Knowledge Transfer Graph for Deep Collaborative Learning

Soma Minami[1], Tsubasa Hirakawa[2], Takayoshi Yamashita[1], and Hironobu Fujiyoshi[3]

[1] Department of Computer Science, Chubu University, Kasugai-shi, 487-0027, Japan
{minami@mprg.cs,takayoshi@isc}.chubu.ac.jp
[2] Chubu Institute for Advanced Studies, Chubu University, Kasugai-shi, 487-0027, Japan
hirakawa@mprg.cs.chubu.ac.jp
[3] Department of Robotics, Chubu University, Kasugai-shi, 487-0027, Japan
fujiyoshi@isc.chubu.ac.jp

Abstract. Knowledge transfer among multiple networks using their outputs or intermediate activations have evolved through manual design from a simple teacher-student approach to a bidirectional cohort one. The major components of such knowledge transfer framework involve the network size, the number of networks, the transfer direction, and the design of the loss function. However, because these factors are enormous when combined and become intricately entangled, the methods of conventional knowledge transfer have explored only limited combinations. In this paper, we propose a novel graph representation called knowledge transfer graph that provides a unified view of the knowledge transfer and has the potential to represent diverse knowledge transfer patterns. We also propose four gate functions that control the gradient and can deliver diverse combinations of knowledge transfer. Searching the graph structure enables us to discover more effective knowledge transfer methods than a manually designed one. Experimental results show that the proposed method achieved performance improvements.

1 Introduction

Deep neural networks have accomplished significant progress by designing their internal structure (e.g., a network's module [1–4] and architecture search [5–8]). The performance of existing networks can be further improved by knowledge transfer among multiple networks, such as knowledge distillation (KD) [9] and deep mutual learning (DML) [10], in extensive tasks without any additional dataset. These methods, which we call "collaborative learning," transfer knowledge between multiple networks using their outputs and/or intermediate activations.

Collaborative learning has been manually designed in extensive studies [9, 11–13, 10, 14–16], including the simple teacher-student approach [9], self-distillation [12],
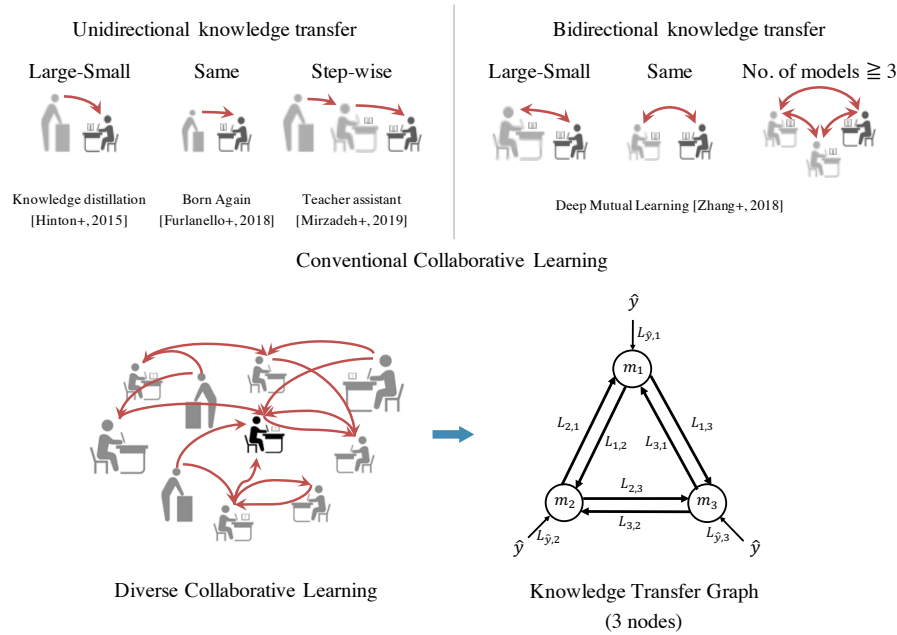
Fig. 1: Concept of the proposed method. From left to right, unidirectional and bidirectional knowledge transfer proposed by previous studies, the goals of our study, and the knowledge transfer graph representation we propose. In the graph, each node represents a network, each edge represents the direction of knowledge transfer, and $L_{s,t}$ represents the loss function used for training node t. The graph can represent diverse collaborative learning, including conventional methods.

an intermediation by teacher assistant [13], and the bidirectional cohort approach [10]. The major components of such collaborative learning are the network size, the number of networks, the transfer direction, and the design of the loss function. In general, increasing the number of networks tends to improve the performance of the target network [10, 13–15]. Cho *et al.* [17] also pointed out that larger models do not often make better teachers. The methods of conventional knowledge transfer have only explored limited combinations because the combination of the key factors is enormous and has become intricately entangled. Therefore, it is necessary to extensively explore diverse patterns of collaborative learning to achieve more effective knowledge transfer.

