CustomKD: Customizing Large Vision Foundation for Edge Model Improvement via Knowledge Distillation

Jungsoo Lee Debasmit Das Munawar Hayat
Sungha Choi[†] Kyuwoong Hwang Fatih Porikli
Qualcomm AI Research[‡]
{jungsool, debadas, hayat, sunghac, kyuwoong, fporikli}@qti.qualcomm.com

Abstract

*We propose a novel knowledge distillation approach, CustomKD, that effectively leverages large vision foundation models (LVFMs) to enhance the performance of edge models (*e.g.*, MobileNetV3). Despite recent advancements in LVFMs, such as DINOv2 and CLIP, their potential in knowledge distillation for enhancing edge models remains underexplored. While knowledge distillation is a promising approach for improving the performance of edge models, the discrepancy in model capacities and heterogeneous architectures between LVFMs and edge models poses a significant challenge. Our observation indicates that although utilizing larger backbones (*e.g.*, ViT-S to ViT-L) in teacher models improves their downstream task performances, the knowledge distillation from the large teacher models fails to bring as much performance gain for student models as for teacher models due to the large model discrepancy. Our simple yet effective CustomKD customizes the well-generalized features inherent in LVFMs to a given student model in order to reduce model discrepancies. Specifically, beyond providing well-generalized original knowledge from teachers, CustomKD aligns the features of teachers to those of students, making it easy for students to understand and overcome the large model discrepancy overall. CustomKD significantly improves the performances of edge models in scenarios with unlabeled data such as unsupervised domain adaptation (*e.g.*, OfficeHome and DomainNet) and semi-supervised learning (*e.g.*, CIFAR-100 and ImageNet), achieving the new state-of-the-art performances.*

1. Introduction

Recent efforts in computer vision have focused on building large vision foundation models (LVFMs), with a substantial number of parameters, generally trained using large-scale pretraining datasets. Owing to their well-generalized representation, LVFMs are widely known to achieve state-of-the-art performances on diverse downstream tasks [20, 23, 31, 37, 42]. However, while LVFMs have demonstrated impressive performance, their utilization in resource-constrained real-world applications is challenging due to their high computational costs and extensive parameters, which often impede their deployment [12, 52, 55]. Consequently, edge models, known for their computational efficiency, may remain the preferred choice for real-world applications, especially when considering deployment on mobile devices. However, these edge models typically offer limited performance, so finding a solution to improve their capabilities while maintaining their low computation costs is necessary.

While training edge models with more labeled data samples to improve their performance on a certain downstream task would be one viable solution, collecting more labeled samples is labor-intensive and expensive. Without further data collection, one feasible solution to tackle such a challenge is performing knowledge distillation (KD) [1, 21, 40, 41, 61] with LVFMs serving as teachers for the edge models. This approach can enhance the performance of edge models to be comparable to that of LVFMs while maintaining their low computational costs. Specifically, by leveraging unlabeled data samples, we can extract meaningful information from LVFMs and transfer it to edge models without incurring expensive data annotation costs.

However, one major challenge of using LVFMs as teachers is the model discrepancy between LVFMs and edge models, leading to differences in representation spaces. In this KD setting, the model discrepancy arises from two primary factors. First, LVFMs are built with a vast number of parameters in order to understand the massive amount of knowledge from large-scale pretraining datasets while edge models are built on only a limited number of parameters. Second, the heterogeneous architectures (*i.e.*, architectural difference) between students (*e.g.*, CNN based) and teachers (*e.g.*, ViT based) may also cause the model discrepancy.

[†] Corresponding author. [‡] Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

To be more specific, prevalent LVFMs employ ViT-based architectures [20, 30, 31, 37, 57] while CNN-based edge models are favored [4, 14, 56] for their fast inference speed. Our preliminary experiments demonstrate that the existing KD methods fail to effectively improve the performance of student model when the model discrepancy is substantial (Section 3.2). Specifically, when utilizing larger backbones of teacher models (*e.g.*, changing the backbone from ViT-S to ViT-L), we observe that existing KD methods fail to further improve the performances of edge models as much as the improved performances of teachers.

To address this issue, we propose a *simple yet effective* feature alignment method called CustomKD, which alternates between two stages: 1) feature customization and 2) knowledge distillation. In the feature customization stage, we customize the well-generalized features of LVFMs to a given student model by aligning them to the representation space of the student using the student's head classifier. In the KD stage, we encourage the student model to imitate two different features: 1) the task-general feature extracted from the frozen teacher, and 2) the customized task-specific feature obtained during the feature customization stage. Since the student model, including its head classifier, is updated during the KD stage, we alternate these two stages to progressively enhance the student model by continuously improving task-specific features. Importantly, we do not alter the architectures or inference processes of edge models, allowing us to significantly improve their performance without increasing inference speed. Although feature alignment has been widely adopted and explored in various studies [2, 22, 44, 48, 63], we want to emphasize that finding how to perform feature alignment using LVFMs as teachers in KD is underexplored and our technical novelty lies in finding answer to such a challenge.

The major contributions of this paper are:

- Although using larger backbones improves the downstream task performances of LVFMs, our experiment demonstrates that existing KD methods fail to bring as much performance gain for edge models as for LVFMs.
- We propose CustomKD, a knowledge distillation method that enables to leverage LVFMs with large backbones as teachers by overcoming the large model discrepancy.
- CustomKD consistently improves the performances of edge models on various tasks (*e.g.*, unsupervised domain adaptation and semi-supervised learning), without architectural changes nor increased inference speed.

2. Related Work

2.1. Large Vision Foundation Models

The unprecedented breakthroughs of large language models in natural language processing have motivated computer vision studies to build large models for computer vision tasks. Due to such efforts, large vision foundation models (LVFMs) have received attention for their outstanding performances in diverse downstream tasks [20, 23, 31, 37, 42, 57], even with a simple linear probing. In this work, the definition of LVFMs includes the image encoders that were pretrained with images only (*e.g.*, DINOv2 [37]) and those of vision-language models (*e.g.*, CLIP [42]). DINOv2 [37] improves the performance of DINO [5] by leveraging a large-scale pretraining dataset curated using their proposed data processing pipeline, achieving state-of-the-art performances in diverse computer vision tasks. Similarly, CLIP [20, 42] shows robust generalization performances on diverse zero-shot classification tasks. However, as aforementioned, utilizing such LVFMs in real-world applications is challenging since they are generally built on millions of parameters, showing slow inference speed [12, 52, 55]. For this reason, we require edge models capable of demonstrating comparable performance to LVFMs while maintaining high computational efficiency in real-world applications.

