# KNOWLEDGE TRANSFER GRAPH FOR DEEP COLLABORATIVE LEARNING

A PREPRINT

Soma Minami, Tsubasa Hirakawa, Takayoshi Yamashita, Hironobu Fujiyoshi Chubu University 1200 Matsumotocho, Kasugai, Aichi, Japan {minami@mprg.cs, hirakawa@mprg.cs, takayoshi@isc, fujiyoshi@isc}.chubu.ac.jp

#### ABSTRACT

Knowledge transfer among multiple networks using their outputs or intermediate activations have evolved through extensive manual design from a simple teacher-student approach (knowledge distillation) to a bidirectional cohort one (deep mutual learning). The key factors of such knowledge transfer 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 new graph-based approach for more flexible and diverse combinations of knowledge transfer. To achieve the knowledge transfer, 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 are introduced into loss functions. The four gates, which control the gradient, can deliver diverse combinations of knowledge transfer methods than a manually designed one. Experimental results on the CIFAR-10, -100, and Tiny-ImageNet datasets show that the proposed method achieved significant performance improvements and was able to find remarkable graph structures.

## 1 Introduction

Deep neural networks have accomplished significant progress by designing their internal structure (e.g., a network's module [12, 8, 29, 11] and architecture search [36, 17, 24, 18]). The performance of existing networks can be further improved by knowledge transfer among multiple networks, such as knowledge distillation (KD) [10] and deep mutual learning (DML) [35], 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 [10, 25, 7, 20, 35, 27, 15, 13], including the simple teacher-student approach [10], self-distillation [7], an intermediation by teacher assistant [20], and the bidirectional cohort approach [35]. The key factors 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 [35, 20, 27, 15]. Cho *et al.* [6] also pointed out that larger models do not often make better teachers. The methods of conventional knowledge transfer have explored only 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 study, 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



Figure 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.

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, compared to the vanilla model, our optimized graph achieved accuracy improvements of 1.05% with CIFAR-10, 4.00% with CIFAR-100, and 2.62% with Tiny-ImageNet. On CIFAR-100, the graphs outperformed a conventional method.
- We verified the generalization of optimized graphs that had been searched on a different dataset from the one used for optimization. In experiments using CIFAR-10, CIFAR-100, and Tiny-ImageNet, we show that the optimized graphs can be reused on another dataset.

# 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.* [10] 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.* [7] 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.* [20] 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, e.g. hint [25], flow of activations between layers [30], and attention map [33]. [22, 31, 19] transfer mutual relations of data samples in a mini-batch. Distillation has been applied to object detection [4], domain adaptation [5], text-to-speech [21], etc.



Figure 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.* [35], 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 [3, 23]. 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 [27, 15], and by sharing the intermediate layers of these networks [27, 15, 28]. DML has been applied to large scale distributed training [2] and re-identification [34]. Dual student [13] 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 *i*th 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 *n*th sample  $x_n$  and the label  $\hat{y}_n$  is represented as  $\mathcal{B} = \{x_n, \hat{y}_n\}_{n=1}^N$ , 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  $KL(p_s(x_n)||p_t(x_n))$ . Here,  $p_s$  and  $p_t$  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  $p_{\hat{y}_n}$ , the loss between  $p_{\hat{y}_n}$  and the output  $p_t(x_n)$  of destination node t is calculated using the cross-entropy function  $H(p_{\hat{y}_n}, p_t(x_n))$ .  $H(p_{\hat{y}_n}, p_t(x_n))$  can be decomposed into the

sum of KL divergence and entropy as follows:

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

Here, since  $p_{\hat{y}_n}$  is a one-hot vector, its entropy  $H(p_{\hat{y}_n}, p_{\hat{y}_n})$  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,  $p_{\hat{y}_n}$  is denoted by  $p_0(x_n)$ .

 $L_{s,t}$  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

$$L_{s,t} = \sum_{n}^{|\mathcal{B}|} G_{s,t}(KL(\boldsymbol{p}_s(\boldsymbol{x}_n)||\boldsymbol{p}_t(\boldsymbol{x}_n))),$$
(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:

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

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

Here, k is the cumulative number of updates, and  $k_{end}$  is the number of updates at the end of 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

$$G_{s,t}^{Correct}(a;\hat{y},y_s) = \delta_{\hat{y},y_s} \cdot a.$$
(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 Network optimization

Algorithm 1 shows how to update the network 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 hyperparameter search. 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

<b>Input:</b> Number of nodes $M$ , number of epochs $E$
<b>Initialize:</b> Initialize all network weights, or read in the weights of a pre-trained network
for $\_ = 1$ to E do
Input the same image to each network $m_n$ , and obtain the output $p_n$ .
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

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 a node that supports training of the target node as an auxiliary node. 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 are the model type of the auxiliary node and the gate type on each edge. The size of the search space for this optimization is  $M^{(n-1)} \cdot G^{N^2}/2$ , 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) [16] 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 verification data at epochs  $1, 2, 4, \dots, 2^k$ . 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.