In this research, we explore more diverse knowledge transfer patterns in the above key factors for collaborative learning. Figure 1 shows the concept of our research. We propose a novel graph representation called *knowledge transfer graph* that can represent both conventional and new collaborative learning. A knowledge transfer graph provides a unified view of knowledge transfer and has the potential to represent diverse knowledge transfer patterns. In the graph, each

node represents a network, and each edge represents a direction of knowledge transfer. On each edge, we define a loss function that is used for transferring knowledge between the two nodes linked by the edge. Combinations of these loss functions can represent any collaborative learning with pair-wise knowledge transfer. In this paper, we propose four types of gate functions (through gate, cutoff gate, linear gate, correct gate) that are introduced into loss functions. These gate functions control the loss value, thereby delivering different effects of knowledge transfer. By arranging the loss functions at each edge, the graphs enable the representation of diverse collaborative learning patterns. Knowledge transfer graphs are searched for the network model on each node and the gate function on each edge, which enables us to discover a more effective knowledge transfer method than a manually designed one.

Our contributions are as follows.

- We propose a knowledge transfer graph that represents conventional and new collaborative learning.
- We propose four types of gates function (through gate, cutoff gate, linear gate, correct gate) to control backpropagation while training the networks. The knowledge transfer graph optimizes the gates by means of a hyperparameter search, which can achieve diverse collaborative learning.
- We found that our optimized graphs outperformed conventional methods.

2 Related Work

2.1 Unidirectional knowledge transfer

In unidirectional knowledge transfer, the outputs of a pre-trained network are used as pseudo labels in addition to supervised labels for learning a target network effectively. Hinton *et al.* [9] proposed knowledge distillation, which trains a student network by using teacher network's outputs. They succeeded in effectively transferring the teacher's internal representation to the student by introducing a temperature parameter into the softmax function. Furlanello *et al.* [12] demonstrated that KD can also train effectively in cases where the teacher network's architecture is the same as that of the student network. Mirzadeh *et al.* [13] proposed a method that adds a middle network, called a teacher assistant, between a teacher and student. When there is a large performance gap between the teacher and the student, students can be effectively trained by separating them with a middle network. Various approaches that transfer from intermediate layers have been also proposed [11, 18–20], e.g. hint [11], flow of activations between layers [18], and attention map [19]. [21–23] transfer mutual relations of data samples in a mini-batch. Distillation has been applied to object detection [24], domain adaptation [25], text-to-speech [26], etc.
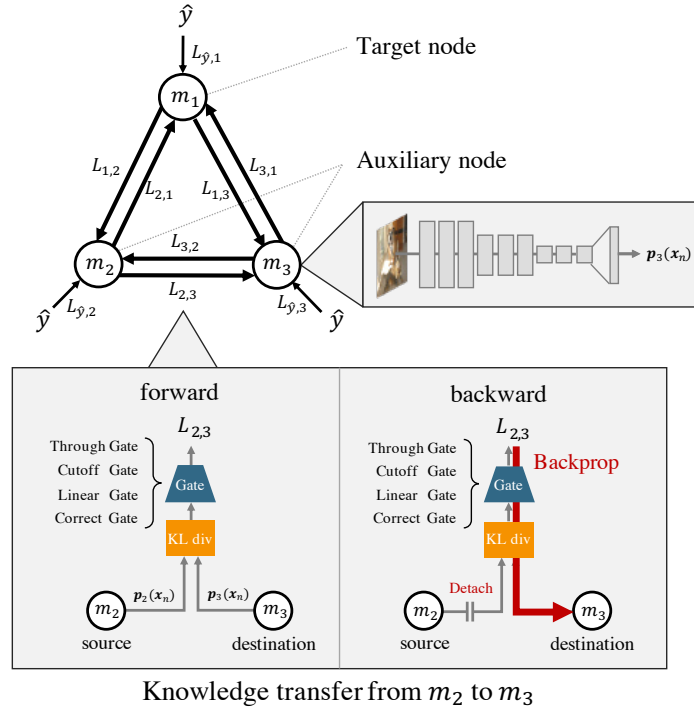
Fig. 2: Knowledge transfer graph (for 3-node case). Each node represents a model, and a loss function $L_{s,t}$ is defined for each edge. $\hat{y}$ is a label. $L_{s,t}$ calculates the KL divergence from the outputs of two nodes and then passes it through a gate function. The calculated loss gradient information is only propagated in the direction of the arrow. We can also represent unidirectional knowledge transfer by cutting off edges with a cutoff gate.

2.2 Bidirectional knowledge transfer

In the bidirectional method, which was first proposed by Zhang *et al.* [10], there is no pre-trained teacher; randomly initialized students teach each other by transferring their knowledge. Even when using networks with identical structures, the accuracy is improved. Zhang *et al.* pointed out that DML is connected to entropy regularization [27, 28]. In this method, all loss functions used in each network are identical. There could be more potential variants in collaborative learning if a combination of different loss functions was used. Further improvements in accuracy can be achieved by using the ensemble outputs of collaboratively trained networks as teachers [14, 15], and by sharing the intermediate layers of these networks [14, 15, 29]. DML has been applied to large scale distributed training [30] and re-identification [31]. Dual student [16] is a method of bidirectional knowledge transfer in semi-supervised learning.

