A Fusion Algorithm of Multi-model Pruning and Collaborative Distillation Learning

Zihan Liu¹, Zhiguo Shi*¹
¹ School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, Beijing, 100083, P.R. China
*Corresponding author’s e-mail: szg@ustb.edu.cn

Abstract. In order to improve the prediction performance, the complex depth network model needs a large number of calculation parameters, and the negative impact is the fact that it greatly increases the calculation time and energy consumption. In this paper, a fusion algorithm of model pruning and joint multi-model knowledge distillation learning is proposed. By constructing an adaptive joint learning loss function including distillation, multi-model training is carried out. This method can replace the training process of model fine-tuning after pruning. In this paper, firstly, multi-classification tasks are carried out on different model structures and data sets, and a complex network model with good effect is trained as a teacher's model. Then, the channel pruning process of random pruning degree is executed to generate multiple student models. Finally, the accuracy of multiple student models is improved by using this method. The experimental results show that this method effectively improves the accuracy of each model after pruning, and the experimental results achieve a suitable balance between model performance and accuracy.

1. Introduction
In recent years, deep neural networks have shown their excellent performance in various tasks of machine learning, such as some traditional problems of convolutional neural networks in the field of computer vision[1] and natural language processing[2].

In order to obtain better results, people often design the network structure more complicatedly and the network will have more layers, which also mean that the number of parameters of the model is getting larger and larger. It limits these the network model to be deployed on platforms and devices with relatively low computing and storage resources. For neural networks with more layers and nodes, it becomes more and more important to reduce their storage and computing costs. Denil M[3] shows that there is significant redundancy in the parameterization of multiple deep learning models.

Among the existing deep model compression methods, the network pruning method attempts to remove redundant and unimportant neurons in view of the redundancy of model parameters, which can reduce network complexity and solve the problem of overfitting[4], and compress the model volume. Network pruning methods are commonly used on pre-trained CNN models. For example, Han S[5] proposed deep compression and pruning weighted connections with little effect on accuracy. The Network Slimming method [6] proposed by Zhuang Liu et al. applies L1 regularization to the scaling factor of the batch normalization (BN) layer, which can analyze unimportant channels and perform channel-level pruning. The method founded on soft weight sharing[7] achieves the combination of quantization and pruning during the retraining process. Luo et al proposed Thinet[8] to use the greedy
algorithm to calculate the static information of the current layer convolution kernel from the next layer to cut the convolution kernel.

Another popular compression method is the knowledge distillation method: Bucilua C[9] used pre-trained complex models to generate labels for large amounts of unlabeled data, and used these data as supplementary data for labeled data to supervise and guide simple models. Hinton G[10] proposed a method of knowledge distillation, using complex and powerful teacher models to guide the learning of simple student models.

Sau B.B[11] expanded the teacher-student framework for network compression, using teacher models to train student models, and adding noise to the output of teacher models to improve the performance of student networks. DML[12] is a two-way distillation method that uses multiple student networks to teach each other without the need for a separate teacher network, and knowledge is spread among each student network. Minami S[13] control the loss of soft targets by referring to the training accuracy of each network to reduce the interference of knowledge transfer in the early training stage of knowledge distillation. Romero A[14] proposed FitNets, used the Hints method to train the first half of the network, and then used traditional knowledge distillation to train the entire FitNets.

In general, network pruning has exceptional performance in the above papers, but the fine-tuning process after network pruning can be improved. In the knowledge distillation method, the knowledge transferred from the teacher network to the student network is provided in the form of soft labels. Soft labels play a part in label smoothing and increase the generalization ability of the model.

2. Methods

2.1. The overall framework of the method

This paper proposes a fusion method of multi-model pruning and distillation for the currently commonly used multi-layer convolutional neural network: first, a sparse training process is performed, that is, by adding regularization constraints[15] on the gamma coefficients of the BN layer[16] in the convolution network, the network model adjusts the parameters gradually with the training process, making the network parameters more sparse.

Then, the bn layer behind each convolutional layer in the trained network is subject to random convolution kernel channel pruning to generate multiple model networks with different model sizes.

Next, the multi-model collaborative distillation algorithm proposed in this article is executed: the complete network model before pruning is used as the static teacher model in knowledge distillation, and its output is not affected by the training process. For multiple models after pruning, in the iterative training process, the model with a larger pruning degree regards the models of all models with a smaller pruning degree as a dynamic teacher model, and provides dynamically updated output targets results. The multi-model collaborative distillation algorithm uses the soft-targets cross-entropy loss generated by the teacher network, the hard-targets cross-entropy loss from the data set, and the adaptive mean of the relative entropy between the models to form a collaborative loss function, which replaces model fine-tuning process in the conventional model compression process. Final compressed network model is generated after the algorithm is executed.

