Search to Distill: Pearls are Everywhere but not the Eyes

Yu Liu*[1,3] Xuhui Jia[1] Mingxing Tan[2] Raviteja Vemulapalli[1] Yukun Zhu[1]
Bradley Green[1] Xiaogang Wang[3]
[1]Google AI [2]Google Brain
[3]Multimedia Laboratory, The Chinese University of Hong Kong
liuyuisanai@gmail.com
{xhjia, tanmingxing, ravitejavemu}@google.com

Abstract

Standard Knowledge Distillation (KD) approaches distill the knowledge of a cumbersome teacher model into the parameters of a student model with a pre-defined architecture. However, the knowledge of a neural network, which is represented by the network's output distribution conditioned on its input, depends not only on its parameters but also on its architecture. Hence, a more generalized approach for KD is to distill the teacher's knowledge into both the parameters and architecture of the student. To achieve this, we present a new Architecture-aware Knowledge Distillation (AKD) approach that finds student models (pearls for the teacher) that are best for distilling the given teacher model. In particular, we leverage Neural Architecture Search (NAS), equipped with our KD-guided reward, to search for the best student architectures for a given teacher. Experimental results show our proposed AKD consistently outperforms the conventional NAS plus KD approach, and achieves state-of-the-art results on the ImageNet classification task under various latency settings. Furthermore, the best AKD student architecture for the ImageNet classification task also transfers well to other tasks such as million level face recognition and ensemble learning.

1. Introduction

Over the past few years, Deep Neural Networks (DNNs) have achieved unprecedented success on various tasks, such as image understanding [16, 24], image generation [23], machine translation [37] and speech recognition [7]. In the deep learning era, one of the main driving forces for performance improvement is designing better neural architectures [5, 32].

Aiming to reduce the need for domain knowledge and minimize human intervention, Neural Architecture Search (NAS) automates the process of neural architecture design via reinforcement learning [43], differentiable search [19, 22], evolutionary search [28], and other algorithms [1]. Recently, some works have also started to explore simultaneous optimization of network parameters and architecture for a given task [31]. It has been shown that NAS can discover novel architectures that surpass human-designed architectures on large-scale image classification problems [28, 44].

In parallel to neural architecture advances, Knowledge Distillation (KD) [8], which trains a (usually small) student neural network using the supervision of a (relatively large) teacher network, has demonstrated great effectiveness over directly training the student network using hard labels. Previous works on KD [6, 39] mostly focus on transferring the teacher's knowledge to a student with a pre-

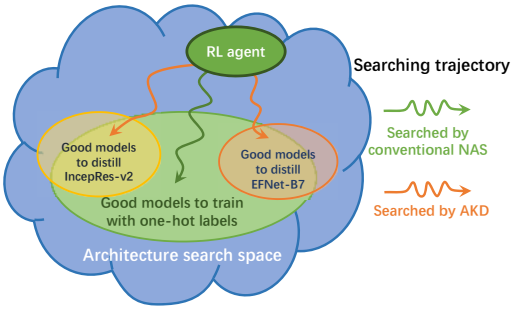


Figure 1. Searching neural architectures by the proposed AKD and conventional NAS [34] lead to different optimal architectures.

Teachers	Student1	Student2	Comparison
EfficientNet-B7 [35]	65.8%	66.6%	student1 < student2
Inception-ResNet-v2 [32]	67.4%	66.1%	student1 > student2

Table 1. ImageNet accuracy for students with different teachers.

*This work is performed in Yu's internship at Google Inc.

defined architecture. However, we argue that the optimal student architectures for different teacher models trained on the same task and dataset might be different. To validate this argument, we study the performance of two randomly constructed student models with different teachers. As shown in Table 1, while student1 outperforms student2 with EfficientNet-B7 [35] as teacher, student2 works better with the Inception-ResNet-v2 [32] as teacher (see Table 3 for more comprehensive studies). These results indicate that each teacher model could potentially have its own best student architecture.

Motivated by these observations, this paper proposes a new generalized approach for KD, referred to as Architecture-aware Knowledge Distillation (AKD), which finds best student architectures for distilling the given teacher model. The proposed approach (Fig 3) searches for student architectures using a Reinforcement Learning (RL) based NAS process [43] with a KD-based reward function. Our results show that the best student architectures obtained by AKD achieve state-of-the-art results on the ImageNet classification task under several latency settings (Table 7) and consistently outperform the architectures obtained by conventional NAS [34] (Table 5). Surprisingly, the optimal architecture obtained by AKD for the ImageNet classification task generalizes well to other tasks such as million-level face recognition (Table 9) and ensemble learning (Table 10). Our analysis of the neural architectures show that the proposed AKD and conventional NAS [34] processes select architectures from different regions of the search space (Fig 5). In addition, our analysis also verifies our assumption that the best student architectures for different teacher models differ (Fig 6). To the best of our knowledge, this is the first work to investigate and utilize structural knowledge of neural architectures in KD on large-scale vision tasks.

