Neurocomputing 512 (2022) 15–24

Contents lists available at ScienceDirect

Neurocomputing

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

One-shot pruning of gated recurrent unit neural network by sensitivity for time-series prediction

Hong Tang [a], Xiangzheng Ling [a,*], Liangzhi Li [b], Liyan Xiong [a], Yu Yao [c,d], Xiaohui Huang [a]

[a] School of Information Engineering, East China Jiaotong University, Nanchang 330013, China
[b] Institute of Data Science, Osaka University, Osaka 565-0871, Japan
[c] School of Electrical and Electronic Engineering, North China Electric Power University, Beijing 102206, China
[d] Guangdong Power Grid Co., Ltd., Guangzhou Power Supply Bureau in Guangzhou, China

ARTICLE INFO

Article history:
Received 31 October 2021
Revised 5 June 2022
Accepted 4 September 2022
Available online 13 September 2022

Keywords:
Deep learning
Gated recurrent units (GRU)
Pruning
Time-series

ABSTRACT

Although deep learning models have been successfully adopted in many applications, they are facing challenges to be deployed on energy-limited devices (*e.g.*, some mobile devices, *etc.*) due to their high computation complexity. In this paper, we focus on reducing the costs of Gated Recurrent Units (GRUs) for time-series prediction tasks and we propose a new pruning method that can recognize and remove the neural connections that have little influence on the network loss, using a controllable threshold on the absolute value of the pre-trained GRU weights. This is different from existing approaches which usually try to find and preserve the connections with large weight values. We further propose a sparse-connection GRU model (SCGRU) that only needs a one-time pruning (with fine-tuning), rather than using multiple prune-retrain cycles. A large number of experimental results demonstrate that the proposed method is able to largely reduce the storage and computation costs while achieving the state-of-arts performance in two datasets. Code is available (https://github.com/imLingo/SCGRU).

1. Introduction

Deep learning has boosted a lot of artificial intelligence applications. However, deep learning models still follow a trend of "the bigger, the better", which may become the obstacle for their deployments in practical applications, especially on the end nodes that are with limited computation capability. A typical example is an early-warning system of smart grid, where an accurate data prediction on edge terminals is preferred, which, however, is very difficult due to the high computational complexity of big time-series models [1]. Excessive feature representation and parameterization in the neural network model will bring a heavy computation and storage burden to the end platform [2]. In addition, although over parameterization is beneficial for network performance optimization, it may be not necessary for accurate prediction [3]. Therefore, it is important to find ways to decrease the costs of time-series models while keeping the model performance, to enable them for the devices with limited resources.

Compared with traditional time-series prediction methods [4,5], Recurrent Neural Networks (RNNs) [6], such as Long Short-Term Memory networks (LSTMs) [7] and Gated Recurrent Units (GRUs) [8], can better formulate the non-linear relationship and long-range dependencies among the time-series data. In addition, GRU has a simpler structure and a comparable performance to LSTM, therefore, GRU is more suitable for resource-limited devices. To further decrease the costs of GRU models, several different approaches have emerged [9–11], we can get sparse models, which have similar performance with the raw models, by decreasing the sparse block, hidden states, and biases of the gate units like the update gate and reset gate. However, its pruning rate is not so high and the pruned models struggle in dealing with complex tasks. Network connection-based pruning methods [12,13] can achieve a better pruning rate. However, it is debatable whether pruning connections depend on their weight values only can lead to the best pruning performance. Also, a lot of time-consuming "prune-retrain" loops exist in their training process, which results in a long training process and may be avoidable. Therefore, the objective of this work is to design a better pruning method that can achieve a high pruning rate while keeping a satisfactory prediction performance, also, without an expensive training process.

For this purpose, this paper designs a new pruning method and, based on that, proposes a new pruned GRU model named Sparse-

* Corresponding author.
E-mail addresses: tang.hong@mail.scut.edu.cn (H. Tang), imlingo@163.com (X. Ling), liliangzhi@ieee.org (L. Li), 445935939@qq.com (L. Xiong), 10215043@qq.com (Y. Yao), hxh016@hotmail.com (X. Huang).

https://doi.org/10.1016/j.neucom.2022.09.026
0925-2312/

Connection Gated Recurrent Units (SCGRU). SCGRU uses a controllable pruning threshold, which is based on the absolute value of the weight, to identify and prune the neural connections that have little effect on the gradient in the back-propagation process. This is different from existing pruning methods which usually assume that weights with larger absolute values play an important role [14,13]. In fact, some regions in the parameter matrix may have more influence on the prediction results [15]. Therefore, contrary to the strategy of "randomly" retaining neural connections or using "prune-retrain" cycles to retain neural connections with large weights, our method is designed to prune neural connections that do not affect the model loss, which only needs to be executed once during the whole training process.

The contributions of this paper are twofold:

- We propose a new pruning method that retains important connections and can achieve a high pruning rate while preserving good prediction accuracy.
- We show a new pruning pipeline that needs no "prune-retrain" cycles, which significantly reduces the training costs.

In addition, we test the SCGRU model through experiments on two actual time-series prediction datasets, which demonstrates its performance.

2. Related works

2.1. Compression methods of deep neural networks

The great success of Deep Neural Networks (DNNs) [16–18] has inspired people to explore lightweight models. Therefore, many classical compression methods based on the DNNs architectures have been proposed [19,20,14,21–23]. According to whether the network structure changes after compression as a reference, the current network compression methods can be roughly divided into three categories [13]:

1) Design a substitute for the original structure by changing the structure of the original network to approximate the effect of the original network [24,21,25–27]. Molchanov et al. [25] uses Taylor expansion to approximate the contribution of each channel over the final loss and pruned the convolutional filters. Li *et al.* [21] removes whole filters and their connected feature maps that had a small effect on the output accuracy to reduce computational cost. Yu *et al.* [26] proposes a global pruning method that uses information gain to quantify the effect of filters on the network output and pruned filters accordingly. Lin *et al.* [24] use global average pools instead of full connectivity layers to reduce the memory requirements of the original network. However, Szegedy *et al.* [28] point out that methods to change the structure of the network apply only to specific tasks, and that such methods can easily undermine the generalization of the original network, making it difficult to conduct transfer learning.

2) Delete redundant connections or some neurons based on the significance criterion to speed up the training process [19,29–33]. Karnin et al. [30] set the significance standard, based on the weight loss sensitivity, of the redundant weight in the network. In addition, Han *et al.* [14] used a weight threshold to prune the gradient of backpropagation, and Guo *et al.* [22] used a dynamic pruning strategy to flexibly choose weights. Guo *et al.* [32] used the backpropagation process in the stochastic and nonstationary environment to find and gradually prune the weights of weak connections. Recently, SNiP proposed by Lee *et al.* [33] introduces a saliency criterion based on the connection sensitivity to identify the structurally important connections in the network for a given task, and prunes the network before pre-training, with almost no precision loss.

