Refining Design Spaces in Knowledge Distillation for Deep Collaborative Learning

Sachi Iwata, Soma Minami, Tsubasa Hirakawa, Takayoshi Yamashita and Hironobu Fujiyoshi
Chubu University
Kasugai, Aichi, Japan
Email: {isachi, minami, hirakawa}@mprg.cs.chubu.ac.jp, {takayoshi, fujiyoshi}@isc.chubu.ac.jp

***Abstract*—Knowledge distillation is one of the most widely utilized methods to improve the performance of a model. The knowledge transfer graph has been proposed for deep collaborative learning that enables a rich diversity of bidirectional knowledge distillation. However, exploring a knowledge transfer graph is difficult due to the many potential combinations it can have, so it is not clear how accurate the resultant graphs will actually be. To address this issue, we propose a method for designing the search space with step by step and analyze the trends of graphs to design graphs with high accuracy on the basis of the acquired results. Experiments on the CIFAR-100 dataset show that we confirm that the accuracy of the best knowledge transfer graph in the search space is better than that derived using the asynchronous successive halving algorithm. We also demonstrate that the explored knowledge transfer graphs can be transferred to different datasets.**

I. Introduction

Deep convolutional neural networks are the engine of visual recognition, and it is crucial that the networks be deep and have a high performance. We can obtain high-performance networks by distilling the knowledge learned during the network learning process [1]–[6] and optimizing the architecture of neural networks [7]–[15].

We can expect to improve the accuracy of network models by transferring soft-labels among network models. Representative methods include knowledge distillation (KD) [1] and deep mutual learning (DML) [2]. KD can improve the network performance of a student network with a small number of parameters by distilling knowledge from a teacher network with a large number of parameters. DML can also improve the network performance by training multiple networks simultaneously and distilling the knowledge from each. Knowledge distillation between networks has been investigated not only for the purpose of network compression but also for improving network performance [2], [3]. In this paper, we refer to learning methods that deal with knowledge among multiple networks (e.g., KD and DML) as collaborative learning among networks.

The knowledge transfer graph (KTG) [3] is a large-scale representation of knowledge distillation for deep collaborative learning between networks. In this technique, KD and DML can be represented by a graph structure to function as a learning method that encompasses them both. In addition, the way knowledge is distilled can be controlled by using a gate function, which reduces the distillation of unnecessary knowledge. A KTG provides a unified view of knowledge transfer and has the potential to represent diverse knowledge distillation patterns. In the KTG, each node represents a network, and each edge represents a direction of knowledge distillation. By automatically optimizing nodes and edges, we can obtain the best distillation method. However, since the search space of a typical KTG is huge, it remains unclear which elements contribute to the creation of a highly accurate graph.

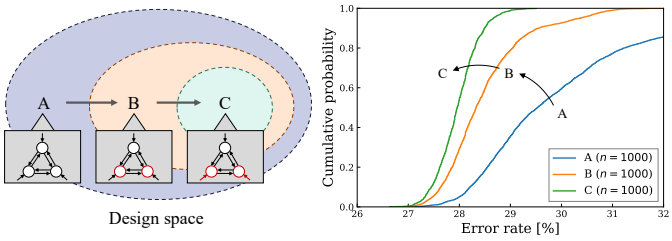


Fig. 1. **Refining knowledge transfer graph design spaces**. Left: refining design spaces. The red edges indicate where the search space is fixed. In this design, we start with an initial design space A and apply two refinement steps to yield design space B and then C. In this case, $C \subseteq B \subseteq A$. Right: empirical distribution function [16] with design space transition. The error distributions strictly improve from A to B to C. We can expect to obtain robust graphs by refining the design space of knowledge transfer graphs in this way.

