1448 (2015)

[35] Sun, X., Wu, P., Hoi, S.C.: Face detection using deep learning: An improved faster rcnn approach. Neurocomputing **299**, 42–50 (2018)

[36] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European Conference on Computer Vision, pp. 21–37 (2016). Springer

[37] He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2961–2969 (2017)

[38] Han, Z., Zhao, J., Leung, H., Ma, K.F., Wang, W.: A review of deep learning models for time series prediction. IEEE Sensors Journal (2019)

[39] Borovykh, A., Bohte, S., Oosterlee, C.W.: Conditional time series forecasting with convolutional neural networks. arXiv preprint arXiv:1703.04691 (2017)

[40] Hoermann, S., Bach, M., Dietmayer, K.: Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 2056–2063 (2018). IEEE

[41] Ding, X., Zhang, Y., Liu, T., Duan, J.: Deep learning for event-driven stock prediction. In: Twenty-fourth International Joint Conference on Artificial Intelligence (2015)

[42] Wallach, I., Dzamba, M., Heifets, A.: Atomnet: a deep convolutional neural network for bioactivity prediction in structure-based drug discovery. arXiv preprint arXiv:1510.02855 (2015)

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

[44] Liu, C.-L., Hsaio, W.-H., Tu, Y.-C.: Time series classification with multivariate convolutional neural network. IEEE Transactions on Industrial Electronics **66**(6), 4788–4797 (2018)

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

[46] Yang, C., Jiang, W., Guo, Z.: Time series data classification based on dual path cnn-rnn cascade network. IEEE Access **7**, 155304–155312 (2019)

[47] Karim, F., Majumdar, S., Darabi, H., Harford, S.: Multivariate lstm-fcns for time series classification. Neural Networks **116**, 237–245 (2019)

[48] Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., Zhang, W.: Informer: Beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of AAAI (2021)

[49] Rußwurm, M., Körner, M.: Self-attention for raw optical satellite time series classification. ISPRS Journal of Photogrammetry and Remote Sensing **169**, 421–435 (2020)

[50] Hu, J., Zheng, W.: A deep learning model to effectively capture mutation information in multivariate time series prediction. Knowledge-Based Systems **203**, 106139 (2020)

[51] Woo, S., Park, J., Lee, J.-Y., So Kweon, I.: Cbam: Convolutional block attention module. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 3–19 (2018)

[52] Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7132–7141 (2018)

[53] Gao, S., Cheng, M.-M., Zhao, K., Zhang, X.-Y., Yang, M.-H., Torr, P.H.: Res2net: A new multi-scale backbone architecture. IEEE transactions on pattern analysis and machine intelligence (2019)

[54] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25**, 1097–1105 (2012)

[55] Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1251–1258 (2017)

[56] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)

[57] Li, W., Zhang, Z., Liu, Z.: Action recognition based on a bag of 3d points.

In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops, pp. 9–14 (2010). IEEE

[58] Schäfer, P., Leser, U.: Multivariate time series classification with weasel+ muse. arXiv preprint arXiv:1711.11343 (2017)

[59] Dau, H.A., Keogh, E., Kamgar, K., Yeh, C.-C.M., Zhu, Y., Gharghabi, S., Ratanamahatana: The UCR Time Series Classification Archive (2018)

[60] Fawaz, H.I., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D.F., Weber, J., Webb, G.I., Idoumghar, L., Muller, P.-A., Petitjean, F.: Inceptiontime: Finding alexnet for time series classification. Data Mining and Knowledge Discovery **34**(6), 1936–1962 (2020)

[61] Middlehurst, M., Large, J., Bagnall, A.: The canonical interval forest (cif) classifier for time series classification. In: 2020 IEEE International Conference on Big Data (Big Data), pp. 188–195 (2020). IEEE

[62] Müller, M.: Dynamic time warping. Information retrieval for music and motion, 69–84 (2007)

[63] Dempster, A., Petitjean, F., Webb, G.I.: Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. Data Mining and Knowledge Discovery **34**(5), 1454–1495 (2020)

[64] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

[65] Xie, S., Girshick, R., Dollar, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)

[66] Zhang, H., Wu, C., Zhang, Z., Zhu, Y., Lin, H., Zhang, Z., Sun, Y., He, T., Mueller, J., Manmatha, R., Li, M., Smola, A.: ResNeSt: Split-Attention Networks (2020)