3) Reduce network storage requirements through weight quantification without changing the network structure [34–36]. Hinton et al. [34] proposed a soft weight sharing strategy to cluster the weights so that the weights in the same group have similar values, and obtain the performance of a regularized network. Ullrich *et al.* [3] used a soft weight sharing strategy to fit a Gaussian mixture prior model on the weights, compress the weights to k-clusters, and finally reduce the bit size of each weight in memory. HashedNets proposed by Chen *et al.* [36] used a low-cost hash function to randomly group connection weights into hash buckets, sharing the same parameter value for all connections in the same bucket, but this requires secondary parameters to record the group membership of each weight.

2.2. Compression methods of RNNs

As the amount of data and the complexity of tasks increase, the size of the recurrent neural network also increases. To effectively deploy RNNs on mobile devices [37], Narang et al. [38,37] designed a magnitude-based pruning strategy that can largely reduce the network size during the training process and can increase the calculation speed by 2–7 times. Dai *et al.*[39] employed grow-and-prune (GP) training to iteratively adjust the hidden layers through gradient-based growth and magnitude-based pruning of connections to ensure LSTM compactness. Wen *et al.* [11] proposed Intrinsic Sparse Structures (ISS) within LSTMs, which are used to prune the gating unit of the k-th hidden state of LSTM to reduce memory requirements. In the task of processing text data, Dey *et al.* [10] explored three GRU variants that reduce update gate and reset gate parameters and proved that GRU variants that only retain 33% of the original parameters can obtain the same performance as the original GRU. Zhang et al. [40] found that the pruning performance of SNiP on GRU is worse than the random pruning method. See *et al.* [41] combined the magnitude-based pruning method with a special RNN structure and got pruned model with good prediction performance. However, this type of method can only prune 80% of the neural connections in the end.

One of the most impressive pruning methods is the RC-LSTM [12], which retains only 1% of the neural connections in the original network and increases computing speed by 30% with a small precision drop. However, RCLSTM only made random connections on the parameter matrix based on the idea of random graph theory to reduce computational costs, which does not retain the connections that are of big importance. Sensitivity is based on the neural network parameter matrix and used to choose targeted redundant neural connections. Xiong et al. [13] designed sensitivity thresholds based on parameter standard deviations to prune the weights that have little effect on model performance, and proposed SCLSTM, which outperforms RCSLTM in the same datasets. However, we find that the sensitivity threshold set by Xiong *et al.* [13] do not pay attention to the negative weight that also affects the result, and they need multiple "prune-retrain" cycles during the training process. Our proposed method tries to retain a part of the connections that play an important role in back-propagation, and set a sensitivity threshold in the set of the absolute value of parameter weights, which follows an approximate chi-square distribution, that is convenient to better eliminate the standard deviation fluctuation caused by outliers, and more accurately choose redundant neural connections, solving the problem that negative weights are usually ignored [13]. In addition, for better comparison with the state-of-the-art methods like [12,13], we extend their works into GRU, which are called RCGRU and X-GRU in the experiments section.

3. A sparse connected GRU architectures

The objective is to design a selective single pruning sche-me on densely connected GRUs to handle redundant connections in a given task, and ultimately, to reduce the memory consumption while ensuring their performance. To this end, different from recursive "prune-retrain" cycles, we design a new pruning strategy that only includes one pruning process. The strategy consists of pre-training, selective pruning, and fine-tuning, as shown in Fig. 1. Fig. 2 shows the pruning process of our proposed sparse GRU model, which we will introduce in detail in the following sections.

3.1. Preparation and notation

For actual tasks, people usually need to adopt a network with appropriate parameter size to obtain good prediction performance while having a satisfactory generalization ability. However, how to determine the size of the adopted neural network is still a problem to be addressed. Actually, when learning a specific task, the neural network is usually over-parameterized. As early as 1993, Reed et al. [42] found that learning tasks can be completed through a smaller network and the generalization ability can be improved. To this end, our goal is to learn a sparse GRU network while maintaining the performance of the standard reference network. We first regard GRU network pruning as an optimization problem.

Given a dataset $\mathscr{D} = \{x^{(t)}\}_{t=1}^{n}$, and a sliding window size $s \in \mathbb{N}^+$ (s.t. $s < n$), where $x^{(t)} \in \mathbb{R}^{1\times 1}$. Through the slider processing, a new input dimension $m = n - s + 1$ is obtained, and a time-series dataset $\mathscr{D}_T = \left\{X^{(t)}, y^{(t)}\right\}_{t=1}^{m}$ is formed, where $X^{(t)} \in \mathbb{R}^{1\times(s-1)}, y^{(t)} \in \mathbb{R}^{1\times 1}$.

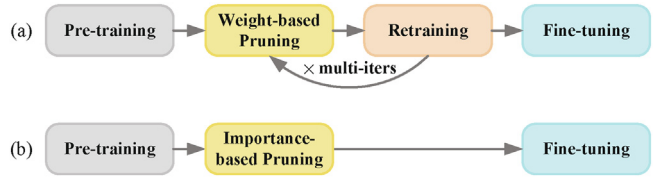


Fig. 1. The pipeline comparison between (a) a typical pruning method [14] and (b) the .proposed method.

There are two specific gate structures in GRU: update gate $z^{(t)}$ and reset gate $r^{(t)}$,

$$z^{(t)} = \sigma\left(W_z\left[x^{(t)}, h^{(t-1)}\right] + b_z\right), \tag{1}$$

$$r^{(t)} = \sigma\left(W_r\left[x^{(t)}, h^{(t-1)}\right] + b_r\right), \tag{2}$$

in which σ is the activation function (Sigmoid function used in this paper), $h^{(t-1)}$ is the hidden state at time $t-1$, W_z and W_r are both $p \times \frac{q}{3}$ dimensional parameter matrices, $0 < p < m, 0 < q < s$.

As a variant of RNN, the components of GRU also inherit the following structure,

$$\widetilde{h}^{(t)} = g\left(W_h\left[x^{(t)}, h^{(t-1)} \cdot r^{(t)}\right] + b_h\right), \tag{3}$$

$$h^{(t)} = \left(1 - z^{(t)}\right) \cdot h^{(t-1)} + z^{(t)} \cdot \widetilde{h}^{(t)}, \tag{4}$$

where g is the activation function, such as the hyperbolic tangent function or the rectified linear unit (ReLU) [43], $W_h \in \mathbb{R}^{p\times\frac{q}{3}}$.