2. Knowledge distillation

In this section, we present the standard KD framework and provide motivation for the proposed architecture-aware knowledge distillation approach.

2.1. Knowledge in a neural network

For a task with input space $\mathcal{I}$ and output space $\mathcal{O}$, an ideal model is a connotative mapping function $f : x \mapsto y$, $x \in \mathcal{I}, y \in \mathcal{O}$ from the task's input space $\mathcal{I}$ to output space $\mathcal{O}$ that produces the correct output label for every input, and the knowledge of this model is represented by the model's conditional probability function $p(y|x)$. The knowledge of a neural network $\hat{f} : x \mapsto \hat{y}, x \in \mathcal{I}, \hat{y} \in \mathcal{O}$, trained for this task is represented by the network's conditional probability function $\hat{p}(\hat{y}|x)$. The difference between $\hat{p}(\hat{y}|x)$ and $p(y|x)$ is the dark part of the neural network's knowledge.

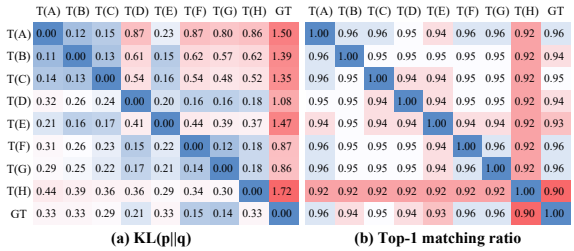
Researchers have tried to understand this dark part in different ways. A recent work [27] showed that the data geometry, such as margin between classes, can strongly benefit KD in the case of a single-layer linear network. While the ideal one-hot output y constrains the angular distances between different classes to be the same $90°$, the soft probability output $\hat{y}$ has a learning-friendly dynamic metric: more similar classes/samples should have smaller angular distances. This latent structure of the soft label distribution contributes to the dark knowledge, and supervising the student model by the soft label distribution would work better in comparison to one-hot labels. Hinton et al. [8] proposed the knowledge distillation framework, which tries to distil the soft distribution of a teacher into a student neural network with a pre-defined architecture.

2.2. Naïve distillation

Interest in KD increased following Hinton et al. [8], who demonstrated a method called *dark* knowledge distillation, in which a student model trained with the objective of matching full softmax distribution of the teacher model. Commonly, the teacher is a high-capacity model with formidable performance, while the student network is compact. By transferring knowledge, one hopes to benefit from both the student's compactness and the teacher's capacity. While this strategy has been widely adopted, especially for edge devices that require low memory or fast execution, there are few systematic and theoretical studies on how and why knowledge distillation improves neural network training. [8] suggest that the success of KD depends on the distribution of logits of the wrong responses, that carry information on the similarity between output categories. [3] argue that soft-target distribution acts as an importance sampling weight based on the teacher's confidence in its maximum value. [42] analyzed knowledge distillation from the posterior entropy viewpoint claiming that soft-targets bring robustness by regularizing a much more informed choice of alternatives than blind entropy regularization. Nevertheless, to our knowledge, no previous works attempt to explain from the perspective of network inherent architecture, as previous efforts investigate mainly from learning theory and distillation methods.

2.3. Teacher-Student relationship

Although KD can be applied to any pair of teacher and student networks, a natural question is: *are all student networks equally capable of receiving knowledge from different teachers?* To answer this question, we grab 8 representative off-the-shelf teacher models, which vary in term of input size, architecture and capacity, as shown in Table 2. Subsequently, we randomly sample 5 different student architectures (which have similar performance when trained with one-hot label vectors) from the search space defined



(a) KL(p||q) **(b) Top-1 matching ratio**

Figure 2. Confusion matrix for models' outputs, p or q denotes softmax probability output, GT means one-hot label.

by MNAS [34] to investigate the role of student architecture when transferring dark knowledge.

Table 3 summarizes the knowledge distillation performance for each pair of teacher and student. Distilling the same teacher model to different students leads to different performance results, and no student architecture produces the best results across all teacher networks. Although the reason behind it could be complicated, we argue its not solely due to:

- Distribution: Fig. 2 shows T(A) & T(B) demonstrate lowest KL divergence among many others, which means their distributions are cloest. Though distribution is only supervision from KD during learning process, in Table 3, S_2 is the pearl (best student) in the eye of T(A), whereas, S_2 is the last choice to T(B). Therefore it indicates that we need to disentangle distribution into finer singularity.

- Accuracy: T(A) is considered as the most accurate model, however students fail to achieve top performance when compared with distillation by other teachers. [25] argues that the teacher complexity could hinder the learning process, as student does not have the sufficient capacity to mimic teacher's behavior, but its worth noting that S_2 perform significantly better than S_1 even though they have similar capacity. [6] tends to believe the output of a high performance teacher network is not significantly different from ground truth, hence KD become less useful. However, Table 2 suggest otherwise, a lower-performance model T(F), whose distribution is closer to GT than a high-performance model T(A) did, while both of them could spot satisfied student network.