2.2. Knowledge Distillation

Knowledge distillation (KD) is one viable solution to improve the performances of edge models (*i.e.*, student models) [1, 2, 7, 10, 17, 19, 21, 29, 35, 38, 40, 41, 44, 47, 49, 58, 61]. KD is widely known for improving the performances of student models by teaching the hidden inter-class relationship at the prediction level [1, 17, 61] or improving the representation by imitating the features of teacher models [40, 41, 44]. Due to this fact, we can utilize unlabeled datasets for further improving the performances of edge models using LVFMs as teachers [51]. However, applying existing KD methods with LVFMs as teacher models brings limited performance gain due to the model discrepancy (Fig. 1), which was underexplored in previous work.

Student-friendly knowledge distillation methods [2, 8, 15, 36, 38, 63] have been proposed to address the discrepancy between teacher and student models. For example, utilizing the frozen classifier of a pretrained student model for training teachers [2] could be one viable approach for the student-friendly knowledge distillation. However, previous studies attempting to reduce the model discrepancy have limited their experiments to using similar architectures for the student and the teacher model (*e.g.*, using ResNet architectures for both student and teacher). While a recent KD study tries to overcome the heterogeneous architectures of students and teachers [15, 33], they still only use small backbones of teacher models (*e.g.*, ViT-S). In this work, we first show that using large backbones (*e.g.*, ViT-L) of LVFMs as teacher models is indeed challenging to further improve the performances of edge models with KD. Then, we propose CustomKD that enhances the performances of edge models even with large backbones of teacher models in heterogeneous architectures.

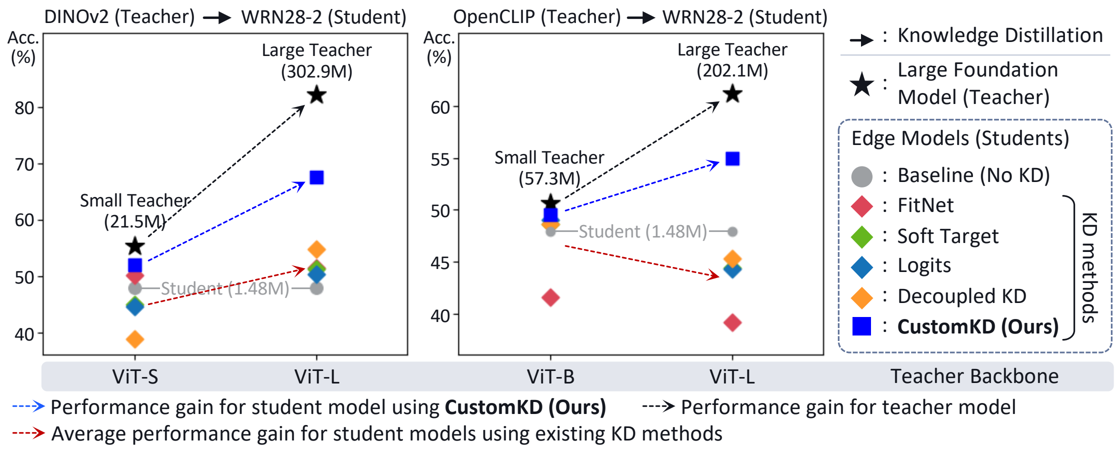

Figure 1. Limited performance gain with larger teachers. While utilizing small teachers (*e.g.*, ViT-S, ViT-B) brings comparable or better performance than the teacher's performance, existing KD methods fail to further improve student's performance with large teachers (*e.g.*, ViT-L). We use FitNet [44], Soft Target [17], Logits [1], and Decoupled KD [61] for conventional KD methods.

3. Method

3.1. Problem Setup

Throughout the paper, we define the student model as $\theta_s = (\theta_s^e, \theta_s^c)$ and the teacher model as $\theta_t = (\theta_t^e, \theta_t^c)$, where θ^e and θ^c represent the encoder and the head classifier, respectively. Generally, knowledge distillation enforces the student model θ_s to imitate the teacher model θ_t either at the feature level [40, 41, 44] or at the prediction level [1, 17, 61]. In this work, we assume that we have a pretrained θ_s that was trained with a small amount of labeled data D_L, composed of $\{x_i, y_i\}_{i=1}^{N_L} \in D_L$. Then, our main goal is to further improve the performance of θ_s by leveraging θ_t with an unlabeled data D_U, composed of $\{x_i,\}_{i=1}^{N_U} \in D_U$, additional to the labeled data D_L. Under such a task setting, we mainly conduct experiments on unsupervised domain adaptation (*i.e.*, UDA) and semi-supervised learning (*i.e.*, SSL), where D_U refers to the target dataset in UDA or the unlabeled dataset in SSL.

3.2. Motivation

Before diving into our proposed method, CustomKD, we first demonstrate the challenge of using LVFMs with large backbones (*e.g.*, ViT-L) as teacher models through a preliminary experiment on the semi-supervised learning task using CIFAR-100 dataset with 400 labeled samples. We use DINOv2 [37] and OpenCLIP [20] for the teacher models and WideResNet28-2 [59] for the student model.

The gray-colored dots indicate the SSL performances of the edge model, and the black-colored stars indicate those of LVFMs trained with a simple linear probing. Our main observation is that the existing KD methods fail to further improve the performance at a comparable level to the teacher's performance when changing the backbone from a small one (*e.g.*, ViT-S, ViT-B) to a larger one (*e.g.*, ViT-L). Specifically, the black dotted lines and the red dotted lines in Fig. 1 indicate the performance gain of teacher models and the averaged performance gain of student models, respectively, by changing the backbone of teacher models. As shown, the slopes of black lines are much steeper than those of red lines, indicating that the student models fail to improve their performances as much as teachers despite the large backbone of teachers used for KD. We conjecture that the main reason for this observation is due to the significant increase in model discrepancy when changing the teacher with a larger backbone, hindering further performance gain. Our proposed method, CustomKD, enables further performance gains even when using large backbones of LVFMs, as shown in blue squares and blue dotted lines in Fig. 1.

3.3. Proposed Method

Overall procedure Fig. 2 depicts the overall process of CustomKD. We have two different stages that alternate throughout the training process: 1) feature customization stage and 2) knowledge distillation stage. At a high level, during the feature customization stage, the well-generalized original features of the teacher model, f_t, are transformed into easily comprehensible features for the student model, $\tilde{f}_t$, by leveraging the classifier, θ_s^c, shared from the student model. Then, during the KD stage, we encourage f_s to learn knowledge from both f_t and $\tilde{f}_t$ by using MSE loss.

