Dynamic Sparse Network for Time Series Classification: Learning What to "See"

Anonymous Author(s)
Affiliation
Address
email

Abstract

The receptive field (RF), which determines what hidden signals can be "seen" in a time series model, is critical to improve the performance for time series classification (TSC). However, the variation of signal scales across and within time series data, makes it challenging to decide proper RF sizes for TSC. In this paper, we propose a dynamic sparse network (DSN) with sparse connections for TSC, which can learn to cover various RF without cumbersome hyper-parameters tuning. The kernels in each sparse layer are sparse and can be explored under the constraint regions by dynamic sparse training, which makes it possible to reduce the resource cost. The experiment results show that the proposed DSN model can achieve the state-of-art performance on both univariate and multivariate TSC datasets with less than 50% computational cost compared with recently baseline methods, opening the path towards more accurate resource-aware methods for time series analyses. Our code is provided in the supplementary material, and it will be made available online.

1 Introduction

Time series classification (TSC) as an important research topic in the data mining communities [19], has a wide range of applications from health care [20, 12], and social security [51], to grid energy [34], and remote sensing [36]. In the last decade, neural networks, especially deep neural networks, have achieved competitive or even better performance than traditional TSC approaches (e.g. DTW [39], BOSS [40], COTE [1]) in many cases [49, 19].

However, how to discover and exploit the various scaled signals hidden in time series (TS) is still a significant challenge for TSC tasks [45, 31, 17]. One main reason lies in several variances, like sampling rate and record length, which are natural during time-series collection [40, 27, 4]. Furthermore, the amplitude offset, warping and occlusion of data points are unavoidable (see Figure 1). So determining the optimal scales for feature extraction is difficult but significant in the TSC task. One main solution is to cover as many receptive field (RF) scales as possible, so it does not ignore any useful signals from the time series inputs [45, 5, 6, 21].

Grounding on the motivation that multiple kernels in combination can capture discriminative patterns despite complex warping, Fawaz et al. [11] aimed to ensemble convolutional neural networks with a large number and variety of kernels. Tang et al. [46] have stacked several OS-CNN blocks, each of which consisted of a list of kernel sizes to cover all scales of RF and achieve better performance across several TSC tasks. In Multi-scale Convolutional Neural Network (MCNN) [3], a grid search was applied to find suitable kernel sizes. Without elaborate setting and fine-grid search, Dempster et al. [5] transformed time series by using random convolutional kernels and then trained a time series classifier. Even though the use of kernels with various sizes can help to extract latent hierarchical features of multiple resolutions, it tends to cause a huge computational problem and overfits the

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

datasets when a limited amount of training data is available [21]. To address this problem, dilated convolution [52] has been used to reduce model size while keeping the receptive field size. Franceschi et al. [13] presented an unsupervised learning model with convolutional kernels for time series feature transformation, in which the dilation factors of kernels increased exponentially layer by layer. To date, there is still an unresolved challenge with hyperparameter selections (e.g. kernel sizes and dilated factors) to capture patterns at different scales. This raises a foundational question: ***Can we design a scalable method for TSC that achieves a trade-off between the computation and performance without cumbersome hyperparameter tuning?***

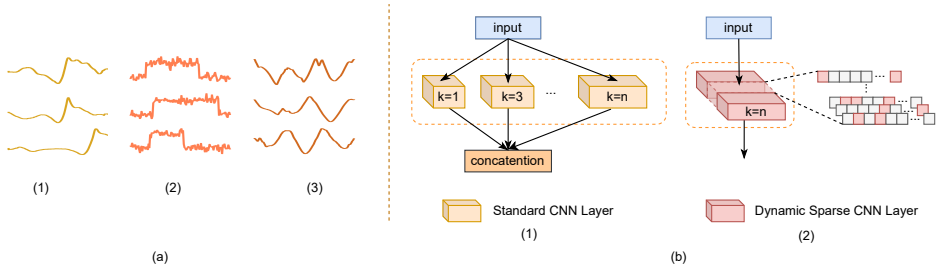

Figure 1: (a) Toy example illustrates the various time scales characteristic for TS data. Data in (1), (2) and (3) are sampled from three different TS datasets. The sequences with same color represent the same class. We can find that several sources of variances like phase, warping and offset are common in TS data. (b) The comparison between other methods [46, 21] (shown in b1) and our proposed DSN (shown in b2) for TSC. To cover various RF, DSN does not need to be configured with a series of kernel sizes k.

Inspired by sparse neural network models which can match the performance of dense counterparts with fewer connections [33, 30, 28, 53], we propose a dynamic sparse network (DSN) that is composed of CNN layers with large but dynamic sparse kernel for TSC tasks, which can automatically learn sparse connections in terms of covering various RF. The proposed DSN is trained by a dynamic sparse training (DST) strategy, which makes it possible to reduce the computational cost (e.g. floating-point operations (FLOPs)) for training and inference as well. Different from the traditional dynamic sparse training methods, which explore the connections in a layer-wise manner [33, 29, 10] and can hardly cover smaller RF, we propose a more fine-grained sparse training strategy for TSC. Specifically, the CNN kernels in each layer are split into several groups, each of which is sparse and can be explored under a constraint region during the course of sparse training. In this way, each layer can extract features in a more scalable and diverse way, and then by cascading, the model can cover more various scales of RF. To sum up, our main contributions are:

- We propose a novel dynamic sparse CNN layer with various effective RF, which can learn to "see" selectively and adaptively for more accurate TSC.
- By stacking the above dynamic sparse CNN layers, we introduce a novel DSN model natively for TSC, which is trained by DST to reduce computational and memory cost.
- To bridge gap between resource awareness and accuracy, we propose a more fine-grained training strategy for DSN, which can more easily cover diverse effective RF for TSC.
- We perform extensive empirical evaluation on both univariate and multivariate TSC tasks, showing that DSN achieves superior performance in terms of accuracy and resource cost.