In this study, we analyze what kinds of nodes and edges are used in graphs with high recognition accuracy. Our objective is to obtain an optimal knowledge distillation method by refining the design space of KTGs based on the trends obtained from the analysis. The design space refers to the search space that people design to find the explored KTGs. First, a random search of KTGs is performed to analyze the trends common to graphs with high recognition accuracy. Then, the design space of a specific KTG is manually restricted on the basis of the identified trends. These steps are repeated until the design space is sufficiently small (Fig. 1). We compare the KTG refined with our method to that derived by DML to determine whether it can obtain a design space with more graphs and a higher recognition accuracy. We also compare the accuracy of the explored graphs found by searching the entire refined design space with the results explored by using the asynchronous successive halving algorithm (ASHA) [8], a conventional search method. We then re-train the graphs

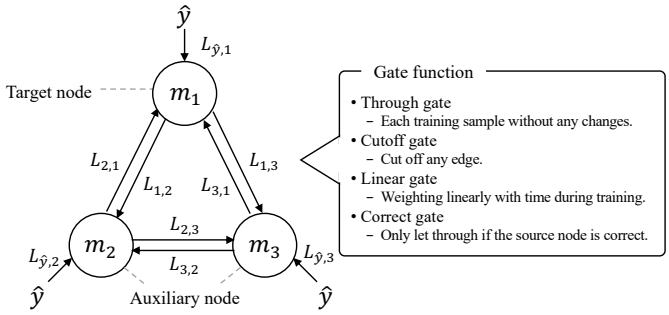


Fig. 2. **Knowledge transfer graph**

obtained by the search on a different dataset to investigate whether we can obtain a graph with generalization ability.

Our contributions are as follows.

- We explore the design space of KTGs by refining the design space. By refining the search space step by step, we can identify a design space where it is easy to explore a distillation method with high recognition accuracy.
- We analyze the KTGs in the design space to obtain an overview of the trends. Our findings show that if we have knowledge of the auxiliary nodes, we can improve the accuracy without directly using the teacher labels.
- We identify a graph with high recognition accuracy on another dataset.

II. Related Work

A. Knowledge transfer graph

One type of a combination unidirectional-bidirectional distillation is the knowledge transfer graph (KTG) method [3].

An example of KTG is shown in Fig. 2, where m refers to network models, L is a loss function, and $\hat{y}$ refers to labels. In a KTG, each network model is represented by a node, and the direction of the knowledge distillation between nodes is represented by an edge. The hyper-parameters of the KTG are the network model and the gate function. There are four types of gate functions that control loss functions: through gate, which outputs the input as it is without any change, cutoff gate, which sets the output to zero with respect to the input, linear gate, which increases the output as it learns, and correct gate, which passes the value only if the inference result is correct during learning. The target node is the one whose accuracy is to be improved, and the nodes that support the target node are the auxiliary nodes. However, for a KTG with three nodes, the number of hyper-parameter combinations (i.e., for both nodes and edges) is 1,179,648, which makes it computationally infeasible to perform the entire search in a realistic amount of time.

In this study, we propose a stepwise refinement of the design space to identify trends in KTGs and obtain graphs with a high recognition accuracy.

B. Hyper-parameter optimization

The main hyper-parameter optimization methods include a technique to suggest the next search point based on the previous search results and a technique to explore the search space.

For exploring the architecture of a network, many different hyper-parameter optimization methods have been proposed. Network architecture search methods include the tree-structured Parzen estimator approach (TPE) [9], which is a Bayesian optimization method, and NAS [7] and ENAS [10], which use reinforcement learning. DARTS [11] maps a discrete parameter space into a continuous space for optimization. BANANAS [12] and GraphBO [13] represent the network structure as a graph for optimization. There is also a simple but powerful method called random search [14] that is utilized by the successive halving algorithm (SHA) [15] and the asynchronous successive halving algorithm (ASHA) [8], both of which increase the efficiency of the search by active branch trimming. SHA uses the hyper-parameters proposed by random search to train a model for a certain period of time, and then repeats the process of training the model further, keeping the top $n\%$ of good trials. ASHA is executed in parallel asynchronously and to efficiently search for hyper-parameters with high accuracy. It achieves equal or better accuracy compared to more complex methods such as TPE, NAS, ENAS, and DARTS [17].

