ScaleKD: Strong Vision Transformers Could Be Excellent Teachers

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this paper, we question if well pre-trained vision transformer (ViT) models could be used as teachers that exhibit scalable properties to advance cross architecture knowledge distillation research, in the context of adopting mainstream large-scale visual recognition datasets for evaluation. To make this possible, our analysis underlines the importance of seeking effective strategies to align (1) feature computing paradigm differences, (2) model scale differences, and (3) knowledge density differences. By combining three closely coupled components namely *cross attention projector*, *dual-view feature mimicking* and *teacher parameter perception* tailored to address the alignment problems stated above, we present a simple and effective knowledge distillation method, called *ScaleKD*. Our method can train student backbones that span across a variety of convolutional neural network (CNN), multi-layer perceptron (MLP), and ViT architectures on image classification datasets with significantly improved performance, achieving state-of-the-art distillation performance. For instance, taking a well-trained Swin-L as the teacher model, our method gets 75.15%|82.03%|84.16%|78.63%|81.96%|83.93%|83.80%|85.53% top-1 accuracies for MobileNet-V1|ResNet-50|ConvNeXt-T|Mixer-S/16|Mixer-B/16|ViT-S/16|Swin-T|ViT-B/16 models trained on ImageNet-1K dataset from scratch, showing 3.05%|3.39%|2.02%|4.61%|5.52%|4.03%|2.62%|3.73% absolute gains to the individually trained counterparts with the same experimental settings. Intriguingly, when scaling up the size of teacher models or their pre-training datasets, our method showcases larger gains to student models. Empirically, the student backbones trained by our method transfer well on downstream MS-COCO and ADE20K datasets. Moreover, our method shows the potential to be an efficient substitution for the time-intensive pre-training of any target student on large-scale datasets if a strong pre-trained ViT is available. The code will be released soon.

1 Introduction

The great success of deep learning in computer vision (CV) has been driven by an explosion of neural network architectures among which convolutional neural networks (CNNs) [1–3], vision transformers (ViTs) [4, 5] and multi-layer perceptrons (MLPs) [6–8] are three major model categories. While CNNs were the de facto models for about a decade, recent progress shows that large ViT models have attained state-of-the-art performance on many visual recognition tasks such as image classification, image segmentation, and object detection. In principle, ViTs extend the philosophy of predominant transformer architectures [9] in natural language processing (NLP) to vision tasks. They convert an image into a sequence of equal-sized patches treated as tokens resembling words in NLP, then apply the dot-product self-attention mechanism over the sequence of image patches. ViTs designed in this way couple with a powerful data-hungry learning paradigm: models are first pre-trained on massive datasets (with supervised or self-supervised [10, 11] or cross-modality learning [12, 13]) and then fine-tuned on target datasets (with supervised learning). As the size of ViT models and pre-training datasets increases, the pre-trained models tend to have improved generalization performance. Despite

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

this notable model performance scalability, the pre-training process of ViTs leads to significantly huge expenses. Besides, in industry, CNNs and MLPs are still widely used, due to the wider availability of effective implementations and optimizations compared to ViTs. Furthermore, large models including ViTs, CNNs, and MLPs are memory-hungry and computationally intensive, prohibiting the deployment of them in many resource-constrained application scenarios.

In parallel, knowledge distillation (KD) has proven to be a promising model compression pathway and has attracted lots of research interests. It relies on a teacher-student framework that transfers the knowledge learned by a large teacher model to a compact student model, aiming to make the student model can have improved performance to substitute the teacher model in deployment. However, most existing KD methods [14–35] focus on CNN architectures, and usually perform evaluation on small datasets with non-mainstream student models for industrial applications, lagging far behind the evolution of neural network architectures. In this paper, *we attempt to connect knowledge distillation research with well pre-trained ViT models that stand out for their remarkable scalability, via a new perspective*. Specifically, *we question* whether well pre-trained ViT models could be used as teachers that effectively transfer scalable properties to target student models with different typed architectures such as CNNs and MLPs or heterogeneous ViT structures (in this work, we refer 'cross architecture KD' to such a more generalized formulation), in the context of using mainstream large-scale visual recognition benchmarks. Note there have been few recent efforts on cross architecture KD research, but they explored narrow focuses such as logits distillation [36] and feature transform [37, 38], following the paths previously explored in CNNs based KD methods. To answer the question in our motivation, we think the knowledge transfer difficulties are rooted in the following three aspects of differences: (1) *Differences in feature computing paradigm*. In terms of semantic units, ViTs operate on a sequence of equal-sized image patches added with positional embeddings, whereas CNNs operate on regular grids of pixels. In terms of core operations, ViTs rely on self-attention operations to model global feature dependencies, whereas CNNs rely on convolution operations to model local features. Although MLPs also use a patchify stem as ViTs, they rely on fully connected operations instead of self-attention operations and do not use positional embeddings, showing inferior feature learning ability. These differences in feature computing paradigm pose the first knowledge transfer barrier to overcome. (2) *Differences in model scale*. On the micro scale, model scale differences among ViTs, CNNs, and MLPs lie in network width, network depth, building blocks, etc. On the macro scale, model scale differences come from the capability of scaling the model size for ViTs, CNNs and MLPs towards better performance and generalization ability. As a result, these differences in model scale make the capacity of different network architectures typically vary significantly, emerging as the second knowledge transfer barrier to address. (3) *Differences in knowledge density*. Under the prevalent pre-training and fine-tuning paradigm, when scaling up pre-training datasets, large ViTs usually exhibit obviously superior performance scalability than top-performing CNNs and MLPs in terms of fine-tuning on both upstream image classification tasks and downstream dense prediction tasks [5, 39, 40]. As for knowledge distillation in this work, we assume that pre-training datasets are no longer accessible and only well pre-trained ViT teacher models are available, avoiding expensive pre-training costs and making the setting suited for real applications. When training student models on upstream image classification datasets like ImageNet-1K, the knowledge density between teacher and student models is different, which appears as the third barrier to handle.

From the above analysis, we can conclude that the design of effective schemes to align (1) feature computing paradigm differences, (2) model scale differences, and (3) knowledge density differences between the pre-trained ViT teacher and target student models, plays the key role to attain our goal. Accordingly, we present scalable knowledge distillation (ScaleKD), a simple and effective cross architecture KD method, which addresses them in a progressive manner. Regarding the first alignment problem, we notice that some previous works leverage cross attention mechanisms to align different modalities [41–43]. Extending this philosophy to cross architecture distillation, we propose our *cross attention projector (CAP)* (shown in Figure 1(a)), bridging the gaps across different semantic units and core operations. For one, CAP utilizes positional embedding and patchify stem to transform the semantic units of CNN and MLP into transformer-like tokens. For another, by employing trainable queries that share the same attributes as the teacher's features, the student's features can be globally queried, modeling interdependencies based on positional information. In this way, CAP aligns feature computing differences in form and serves as the base component of the other two designs. Regarding the second and third alignment problems, as the high model capacity and the pre-training are inextricably bound up with each other for ViTs, we think they co-act on models' feature space and parameter space. Surprisingly, we observe two interesting phenomena related to the two spaces:

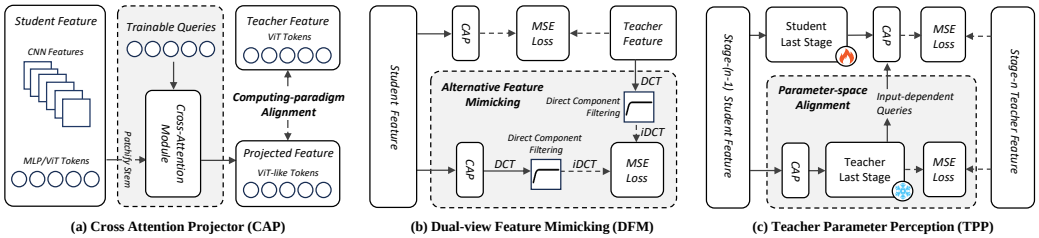


Figure 1: Overview of three core components in our ScaleKD, which are cross attention projector (left), dual-view feature mimicking (middle), and teacher parameter perception (right). Notably, the teacher's parameters are frozen in the training stage,

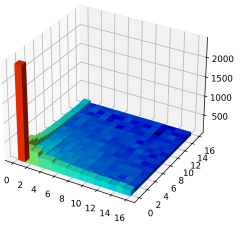
- As shown in Figure 2, the frequency distributions of the features of the pre-trained ViTs are extremely imbalanced, where the direct component (zero frequency) response is dominant among all frequencies. Conducting feature distillation under such an imbalanced distribution may neglect the features of all other alternative components.
- As the parameters in the fine-tuning stage are slightly changed, the pre-training knowledge remains in the parameter space of the pre-trained ViTs. Although the pre-training datasets are invisible, the student still has the opportunity to obtain the knowledge by aligning its parameter space to the teacher's.