2 Related Work

2.1 Time Series Classification

In the last decade, the success of deep learning encourages researchers to explore and extend its application on TSC tasks [24, 19, 49, 18, 25]. For univariate TSC, deep learning-based models attempt to directly transform raw time series data into low-dimensional feature representation via convolutional or recurrent networks [21, 38, 35]. For multivariate TSC, LSTM and attention layers are often stacked with CNN layers to extract complementary features from TS data [22, 54].

Recently, many works attempt to extract features in a more scalable manner as TS data is naturally composed of multiple signals on different scales [40, 4, 45]. One main solution is to configure

suitable kernels of various sizes such that they increase the probability to capture the proper scales [45, 46, 21]. Rocket-based methods [5, 6, 7] aim to use random kernels with several sizes and dilation factors to cover diverse RF for univariate TSC. Different from these works, our proposed dynamic sparse CNN layer can adaptively learn the various RF with deformable dilation factors and trade-off both computation and performance without cumbersome hyperparameter tuning.

2.2 Sparse Training

Recently, the lottery ticket hypothesis has shown the possibility of training a sparse sub-network to match the performance of the dense network with less computational cost [14]. Rather than iteratively pruning from a dense network [15, 55, 14], recent works try to find an initial mask with one-shot pruning based on gradient information during training [26, 48]. After pruning, the topology of the neural network will be fixed during training. However, this kind of models can hardly match the accuracy achieved by their dense counterparts [32, 48, 44].

Introduced as a new training paradigm before the lottery ticket hypothesis, DST starts from a sparse neural network and allows the sparse connectivity to evolve dynamically with a fixed number of parameters throughout the training [33, 10, 30, 53]. Nowadays, DST attracts increasing attentions from other research fields such as reinforcement learning [42] and continual learning [41], due to its potential of outperforming dense neural networks training [33, 53, 30]. Different from the traditional DST methods, our proposed DSN is trained with a fine-grained sparse training strategy rather than a traditional layer-wise manner to capture more diverse RF for TSC.

2.3 Adaptive Receptive Field

The RF, which can be adaptively changed during the training, has proven to be effective in many domains [47, 37, 50]. The adaptive RF can be usually captured by learning the optimal kernel sizes or the masks of kernels during the training stage [43, 16, 50]. However, neither the kernels nor the masks are sparse, which may cause the huge computational problem when larger RF needed. Different from these methods, our dynamic sparse CNN layers in DSN can be trained with DST, which can learn to capture variable RF with deformable dilation factors. What's more, the kernels are always sparse during training and inference, which makes it possible to reduce the computational cost.

3 The Proposed Model

3.1 Problem Definition

Definition 1. ***(Time Series Classification (TSC))****. Given a TS instance $\mathbf{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_n\} \in \mathbb{R}^{n \times m}$, where m denotes the number of variates and n denotes the number of time steps, TSC aims to accurately predict the class label $y \in \{1, \ldots, c\}$ from c classes. When m equals 1, TSC is univariate and otherwise it is multivariate.*

Definition 2. ***(Time Series (TS) Training Set)****. A training set $\mathcal{D} = \{(\mathbf{X}^{(1)}, y^{(1)}), \ldots, (\mathbf{X}^{(N)}, y^{(N)})\}$ consists of N time series instances, where $\mathbf{X}^{(i)} \in \mathbb{R}^{n \times m}$ could either be a univariate or multivariate time series instance with its corresponding label $y^{(i)} \in \{1, \ldots, c\}$.*

Note that, in our case, all instances have the same number of time steps in a TS dataset. Without the loss of generality, given a training set, we aim to train a CNN classifier with adaptive and various RF with a low resource cost (e.g. memory and computation) for the TSC task.

3.2 Dynamic Sparse CNN Layer with Adaptive Receptive Field

A straightforward strategy to cover various RF is to apply multi-sized kernels in each CNN layer, but it has several limitations. Firstly, TS instances from different TS datasets do not have the same length and cycles [1] in most cases, making it difficult to set a fixed kernel configuration for all the datasets even with prior knowledge. Secondly, obtaining a large receptive field commonly requires a large kernel or stacking more layers, which introduces more parameters and thus increases storage and computational costs.

[1] The term 'cycle' means that the data exhibit rising and falling that are not of a fixed frequency.

To tackle those challenges, the proposed dynamic sparse CNN layer possesses large but sparse kernels, which are learnable to capture adaptive RF. Specifically, given the input feature map $x^l \in \mathbb{R}^{c_{l-1} \times h \times w}$ in the lth layer (h equals 1 for univariate TSC and c_{l-1} is the number of input channels) , and kernel weights $\Theta^l \in \mathbb{R}^{c_{l-1} \times c_l \times 1 \times k}$ (k is the kernel size and c_l is the number of output channels), the convolution with stride 1 and padding in our proposed dynamic sparse CNN layer is formulated as

$$\mathbf{O}_j = \sum_{0 < i \leq c_{l-1}, i \in \mathbb{Z}} \left(\mathbf{I}^l(\Theta^l)_{i,j} \odot \Theta^l_{i,j} \right) \cdot x^l_i, \tag{1}$$

where $\mathbf{O}_j \in \mathbb{R}^{h \times w}$ denotes the output feature representation at the j-th output channel, $\mathbb{Z}$ denotes the set of integers, $\mathbf{I}^l(\cdot) : \mathbb{R}^{c_{l-1} \times c_l \times 1 \times k} \rightarrow \{0,1\}^{c_{l-1} \times c_l \times 1 \times k}$ is an indicator function, $\mathbf{I}^l(\Theta^l)_{i,j}$ indicates activated weights for $\Theta^l_{i,j} \in \mathbb{R}^{1 \times k}$ which is the kernel in the (i,j)-th channel, $\odot$ denotes the element-wise product, and $\cdot$ denotes the convolution operator. The indicator function $\mathbf{I}^l(\cdot)$, which is learned during the training of the proposed DSN, satisfies that $\|\mathbf{I}^l(\cdot)\|_0 \leq (1-S)N_l$ where $0 \leq S < 1$ is the sparsity ratio, $\|.\|_0$ denotes the L_0 norm and $N_l = c_{l-1} * c_l * 1 * k$. When $S > 0$, the kernel is sparse, and we can use a large k in the dynamic sparse CNN layer to obtain a large RF with reduced computational cost.