Feature customization stage The main purpose of this stage is to customize the well-generalized features of LVFMs to easily comprehensible features for a given student model. We extract the original feature of the teacher model, $f_t = \theta_t^e(x)$, and forward to a projection layer θ_t^h in order to obtain $\tilde{f}_t = \theta_t^h(f_t)$. During this process, assuming that we have a pretrained student model θ_s, we replace the head classifier of the teacher model with the head classifier of the student model, denoted as θ_s^c. We forward $\tilde{f}_t$ to the frozen θ_s^c and only update the projection layer θ_t^h using the labeled data, $(x, y) \in D_L$, formulated as,

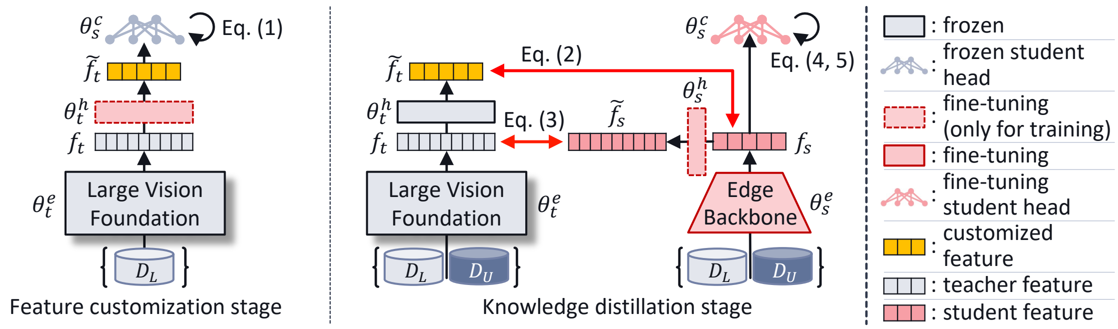

Figure 2. Overall framework of CustomKD. In the feature customization stage, we customize the well-generalized features of LVFMs to a given edge model using its head classifier (θ_s^c). In the KD stage, we enforce the edge model to imitate the 1) task-general feature and 2) customized task-specific feature from the teachers. We alternate these two stages every epoch throughout the training process.

$$\min_{\theta_t^h} \mathcal{L}_t = CE(\theta_s^c(\tilde{f}_t), y). \tag{1}$$

Through this process, we obtain $\tilde{f}_t$, the feature customized to a given student model derived from the well-generalized representation of the teacher model. Here, we do not use unlabeled data for this stage since our main goal is to use the well-generalized representation of θ_t^e, rather than obtaining the optimal parameters of θ_t^h that adapted well to the unlabeled data (*e.g.*, target domain for UDA task).

Knowledge distillation stage During this stage, we encourage f_s to learn knowledge from two different features: 1) f_t, which is the original feature from the teacher model, and 2) $\tilde{f}_t$, which is the feature we obtained during the feature customization stage. Intuitively, the two features contain different knowledge. While f_t preserves the knowledge that the teacher model learned during the pretraining stage without loss of information, it only contains the task-general knowledge. On the other hand, while $\tilde{f}_t$ includes task-specific knowledge customized for promoting the understanding of a given student model, it inevitably has loss of information due to the projection layer θ_t^h. Our goal is to encourage the student model to learn both task-specific and task-general knowledge from the two different features, which the necessity of each knowledge is demonstrated in Table 7.

In this stage, we use both labeled and unlabeled datasets, $x \in \{D_L, D_U\}$. Since the customized feature $\tilde{f}_t = \theta_t^h(f_t)$ has the same embedding dimension with f_s, the supervision for imitating the task-specific feature $\tilde{f}_t$ is formulated as,

$$\min_{\theta_s^e} \mathcal{L}_{\tilde{f}_t} = ||f_s - \tilde{f}_t||^2. \tag{2}$$

Regarding the supervision on task-general feature f_t, f_t generally have different embedding dimensions with f_s, so we forward f_s to a projection layer θ_s^h and obtain $\tilde{f}_s = \theta_s^h(f_s)$ for imitating f_t, which is formulated as,

$$\min_{\theta_s^h, \theta_s^e} \mathcal{L}_{f_t} = ||\tilde{f}_s - f_t||^2. \tag{3}$$

Additionally, we use the standard cross entropy loss for the labeled data and entropy minimization for the unlabeled data, widely used in domain adaptation studies [9, 25, 27, 28, 39, 46], in order to prevent the model from being overfitted to the labeled data. The two loss functions are formulated as,

$$\min_{\theta_s^c, \theta_s^e} \mathcal{L}_L = CE(\theta_s(x_L), y_L), \tag{4}$$

$$\min_{\theta_s^c, \theta_s^e} \mathcal{L}_U = H(\hat{\theta}_s(x_U)), \tag{5}$$

where $H(p) = \Sigma_{k=1}^C p^k \log p^k$ with C number of classes. Then, our final loss function is formulated as follows,

$$\min_{\theta_s^h, \theta_s^c, \theta_s^e} \mathcal{L}_s = \mathcal{L}_L + \lambda_U \mathcal{L}_U + \lambda_{f_t} \mathcal{L}_{f_t} + \lambda_{\tilde{f}_t} \mathcal{L}_{\tilde{f}_t}. \tag{6}$$

As aforementioned, CustomKD alternates the feature customization stage and the KD stage after each epoch. In other words, we bring θ_s^c for the head classifier of the teacher during the feature customization stage after every epoch of the knowledge distillation stage.

Following are the advantages of CustomKD. First, CustomKD is independent of the original training process of the edge model, so it improves the performance of any given off-the-shelf pretrained edge model. Second, CustomKD does not incur additional computational costs during the inference stage as we discard θ_t^h and θ_s^h, which are updated during the training phase. Consequently, we believe that CustomKD can significantly improve the performance of a given off-the-shelf edge model without additional inference costs by leveraging LVFMs even with heterogeneous architectures distinct from the edge model.

