Generalizing Teacher Networks for Effective Knowledge Distillation Across Student Architectures

Kuluhan Binici
kuluhan@comp.nus.edu.sg
Weiming Wu
weiming.wu@u.nus.edu
Tulika Mitra
tulika@comp.nus.edu.sg

School of Computing
National University of Singapore
Singapore

Abstract

Knowledge distillation (KD) is a model compression method that entails training a compact student model to emulate the performance of a more complex teacher model. However, the architectural capacity gap between the two models limits the effectiveness of knowledge transfer. Addressing this issue, previous works focused on customizing teacher-student pairs to improve compatibility, a computationally expensive process that needs to be repeated every time either model changes. Hence, these methods are impractical when a teacher model has to be compressed into different student models for deployment on multiple hardware devices with distinct resource constraints. In this work, we propose *Generic Teacher Network (GTN)*, a one-off KD-aware training to create a generic teacher capable of effectively transferring knowledge to any student model sampled from a given finite pool of architectures. To this end, we represent the student pool as a weight-sharing supernet and condition our generic teacher to align with the capacities of various student architectures sampled from this supernet. Experimental evaluation shows that our method both improves overall KD effectiveness and amortizes the minimal additional training cost of the generic teacher across students in the pool.

1 Introduction

In practical scenarios, such as deploying AI solutions in mobile devices, IoT devices, and embedded systems, there is a vast spectrum of computational capabilities and memory limitations. These differences necessitate the use of neural network models of varying sizes and complexities, tailored to the specific resources available on each device. Selecting the most appropriate model is challenging as high-performing models are often resource-demanding, while the resource-efficient ones often cannot achieve high performance. To break this trade-off, a popular approach is reducing the resource footprint of large pre-trained models, without sacrificing significant performance through model compression. *Knowledge distillation* (KD) [10] is a prominent model compression technique that involves transferring the predictive behavior acquired by a large pre-trained model, known as the "teacher", to a compact

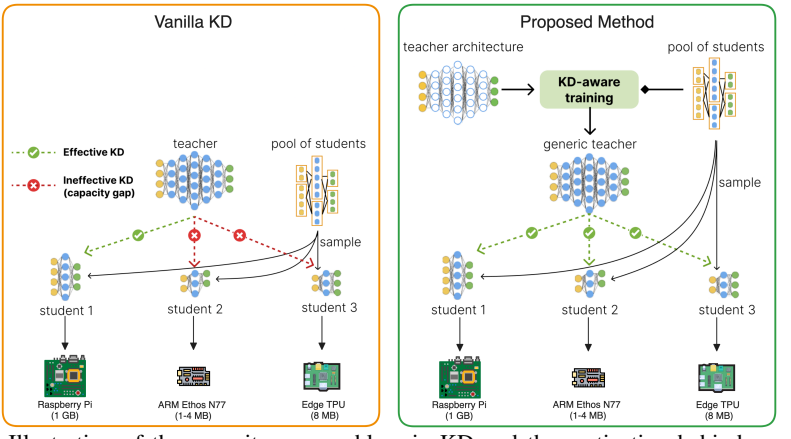

Figure 1: Illustration of the capacity gap problem in KD and the motivation behind our proposed generic teacher approach.

"student" architecture. However, not every neural network effectively benefits from KD due to the potential capacity mismatch with the teacher model.

Addressing this issue, some works explored methods for bridging such capacity gap at distillation time [24]. These typically involve controlling the influence of the teacher model throughout the training process of the student to prevent adverse impacts on accuracy. However, despite their user-friendly nature, as they do not require modifications to the teacher model, their effectiveness is constrained, as evidenced by our experiments. An alternative approach is specializing a given teacher architecture to be conducive for effective knowledge transfer to a particular reference student [16]. This is achieved by conditioning the training process of the given teacher, which involves optimizing a snapshot of the reference student model jointly with the teacher on the target dataset. Throughout this process, the teacher is additionally constrained to match the outputs of the student snapshot. This causes the teacher to converge to a function that the student can easily approximate within its capacity later during the KD stage. However, the specialization procedure has to be repeated between every teacher-student pair as teacher models tailored to a particular student might be ineffective or even detrimental for other students, as highlighted in our experiments. This need for iterative repetition upscales the time cost, rendering the method impractical when dealing with scenarios with multiple students requiring KD. This creates a tradeoff between performance and scalability.