Inspired by these two insightful observations, we try to tackle the problems from two new perspectives. Based on the first observation, we design *dual-view feature mimicking (DFM)* (shown in Figure 1(b)), whose key insight is to complement the neglected alternative features in KD process. Specifically, DFM employs CAP as the feature projector and incorporates two feature mimicking paths. In the first path, DFM conducts feature mimicking in the original space to learn the teacher's global features. In the second path, by removing the direct component in frequency space, DFM highlights the subtle alternative responses in feature mimicking, thus avoiding the neglect of these features. As a result, the two paths are complementary to each other, jointly promoting the feature space alignment. Based on the second observation, we propose *teacher parameter perception (TPP)* (shown in Figure 1(c)), whose target is to align the teacher's and student's parameter spaces. Thanks to the aligned feature computing paradigm by CAP, TPP could bridge the student's early stages and the teacher's later stages and form a proxy feature processing path. By applying feature distillation in this path, the student's parameter space would be gradually aligned to the teacher's. In this way, the pre-training knowledge could be transferred from the teacher to the student. Notably, the distillation learning processes in feature space and parameter space are the two sides of the same coin, indicating that DFM and TPP are mutually promoted by each other. Thanks to progressive designs, CAP, DFM, and TPP can be seamlessly integrated into a neat and effective cross-architecture knowledge distillation method, called *ScaleKD*, which addresses the above three problems simultaneously. Although ScaleKD has multiple feature mimicking paths, they only exist in the training stage, which means that ScaleKD does not alter the student's structure and introduces no additional cost in the inference stage. By conducting systematic experiments on several large mainstream benchmarks, we validate the effectiveness and generalization ability of our method.

2 Scalable Knowledge Distillation

Given a pre-trained ViT teacher having m stages and a target student (CNN or MLP or ViT) having n stages, let F^{s_i} and F^{t_j} denote features from *i-th* stage of the student and *j-th* of the teacher, respectively. In what follows, we formulate all components in the form of performing feature distillation, for better clarifying the relationships of three coupled components in our method.

Figure 2: Feature distribution of BEiT-L [40] in the frequency domain, where the direct component response is dominant. Details on drawing this figure are shown in Figure 5.



2.1 Cross Attention Projector

As shown in Figure 1(a), CAP adopts the structure of the standard transformer decoder block, consisting of a transformer decoder layer and an MLP layer, but incorporates three critical modifications: i) patchifying regular grids of pixels in CNN; ii) adding

positional embedding; iii) setting queries in the transformer decoder block as trainable variables that share the same resolution with the teacher's features. The first two modifications intend to narrow the discrepancy between different semantic units, while the last modification grants the employed transformer decoder block great flexibility in aligning feature semantics and spatial resolution. Based on these modifications, the cross attention operation further provides global dependencies to the projected features. With the projected features by CAP, the feature distillation loss is defined as: [1]:

$$\mathcal{L}_{CAP} = \alpha L(F^t, f_p(F^s; q)) = \alpha ||F^t - f_p(F^s; q)||_2^2, \tag{1}$$

where f_p, q, $\alpha(\geq 0)$ and $L(\cdot)$ denote the CAP and the trainable queries, the loss weight, and the L_2-normed distance, respectively.

2.2 Dual-view Feature Mimicking

As shown in Figure 1(b), building upon CAP, DFM contains two feature mimicking paths. As we stated in the Introduction section, the first path aims to learn the teacher's global features and the second path aims to mimic the neglected alternative features. Specifically, in the first path, DFM conducts feature mimicking in the teacher's original feature space, which is formulated as: $\mathcal{L}_{ori} = \alpha L(F^t, f_{p_1}(F^s))$, where f_{p_1} denotes the CAP in the first path. In the second path, the dominant direct component should be removed. To achieve this goal, we employ Discrete Cosine Transform (DCT), which maps the feature from the spatial domain to the frequency domain: $DCT : \mathcal{X} \rightarrow \mathcal{Z}$. We then define an operator ϕ that removes direct component response from the features:

$$\phi(x) = DCT^{-1}(\sigma(DCT(x))) \qquad s.t.\ \sigma(z) = \begin{cases} 0, & z = 0 \\ z, & z \neq 0 \end{cases}. \tag{2}$$

Next, feature mimicking in the second path is formulated as: $\mathcal{L}_{alt} = \alpha L(\phi(F^t), \phi(f_{p_2}(F^s)))$, where f_{p_2} denotes the CAP in the second path. Now, the feature distillation loss of DFM is formulated as:

$$\mathcal{L}_{DFM} = \beta \mathcal{L}_{ori} + (1 - \beta)\mathcal{L}_{alt}, \tag{3}$$

where $\beta \in [0, 1]$ denotes the balancing weight.

2.3 Teacher Parameter Perception

As we stated in the Introduction section, TPP establishes a proxy feature processing path by connecting the student's early stages with the teacher's later stages by a CAP. In our implementation, this proxy path consists of the student's first $n-1$ stages and the teacher's last stage, as illustrated in Figure 1(c). By conducting feature mimicking in this path, the parameters of the student part are gradually aligned with the parameters of the teacher part, thus enabling the transfer of the teacher's pre-training knowledge. The output features of this path are $F^{st} = g_{t_m}(f_p^{st}(F^{s_{n-1}}; q))$, where g_{t_m} and f_p^{st} denote the teacher's last stage and CAP in the proxy path, respectively. The feature mimicking in the proxy path is formulated as: $\mathcal{L}^{st} = \alpha L(F^t, F^{st})$. We further introduce F^{st} as input-dependent queries for the CAP in the original path. This design aims to enhance the capability of CAP, as such queries contain more teacher-related information. Under such condition, the feature mimicking loss is changed to $\mathcal{L}^s = \alpha L(F^t, f_p^s(F^{s_n}; F^{st}))$. As the two paths are treated equally, the feature distillation loss of TPP is defined as:

$$\mathcal{L}^{TPP} = \mathcal{L}^s + \mathcal{L}^{st}. \tag{4}$$

2.4 Formulation of ScaleKD

From a general perspective, the progressive designs of our above three components are naturally cohesive. As CAP serves as the basic component in DFM and TPP, we further introduce how to apply DFM in TPP and thus get a neat formulation of our method, ScaleKD. Specifically, if treating DFM as an improved version of traditional feature mimicking, it can substitute the original feature mimicking in each path of TPP. In this way, we present our overall design - ScaleKD, which is formulated as:

$$\mathcal{L}_{ScaleKD} = \mathcal{L}_{task} + \underbrace{\beta\mathcal{L}^s_{ori} + (1-\beta)\mathcal{L}^s_{alt}}_{\textit{DFM for TPP Student Path}} + \underbrace{\beta\mathcal{L}^{st}_{ori} + (1-\beta)\mathcal{L}^{st}_{alt}}_{\textit{DFM for TPP Teacher Path}} + \mathcal{L}_{kd}, \tag{5}$$

[1] We normalize each token before calculating distillation loss.

Table 1: Pilot experiments on cross architecture distillation with ScaleKD and FD. s_i denotes the distillation is conducted on stage-i. To clearly show the performance gain, experiments in this table are conducted without L_{kd}.

Teacher	Student	Method	Top-1(%)	ΔTop-1(%)
Swin-S (83.02)	ResNet-50	*Baseline*	76.55	-
		FD (s_4)	77.43	+0.88
		FD (s_3,s_4)	77.74	+1.19
		ScaleKD	79.30	+2.75
	Mixer-S	*Baseline*	74.02	-
		FD (s_4)	74.88	+0.86
		FD (s_3,s_4)	75.32	+1.30
		ScaleKD	77.24	+3.22

Table 2: Pilot experiments on scaling up the teachers. The advanced training strategy has more advanced data augmentation, optimizer, and longer training epochs, as shown in Table 10.