4. Experiments

4.1. Experimental Setup

Evaluation settings As aforementioned, the main goal of this work is to utilize the unlabeled data for further improving downstream tasks of edge models leveraging LVFMs as the teacher models. For this, we conduct experiments on tasks where unlabeled data are given: 1) unsupervised domain adaptation (UDA) and 2) semi-supervised learning

Category	Method	A2CA	A2P	A2RW	CA2A	CA2P	CA2RW	P2A	P2CA	P2RW	RW2A	RW2CA	RW2P	Avg.
	DINOv2 [37]	73.95	87.07	88.50	82.53	86.84	86.21	73.55	68.29	85.61	84.51	75.58	92.95	82.13
	MobileNetV3 [18]	37.57	46.74	58.39	33.79	48.50	48.73	32.92	35.12	56.69	49.03	42.36	66.84	46.39
UDA	DAN [34]	33.68	43.34	54.62	31.81	46.86	48.22	31.69	35.21	55.27	47.01	40.76	64.16	44.39
	Deep Coral [48]	34.16	44.04	54.49	31.60	46.79	48.18	31.69	35.19	55.36	46.64	40.64	64.02	44.40
	DANN [13]	36.54	45.69	58.34	33.75	47.67	49.19	32.01	35.21	56.32	48.54	43.14	66.86	46.11
	DSAN [64]	37.73	47.08	58.94	34.98	48.93	50.75	33.75	37.27	58.73	50.14	44.08	67.90	47.52
Feature KD	CC [41]	36.04	50.82	60.73	37.29	52.26	52.90	33.66	36.40	59.95	51.30	43.39	70.85	48.80
	RKD [40]	44.51	58.30	63.99	40.34	59.52	57.22	37.37	46.46	64.59	56.94	54.00	76.41	54.97
	FitNet [44]	55.05	69.36	72.92	55.42	68.19	70.12	50.35	55.92	76.36	68.52	63.51	83.98	65.81
	CustomKD	58.65	72.65	74.57	62.67	76.08	74.04	57.97	56.72	78.08	71.61	63.83	85.45	**69.36**
Prediction KD	Soft Target [17]	53.33	63.08	69.57	50.60	64.9	62.52	44.75	49.67	67.11	63.33	59.61	80.47	60.75
	Logits [1]	58.47	73.06	76.68	64.24	75.74	73.15	56.7	58.79	73.93	73.38	64.15	85.90	69.52
	DKD [61]	60.27	72.47	76.91	63.25	75.74	72.32	55.62	57.89	74.71	73.96	64.63	85.94	69.48
	DKD [61] + CustomKD	63.60	77.95	80.17	70.70	80.63	79.00	64.36	64.40	80.67	77.59	67.90	88.60	**74.63**

Table 1. Image classification accuracy on OfficeHome. Bold digits indicate the best results in averaged accuracy among each category of knowledge distillation baselines.

Category	Method	RW2CA	RW2P	RW2S	CA2RW	CA2P	CA2S	P2RW	P2CA	P2S	S2RW	S2CA	S2P	Avg.
	DINOv2 [37]	69.63	64.57	61.41	71.05	58.46	63.62	72.64	63.14	59.77	70.30	73.62	62.18	65.87
	MobileNetV3 [18]	38.43	35.83	24.54	39.76	24.36	28.59	45.29	33.42	26.40	38.93	43.16	30.46	34.10
UDA	DAN [34]	32.47	32.83	21.54	36.03	21.59	24.28	40.84	26.90	22.61	34.31	35.28	27.12	29.65
	Deep Coral [48]	32.55	32.86	21.39	36.00	21.57	24.23	40.87	26.90	22.46	34.34	35.20	27.07	29.62
	DANN [13]	34.71	33.73	22.23	37.52	22.73	25.62	42.25	29.49	23.89	36.01	38.76	28.16	31.26
	DSAN [64]	38.74	37.20	26.36	39.88	25.25	29.94	44.57	32.76	27.37	38.40	42.74	30.73	34.50
Feature KD	CC [41]	36.75	36.90	21.20	39.30	24.13	27.88	45.79	32.72	25.08	39.46	46.17	30.92	33.86
	RKD [40]	38.63	36.91	24.31	37.28	23.94	29.57	44.15	33.58	26.70	36.41	45.80	31.00	34.02
	FitNet [44]	43.08	42.42	27.99	41.71	26.24	34.59	48.83	37.78	29.14	42.04	50.91	34.11	38.24
	CustomKD	43.26	42.76	29.08	41.38	26.91	34.3	48.43	37.72	30.91	41.98	51.18	34.15	**38.51**
Prediction KD	Soft Target [17]	42.88	39.07	28.45	40.81	26.46	31.66	46.06	36.75	29.39	39.45	47.67	32.24	36.74
	Logits [1]	39.36	36.69	27.16	36.90	23.08	30.56	42.72	30.52	26.49	32.49	46.05	26.97	33.25
	DKD [61]	45.00	41.40	31.70	39.91	26.82	34.58	46.53	36.61	31.26	38.85	50.92	32.34	37.99
	DKD [61] + CustomKD	47.01	43.44	35.88	45.15	31.25	36.42	48.91	38.63	33.79	43.90	52.82	36.00	**41.10**

Table 2. Image classification accuracy on DomainNet. Bold digits indicate the best results in averaged accuracy among each category of knowledge distillation baselines.

(SSL). Then, we bring a pretrained off-the-shelf edge model and show that CustomKD consistently improves its performances across diverse tasks. Since our framework shows the effectiveness of utilizing large models as teachers in KD for tasks with unlabeled data, we compare CustomKD with baseline methods of both KD and each task.

Datasets We use OfficeHome and DomainNet for the UDA task and CIFAR-100 and ImageNet for the SSL task. For CIFAR-100, we vary the number of labeled samples to 400, 2500, and 10000, following the Unified SSL Benchmark (USB) [53]. For ImageNet, we use subsets of 1% and 10% of labeled images and conduct comparisons with other existing baselines, following SimMatch [62].

Implementation details For the teacher models, we use DINOv2 [37] and OpenCLIP [20] for both tasks. For the student models, we use MobileNetV3 [18] for the UDA task, and WideResNet28-2 [59], ResNet18 [16], and ResNet50 [16] for the SSL task. Regarding the KD baseline methods, we use the codes provided by the repository named Knowledge-Distillation-Zoo [1]. For our experiments, we select conventional KD baseline methods that are categorized into prediction-level and feature-level methods. For the prediction-level KD methods (*e.g.*, Logits [1], Soft Targets [17], and Decoupled KD [61]), we perform linear probing and obtain a pretrained head classifier of teacher model, θ_t^c. For the feature-level KD methods (*e.g.*, Relational KD [40], Correlation Congruence [41], and FitNet [44]), including CustomKD, we only use the frozen teacher model θ_t^e, without training beforehand. Beyond conventional KD methods, we further compare CustomKD with recent KD approaches, including TfFD [35], NORM [29], and DisWot [10], on OfficeHome to demonstrate the superiority of CustomKD. For the UDA task, we follow the protocol of the repository named DeepDA [2]. Regarding the SSL, we follow USB [53] for CIFAR-100 and SimMatch [62] for ImageNet. Since we observe a performance gap between the reported values of USB and our obtained results by running the codes of USB on CIFAR-100, we additionally report our obtained results by denoting $*$ next to the results.