In this work, we present a one-off KD-aware teacher training method that trains a generic teacher model with consideration to the capacities of all student models contained in a given finite pool of architectures, as illustrated in Figure 1. This eliminates the aforementioned performance vs. scalability tradeoff by limiting the time cost to the one-off training time of the generic teacher. Our method involves partially training reference student models sampled from the pool, to regularize the output predictions of the teacher. To avoid significant time costs, we allow parameter sharing among these reference student models. For this, we configure the pool as an over-parameterized supernet architecture [12], containing a diverse range of neural network blocks that can be connected in various ways to represent different neural network architectures. At each training iteration of the teacher model, the supernet model is reconfigured to represent a different reference student model by selecting a single NN block at each supernet layer. Therefore, parameters are shared across all students containing the same NN blocks within their architectures. By reconfiguring the supernet at each

iteration, the teacher model is regularized based on the capacity of a different student model.

Experimental evaluation of our method reveals consistent improvements in the KD accuracy of students sampled from the pool of architectures, resolving the limited effectiveness of specialized teachers in generalizing to different students. While providing such flexibility, our approach maintains a constant time overhead which is equivalent to training 2-3 teacher models specialized for different students. Furthermore, as an extended evaluation, we explore the use of students selected through Neural Architecture Search (NAS) rather than manual selection from the given pool. The results of these experiments indicate the viability of our approach in NAS scenarios to boost the performance of various student models customized for different deployment platforms.

2 Related Work

2.1 Knowledge Distillation (KD)

Knowledge Distillation (KD) [10] is a technique that involves training a compact neural network, referred to as the "student" model, to approximate the decision-making capabilities of a more complex one known as the "teacher." Typically, the student model is trained to match the logit scores or softmax probabilities of the teacher model [19]. However, alternative approaches, such as employing activation maps or attention scores, have also been explored [22] for this purpose. Recently Zhao et al. [23] identified that the skewness of the predictive distribution of complex teachers towards target classes limits the transfer of useful knowledge from non-target classes. Introducing Decoupled Knowledge Distillation (DKD), they separate knowledge transfer from target and non-target classes. DKD dynamically adjusts the weighting of both learning objectives to ensure that the impact of non-target class-related information is not overlooked.

2.2 Teacher-Student Capacity Gap in KD

Research by Cho et al. [3], Mirzadeh et al. [15], and Menon et al. [14] challenges the notion that more accurate teachers always enhance student learning, highlighting that mismatches in model capacities can hinder KD. Liu et al. [13] further suggest that different student models perform better with different teachers, indicating that compatibility between teacher and student models affects KD effectiveness. Zhu et al. [24] introduced Student-customized Knowledge Distillation (SCKD) to mitigate issues arising from the capacity gap by adapting knowledge transfer based on gradient orientations, although it does not alter the teacher's teaching abilities.

To address the capacity gap more effectively, SFTN [16] customizes the teacher model for specific students, enhancing KD outcomes for those students but potentially harming others. This customization is not scalable when multiple students with different resource needs must be accommodated, leading to significant time costs.

2.3 Neural Architecture Search & Supernet Architectures

Neural Architecture Search (NAS) automates the design of neural architectures, often outperforming manual designs for various tasks [5, 8]. Initial approaches utilized genetic algorithms [18] or reinforcement learning [25], which are time-consuming due to the need to train and evaluate many models. To enhance efficiency, DARTS [12] introduced a differentiable search strategy within a weight-sharing "supernet", allowing gradient-based optimization to find optimal models. However, this method significantly increases memory

consumption. ProxylessNAS [2] addresses this by binarizing paths in the supernet, reducing memory demands by loading only one path at a time during the search. Our method's supernet configuration is inspired by the design used in ProxylessNAS.