These observations inspire us to rethink the knowledge in a teacher network, which we argue depends not only on its parameters or performance but also on its architecture. In other words, if we distill knowledge only with a pre-defined student architecture like standard KD does, it might forces the student to sacrifice its parameters to learn teacher's architecture, which end up with a non-optimal solution.

Tag	Model name	Input size	Top-1 accuracy
T(A)	EfficientNet-B7 [35]	600	84.4
T(B)	PNASNet-large [18]	331	82.9 74
T(C)	SE-ResNet-154 [11]	224	81.33
T(D)	PolyNet [41]	331	81.23
T(E)	Inception-ResNet-v2 [32]	299	80.217
T(F)	ResNeXt-101 [38]	224	79.431
T(G)	Wide-ResNet-101 [40]	224	78.84
T(H)	ResNet-152 [5]	224	78.31

Table 2. A comparison of popular off-the-shelf models, sorted by top-1 accuracy.

Teacher models				
GT	T(A)	T(B)	T(E)	T(F)
S_3 (65.6)	S_2 (66.6)	S_3 (66.9)	S_1 (67.4)	S_5 (67.1)
S_4 (65.6)	S_3 (66.5)	S_5 (66.4)	S_4 (67.0)	S_1 (67.1)
S_5 (65.5)	S_4 (66.3)	S_4 (66.1)	S_5 (66.9)	S_4 (66.6)
S_1 (65.5)	S_5 (66.0)	S_1 (65.7)	S_3 (66.5)	S_3 (66.3)
S_2 (65.4)	S_1 (65.8)	S_2 (65.4)	S_2 (66.1)	S_2 (66.0)

Table 3. Distillation performance of five student architectures from the MNAS [34] search space under the supervision of different teacher models. The numbers in parentheses denote the top-1 accuracy on ImageNet validation set. In each column, the student models are ordered based on their performance. Note that different teachers have different best students.

3. Architecture-aware knowledge distillation

Our AKD leverages neural architecture search (NAS) to explore student architectures that can better distill the knowledge from a given teacher. In this section, we will describe our KD-guided NAS and discuss our insights.

3.1. KD-guided NAS

Inspired by recent mobile NAS works [34, 35], we employ a reinforcement learning (RL) approach to search for latency-constrained Pareto optimal solutions from a large factorized hierarchical search space. However, unlike previous NAS methods, we add a teacher in the loop and use knowledge distillation to guide the search process. Fig. 3 shows our overall NAS flow, which consists of three major components: an RL agent for architecture search, a search space for sampling student architectures, and a training environment for obtaining KD-guided reward.

RL agent: Similar to other RL-based approaches [43, 45, 34], we use a RNN-based actor-critic agent to search for the best architecture from the search space. Each RNN unit de-

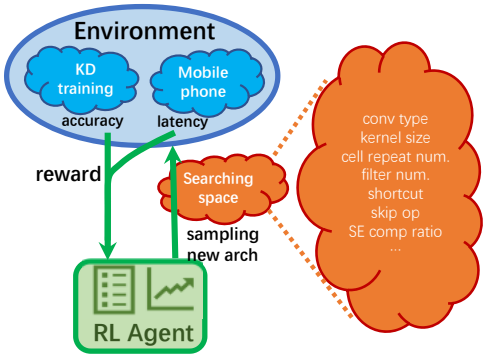


Figure 3. Pipeline of searching process in AKD. there are three core components: environment, RL agent and search space.

termines the probability of different candidates for a search option. The RNN weights are updated using PPO algorithm [30] by maximizing the expected KD-guided reward.

Search Space: Similar to [34], our search space consists of seven predefined blocks, with each block contains a list of identical layers. Our search space allows us to search for the number of layers, the convolution and skip operation type, conv kernel size, squeeze-and-excite ratio, and input/output filter size, for each block independently.

KD-guided reward: Given a sampled student architecture from our search space, we train it on a proxy task to obtain the reward. Unlike prior NAS works, we perform KD training with a teacher network to obtain KD-guided accuracy. Meanwhile, we also measure the latency on mobile devices, and use the same weighted product of accuracy and latency in [34] as the final KD-guided reward to approximate Pareto-optimal solutions.

3.2. Implementation details

Unless otherwise specified, we adopt Inception-ResNet-v2 [32] as the teacher model in ImageNet experiments. Our NAS process mostly follows the same settings in MNAS [34]. We reserve 50K images from training set as mini-val, and use the rest as mini-train. We treat each sampled model as a student, and distill teacher's knowledge by training on the mini-train for 5 epochs, including the first epoch with warm-up learning rate, and then evaluating on the mimi-val to obtain its accuracy. We have also tried to train each sampled model for a longer time (15 epochs) and observe minor performance improvement. After that, the sampled model will be evaluated on the single-thread big CPU core of Pixel 1 phones to measure its latency. Finally, the reward is calculated based on the combination of accuracy and latency. We do not use a shared weight among different sampled models.