Remark 1. ***(Effective Receptive Field (eRF) size in the dynamic sparse CNN layer)****. In the standard CNN layer, the receptive field is defined as the input region that is used for generating every pixel feature. Thus, the RF size is equivalent to the kernel size. Differently, the eRF size of the proposed dynamic sparse CNN layer is smaller than the kernel size, when the first or last weight in a kernel is not activated, e.g., $\exists i \in \{1, \ldots, c_{l-1}\}$ and $j \in \{1, \ldots, c_l\}$, such that $I^l(\Theta^l)_{i,j,1,1} = 0$ or $I^l(\Theta^l)_{i,j,1,k} = 0$. As illustrated in Figure 2, we defined the eRF size $f^l_{i,j}$ of the kernel $\Theta^l_{i,j}$ as the distance between the first activated weight and last activated weight in the l-th CNN layer as*

$$f^l_{i,j} := \begin{cases} \max(Ind^l_{i,j}) - \min(Ind^l_{i,j}) + 1, & \text{if } Ind^l_{i,j} \neq \varnothing \\ 0, & \text{otherwise} \end{cases}, \tag{2}$$

where $Ind^l_{i,j}$ denotes the set of indices corresponding to non-zero weights in the kernel $\Theta^l_{i,j}$. Obviously we have $0 \leq f^l_{i,j} \leq k$.

The eRF size set of l-th dynamic sparse CNN layer is denoted by $\mathbb{F}^{(l)}$, which satisfies that $0 \leq \min(\mathbb{F}^{(l)})$ and $\max(\mathbb{F}^{(l)}) \leq k$. Take the case in the Figure 2 as an simple example for l-th layer, then $\mathbb{F}^{(l)} = \{0, 1, 2, 3, 4, 5, 6\}$. Each dynamic sparse CNN layer is likely to cover various eRF sizes ranging from 1 to k. When global information is expected, $\mathbf{I}^l(\cdot)$ could activate weights more dispersed to get a larger eRF and exploit the input features selectively. To capture the local context, a smaller eRF is expected, thus the activated weights tend to be concentrated. By taking $k = 5$ as an example, $\mathbf{I}^l(\Theta^l)_{i,j}$ may be [1, 0, 1, 0, 1] for global context, while it may be [0, 0, 1, 1, 0] for local context. In this way, the eRF can be adaptively adjusted.

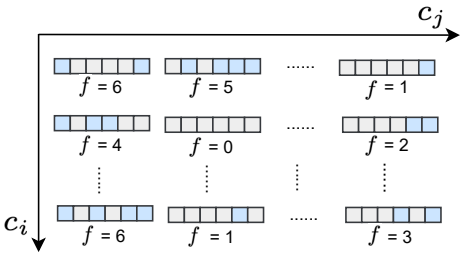

Figure 2: Illustrations for the eRF size. The weights with blue color are activated.

3.3 Architecture in DSN

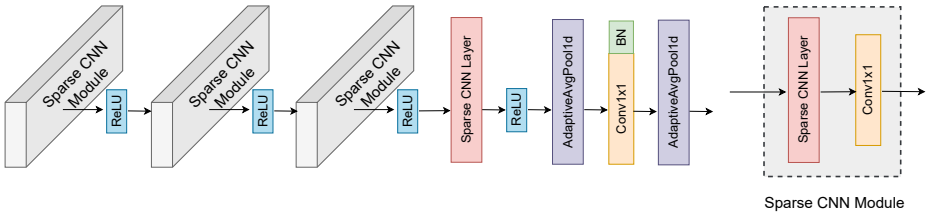

Figure 3: The proposed DSN model (left), which includes several sparse CNN modules (right) followed by a dynamic sparse CNN layer, two adaptive average pooling layers, and one 1×1 convolution layer.

The proposed DSN model consists of three sparse CNN modules, each of which composes of a dynamic sparse CNN layer and a 1×1 CNN layer. Following the stacked sparse CNN modules,

there is an additional dynamic sparse CNN layer, two adaptive average pooling layers, and one 1×1 convolution layer acting as the classifier in the DSN model. The overall architecture is shown in Figure 3.

The set of eRF sizes $\mathbb{S}^{(l)}$ in the lth sparse CNN module is equal to that in the dynamic sparse CNN layer, since the eRF size of 1×1 convolution is constantly equal to 1. $\mathbb{S}^{(l)}$ satisfies that $0 \leq \min(\mathbb{S}^{(l)})$ and $\max(\mathbb{S}^{(l)}) \leq k$. Then, the set of eRF sizes for the three successively stacked sparse CNN modules can be described by $\mathbb{RF}$ as

$$\mathbb{RF} = \left\{ \max(0, s^{(1)} + s^{(2)} + s^{(3)} - 2) \mid s^{(i)} \in \mathbb{S}^{(i)}, i \in \{1, 2, 3\} \right\}. \tag{3}$$

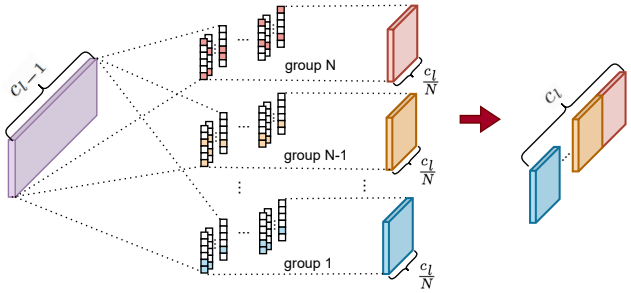

Figure 4: An illustration of the dynamic sparse CNN layer. In a dynamic sparse CNN layer, the kernels are split into N groups, each of which is with sparse connections learned by dynamic sparse training across specific constraint regions. The connections with color indicate non-zero values.