3 Method

To obtain our generic teacher we condition its training process with consideration to the capacities of various reference students drawn from the given set. For such conditioning, we use the SFTN algorithm. As this would involve training snapshots of each of these reference students from scratch, we allow weight-sharing among them to avoid excessive time costs. This is achieved by using a supernet architecture that contains all possible candidate operations required to build any student model from the pool. We name our method as *Generic Teacher Networks* (GTN). Technical details are discussed in the following sections.

3.1 Conditioning the Teacher Based on the Capacity of a Reference Student Architecture

This process aims to achieve two goals: (i) training a teacher model until convergence to a function that yields high accuracy, (ii) constraining the set of functions that the teacher can converge to, based on the reference students' capacity. The first target can be simply achieved by minimizing the cross-entropy between the predictions of the teacher and the ground truth labels. As for meeting the second objective, we make use of SFTN approach. First, the reference student model is partitioned into a number of blocks, and various combinations of these blocks are grafted on top of teacher blocks as in Figure 2 (a). Later during training, the outputs of the teacher are forced to match those of each of these grafted student branches. Therefore the final optimization objective is minimizing the loss function in Eq. 1.

$$L_{CT} = \frac{1}{n}\sum_{i=1}^{n}\left(\underbrace{y_{gt}\log(\hat{y}_{s_i})}_{L_{CE_S}} + \alpha\underbrace{T^2 D_{KL}\left(\frac{z_t}{T}\middle\|\frac{z_{s_i}}{T}\right)}_{L_{KL}}\right) + \underbrace{y_{gt}\log(\hat{y}_t)}_{L_{CE_T}} \tag{1}$$

In Eq. 1 n denotes the number of student branches, while $\hat{y}_t$ and $\hat{y_{s_i}}$ represent the predictive outputs of the teacher and student branches indexed with i respectively. α is a coefficient that adjusts the weighting of the L_{KL} with respect to other terms. The loss term is composed of three sub-terms, namely L_{CE_S}, L_{KL} and L_{CE_T}. The first and the last terms are the cross-entropy losses from the student and teacher branches calculated using the ground truth labels y_{gt}. Moreover, the second term, L_{KL}, is the KL-divergence between the re-scaled logits of the teacher and student (z_t and z_{s_i}) by temperature T. Minimizing this term constrains the outputs of the teacher to match those of the student branches.

3.2 GTN Training Using a Supernet as Reference Architecture

The most straightforward way for conditioning the teacher using various reference students, would be extending the amount of grafted student branches with blocks from different architectures. However, this would increase the memory footprint significantly and render the training process infeasible on many hardware systems. Moreover, optimizing such a large

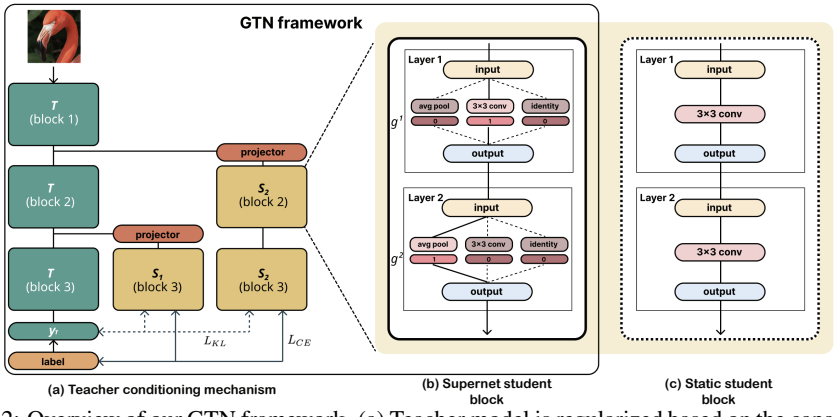

Figure 2: Overview of our GTN framework. (a) Teacher model is regularized based on the capacity of a reference student. (b) Supernet blocks allow the architecture of the student branches to be reconfigured, exposing the teacher to various reference students for regularization. (c) Static student blocks used to train specialised teachers.