4.2. Image Classification

Unsupervised domain adaptation Table 1 and Table 2 compare the UDA performances of both UDA and conventional KD baseline methods on OfficeHome and DomainNet, respectively. We use MobileNetV3 for the student model and DINOv2 (ViT-L) for the teacher model. CustomKD achieves the best performances compared to ex-

[1]https://github.com/AberHu/Knowledge-Distillation-Zoo

[2]https://github.com/jindongwang/transferlearning/tree/master/code/DeepDA

Method	CA2A	CA2P	CA2RW	Avg.
Source Only	33.79	48.50	48.73	43.67
TfFD [35]	37.82	53.14	53.45	48.14
TfFD+ CustomKD	48.87	63.60	61.40	**57.96**
NORM [29]	49.03	63.73	64.84	59.20
NORM+ CustomKD	57.40	72.52	70.03	**66.65**
DisWot [10]	71.61	80.99	78.75	77.12
DisWot+ CustomKD	71.94	81.30	79.39	**77.54**

Table 3. Comparisons of CustomKD with recent knowledge distillation methods on UDA task using OfficeHome, using clipart as the source domain.

isting both UDA and feature-level KD baseline methods. When comparing with the source model, we improve the UDA performance of the source model by a large margin (*e.g.*, an average of 22.97% performance gain for OfficeHome). Since CustomKD applies KD at the feature level, it is also applicable with prediction-level KD (*e.g.*, Logits [1], Soft Targets [17], DKD [61]), which we show its applicability by using CustomKD on DKD [61]. As shown, applying CustomKD on DKD achieves the best performances compared to other baseline methods. Since DomainNet is a relatively more challenging dataset than OfficeHome, we bring relatively small performance gain for DomainNet compared to OfficeHome. Still, applying CustomKD on DKD achieves an average of 7% performance gain compared to the source model, which is the best UDA performance among existing baseline methods.

Table 3 compares CustomKD with recent KD baseline methods, further demonstrating its superiority. For the experiments, we apply CustomKD to various recent KD methods, including TfFD [35], NORM [29], and DisWot [10]. As shown, CustomKD is applicable to an arbitrarily given KD method and shows consistent performance improvements. This result clearly highlights the effectiveness of CustomKD, even when compared with recent KD methods.

Semi-supervised learning We also demonstrate the effectiveness of CustomKD on SSL, another task improving performances with unlabeled datasets. Table 4 compares the error rates of CIFAR-100 on both SSL and KD methods. We apply KD methods on WideResNet28-2 [59] pretrained with AdaMatch [3]. While we report the results of AdaMatch that we obtain by running the codes of USB, we use the mean results of other SSL methods reported in USB. We only report the recent state-of-the-art SSL methods while we show the results including other SSL methods in our Supplementary. As shown, we achieve the best error rates of CIFAR-100 on AdaMatch, outperforming the existing SSL methods. We want to emphasize that CustomKD does not require sophisticated techniques such as strong data augmentations or thresholding, which are required in various previous SSL studies [3, 26, 53, 60, 62]. Additionally, due to the page limit, we provide experimental

Category	Methods	Labels 400	2500	10000
SSL	FixMatch [45]	53.37	34.29	28.28
	SimMatch [62]	48.82	32.54	26.42
	FreeMatch [54]	49.24	32.79	27.17
	SoftMatch [6]	49.64	33.05	27.26
	AdaMatch* [3]	52.07	37.92	32.5
KD	Soft Target [17]	48.71	31.73	27.66
	Logits [1]	49.71	33.42	28.16
	DKD [61]	45.18	30.43	26.19
	RKD [40]	50.11	34.24	29.11
	CC [41]	49.85	33.72	28.75
	FitNet [44]	48.58	30.87	29.41
	CustomKD	**32.51**	**25.52**	**24.66**

Table 4. Error rates of semi-supervised learning on CIFAR-100. * indicates reproduced results using codes of USB benchmark.

results on ImageNet in our Supplementary to demonstrate the superiority of CustomKD over other SSL studies.

5. Analysis

5.1. Enhancing Knowledge Distillation Across Backbone Scales

The preliminary experiment in Sec. 3.2 motivates our work to propose CustomKD that overcomes the large model discrepancy between the student and the teacher model. To demonstrate that CustomKD indeed improves the performance of the edge model even with the large backbone of teacher models, we conduct experiments with both small and large backbones of teacher models. Under the SSL task, we use WideResNet28-2 [59] and ResNet18 [16] for the student models and DINOv2 [37] and OpenCLIP [20] for the teacher models. We mainly compare with using $\mathcal{L}_{f_t}$ to demonstrate the necessity of training with $\mathcal{L}_{\tilde{f}_t}$ when the model discrepancy is substantial.

In Table 5, we observe that using $\mathcal{L}_{\tilde{f}_t}$ additional to $\mathcal{L}_{f_t}$ consistently improves the SSL performances of edge models, regardless of the backbones of the teacher models. Specifically, the performance gap between the two loss functions generally enhances as the model discrepancy increases. For example, with WideResNet28-2 as the student model, the performance gap increases from 1.91 to 16.07 for DINOv2 and from 7.97 to 15.87 for OpenCLIP by using larger backbones of teacher models. The reason behind such an observation is as follows. Using $\mathcal{L}_{f_t}$ simply projects the feature of the student to imitate that of the teacher. Simply utilizing a projection layer may limit fully understanding the knowledge of the teacher, especially when the model discrepancy is substantial. On the other hand, using $\mathcal{L}_{\tilde{f}_t}$ additional to $\mathcal{L}_{f_t}$ not only teaches the task-general knowledge but also the task-specific knowledge customized for the student model, further boosting the performance of student regardless of the model discrepancy.