To facilitate understanding and calculation, GRU is defined as a mapping relationship completed by iterative calculation:

$$\left(h^{(t)}, \hat{y}^{(t)}\right) = GRU\left(W; h^{(t-1)}, X^{(t)}\right), \tag{5}$$

where $W \in \mathbb{R}^{p\times q}$ is the consecutive combination of the three linear operation parameter matrices (*i.e.*, the W_z, W_r and W_h) in the GRU model, and $\hat{y}^{(t)}$ is the predicted value at time t.

3.2. GRU parameter pruning: architectural perspective

With the parameter matrix W after pre-training, we can get the sensitivity threshold εas

$$\varepsilon = \lambda \times \sqrt{\frac{\sum_{i,j}^{p,q}\left||W_{ij}| - \overline{|W|}\right|^2}{p \cdot q}}, \tag{6}$$

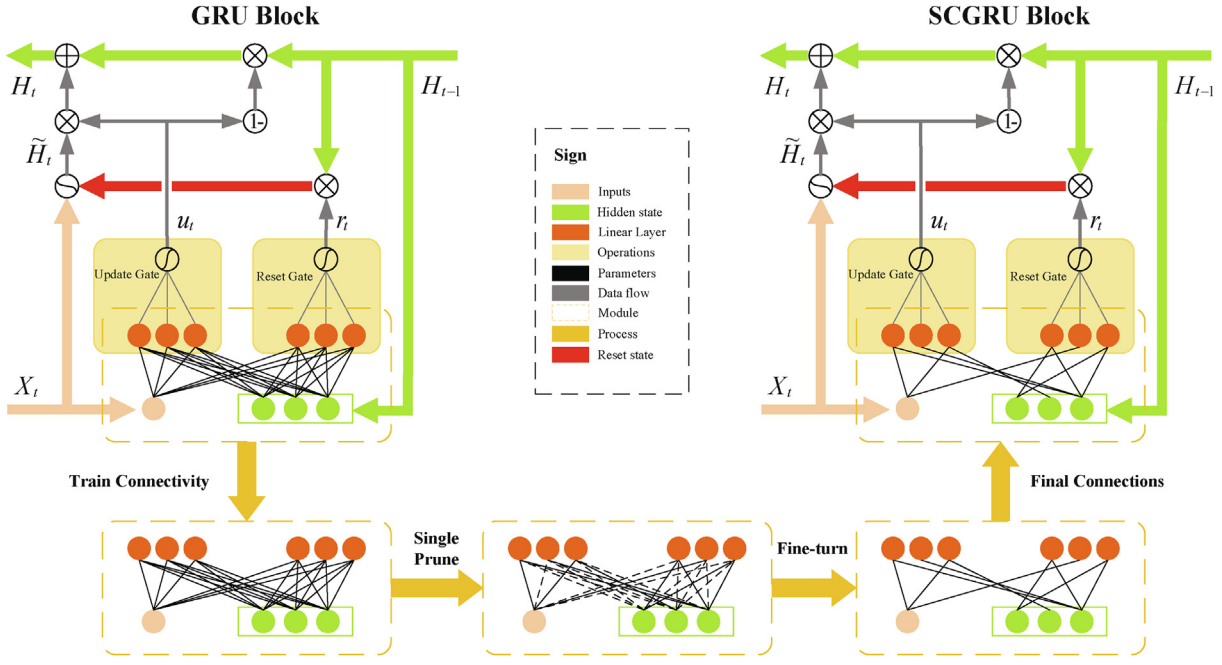


Fig. 2. The pruning process of neural connections from standard GRU to SCGRU.

where λ is a parameter to control the threshold ε, and $\overline{|W|} \in \mathbb{R}^{1\times 1}$ is the mean value after the absolute value of the weight. The sensitivity threshold ε is used to compare with each weight in the matrix W, finding the targeted redundant parameters, and then guides the parameter selection in the matrix W in the forward propagation process. Given a desired sparsity level k (*i.e.*, the proportion of non-zero weights), which is an upper bound on the true pruning rate, the model constraints are generated as

$$k \geqslant \frac{\sum_{i,j}^{p,q}(|W_{ij}| < \varepsilon)}{p \cdot q} = \frac{\||W| < \varepsilon\|_0}{p \cdot q}. \tag{7}$$

We introduce auxiliary indicator variables $c \in \{0,1\}^p$ to ensure that the pruned connections are no longer revived through back-propagation. Corresponds to parameter matrix W, mask matrix $C = \left\{c_1^\top, c_2^\top, \cdots, c_q^\top\right\}$ that value obeys:

$$C_{ij} = \begin{cases} 0, & |W_{ij}| < \varepsilon; \\ 1, & otherwise. \end{cases} \tag{8}$$

where C_{ij} is the element in the i-th row and j-th column of the mask matrix C that is mainly used to record the coordinates of redundant connections. To ensure the effectiveness of selective single pruning, the optimization problem is transformed into:

$$\begin{aligned} &\min L(W; \mathscr{D}_T) = \min \tfrac{1}{pq} \sum_{i,j}^{p,q} \mathscr{L}(C \odot W; (X_{ij}, y_i)), \\ &s.t. \quad k \geqslant \||W| < \varepsilon\|_0 = \|C\|_0, \end{aligned} \tag{9}$$

where $\mathscr{L}$ is the loss function. The mask matrix C is the key factor to ensure a single pruning, which not only remove the weight of the corresponding position in the forward propagation process, but also prevent the discarded weight from repeating during the training phase.

Finally, the overview and implementation steps of the entire optimization process are shown in Algorithm 1.

Algorithm 1 One-shot Pruning GRU

Require: Dataset $\mathscr{D}_T$, Parameters $W\{W_g, W_o\}$, sensitivity λ, sparsity level k

Ensure: $\lambda > 0$, $p = row(W)$, $q = col(W)$, $k \geqslant \frac{\|W\prime\|_0}{p \cdot q}$

$W \Leftarrow$Pre-training

for all $W = \{W_g, W_o\}$ **do**

$\theta \Leftarrow abs(W)$;

$\bar{\varepsilon} \Leftarrow mean\left\{\sum_{i,j}(abs(W) - mean(\theta))\right\}$

Label : $\varepsilon \Leftarrow \lambda \cdot sqrt(\bar{\varepsilon})$

for all $\{\theta_{ij} | i = 1, 2, \cdots p; j = 1, 2, \cdots q\}$ **do**

$C_{ij} \Leftarrow \theta_{ij} < \varepsilon \quad ? \quad 0 : 1$

end for

$W\prime \Leftarrow W \odot C$

end for

4. Experiments results

4.1. Datasets description

We use two time-series datasets, including the Local Area Network (LAN) traffic dataset from GÉANT [44] and the power load dataset of a province in southern China, to evaluate our method's performance.