According to Eq. (3), we can see that stacking multiple sparse CNN modules can increase the size of $\mathbb{RF}$ linearly, where each dynamic sparse CNN layer increases by the size of $\mathbb{S}^{(l)}$. For simplicity, the kernel size in each dynamic sparse CNN layer is consistently set to k in our study. Then, $\mathbb{RF}$ satisfies that $\max(\mathbb{RF}) \leq 3k - 2$ and $0 \leq \min(\mathbb{RF})$. Thus, a large k in sparse CNN modules can increase the range of the eRF sizes for stacked sparse CNN modules.

3.4 Dynamic Sparse Training for DSN Model

In this section, we present the training strategy to discovery the weights required to be activated in order to ensure well-performing eRF. That is, we need to study how to update the indicator function $\mathbf{I}^l(\cdot)$ during the training of the proposed DSN.

We sparsely train the proposed DSN model from scratch to keep kernel sparse following the main idea of DST methods [33]. By design, during the training phase, the total number of activated weights must not exceed $N_l(1 - S)$. However, we observe that the tiny eRF is hard to capture by DST methods with a layer-wise exploration manner, namely activated weights are discovered layer by layer, especially when the sparsity ratio S is undersized (more analysis in Section 4.5). Motivated by this observation, the kernels in each dynamic sparse CNN layer are divided into different groups, whose corresponding exploration regions are of different sizes as shown in Figure 4. Contrary to the DST methods, in the DSN the exploration of activated weights is separately performed in different kernel groups, which is a more fine-grained strategy. Specifically, the kernel weights $\Theta^l \in \mathbb{R}^{c_{l-1} \times c_l \times 1 \times k}$ in the lth layer are split into N groups along the output channel, that is, $\Theta_1^l, \ldots, \Theta_N^l \in \mathbb{R}^{c_{l-1} \times \frac{c_l}{N} \times 1 \times k}$ with corresponding exploration regions $\mathcal{R}_1^l, \ldots, \mathcal{R}_N^l$. The exploration

Algorithm 1 Dynamic Sparse Training for DSN Model

Input: Dataset $\mathcal{D}$, Network f_Θ, Sparsity ratio S
Exploration regions in layer l: $\{\mathcal{R}_1^l, ..., \mathcal{R}_N^l\}$
Exploration schedule: T, ΔT, α, f_{decay}

$\theta_i^l \leftarrow$ initialize activated weights in Θ_i^l using S and $\mathcal{R}_i^l$
for $t = 1$ **to** T **do**
  Sample a batch $B_t \sim \mathcal{D}$
  $L_t = \sum_{i \sim B_t} L\left(\left(f_\theta\left(x_i\right), y_i\right)\right.$
  if $(t \bmod \Delta T) == 0$ **then**
    for each layer $l = 1$ **to** L **do**
      for each group $i = 1$ **to** N **do**
        $u = n(\mathcal{R}_i^l) f_{\text{decay}}\left(t; \alpha, T\right)(1 - S)$
        $\mathbb{I}_{\text{prune}} = \text{ArgTopK}\left(-\left|\theta_i^l\right|, u\right)$
        $\mathbb{I}_{\text{grow}} = \text{RandomK}\left(\mathcal{R}_i^l \backslash \theta_i^l, u\right)$
        $\mathbf{I}^l(.) \leftarrow$ Update $\mathbf{I}^l(.)$ using $\mathbb{I}_{\text{prune}}$ and $\mathbb{I}_{\text{grow}}$
        $\theta_i^l \leftarrow$ Update activated weights $\mathbf{I}^l(\Theta_i^l)\Theta_i^l$
      end for
    end for
  else
    $\theta = \theta - \alpha \nabla_\theta L_t$
  end if
end for

region $\mathcal{R}_i^l$ in the ith group of the lth layer is defined as the first $\frac{i \times k}{N}$ positions of each kernel in this group. By taking $k = 6$ and $N = 3$ for example, activated weights in the first group are only within the first two positions of each kernel in Θ_1^l, while they are within the whole kernel for the last group (as shown in Figure 4). In this way, the various eRF can be covered, and exploration space can be reduced (more details can be found in Appendix A.1) to improve exploration efficiency.

Given the weight exploration regions $\mathcal{R}_1^l, \ldots, \mathcal{R}_N^l$ and the sparsity ratio S, we train the proposed DSN model as shown in Algorithm 1. The activated weights are explored within the exploration regions and updated every Δ_t iterations. The fraction of updated weight decays over time according the function $f_{\text{decay}}(t; \alpha, T)$, which follows the cosine annealing [8] as

$$f_{decay}(t; \alpha, T) = \frac{\alpha}{2}\left(1 + \cos\left(\frac{t\pi}{T}\right)\right), \tag{4}$$

where α is the initial fraction of activated weights updated, t is the current training iteration, and T is the number of training iterations. Thus, during the tth iteration, the number of updated activated weights in the ith group of the lth layer is $n(\mathcal{R}_i^l) f_{\text{decay}}(t; \alpha, T)(1 - S)$, where $n(\mathcal{R}_i^l)$ is the number of weights can be explored in region R_i^l. During the update of activated weights, we firstly prune the activated weights determined by $\mathrm{ArgTopK}\left(-\left|\theta_i^l\right|, u\right)$ and then randomly grow new activated weights by $\mathrm{RandomK}\left(\mathcal{R}_i^l \backslash \theta_i^l, u\right)$, where $\mathrm{ArgTopK}(\mathbf{v}, u)$ gives the indices of top u elements with larger values in a vector $\mathbf{v}$, $\mathrm{RandomK}(\mathbf{v}, u)$ outputs the indices of random u elements in a vector $\mathbf{v}$, and $\mathcal{R}_i^l \backslash \theta_i^l$ denotes the weights within $\mathcal{R}_i^l$ but except θ_i^l.