number of student branches from scratch along with the teacher would consume a significant amount of time. Instead, we propose configuring a single reference architecture that embodies all student models in the given pool, and allows weight sharing among them. For this purpose we use a supernet architecture consisting of multiple alternate paths at every layer, with each path containing a different candidate operation that maps the inputs to the outputs, as shown in Figure 2 (b). Operations that are commonly shared among different student models at corresponding layers, use the same trainable parameters. This prevents excessive time costs due to re-training the same set of parameters for each student. To condition the teacher with this supernet as the reference architecture, we again partition it into blocks and create the student branches. Later during subsequent training, we employ path binarization to reduce the memory requirement, as done in Cai et al. [2]. This technique trains a supernet stochastically, by sampling and updating a single operation path from every layer at each iteration. Therefore, at any time, the memory occupation is limited to the size of a single sub-network contained within the supernet. In the context of our GTN framework, these sampled sub-networks correspond to individual student models from the given pool. As sampling operations randomly can cause the teacher model to be regularized by aribtrarily bad student models and damage the conditioning outcome, we instead sample them from a trainable probability distribution. For this, we use a multinomial probability distribution $p_\phi = softmax(\phi)$ parameterized by a single trainable random variable ϕ. To activate the sampled operation paths and deactivate the rest, we use binary gates. For k-many candidate operations, we use the same amount of binary gates, with each gate g^j controlling whether the j^{th} operation will be activated or not. The index of the active operation is denoted as j^{act}.

$$g^j = \begin{cases} 1, & \text{if } j = j^{act} \sim p_\phi \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The random variable ϕ has k-many possible outcomes (one for each operation) and is learned during the GTN training process by minimizing L_ϕ loss given in Eq. 3.

$$L_\phi = L_{CE_S} - \alpha L_{KL} \tag{3}$$

Different from L_{CT}, L_{KL} appears with a negative α coefficient in L_ϕ This is because, ideally, the gate parameters should be updated in a way that would motivate an extensive exploration of candidate operations at each training step. While sampled student parameters θ are updated to match the predictions of the teacher, those that are not sampled remain divergent from the teacher. This means that the unexplored operations would yield high KL divergence from the outputs of the teacher. Therefore, by minimizing the $-\alpha L_{KL}$ term, the ϕ variable is updated to encourage the sampling of operations that yield high KL-divergence loss — typically, those that are yet to be explored.

Once the gate values are set for a certain iteration, the function $f_\theta^{l,act}$ that maps the inputs of the l^{th} supernet layer to the outputs is computed based on Eq. 4. The θ denotes the trainable parameters of the candidate operations.

$$f_\theta^{l,act}(x) = \sum_j g^{l,j} f_\theta^{l,j}(x) \quad \text{where } g^{l,j} \in \{0,1\} \tag{4}$$

Using this definition, the forward process of each student branch s_i, containing $\mathbb{L}_{s_i}$ number of layers, can be represented by the composite function given in Eq. 5.

$$\hat{y}_{s_i} = f_{\theta_{s_i}}^{\mathbb{L}_{s_i},act} \circ f_{\theta_{s_i}}^{\mathbb{L}_{s_i}-1,act} \circ \ldots \circ f_{\theta_{s_i}}^{2,act} \circ f_{\theta_{s_i}}^{1,act}(x) \tag{5}$$

In training our GTN with the supernet architecture, we follow Algorithm 1. The parameters of the sampled operations in the supernet (θ_S) and the teacher (θ_T) are updated together (line 12). The ϕ values, controlling the sampling probability, are updated alternately with θ_S and θ_T (lines 11-15). Initially, ϕ is fixed, and operations are sampled for each layer (lines 4-8), which helps regularize the teacher's optimization by minimizing L_{CT}. In the following iteration, while θ_S and θ_T are fixed, ϕ is updated to minimize L_ϕ (line 14). This alternating process is repeated until the training concludes after the designated number of epochs.