3   Proposed Method

We explore graph structures representing diverse knowledge transfer by combining loss functions with four types of gate. We describe how to represent knowledge transfer graphs in Sec. 3.1, loss function of our proposed method in Sec. 3.2, four types of gate function in Sec. 3.3, optimization method of each model in Sec. 3.4, and graph optimization method in Sec. 3.5.

3.1   Knowledge transfer graph representation

Figure 2 shows the knowledge transfer graph representation with three nodes. In the proposed method, the direction of knowledge transfer between networks is represented by a directed graph, and a different loss function is defined for each edge. By defining different loss functions, it is possible to express various knowledge transfer methods.

   We define a directed graph where node m_i represents the ith model used for training. Each edge represents the directions in which gradient information is transferred. In this paper, we refer to a node that transfers its knowledge to another as a source node, and a node to which the source node transfers its knowledge as a destination node. The losses calculated from the outputs of the two models are back-propagated towards the destination node. Losses are not back-propagated to the source node.

3.2   Loss function

The mini-batch comprising the image of the nth sample $\boldsymbol{x}_n$ and the label $\hat{y}_n$ is represented as $\mathcal{B} = \{\boldsymbol{x}_n, \hat{y}_n\}_{n=1}^N$, and the batch size of mini-batch $\mathcal{B}$ is represented as $|\mathcal{B}|$. The label $\hat{y}_n$ represents class id. The number of models used for learning is M, and the source and destination nodes are m_s and m_t, respectively.

   When obtaining the difference in output probabilities between nodes, we use the Kullback-Leibler (KL) divergence $KL(\boldsymbol{p}_s(\boldsymbol{x}_n)||\boldsymbol{p}_t(\boldsymbol{x}_n))$. Here, $\boldsymbol{p}_s$ and $\boldsymbol{p}_t$ are the outputs of the source and destination nodes, respectively, and consist of probability distributions normalized by the softmax function.

   If the one-hot vector representation of the label $\hat{y}_n$ is $\boldsymbol{p}_{\hat{y}_n}$, the loss between $\boldsymbol{p}_{\hat{y}_n}$ and the output $\boldsymbol{p}_t(\boldsymbol{x}_n)$ of destination node t is calculated using the cross-entropy function $H(\boldsymbol{p}_{\hat{y}_n}, \boldsymbol{p}_t(\boldsymbol{x}_n))$. $H(\boldsymbol{p}_{\hat{y}_n}, \boldsymbol{p}_t(\boldsymbol{x}_n))$ can be decomposed into the sum of KL divergence and entropy as follows:

$$
\begin{aligned}
H(\boldsymbol{p}_{\hat{y}_n}, \boldsymbol{p}_t(\boldsymbol{x}_n)) &= KL(\boldsymbol{p}_{\hat{y}_n}||\boldsymbol{p}_t(\boldsymbol{x}_n)) + H(\boldsymbol{p}_{\hat{y}_n}, \boldsymbol{p}_{\hat{y}_n}) \\
&= KL(\boldsymbol{p}_{\hat{y}_n}||\boldsymbol{p}_t(\boldsymbol{x}_n)).
\end{aligned}
\tag{1}
$$

Here, since $\boldsymbol{p}_{\hat{y}_n}$ is a one-hot vector, its entropy $H(\boldsymbol{p}_{\hat{y}_n}, \boldsymbol{p}_{\hat{y}_n})$ is zero. Therefore, the loss between the label and the output can also be represented by the KL divergence in the same way as the loss between the node outputs. In the following, $\boldsymbol{p}_{\hat{y}_n}$ is denoted by $\boldsymbol{p}_0(\boldsymbol{x}_n)$.
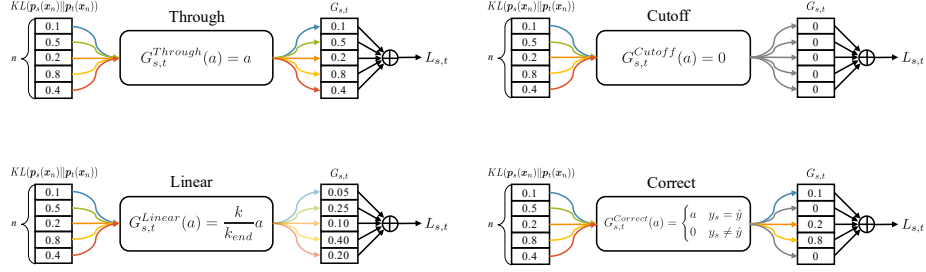
Fig. 3: Illustration of four types of gates.

$L_{s,t}$ represents the loss function used when knowledge is propagated from the source node m_s to the destination node m_t, which is defined by

$$L_{s,t} = \sum_n^{|\mathcal{B}|} G_{s,t}(KL(\boldsymbol{p}_s(\boldsymbol{x}_n)||\boldsymbol{p}_t(\boldsymbol{x}_n))), \tag{2}$$

where $G_{s,t}(\cdot)$ is a gate function.