However, as the above search methods do not take into account the trends of the entire search space when performing optimization, they do not always arrive at a global optimization result. In other words, it is not always possible to obtain generalized features (e.g., residual and skip structures) in networks with high accuracy. Designing network design spaces [18] has been proposed as a method to discover generalized features by optimizing the search space while investigating its trends using the empirical distribution function [16]. However, this method requires a huge number of searches to refine the design space after searching.

Our solution to this problem is to focus on the trends of the search space. And we refining the design space step by step. This enables us to explore the KTG with fewer searches than when searching the initial search space.

III. Proposed Method

Since the search space for KTGs is huge, the common tendency of graphs with high accuracy is not yet clear. Therefore, instead of searching for the optimal KTG in a particular setting, we explore the entire population of KTGs. This enables us to acquire trends in the entire design space, which will help us to improve the recognition accuracy of the KTG. Our goal is to design a better KTG for visual recognition. To this end, we analyze the hyper-parameters common to highly accurate graphs and propose a method to refine the design space step by step. Fig. 3 shows the flow of the proposed method. The initial design space is the space at the beginning of the search, and by repeating the search, a design space with many more accurate graphs can be found. There are three reasons for repeating the process of refining the design space step by step, as shown below.

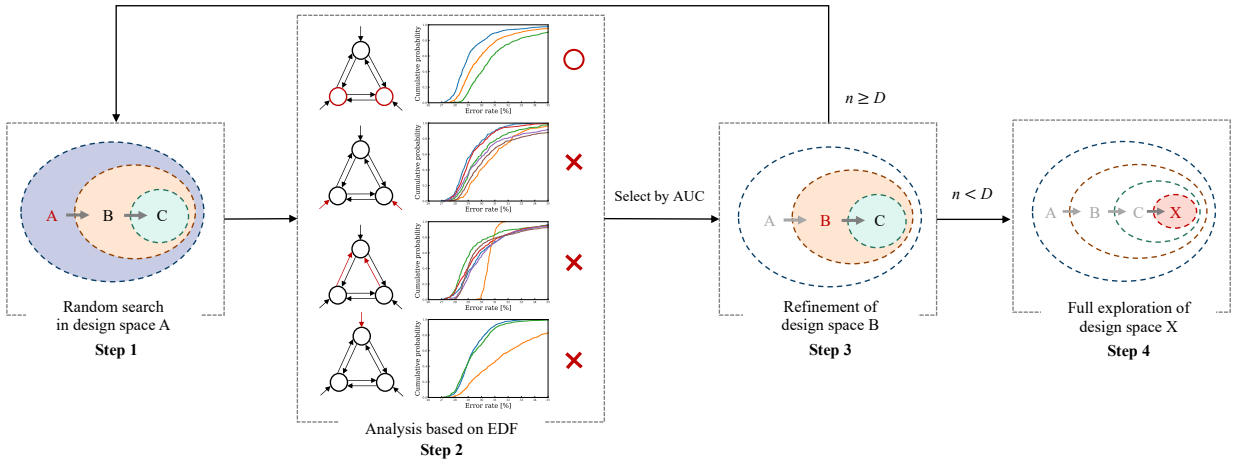


Fig. 3. **Refining the design space**. After a random search of design space A, the EDF results are plotted under four conditions. After that, the design space is refined on the basis of knowledge of others. Finally, we explore the entire design space X.

- To simplify the structure of the design space by exploring and analyzing it.
- To improve or at least maintain the design space quality by refining it step by step.
- To maintain KTG diversity by optimizing the design space.

In this method, since the design space is refined step by step, the number of searches becomes large. However, it is possible to grasp the trend of the entire design space. We can learn which elements are important for the way in knowledge distillation.