Teacher Type	Teacher Backbone	Teacher Error Rate*	Methods	WideResNet28-2 [59] (1.48M, 0.22G)	ResNet18 [16] (11.23M, 0.04G)
-	-	-	Source	52.07	73.10
DINOv2 [5]	ViT-S (21.52M, 5.52G)	44.69	$\mathcal{L}_{f_t}$	49.88	63.50
			$\mathcal{L}_{f_t} + \mathcal{L}_{\tilde{f}_t}$	**47.97 (-1.91)**	**53.22 (-10.28)**
	ViT-L (302.91M, 77.82G)	17.92	$\mathcal{L}_{f_t}$	48.58	50.89
			$\mathcal{L}_{f_t} + \mathcal{L}_{\tilde{f}_t}$	**32.51 (-16.07)**	**46.73 (-4.16)**
OpenCLIP [20]	ViT-B (57.26M, 11.27G)	49.38	$\mathcal{L}_{f_t}$	58.46	75.51
			$\mathcal{L}_{f_t} + \mathcal{L}_{\tilde{f}_t}$	**50.49 (-7.97)**	**58.56 (-16.95)**
	ViT-L (202.05M, 51.89G)	38.78	$\mathcal{L}_{f_t}$	60.89	75.70
			$\mathcal{L}_{f_t} + \mathcal{L}_{\tilde{f}_t}$	**45.02 (-15.87)**	**56.55 (-19.15)**

Table 5. Error rates of CIFAR-100 using 400 labeled samples. Brackets indicate the number of parameters and Multiply-Accumulate Operations (MACs),. * indicates that we performed linear probing using only labeled samples for the teacher.

Teacher Head Classifier	Teacher Acc.	Student Acc.↑	CKA(f_s, f_t)↑	CKA(f_s, $\tilde{f}_t$)↑
θ_t^c	76.70	47.15	0.45	0.44
θ_s^c	75.48	**56.47**	**0.54**	**0.62**

Table 6. Comparisons on the initialization of the head classifier of teacher models during the feature customization stage. θ_t^c indicates randomly initialized head classifier.

Additionally, student models may achieve better performance than the teacher model, even with a smaller number of parameters and lower computational costs. The first and the second numbers in the bracket under each model indicate the number of parameters and Multiply-Accumulate Operations (MACs), respectively. Applying CustomKD on WideResNet28-2 using DINOv2 with a large backbone (*i.e.*, ViT-L) as the teacher achieves the error rate of 32.51%. On the other hand, performing linear probing of DINOv2 with a small backbone (*i.e.*, ViT-S) achieves the error rate of 44.69%. While the number of parameters and MACs of WideResNet28-2 are 1.48M and 0.22G, respectively, it achieves better performance compared to DINOv2 with ViT-S backbone, which has 21.52M number of parameters and 5.52G MACs. Such a result clearly demonstrates the practicality of CustomKD when deploying such lightweight models on real-world applications with a comparable or better performance than teacher models.

5.2. Importance of Using Student Head Classifier

During the feature customization stage, we bring θ_s^c after every epoch of knowledge distillation stage. Table 6 compares such a design choice with utilizing a randomly initialized head classifier for the teacher model. We conduct experiments on the UDA task on OfficeHome with MobileNetV3 [18] and OpenCLIP [20] (ViT-L) for the student and the teacher, respectively. In Table 6, θ_t^c indicates the randomly initialized head classifier for the teacher model while θ_s^c refers to using the student head classifier for the teacher model. As shown, utilizing θ_t^c improves the UDA performance of the teacher model, denoted as Teacher Acc. However, improving the performance of the teacher model does not guarantee a better performance of the student model, as supported by previous studies [8, 36, 63]. In other words, using θ_s^c for the head classifier of the teacher model improves the UDA performance of the student model, even with degraded performance of the teacher model.

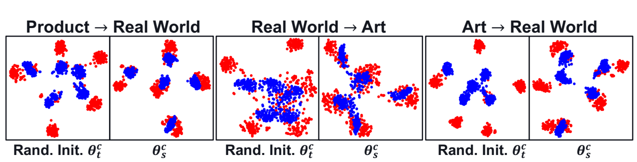

Figure 3. t-SNE visualization of $\tilde{f}_t$ (red) and f_s (blue) on OfficeHome. For each domain, the left and right indicates training with randomly initialized θ_t^c and θ_s^c, respectively, for the head classifier.

We further analyze such a result by measuring the centered kernel alignment (CKA) [24] between 1) f_s and f_t and 2) f_s and $\tilde{f}_t$. CKA enables to compute the representation similarity of two matrices even with different embedding dimensions [15, 43]. The higher the CKA values are, the similar the two matrices is. Table 6 shows that using θ_s^c for the head classifier of the teacher model increases both CKA(f_s, f_t) and CKA$(f_s, \tilde{f}_t)$. In other words, θ_s^c enables the student feature f_s to imitate both the original task-general teacher feature f_t and the customized task-specific teacher feature $\tilde{f}_t$ well. Additionally, using θ_s^c further increases CKA$(f_s, \tilde{f}_t)$ compared to CKA(f_s, f_t), indicating that the representation of $\tilde{f}_t$ is aligned better with that of f_s compared to f_t due to the feature customization stage.

The t-SNE visualization [50] of $\tilde{f}_t$ and f_s using 1) randomly initialized θ_t^c and 2) θ_s^c in Fig. 3 also supports such a result. The red and the blue dots indicate the projection of $\tilde{f}_t$ and f_s, respectively. While we use OfficeHome for the dataset of the visualization, we only select most frequent 5 categories among 65 categories to avoid information overload. When utilizing the randomly initialized θ_t^c, the blue dots misalign with the red dots, indicating the distinct representation spaces of $\tilde{f}_t$ and f_s. On the other hand,

$\mathcal{L}_L$, $\mathcal{L}_U$	$\mathcal{L}_{f_t}$	$\mathcal{L}_{\tilde{f}_t}$	Avg. Acc.
✓	–	–	56.16
✓	✓	–	56.32
✓	–	✓	62.14
✓	✓	✓	**62.76**

(a) KD loss functions.

Alternating Epochs	Avg. Acc.
30:1	62.22
10:1	62.20
5:1	62.28
1:1	**62.76**

(b) Alternating epochs for θ_t^h

Table 7. Further studies on (a) our loss functions and (b) the frequency of the feature customization stage. We use MobileNetV3 [18] and OpenCLIP [20] (ViT-B) for the student and the teacher model, respectively. We average the results of three target domains (*i.e*., art, clipart, and product) using the student model pretrained on the images of real world as the source domain in OfficeHome.

when employing θ_s^c, the blue dots align well with the red dots, indicating that the two representation spaces are similar and therefore demonstrating the necessity of using θ_s^c for the head classifier of the teacher model.