The power load dataset comes from a power grid and is recorded every 5 min from January 2014 to June 2016. The data unit is MW. There are 257,184 items in the original dataset, and each item is composed of record time and power load value. As shown in Fig. 3, these records reflect the annual periodicity of the electrical load.

The LAN traffic dataset is the actual traffic data captured from the links in the GÉANT (*i.e.* a pan-European data source for research and education) backbone network. 23 bandwidth points were sampled every 15 min for 4 months, and the unit of data point is Kbps. The dataset is composed of a flow matrix constructed using the complete interior gateway protocol (IGP) routing information, and is recorded in extensible markup language (XML) form, where each piece of data can be mapped into a flow matrix with a size of 23×23. For comparison, we selected 10,772 traffic data points therein. The detailed information of the LAN traffic datasets is shown in Table 1.

4.2. Experimental setup

In the power load forecasting experiment, based on the needs of real-time traffic forecasting, we introduce a sliding window and set the input flow sequence length to 100. We normalize the original data as the logarithm of base 10 to make the neural network model converge faster in the training phase. Meanwhile, the hidden units of all GRU-based neural networks (except GRU-35 [8]) are uniformly set to 350, the batch size is set to 128, the random seed is set to 42, and the ratio of the number of training samples to the number of test samples is set to 9:1. The adaptive moment estimation (Adam) [45] optimizer is used in the training process, the learning rate is set to 0.001, and the weight decay is 1e-5. The Root Mean Square Error (RMSE) is used to measure the difference between the predicted value and the actual value. We compare the proposed method with RCGRU[12], Xiong's approach (called X-GRU for short) [13], GVGRUs [10], Support Vector Regression (SVR) [46], and Feed Forward Neural Networks (FFNNs) [47].

Table 1
Detaile information of the LAN traffic datasets.

Properties	Content
Bandwidth Points	23
Time Span	2005/01/01–2005/04/29
Time Interval	15 min
Data Unit	Kbps
Daily Range	All day

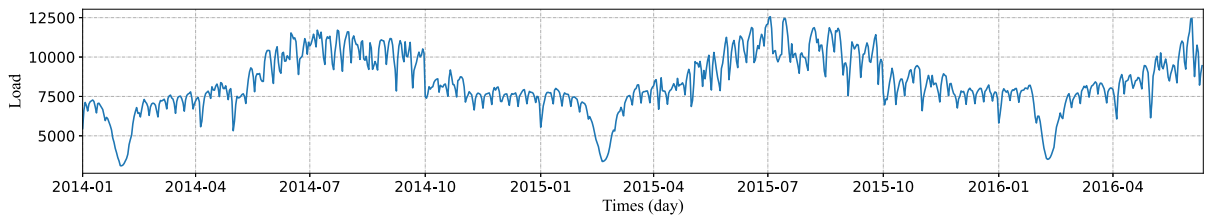


Fig. 3. The visualization of the power load dataset, which is recorded every 5 min from January 2014 to June 2016.

Among them, the input dimension of FFNN is set to 100, three hidden layers are defined, and the number of neurons in each hidden layer is set to 50. SVR selects RBF as the kernel, the tolerance of the stopping criterion is 0.001, and the number of input features is set to 100.

On the other hand, the parameter settings in the LAN traffic prediction experiment are basically the same as above. The difference is that the batch size is set to 32, the weight decay is 9e-5, FFNN sets two hidden layers with a dimension of 50, and the Autoregressive Integrated Moving Average (ARIMA) [48] model is added to the comparison. In ARIMA, the autoregressive term is set to 5, the number of non-seasonal differences is 1, and the moving average is 0.

In addition, we compare the prediction performance of three neural connection-based sparse models, namely RCGRU, X-GRU, and SCGRU, with and without fine-tuning.

4.3. Results on the power load dataset

Fig. 4(c) shows the parameter sizes of the models used in the power load forecasting task. The orange in the figure indicates the number of pruned parameters, and the blue indicates the number of parameters retained. It can be seen from Fig. 4(c) that compared with the original model, GVGRU1 only prune out a small number of parameters, and the parameters of GVGRU2 and GVGRU3 are about 1/3 of the original. The parameters of RCGRU, X-GRU, and SCGRU are of the same order of magnitude. For subsequent experiments, the parameters of FFNNs and GRU-35 are also set to be the same as SCGRU.

Fig. 4(b) shows the convergence performance comparison of SCGRU with traditional models SVR, FFNNs, and GRU-based sparse models (*i.e.* GVGRUs, RCGRU, and X-GRU) on the power load dataset. It can be seen from the convergence performance that all GRU-based models are better than SVR and FFNNs. The convergence performance of GVGRU1 and GVGRU2 with more parameters is better than GVGRU3 with fewer model parameters. In addition, among the sparse models, our SCGRU has a better convergence performance than RCGRU and X-GRU under the same parameters.

Fig. 4(a) shows the performance comparison between our SCGRU and some state-of-the-art models in power load forecasting. From the comparison results, it can be seen that the performance of the GRU-based model is better than that of FFNNs and SVR. Also, SCGRU, with a pruning rate of 97%, can give more accurate predictions than all other models, even densely-connected GRU. It has been proven[20,14] that the performance of the pruned network is similar or even better than that of the densely-connected network.

In order to better analyze the differences and details between our model and the recently proposed state-of-the-art sparse models. We compared the number of parameters, the sparse ratio, the pruning period, and the prediction error (RMSE) of each model in Table 2. We can see that SCGRU has the best performance in terms of parameter size, pruning rate, and prediction accuracy. Also, GVGRU1 can reduce the parameters by 0.19% and has a better performance than the fully connected GRU (FC-GRU); GVGRU2, which has abandoned the bias, has a 66.41% pruning rate and the model performance is slightly worse than the fully connected GRU; RCGRU has a large performance drop due to its large pruning rate; X-GRU is able to prune 95% of the parameters with a small performance drop.

4.4. Results on the LAN traffic dataset

Fig. 5 shows the comparison of the prediction performance of SCGRU and some existing models on the LAN traffic dataset, Fig. 6 shows the convergence performance of all these models. It can be seen that SCGRU has better prediction performance and convergence performance than other connection-based methods (RCGRU and X-GRU).

Also, as shown in Table 3GVGRU1 prunes 0.19% of the parameters but leads to worse prediction accuaracy; GVGRU2 and GVGRU3 can prune about 66% of the original parameters and their performance do not decrease significantly; RCGRU and X-GRU have a high pruning rate (97%) and good prediction performances but there are multiple pruning operations. In particular, SCGRU has the highest pruning rate and the best prediction accuracy. Also, it has the best performance in most metrics among the connection-based methods.