We use actor-critic based RL agent for neural architecture search. In each searching experiment, the RL agent samples $\sim$10K models. Then we pick the top models that meet the latency constrain, and train them for further 400 epochs by either distilling the same teacher model or using ground truth labels. Following common practices in KD, temperature is set to 1 and the distilling weight α is set to 0.9. We launch all the searching experiments on a large TPU Donut cluster [14]. Since each sampled model distills the teacher in an online manner, the searching time of AKD is 3$\sim$4 times longer than the conventional NAS. Each experiment takes about 5 days on 200 TPUv2 Donut devices.

3.3. Understanding the searching process

For the better readability, we denote the family of architecture searched by KD-guided reward as **AKDNet**, and the family searched by classification-guided reward as **NASNet** in the following paragraphs. Note that in each comparable experiment we initialize the RL agent with the same random weights and latency target.

Fig. 4 presents some interesting statistical results of an AKD searching process, where the latency target is set to 15ms. We can see that with the soft reward design by [34], all generations of sampled models have the most dense distribution around the target latency in Fig. 4 (A). Fig. 4 (B) illustrates the latency-accuracy trade-off among all sampled models, where the Pareto optimal curve is highlighted. All the quantitative results in the latter sections are produced by the architectures close to this curve. Fig. 4 (C) and (D) show how the latency and accuracy of sampled models changes along the searching generations. It is interesting to see the RL agent firstly targets on higher accuracy, and does not start finding efficient architectures until $\sim$5K generations. Fig. 4 (E) shows the correlation between latency and FLOPS. Intriguingly, consider a same latency (in vertical), we can see AKD favor models with higher FLOPS, which is quite reasonable. More analysis for FLOPS will be shown in Sec. 5.3.

To better investigate how different the AKDNet and NASNet are, Fig. 5 shows how the sampled architecture evolves in the search space. In the first frame, the 'initial' circle denote the architectures sampled by a randomly initialized RL agent. The latter 4 frames show 4 groups of 500 models sampled at different stages. The groups searched by AKD and conventional NAS are separable, indicating the distillation task introduces some new **undiscovered information** to rectify the student architecture evolves into another direction. We demonstrate it ***is*** the structural knowledge of the teacher model in the next section.

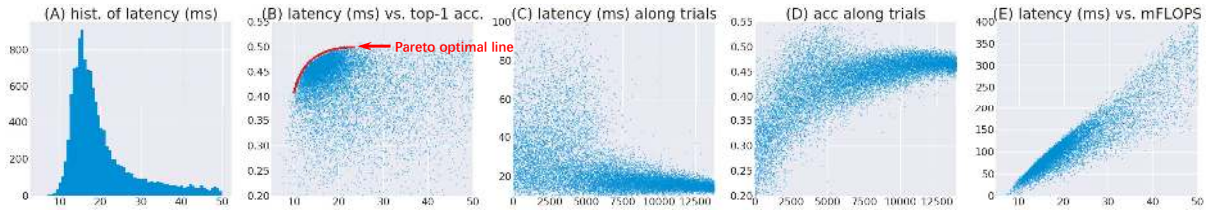
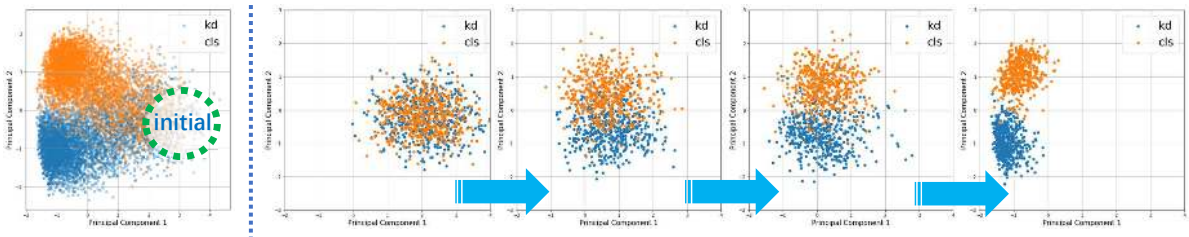


Figure 4. Attributes of searching process. Each dot indicates one architecture.



All generations | Generation 0~500 | Generation 1000~1500 | Generation 2000~3500 | Generation 4000~4500

Figure 5. The architecture evolves during searching. Each dot represents an architecture. Different colors indicate different NAS policies – orange for conventional NAS and blue for AKD based NAS. PCA is used for visualization. Best view in color.