The method structure diagram is as follows:
2.2. Sparse training and channel pruning process

Before the teacher model training process, we add a scaling factor $\gamma$ to each channel of the BN layer, and then multiply it with the output of the channel. Then train the network weights and these scaling factors, and finally cut the channels with small scaling factors directly to complete the pruning operation. During the training process, our loss function can be defined as:

$$L = L_0 + \lambda \sum \gamma g(\gamma)$$

$$L_0 = \sum_{(x,y)} l_0(\hat{y}, y)$$

$$g(\gamma) = \|\gamma\|_1$$

Where $(x, y)$ is the input and targets value of a single batch of data, $L_0$ is the original loss function of the CNN network, $g(\gamma)$ is the penalty term on the scaling factor, which is only applied to the $\gamma$ term of the BN layer, $\lambda$ is the balance factor. We set $g(\gamma) = \|\gamma\|_1$, which is the L1 norm. The L1 norm refers to the sum of the absolute values of the elements in the vector, otherwise known as lasso regularization. Adding L1 norm can achieve the effect of sparse solution[15]. In the back-propagation, the gradient update process for the $\gamma$ term of the BN layer is as follows:

$$\frac{\partial L}{\partial \gamma} = \frac{\partial L_0}{\partial \gamma} + \lambda (sgn(\gamma))$$

In the formula, $sgn(\gamma)$ is the partial derivative of $L$ with respect to $g(\gamma)$. After adding the scaling factor regular term, various scaling factors in the BN layer of the model will tend to zero. Then we cut the channel corresponding to the scaling factor close to zero to complete the pruning operation of the convolution layer channel.

2.3. Collaborative distillation algorithm

The purpose of knowledge distillation[10] is to compress the network, and the teacher model can transfer the learned knowledge to the student model to obtain the network effect as similar as possible with fewer parameters.

The classification probability generated by a complex model using the same training set can be used as soft targets for training small models. Hard targets included in the data set have a probability of 1 on the corresponding index and 0 for other categories. Compare with hard targets, soft targets often have a certain probability value in each category, which represents a certain degree of similarity between the category corresponding to the targets and the real category. Softened targets can provide more
information entropy. Using the soft targets, the student network can learn more information and get higher accuracy.

![Knowledge distillation diagram](image_url)

Figure 2. Knowledge distillation.

The neural network often obtains the network output through the softmax layers at the end. By comparing the size of each output $Z_i$, it is converted into a probability value $y_i$ of this class. Here, a temperature parameter $T$ is introduced to softmax to control the smoothness of the output distribution. The modified softmax formula is as follows (softmax-T):

$$y_i = \frac{\exp\left(\frac{Z_i}{T}\right)}{\sum \exp\left(\frac{Z_j}{T}\right)} \tag{5}$$

When the $T$ parameter is placed at 1, it is the ordinary Softmax formula. The larger the $T$ value, the smoother the curve of the obtained Softmax function. After setting the temperature $T$ to a larger value, the Softmax-T of the teacher model and the student model output $y_i^{Teacher-T}, y_i^{Student-T}$ and then set the temperature $T$ to 1 to get the conventional output $y_i^{Student}$. In order to calculate the similarity between the output vector and the ideal vector in the neural network, cross entropy is usually describe the distance between the two probability distributions. KL divergence (relative entropy) refers to describe the difference between two probability distributions.

In knowledge distillation, cross entropy is a loss function commonly used with softmax in multi-classification problems:

$$H(P||Q) = \sum_i p(i) \ast \left[ \log \left( q(i) \right) \right] \tag{6}$$

For the two probability distributions $P$ and $Q$, the relative entropy is defined as follows:

$$D(P||Q) = \sum_i p(i) \ast \left[ \log \left( \frac{p(i)}{q(i)} \right) \right] \tag{7}$$

The combined neural network loss function formula is as follows:

$$L_{total} = aKL(y^{Teacher-T}, y^{Student-T}) + CrossEntropy(y^{Student}, y^{True}) \tag{8}$$

The $KL(y^{Teacher-T}, y^{Student-T})$ in the above formula is called $L_{soft}$, that is, the cross-entropy loss obtained by the soft targets generated by the teacher network, Cross Entropy ($y^{Student}, y^{True}$) is called $L_{hard}$, which is the cross-entropy loss obtained by using the hard targets in the training set. Use $a$ to adjust the proportion of the two loss values.