5.3. Ablation Studies

Loss functions Table 7(a) shows the performance gain of using each loss function $\mathcal{L}_{f_t}$ and $\mathcal{L}_{\tilde{f}_t}$. Additional to using only $\mathcal{L}_L$ and $\mathcal{L}_U$, individually utilizing $\mathcal{L}_{f_t}$ and $\mathcal{L}_{\tilde{f}_t}$ brings performance gain. When using the two KD loss functions together, we achieve the best performance. While using $\mathcal{L}_{\tilde{f}_t}$ brings larger performance gain, this clearly demonstrates the necessity of learning both task-general knowledge from $\mathcal{L}_{f_t}$ and customized task-specific knowledge from $\mathcal{L}_{\tilde{f}_t}$.

Alternating epochs Given that we perform the knowledge distillation stage every epoch, Table 7(b) investigates how the frequency of the feature customization stage influences the performance of the student model θ_s. For example, $5:1$ refers to updating θ_t^h in the feature customization stage after every 5 epochs of the knowledge distillation stage. We observe that the performance degrades as θ_t^h is updated less frequently. The main reason is as follows. While θ_s is consistently updated during training, the teacher model may fail to provide adequate $\tilde{f}_t$ for θ_s if θ_t^h is less frequently updated, demonstrating the necessity of consistent update of $\tilde{f}_t$. While the alternating epoch is a hyper-parameter, we emphasize that alternating the two stages itself is important for improving KD performances.

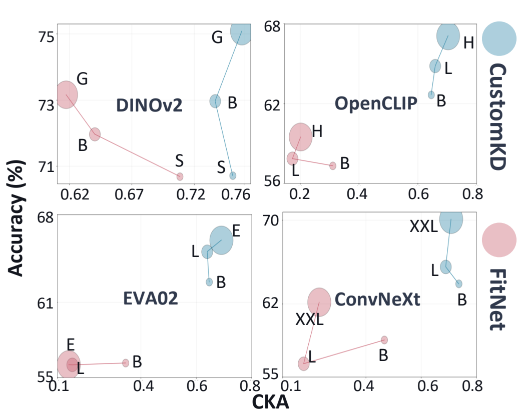

Figure 4. Consistent performance gains of using CustomKD compared to FitNet across diverse teachers and backbone scales.

5.4. Consistent Performance Gains Across Teachers

Additionally, Fig. 4 demonstrates that CustomKD consistently improves KD performance across various large teacher backbones. For our analysis, we conduct experiments using UDA with MobileNetV3 as the student model, OfficeHome (with Real World as the source domain) as the dataset, and FitNet [44] for the baseline. The x-axis and y-axis indicate the centered kernel alignment (CKA) [24] and accuracy, respectively. For computing CKA values, we use f_s for the student model, while utilizing f_t and $\tilde{f}_t$ for FitNet and CustomKD, respectively. As illustrated, CustomKD outperforms FitNet across diverse teacher models, including DINOv2 [37], OpenCLIP [20], EVA02 [11], and ConvNeXt [32], spanning various backbone scales.

6. Future Work and Conclusions

In this work, we introduce CustomKD, a method that customizes the well-generalized features of large vision foundation models for a given edge model, aiming to further improve performances of edge models on downstream tasks. Our preliminary experiment shows that existing KD methods bring limited performances gains of edge models, even when employing large backbones in teacher models, due to large model discrepancies. To address this issue, we propose aligning the representation of the teacher model to that of the student model by bringing the head classifier of the student. CustomKD alternates two stages: 1) feature customization stage that aligns the representation space of the teacher to that of the student and 2) knowledge distillation stage that encourages the feature of the student to imitate features of the teacher. Our work achieves new state-of-the-art performances on tasks with unlabeled data given, including UDA and SSL. While our work mainly focuses on image classification tasks, we believe that this framework could also be applied to dense prediction tasks (*e.g*., semantic segmentation). To extend to such tasks, performing KD of spatial-wise knowledge should also be considered, which we leave for future work. We believe that our work inspires future researchers to further endeavor to build fast and well-performing edge models by leveraging the rapidly developing large vision foundation models.

Acknowledgement We would like to thank Sungrack Yun, Simyung Chang, Hyunsin Park, Janghoon Cho, Juntae Lee, Hyoungwoo Park, Seokeon Choi, Kyuhong Shim, Seunghan Yang, Jihwan Bang, and Sunghyun Park of the Qualcomm AI Research team for their valuable discussions.

References

[1] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *NeurIPS*, 2014. 1, 2, 3, 5, 6

[2] Emanuel Ben-Baruch, Matan Karklinsky, Yossi Biton, Avi Ben-Cohen, Hussam Lawen, and Nadav Zamir. It's all in the head: Representation knowledge distillation through classifier sharing, 2022. 2

[3] David Berthelot, Rebecca Roelofs, Kihyuk Sohn, Nicholas Carlini, and Alexey Kurakin. Adamatch: A unified approach to semi-supervised learning and domain adaptation. In *ICLR*, 2022. 6

[4] Hamidreza Bolhasani, Maryam Mohseni, and Amir Masoud Rahmani. Deep learning applications for iot in health care: A systematic review. *Informatics in Medicine Unlocked*, 2021. 2

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 2, 7

[6] Hao Chen, Ran Tao, Yue Fan, Yidong Wang, Jindong Wang, Bernt Schiele, Xing Xie, Bhiksha Raj, and Marios Savvides. Softmatch: Addressing the quantity-quality trade-off in semi-supervised learning. 2023. 6

[7] 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. 2

[8] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *ICCV*, 2019. 2, 7

[9] Sungha Choi, Seunghan Yang, Seokeon Choi, and Sungrack Yun. Improving test-time adaptation via shift-agnostic weight regularization and nearest source prototypes. *ECCV*, 2022. 4

[10] Peijie Dong, Lujun Li, and Zimian Wei. Diswot: Student architecture search for distillation without training. In *CVPR*, 2023. 2, 5, 6

[11] Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva-02: A visual representation for neon genesis. *Image Vis. Comput.*, 2024. 8