Finally, the loss function of the destination node m_t is expressed as the sum of losses for all nodes as follows:

$$L_t = \sum_{s=0, s\neq t}^{M} L_{s,t}. \tag{3}$$

3.3   Gates

If all information is transferred to the destination node from the source node throughout the entire training phase, the learning of the destination node is liable to be disrupted. We introduce a gate that controls the gradient to a destination node by weighting losses for each training sample. We define four types of gate: through gate, cutoff gate, linear gate, and correct gate, and are illustrated in Fig. 3. A through gate simply passes through the losses of each training sample without any changes.

$$G_{s,t}^{Through}(a) = a \tag{4}$$

A cutoff gate is a gate that performs no loss calculation. It can be used to cut off any edge in a knowledge transfer graph. This function is required in methods such as KD, where knowledge transfer is only performed in one direction.

$$G_{s,t}^{Cutoff}(a) = 0 \tag{5}$$

A linear gate changes its loss weighting linearly with time during training. It has a small weighting at the initial epoch, and its weighting becomes larger as

training progresses.

$$G_{s,t}^{Linear}(a) = \frac{k}{k_{end}}a \qquad (6)$$

Here, k is the number of the current iteration, and k_{end} is the total number of iteration at the end of the training.

A correct gate is a gate that only passes the losses of samples whose source node is correct. If the top-1 class number of a source node m_s is y_s, a correct gate can be expressed as

$$G_{s,t}^{Correct}(a) = \begin{cases} a & y_s = \hat{y} \\ 0 & y_s \neq \hat{y}. \end{cases} \qquad (7)$$

When the source node is not a pre-trained model, the propagation of false information can be suppressed at the initial epoch. While a linear gate weights the overall loss, a correct gate selects the samples from which the loss is calculated.

3.4   Proposed algorithm

Algorithm 1 shows how to update the network parameters of each node during training. First, all the model weights are randomly initialized unless all the gates $G_{i,t}$ corresponding to nodes m_i are cutoff gates, in which case m_i is initialized with the weights of the pre-trained model. The pre-trained model is trained only with the labels, using the same dataset as the one used for the following hyperparameter search (the details are described in Sec. 3.5). Here, m_i is frozen during training and its weights are not updated. This node performs a role being equivalent to that of the teacher network used in KD.

The losses are obtained by inputting the same samples to all nodes. Gradients are obtained from the resulting losses, and all nodes are updated simultaneously. The gradient of loss L_t obtained from Eq. (3) is back-propagated only to node m_t, and has no effect on the other nodes. In DML, after updating the weights of the first node, the training samples are input again to the updated nodes to obtain an output. The losses between every node are then recalculated from this outputs, and gradient descent is performed for the second node. These steps are repeated until every node has been updated. The drawback of DML is that this updating method causes a significant increase in computational cost as the number of nodes increases. In our proposed method, since the weights of every node are updated during a single forward calculation, it is possible to reduce the computational cost during training.

3.5   Graph optimization

We refer to an optimized node by hyperparameter search as a target node m_1, and nodes that supports training of the target node as auxiliary nodes. A target node to be optimized is specified, and the knowledge transfer graph is optimized to maximize the accuracy of this node. The hyperparameters to be optimized

---

Algorithm 1 Network parameter update

Input: Number of nodes M, number of epochs E
Initialize: Initialize all network weights, or read in the weights of a pre-trained network
    for $\_ = 1$ to E **do**
        Input the same image $\boldsymbol{x}_n$ to each network.
        Obtain the output $\boldsymbol{p}_1(\boldsymbol{x}_n), \boldsymbol{p}_2(\boldsymbol{x}_n), \cdots, \boldsymbol{p}_M(\boldsymbol{x}_n)$.
        Obtain the loss L_n according to Eq. (3).
        Obtain the update quantity of m_n from the gradient L_n.
        Update the weights of all networks.
    end for

---

are the model type of the auxiliary nodes and the gate type on each edge. The size of the search space for this optimization is $M^{(n-1)} \cdot G^{N^2}$, where N is the number of nodes, M is the number of model types, and G is the number of gate types. For example, if $N = 3$, $M = 3$, and $G = 4$, there are over one million patterns.

We used the Asynchronous Successive Halving Algorithm (ASHA) [32] as the hyperparameter optimization method. First, using D GPU servers, we randomly create a knowledge transfer graph with D servers and perform distributed asynchronous learning. In each knowledge transfer graph, the accuracy of the target node is evaluated using validation set at epochs $1, 2, 4, \cdots, 2^k$. If this accuracy is in the lower 50% of all the accuracy values evaluated in the past, the graph is abandoned and training is performed again after generating a new graph. This process is repeated until the total number of trials reaches T. ASHA can achieve improvements in terms of both temporal efficiency and accuracy by performing a random search with active early termination in a parallel distributed environment. We performed optimization with $D = 30$ and $T = 1500$.