In knowledge distillation, the teacher network only conducts one-way knowledge transfer to the student network during the learning process. In the method proposed in this paper, multiple networks are used for training at the same time. During the training process, each network not only accepts the
supervision of the true value label, and the fixed probability estimation of the teacher network, but also refers to the dynamic probability estimation of other networks. This method can further improve the generalization ability. Through the process, feedback information is obtained from the learning state between the student networks to optimize the training process and achieve collaborative training. In each iteration, we calculate the predictions of different models and update the parameters of the network according to the predictions of other models. The optimizing process of each student network is iterative until it converges gradually.

For multiple student models \( M_1, \ldots, M_M \), their output is \( Y^1, \ldots, Y^M \), for two student models \( M_a \) and \( M_b \). Relative entropy between soft targets generated by softmax-T formula:

\[
KL(Y^b-T, Y^a-T) = \sum_i Y^a_i \cdot \log \left( \frac{Y^a_i}{Y^b_i} \right)
\]  

(9)

The loss function of the student model \( M_1 \) consists of the cross entropy loss of the soft targets and the cross entropy loss of the hard targets:

\[
L^1 = \alpha L^1_{soft} + L^1_{hard}
\]  

(10)

The loss function of student model \( M_2 \) is the cross entropy loss of soft label, the cross entropy loss of hard label and the relative entropy composition of \( M_2 \) to \( M_1 \). The parameters \( \alpha \) and \( \beta \) are the coefficients of cross entropy and relative entropy.

\[
L^2 = \alpha L^2_{soft} + L^2_{hard} + \beta KL(Y^2-T, Y^1-T)
\]  

(11)

The loss function of student model \( M_n \) is composed of cross entropy loss of soft label, cross entropy loss of hard label and mean value of relative entropy of \( M \) to other models:

\[
L^n = \alpha L^n_{soft} + L^n_{hard} + \beta \frac{\sum_{i=1}^{n-1} KL(Y^n-T, Y^i-T)}{n-1}
\]  

(12)

The original knowledge of each student model is affected due to the pruning process. At the beginning of the distillation training of the network, the network with poor accuracy will interfere with the model training and affect the convergence speed of each model. Therefore, the probability that the relative entropy is calculated should be reduced when the model accuracy is low. In this section, the relative entropy of each model will be randomly sampled. For the model \( M_n \), the accuracy \( acc_i \) generated by the other student models with the training of different batches of data will be used as the parameter of random sampling[13]. The dynamic adaptive parameter \( \rho_i \) follows the Bernoulli distribution with the parameter \( acc_i \). With the continuous improvement of the accuracy of each model, the probability of the relative entropy being selected will gradually increase:

\[
acc_i = \frac{1}{N} \sum_{j=1}^{N} \delta_{g^j, y^j_{true}}
\]  

(13)

\[
\rho_i \sim Bernoulli \ (acc_i)
\]  

(14)

The adaptive relative loss term of \( M_n \) to other models is:

\[
L^n_{adaptive} = \sum_{i=1}^{n-1} \rho_i KL(Y^n-T, Y^i-T)
\]  

(15)

\[
L^n = \alpha L^n_{soft} + L^n_{hard} + L^n_{adaptive}
\]  

(16)

The multi-model collaborative training process has the effect of regularization. In the regular training process, the truth-value label coding will make the model too confident in the prediction results during the training process, which may easily lead to overfitting. The collaborative training method transfers the knowledge of the network to other networks, reduces the occurrence probability of overfitting, and plays a role similar to label smoothing[27].
Figure 3. Multi-model collaborative distillation method.

The process of transferring knowledge between student networks is asynchronously updated according to different batches of data. When training each batch of data, the $M_1$ network parameters are first updated according to the output of the teacher model, and then $M_2$ generates a new output $Y_i$ to replace the original output, and then update the $M_2$ network parameters according to the output of the teacher model and the output of the $M_2$ network, so as to update the parameters of each student model in turn, calculate the accuracy of the model on the test set, and update its own output in time. The algorithm is as follows:

**Algorithm 1:**

- **Input:** Train set $D_{train}$, Test set $D_{test}$, Teacher model $W_t$, Student models $W_1, ..., W_n$, Iteration number $t$
- **Initialization:** $i = 1$
  1. While $i \leq t$ do
  2. Use $D_{test}$ to calculate accuracy of $W_1, ..., W_n$
  3. For $j = 1$ to $n$
  4. Compute predictions $p^j$ of $W_j$
  5. Compute $L_{soft}^j$ and $L_{hard}^j$ with $W_t$ and $D_{train}$ by Eq. (8)
  6. Compute Dynamic adaptive parameters $p_j$ by Eq. (14)
  7. Compute Adaptive relative loss function $L_{adaptive}^j$ by Eq. (15)
  8. Compute final loss function $L^j$ by Eq. (16)
  9. Compute gradients and update parameters
  10. update new predictions $p^j$ of $W_j$
  11. End for
  12. $t \leftarrow t + 1$
  13. End while

3. **Experimental Results and Discussion**

3.1. **Environment configuration**

In order to evaluate the effect of the method in this article on the network model, the experiments used the currently popular deep learning tool PyTorch. PyTorch is an open source Python machine learning library that inherits many advantages of NumPy and has better computing efficiency. PyTorch has a
large number of APIs, which can quickly complete the construction and training of deep neural network models. The experimental platform hardware configuration is: CPU: Intel Core i3-8100@4.00GHz*8, GPU: GeForce GTX 1060; 16GB RAM and 6GB Video Memory. The operating system is a 64-bit version of windows10.

3.2. Dataset introduction
In this section, the CIFAR-100 dataset[18] and the VeRi-776 dataset[19] are used for the model training process. CIFAR-100 is a color image data set containing universal objects. It is an image classification[20] data set used in the field of machine vision. It has 20 categories and a total of 100 categories, each of which contains 600 images (500 training images and 100 test images). VeRi-776 dataset contains more than 50,000 images of 776 vehicles, captured by 20 cameras in 24 hours, covering a 1.0 km² area. These images were captured in unrestricted surveillance scenes[21] in the real world. The images are captured in a real-world unconstrained surveillance scene and labeled with varied attributes like types, colors and brands. So complicated models can be learnt and evaluated for vehicle Re-Id[22]. The model in this article use this data set for vehicle type classification tasks.

3.3. Experimental Results and Discussion
In this section, the commonly used resNet164[24] and DenseNet40[25] are used to test the effect, and the effect of different algorithms is compared through the fine-tuning method for recovering precision after comparing the channel prune[6], the knowledge distillation[10], and the collaborative distillation fusion method. In the experiment, the data preprocessing process adjusted the training samples according to the requirements of the model input size, and completed Data Augmentation[23], such as random scaling and random tailoring, and finally performed the data normalization process.

Using resNet164 network and CIFAR-100 data set, after training 75 epochs, the accuracy of the teacher model on the test set is 75.4%. Using the resNet164 network and the VeRi-776 data set, after training 40 epochs, the accuracy of the teacher model on the test set is 85.3%. After channel pruning, according to different pruning degrees, four pruning models are generated as student models, and FLOPs are used to measure the complexity of the algorithm/model. Detailed information of each model is shown in the following table.

| Dataset  | Teacher model | Student model | Model 1 | Model 2 | Model 3 | Model 4 |
|----------|---------------|---------------|---------|---------|---------|---------|
| CIFAR-100| Size(MB) 6.92  | 6.52 6.26 5.96 | 5.61    |         |         |         |
|          | Flops(M) 254.5 | 226.0 210.8 192.8 | 172.5   |         |         |         |
| VeRi-776 | Size(MB) 6.92  | 4.65 4.26 3.86 | 3.50    |         |         |         |
|          | Flops(M) 254.5 | 149.7 129.1 109.6 | 92.4    |         |         |         |

Using DenseNet40 network and CIFAR-100 data set, after training 100 epochs, the accuracy of the teacher model on the test set is 72.59%. Using the DenseNet40 network and the VeRi-776 data set, after training 40 epochs, the accuracy of the teacher model on the test set is 86.2%. The detailed information of each model is shown in the following table.