Teacher	Student	Ratio T/S Params	Top-1(%)	ΔTop-1(%)
Traditional Training Strategy				
Baseline	ResNet-50	-	76.55	-
Swin-S (83.02)		1.94×	79.62	+3.07
Swin-B (85.16)		3.43×	79.80	+3.25
Swin-L (86.24)		7.68×	80.10	+3.55
Advanced Training Strategy				
Baseline	ResNet-50	-	78.64	-
Swin-S (83.02)		1.94×	81.43	+2.79
Swin-B (85.16)		3.43×	81.77	+3.13
Swin-L (86.24)		7.68×	82.03	+3.39

where $\beta \in [0, 1]$ indicates the balancing weight and $\mathcal{L}_{task}$ is the cross-entropy loss. Besides, following previous KD research, we use the vanilla logits-based KD loss [14]. As the features are standardized, we set $\alpha = 1$ as the default. Thus, we only have one hyper-parameter β in our method.

3 Experiments

3.1 Datasets and General Experimental Setups

We systematically validate the efficacy of our method under different setups: i) training various student backbones by our ScaleKD on ImageNet-1K [44] (IN-1K) and showing performance gain compared to from-scartch counterparts; ii) conducting transfer learning on downstream tasks on MS-COCO [45] and ADE20K [46] to examine whether the performance gains from our method could be well preserved; iii) showcase linear probing on CIFAR-100 [44] to explore the generalization ability of our method across different data distributions. *More details are shown in Appendix A and B*.

3.2 Experiments on Image Classification Task

To obtain pre-training knowledge from large pre-trained ViTs, we design a series of experiments to extensively study the effectiveness and potentials of our method step by step.

Step 1: Pilot Experiments on Our Basic Settings

Cross Architecture Knowledge Distillation. To illustrate the difficulty of cross architecture feature distillation and validate the efficacy of ScaleKD under this setting, we compare our ScaleKD with traditional feature distillation (FD) [15] on two different cross architecture teacher-student network pairs. From the results shown in Table 1, we can observe: i) Due to architecture gaps between the teacher and the student, traditional FD shows limited gains; ii) Comparatively, our ScaleKD achieves significantly better performance, bringing 2.75%|3.22% absolute top-1 gain for ResNet-50|Mixer-S. By now, we preliminarily verify that ScaleKD could effectively handle cross architecture feature distillation, which is difficult for traditional FD.

Large Pre-trained ViTs as Teachers. After showcasing the effectiveness of ScaleKD in cross architecture KD, we further examine the rationality of selecting large pre-trained ViT as teachers. Specifically, we gradually scale up the teacher's model size and capability under two different training strategies. From the results shown in Table 2, we can observe: *i) ScaleKD can adapt large-scale teachers.* It can be seen that the performance gain is constantly increased under both training strategies when scaling up the teacher's model size; *ii) ScaleKD can adapt teachers' training strategies.* It can be seen that our ScaleKD still brings comparable performance gain under the advanced training strategy, although the baseline performance is relatively high.

Conclusion. According to the above pilot experiments, *our ScaleKD shows basic capabilities on handling cross architecture distillation from large pre-trained ViTs to CNN and MLP students.*

Step 2: Main Experiments on Various Teacher-Student Network Pairs

Based on the conclusion of the first step, we move forward and perform extensive experiments to examine the generalization ability of ScaleKD, considering more teacher-student network pairs. Specifically, we design **eleven** teacher-student network pairs, by choosing two large teachers and **ten** popular models for students, covering mainstream architectures across ViT, MLP, and CNN.

Table 3: Main results of ScaleKD on eleven teacher-student network pairs. † denotes the model pre-trained on IN-22K [44] and ‡ denotes the model pre-trained by EVA [40], an MIM distillation on IN-22K from CLIP-based ViT. Underlined results are for the experiments performed by us.

Teacher	Student	Params (M)		FLOPs (G)		Accuracy (%)	
		T	S	T	S	Top-1	ΔTop-1
Swin-L† (86.24)	MobileNet-V1 (<u>72.10</u>)	196.53	4.23	34.04	0.58	75.15	+3.05
	ResNet-50 (<u>78.64</u>)		25.56		4.12	82.03	+3.39
	ConvNeXt-T (82.14)		28.59		4.46	84.16	+2.02
	Mixer-S/16 (74.02)	196.53	18.53	34.04	3.78	78.63	+4.61
	Mixer-B/16 (76.44)		59.88		12.61	81.96	+5.52
	ViT-S/16 (79.90)	196.53	22.05	34.04	4.61	83.93	+4.03
	Swin-T (81.18)		28.29		4.36	83.80	+2.62
	ViT-B/16 (81.80)		86.57		17.58	85.53	+3.73
BEiT-L/14‡ (88.58)	ResNet-50 (<u>78.64</u>)	304.14	25.56	81.06	4.12	82.34	+3.70
	Mixer-B/14 (<u>76.62</u>)		59.88		16.45	82.89	+6.27
	ViT-B/14 (<u>82.02</u>)		86.57		23.09	86.43	+4.41

Table 4: Performance comparisons with recent top KD methods. Following their settings, the students are trained under the advanced training strategy. Best results are **bolded**.

Model	Method	Teacher	# Epochs	Top-1(%)
Swin-T	From Scratch	-	300	81.18
	DIST [47]	**Swin-L (86.30)**	300	82.30
	DiffKD [48]	**Swin-L (86.30)**	300	82.50
	ScaleKD	Swin-L (86.24)	300	**83.80**
ResNet-50	From Scratch	-	300	78.60
	DIST[47]	Swin-L (86.30)	450	80.20
	DiffKD [48]	Swin-L (86.30)	450	80.50
	OFA [36]	**ViT-B (86.53)**	300	81.33
	ScaleKD	Swin-L (86.24)	300	**82.03**
	FunMatch [49]	BiT-Res152x2 (N/A)	1200	81.54
	FunMatch [49]	BiT-Res152x2 (N/A)	9600	82.31
	ScaleKD	Swin-L (86.24)	600	**82.55**

Table 5: Performance comparisons with various models. More comparisons are shown in Table 15 in the Appendix.

Model	Params (M)	FLOPs (G)	Top-1(%)
CNN-based Architecture			
ResNet-50 [2]	22.56	4.12	78.64
ResNet-50 + ScaleKD	22.56	4.12	82.55
ConvNext-T [3]	28.59	4.46	82.14
RepViT-2.3M [50]	22.90	-	82.50
MLP-based Architecture			
Mixer-B/16 [6]	59.88	12.61	76.44
Mixer-B/16 + ScaleKD	59.88	12.61	81.96
gMLP-B [8]	73.00	15.80	81.60
ResMLP-B24 [7]	115.7	23.00	81.00
ViT-based Architecture			
ViT-S/16 [4]	22.05	4.61	79.90
ViT-S/16 + ScaleKD	22.05	4.61	83.93
Swin-T [5]	73.00	15.80	81.18
Swin-B [5]	87.77	15.14	83.50

Experimental Results. From the results shown in Table 3, we observe: i) In general, our ScaleKD shows great generalization ability across various teacher-student network pairs, no matter for CNN, MLP and ViT students. Over eleven teacher-student pairs, the mean top-1 accuracy improvement reaches 3.94%, and the maximal gain is 6.27%; ii) Considering the acceleration, with Swin-L as the teacher, ResNet-50|Mixer-S/16|ViT-S/16 trained by ScaleKD even outperforms individually trained ResNet-152|Mixer-B/16|ViT-B/16 by a margin of 0.28%|2.19%|2.13%, achieving over 2.35×|3.23×|3.83× compression on model parameter size; iii) The absolute performance gain could still be increased, when choosing stronger teachers. For instance, ScaleKD bring 0.32% additional gain to ResNet-50 when changing the teacher from Swin-L to BEiT-L/14.

Step 3: Performance Comparisons with Three Different Families of Methods

In Step 2, we have verified our ScaleKD could help various student models obtain large performance gains. To further evaluate the value of these gains, we make three kinds of comparisons.

Comparisons with KD Methods. We first compare our ScaleKD with various top KD methods, to demonstrate the superiority of our method. From the results shown in Table 4, we can see that: i) Compared to DIST, DiffKD and OFA, although our teacher is not the best and the number of training epochs is not the largest, our ScaleKD still outperforms all these methods by clear margins (0.70%|1.30% on ResNet-50|Swin-T); ii) Comparing to FunMatch, our cross architecture distillation is even superior, outperforming FunMatch by a margin of 0.24% but only using less than 1/10 training epochs.

Table 6: Performance comparisons with various pre-training methods. $\Rightarrow$ denotes transfer learning and * denotes the model is trained and tested with 384$\times$ 384 resolution samples.