The exploration for activated weights is straightforward. Firstly, pruning weights with small magnitudes is intuitive, because the contribution of weights with smaller magnitudes is insignificant or even negligible. Considering the pruning recoverability, we randomly regrow new weights with the same number as the pruned weights to achieve better activated weight exploration. In this way, the exploration of weights is dynamic and of plasticity.

4 Experiments

4.1 Datasets and Baselines

We evaluate the proposed method on both univariate and multivariate TSC datasets. The details of each kind of dataset are as follows.

- **Univariate TS dataset from UCR 85 archive [2]**: This archive consists of 85 univariate TS datasets, which are collected from various domains (e.g. health care and remote sensing), have distinguishable characteristics, and various levels of complexity. The number of instances in the training set varies from 16 to 8926, while their time step resolutions range from 24 to 2709.
- **Three multivariate TS datasets from UCI [9]**: The EEG2 dataset contains 1200 instances of 2 categories and 64 variates. The Human Activity Recognition (HAR) dataset consists of 10,299 instances with 6 categories, and the number of variates is 9. The Daily Sport dataset contains 9,120 instances of 19 categories, and its number of variates is 45. The processing of these dataset is the same as in [22].

Univariate TSC baselines: The performance of the following three baseline methods are reported, OS-CNN [45], InceptionTime [21], and ResNet[49], as they are widely used for univariate TSC. **Multivariate TSC baselines:** Similar with [54], the following three multivariate baselines are selected: OS-CNN, TapNet [54], MLSTM-FCN [22]. To avoid an unfair comparison, we let outside the ensemble methods. However, based on our best knowledge, our performance analysis accounts for most of state-of-the-art methods.

4.2 Experimental Settings and Implementation Details

For optimization, we use the Adam optimizer with an initial learning rate of 3×10^{-4}, cosine decayed to 10^{-4}. Our model is trained for 1,000 epochs with a mini-batch size of 16. As in [21, 22], the best model, which corresponds to the minimum training loss, is used to evaluate performance over the testing sets. Inspired by [21], the kernel sizes k in each dynamic sparse CNN layer are set to 39. The

number of kernel groups N in Algorithm 1 is set to 3, which helps to cover the small, medium and large eRF. Each setting is repeated for five times and the average results are reported.

4.3 Results and Analysis

We show the performance on both univariate and multivariate TSC benchmarks in Table 1-2. For univariate TSC, we can see that our method outperforms the baseline methods in most cases with a smaller number of parameters (e.g. Params [2]) and less computation cost (e.g. FLOPs) on UCR 85 archive datasets. For multivariate TSC, our proposed DSN method achieves better performance on EEG2 and HAR datasets, while for Daily Sport, MLSTM-FCN was with 0.46% more accurate. It is worth highlighting that the resource cost (i.e. Params and FLOPs) of our proposed DSN is much smaller than that of baseline methods. Due to the page limitation, the detailed results including test accuracy and resource cost on each dataset of UCR 85 archive can be found in Appendix C.

Table 1: Pairwise comparison of test accuracy (%) and mean resource cost (i.e. Params (K)↓ and FLOPs (M)↓) of our method and other univariate TS baseline methods on UCR 85 archive datasets. Note that all the accuracies in comparison are rounded to two decimal places.

Archive	Methods	Baseline wins	**DSN** wins	Tie	Params	FLOPs
UCR 85 archive	ResNet	29	**53**	3	479.20	803.38
	InceptionTime	**40**	**40**	5	389.34	325.49
	OS-CNN	33	**49**	3	262.14	200.12
	DSN (ours)	-	-	-	**126.22**	**104.88**

Table 2: Test accuracy (ACC(%)) and resource cost (i.e. Params (K)↓ and FLOPs (M)↓) of different models on three multivariate TS datasets from UCI.

	Daily Sport			EEG2			HAR		
Methods	ACC	Params	FLOPs	ACC	Params	FLOPs	ACC	Params	FLOPs
TapNet	99.56	1885.58	346.97	85.30	1369.21	780.19	93.33	1275.78	310.61
MLSTM-FCN	**99.65**	324.77	70.59	91.00	342.51	161.53	96.71	284.97	63.38
OS-CNN	99.30	317.05	78.01	93.03	320.34	162.04	96.37	280.39	70.98
DSN (ours)	99.19	**163.68**	**40.48**	**99.10**	**162.14**	**82.31**	**96.82**	**160.13**	**40.64**

4.4 Sensitivity Analysis

Effect of Sparsity Ratio: From Table 3, we can find out how the sparsity ratio of dynamic sparse CNN layer affects the final test accuracy and the resource cost in multivariate TS datasets. Here, we analyze the trade-off between test accuracy and the resource cost under various sparsity ratios (i.e. $S \in [50\%, 80\%, 90\%]$) and a dense DSN model. Note that the dense model is defined exactly as our DSN model with dense connections along entire kernels. We can see that it is not that the more parameters of the model, the better performance it will achieve. This is mainly because more parameters will lead to overfitting and the proper sparse ratio will easily cover the desired receptive field (more demonstrations can be seen in Appendix B.1). Taking EEG2 data set as an example, with the decrease of sparse rate, the performance on it drops sharply. We hypothesise that this data set has more expectations for features from small receptive fields (the validation can be found in Appendix B.2). According to the experiments, a 80% sparsity ratio reflects a good trade-off between accuracy and resource cost, which is the default setting for the dynamic sparse CNN layer in our DSN model.