4   Experiments

We performed experiments to determine the efficacy of knowledge transfer graphs searched by ASHA. We describe the graphs visualization in Sec. 4.2, comparison to conventional methods in Sec. 4.3, investigation of the performance of a target node when the graph lacks diversity in Sec. 4.4 and evaluation of graph transferability between different datasets in Sec. 4.5.

4.1   Experimental setting

Datasets We used the CIFAR-10, CIFAR-100 [33], and Tiny-ImageNet [34], which are typically used for general object recognition. CIFAR-10 and CIFAR-100 consist of 50,000 images for training and 10,000 images for testing. Both datasets consist of images with dimensions of 32×32 pixels and include labels for 10 and 100 classes, respectively. Data augmentation was performed by processing the training images with 4-pixel padding (reflection), random cropping, and

random flipping. Data augmentation was not applied to the test images. For optimizing graphs, we randomly split the training samples into 10,000 samples as the validation set and 40,000 samples as the training set. The Tiny-ImageNet consists of 100,000 training images and 10,000 test images sampled from the ImageNet [35]. This dataset consists of images with dimensions of 64×64 pixels and labels for 200 classes. The data augmentation settings were the same as those for the CIFAR datasets. For optimizing graphs, we randomly split the training samples into 10,000 samples as the validation set and 90,000 samples as the training set.

Models We used three networks: ResNet32, ResNet110 [36], and Wide ResNet 28-2 [37]. Table 1 shows the accuracy achieved when each model was trained with supervised labels only. However, when training with Tiny-ImageNet, since the images are larger in size, the stride of the initial convolution layer was set to 1.

Implementation details For the optimization algorithm, we used SGD and Nesterov momentum in all experiments. The initial learning rate was 0.1, the momentum was 0.9, and the batch size was 64. When training on CIFAR, the learning rate was reduced to one tenth every 60 epochs, for a total of 200 epochs. When training on the Tiny-ImageNet, the learning rate was reduced to one tenth at the 40th, 60th, and 70th epochs, for a total of 80 epochs. The reported accuracy values with test set are averaged over five trials with a fixed graph structure implemented after obtaining the optimized graph. The standard deviation over each set of five trials is also shown. Our experiments were implemented using the Pytorch framework [38] for deep learning and the Optuna framework [39] for hyperparameter searching. The computations were performed using 90 Quadro P5000 servers. Our implementation is available at `https://github.com/somaminami/DCL`.

4.2   Visualization of graphs

Figure 4 shows the visualization of the knowledge transfer graphs with two to seven nodes optimized on CIFAR-100. For all numbers of nodes, the target node had much better accuracy than that in individual learning (see Tab. 1). The

Table 1: Accuracy of vanilla models. Mean and standard deviation of single network accuracies on test set.

Model	CIFAR-10	CIFAR-100	TinyImageNet
ResNet32	92.99 ± 0.28	70.71 ± 0.39	52.89 ± 0.18
ResNet110	94.01 ± 0.28	72.59 ± 0.54	55.49 ± 0.55
Wide ResNet 28-2	94.40 ± 0.07	74.60 ± 0.38	58.60 ± 0.25

(a) 2 nodes (72.93%)

(b) 3 nodes (73.58%)

(c) 4 nodes (74.26%)

(d) 5 nodes (74.31%)
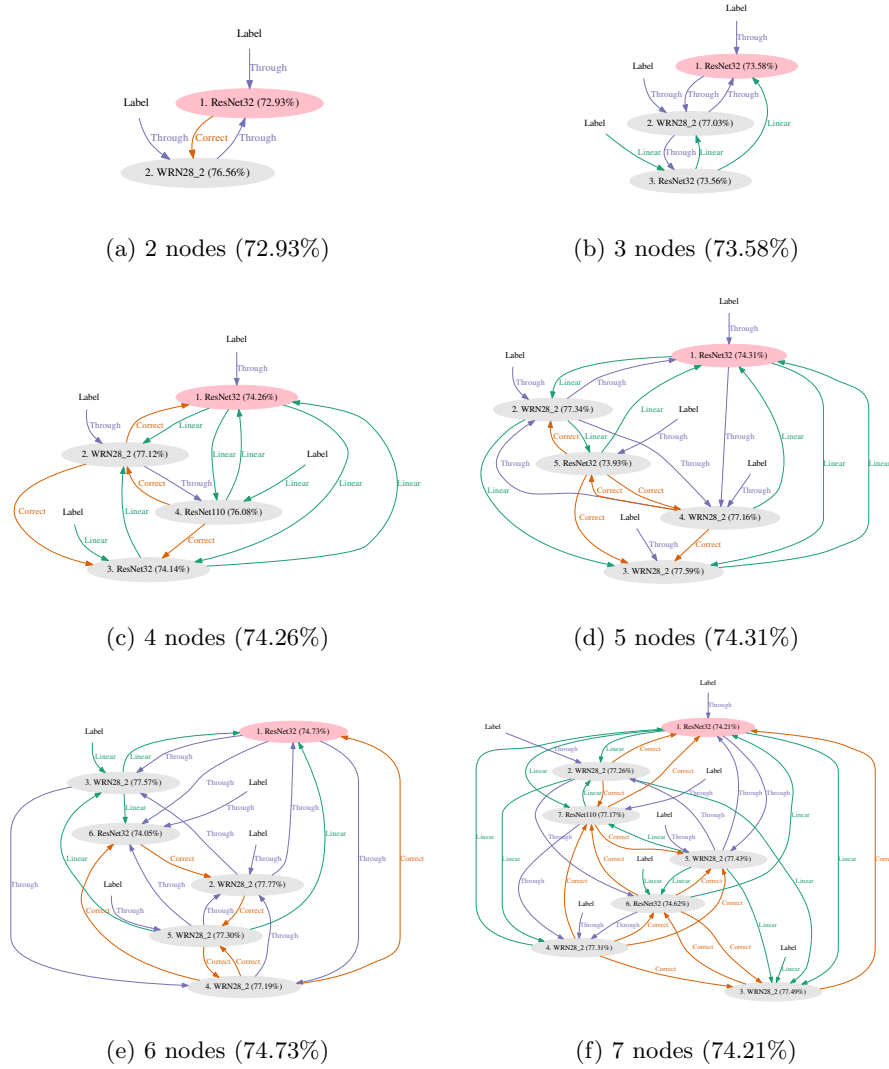
(e) 6 nodes (74.73%)