Model	Method	Training Dataset	Dataset Samples $\times$ Epochs (M)	Viewed Samples (M)	Top-1(%)
Supervised pre-training					
ViT-B/16	Pre-training [4]	IN-22K $\Rightarrow$ IN-1K	13.7$\times$90 + 1.28$\times$32	1274	83.97
		JFT-300 $\Rightarrow$ IN-1K	300$\times$7 + 1.28$\times$32	2141	84.15
	ScaleKD	IN-1K	1.28$\times$300	384	85.53
Self-supervised pre-training					
ViT-B/16	BEiT [39]	IN-22K $\Rightarrow$ IN-1K	13.7$\times$150 + 1.28$\times$100	2183	83.70
	iBOT [11]	IN-22K $\Rightarrow$ IN-1K	13.7$\times$320 + 1.28$\times$100	4512	84.40
	ScaleKD	IN-1K	1.28 $\times$ 300	384	85.64
Cross-modal pre-training					
ViT-B/16	CLIP [13]	LAION-2B $\Rightarrow$ IN-1K	2320$\times$32 + 1.28$\times$50	74304	85.47
		LAION-2B $\Rightarrow$ IN-12K $\Rightarrow$ IN-1K	2320$\times$32 + 12.1$\times$60 + 1.28$\times$50	75030	86.17
	ScaleKD	IN-1K	1.28 $\times$ 300	384	86.19
EVA Hybrid pre-training (MIM distillation from the cross-modal pre-trained teacher)					
EVA02-S/14*	EVA-02 [51]	IN-22K $\Rightarrow$ IN-1K	13.7$\times$240 + 1.28$\times$50	3352	85.80
	ScaleKD	IN-1K	1.28 $\times$ 300	384	86.22

Table 7: Transfer learning performance on MS-COCO.

Framework	Backbone	Pre-training	Classification (IN-1K) Top-1(%)	Object Detection (COCO) AP	AP_S	AP_M	AP_L	Instance Segmentation (COCO) AP	AP_S	AP_M	AP_L
Mask R-CNN	ResNet-50	*Baseline*	78.64	40.2	23.0	44.3	52.5	37.1	18.0	40.1	54.9
		Ours	82.03 (+3.39)	42.3	25.5	46.5	54.6	39.1	19.3	42.5	57.1
	Swin-T	*Baseline*	81.18	42.7	26.5	45.9	56.6	39.3	20.5	41.8	57.8
		Ours	83.80 (+2.62)	44.4	28.7	47.9	58.6	40.8	21.8	43.7	59.8

Comparisons with Model Designs. To illustrate the value of our method to model performance, we examine whether our ScaleKD could bring comparable performance gains as network architecture engineering. We apply ScaleKD to the basic design of each architecture and make comparisons with recent advanced designs. From the results shown in Table 5, we observe that: i) The performance gains from our ScaleKD could help the basic designs reach similar or even higher performance than advanced models. ii) For some basic models that suffer from the data-hungry problem (Mixer-B) or low convergence speed (ViT-S), our ScaleKD could even help them exceed larger advanced models.

Comparisons with Pre-training Methods. We finally examine the value of our method to the training regime. We compare the performance between models trained by ScaleKD and models pre-trained by various methods on upstream datasets, to figure out whether our ScaleKD could be an efficient substitution to pre-training of any target student on large-scale datasets if we have a strong pre-trained ViT. For making comprehensive comparisons, we select the methods across various pre-training paradigms, such as supervised pre-training, self-supervised pre-training, cross-modal pre-training, and hybrid pre-training. From the results shown in Table 6, we can summarize: i) Our ScaleKD has superior performance than various methods across all kinds of pre-training; ii) Our ScaleKD is a more efficient approach, viewing less than 5.58$\times$|11.75$\times$|195.39$\times$|8.73$\times$ samples than these counterpart methods based on different pre-training paradigms.

3.3 Experiments on Downstream Tasks

To further examine whether the performance gains from our method could be well preserved in transfer learning, we conduct comparative experiments on MS-COCO for object detection and instance segmentation, and on ADE20K for semantic segmentation.

Table 8: Transfer learning performance on ADE20K

Framework	Backbone	Pre-training	IN-1K (Top-1)	ADE20K (mIOU)
UperNet	ResNet-50	*Baseline*	78.64	42.37
		Ours	82.03 (+3.39)	44.50 (+2.13)
	Swin-T	*Baseline*	81.18	44.41
		Ours	83.80 (+2.62)	46.33 (+1.92)
	ViT-B/16	*Baseline*	81.80	46.75
		Ours	85.53 (+3.73)	50.84 (+4.09)

Experimental Results. The results on MS-COCO and ADE20K are shown in Table 7 and Table 8, respectively, from which we can observe that: i) Overall, our pre-trained models outperform baselines by significant margins across three tasks and different architectures, demonstrating the performance gains from ScaleKD could be transferred to downstream tasks; ii) For semantic segmentation tasks,

Table 9: Ablation studies. Experiments in (b)-(d) are performed on Swin-S→ResNet-50. As DFM and TPP are designed based on CAP, CAP is added by default when choosing DFM and TPP in (a). Because of this, we treat CAP as another baseline method, when analyzing DFM and TPP in (c)-(d).

(a) Ablation on the overall design

Teacher	Student	Ablation Design CAP	DFM	TPP	KD	Top-1
Swin-S	ResNet-50					76.55
		✓				77.87
		✓	✓			78.51
		✓		✓		78.62
		✓	✓	✓		79.30
		✓	✓	✓	✓	79.62
	Mixer-S					74.02
		✓				75.03
		✓	✓			76.42
		✓		✓		76.28
		✓	✓	✓		77.24
		✓	✓	✓	✓	77.59

(b) Ablation on CAP

Projector	Top-1	Δ Top-1
Baseline	76.55	-
Linear	77.43	+0.88
Conv	77.52	+0.97
CAP	77.87	+1.32

(c) Ablation on DFM

Feature Mimicking	Top-1	ΔTop-1
Baseline	76.55	-
CAP	77.87	+1.32
DFM w/o filtering	78.12	+1.57
DFM w/ filtering	78.51	+1.96

(d) Ablation on TPP

Method	TPP Design Teacher's Param	Provide Queries	Accuracy (%) Top-1	ΔTop-1
Baseline	-	-	76.55	-
CAP	-	-	77.87	+1.32
TPP	✓	-	78.50	+1.95
TPP	✓	✓	78.62	+2.07

ViT-B/16 achieves the highest 4.09% absolute performance gain across three backbones, even higher than its gain on IN-1K; iii) For object detection and instance segmentation on MS-COCO, ResNet-50|Swin-T pre-trained by ScaleKD outperforms baseline on *AP* by a margin of 2.1%|1.7% and 2.0%|1.5%.

3.4 Ablation Study

Ablation Study on the Overall Design. To exploit the gains from different designs and further examine whether our three components are complementary to each other, we perform this study. From the results shown in Table 9a, we notice: i) When gradually applying more of the designs, the performance of ResNet-50 and Mixer-S show similar increasing trends, showing that each component of ScaleKD is not designed for specific student architecture; ii) Although CAP can bring the two students promising performance gains, DFM and TPP can further bring ResNet-50|Mixer-S extra performance improvements, 0.64%|0.75% and 1.25%|1.39%, respectively. This observation verifies that DFM and TPP are complementary to CAP; iii) When using DFM and TPP together, both ResNet-50 and Mixer-S can obtain additional performance boosts, which indicates that DFM and TPP are complementary with each other.

Ablation Study on CAP, DFM and TPP. i) We compare CAP with two popular projectors, *Linear-projector* and *Conv-projector*, to verify the superiority of CAP. The former projector consists of a linear layer and the latter projector consists of two 3×3 convolutional layers with ReLU. From the results shown in Table 9b, we can notice that our CAP outperforms them, which partially validates that our CAP can boost the distillation performance by aligning the computing paradigm in form. ii) We then investigate the effectiveness of two designs in DFM. From the results shown in Table 9c, we find that plain dual-path feature mimicking brings an extra 0.25% performance gain to CAP, and direct component filtering further brings 0.39% performance improvement based on dual-path feature mimicking. This experiment validates that the two designs are essential and complementary to each other. iii) To explore the necessities of teacher parameter reusing and input-dependent queries in TPP, we conduct the experiment in Table 9d. The results show that reusing teacher parameters can provide students with performance improvement and providing input-dependent queries can further enhance the effectiveness of TPP.

More ablation studies on hyper-parameters and the training efficiency are provided in Appendix D.

3.5 Verification Experiments and Discussion

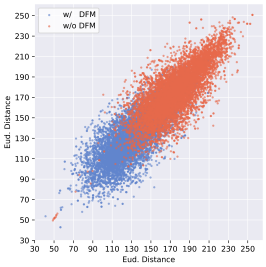
Finally, we conduct experiments to study: i) Could ScaleKD help the student mimic the teacher's high-frequency features? ii) Could ScaleKD help the student learn the teacher's pre-training knowledge?