3.3 Knowledge Distillation

After training the GTN, the auxiliary branches containing supernet blocks are discarded and the remaining teacher model can be used to enhance the accuracy of any student model from

Algorithm 1 GTN Training

1: **INPUT:** Teacher T parameterized by θ_T, student branches $S = \{s_1, s_2, \ldots, s_n\}$ parameterized by $\theta_S = \{\theta_{s_1}, \theta_{s_2}, \ldots, \theta_{s_n}\}$, and gate params $\Phi = \{\phi^{s_1,1}, \ldots, \phi^{s_i,l}, \ldots, \phi^{s_n,\mathbb{L}_{s_n}}\}$
2: **for** iteration i **in** total # of iterations
3: **for all** s_i **in** S
4: **for** layer l in set of layers in s_i
5: $j_{act}^l \sim \phi^{s_i,l}$
6: set $g^{l,j}$ using j_{act}^l based on Eq. 2
7: determine $f_{\theta_{s_i}}^{l,\text{act}}$ based on Eq. 4
8: **end for**
9: **end for**
10: $\hat{y}_{s_i} \leftarrow f_{\theta_{s_i}}^{\mathbb{L},act} \circ \ldots \circ f_{\theta_{s_i}}^{2,act} \circ f_{\theta_{s_i}}^{1,act}(x)$
11: **if** $i\%2 == 0$ **then**
12: $optimizer.step(backward(L_{CT}), \theta_T, \theta_S)$
13: **else**
14: $optimizer.step(backward(L_\phi), \Phi)$
15: **end if**
16: **end for**

the pool. At this stage, the target student model is trained with the combined guidance of the ground truth labels and the teacher's logit scores. While any KD optimization objective can be used, we experimented with the vanilla loss function given in Eq. 6 and the one introduced in DKD.

$$L_{KD} = L_{CE_S} + \alpha L_{KL} \tag{6}$$

4 Experimental Evaluation

To assess the effectiveness of our method, we evaluate the accuracies of student models randomly selected from a pool of architectures. These students are trained using our method and compared against five different approaches: SFTN, SCKD, DKD, Vanilla KD [10], and supervised training (denoted as no-KD). Vanilla KD and DKD, which do not address the capacity gap, establish the lower performance bounds for our comparison. In contrast, SFTN and SCKD, which consider this gap, serve as the main baselines to assess our method's effectiveness in mitigating it. We also include students derived from NAS using Proxyless-NAS [2], comparing their performance on CIFAR-100 [11] and ImageNet-200 [4], to further validate our approach. Lastly, we display the memory sizes of these student models in conjunction with the on-chip memory availability of three edge devices to showcase a real-world scenario of tailoring student models for specific hardware platforms.

Implementation Details: In our KD experiments, we employ three teacher architectures: ResNet-32, WRN40-2, and EfficientNet-b0 [8, 20, 21]. During teacher training, our GTN framework modifies the teacher architecture by grafting blocks of the reference supernet architecture, onto the teacher to form different student branches. Once the teacher is trained, these branches are discarded and the teacher model is used to train student models via KD. The supernet is constructed with ResNet layers varying in depth and filter size, equipped with identity and zero operations to allow flexible architecture sampling. For knowledge transfer, we use Vanilla KD and DKD methods, training student models over 240 epochs with learning rates of 0.05 and 0.1 for CIFAR-100 and ImageNet-200 datasets respectively. Cosine annealing is used for adjusting the learning rates. Further implementation details and training configurations are provided in the Appendix.

KD with random students: We first compare our KD approach with baseline methods based on the performance of student models randomly drawn from the pool of architectures represented by the supernet. For each dataset, we randomly select seven student architectures. Each method except SFTN uses a single teacher model leading to seven teacher-student pairings for each dataset and teacher architecture combination (e.g. ResNet-32 & CIFAR-100). As for SFTN, we train four teachers, each specialized for a different student, leading to twenty-eight teacher-student pairings. The results are summarized in Table 1, with Δ indicating the relative improvements compared to the vanilla KD method. μ_Δ and

Dataset	CIFAR-100								ImageNet-200							
Teacher	ResNet-32				WRN40-2				ResNet-32				EfficientNet-b0			
Training Method	DKD	SCKD	SFTN	GTN (ours)	DKD	SCKD	SFTN	GTN (ours)	DKD	SCKD	SFTN	GTN (ours)	DKD	SCKD	SFTN	GTN (ours)
μ_Δ (↑)	0.68	0.59	2.14	**2.66**	0.46	0.47	1.77	**1.99**	2.10	1.93	3.29	**3.91**	4.11	3.84	4.45	**4.82**
σ_Δ (↓)	**0.19**	0.34	0.89	0.38	0.31	0.43	**0.28**	0.38	0.31	**0.22**	0.41	0.33	0.49	0.45	0.42	**0.42**