## 4.1 Experimental setting

**Dataset:** We used the CIFAR-10, CIFAR-100 [14], and Tiny-ImageNet [1] datasets, 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 verification. Both datasets consist of images with dimensions of  $32 \times 32$  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 verification images. The Tiny-ImageNet dataset consists of 100,000 training images and 10,000 verification images sampled from the ImageNet [26] dataset. This dataset consists of images with dimensions of  $64 \times 64$  pixels and labels for 200 classes. The data augmentation settings were the same as those for the CIFAR datasets.

**Models:** We used three networks: ResNet32, ResNet110 [9], and Wide ResNet 28-2 [32]. 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 are averaged over five trials with a fixed graph structure implemented after obtaining the optimized



Figure 3: Optimized knowledge transfer graph. Red node is the target node, blue node is the pre-trained 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.

Model	Accuracy [%]
ResNet32	$70.71\pm0.39$
ResNet110	$72.59\pm0.54$
Wide ResNet 28-2	$74.60\pm0.38$

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

graph. The standard deviation over each set of five trials is also shown. Our experiments were implemented using the Pytorch framework for deep learning and the Optuna framework for hyperparameter searching. The computations were performed using 90 Quadro P5000 servers. Our implementation is available at {the URL of our github code: It will be available at camera ready.}

## 4.2 Optimized knowledge transfer graphs

Figure 3 shows the visualization of the knowledge transfer graphs with two to seven nodes optimized on CIFAR-100. The graph for two nodes (Fig. 3a) is similar to the conventional bidirectional method, where only the edge from ResNet32 to Wide ResNet 28-2 is the correct gate, and the other edges are the through gate. From the small network (ResNet32) to the large network (Wide ResNet), only the correct outputs that match the supervised labels are transferred. The graph for three nodes (Fig. 3b) is a fusion of unidirectional and bidirectional knowledge transfer. At the beginning of training, the target node (node 1) imitates only the pre-trained node 3, which is a unidirectional method, because the linear gate and correct gate deactivate the specific edges between the target node and the labels and between the target node and node 2. As training progresses, linear and correct gates activate those edges gradually, resulting in unidirectional and bidirectional knowledge transfer. In the graph for four nodes (Fig. 3c), two of them are ResNet32 and the other two are Wide ResNet28-2. Knowledge of supervised labels is not directly transferred to node 2, and the node acts as an intermediary for other nodes, like a teacher assistant [20].

For all numbers of nodes, the target node had much better accuracy than that in individual learning (see Tab. 1). The 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 tended to improve when the number of nodes was increased. Our quantitative evaluation is discussed in Sec. 4.4.

Method	Accuracy (Node 1)	Node 1	Node 2	Node 3	Node 4	TeCost	TrCost
Vanilla	$70.71\pm0.39$	ResNet32	_	_	_	70.2	70.2
DML [35]	$72.00\pm0.44$	ResNet32	ResNet32	-	-	70.2	140.5
KD(T = 2)[10]	$71.88 \pm 0.78$	ResNet32	WRN28-2*	-	_	70.2	286.8
DML [35]	$72.71\pm0.18$	ResNet32	WRN28-2	-	_	70.2	286.8
Ours	$\textbf{72.88} \pm 0.41$	ResNet32	WRN28-2	-	_	70.2	286.8
DML [35]	$72.09\pm0.43$	ResNet32	ResNet32	ResNet32	_	70.2	210.8
DML [35]	$72.20\pm0.47$	ResNet32	ResNet110	ResNet32	_	70.2	398.2
Ours	$\textbf{73.46} \pm 0.42$	ResNet32	ResNet110	ResNet32*	_	70.2	398.2
DML [35]	$72.76\pm0.35$	ResNet32	ResNet32	ResNet32	ResNet32	70.2	281.0
Song <i>et al</i> . [27]	$73.68^{**} \pm 0.26$	(4×ResN	Vet32 with shar	ed intermedia	te layers)	70.2	160.9
ONE [15]	$73.42^{**} \pm \text{N/A}$	(4×ResN	let32 with shar	ed intermedia	te layers)	70.2	138.1
DML [35]	$73.32\pm0.55$	ResNet32	WRN28-2	WRN28-2	ResNet32	70.2	573.6
Ours	$74.34 \pm 0.32$	ResNet32	WRN28-2	WRN28-2	ResNet32	70.2	573.6

**Table 2: Comparison with conventional methods**. "\*" denotes a pre-trained model. *T* is a temperature parameter. "\*\*" denotes a value cited from the paper. TeCost and TrCost represent the computational complexity (MFLOPs) for testing and training, respectively.