High-Frequency Feature Mimicking. To clearly illustrate that our ScaleKD could help the student capture the teacher's high-frequency features, we make comparisons between the method with DFM and without DFM. Specifically, we measure the distances between the student's features and the

teacher's features of alternative components in the spatial domain, as shown in Figure 3. Clearly, methods with DFM can reduce high-frequency feature distances, answering the first question.

Pre-training Knowledge Learning. The pre-training knowledge of teachers can be generally summarized into two parts: the generalization ability within training data distribution and the generalization ability across different data distributions. In Table 6, we have verified the capability of ScaleKD under the former setting. To further examine the generalization across different distributions, we conduct several linear probing experiments on CIFAR-100, based on models in Table 6. From the results shown in Table 12 in the Appendix, we can observe that: i) models pre-trained by CLIP greatly improve the backbone's generalization ability across datasets; ii) our ScaleKD helps the student model reach similar performance as CLIP-based pre-training, without viewing pre-training data. The two observations prove the gain from ScaleKD in this experiment is from the knowledge of upstream datasets, thus answering the second question.

Figure 3: Feature distances of alternative components in the spatial domain. Details on the figure drawing are in Figure 6



4 Related Work

Knowledge Distillation. Traditional Knowledge distillation methods [14–35] generally focus on CNN-based teacher-student network pairs with small model scale gaps. Some recent works [52, 53, 47] further study how to conduct knowledge distillation on larger teachers. As vision transformers suffer from low convergence speeds, some recent works [54–56] explore leveraging CNNs to accelerate the training of vision transformers. Meanwhile, [36–38] discuss how to bridge the architecture gap when the teacher and the student are in different model categories.

Frequency-based Knowledge Distillation. As traditional feature distillation only focus on pixel-to-pixel differences, FAM [57] designs frequency-based attention and conducts KD between attention maps. FreeKD [58] explores how to eliminate unfavorable information in the frequency domain to enhance the distillation performance on dense prediction tasks. Different from our ScaleKD, they consider feature distillation on CNN-based network pairs and have different formulations.

Teacher Parameter Reuse. Some previous KD methods also leverage the teacher's parameter for reusing a better classifier [32] and initializing the student's neck and head [59, 60] or dismissing the shortcuts in residual architectures [61]. Unlike our ScaleKD, the motivation of these works focuses on parameter reuse or equivalent substitution, rather than aligning the two parameter spaces for pursuing the teacher's pre-training knowledge.

5 Conclusion

In this paper, we present ScaleKD, a new cross architecture KD approach for transferring the scalable properties from pre-trained vision transformers to various CNNs and MLPs or heterogeneous ViTs. Our method presents principled designs to address the difficulties of feature computing paradigm differences, model scale differences, and knowledge density differences between teacher and student. By conducting systematic experiments on various large mainstream benchmarks, we broadly validate the effectiveness and generalization ability of our method.

Limitations and Broader Impacts. Restricted by our computational resources, we do not conduct experiments on very large teachers, such as ViT-22B [62], or on large students, such as ViT-L [4]. Furthermore, with the increasing model scale of teachers, the training costs of ScaleKD increase, which is a common limitation to KD research. According to the analysis in Appendix D, the extra training costs of ScaleKD is acceptable in most cases, because its performance gain shows the potential to replace time-intensive pre-training of students on large-scale datasets, thus being significantly more efficient. For broader impacts, we think our research could be beneficial to many real applications by promoting their model performance and training efficiency, which could also be a reference for the research on KD with large vision models.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[3] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, 2022.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020.

[5] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

[6] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. In *NeurIPS*, 2021.

[7] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. Resmlp: Feedforward networks for image classification with data-efficient training. *TPAMI*, 2022.

[8] Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. Pay attention to mlps. In *NeurIPS*, 2021.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

[10] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[11] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. Image bert pre-training with online tokenizer. In *ICLR*, 2021.

[12] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[13] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *CVPR*, 2023.

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

[15] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

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

[17] Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. In *CVPR*, 2019.

[18] Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *ICCV*, 2019.

[19] Baoyun Peng, Xiao Jin, Jiaheng Liu, Dongsheng Li, Yichao Wu, Yu Liu, Shunfeng Zhou, and Zhaoning Zhang. Correlation congruence for knowledge distillation. In *ICCV*, 2019.

[20] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *CVPR*, 2019.

[21] Nikolaos Passalis and Anastasios Tefas. Learning deep representations with probabilistic knowledge transfer. In *ECCV*, 2018.

[22] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *AAAI*, 2019.

[23] Jangho Kim, SeongUk Park, and Nojun Kwak. Paraphrasing complex network: Network compression via factor transfer. In *NeurIPS*, 2018.

[24] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *CVPR*, 2017.

[25] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *ICLR*, 2020.

[26] Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation. In *ICCV*, 2019.

[27] Guodong Xu, Ziwei Liu, Xiaoxiao Li, and Chen Change Loy. Knowledge distillation meets self-supervision. In *ECCV*, 2020.

[28] Xiatian Zhu, Shaogang Gong, et al. Knowledge distillation by on-the-fly native ensemble. In *NeurIPS*, 2018.

[29] Guile Wu and Shaogang Gong. Peer collaborative learning for online knowledge distillation. In *AAAI*, 2021.

[30] Jing Yang, Brais Marinez, Bulat Adrian, and Tzimiropoulos Georgios. Knowledge distillation via softmax regression representation learning. In *ICLR*, 2021.

[31] Defang Chen, Jian-Ping Mei, Yuan Zhang, Can Wang, Zhe Wang, Yan Feng, and Chun Chen. Cross-layer distillation with semantic calibration. In *AAAI*, 2021.

[32] Defang Chen, Jian-Ping Mei, Hailin Zhang, Can Wang, Yan Feng, and Chun Chen. Knowledge distillation with the reused teacher classifier. In *CVPR*, 2022.

[33] Xueqing Deng, Dawei Sun, Shawn Newsam, and Peng Wang. Distpro: Searching a fast knowledge distillation process via meta optimization. In *ECCV*, 2022.

[34] Xiaolong Liu, LUJUN LI, Chao Li, and Anbang Yao. Norm: Knowledge distillation via n-to-one representation matching. In *ICLR*, 2022.

[35] Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In *CVPR*, 2022.

[36] Zhiwei Hao, Jianyuan Guo, Kai Han, Yehui Tang, Han Hu, Yunhe Wang, and Chang Xu. One-for-all: Bridge the gap between heterogeneous architectures in knowledge distillation. In *NeurIPS*, 2023.

[37] Yufan Liu, Jiajiong Cao, Bing Li, Weiming Hu, Jingting Ding, and Liang Li. Cross-architecture knowledge distillation. In *ACCV*, 2022.

[38] Yixuan Wei, Han Hu, Zhenda Xie, Zheng Zhang, Yue Cao, Jianmin Bao, Dong Chen, and Baining Guo. Contrastive learning rivals masked image modeling in fine-tuning via feature distillation. *arXiv preprint arXiv:2205.14141*, 2022.

[39] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. In *ICLR*, 2021.

[40] Yuxin Fang, Wen Wang, Binhui Xie, Quan Sun, Ledell Wu, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva: Exploring the limits of masked visual representation learning at scale. In *CVPR*, 2023.

[41] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023.

[42] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *CVPR*, 2023.

[43] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[44] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[45] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[46] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *IJCV*, 2019.

[47] Tao Huang, Shan You, Fei Wang, Chen Qian, and Chang Xu. Knowledge distillation from a stronger teacher. In *NeurIPS*, 2022.

[48] Tao Huang, Yuan Zhang, Mingkai Zheng, Shan You, Fei Wang, Chen Qian, and Chang Xu. Knowledge diffusion for distillation. In *NeurIPS*, 2023.

[49] Lucas Beyer, Xiaohua Zhai, Amélie Royer, Larisa Markeeva, Rohan Anil, and Alexander Kolesnikov. Knowledge distillation: A good teacher is patient and consistent. In *CVPR*, 2022.

[50] Ao Wang, Hui Chen, Zijia Lin, Hengjun Pu, and Guiguang Ding. Repvit: Revisiting mobile cnn from vit perspective. *arXiv preprint arXiv:2307.09283*, 2023.

[51] Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva-02: A visual representation for neon genesis. *arXiv preprint arXiv:2303.11331*, 2023.

[52] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *AAAI*, 2020.