4. Understanding the structural knowledge

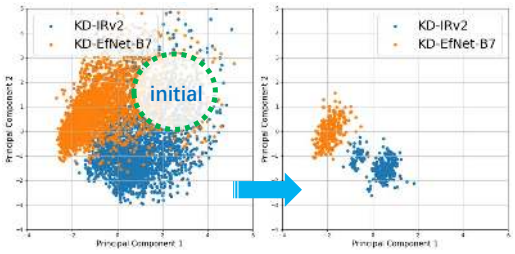
4.1. Existence of structural knowledge

The main assumption of AKD is the knowledge behind the teacher network includes structural knowledge which sometimes can hardly be transferred to parameters. Although the universal approximation theorem [9] asserts that a simple two-layer neural network can approximate any continuous function in an ideal condition, [9] assumes the size of the universal model can be extremely large, which is unpractical.

In this section we investigate the existence of structural knowledge by answering two questions:

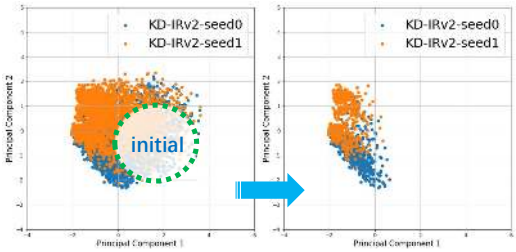
1. If two identical RL agents perform AKD on two different teacher architectures, will they converge to different area in search space?
2. If two different RL agents perform AKD on the same teacher, will they converge to similar area?

Different teachers with same RL agent We answer the first question by launching two AKD searching processes. The teacher models are the off-the-shelf Inception-ResNet-v2 [32] and EfficientNet-B7 [35], respectively. Besides the teacher model, all the other settings, random seed, latency target and mini-val data are fixed to the same setting. Fig. 6 shows the initial distribution (in the green dot circle), all generations (left) and the final converged architectures (right). The final optimal architectures are clearly separable.



All generations | Last 200 generations

Figure 6. All generations searched by AKD on two different teachers. Their final generations converge into different areas. This proves the structural knowledge does exist in the teacher model.



All generations | Last 200 generations

Figure 7. All generations searched by AKD on the same teacher model but different RL agents. Their final generations converge into the same area. Note that there are a large amount of blue dots overlapped by the orange dots. Best view in color.

Same teacher with different RL agent We further doubt whether the separability is produced by the random factor lies in the RL searching program. To answer this question, we launch two AKD searching programs for the same

Generation	initial	∼1k	∼3k	∼10k
Winning ratio	10 / (20-2)	12 / 20	16 / 20	**18 / 20**
Average gain	-0.07	0.10	0.46	**1.05**

Table 4. Relative performance gain between KD(AKDNet) and KD(NASNet) during different searching stages.

teacher model. In the two experiments we set different random seeds for the RL agent and different splits of mini-train / mini-val. These changes will introduce significant random factors for the initial population and reward distribution. We denote them as `KD-IRv2-seed0` and `KD-IRv2-seed1` in Fig. 7. Not surprisingly they finally converge to the close area.

These two experiments jointly prove the structural knowledge does exist in a neural network.

Difference between AKDNet and NASNet In the previous paragraph we investigated the structural knowledge. But it is still too abstract to make a visual comprehension. In this work we try to get some insight by comparing the statistical divergence between the architecture family of AKDNet and NASNet. We first expand the search space to make it continuous: *e.g.* the original 'skip_op' dimension has 4 values for 4 denoting skip operator, and we need to expand it to 4 one-hot dimensions. In this way the original 35-dim space is expanded to 77-dim. After that, we calculate the probability of each operator among the top 100 optimal architectures of AKDNet and NASNet respectively, constrained by a range of latency, and get the difference between them (AKDNet minus NASNet). Result is shown in Fig. 8. Some interesting things are: compared with NASNet, students of Inception-ResNet-v2 favor the larger kernel size but small expand ratio of `depth-wise conv`, and tend to reduce the layer number at the beginning of the network.

4.2. AKDNet becomes KD-friendly along searching

A good metric to measure how AKDNet improves KD result is the relative performance gain. It is defined as:

$$[\mathrm{KD(AKDNet)} - \mathrm{CLS(AKDNet)}] - [\mathrm{KD(NASNet)} - \mathrm{CLS(NASNet)}], \quad (1)$$

in which 'KD' and 'CLS' denote the final performance of the network which will be trained for 400 epochs with KD or classification loss.

We randomly sample 80 architectures from 4 stages among 10,000 generations for both NAS and AKD, and calculate the relative gain in each stage. Tab. 4 shows the results. The wining ratio indicates how many AKDNet models beat the NASNet. It can be observed that the average gain and the wining ratio go up along the searching process. AKDNet becomes more and more friendly to the teacher model distillation.

Latency	searching by	training by	top-1	top-5
15±1 ms	hard label	hard label	59.73	81.39
	hard label	distillation	63.9	84.26
	distillation	hard label	61.4	83.1
	distillation	**distillation**	**66.5**	**87.5**
25±1 ms	hard label	hard label	67.0	87.4
	hard label	distillation	68.1	88.0
	distillation	hard label	67.2	87.5
	distillation	**distillation**	**69.6**	**89.1**
75±1 ms	hard label	hard label	73.0	92.1
	hard label	distillation	74.7	92.54
	distillation	hard label	73.6	92.2
	distillation	**distillation**	**75.5**	**93.1**

Table 5. Ablation study on ImageNet.