A. Tool for evaluating the design space

We use the empirical distribution function (EDF) introduced by Radosavovic *et. al.* [16] as a tool for evaluating the design space. The EDF, which is commonly used to evaluate the design spaces for KTG, is a measure of the percentage of KTGs with error rates lower than a given value of e. The EDF of n graphs with errors $\{e_i\}$ is given by

$$F(e) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}[e_i < e], \tag{1}$$

where $F(e)$ is given as the fraction of KTGs with a validation error lower than e. In deep learning, accuracy may vary depending on the number of trials. EDF can be evaluated for accuracy across the entire design space, not just the highest value. We evaluate the design space with EDF by plotting it on a graph and judging by the area under the curve. In doing so, it is possible to quantitatively evaluate whether the design space is good and explore automation. Simply put, the more KTGs in the design space that have a low error rate, the better the design space. When refining the design space, the EDF is plotted on a graph and judges a large area to be a good design space by the area under the curve. A sufficient number of searches is required to determine whether the design space is good or bad when using this index. Therefore, we reduce the exploring time of KTGs by learning for half the number of epochs in which we evaluate it.

B. Flow of exploring KTGs

The proposed method refines the design space by focusing on hyper-parameters common to graphs with a high recognition accuracy.

The flow is shown in Fig. 3. First, we refine a hyper-parameter search to find which KTG is better in design space A. Second, we analyze the design space based on EDF to discover which hyper-parameters have higher effectiveness. We assume that the area under the curve is the best design space, and select that. Third, we use the next design space as the design space refined with the selected hyper-parameters. Then, the design space is used as the next design space to be explored, and the design space is refined. This process is repeated until the number of combinations is sufficiently reduced to obtain design space X. In this method, we explore design space is refined step by step. In other words, the exploration gradually refines the hyper-parameters (i.e., auxiliary nodes and gate functions in KTGs). Consequently, we can avoid falling into a local optimum solution. Finally, we search for a better graph by exploring the entire refined design space. Then, we obtain the best graph in that design space. Algorithm 1 shows how to refine the design space and explore the KTGs. In order to analyze which hyper-parameters would have the best impact on the target node, the following conditions are set in Step 2.

- Auxiliary node
- Edge from label to auxiliary node
- Edge from label to target node
- Edge from auxiliary node to target node

IV. Experiments

We performed experiments to compare a graph without diversity (i.e., DML) by using a through gate function with the design space X restricted by the proposed method. We also

Algorithm 1 Refining design space

Require: Number of combinations in design space of KTGs n, lower limit hyper-parameters that refine the design space D, dataset M.
Manually set up a design space A.
while $n > D$ **do**
Randomly search in n and train them on M (**Step 1**).
Analyze AUC based on EDFs under four conditions (**Step 2**).
Select hyper-parameters that are common to graphs with high accuracy and manually refine the design space (**Step 3**).
n = number of combinations in refined design space
end while
Full exploration of design space X (**Step 4**).

examined whether a generalizable graph could be obtained by the proposed method by training the explored graph using another dataset.

A. Experimental setting

Dataset: We used the CIFAR-10, CIFAR-100 [19], Tiny-ImageNet [20], and CINIC [21] datasets, which are commonly used for object recognition. CIFAR-10 and CIFAR-100 respectively consist of 60,000 and 50,000 images for training and 10,000 images for testing, and contain 32×32 images and labels of 10 and 100 classes. When exploring graphs, 40,000 of the 50,000 datasets for train set are used for training, and 10,000 for testing are randomly assigned. When evaluating the graphs obtained by the search, 50,000 datasets for training and 10,000 datasets for testing are used. The training data images are subjected to data augmentation using 4-pixel padding (reflection), random cropping, and random flipping. The images of the validation data do not receive data augmentation. Tiny-ImageNet consists of 100,000 training data and 10,000 validation data sampled from the ImageNet [22] dataset. Tiny-ImageNet consists of 64×64 images and 200 classes of teacher labels. The settings for data augmentation are the same as those for the CIFAR datasets. CINIC-10 consists of 270,000 images (60,000 from the original CIFAR-10 data and the remaining from ImageNet), 90,000 images for training and 90,000 images for testing, and contain 32×32 images and labels of 10 classes. The settings for data augmentation are the same as those for the CIFAR datasets.