4.5. Pruning Performance Analysis

4.5.1. In Power Load Task

Fig. 7 shows the performance comparison of SCGRU, X-GRU, and RCGRU in the power load prediction task under different pruning rates. Without fine-tuning, the performance of the random sparse model decreases rapidly as the number of parameters decreases,

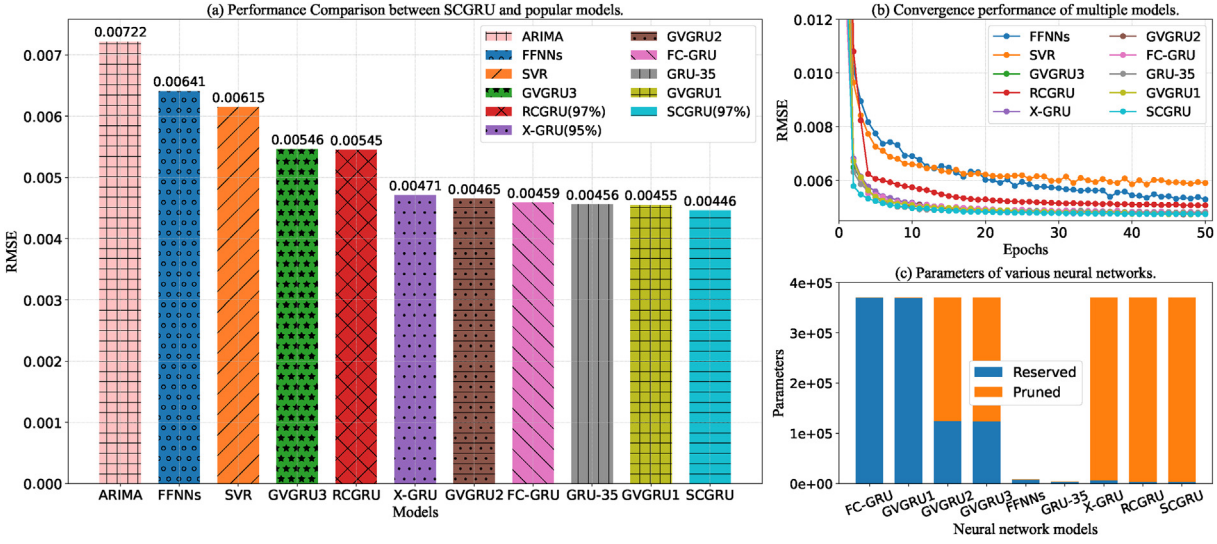


Fig. 4. The comparison of (a) prediction performance, (b) convergence speed, and (c) parameter size between SCGRU and some state-of-the-art models on the power load dataset.

Table 2
Comparison results in the power load prediction task. NVIDIA GTX 1080 GPU is used for model training and time calculation.

Architecture	Model	Parameters	Sparse(%)	#Prune	RMSE
Baseline	FC-GRU	361.3 K	–	–	0.004593
	GRU-35	3.8 K	–	–	0.004557
Classic	FFNNs	7.5 K	–	–	0.006411
	SVR	–	–	–	0.006145
	ARIMA	–	–	–	0.007222
Structure based	GVGRU1	360.6 K	0.19	Sparse structure	0.004547
	GVGRU2	121.3 K	66.41	Sparse structure	0.004649
	GVGRU3	120.6 K	66.61	Sparse structure	0.005463
Connection based	RCGRU	**3.6 K**	**97**	many	0.005450
	X-GRU	6 K	95	many	0.004712
	SCGRU(Our)	**3.6 K**	**97**	**1**	**0.004455**

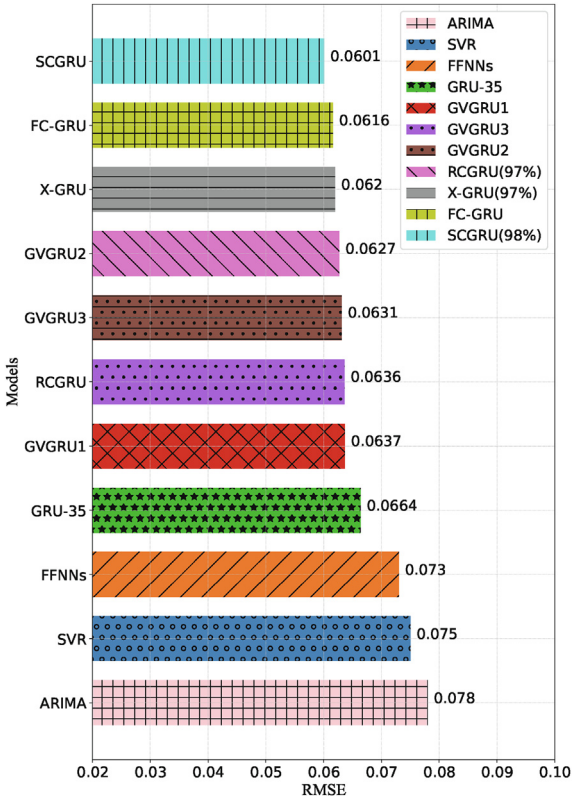


Fig. 5. Prediction performance comparison between SCGRU and some existing models on the LAN traffic dataset.

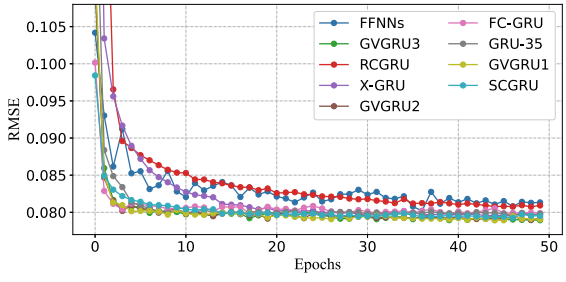


Fig. 6. Convergence curves of SCGRU and some existing models on the LAN traffic dataset.

as shown in Fig. 7(a)(a). X-GRU can maintain good performance with a pruning rate lower than 80%. The performance of SCGRU starts to decrease when more than 70% of the parameters are pruned. However, SCGRU is the only model that can maintain good performance even at a 90% pruning rate. As shown in Fig. 7(b). The performance of the three models improved significantly after fine-tuning. The optimal pruning rate of X-GRU is the lowest among the three, which is around 95%. Although RCGRU can have a pruning rate beyond 97%, its performance is not as good as X-GRU. We can see that, among these three pruning methods, the proposed SCGRU has the best performance with any pruning rate. With a pruning rate of 97%, our model performance can still maintain a good performance (In fact, it is still better than FC-GRU according to the results in Table 2).