[53] Wonchul Son, Jaemin Na, Junyong Choi, and Wonjun Hwang. Densely guided knowledge distillation using multiple teacher assistants. In *ICCV*, 2021.

[54] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, 2021.

[55] Xianing Chen, Qiong Cao, Yujie Zhong, Jing Zhang, Shenghua Gao, and Dacheng Tao. Dearkd: Data-efficient early knowledge distillation for vision transformers. In *CVPR*, 2022.

[56] Borui Zhao, Renjie Song, and Jiajun Liang. Cumulative spatial knowledge distillation for vision transformers. In *ICCV*, 2023.

[57] Cuong Pham, Van-Anh Nguyen, Trung Le, Dinh Phung, Gustavo Carneiro, and Thanh-Toan Do. Frequency attention for knowledge distillation. In *WACV*, 2024.

[58] Yuan Zhang, Tao Huang, Jiaming Liu, Tao Jiang, Kuan Cheng, and Shanghang Zhang. Freekd: Knowledge distillation via semantic frequency prompt. In *CVPR*, 2024.

[59] Zijian Kang, Peizhen Zhang, Xiangyu Zhang, Jian Sun, and Nanning Zheng. Instance-conditional knowledge distillation for object detection. In *NeurIPS*, 2021.

[60] Zhendong Yang, Zhe Li, Xiaohu Jiang, Yuan Gong, Zehuan Yuan, Danpei Zhao, and Chun Yuan. Focal and global knowledge distillation for detectors. In *CVPR*, 2022.

[61] Guilin Li, Junlei Zhang, Yunhe Wang, Chuanjian Liu, Matthias Tan, Yunfeng Lin, Wei Zhang, Jiashi Feng, and Tong Zhang. Residual distillation: Towards portable deep neural networks without shortcuts. In *NeurIPS*, 2020.

[62] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *ICML*, 2023.

[63] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Tech Report*, 2009.

[64] MMPreTrain Contributors. Openmmlab's pre-training toolbox and benchmark. `https://github.com/open-mmlab/mmpretrain`, 2023.

[65] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[66] Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Distilling knowledge via knowledge review. In *CVPR*, 2021.

[67] Zhendong Yang, Zhe Li, Mingqi Shao, Dachuan Shi, Zehuan Yuan, and Chun Yuan. Masked generative distillation. In *ECCV*, 2022.

[68] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[69] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. `https://github.com/open-mmlab/mmsegmentation`, 2020.

[70] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[71] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018.

[72] Xiaohan Ding, Yiyuan Zhang, Yixiao Ge, Sijie Zhao, Lin Song, Xiangyu Yue, and Ying Shan. Unireplknet: A universal perception large-kernel convnet for audio, video, point cloud, time-series and image recognition. *arXiv preprint arXiv:2311.15599*, 2023.

[73] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

[74] Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Tinyvit: Fast pretraining distillation for small vision transformers. In *ECCV*, 2022.

[75] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *ICCV*, 2021.

[76] Mingyu Ding, Bin Xiao, Noel Codella, Ping Luo, Jingdong Wang, and Lu Yuan. Davit: Dual attention vision transformers. In *ECCV*, 2022.

[77] Jiahao Wang, Wenqi Shao, Mengzhao Chen, Chengyue Wu, Yong Liu, Kaipeng Zhang, Songyang Zhang, Kai Chen, and Ping Luo. Adapting llama decoder to vision transformer. *arXiv preprint arXiv:2404.06773*, 2024.

Appendix for "ScaleKD: Strong Vision Transformers Could Be Excellent Teachers"

Anonymous Author(s)
Affiliation
Address
email

A Dataset

ImageNet-1K [44] is a well-known large-scale classification dataset, comprising over 1.2 million training images and 50,000 validation images with 1,000 object categories.

MS-COCO [45] is a large-scale dataset for object detection and instance segmentation, which contains 118,000 training images and 5,000 validation images with 80 object categories.

ADE20K [46] is a challenging semantic segmentation dataset, containing 20,210 training samples, 2,000 validation samples, and 3,352 testing samples with 150 categories.

CIFAR-100 [63] is a popular classification dataset, which contains 50,000 training images with 500 images per class and 10,000 test images.

B Experimental Setups

B.1 Experimental Setups on IN-1K

Training Strategy. We conduct our experiments with two training strategies: traditional training strategy and advanced training strategy. The traditional training strategy is commonly used in previous KD approaches (shown in Table 10a) and the advanced training strategy is adopted in training recently proposed CNNs, MLPs, and ViTs (shown in Table 10b).

Hardware. The experiments using the traditional training strategy are conducted on 8 $\times$ NVIDIA Tesla-V100 GPUs, while the experiments using the advanced training strategy are conducted on 32 $\times$ NVIDIA Tesla-V100 GPUs.

Implementation Codebase. We implement our method based on MMClassification [64].

Hyper-parameter Settings. The overall loss for our ScaleKD is defined as Formulation 5. Thanks to the simplicity of our design, we have only one hyper-parameter β in our ScaleKD. From the ablation study in the Appendix D, we find that the best choice is $\beta = 0.6$ and we use it as the default setting throughout all experiments.

Selection of Teacher-Student Network Pairs. Overall, we choose eleven teacher-student network pairs, which consist of two strong ViTs and ten students, covering mainstream architectures of ViT, MLP, and CNN. Specifically, for the teacher, we choose two different types of strong ViTs: supervised pre-trained Swin-L [5] with the hierarchical architecture and hybrid pre-trained BEiT-L [39] with the typical ViT architecture. Moreover, compared to Swin-L, BEiT-L is much larger in terms of model size and stronger in terms of model performance. For the student selections, we first choose the basic design in each architecture, such as ResNet-50 [2], Mixer-S/16 [6], and ViT-S/16 [4]. Then, we also select some popular models, such as MobileNet-V1 [65], ConvNeXt-T [3], and Swin-S. Next, we expand the basic designs to larger ones, like Mixer-B/16, Mixer-B/14, ViT-B/16 and ViT-B/14. After

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

Table 10: Detailed settings of traditional training strategy and advanced training strategy on IN-1K.

(a) Traditional training strategy

Configuration	CNN
Batch Size	256
Learning Rate	0.1
Learning Rate Schedule	Stepwise Decay/Cosine Decay
Optimizer	SGD
Optimizer Hyper-Parameters	momentum= 0.9
Weight Decay	1e-4
Training Epochs	100
Warmup Epochs	✗
Drop Path	✗
Label Smoothing	✗
Random Flip	0.5
Random Resize Crop	(0.08,1)
Random Augmentation	✗
Random Erasing	✗

(b) Advanced training strategy

Config	CNN / MLP / ViT
Batch Size	2048 / 1536 / 1024
Learning Rate	5e-3 / 7e-4 / 1e-3
Learning Rate Schedule	Cosine Decay
Optimizer	Lamb / AdamW / AdamW
Optimizer Hyper-Parameters	$\beta_1, \beta_2, \epsilon = 0.9, 0.009$, 1e-8
Weight Decay	0.02 / 0.07 /0.05
Training Epochs	300
Warmup Epochs	5 / 20 / 20
Drop Path	0.05 / 0.1 / 0.1
Label Smoothing	0.1
Random Flip	0.5
Random Resize Crop	(0.08,1)
Random Augmentation	(7,0.5) / (9,0.5) / (9,0.5)
Random Erasing	0.25

Table 11: Detailed settings of transfer learning strategies on MS-COCO and ADE20K.

(a) MS-COCO

Configuration	ResNet-50 / Swin-S
Weight Initialization	Pre-trained Checkpoint
Batch Size	16
Learning Rate	1e-4
Learning Rate Decay	Stage (0.7)
Learning Rate Schedule	Cosine Decay
Optimizer	AdamW
Optimizer Hyper-Parameters	$\beta_1, \beta_2, \epsilon = 0.9, 0.009$, 1e-8
Weight Decay	0.05
Training Epochs	8
Crop Size	(1333, 800)
Drop Path	0.0 / 0.2

(b) ADE20K

Configuration	ResNet-50 / Swin-S / ViT-B
Weight Initialization	Pre-trained Checkpoint
Batch Size	16
Learning Rate Decay	1e-4 / 1e-4 / 2e-4
LR decay	Stage (0.9) / Stage (0.9) / Layer (0.6)
Learning Rate Schedule	Cosine Decay
Optimizer	AdamW
Optimizer Hyper-Parameters	$\beta_1, \beta_2, \epsilon = 0.9, 0.009$, 1e-8
Weight Decay	0.05
Training Iterations	160000
Crop Size	(512, 512)
Drop Path	0.0 / 0.3 / 0.2

separately selecting the teachers and students, we finally organize them into eleven teacher-student network pairs for experiments.