Models: We used ResNet32 [24] and WRN28-2 [25]. Table I lists the average accuracy and standard deviation achieved when each model was trained with supervised labels only on CIFAR-100 for five times the experiment. As we can see, WRN28-2 had the higher accuracy. When training with Tiny-ImageNet, the stride of the first convolution layer should be set to 1 because of the large image size.

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. The scheduler for the learning rate utilized cosine decay [23]. We set the lower limit of hyper-parameters that refine the design space D to 500. We trained for 100 epochs when searching design spaces A to C. We trained for 200 epochs when evaluating the graphs by the search using CIFAR-10 and CIFAR-100. We trained for 80 epochs when evaluating the graphs by the search using Tiny-Image Net. The gate functions used were through gate, cutoff gate, and linear gate. The target node was ResNet32, and the auxiliary nodes were ResNet32 and WRN28-2. Design spaces A to C were searched 2,500 times each, and design space X had five full searches. The computations were performed using 30 Quadro P5000 servers.

TABLE I
ACCURACY OF VANILLA MODELS

Model	Accuracy
ResNet32	71.59 ± 0.27
WRN28-2	75.73 ± 0.46

Since KTGs optimize nodes and edges to make the accuracy of the target node high, it is not always possible to obtain a graph with high accuracy. One example of recognition accuracy degradation is when the teacher labels are not used to train the target node. In this experiment, graphs with extremely low recognition accuracy of the target node were not used in the analysis.

B. Exploring KTG design spaces

We refine the design space to obtain design space X, which is the design space of KTGs with high accuracy.

Design space A: First, we explored design space A, which has no restrictions on the hyper-parameters used for exploration. Table II shows the AUC for the auxiliary node, including the given value and the percentage of graphs whose error rate is below the given value. n is the number of graphs acquired by the search. We can see here that, by using two WRN28-2s as auxiliary nodes, it was possible to obtain many graphs with high recognition accuracy. This suggests that the model with high recognition accuracy is expressive and contributes to the improvement of the recognition accuracy of the target node.

Design space B: Next, we explored design space B, which is the space that includes the WRN28-2 (auxiliary node) found to be most effective in the search of design space A. Table II shows the AUC of the edge from the label to the auxiliary node. We can see here that using a through gate from the label to the auxiliary node resulted in a large number of highly accurate graphs.

Design space C: Next, we explored design space C, which is the space with the most effective through gate (label to auxiliary node) identified after searching design space B. Table II shows the AUC for the edges from auxiliary node to target node. We can see here that use of a through gate or linear gate from the auxiliary node to the target node resulted in the acquisition of many graphs with high recognition accuracy. This suggests that the knowledge of each auxiliary

TABLE II
AUC OF DESIGN SPACES

Item of analysis	Node or gate	Design spaces A → B	Design spaces B → C	Design spaces C → X
Auxiliary node	ResNet32 & ResNet32	68.80	–	–
	ResNet32 & WRN28-2	69.75	–	–
	WRN28-2 & WRN28-2	**70.74**	–	–
Edge from label to auxiliary node	Through & Through	70.70	**71.69**	–
	Cutoff & Cutoff	69.35	69.62	–
	Linear & Linear	70.10	70.94	–
	Through & Linear	70.64	71.44	–
	Through & Cutoff	69.58	70.81	–
	Linear & Cutoff	68.58	68.84	–
Edge from label to target node	Through & Through	69.86	70.86	72.07
	Cutoff & Cutoff	69.61	69.62	69.61
	Linear & Linear	70.52	71.28	**72.12**
	Through & Linear	69.98	70.80	72.11
	Through & Cutoff	69.66	70.74	71.52
	Linear & Cutoff	69.67	69.76	71.23
Edge from auxiliary node to target node	Through	70.70	71.15	71.38
	Cutoff	67.67	68.63	71.47
	Linear	70.56	71.08	71.73

TABLE III
SUMMARY OF DESIGN SPACES.