Table 1: Statistical comparison of different KD methods in terms of improvement w.r.t. vanilla KD. Δ is the relative accuracy improvement (%) w.r.t. vanilla teachers. μ_Δ, σ_Δ and range$_\Delta$ represent mean, standard deviation, and range of Δ for KD across 7 different students sampled from the architecture pool. DKD is used as the base distillation method.

Method	no KD	Vanilla KD	SCKD	SFTN				GTN (ours)
CIFAR-100								
Reference Student	N/A	N/A	N/A	s1	s2	s3	s4	supernet
Teacher acc.	N/A	78.04	78.04	81.54	81.46	81.49	80.97	80.71
s1 acc. (Δ)	73.22	75.40	75.34 (−0.06)	75.82 (+0.42)	76.09 (+0.69)	75.34 (−0.06)	76.03 (+0.63)	**76.70** (+1.30)
s2 acc. (Δ)	77.92	77.89	77.47 (−0.42)	78.23 (+0.34)	78.52 (+0.63)	77.90 (+0.01)	77.76 (−0.13)	**78.85** (+0.96)
s3 acc. (Δ)	76.87	77.21	77.08 (−0.13)	77.66 (+0.45)	78.05 (+0.84)	78.05 (+0.84)	77.94 (+0.73)	**78.22** (+1.01)
s4 acc. (Δ)	75.60	76.42	75.77 (−0.65)	77.20 (+0.78)	77.71 (+1.29)	77.23 (+0.81)	77.66 (+1.24)	**77.79** (+1.37)
ImageNet-200								
Reference Student	N/A	N/A	N/A	s5	s6	s7	s8	supernet
Teacher acc.	N/A	65.28	65.28	69.71	69.08	68.96	69.00	70.10
s5 acc. (Δ)	62.63	64.86	65.36 (+0.50)	65.54 (+0.68)	64.99 (+0.13)	65.08 (+0.22)	65.30 (+0.44)	**65.63** (+0.77)
s6 acc. (Δ)	64.34	64.79	65.68 (+0.45)	66.00 (+1.21)	65.71 (+0.92)	65.88 (+1.09)	65.62 (+0.83)	**66.08** (+1.29)
s7 acc. (Δ)	62.62	64.37	63.97 (−0.40)	64.69 (+0.32)	64.73 (+0.36)	64.71 (+0.34)	65.13 (+0.76)	**65.74** (+1.37)
s8 acc. (Δ)	62.15	63.53	63.95 (+0.42)	63.48 (−0.05)	64.13 (+0.60)	63.88 (+0.35)	64.13 (+0.60)	**64.29** (+0.76)

Table 2: Accuracy values (%) of randomly selected student models distilled using different methods. Δ denotes improvement (%) over vanilla KD. Vanilla KD is used as the base distillation method.

σ_Δ represent the mean and standard deviation of Δ recorded across seven different reference students. The range_Δ is the gap between the maximum and minimum Δ. The results showcase GTN's consistent advantage in improving student accuracy, underscoring its versatility in mitigating the teacher-student capacity gap. Table 2 details individual accuracies for four students among the aforementioned seven randomly sampled students per dataset, trained with the supervision of ResNet-32 teachers. These are identified as $s1, s2, s3, s4$ in CIFAR-100 and $s5, s6, s7, s8$ in ImageNet-200. While GTN generally enhances KD accuracy across all experiments SFTN sometimes result in accuracy decreases when applied to non-reference students, as noted in Table 2. For instance, a teacher optimized for $s4$ might yield reduced accuracy, i.e. $\Delta = -0.13$ when used to train $s2$, indicating that SFTN training does not universally benefit all student types. SCKD also causes accuracy drops for some students. Additionaly, our findings reveal that SFTN does not always yield the highest accuracy improvements for the students it is specialized for. For example, a teacher trained specifically for $s2$ might achieve a larger improvement for $s4$, achieving a Δ of $+1.29$ vs $+0.63$, suggesting that while SFTN acts as a regularizer, it may inadvertently benefit other students more than the intended ones. SFTN and GTN teachers generally achieve higher accuracy than vanilla teachers on target test sets due to regularization process they involve preventing overfitting. However, as seen in Table 2, teacher model accuracy does not directly correlate with effectiveness in KD.