Counterpart Knowledge Distillation Methods. In the main paper, we make comparisons with many recent KD methods, such as DIST [47], DiffKD [48], OFA [36] and FuncMatch [49]. In the appendix, we further compare with CNN-based methods, such as KD [14], AT [16], OFD [26], RKD [20], CRD [25], DKD [35], SRRL [30], ReviewKD [66], DistPro [33] and MGD [67].

Counterpart Model Designs. To explicitly evaluate the significance of the performance gains, we apply ScaleKD on the basic designs of each architecture and make comparisons with various advanced counterparts. Driven by this target, we mainly select the next-generation model based on the selected designs. For ResNet-50, we select ConvNeXt-T [3] and RepViT-2.3M [50] for making comparisons. The former one is the typical design of the new-era CNN and the latter one is a popular model for deployment. For Mixer-B, we select gMLP-B [8] and ResMLP-B24 [7], which are optimized based on the weaknesses of MLP-Mixer. For ViT-S, we choose Swin-S [5] and Swin-B as compared counterparts, to validate whether our ScaleKD could outperform larger advanced designs.

Counterpart Pre-training Methods. We select state-of-the-art methods in each paradigm. For supervised pre-training, we directly compare with the results in the original paper [4]. For self-supervised pre-training, we choose BEiT [39] and iBoT [11]. For cross-modal pre-training and hybrid pre-training, we choose CLIP [13] and EVA-02 [51], respectively.

Table 12: Performance comparisons of ViT-B/16 between ScaleKD and CLIP for linear probing.

Model	Method	Pre-training Dataset	IN-1K (Training)	CIFAR-100 (Linear Probing) 1 shot	5 shot	10 shot	25 shot	Full
ViT-B/16	From-scratch	IN-1K	81.80	33.86	60.30	66.77	72.65	81.76
	CLIP	LAION-300M	-	-	-	71.96	77.21	84.07
		LAION-2B, IN-1K	85.49	41.10	69.00	72.34	78.64	85.51
		LAION-2B, IN-12K, IN-1K	86.17	44.90	70.19	76.77	81.43	88.88
		CLIP OpenAI, IN-12K, IN-1K	85.99	47.40	70.85	77.37	**81.52**	88.92
	Ours	IN-1K	86.19	**48.14**	**70.91**	**77.52**	81.50	**89.11**
Teacher: BEiT-L/14	EVA	CLIP OpenAI, IN-22K, IN-1K	88.58	63.74	85.20	87.39	89.27	93.36

Table 13: Experiments on the training efficiency of ScaleKD. The student model in all experiments is ResNet-50. In (a), we compare our ScaleKD with traditional FD on three teachers with different model scales. In (b), we conduct the experiments based on Swin-S→ResNet-50 teacher-student network pair to illustrate the training costs (memory and time) introduced by each component. Experiments are conducted on 8 × NVIDIA Tesla-V100 GPUs

(a) Training costs comparisons with FD

Teacher	Method	Top-1 (%)	GPU Memory (G)	T_{train} (d)
Swin-S	FD	77.43	3.66	1.67
	ScaleKD	79.62	6.65	2.10
Swin-B	FD	77.76	3.66	1.83
	ScaleKD	79.80	7.26	2.53
Swin-L	FD	77.72	4.72	2.24
	ScaleKD	80.10	9.11	3.51

(b) Training costs of each design in ScaleKD

Method	Designs CAP	DFM	TPP	KD	Top-1 (%)	GPU Memory (G)	T_{train} (d)
FD	-	-	-	-	77.43	3.66	1.67
ScaleKD	✓				77.87	3.77	1.70
	✓	✓			78.51	4.02	1.77
	✓		✓		78.62	5.13	1.84
	✓	✓	✓		79.30	6.60	2.08
	✓	✓	✓	✓	79.62	6.65	2.10

B.2 Experimental Setups on MS-COCO and ADE20K

Training Strategy and Hyper-parameter Settings. For the experiments on MS-COCO, we adopt the settings shown in Table 11a, while for experiments on ADE20K, we adopt the settings shown in Table 11b.

Hardware. All experiments on MS-COCO and ADE20K are conducted on 8 × NVIDIA Tesla-V100 GPUs.

Implementation Codebase. We conduct experiments based on MMDetection [68] and MMSegmentation [69].

Framework and Backbone Selections. For task frameworks, we choose Mask R-CNN [70] for object detection and instance segmentation, and UperNet [71] for semantic segmentation. As for backbones, we select ResNet-50, Swin-T, and ViT-B/16.

C More Experimental Results

Performance Comparisons with More KD Methods. In our main paper, we only compare our ScaleKD with some related cross architecture KD approaches in Table 4, as few previous works conduct experiments based on medium-sized students, such as ResNet-50. To make comparisons with more CNN-based KD methods, we conduct experiments on a traditional student network, MobileNet-V1, using the same training strategy as them. From the results shown in Table 14, we can observe that by utilizing Swin-L as the teacher, our ScaleKD could help MobileNet-V1 reach 74.21% top-1 accuracy, outperforming previous methods which utilize ResNet-50 as the teacher by clear margins.

Performance Comparisons with More Model Designs. In our main paper, we apply our ScaleKD to the basic design of each architecture and make comparisons with more recent variant architectures. As illustrated in Table 15, we choose more designs to make more comprehensive comparisons.

D More Ablation Studies

Ablation on Training Efficiency of ScaleKD. As the TPP in our ScaleKD leverages the teacher's last stage, it will introduce additional training costs. To clearly study its training efficiency, we conduct

Table 14: Top-1 accuracy (%) comparison on IN-1K with more CNN-based KD methods. In the experiment, we adopt the same traditional training strategy as these methods.

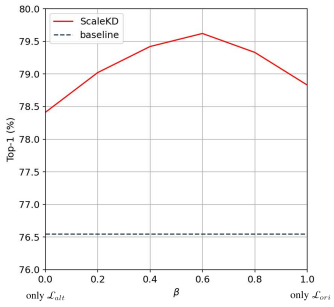
Model	Teacher	Method	Top-1(%)
MobileNet-V1	ResNet-50 (76.16)	From Scratch	69.63
		KD [14]	70.68
		AT [16]	70.72
		OFD [26]	71.25
		RKD [20]	71.23
		CRD [25]	71.40
		DKD [35]	72.05
		SRRL [30]	72.49
		ReviewKD [66]	72.56
		DIST [47]	73.24
		DistPro [33]	73.26
		MGD [67]	73.35
		DiffKD [48]	73.62
	Swin-L (86.24)	ScaleKD	**74.21**

Table 15: Performance comparisons on IN-1K with various models. We conduct ScaleKD on the simplest design of each architecture and then make performance comparisons with various designs. Ours*, Ours†, Ours‡ denote choosing ViT-B (training from scratch), Swin-L (with IN-22K pre-training) and BEiT-L (with EVA pre-training) as the teacher, respectively.