Design space	Restriction	Combinations
A	None	39,366
B	$+m_2 = m_3 =$ WRN28_2	9,842
C	$+L_{\hat{y},2} = L_{\hat{y},3} =$ Through gate	1,094
X	$+L_{2,1} = L_{3,1} =$ Linear gate	121

TABLE IV
AUC IN DESIGN SPACE X.

Gate function	AUC
Through	72.36
Cutoff	**73.10**
Linear	72.88

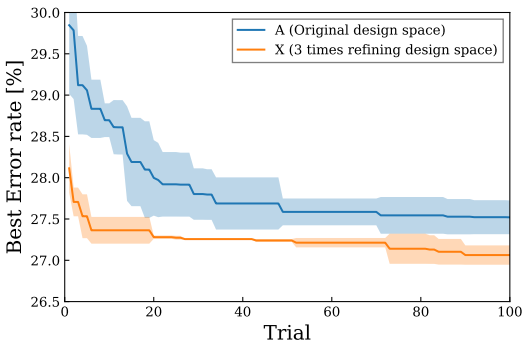


Fig. 4. **Minimum test error rate in random search of design spaces A and X**. The mean and standard deviation of three iterations of random search are plotted.

node learned through training with labels can improve the recognition accuracy of the target node.

Design space X: Design space X is the space using a linear gate (from auxiliary node to target node) identified as the most effective after searching design space C. Table III summarizes the design spaces refined by the proposed method. The number of combinations of design space X obtained by design space refinement was 61/19683 the number of combinations in the initial design space (design space A). Compared to the full search of design space A, the number of full searches in design space X can be greatly reduced.

C. Effects of refining the design space

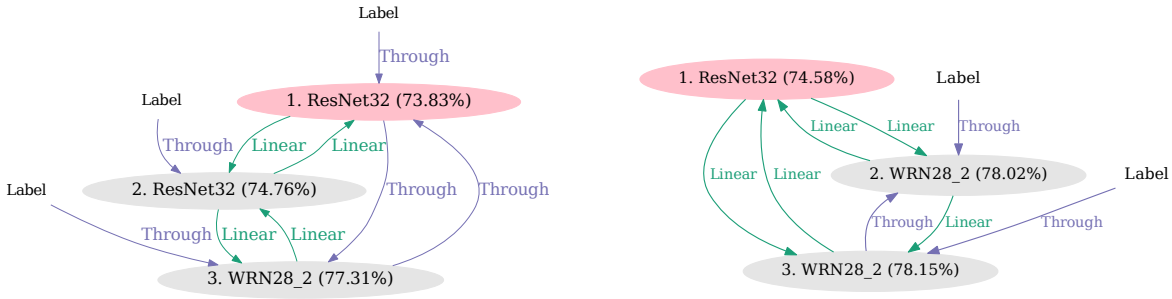
We performed 200 random searches for each of the two design spaces (with and without design space reduction) to determine which one acquired the most accurate graph faster. The results are shown in Fig. 4, where the horizontal axis shows the number of searches and the vertical axis shows the minimum error rate at the time of searching. We can see here that design space X could find graphs with a higher error rate faster. This result demonstrates that the design space acquired by the proposed method (design space X) is more likely to acquire highly accurate graphs compared to the design space in the initial stage of the search (design space A).

D. Analysis of design space X

We investigated the trends of the KTGs acquired in the final design space X. The results are shown in Table IV, where the AUCs for each type of edge from the teacher labels to the target node in the full search of design space X are depicted. As we can see, the accuracy tended to be higher when the cutoff gate was selected, indicating that using the knowledge from the teacher labels directly reduces the accuracy. This means that knowledge from the teacher labels is not necessary when distilling knowledge from auxiliary nodes that have been well trained using the label.