No. of nodes	Gates	CIFAR-10	CIFAR-100	Tiny-ImageNet
1	-	$93.12\pm0.27$	$70.71\pm0.39$	$53.18\pm0.08$
2	Fixed (Through)	$93.25\pm0.50$	$72.47\pm0.38$	$54.93\pm0.29$
2	Optimized	$93.65 \pm 0.14$	$72.88 \pm 0.41$	$54.69\pm0.16$
3	Fixed (Through)	$93.53\pm0.24$	$71.88\pm0.43$	$53.78 \pm 0.78$
5	Optimized	$93.92\pm0.20$	$73.46\pm0.42$	$55.02\pm0.31$
	Fixed (Through)	$93.01\pm0.79$	$73.40\pm0.39$	$53.92\pm0.21$
4	Optimized	$93.99\pm0.27$	$74.34\pm0.32$	$\textbf{55.80} \pm 0.26$
5	Fixed (Through)	$93.61 \pm 0.23$	$73.40\pm0.28$	$52.12\pm0.30$
5	Optimized	$94.14 \pm 0.16$	$74.54\pm0.59$	$55.30\pm0.16$
6	Fixed (Through)	$93.84\pm0.39$	$73.85\pm0.45$	$49.37 \pm 1.70$
0	Optimized	$94.17 \pm 0.21$	$74.22\pm0.22$	$55.16\pm0.19$
7	Fixed (Through)	$93.75\pm0.27$	$73.53\pm0.27$	$53.10\pm0.44$
1	Optimized	$94.07\pm0.14$	$\textbf{74.71} \pm 0.23$	$54.78\pm0.36$

 Table 3: 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.

#### 4.3 Comparison with conventional methods

Table 2 compares the results of the proposed and conventional methods. "Ours" shows the results of the proposed method for optimized graphs with two, three, or four nodes. "KD [10]" uses a pre-trained Wide ResNet 28-2 network as a teacher, and sets the temperature parameter to T = 2. In "DML [35]" 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.* [27]" 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.

The model learned by the optimized knowledge transfer graph achieved better accuracy than KD, DML, and the latest proposed method.

#### 4.4 Comparison with graphs lacking diversity

Table 3 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 [35].

The proposed method achieved higher accuracy than the comparative method in almost every condition, thus demonstrating the importance of using gates to control the gradient. Moreover, the optimized graphs tended to improve the

	evaluated on CIFAR-100			evaluated on Tiny-ImageNet		
	fixed to	searched on	searched on	fixed to	searched on	searched on
No. of nodes	through gate	different dataset	same dataset	through gate	different dataset	same dataset
ι   Ι	through gate	(CIFAR-10)	(CIFAR-100)	unrough gate	(CIFAR-10)	(Tiny-ImageNet)
2	$72.47 \pm 0.38$	$72.50\pm0.33$	$\textbf{72.88} \pm 0.41$	<b>54.93</b> ± 0.29	$52.79 \pm 0.31$	$54.69\pm0.16$
3	$71.88 \pm 0.43$	$\textbf{73.63} \pm 0.18$	$73.46\pm0.42$	$53.78 \pm 0.78$	$54.21 \pm 0.44$	$\textbf{55.02} \pm 0.31$
4	$73.40 \pm 0.39$	$73.76\pm0.25$	$\textbf{74.34} \pm 0.32$	$53.92 \pm 0.21$	$54.50\pm0.36$	$\textbf{55.80} \pm 0.26$
5	$73.40 \pm 0.28$	$\textbf{74.62} \pm 0.24$	$74.54\pm0.59$	$52.12 \pm 0.30$	$54.42\pm0.14$	$\textbf{55.30} \pm 0.16$

 Table 4: Accuracy rate of reused graphs optimized on another dataset. Graphs are trained on CIFAR-100 and Tiny-ImageNet, where graphs are searched on CIFAR-10. Target node is ResNet32.