Effect of Architecture: To show how the architecture of DSN affects the final performance, we adopt the critical difference diagram for a detailed comparison, which is usually used for evalution in TSC [45, 21]. From Figure 5, we can find that the DSN model with 4 dynamic sparse CNN layers and 177 output channels in each layer performs consistently better than the other structures. The proposed model with more output channels seems to be crucial for preserving the capacity of the network, while requires more parameters and FLOPs compared to other counterparts. This highlights

[2] The amount of non-zero parameters in the model as in [42, 23, 26].

Table 3: Test accuracy (ACC(%)) and resource cost (i.e. Params $(K)\downarrow$ and FLOPs $(M)\downarrow$) of the proposed DSN model with different sparsity ratios on three multivariate TS datasets from UCI.

	Daily Sport			EEG2			HAR		
Methods	ACC	Params	FLOPs	ACC	Params	FLOPs	ACC	Params	FLOPs
Dense	99.19	1059.65	262.10	89.60	1058.11	536.17	96.06	1056.10	267.57
$S = 50\%$	**99.28**	370.44	91.63	96.23	368.90	187.05	96.62	366.89	93.01
$S = 80\%$	99.19	163.68	40.48	**99.10**	162.14	82.31	**96.82**	160.13	40.64
$S = 90\%$	99.25	**94.76**	**23.44**	98.90	**93.22**	**46.64**	96.75	**91.21**	**23.18**

an interesting trade-off between accuracy and computational efficiency. According to the experiments, the DSN model with 4 dynamic sparse CNN layers and 141 output channels in each layer is the trade-off between the performance and resource cost, which is the default setting for our DSN model.

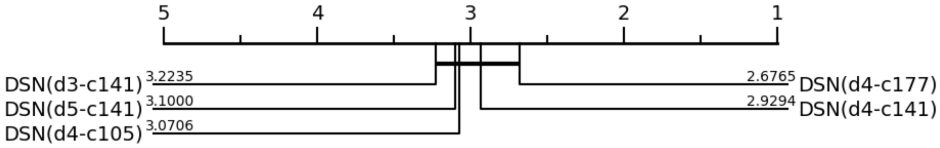

Figure 5: The critical difference diagram shows the average rank of the proposed DSN model with different architectures in UCR 85 archive. DSN(di-cj) represents the DSN model with i dynamic sparse CNN layers and j output channels in each layer. The smaller average rank corresponds to better performance of the model.

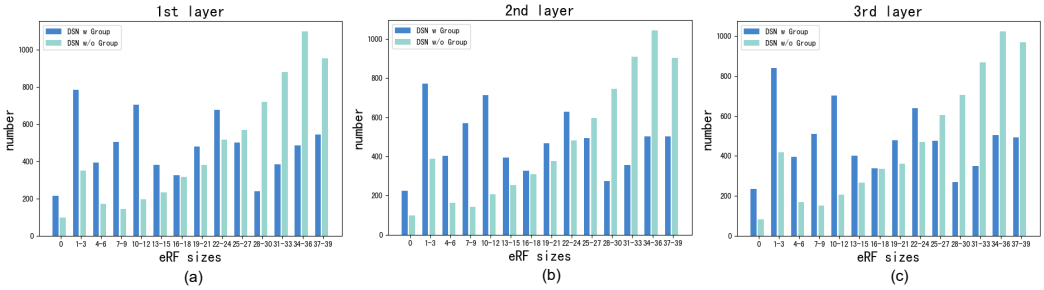

Figure 6: The distribution of eRF sizes in different sparse CNN layers between DSN models training with and without grouping on HAR dataset. The number on y-axis corresponds to the frequency of eRF sizes.

4.5 Ablation Study

Effect of Kernel Group: As shown in Figure 6, training DSN without kernel group hampers the capture of small RF, which can not guarantee satisfactory performance on the datasets such as EEG2 expecting more local information. With the decrease in sparsity ratio S, the phenomenon that large RF occupies the majority becomes more severe (more demonstrations see in Appendix B.1), which increases the performance gap between training with group and without it as shown in Table 4. Grouping the kernels can achieve various RF coverage and adorable accuracy for all datasets.

Table 4: Average accuracies (%) of the proposed DSN model affected by kernel grouping.

Sparsity ratio	Methods	Daily Sport	EEG2	HAR
$S = 60\%$	DSN w Group	**99.27**	**97.87**	96.53
	DSN w/o Group	99.23	96.77	**96.87**
$S = 70\%$	DSN w Group	99.20	**98.70**	**97.14**
	DSN w/o Group	**99.25**	97.27	96.52
$S = 80\%$	DSN w Group	99.19	**99.10**	**96.82**
	DSN w/o Group	**99.28**	98.77	96.79

Effect of Dynamic Sparse Training: Furthermore, we study the effect of the dynamic sparse training in our DSN method. We conduct ablation studies to compare DSN with its static variant (i.e. DSN^{fix}), which fixes its topology during training after activated weights initialization. In other words, the indicator function $\mathbf{I}^l(\cdot)$ in Eq. (1) will not update during the training process. According to the results

shown in Figure 7 (a), we can find that the DSN model mostly outperforms the DSN^{fix} on UCR 85 archive, indicating that what are learned by dynamic sparse training is the suitable activated weights together with their values.