E. Comparison with conventional search methods

We compared the recognition accuracy of the optimal graph obtained by searching using ASHA and by searching the entire design space X. ASHA is a method that terminates learning and moves on to the next search if less than 50 percent of the



(a) ASHA (Design space A)

(b) Ours (Design space X)

Fig. 5. **KTGs explored on CIFAR-100**. Red nodes indicate target nodes and numbers in parentheses indicate the recognition accuracy of each node. "Label" indicates the teacher labels. Edges not shown are where the cutoff gate was selected.

TABLE V
ACCURACY OF KTGS

Method of exploration	Fixed (Through)	Explored
ASHA (Design space A)	73.66 ± 0.35	73.96 ± 0.16
Ours (Design space X)	74.06 ± 0.26	$\mathbf{74.52} \pm 0.19$

TABLE VI
EVALUATION USING DIFFERENT DATA SETS.

Dataset	Fixed (Through)	Explored (Searched by CIFAR-100)
CIFAR-100	74.06 ± 0.26	$\mathbf{74.52} \pm 0.19$
CIFAR-10	93.93 ± 0.21	$\mathbf{94.22} \pm 0.07$
Tiny ImageNet	$\mathbf{55.76} \pm 0.30$	54.54 ± 0.30
CINIC-10	85.26 ± 0.15	$\mathbf{85.50} \pm 0.30$

graphs acquired up to epoch $1,\ 2,\ 4,\ \cdots,\ 2^n$ are recognized. Table V shows the average accuracy and standard deviation of the best graph obtained by the search using ASHA and by the full search of design space X, trained five times each. "Fixed (Through)" means that all edges of the explored graph are set to through gate and the graph has lost its diversity. "Explored" refers to the best graph obtained by each search. Compared with the results of the search using ASHA, our proposed method that searches the entire design space obtained graphs with the higher accuracy.

Fig. 5(a) shows the explored KTGs obtained by searching design space A using ASHA. From the acquired graphs, we can confirm that graphs that suppress the strength of knowledge distillation in the early stage of learning have high recognition accuracy. Fig. 5(b) shows the explored KTGs obtained using the proposed method. From the acquired graphs, we can confirm that graphs that learn auxiliary nodes in the initial stage of learning and gradually distill the knowledge to the target nodes have a higher recognition accuracy.

F. Graph translatability

We investigated the generalization ability of the proposed method with graphs on different datasets. An untrained network model was used for each of the network models. The graphs explored using CIFAR-100 were used to investigate the change in accuracy after retraining on CIFAR-10, Tiny-ImageNet, and CINIC-10. Table VI lists the results, where "Fixed (Through gate)" denotes a graph whose nodes are WRN28-2 and whose edges have lost their diversity. "Explored" indicates the graph structure obtained by the proposed method. We can see that the graph explored on CIFAR-100 was valid on CIFAR-10 and CINIC-10, while on Tiny-ImageNet, the graphs without edge diversity were more accurate than those explored by CIFAR-100. CIFAR-10 and CINIC-10 are 10-class datasets consisting of vehicles and animals, respectively, while in contrast, Tiny-ImageNet is a 200-class dataset that includes many classes other than vehicles and animals. Therefore, our findings here show that it is possible to obtain a graph with good generalization performance if the graph is transferred between datasets with similar distributions.

V. CONCLUSION

In this paper, we proposed a method to refine the design space of KTGs step by step so that we can obtain a design space containing more accurate graphs than DML. Our findings show that the proposed method can obtain a design space X containing graphs with high recognition accuracy in a smaller number of searches compared to the full search of the design space A. Our analysis of the search process showed that there is no need to distill knowledge directly from the teacher labels if the knowledge comes from a model with high expressive power that was learned using the teacher labels. In addition, the graphs obtained were more accurate than those explored by ASHA.

Since our proposed method uses EDFs, the total number of searches is huge. Therefore, one of our future tasks is to find better graphs with fewer searches.