Table 4 shows the best pruning rate (BPR) and the best extreme pruning rate (BEPR) of each model without fine-tuning of RCGRU, X-GRU, and RCGRU. From Table 4, we can know that RCGRU's performance is not good when fine-tuning is not used. The BPR of X-GRU is better than SCGRU, but SCGRU has the best extreme pruning performance. Without fine-tuning, SCGRU can have a pruning rate of 90% and the performance drop is not significant. Table 5 shows the performance comparison of each model after fine-tuning. RCGRU has a very high sparse ability, but with some performance drop. X-GRU can guarantee good performance with a lower pruning rate at 95%. SCGRU can achieve the best performance in all these four metrics.

4.5.2. In LAN task

Fig. 8 shows the performance comparison of SCGRU, RCGRU, and X-GRU in LAN traffic prediction tasks under different pruning rates. Fig. 8(a) gives the results of pruned models without fine-tuning. With the decrease of parameters, the performance of the random sparse model drops accordingly. When X-GRU prunes more than 70% of the parameters, its performance began to drop sharply. SCGRU can prune 80% of the parameters without significant performance loss. Fig. 8(b) is the results of pruned models with fine-tuning. Compared with no fine-tuning, the performance of the three models is significantly improved. The best pruning rate of RCGRU and X-GRU is 97%, and SCGRU has the best model performance with the pruning rate of 98%. Notably, SCGRU can achieve a 99.3% pruning rate without significant performance loss.

Table 6 shows that, without fine-tuning, the performance of RCGRU drops a lot. The proposed SCGRU can prune 70% of the parameters without significant performance loss. X-GRU can have the second-best performance with a slightly lower pruning rate. Table 7 shows the results of fine-tuned models. We can see that SCGRU has the best performance while having the highest pruning rates.

4.6. Performance on LSTM

In addition, we extend proposed pruning method to LSTM and test its performance on two datasets (*i.e.*, the LAN traffice dataset and the power load dataset). In the LAN traffic prediction experiment, the experimental settings are following SCLSTM [13], while

Table 3
Comparison results in the LAN traffic prediction task. NVIDIA GTX 1080 GPU is used for model training and test.

Architecture	Model	Parameters	Sparse(%)	#Prune	RMSE
Baseline	FC-GRU	361.3 K	–	–	0.0616
	GRU-35	3.8 K	–	–	0.0664
Classic	FFNNs	7.5 K	–	–	0.0730
	SVR	–	–	–	0.0750
	ARIMA	–	–	–	0.0780
Structure based	GVGRU1	360.6 K	0.19	Sparse structure	0.0637
	GVGRU2	121.3 K	66.41	Sparse structure	0.0627
	GVGRU3	120.6 K	66.61	Sparse structure	0.0631
Connection based	RCGRU	3.6 K	97	many	0.0636
	X-GRU	3.6 K	97	many	0.0620
	SCGRU(Our)	**2.4 K**	**98**	**1**	**0.0601**

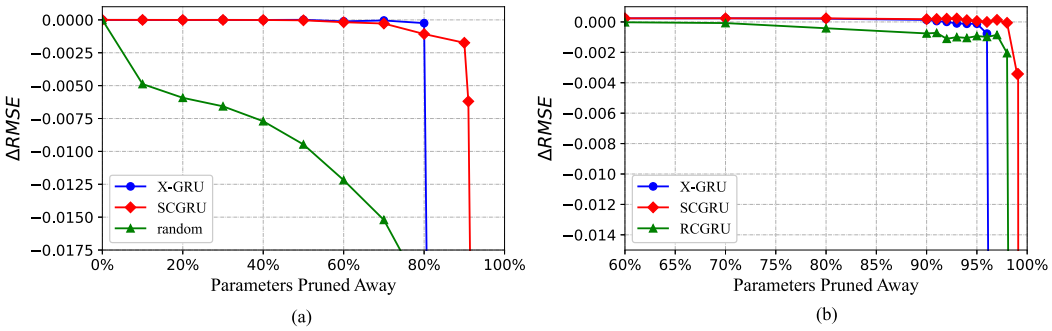


Fig. 7. The performance loss of SCGRU, X-GRU and RCGRU under different pruning rates in the power load prediction task. (a) without fine-tuning; (b) with fine-tuning. Here $\Delta RMSE = RMSE_{before} - RMSE_{after}$.

Table 4
BPR/BERP comparison of SCGRU, X-GRU, and RCGRU (without fine-tuning) on the power load dataset.

Model	BPR	Best RMSE	BEPR	BEP RMSE
RCGRU	10%	0.00947	–	–
X-GRU	**80%**	**0.00484**	–	–
SCGRU	70%	0.00487	**90%**	**0.00567**

Table 5
BPR/BERP comparison of SCGRU, X-GRU, and RCGRU (with fine-tuning) on the power load dataset.

Model	BP	Best RMSE	BEP	BEP RMSE
RCGRU	97%	0.00545	98%	0.00665
X-GRU	95%	0.00471	96%	0.00536
SCGRU	**97%**	**0.00445**	**98.4%**	**0.00468**

in the power data prediction experiment, the experimental setting is the same as that in Section 4.2. Among the comparison methods, SCLSTM was proposed by Xiong *et al.* [13], and RCLSTM was proposed by Hua *et al.* [12], and the baseline is a densely connected LSTM.

Table 8 shows the comparison results of RCLSTM, SCLSTM, our method, and the baseline method on the LAN dataset. Our method can prune 99.24% of the parameters in the densely-connected LSTM, while having the best prediction performance among the comparison methods. Table 9 shows that the results on the power-load dataset. The pruning methods all have an accuracy loss under their respective optimal pruning rates. For RCLSTM, the accuracy loss is 28.9%; for SC-LSTM, it is 22%; for our method, it is only 7.3% while pruning 99% parameters of the densely-connected networks.

Experimental results show that our method can be extended to LSTMs and can outperform existing methods. Table 10.