Training method	Latency	Advanced KD method		
		TA-KD	RCO-KD	CC-KD
MNet-v2 w/o KD	33ms	65.4		
MNet-v2 w/ KD		67.6 ↑**2.2**	68.2 ↑**2.8**	67.7 ↑**2.3**
AKDNet-M w/o KD	32.8ms	68.9		
AKDNet-M w/ KD		72.0 ↑**3.1**	72.4 ↑**3.5**	72.2 ↑**3.3**

Table 6. AKDNet transfers to other advanced KD method. 'MNet-v2' denotes the MobileNet-v2 0.5×. The 'M' in AKDNet-M denotes the 32.8ms version of ADKNet. Even with a much higher baseline, AKDNet consistently brings considerable gain (∼ 1%) under each KD method compared to MobileNet-v2.

5. Preliminary experiments on ImageNet

Table 5 shows the performance of our models on ImageNet [17], with several target latencies. After search, we pick the top-performing AKDNet and NASNet, and train them until converge by distillation or hard label respectively. As shown in the table, AKDNets consistently outperform NASNet in all settings by a large margin, e.g, AKDNet achieves 61.4% top-1 / 83.1% top-5 accuracy compared to NASNet's 59.73% top-1 / 81.39% top-5 accuracy with 15ms latency. Moreover, If we further train them with distillation, AKDNet improve 61.4% → 66.5% while NASNet 59.73% → 63.9%, which suggests that our framework capable to find architecture that good for distillation.

5.1. Transfer AKDNet to advanced KD methods

Since AKD uses the same training method during searching and final training, it is interesting to investigate whether AKD overfits the original KD policy, such as the

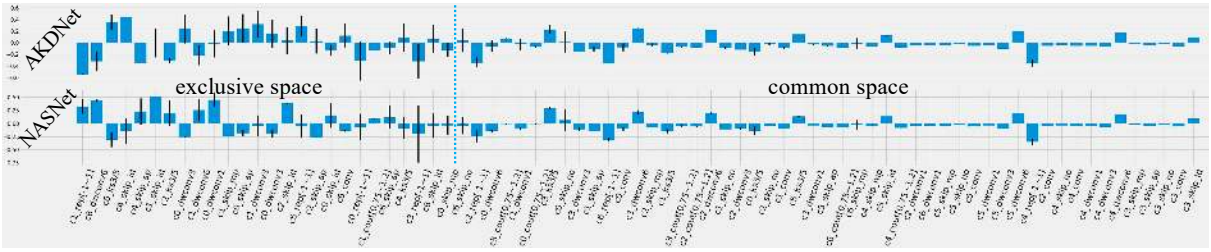


Figure 8. Mean and standard deviation of probability of each operator. Calculated by the 100 optimal AKDNet and NASNet and sorted by the significance of difference.

algorithm and hyper-parameters. So we verify it by transferring the Pareto optimal architecture at ∼33 ms (denoted as AKDNet-M) to the three recent state-of-the-art KD methods, TA-KD [25], RCO-KD [13] and CC-KD [26]. We select the 33 ms to match the most common settings in these three works. Note that since the TA-KD does not have experiments on ImageNet and MobileNet-v2, we use a new implementation that has 3 TA [25] networks which are channel-expanded version of the target student architecture.

Results are shown in Tab. 6. It can be observed that even with a quite strong baseline when training without KD, AKDNet-M still can get more improvement under all KD methods.

5.2. Compare with SOTA architectures

In the previous experiments, we focus on the architectures sampled or searched in the same search space. This may introduces bias into our conclusion. Now we evaluate the performance gain produced by distillation under different architecture settings. In each setting, we first select a state-of-the-art architecture, either designed by human or searched by NAS. Then we select an architecture in the Pareto optimal set of AKD that has the most similar latency with that of the SOTA model. We compare how much improvement can be made when using them to distill the same teacher model (Inception-ResNet-v2). MobileNet-v2 [29], MNasNet [34] and MobileNet-v3 [10] are compared in Tab. 7. We adopt the original KD and the RCO-KD to show the consistent conclusion.

It is worth to mention that this work does not aim at beating the other backbone designing for classification, but finding an architecture that can get more gain when training by KD. However not surprisingly the Tab. 7 also proves the AKDNet can achieve great performance when trained without KD thanks to the good search space.

5.3. Latency vs. FLOPS

As some of recent works [34] argue that latency can be a better metric to describe the computational cost compared with other proxy metrics like FLOPS, we constrain the NAS process by a soft target of latency in all settings.