ACKNOWLEDGMENTS

This paper is based on results obtained from a project, JPNP18002, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

REFERENCES

[1] G. Hinton, O. Vinyals, and J. Dean, Distilling the Knowledge in a Neural Network, in *Neural Information Processing Systems Deep Learning Workshop*, 2014.

[2] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, Deep Mutual Learning, in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4320-4328.

[3] S. Minami, T. Hirakawa, T. Yamashita, and H. Fujiyoshi, Knowledge Transfer Graph for Deep Collaborative Learning, in *Asian Conference on Computer Vision*, 2020.

[4] X. Lan, X. Zhu, and S. Gong, Knowledge Distillation by On-the-Fly Native Ensemble, in *Neural Information Processing Systems*, 2018, volume 38.

[5] D. Chen, M. Jian-Ping, C. Wang,Y. Feng and C. Chen, Online Knowledge Distillation with Diverse Peers, in *Association for the Advancement of Artificial Intelligence*, 2020, pp. 3430-3437.

[6] R. Masumura, M. Ihori, A. Takashima, T. Tanaka, T. Ashihara, End-to-End Automatic Speech Recognition with Deep Mutual Learning, in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, 2020.

[7] B. Zoph and Quoc V. Le, Neural Architecture Search with Reinforcement Learning, in *International Conference on Learning Representations*, 2017.

[8] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, M. Hardt, B. Recht, and A. Talwalkar, A System for Massively Parallel Hyperparameter Tuning, in *Proceedings of Machine Learning and Systems*, 2020, pp. 230-246.

[9] J. S. Bergstra, R. Bardenetm, Y. Bengio and B. Kégl, Algorithms for hyper-parameter optimization, in *Advances in Neural Information Processing Systems*, 2011, pp. 2546–2554.

[10] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, Efficient Neural Architecture Search via Parameters Sharing, in *International Conference on Machine Learning*, 2018, pp. 4095-4104.

[11] H. Liu, K. Simonyan, and Y. Yang, DARTS: Differentiable Architecture Search, in *International Conference on Learning Representations*, 2019.

[12] C. White, W. Neiswanger, and Y. Savani, BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search, in *Association for the Advancement of Artificial Intelligence*, 2021, pp. 10293-10301.

[13] L. Ma, J. Cui, B. Yang, Deep Neural Architecture Search with Deep Graph Bayesian Optimization, in *International Conference on Web Intelligence*, 2019.

[14] J. Bergstra and Y. Bengio, Random search for hyper-parameter optimization, Journal of Machine Learning Research, 2012.

[15] Z. Karnin and T. Koren and O. Somekh, Almost optimal exploration in multi-armed bandits, in *International Conference on Machine Learning*, 2013.

[16] I. Radosavovic, J. Johnson, S. Xie, W. Lo, and P. Dollár, On Network Design Spaces for Visual Recognition, in *International Conference on Computer Vision*, 2019.

[17] L. Li and T. Ameet, Random search and reproducibility for neural architecture search, in *International Conference on Machine Learning*, 2019.

[18] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He and P. Dollár, Designing Network Design Spaces, in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp.10428-10436.

[19] A. Krizhevsky and G. Hinton, Learning multiple layers of features from tiny images, Citeseer, 2009.

[20] Tiny ImageNet Visual Recognition Challenge, https://tiny-imagenet.herokuapp.com/

[21] L. N. Darlow, E.J. Crowley, A. Antoniou, and A. J. Storkey, CINIC-10 is not ImageNet or CIFAR-10, *Report EDI-INF-ANC-1802, arXiv: 1810.03505*, 2018.

[22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. Fei-Fei, ImageNet Large Scale Visual Recognition Challenge, International Journal of Computer Vision, 2015.

[23] I. Loshchilov and F. Hutter, SGDR: Stochastic Gradient Descent with Warm Restarts, in *International Conference on Learning Representations*, 2017.

[24] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 770-778.

[25] S. Zagoruyko and N. Komodakis, Wide Residual Networks, in *British Machine Vision Conference*, 2016, pp. 87.1-87.12.