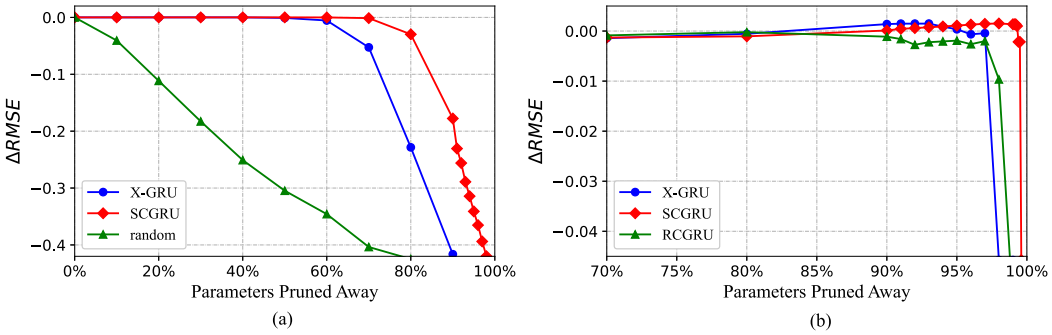


Fig. 8. The performance loss of SCGRU, X-GRU and RCGRU under different pruning rates in the LAN traffic prediction task. (a) without fine-tuning; (b) with fine-tuning. Here $\Delta RMSE = RMSE_{before} - RMSE_{after}$.

Table 6
BPR/BERP comparison of SCGRU, X-GRU, and RCGRU (without fine-tuning) on the LAN traffic dataset.

Model	BPR	Best RMSE	BEPR	BEP RMSE
RCGRU	10%	0.1022	–	–
X-GRU	60%	0.0671	70%	0.1143
SCGRU	**70%**	**0.0629**	**80%**	**0.0913**

Table 7
BPR/BERP comparison of SCGRU, X-GRU, and RCGRU (with fine-tuning) on the LAN traffic dataset.

Model	BP	Best RMSE	BEP	BEP RMSE
RCGRU	97%	0.0636	98%	0.0713
X-GRU	97%	0.0620	98%	0.1073
SCGRU	**98%**	**0.0601**	**99.3%**	**0.0637**

Table 8
Our approach is extended to LSTMs and results compared with other models on the LAN traffic dataset.

Model	Parameters	Sparse(%)	RMSE
baseline	0.36 M	–	0.0605
RCLSTM	7.4 K	98.3%	0.0640
SCLSTM	3.1 K	99.1%	0.0601
Our	**2.8 K**	**99.2%**	**0.0599**

Table 9
Our approach is extended to LSTMs and results compared with other models on the power load dataset.

Model	Parameters	Sparse(%)	RMSE
baseline	482 K	–	**0.005604**
RCLSTM	14.4 K	97%	0.007227
SCLSTM	**5.1 K**	**99%**	0.006851
Our	**5.1 K**	**99%**	0.006015

Table 10
The relationship between pre-training batches and pruning results on the power load dataset.

Epochs	RMSE
0	0.1019
1	0.0619
2	0.0612
5	**0.0601**
10	0.0624

4.7. Pre-training analysis

Small batch pre-training can improve the final pruning performance. We conduct a pre-training experiment on the LAN dataset. By setting different pre-training epochs, the final pruning performance varies. The best performance is obtained when the pre-training epoch is set as 5.

5. Conclusions and future work

In this work, we redesigned the neural connection of the GRU model for time-series prediction tasks. We proposed a model called SCGRU, the basic idea of which is to select important connections based on the sensitivity of neural connections rather than the weights values. This design can lead to GRU models with high pruning rates while having a satisfactory performance. Different from other connection-based pruning methods, SCGRU only needs one pruning process, which significantly decreases the training costs. Experimental results show that the proposed SCGRU method outperforms the existing pruning methods in terms of model performance and pruning rate. Also, it has the least training/inference costs among the connection-based methods. In summary, SCGRU can give GRU models with fewer computation costs, storage burden, as well as network complexity while maintaining good performance, making it easier to deploy on platforms with limited resources.

In future work, we may study the relationship between the sparsity and robustness of GRUs and improve the anti-interference ability of GRUs in time series prediction, which has very high commercial and civilian value. In particular, it is also of great significance to study an algorithm that can automatically give the reference sparsity rate of GRU in different learning tasks, so there is still a lot of work to do.

CRediT authorship contribution statement

Hong Tang: Conceptualization, Methodology, Supervision, Software. **Xiangzheng Ling:** Conceptualization, Methodology, Writing - original draft, Software, Visualization. **Liangzhi Li:** Writing - review & editing. **Liyan Xiong:** Conceptualization, Writing - review & editing. **Yu Yao:** Writing - review & editing. **Xiaohui Huang:** Conceptualization, Writing - review & editing.

Declaration of Competing Interest

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

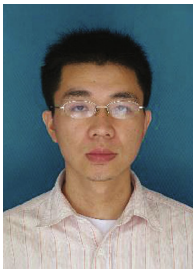
Acknowledgment

The authors would like to express thanks to the Science Research Project of Jiangxi Provincial Department of Education (under Project No. GJJ190319), the National Natural Science Foundation of China (Project No. 62067002, No. 52062016, No. 62062033), and the Natural Science Foundation of Jiangxi Province (Project No. 20192ACBL21006) for their financial supports.

References

[1] Wenti Yang, Zhitao Guan, Longfei Wu, Xiaojiang Du, and Mohsen Guizani. Secure data access control with fair accountability in smart grid data sharing: An edge blockchain approach, IEEE Internet Things J. (2020).
[2] Yiwen Guo, Chao Zhang, Changshui Zhang, and Yurong Chen. Sparse dnns with improved adversarial robustness. In Advances in neural information processing systems, pages 242–251, 2018.
[3] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. arXiv preprint arXiv:1702.04008, 2017.
[4] Nikola K Kasabov, Qun Song, Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction, IEEE Trans. Fuzzy Syst. 10 (2) (2002) 144–154.
[5] Nicholas I Sapankevych, Ravi Sankar, Time series prediction using support vector machines: a survey, IEEE Comput. Intell. Mag. 4 (2) (2009) 24–38.
[6] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. arXiv preprint arXiv:1704.02971, 2017.
[7] Sepp Hochreiter, Jürgen Schmidhuber, Long short-term memory, Neural Computation 9 (8) (1997) 1735–1780.
[8] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, pages 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics.
[9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015.
[10] Rahul Dey, Fathi M Salemt, Gate-variants of gated recurrent unit (gru) neural networks, in: 2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS), IEEE, 2017, pp. 1597–1600.
[11] Wei Wen, Yuxiong He, Samyam Rajbhandari, Minjia Zhang, Wenhan Wang, Fang Liu, Bin Hu, Yiran Chen, and Hai Li. Learning intrinsic sparse structures