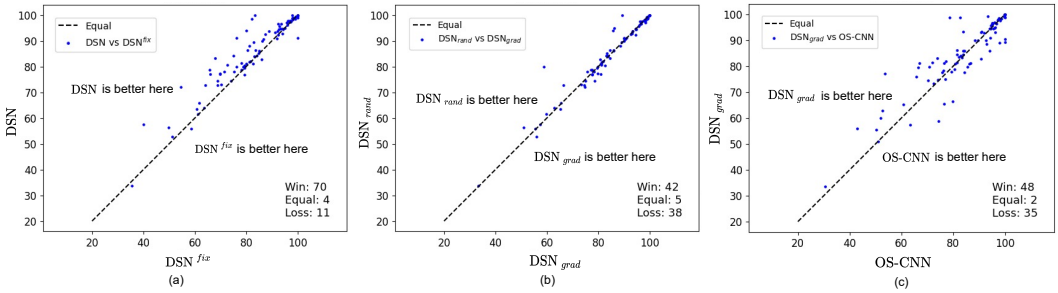

Figure 7: Accuracy plotting showing how the proposed DSN model is affected by the dynamic sparse training in UCR 85 archive.

4.6 Case Study on other DST Method

Likewise, the effectiveness of the proposed DSN method is also analyzed in the presence of other DST methods, such as RigL [10]. Rather than growing weights randomly (as in the line 9 of Algorithm 1), RigL grows them with the highest magnitude gradients. From Figure 7 (c), we can see that growing weights according to gradients (i.e. DSN$_{grad}$) can outperforms the OS-CNN method in most cases on UCR 85 archive. Furthermore, as shown in Figure 7 (b), we can find that DSN$_{grad}$ has a similar performance as growing weights randomly (i.e. DSN$_{rand}$). Note that, DSN$_{rand}$ is the main setting of the proposed method in this paper with the reason that we don't need to compute the full gradients at any moment in time. In summary, the proposed DSN with fine-grained sparse strategy can be easily combined with the existing DST techniques, showing the potential for further improvements by merging it with other advanced DST methods in the future.

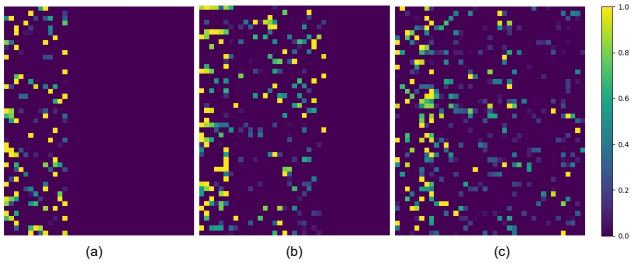

Figure 8: The activated weights in different kernel groups.

4.7 Visualization of Activated Weights

Figure 8 shows the normalized activated weights of the DSN model after sparse training on HAR dataset. (a), (b) and (c) correspond to the activated weights of the three kernel groups in the first dynamic sparse CNN layer, and each row of the figure represents one kernel. We can see that the weights in each kernel are sparse, and can be activated within the constraint region defined in Section 3.4.

5 Conclusions

This paper proposes dynamic sparse network (DSN) to cover diverse effective receptive field (eRF) for time series classification (TSC) without the cumbersome hyper-parameters tuning. The proposed DSN can achieve about $2\times$ reduction in memory and computational cost, while achieving superior performance in terms of accuracy on both univariate and multivariate TSC datasets. Furthermore, DSN explores the activated weights in a more fine-grained strategy, which can be easily merged with existing DST techniques, showing a potential for further improvements when combined with advanced technology. DSN provides a feasible solution to bridge gap between resource awareness and various eRF coverage for TSC, hoping to inspire other researches in different fields. It is worthwhile extending DSN to other research domains (e.g. time series forecasting and computer vision). Our DSN model has not yet been executed on real edge devices, and the performance in terms of accuracy and memory footprint on edge devices is not investigated. In the future, we hope to extend our work to other fields and decentralise training on edge devices.

References

[1] Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. Time-series classification with cote: the collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2522–2535, 2015.

[2] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, July 2015. `www.cs.ucr.edu/~eamonn/time_series_data/`.

[3] Zhicheng Cui, Wenlin Chen, and Yixin Chen. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995*, 2016.

[4] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

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

[6] Angus Dempster, François Petitjean, and Geoffrey I. Webb. ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Min. Knowl. Discov.*, 34(5):1454–1495, 2020.

[7] Angus Dempster, Daniel F. Schmidt, and Geoffrey I. Webb. Minirocket: A very fast (almost) deterministic transform for time series classification. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 248–257, 2021.

[8] Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *CoRR*, abs/1907.04840, 2019.

[9] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[10] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 2943–2952. PMLR, 2020.

[11] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep neural network ensembles for time series classification. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2019.

[12] Germain Forestier, François Petitjean, Pavel Senin, Fabien Despinoy, Arnaud Huaulmé, Hassan Ismail Fawaz, Jonathan Weber, Lhassane Idoumghar, Pierre-Alain Muller, and Pierre Jannin. Surgical motion analysis using discriminative interpretable patterns. *Artificial intelligence in medicine*, 91:3–11, 2018.

[13] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. *Advances in neural information processing systems*, 32, 2019.

[14] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

[15] Sharath Girish, Shishira R. Maiya, Kamal Gupta, Hao Chen, Larry S. Davis, and Abhinav Shrivastava. The lottery ticket hypothesis for object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 762–771, 2021.

[16] Shizhong Han, Zibo Meng, Zhiyuan Li, James O'Reilly, Jie Cai, Xiaofeng Wang, and Yan Tong. Optimizing filter size in convolutional neural networks for facial action unit recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, pages 5070–5078, 2018.

[17] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony J. Bagnall. Classification of time series by shapelet transformation. *Data Min. Knowl. Discov.*, 28(4):851–881, 2014.

[18] Seth H Huang, Xu Lingjie, and Jiang Congwei. Residual attention net for superior cross-domain time sequence modeling. *arXiv preprint arXiv:2001.04077*, 2020.

[19] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

[20] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, François Petitjean, Lhassane Idoumghar, and Pierre-Alain Muller. Automatic alignment of surgical videos using kinematic data. In *Conference on Artificial Intelligence in Medicine in Europe*, pages 104–113. Springer, 2019.

[21] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, 2020.