(f) 7 nodes (74.21%)

Fig. 4: Knowledge transfer graph optimized on CIFAR-100. Red node is the target node, and "Label" represents supervised labels. At each edge, the selected gate is shown, exclusive of cutoff gate. Numbers in parentheses show the accuracy achieved in one out of five trials.

accuracy of nodes other than the target node was also improved. We found that ResNet32 and ResNet110 were selected as the nodes of top-1 graphs as well as the highest performance Wide ResNet 28-2, and the performance of the target node

Table 2: Comparison with conventional methods on CIFAR-100. "*" denotes a pre-trained model. T is a temperature parameter. "**" denotes a value cited from the paper.

Method	Accuracy (Node 1)	Node 1	Node 2	Node 3	Node 4
Vanilla	70.71 ± 0.39	ResNet32	–	–	–
DML [10]	72.00 ± 0.44	ResNet32	ResNet32	–	–
KD ($T = 2$) [9]	71.88 ± 0.78	ResNet32	WRN28-2*	–	–
DML [10]	72.71 ± 0.18	ResNet32	WRN28-2	–	–
Ours	**72.88** ± 0.41	ResNet32	WRN28-2	–	–
DML [10]	72.09 ± 0.43	ResNet32	ResNet32	ResNet32	–
DML [10]	72.89 ± 0.21	ResNet32	WRN28-2	ResNet32	–
Ours	**73.46** ± 0.28	ResNet32	WRN28-2	ResNet32	–
DML [10]	72.76 ± 0.35	ResNet32	ResNet32	ResNet32	ResNet32
Song [14]	73.68** ± 0.26	(4×ResNet32 with shared intermediate layers)			
ONE [15]	73.42** ± N/A	(4×ResNet32 with shared intermediate layers)			
DML [10]	72.87 ± 0.49	ResNet32	WRN28-2	ResNet32	ResNet110
Ours	**74.06** ± 0.34	ResNet32	WRN28-2	ResNet32	ResNet110

tended to improve when the number of nodes was increased. Our quantitative evaluation is discussed in Sec. 4.4.

4.3 Comparison with conventional methods

Table 2 compares the performance of the proposed and conventional methods on CIFAR-100. "Ours" shows the results of the proposed method for optimized graphs with two, three, or four nodes. "KD [9]" uses a pre-trained Wide ResNet 28-2 network as a teacher, and sets the temperature parameter to $T = 2$. In "DML [10]" using over three nodes, all student networks have the same architecture. Since the proposed method chooses which model to use as a hyper parameter, it is possible to select the optimal combination of models. In "Song *et al.* [14]" and "ONE [15]", the intermediate layers of multiple networks are shared during training. Then, only layers that are close to the output layer are branched, and the ensemble output of the branched output layers is used as a teacher.

Compared with the results of DML having the same nodes with the proposed method, the proposed method outperforms the accuracy. The result of Song *et al.* [14] is close to the proposed method in the case of four nodes. Because their method shares the intermediate layers as described above, their method could acquire parameters to extract more representative features. The proposed method achieved the best result, although it does not share the intermediate layers updates parameters using only the gradients computed from the loss of auxiliary nodes and teacher label. Therefore, transferring knowledge from an intermediate layer could improve the performance of the target node, which is one of our future works.
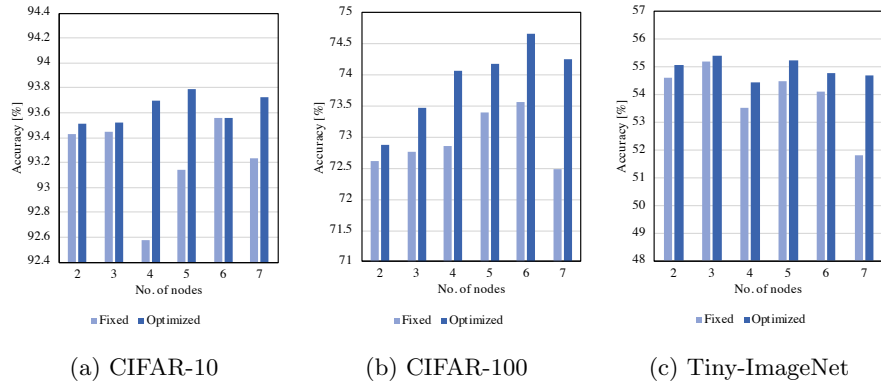
(a) CIFAR-10   (b) CIFAR-100   (c) Tiny-ImageNet