within long short-term memory. In International Conference on Learning Representations, 2018.
[12] Yuxiu Hua, Zhifeng Zhao, Rongpeng Li, Xianfu Chen, Zhiming Liu, and Honggang Zhang. Deep learning with long short-term memory for time series prediction. IEEE Commun. Mag., 2019.
[13] Liyan Xiong, Xiangzheng Ling, Xiaohui Huang, Hong Tang, Weimin Yuan, Weichun Huang, A sparse connected long short-term memory with sharing weight for time series prediction, IEEE Access 8 (2020) 66856–66866.
[14] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In Advances in neural information processing systems, pages 1135–1143, 2015.
[15] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In International Conference on Learning Representations, 2018.
[16] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[17] Yann LeCun, Yoshua Bengio, Geoffrey Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[19] Yann LeCun, John Denker, Sara Solla, Optimal brain damage, Adv. Neural Inform. Process. Syst. 2 (1989) 598–605.
[20] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149, 2015.
[21] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710, 2016.
[22] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. In Advances In Neural Information Processing Systems, pages 1379–1387, 2016.
[23] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In International Conference on Learning Representations, 2019.
[24] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. arXiv preprint arXiv:1312.4400, 2013.
[25] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient inference, in: 5th International Conference on Learning Representations, ICLR 2017-Conference Track Proceedings, 2019.
[26] Fang Yu, Li Cui, Pengcheng Wang, Chuanqi Han, Ruoran Huang, and Xi Huang. Easiedge: A novel global deep neural networks pruning method for efficient edge computing. IEEE Internet of Things Journal, 2020.
[27] Guoan Yang, Junjie Yang, Lu. Zhengzhi, Deyang Liu, A convolutional neural network with sparse representation, Knowl.-Based Syst. 209 (2020) 106419.
[28] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9, 2015.
[29] Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In Advances in neural information processing systems, pages 107–115, 1989.
[30] Eh.ud.D. Karnin, A simple procedure for pruning back-propagation trained neural networks, IEEE Trans. Neural Networks 1 (2) (1990) 239–242.
[31] Suraj Srinivas, R. Venkatesh Babu, Data-free parameter pruning for deep neural networks, Comput. Sci. (2015) 2830–2838.
[32] Haonan Guo, Shenghong Li, Bin Li, Yinghua Ma, Xudie Ren, A new learning automata-based pruning method to train deep neural networks, IEEE Internet Things J. 5 (5) (2017) 3263–3269.
[33] Namhoon Lee, Thalaiyasingam Ajanthan, Philip Torr, Snip: Single-shot network pruning based on connection sensitivity, in: International Conference on Learning Representations, 2019.
[34] Steven J Nowlan, Geoffrey E Hinton, Simplifying neural networks by soft weight-sharing, Neural Comput. 4 (4) (1992) 473–493.
[35] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. arXiv preprint arXiv:1412.6115, 2014.
[36] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In International Conference on Machine Learning, pages 2285–2294, 2015.
[37] Sharan Narang, Erich Elsen, Gregory Diamos, and Shubho Sengupta. Exploring sparsity in recurrent neural networks. arXiv preprint arXiv:1704.05119, 2017.
[38] Sharan Narang, Eric Undersander, and Gregory Diamos. Block-sparse recurrent neural networks. arXiv preprint arXiv:1711.02782, 2017.
[39] Xiaoliang Dai, Hongxu Yin, Niraj K Jha, Grow and prune compact, fast, and accurate lstms, IEEE Trans. Comput. 69 (3) (2019) 441–452.
[40] Shunshi Zhang, Bradly C Stadie, One-shot pruning of recurrent neural networks by jacobian spectrum evaluation, in: International Conference on Learning Representations, 2019.
[41] Abigail See, Minh-Thang Luong, and Christopher D Manning. Compression of neural machine translation models via pruning. arXiv preprint arXiv:1606.09274, 2016.
[42] Russell Reed, Pruning algorithms-a survey, IEEE Trans. Neural Networks 4 (5) (1993) 740–747.
[43] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In ICML, 2010.
[44] Steve Uhlig, Bruno Quoitin, Jean Lepropre, Simon Balon, Providing public intradomain traffic matrices to the research community, ACM SIGCOMM Computer Communication Review 36 (1) (2006) 83–86.
[45] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.
[46] Chih-Chung Chang, Chih-Jen Lin, Libsvm: a library for support vector machines, ACM transactions on intelligent systems and technology (TIST) 2 (3) (2011) 1–27.
[47] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.
[48] George EP Box, David A Pierce, Distribution of residual autocorrelations in autoregressive-integrated moving average time series models, J. Am. Stat. Assoc. 65 (332) (1970) 1509–1526.



Hong Tang received PhD from Computer Science and Engineering of South China University of Technology (SCUT), China. He is currently in computer technology at East China Jiaotong University, China. His main fields of research interest include machine learning, neural networks, data mining. He was a project manager in Guangzhou JHComn Science & Technology Development Ltd. from 2009 to 2013, engaged in the research and development of Machine learning application.

Xiangzheng Ling received the B.Sc degree in computer science and technology from Jiujiang University, Jiujiang, China, in 2018 and the M.S. degree in computer technology from East China Jiaotong University, China, in 2021. He currently works in the Innovation Research Institute of the Information Technology Department of Shanghai Pudong Development Bank, and was a researcher of the Institute of data science and deep learning of East China Jiaotong University. He has won the second prize of the China Postgraduate Electronic Design Competition. His research interests include machine learning, deep learning, and traffic flow prediction.

Liangzhi Li is a researcher with Institute for Datability Science, Osaka University, Japan. He received the B.Sc and M.Eng degrees in Computer Science from South China University of Technology (SCUT), China, in 2012 and 2016, respectively, and Ph.D. degree in Engineering from Muroran Institute of Technology, Japan, in 2019. His main fields of research interest include computer vision, deep learning, and medical imaging. He has received the best paper award from FCST 2017 and IEEE Sapporo Section (2018).

Liyan Xiong received the B.S. degree in computer science from Jiangxi Normal University, Nanchang, China, in 1997 and the M.S. degree in computer application from East China Jiaotong University, Nanchang, in 2007. She is now a professor of the school of information engineering, East China Jiaotong University, and a graduate supervisor of the Institute of data science and deep learning, East China Jiaotong University. Her research interests include database system, natural language processing, decision support system and data mining.

Yu Yao received Master's Degree from Electrical and Electronic Engineering of North China Electric Power University(NCEPU), China. He is currently working at Marketing Department of Guangdong Power Grid Co., Ltd. Guangzhou Power Supply Bureau, engaged in the research and development of Electric Power Market. He was a chief power dispatcher at the same company from 2009 to 2015.

Xiaohui Huang received the BEng degree and the master's degree from Jiangxi Normal University, Nanchang, P.R. China, in 2005 and 2008, respectively, and the PhD degree in the Harbin Institute of Technology, P. R.China, in 2014. Since Dec, 2014, he has been at the School of Information Engineering Department, East China Jiaotong University, P.R. China, where he is currently an associate professor of Computer Science. His research interests are in the areas of machine learning, deep learning and clustering algorithm.