While consistently improving accuracy across student models, our method incurs only a one-off time cost that is slightly under that of training three SFTN teachers. As seen in Fig. 3 the time cost of SFTN scales linearly with the number of students involved. When more than two deployment scenarios exist, our GTN approach takes over the time advantage.

KD with students obtained by NAS Besides selecting the student models for different deployment scenarios manually, NAS can also be employed to sample students that exhibit close to optimal performance within the given resource budgets. To test the effectiveness of

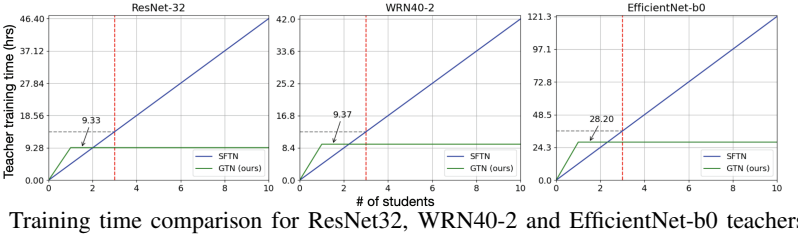

Figure 3: Training time comparison for ResNet32, WRN40-2 and EfficientNet-b0 teachers respectively. Dashed vertical lines colored in red mark the # of students after which our GTN method attains the time cost advantage.

Dataset	CIFAR-100								ImageNet-200							
Teacher	ResNet-32				WRN40-2				ResNet-32				EfficientNet-b0			
Training Method	DKD	SCKD	SFTN	GTN (ours)	DKD	SCKD	SFTN	GTN (ours)	DKD	SCKD	SFTN	GTN (ours)	DKD	SCKD	SFTN	GTN (ours)
s^l_{nas} acc.	78.18	77.51	79.48	**79.95**	77.98	78.02	79.33	**79.55**	67.09	66.17	68.17	**68.87**	69.63	69.26	69.78	**70.19**
s^m_{nas} acc.	77.64	77.89	79.35	**79.63**	77.96	78.31	79.27	**79.33**	65.67	65.97	67.24	**67.41**	67.83	67.77	68.44	**68.48**
s^s_{nas} acc.	75.76	76.08	76.16	**77.23**	75.77	75.79	76.45	**76.80**	65.19	65.35	66.63	**67.52**	67.42	67.27	67.92	**68.55**

Table 3: Accuracy values (%) of student models obtained by NAS, distilled using different methods. Δ denotes improvement (%) over vanilla KD. DKD is used as the main distillation method.

	s1		**s2**		**s3**		**s4**		**s^s_{nas}**		**s^m_{nas}**		**s^l_{nas}**	
Bit-width	32b	8b	32b	8b	32b	8b	32b	8b	32b	8b	32b	8b	32b	8b
Memory Size (MB)	22.2	5.5	29.9	7.5	50.6	12.6	25.4	6.4	6.7	1.7	28.6	7.2	77.1	19.3
Arm Ethos N77 (1-4 MB)	X		X		X		X		✓		X		X	
Edge TPU (8 MB)	✓		✓		X		✓		✓		✓		X	
Raspberry Pi (1 GB)	✓		✓		✓		✓		✓		✓		✓	