[22] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. Multivariate lstm-fcns for time series classification. *Neural Networks*, 116:237–245, 2019.

[23] Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham M. Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learnable sparsity. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, volume 119, pages 5544–5555, 2020.

[24] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

[25] Dongha Lee, Seonghyeon Lee, and Hwanjo Yu. Learnable dynamic temporal pooling for time series classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8288–8296, 2021.

[26] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: single-shot network pruning based on connection sensitivity. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[27] Jason Lines, Luke M Davis, Jon Hills, and Anthony Bagnall. A shapelet transform for time series classification. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 289–297, 2012.

[28] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Zahra Atashgahi, Lu Yin, Huanyu Kou, Li Shen, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. Sparse training via boosting pruning plasticity with neuroregeneration. *Advances in Neural Information Processing Systems*, 34, 2021.

[29] Shiwei Liu, Decebal Constantin Mocanu, Yulong Pei, and Mykola Pechenizkiy. Selfish sparse RNN training. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML*, volume 139, pages 6893–6904. PMLR, 2021.

[30] Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. In *International Conference on Machine Learning*, pages 6989–7000. PMLR, 2021.

[31] Benjamin Lucas, Ahmed Shifaz, Charlotte Pelletier, Lachlan O'Neill, Nayyar Abbas Zaidi, Bart Goethals, François Petitjean, and Geoffrey I. Webb. Proximity forest: an effective and scalable distance-based classifier for time series. *Data Min. Knowl. Discov.*, 33(3):607–635, 2019.

[32] Decebal Constantin Mocanu, Elena Mocanu, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. A topological insight into restricted boltzmann machines. *Machine Learning*, 104(2):243–270, 2016.

[33] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018.

[34] Elena Mocanu, Decebal Constantin Mocanu, Phuong H Nguyen, Antonio Liotta, Michael E Webber, Madeleine Gibescu, and Johannes G Slootweg. On-line building energy optimization using deep reinforcement learning. *IEEE transactions on smart grid*, 10(4):3698–3708, 2018.

[35] Kewei Ouyang, Yi Hou, Shilin Zhou, and Ye Zhang. Convolutional neural network with an elastic matching mechanism for time series classification. *Algorithms*, 14(7):192, 2021.

[36] Charlotte Pelletier, Geoffrey I Webb, and François Petitjean. Temporal convolutional neural network for the classification of satellite image time series. *Remote Sensing*, 11(5):523, 2019.

[37] Silvia L. Pintea, Nergis Tomen, Stanley F. Goes, Marco Loog, and Jan C. van Gemert. Resolution learning in deep convolutional networks using scale-space theory. *IEEE Trans. Image Process.*, 30:8342–8353, 2021.

[38] Elahe Rahimian, Soheil Zabihi, Seyed Farokh Atashzar, Amir Asif, and Arash Mohammadi. Xceptiontime: Independent time-window xceptiontime architecture for hand gesture classification. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1304–1308, 2020.

[39] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 262–270, 2012.

[40] Patrick Schäfer. The boss is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29(6):1505–1530, 2015.

[41] Jonathan Schwarz, Siddhant M. Jayakumar, Razvan Pascanu, Peter E. Latham, and Yee Teh. Powerpropagation: A sparsity inducing weight reparameterisation. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*, pages 28889–28903, 2021.

[42] Ghada Sokar, Elena Mocanu, Decebal Constantin Mocanu, Mykola Pechenizkiy, and Peterk Stone. Dynamic sparse training for deep reinforcement learning. In *31st International Joint Conference on Artificial Intelligence and the 23rd European Conference on Artificial Intelligence*, 2022.

[43] Domen Tabernik, Matej Kristan, and Ales Leonardis. Spatially-adaptive filter units for compact and efficient deep neural networks. *Int. J. Comput. Vis.*, 128(8):2049–2067, 2020.

[44] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in Neural Information Processing Systems*, 33:6377–6389, 2020.

[45] Wensi Tang, Guodong Long, Lu Liu, Tianyi Zhou, Michael Blumenstein, and Jing Jiang. Omni-scale cnns: a simple and effective kernel size configuration for time series classification. In *International Conference on Learning Representations*, 2022.

[46] Wensi Tang, Guodong Long, Lu Liu, Tianyi Zhou, Jing Jiang, and Michael Blumenstein. Rethinking 1d-cnn for time series classification: A stronger baseline. *arXiv preprint arXiv:2002.10061*, 2020.

[47] Nergis Tomen, Silvia-Laura Pintea, and Jan van Gemert. Deep continuous networks. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, volume 139, pages 10324–10335, 2021.

[48] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2019.

[49] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.

[50] Zhitong Xiong, Yuan Yuan, Nianhui Guo, and Qi Wang. Variational context-deformable convnets for indoor scene parsing. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, pages 3991–4001. Computer Vision Foundation / IEEE, 2020.

[51] Fei Yi, Zhiwen Yu, Fuzhen Zhuang, Xiao Zhang, and Hui Xiong. An integrated model for crime prediction using temporal and spatial factors. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 1386–1391. IEEE, 2018.

[52] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*, May 2016.

[53] Geng Yuan, Xiaolong Ma, Wei Niu, Zhengang Li, Zhenglun Kong, Ning Liu, Yifan Gong, Zheng Zhan, Chaoyang He, Qing Jin, et al. Mest: Accurate and fast memory-economic sparse training framework on the edge. *Advances in Neural Information Processing Systems*, 34, 2021.

[54] Xuchao Zhang, Yifeng Gao, Jessica Lin, and Chang-Tien Lu. Tapnet: Multivariate time series classification with attentional prototypical network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6845–6852, 2020.

[55] Zhenyu Zhang, Xuxi Chen, Tianlong Chen, and Zhangyang Wang. Efficient lottery ticket finding: Less data is more. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 12380–12390. PMLR, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.2
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix D
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]