Latency	architecture	with KD?	top-1	top-5
15∼20 ms	AKDNet		61.4	83.1
	AKDNet	✓	↑2.6	↑3.24
	AKDNet	RCO-KD	↑3.1	↑3.8
	MNet-v2-a		59.2	79.8
	MNet-v2-a	✓	↑1.4	↑2.1
	MNASNet-a		62.2	83.5
	MNASNet-a	✓	↑1.49	↑2.3
	MNet-v3-a		64.1	85.0
	MNet-v3-a	✓	↑1.3	↑ 2.2
25∼27 ms	AKDNet		67.2	87.5
	AKDNet	✓	↑2.4	↑1.6
	AKDNet	RCO-KD	↑2.8	↑1.5
	MNASNet-b		66.0	86.1
	MNASNet-b	✓	↑1.1	↑0.6

Table 7. Comparison to the state-of-the-art architecture. AKDNet has the comparable results when trained without KD, and its performance get improved by a large margin when trained with KD. We reduce the channels of all the compared model with the same reduction ratio in each layer to obtain a comparable latency. We select the AKDNet to match the lower bound latency in each comparable group.

However it is still interesting to see the exact correlation between latency and FLOPS in our search space. Figure 4 (e) presents how FLOPS changes *w.r.t.* the latency, which covers around 14,000 models in the searching trajectory for the 15ms-target. By performing a simple linear regression, we observe that the FLOPS and latency are empirical linearly correlated as follows:

$$3.4 \times (\text{latency} - 7) \leq \text{mFLOPS} \leq 10.47 \times (\text{latency} - 7)$$

The variance becomes larger when the model goes up. To verify whether the conclusion still holds on searching by a slack FLOPS constraint, we replace the latency term in our reward function by a FLOPS term, and set the target to 300 mFLOPS. Tab. 8 shows AKD leads to consistent improvement regardless of how we define the searching target and how we distill the teacher.

	searching by	training by	top-1	top-5
NASNet	hard label	hard label	69.92	89.1
	hard label	distillation	71.2	90.4
AKD	distillation	hard label	70.0	89.4
	distillation	**distillation**	**72.1**	**91.7**
	distillation	**RCO-KD**	**73.0**	**92.2**

Table 8. ImageNet Performance with FLOPS-constrained AKD. All settings are the same as in Tab. 5 except using FLOPS (rather than latency) as the reward. Target FLOPS is about 300M. Despite using different reward, AKD consistently improves performance over NASNet.

Training method	KD method	Distractor num. 1e3	1e4	1e5	1e6
Teacher	-	99.56	99.3	99.0	98.2
MNet-v2	-	91.49	84.45	75.6	65.9
MNet-v2	CC-KD	97.93	95.76	91.99	86.29
MNet-v2	RCO-KD	98.29	95.01	90.97	85.9
AKDNet-M	-	93.8	86.4	78.2	68.6
AKDNet-M	CC-KD	98.26	97.48	93.85	88.41
AKDNet-M	RCO-KD	98.42	97.56	94.1	90.2

Table 9. Transfer the AKDNet on MegaFace. The teacher model in all KD settings is Inception-ResNet-v2.

6. Towards million level face retrieval

It is much harder to learn a complex data distribution for a tiny neural network, but some previous works [13, 26] have shown that a huge improvement can be achieved by introducing KD in complex metric learning. We adopt the most challenging face retrieval benchmark MegaFace [15] to verify the transferability of our searched models. The MegaFace contains 3,530 probe faces and more than 1 million distractor faces. The target is to learn a neural network on an irrelevant training set like MS-Celeb-1M [4] and test the model on MegaFace in a zero-shot retrieval manner.

We first train an Inception-ResNet-v2 on the MS-Celeb-1M [4], pre-processed by RSA [21], and directly distill its representation to the AKDNet-M (described in Table 6). The distillation setting between ImageNet and face retrieval has a little bit difference. Since MegaFace is an open-set zero-shot benchmark, we just distill the output feature of the penultimate layer by minimizing the mean square error. It is worth emphasizing that the AKDNet-M is searched on the ImageNet with Inception-ResNet-v2 as a teacher model. We hope it can benefit from the learned structural knowledge and directly transfer it to the face retrieval task. Respecting the common setting in this area, we report the Top-1 accuracy under different number of distractors when performing retrieval. Results are shown in Tab. 9. With a much higher baseline performance, AKDNet further achieves significant improvement on two different KD methods.

Training Method	Architecture	Training method	Distractor num. 1e5	1e6
Ensemble Teacher	HRNet-w48 [36] + R100 [2] + EPolyFace [20] + IncRes-v2 [33] + SE154 [12]	-	99.6	99.3
AKDNet-M	AKDNet	hard label	78.2	68.6
AKDNet-M	AKDNet	original-KD	94.9	90.1
AKDNet-M	AKDNet	RCO-KD	**95.7**	**90.9**

Table 10. AKD with ensemble teacher. We ensemble 5 state-of-the-art models, and use it as a teacher to compare hard label, original-KD, and RCO-KD. Our AKDNet-M achieves consistent improvements when teacher is used (either in original-KD or RCO-KD).