Fig. 5: Results of optimization on various datasets. ResNet32 was used as the target node. "Fixed" indicates all gates are through gates. "Optimized" indicates they have been optimized.

4.4   Comparison with graphs lacking diversity

Figure 5 shows the accuracy of target nodes in graphs searched on CIFAR-10, CIFAR-100, and Tiny-ImageNet. The comparison is a non-diverse graph, where each edge has only a through gate and each node is the same model as that of the graph, which is similar to the conventional unidirectional method [10].

The proposed method achieved higher accuracy than the comparative method in every condition, thus demonstrating the importance of using gates to control the gradient. Moreover, the optimized graphs tended to improve the accuracy when the number of nodes was increased in CIFAR-100. The fixed gates method has the same loss function on all the edges, making it difficult to generate diversity even when the number of nodes is increased.

In our experiments, due to the limitation of computational resources, we ran only 1,000 trials for searching the knowledge transfer graphs. This may not be sufficient because the search space exponentially increases with the number of nodes. Moreover, if we searched on a larger number of trials, it might be possible to acquire a better knowledge transfer graph than we discovered. We will explore this possibility in future work.

4.5   Graph transferability

We investigated the generalization ability of graphs on different datasets. Table 3 shows accuracies of the networks trained on CIFAR-100, where the graphs are searched on CIFAR-10 or CIFAR-100. CIFAR-10, which is a 10-class dataset consisting of images of vehicles and animals, has a different distribution from CIFAR-100, which is a 100-class dataset featuring plants, insects, furniture, etc..

The graphs searched on CIFAR-10 achieved the comparable performance as those searched on CIFAR-100. The results indicate that the knowledge transfer

Table 3: Accuracy of reused graphs optimized on another dataset. Graphs are trained on CIFAR-100, where graphs are searched on CIFAR-10 or CIFAR-100. Target node is ResNet32. Red/Blue indicate best and second best results.

No. of nodes	Fixed to through gate	Searched on different dataset (CIFAR-10)	Searched on same dataset (CIFAR-100)
2	72.62 ± 0.33	72.50 ± 0.33	72.88 ± 0.41
3	72.77 ± 0.26	73.63 ± 0.18	73.46 ± 0.28
4	72.86 ± 0.44	73.76 ± 0.25	74.06 ± 0.34
5	73.40 ± 0.15	74.62 ± 0.24	74.18 ± 0.21

graph can be reused to different dataset. As such, the reused graphs can greatly reduce the computational cost, since the searching process can be omitted.

5   Conclusion and Future Work

In this paper, we propose a new learning method for more flexible and diverse combinations of knowledge transfer using a novel graph representation called knowledge transfer graph. The graph provides a unified view of the knowledge transfer and has the potential to represent diverse knowledge transfer patterns. We also propose four gate functions that can deliver diverse combinations of knowledge transfer. Searching the graph structure, we discovered remarkable graphs that achieved significant performance improvements. We searched graphs over 1,000 trials, but the actual search space is much larger. A more exhaustive search will be the focus of future work.

Since our proposed method defines nodes as individual networks, it only transfers knowledge from the output layers of these networks. Future work will include knowledge transfer from an intermediate layer. It should also be possible to perform knowledge transfer using the ensemble inference of multiple networks. Other interesting possibilities include the introduction of an encoder/decoder model, and the use of multitasking.

Acknowledgement

References

1. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: IEEE conference on computer vision and pattern recognition. (2017) 4700–4708