Model	Training Dataset	Sample Resolution	Params (M)	FLOPs (G)	Top-1(%)
CNN-based Architecture					
ConvNext-T [3]	IN-1K	224^2	28.59	4.46	82.14
ConvNext-T + Ours†	IN-1K	224^2	28.59	4.46	84.16
ConvNext-T [3]	IN-22K $\Rightarrow$ IN-1K	224^2	28.59	4.46	82.90
ConvNext-B [3]	IN-1K	224^2	87.77	15.14	83.80
UniRepLKNet-T [72]	IN-1K	224^2	31.00	4.90	83.20
EfficientNet-B5 [73]	IN-1K	456^2	30.00	9.90	83.60
RepViT-M2.3 [50]	IN-1K	224^2	22.90	-	83.70
MLP-based Architecture					
Mixer-B/16 [6]	IN-1K	224^2	59.88	12.61	76.44
Mixer-B/16 + Ours*	IN-1K	224^2	59.88	12.61	81.62
Mixer-B/16 + Ours†	IN-1K	224^2	59.88	12.61	81.96
Mixer-B/14 + Ours‡	IN-1K	224^2	59.88	16.45	82.89
Mixer-B/16 [6]	IN-22K $\Rightarrow$ IN-1K	224^2	59.88	12.61	80.64
Mixer-L/16 [6]	IN-22K $\Rightarrow$ IN-1K	224^2	208.2	44.57	82.89
ResMLP-B24 [7]	IN-1K	224^2	115.7	23.0	81.00
gMLP-B [8]	IN-1K	224^2	73.00	15.80	81.60
Transformer-based Architecture					
ViT-S/16 [4]	IN-1K	224^2	22.05	4.61	79.90
ViT-S/16 + Ours†	IN-1K	224^2	22.05	4.61	83.93
ViT-S/16 [4, 74]	IN-22K $\Rightarrow$ IN-1K	224^2	22.05	4.61	80.50
Swin-T [5, 74]	IN-22K $\Rightarrow$ IN-1K	224^2	28.29	4.36	81.90
T2T-ViT$_t$-14 [75]	IN-1K	224^2	21.47	4.34	81.83
DaViT-T [76]	IN-1K	224^2	28.36	4.54	82.24
iLLaMA-S [77]	IN-1K	224^2	21.90	-	79.90
EVA-02-S/14 [51]	IN-1K	336^2	22.13	15.51	81.12
EVA-02-S/14 + Ours‡	IN-1K	336^2	22.13	15.51	86.22
ViT-B/16 [4]	IN-1K	224^2	86.57	17.58	81.80
Swin-B [5]	IN-1K	224^2	87.77	15.14	83.50
T2T-ViT$_t$-24 [75]	IN-1K	224^2	64.00	12.69	82.71
DaViT-B [76]	IN-1K	224^2	87.95	15.51	84.09
ViT-B/16 [4]	IN-22K $\Rightarrow$ IN-1K	224^2	86.57	17.58	83.97
Swin-B [5]	IN-22K $\Rightarrow$ IN-1K	224^2	87.77	15.14	85.20
iLLaMA-B [77]	IN-22K $\Rightarrow$ IN-1K	224^2	86.30	-	85.00

ablative experiments in this section: i) As shown in Table 13a, we first compare the training efficiency of ScaleKD with traditional FD on three network pairs with increasing teacher's model scales; ii) Then, as shown in Table 13b, we conduct the experiments on each design in ScaleKD. The experimental results show that: i) Using large teachers would induce more GPU memory occupation and longer training time. ii) Comparatively, TPP is the most resource-consuming component, especially after combining it with DFM. In summary, our ScaleKD needs additional training resources compared to traditional FD. However, if considering the significant performance gain it brings, these additional costs are acceptable in most cases.

Figure 4: Ablation Study on the hyper-parameter β.



Ablation Study on Hyper-parameter β. We conduct the ablation study on Swin-S→ResNet-50 network pair. According to Formulation 5 in the main paper, our method only has one hyper-parameter β, which is the balancing weight of the two views in DFM. To be specific, $\beta = 1.0$ indicates that only the first feature mimicking path exists, while $\beta = 0$ indicates that only the second feature mimicking path exists. And we select the β uniformly from 0 to 1 to determine the optimal β in ScaleKD. As shown in Figure 4, we can observe: i) In general, ScaleKD outperforms the baseline by significant margins at each setting, validating the efficacy of ScaleKD; ii) When $\beta = 0.6$, our ScaleKD achieves the best performance; iii) Though the second feature mimicking path could be used individually, it is inferior to the first path, indicating that the direct component is essential in feature mimicking; iv) When the two paths work collaboratively, they perform better than individually applied, which suggests that the two views are complementary with each other.

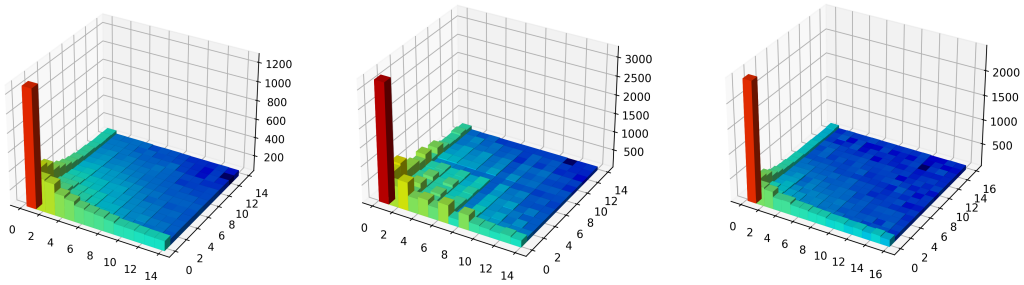
Ablation Study on Pre-training and Distillation. In this study, we explore the originality of the gains. We compare models trained by ScaleKD with upstream pre-trained models and upstream pre-training models with KD. From the results shown in Table 16, we notice: i) Compared to individual pre-training, applying KD under this stage can significantly boost model performance; ii) ViT-S/16 trained by ScaleKD significantly outperforms the models trained with KD on IN-22K. The two observations illustrate that: i) Small students are difficult to capture pre-training knowledge; ii) Our ScaleKD could effectively help the student to learn useful pre-training knowledge from the teacher without viewing the pre-training dataset.

Table 16: Ablation study on pre-training and distillation.

Model	Method	Top-1(%)
ViT-S/16	Training from scratch on IN-1K	79.90
	Training from scratch on IN-1K w/ KD	81.42
	Pre-training on IN-22K	80.05
	Pre-training on IN-22K w/ KD	82.00
	Training from scratch on IN-1K w/ ScaleKD	83.93

E More Visualization Results

In this section, we provide more visualization results. In Figure 5, we provide the frequency distributions of three large pre-trained ViT models. We can observe the consistency in pre-trained ViTs' unbalanced frequency distributions, that the direct responses are salient and obviously stronger than any other responses.



(a) ViT-L (b) Swin-L (c) BEiT-L

Figure 5: More illustrative feature distributions of large pre-trained ViTs in the frequency domain. We first collect the last stage output feature maps of 1600 samples from IN-1K, then conduct DCT on each channel, and finally take the average value across these samples, after turning all responses into absolute values.

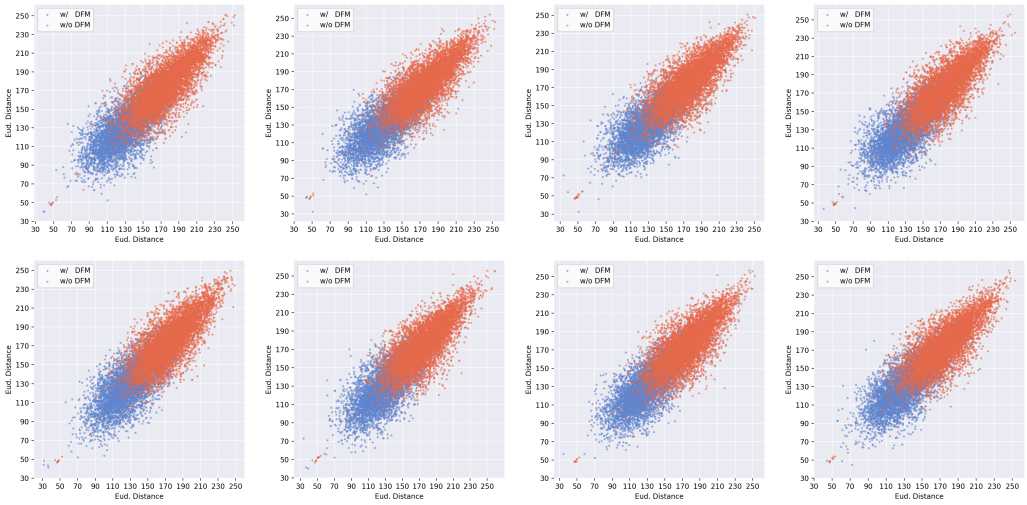


Figure 6: Feature distance distribution of alternative components for the last stage features between teacher and student on IN-1K. We obtain 64,000 feature pairs from 64,000 samples. After calculating the distance between teacher and student, we project the high-dimension distances into two-dimension space for illustration. Finally, we randomly select 6,400 data points for 8 times to draw the scatters. Blue points denote the distances without DFM, while orange points denote the distances with DFM.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction contain three parts: i) motivation, ii) methodology, and iii) experimental results, which are explained in Section 1, Section 2, and Section 3. We discuss the related works in Section 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in the Conclusion part in our main paper

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper does not include theoretical results

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide all training setups and implementation details in Appendix B

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The code will be made publicly available. While it is currently under internal review process, we decide not to release the code at this moment.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We detailed explain all experimental settings in Appendix B

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The main experiments are conducted on large-scale datasets, whose results are stable.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We give the demand compute resources in Appendix B

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the broader impacts in the Conclusion Part of the main paper.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We clearly illustrate the dataset and codebase we use in the Appendix A and the Appendix B, respectively.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We obtained some high-performance models in this paper.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.