Table 4: Compatibility of Edge Devices with Various Student Models

our method in improving KD accuracy for NAS-discovered student architectures, we search our pool of student architectures to obtain three distinct student architectures of varying sizes. Since we represent the architecture pool using a supernet, we employ a differentiable search strategy, that is ProxylessNAS [2]. To obtain each student, we limit the searched model size by a different number of layers. The largest architecture, denoted as s^l_{nas}, is derived from the full-scale supernet consisting of 6 searchable layers without any constraints. For the medium (s^m_{nas}) and small (s^s_{nas}) architectures, we reduce the number of searchable layers to 5 and 3, respectively. Subsequently, to prepare the SFTN baseline for comparison, we train specialized teachers using these three architectures as reference students. We then evaluate these teachers based on their KD performance for all three NAS-discovered students. We again use DKD as the base distillation method in this evaluation. From Table 3 we can observe that GTN method again caused the largest improvement in student accuracies in almost all experiments. This means that when a new student is configured via NAS due to changes in the deployment platform, our GTN method is more likely to provide higher accuracy benefits through KD.

In Table 4, we display the memory sizes of seven student models we considered in CIFAR-100 experiments, detailing both the full-scale (32-bit) and the 8-bit weight quantized versions. The varied memory sizes of these student models, demonstrate the diversity of our student model pool, which contains a range of architectures with different capacities. Moreover, we provide the on-chip memory capacities of three edge devices—Arm Ethos N77 [1] , Edge TPU [7], and Raspberry Pi [17]—, focusing on the feasibility of deploying each student model on these platforms. The green checkmarks and red Xs, show whether each 8-bit quantized model fits within the memory constraints of these devices.

5 Conclusion

In conclusion, this study tackles the teacher-student capacity gap problem, which undermines the effectiveness of Knowledge Distillation (KD) in optimizing neural networks. Previous methods have concentrated on customizing teacher-student pairs, a process that needs to be repeated for each student architecture requiring KD. This characteristic renders these approaches impractical, particularly in scenarios involving multiple deployment platforms, each demanding KD for distinct deployment-ready students. Our novel approach introduces a single generic teacher model capable of transferring knowledge effectively across a variety of student architectures. The proposed technique's advantage is that it both improves KD performance and maintains constant time cost, which is equivalent to that of a few specialized teachers. Moreover, the adaptability of our approach to NAS scenarios further highlights its practicality.

References

[1] Arm. Arm ethos: A suite of products for intelligent, immersive experiences. Retrieved from https://newsroom.arm.com/news/new-arm-ip-brings intelligent-immersive-experiences-to-mainstream-markets.

[2] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2018.

[3] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4794–4802, 2019.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[5] Thomas Elsken, Jan Hendrik Metzen Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(01):1997–2017, 2020.

[6] Roy Henha Eyono, Fabio Maria Carlucci, Pedro M. Esperança, Binxin Ru, and Phillip Torr. Autokd: Automatic knowledge distillation into a student architecture family. *arXiv:2111.03555*, 2021.

[7] Google. Edge tpu: A custom asic for running inference at the edge. Retrieved from https://cloud.google.com/edge-tpu.

[8] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[11] A Krizhevsky and G Hinton. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009.

[12] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2018.

[13] Yu Liu, Xuhui Jia, Mingxing Tan, Raviteja Vemulapalli, and Yukun Zhu. Search to distill: Pearls are everywhere but not the eyes. 2020.

[14] Aditya K Menon, Ankit Singh Rawat, Sashank Reddi, Seungyeon Kim, and Sanjiv Kumar. A statistical perspective on distillation. In *International Conference on Machine Learning*, pages 7632–7642. Proceedings of Machine Learning Research, 2021.

[15] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5191–5198, 2020.

[16] Dae Young Park, Moon-Hyun Cha, Daesin Kim, Bohyung Han, et al. Learning student-friendly teacher networks for knowledge distillation. *Advances in Neural Information Processing Systems*, 34:13292–13303, 2021.

[17] Raspberry Pi Ltd. Raspberry Pi Documentation. Retrieved from https://www.raspberrypi.com/documentation/.

[18] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4780–4789, 2019.

[19] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[20] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. Proceedings of Machine Learning Research, 2019.

[21] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Procedings of the British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

[22] 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*, 2017.

[23] Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 11953–11962, 2022.

[24] Yichen Zhu and Yi Wang. Student customized knowledge distillation: Bridging the gap between student and teacher. *In Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*, pages 5057–5066, 2021.

[25] Barret Zoph and Quoc Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2016.