2. Han, D., Kim, J., Kim, J.: Deep pyramidal residual networks. IEEE Conference on Computer Vision and Pattern Recognition (2017)
3. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition. (2017)
4. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: IEEE Conference on Computer Vision and Pattern Recognition. (2018)
5. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: International Conference on Learning Representations. (2017)
6. Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.J., Fei-Fei, L., Yuille, A., Huang, J., Murphy, K.: Progressive neural architecture search. In: European Conference on Computer Vision. (2018) 19–34
7. Pham, H., Guan, M., Zoph, B., Le, Q., Dean, J.: Efficient neural architecture search via parameters sharing. In: International Conference on Machine Learning. (2018) 4095–4104
8. Liu, H., Simonyan, K., Yang, Y.: DARTS: Differentiable architecture search. In: International Conference on Learning Representations. (2019)
9. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. NIPS Deep Learning and Representation Learning Workshop (2015)
10. Zhang, Y., Xiang, T., Hospedales, T.M., Lu, H.: Deep mutual learning. In: IEEE Conference on Computer Vision and Pattern Recognition. (2018)
11. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: Fitnets: Hints for thin deep nets. In: International Conference on Learning Representations. (2015)
12. Furlanello, T., Lipton, Z., Tschannen, M., Itti, L., Anandkumar, A.: Born again neural networks. In: International Conference on Machine Learning. Volume 80 of Proceedings of Machine Learning Research., Stockholmsmässan, Stockholm Sweden, PMLR (2018) 1607–1616
13. Mirzadeh, S.I., Farajtabar, M., Li, A., Ghasemzadeh, H.: Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher. arXiv preprint arXiv:1902.03393 (2019)
14. Song, G., Chai, W.: Collaborative learning for deep neural networks. In: Advances in Neural Information Processing Systems. (2018) 1837–1846
15. Lan, X., Zhu, X., Gong, S.: Knowledge distillation by on-the-fly native ensemble. In: Advances in Neural Information Processing Systems. (2018) 7527–7537
16. Ke, Z., Wang, D., Yan, Q., Ren, J., Lau, R.W.: Dual student: Breaking the limits of the teacher in semi-supervised learning. In: Proceedings of the IEEE International Conference on Computer Vision. (2019)
17. Cho, J.H., Hariharan, B.: On the efficacy of knowledge distillation. In: International Conference on Computer Vision. (2019)
18. Yim, J., Joo, D., Bae, J., Kim, J.: A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In: IEEE Conference on Computer Vision and Pattern Recognition. (2017) 4133–4141
19. Zagoruyko, S., Komodakis, N.: Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In: International Conference on Learning Representations. (2017)
20. Heo, B., Kim, J., Yun, S., Park, H., Kwak, N., Choi, J.Y.: A comprehensive overhaul of feature distillation. In: International Conference on Computer Vision. (2019)
21. Park, W., Kim, D., Lu, Y., Cho, M.: Relational knowledge distillation. In: IEEE Conference on Computer Vision and Pattern Recognition. (2019)

22. Yu, L., Yazici, V.O., Liu, X., Weijer, J.v.d., Cheng, Y., Ramisa, A.: Learning metrics from teachers: Compact networks for image embedding. In: IEEE conference on Computer Vision and Pattern Recognition. (2019)
23. Liu, Y., Cao, J., Li, B., Yuan, C., Hu, W., Li, Y., Duan, Y.: Knowledge distillation via instance relationship graph. In: IEEE Conference on Computer Vision and Pattern Recognition. (2019)
24. Chen, G., Choi, W., Yu, X., Han, T., Chandraker, M.: Learning efficient object detection models with knowledge distillation. In: Advances in Neural Information Processing Systems. (2017) 742–751
25. Chen, Y., Li, W., Van Gool, L.: Road: Reality oriented adaptation for semantic segmentation of urban scenes. In: IEEE Conference on Computer Vision and Pattern Recognition. (2018) 7892–7901
26. Oord, A.v.d., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G.v.d., Lockhart, E., Cobo, L.C., Stimberg, F., et al.: Parallel wavenet: Fast high-fidelity speech synthesis. arXiv preprint arXiv:1711.10433 (2017)
27. Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., Zecchina, R.: Entropy-sgd: Biasing gradient descent into wide valleys. International Conference on Learning Representations (2017)
28. Pereyra, G., Tucker, G., Chorowski, J., Kaiser, Ł., Hinton, G.: Regularizing neural networks by penalizing confident output distributions. International Conference on Learning Representations (2017)
29. Sun, D., Yao, A., Zhou, A., Zhao, H.: Deeply-supervised knowledge synergy. In: IEEE conference on computer vision and pattern recognition. (2019)
30. Anil, R., Pereyra, G., Passos, A., Ormandi, R., Dahl, G.E., Hinton, G.E.: Large scale distributed neural network training through online distillation. arXiv preprint arXiv:1804.03235 (2018)
31. Zhang, X., Luo, H., Fan, X., Xiang, W., Sun, Y., Xiao, Q., Jiang, W., Zhang, C., Sun, J.: Alignedreid: Surpassing human-level performance in person re-identification. arXiv preprint arXiv:1711.08184 (2017)
32. Li, L., Jamieson, K., Rostamizadeh, A., Gonina, E., Hardt, M., Recht, B., Talwalkar, A.: Massively parallel hyperparameter tuning. arXiv preprint arXiv:1810.05934 (2018)
33. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical report, Citeseer (2009)
34. : Tiny imagenet visual recognition challenge. `https://tiny-imagenet.herokuapp.com/` (2015)
35. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision **115** (2015) 211–252
36. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE conference on computer vision and pattern recognition. (2016) 770–778
37. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: British Machine Vision Conference, BMVA Press (2016) 87.1–87.12
38. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in neural information processing systems. (2019) 8026–8037
39. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25th

ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.
(2019) 2623–2631