Figure 4: Visualization of optimized graphs with three types of target node. Graphs whose target node is ResNet32, ResNet110, or Wide ResNet 28-2 are searched on CIFAR-100.

accuracy when the number of nodes was increased in all datasets. 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 1,500 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 transfer to different dataset

We investigated the generalization of graphs on different datasets. Table 4 shows the accuracy of graphs trained on CIFAR-100 and Tiny-ImageNet, where the graphs are searched on CIFAR-10. 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., and from Tiny-ImageNet.

When evaluated on CIFAR-100, the graphs searched on CIFAR-10 achieved the same performance as those searched on CIFAR-100. When evaluated on Tiny-ImageNet, while the performance of the graphs searched on CIFAR-10 was lower than that of graphs searched on Tiny-ImageNet, it was higher than that of non-diverse graphs whose gates were all through gates, which is a conventional bidirectional method. These results indicate that the knowledge transfer graph can be reused not only for datasets with relatively close distribution, such as CIFAR-100, but also for those with very different images, such as Tiny-ImageNet. As such, the reused graphs can greatly reduce the computational cost, since the searching process can be omitted.

## 4.6 Optimization of different target nodes

Table 5 shows the performances when ResNet32, ResNet110, or Wide ResNet 28-2 were set as the target node. The knowledge transfer graphs are trained on CIFAR-100 with three nodes. All the optimized graphs achieved a higher performance than the non-diverse graph, each of whose edges had only a through gate.

Figure 4 shows a visualization of the optimized graphs. We found that the three nodes in the graphs included at least one model that was different from the target node. Even when Wide ResNet 28-2, which had the highest performance, was selected as the target node, the graph included ResNet110, which had a lower performance than Wide ResNet.

Target node	Gates	Accuracy [%]
ResNet32	Fixed (Through)	$71.88\pm0.43$
Residence 2	Optimized	$\textbf{73.46} \pm 0.42$
PosNat110	Fixed (Through)	$76.08\pm0.92$
Respectivo	Optimized	$\textbf{77.63} \pm 0.22$
WDN28 2	Fixed (Through)	$77.16\pm0.10$
WIN120-2	Optimized	$\textbf{77.62} \pm 0.24$

Table 5: Accuracy of three types of target node. Graphs were optimized with different target nodes on CIFAR-100.

	Auxiliary node			
Target node	ResNet32	ResNet110	WRN28-2	
ResNet32	25 %	10 %	65 %	
ResNet110	15 %	10 %	75 %	
WRN28-2	30 %	20 %	50 %	

 Table 6: Tendency of the selected auxiliary nodes. Percentages of model type contained in the auxiliary nodes of the top-10 graphs in the experiment of Tab. 5 are shown.

This result suggests that diverse model selection is necessary to create a good knowledge transfer graph. To verify this hypothesis, we investigated which model had been selected for each auxiliary node of the high performance graphs. Table 6 shows the percentages of model type contained in the auxiliary nodes of the top-10 graphs for the experiment shown in Tab. 5. These percentages show the same tendency among all evaluated target nodes. This result demonstrates that diverse model selection contributes to a high performance for collaborative learning.

# 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 1,500 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.

### References

- [1] Tiny ImageNet Visual Recognition Challenge. https://tiny-imagenet.herokuapp.com/.
- [2] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. Large scale distributed neural network training through online distillation. *arXiv preprint arXiv:1804.03235*, 2018.
- [3] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *International Conference on Learning Representations (ICLR)*, 2017.
- [4] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In Advances in Neural Information Processing Systems (NeurIPS), pages 742–751, 2017.
- [5] Yuhua Chen, Wen Li, and Luc Van Gool. Road: Reality oriented adaptation for semantic segmentation of urban scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7892–7901, 2018.
- [6] Jang-Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *International Conference on Computer Vision (ICCV)*, 2019.