7. Neural architecture ensemble by AKD

The original knowledge distillation [8] can significantly improve the single model performance by distilling the knowledge in an ensemble of models. It is skeptical whether the AKDNet over-fits the teacher architecture and whether it still works well when the teacher model is an ensemble of multiple models whose architectures are different. In Tab. 10, we show 5 current state-of-the-art models on MegaFace and use their ensemble to be the teacher. We can observe that AKDNet can still leverage other teachers' knowledge, and obtain consistent improvements with either original-KD or RCO-KD compared to the hard label training without teacher.

8. Conclusion & further thought

This paper is the first that points out the significance of structural knowledge of a model in KD, motivated by the inconsistent distillation performance between different student and teacher models. We raise the presumption of structural knowledge and propose a novel RL based architecture-aware knowledge distillation method to distill the structural knowledge into students' architecture, which leads to surprising results on multiple tasks. Further, we design some novel approach to investigate the NAS process and experimentally demonstrate the existence of structural knowledge.

The optimal student models in AKD can be deemed as the most structure-similar to the teacher model. This implies a similarity metric of the neural network may occur but never be discovered. It is interesting to see whether we can find a new metric space that can measure the similarity between two arbitrary architectures. If so, it would nourish most of the areas in machine learning and computer vision.

References

[1] J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search, 2012. 1

[2] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019. 8

[3] Tommaso Furlanello, Zachary C. Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks, 2018. 2

[4] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *European Conference on Computer Vision*, pages 87–102. Springer, 2016. 8

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 1, 3

[6] Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation, 2019. 1, 3

[7] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdelrahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, and Tara Sainath. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29, November 2012. 1

[8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1, 2, 8

[9] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991. 5

[10] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. *ICCV*, 2019. 7

[11] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2017. 3

[12] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 8

[13] Xiao Jin, Baoyun Peng, Yichao Wu, Yu Liu, Jiaheng Liu, Ding Liang, Junjie Yan, and Xiaolin Hu. Knowledge distillation via route constrained optimization. *arXiv preprint arXiv:1904.09149*, 2019. 7, 8

[14] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-datacenter performance analysis of a tensor processing unit, 2017. 4

[15] Ira Kemelmacher-Shlizerman, Steven M Seitz, Daniel Miller, and Evan Brossard. The megaface benchmark: 1 million faces for recognition at scale. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4873–4882, 2016. 8

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 1

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 6

[18] 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 *The European Conference on Computer Vision (ECCV)*, September 2018. 3

[19] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search, 2018. 1

[20] Yu Liu et al. Towards flops-constrained face recognition. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019. 8

[21] Yu Liu, Hongyang Li, Junjie Yan, Fangyin Wei, Xiaogang Wang, and Xiaoou Tang. Recurrent scale approximation for object detection in cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 571–579, 2017. 8

[22] Yu Liu, Jihao Liu, Ailing Zeng, and Xiaogang Wang. Differentiable kernel evolution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1834–1843, 2019. 1

[23] Yu Liu, Fangyin Wei, Jing Shao, Lu Sheng, Junjie Yan, and Xiaogang Wang. Exploring disentangled feature representation beyond face identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2080–2089, 2018. 1

[24] Yu Liu, Junjie Yan, and Wanli Ouyang. Quality aware network for set to set recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5790–5799, 2017. 1

[25] Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher, 2019. 3, 7

[26] Baoyun Peng, Xiao Jin, Jiaheng Liu, Dongsheng Li, Yichao Wu, Yu Liu, Shunfeng Zhou, and Zhaoning Zhang. Correlation congruence for knowledge distillation. In *Proceedings*

of the IEEE International Conference on Computer Vision, pages 5007–5016, 2019. 7, 8

[27] Mary Phuong and Christoph Lampert. Towards understanding knowledge distillation. In *International Conference on Machine Learning*, pages 5142–5151, 2019. 2

[28] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search, 2018. 1

[29] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *CVPR*, 2018. 7

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 4

[31] Richard Shin*, Charles Packer*, and Dawn Song. Differentiable neural network architecture search, 2018. 1

[32] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning, 2016. 1, 2, 3, 4, 5

[33] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. 8

[34] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. *CVPR*, 2019. 1, 2, 3, 4, 7

[35] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *ICML*, 2019. 1, 2, 3, 5

[36] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *CoRR*, abs/1908.07919, 2019. 8

[37] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016. 1

[38] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 3

[39] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer, 2016. 1

[40] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2016. 3

[41] Xingcheng Zhang, Zhizhong Li, Chen Change Loy, and Dahua Lin. Polynet: A pursuit of structural diversity in very deep networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 3

[42] Ying Zhang, Tao Xiang, Timothy M. Hospedales, and Huchuan Lu. Deep mutual learning, 2017. 2

[43] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *ICLR*, 2017. 1, 2, 3

[44] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition, 2017. 1

[45] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. *CVPR*, 2018. 3