| Dataset  | Teacher model | Student model | Model 1 | Model 2 | Model 3 | Model 4 |
|----------|---------------|---------------|---------|---------|---------|---------|
| CIFAR-100| Size(MB) 4.22  | 3.99 3.22 2.45 | 1.67    |         |         |         |
|          | Flops(M) 290.2 | 264.7 224.6 178.1 | 129.8   |         |         |         |
| VeRi-776 | Size(MB) 4.22  | 2.29 1.87 1.66 | 0.85    |         |         |         |
|          | Flops(M) 290.2 | 169.2 134.5 117.3 | 46.2    |         |         |         |
In this section, we use the pytorch version of the Adam optimizer to perform all accuracy restoration experiments according to Algorithm 1, we set the initial learning rate to 0.01, and bind learning rate adjustment method ReduceLROnPlateau to the optimizers: When the accuracy of the test set no longer increases, the learning rate is adjusted. For the resNet164 network and the CIFAR-100 data set, set the Step to 3, the factor to 0.2, train a total of 10 epochs, and set the batch size to 20. Choose the model that performs best in the test set for comparison. For the resNet164 network and the VeRi-776 data set, set the Step to 4, the factor to 0.1, train a total of 15 epochs, and set the batch size to 20. Choose the model that performs best in the test set for comparison.

Table 3. Comparison of resNet164 network precision recovery methods.

|           | Teacher model | Model 1 | Model 2 | Model 3 | Model 4 |
|-----------|---------------|---------|---------|---------|---------|
| CIFAR-100 | Channel pruning | 71.7%   | 57.4%   | 28.4%   | 11.2%   |
|           | Fine-tuning   | 73.6%   | 73.3%   | 73.2%   | 72.6%   |
|           | Knowledge distillation | 73.8% | 73.5% | 73.4% | 72.7% |
|           | Ours          | 73.8%   | 74.0%   | 73.7%   | 72.9%   |
| VeRi-776  | Knowledge distillation | 72.5% | 58.3% | 30.0% | 11.8% |
|           | Ours          | 86.3%   | 86.7%   | 86.4%   | 86.5%   |

For the DenseNet40 network and the CIFAR-100 dataset, set the Step to 3, set the factor to 0.2. Train a total of 10 epochs, and set the batch size to 20. Choose the model that performs best in the test set for comparison. For the DenseNet40 network and the VeRi-776 data set, set the Step to 4, the factor to 0.1, train a total of 15 epochs, and set the batch size to 20, select the model that performs best in the test set for comparison.

Table 4. Comparison of DenseNet40 network precision recovery methods.

|           | Teacher model | Model 1 | Model 2 | Model 3 | Model 4 |
|-----------|---------------|---------|---------|---------|---------|
| CIFAR-100 | Channel pruning | 68.1%   | 59.9%   | 41.0%   | 0.61%   |
|           | Fine-tuning   | 69.5%   | 69.3%   | 69.0%   | 67.2%   |
|           | Knowledge distillation | 69.8% | 69.4% | 69.3% | 67.2% |
|           | Ours          | 69.8%   | 69.6%   | 69.3%   | 67.5%   |
| VeRi-776  | Knowledge distillation | 81.9% | 69.7% | 40.3% | 21.2% |
|           | Ours          | 86.7%   | 86.4%   | 87.1%   | 86.0%   |

In this section, we use the fine-tuning method, knowledge distillation recovery method and our collaborative distillation method for each student model to compare it with the untreated student model. The accuracy comparison results of the experiment with different accuracy recovery methods are shown in Table 3&4. According to the experimental results, the last two methods of the resNet164 network containing distillation methods are better than the fine-tuning methods, and the collaborative distillation method compares the knowledge distillation recovery method, and the collaborative training results of the last three models have improved to varying degrees. In the experimental results of DenseNet40, the
two methods with distillation method are better than the fine-tuning method, and the combined distillation method proposed in this paper is not inferior to the knowledge distillation recovery method for the training effect of all student models. Experiments have proved that comparing the performance of different recovery methods, and the collaborative distillation method proposed in this paper can improve the performance of each student network, and the student network can obtain more effective knowledge from the dynamically updated dynamic teacher network.

![Comparison of the number of student models participating in training](image1)

Figure 4. Comparison of the number of student models participating in training.

We use different number of resnet164 student models to carry out collaborative training. After 10 epochs training, we get data for analysis. Comparison of the number of student models participating in training. It can be seen from the above figure that for the same model, with the increase of the number of student networks, the accuracy is also improved. When the number of models is larger, the performance of the model is better, while the more the number, the smaller the accuracy improvement.

![Comparison of relative entropy weighting scheme](image2)

Figure 5. Comparison of relative entropy weighting scheme.
In this section, we use Eq. (12) instead of Eq. (16) to change the adaptive relative loss $L_{n_{\text{adaptive}}}^\text{relative}$ into the mean value of the relative loss, and use resnet164 network to compare the accuracy of the two methods after 10 epochs of training. From the above table, we can see that for all student models, the combined distillation method is not inferior to the training results of the equal weight method.