- [7] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 1607–1616, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.
- [8] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016.
- [10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *NIPS Deep Learning and Representation Learning Workshop*, 2015.
- [11] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. 2018.
- [12] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 3, 2017.
- [13] Zhanghan Ke, Daoye Wang, Qiong Yan, Jimmy Ren, and Rynson W.H. Lau. Dual student: Breaking the limits of the teacher in semi-supervised learning. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.
- [14] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.
- [15] Xu Lan, Xiatian Zhu, and Shaogang Gong. Knowledge distillation by on-the-fly native ensemble. In Advances in Neural Information Processing Systems (NeurIPS), pages 7527–7537, 2018.
- [16] Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. Massively parallel hyperparameter tuning. *arXiv preprint arXiv:1810.05934*, 2018.
- [17] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *European Conference on Computer Vision* (ECCV), pages 19–34, 2018.
- [18] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations (ICLR)*, 2019.
- [19] Yufan Liu, Jiajiong Cao, Bing Li, Chunfeng Yuan, Weiming Hu, Yangxi Li, and Yunqiang Duan. Knowledge distillation via instance relationship graph. In *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), 2019.
- [20] Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher. arXiv preprint arXiv:1902.03393, 2019.
- [21] Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C Cobo, Florian Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. arXiv preprint arXiv:1711.10433, 2017.
- [22] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [23] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *International Conference on Learning Representations* (*ICLR*), 2017.
- [24] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning (ICML)*, pages 4095–4104, 2018.
- [25] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *International Conference on Learning Representations (ICLR)*, 2015.
- [26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
- [27] Guocong Song and Wei Chai. Collaborative learning for deep neural networks. In Advances in Neural Information Processing Systems (NeurIPS), pages 1837–1846, 2018.
- [28] Dawei Sun, Anbang Yao, Aojun Zhou, and Hao Zhao. Deeply-supervised knowledge synergy. In *IEEE conference* on computer vision and pattern recognition (CVPR), 2019.
- [29] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

- [30] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), pages 4133–4141, 2017.
- [31] Lu Yu, Vacit Oguz Yazici, Xialei Liu, Joost van de Weijer, Yongmei Cheng, and Arnau Ramisa. Learning metrics from teachers: Compact networks for image embedding. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [32] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, September 2016.
- [33] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations* (*ICLR*), 2017.
- [34] Xuan Zhang, Hao Luo, Xing Fan, Weilai Xiang, Yixiao Sun, Qiqi Xiao, Wei Jiang, Chi Zhang, and Jian Sun. Alignedreid: Surpassing human-level performance in person re-identification. arXiv preprint arXiv:1711.08184, 2017.
- [35] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [36] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *International Conference* on Learning Representations (ICLR), 2017.

## 6 Appendix

#### 6.1 Graph Optimization by ASHA

Figure 5 shows the error curves of each trial in searching a graph using the asynchronous successive halving algorithm (ASHA) [16]. ASHA can significantly reduce the computational complexity for searching graphs by terminating trials that are unlikely to achieve a good performance. In our experiments, pruning a trial was performed at the 1, 2, 4, 8, 16, 32, and 64th epochs.



Figure 5: Error curves when optimizing with ASHA. We optimized a graph with three nodes on Tiny-ImageNet while pruning with ASHA. Each line represents one trial. The total number of trials is 1500.

#### 6.2 Optimization on Reduced Dataset

We investigated the relationship between the accuracy of a searched top-1 graph and the number of training samples in the dataset used for searching graphs. Table 7 lists the accuracies of graphs optimized on CIFAR-10 and CIFAR-100 with a reduced number of training samples. We report the recognition rate when retraining the top-1 graphs using all training data after searching the graphs. The results show that the performance of graphs tended to decrease when we reduced the number of training samples used for optimization. This suggests that a certain number of samples are required for optimization of the knowledge transfer graph. When the reduction factor is 1/10, reducing the training samples has little impact on the performance of graphs. When the computational cost for searching is also reduced to 1/10, and the search time can be greatly reduced.

Reduction factor	No. of samples	CIFAR-10	CIFAR-100
1	50000	$94.14\pm0.16$	$74.54\pm0.59$
1/10	5000	$94.08\pm0.12$	$74.49 \pm 0.56$
1/30	1666	$93.83\pm0.04$	$74.17\pm0.36$
1/50	1000	$93.91\pm0.19$	$74.29\pm0.16$
1/70	714	$94.18\pm0.12$	$73.67\pm0.34$
1/100	500	$93.79\pm0.12$	$73.11\pm0.39$

Table 7: Accuracy of target node optimized on reduced dataset. We optimized graphs with five nodes on the dataset whose number of training samples was reduced and report the average and standard deviation of the recognition rate when the optimized graph was retrained five times using all training data.

#### 6.3 Visualization of Knowledge Transfer Graphs

Figure 6 shows the graphs that achieved high performance with three to seven nodes on the CIFAR-100 dataset. The graph shown in Fig. 6b is an example of weakly supervised learning. It does not have supervised labels, and nodes 1 and 3 learn from node 2, which is a pre-train model. In addition, this graph can be regarded as self-distillation [7] from node 2 to node 1, as these nodes have the same architecture. Node 2 (ResNet32, a small teacher) transfers its knowledge to node 3 (WRN28-2, a large student), and node 3 transfers its knowledge to node 1 (target node). This graph suggests that a network distilled by self-distillation through a larger network can obtain a high generalization ability.