4. Conclusion
This paper proposes a fusion method of multi-model pruning and distillation for the currently commonly used multi-layer convolutional neural network: Using the soft targets, cross-entropy loss generated by the teacher network, the hard targets, cross-entropy loss from the data set, and the adaptive mean of the relative entropy between the models to form a collaborative loss function, which replaces model fine-tuning process in the conventional model compression process. By comparing different precision recovery methods, comparing the number of student models participating in the training, and comparing the relative entropy weighting scheme, the effectiveness of the proposed method is proved.

Acknowledgment
This paper is based on the project of urban intelligent traffic perception and decision-making system, and supported by the School of Computer and Communication Engineering, University of Science and Technology Beijing. and supported by Science and Technology Projects of Sichuan Province (No.2018GZDZX0034).

References
[1] Szeliski, R. (2010) Computer vision: algorithms and applications. Springer Science & Business Media, Berlin.
[2] Kim, Y. (2014) Convolutional neural networks for sentence classification. In: Empirical Methods in Natural Language Processing (EMNLP). Doha. pp. 1746–1751.
[3] Denil, M., Shakibi, B., Dinh, L. (2013) Predicting parameters in deep learning. In: Neural Information Processing Systems, Nevada. MIT Press. pp. 2148-2156.
[4] Lawrence S, Giles C L. Overfitting and neural networks: conjugate gradient and backpropagation[C]/Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium. IEEE, 2000, 1: 114-119.
[5] Han S, Mao H, Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding[J]. arXiv preprint arXiv:1510.00149, 2015.
[6] Liu Z, Li J, Shen Z, et al. Learning Efficient Convolutional Networks through Network Slimming[C]. international conference on computer vision, 2017: 2755-2763.
[7] Ullrich K, Meeds E, Welling M, et al. Soft Weight-Sharing for Neural Network Compression[C]. international conference on learning representations, 2017.
[8] Luo J, Wu J, Lin W, et al. ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression[C]. international conference on computer vision, 2017: 5068-5076.
[9] Bucilua C, Caruana R, Niculescu-Mizil A. Model compression[C]/Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2006: 535-541.
[10] Hinton G, Vinyals O , Dean J. Distilling the Knowledge in a Neural Network[J]. Computer Science, 2015, 14(7):38-39.
[11] Sau B B, Balasubramanian V N. Deep model compression: Distilling knowledge from noisy teachers[J]. arXiv preprint arXiv:1610.09650, 2016.
[12] Zhang Y, Xiang T, Hospedales T M, et al. Deep Mutual Learning[C]. computer vision and pattern recognition, 2018: 4320-4328.
[13] Minami S, Yamashita T, Fujiyoshi H, et al. Gradual Sampling Gate for Bidirectional Knowledge Distillation[C]. international conference on machine vision, 2019: 1-6.
[14] Romero A, Ballas N, Kahou S E, et al. FitNets: Hints for Thin Deep Nets[C]. international conference on learning representations, 2015.
[15] Micchelli C A, Morales J, Pontil M, et al. Regularizers for structured sparsity[J]. Advances in Computational Mathematics, 2013, 38(3): 455-489.
[16] Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift[C]. international conference on machine learning, 2015: 448-456.
[17] Krizhevsky A, Sutskever I, Hinton G E, et al. ImageNet Classification with Deep Convolutional Neural Networks[C]. neural information processing systems, 2012: 1097-1105.
[18] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images[J]. 2009.
[19] Liu X, Liu W, Mei T, et al. Provid: Progressive and multimodal vehicle reidentification for large-scale urban surveillance[J]. IEEE Transactions on Multimedia, 2017, 20(3): 645-658.
[20] Liu X, Liu W, Mei T, et al. A deep learning-based approach to progressive vehicle re-identification for urban surveillance[C]//European conference on computer vision. Springer, Cham, 2016: 869-884.
[21] Liu X, Liu W, Ma H, et al. Large-scale vehicle re-identification in urban surveillance videos[C]//2016 IEEE International Conference on Multimedia and Expo (ICME). IEEE, 2016: 1-6.
[22] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.
[23] Uzunova H, Wilms M, Handels H, et al. Training CNNs for image registration from few samples with model-based data augmentation[C]//International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, Cham, 2017: 223-231.
[24] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
[25] Huang G, Liu Z, Van Der Maaten L, et al. Densely connected convolutional networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 4700-4708.