[12] Roya Firoozi, Johnathan Tucker, Stephen Tian, Anirudha Majumdar, Jiankai Sun, Weiyu Liu, Yuke Zhu, Shuran Song, Ashish Kapoor, Karol Hausman, Brian Ichter, Danny Driess, Jiajun Wu, Cewu Lu, and Mac Schwager. Foundation models in robotics: Applications, challenges, and the future, 2023. 1, 2

[13] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 2016. 5

[14] Tian Guo. Cloud-based or on-device: An empirical study of mobile deep inference. *2018 IEEE International Conference on Cloud Engineering (IC2E)*, 2017. 2

[15] 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. 2, 7

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 5, 6, 7

[17] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. 2, 3, 5, 6

[18] Andrew Howard, Ruoming Pang, Hartwig Adam, Quoc V. Le, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, and Yukun Zhu. Searching for mobilenetv3. In *ICCV*, 2019. 5, 7, 8

[19] Tao Huang, Shan You, Fei Wang, Chen Qian, and Chang Xu. Knowledge distillation from a stronger teacher. In *NeurIPS*, 2022. 2

[20] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, 2021. 1, 2, 3, 5, 6, 7, 8

[21] Mingi Ji, Byeongho Heo, and Sungrae Park. Show, attend and distill: Knowledge distillation via attention-based feature matching. *AAAI*, 2021. 1, 2

[22] Sanghun Jung, Jungsoo Lee, Nanhee Kim, Amirreza Shaban, Byron Boots, and Jaegul Choo. Cafa: Class-aware feature alignment for test-time adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19060–19071, 2023. 2

[23] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. Segment anything. In *ICCV*, 2023. 1, 2

[24] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *ICML*, 2019. 7, 8

[25] Jungsoo Lee, Debasmit Das, Jaegul Choo, and Sungha Choi. Towards open-set test-time adaptation utilizing the wisdom of crowds in entropy minimization. In *ICCV*, 2023. 4

[26] Junnan Li, Caiming Xiong, and Steven C.H. Hoi. Semi-supervised learning with contrastive graph regularization. In *ICCV*, 2021. 6

[27] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? Source hypothesis transfer for unsupervised domain adaptation. In *ICML*, 2020. 4

[28] Hyesu Lim, Byeonggeun Kim, Jaegul Choo, and Sungha Choi. TTN: A domain-shift aware batch normalization in test-time adaptation. In *ICLR*, 2023. 4

[29] Xiaolong Liu, Lujun Li, Chao Li, and Anbang Yao. Norm: Knowledge distillation via n-to-one representation matching. In *ICLR*, 2023. 2, 5, 6

[30] 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. 2

[31] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2

[32] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *CVPR*, 2022. 8

[33] Zichang Liu, Qingyun Liu, Yuening Li, Liang Liu, Anshumali Shrivastava, Shuchao Bi, Lichan Hong, Ed H. Chi, and Zhe Zhao. Wisdom of committee: Distilling from foundation model to specialized application model, 2024. 2

[34] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *ICML*. PMLR, 2015. 5

[35] Li Lujun. Self-regulated feature learning via teacher-free feature distillation. In *European Conference on Computer Vision (ECCV)*, 2022. 2, 5, 6

[36] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. *AAAI*, 2020. 2, 7

[37] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023. 1, 2, 3, 5, 6, 8

[38] Dae Young Park, Moon-Hyun Cha, Changwook Jeong, Daesin Kim, and Bohyung Han. Learning student-friendly teacher networks for knowledge distillation. In *NeurIPS*, 2021. 2

[39] Sunghyun Park, Seunghan Yang, Jaegul Choo, and Sungrack Yun. Label shift adapter for test-time adaptation under covariate and label shifts. In *ICCV*, 2023. 4

[40] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *CVPR*, 2019. 1, 2, 3, 5, 6

[41] 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. 1, 2, 3, 5, 6

[42] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 1, 2

[43] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? In *NeurIPS*, 2021. 7

[44] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yann LeCun Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015. 2, 3, 5, 6, 8

[45] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence, 2020. 6

[46] Junha Song, Jungsoo Lee, In So Kweon, and Sungha Choi. Ecotta: Memory-efficient continual test-time adaptation via self-distilled regularization. *CVPR*, 2023. 4

[47] Samuel Stanton, Pavel Izmailov, Polina Kirichenko, Alexander A Alemi, and Andrew G Wilson. Does knowledge distillation really work? In *NeurIPS*, 2021. 2

[48] Baochen Sun and Kate Saenko. Deep CORAL: correlation alignment for deep domain adaptation. In *Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III*, 2016. 2, 5

[49] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 2

[50] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 2008. 7

[51] Raviteja Vemulapalli, Hadi Pouransari, Fartash Faghri, Sachin Mehta, Mehrdad Farajtabar, Mohammad Rastegari, and Oncel Tuzel. Knowledge transfer from vision foundation models for efficient training of small task-specific models. In *ICML*. PMLR, 2024. 2

[52] Jiaqi Wang, Zheng Liu, Lin Zhao, Zihao Wu, Chong Ma, Sigang Yu, Haixing Dai, Qiushi Yang, Yi-Hsueh Liu, Songyao Zhang, Enze Shi, Yi Pan, Tuo Zhang, Dajiang Zhu, Xiang Li, Xi Jiang, Bao Ge, Yixuan Yuan, Dinggang Shen, Tianming Liu, and Shu Zhang. Review of large vision models and visual prompt engineering. *ArXiv*, 2023. 1, 2

[53] Yidong Wang, Hao Chen, Yue Fan, Wang Sun, Ran Tao, Wenxin Hou, Renjie Wang, Linyi Yang, Zhi Zhou, Lan-Zhe Guo, Heli Qi, Zhen Wu, Yu-Feng Li, Satoshi Nakamura, Wei Ye, Marios Savvides, Bhiksha Raj, Takahiro Shinozaki, Bernt Schiele, Jindong Wang, Xing Xie, and Yue Zhang. Usb: A unified semi-supervised learning benchmark for classification. In *NeurIPS*, 2022. 5, 6

[54] Yidong Wang, Hao Chen, Qiang Heng, Wenxin Hou, Yue Fan, , Zhen Wu, Jindong Wang, Marios Savvides, Takahiro Shinozaki, Bhiksha Raj, Bernt Schiele, and Xing Xie. Freematch: Self-adaptive thresholding for semi-supervised learning. 2023. 6

[55] Xu Yan, Haiming Zhang, Yingjie Cai, Jingming Guo, Weichao Qiu, Bin Gao, Kaiqiang Zhou, Yue Zhao, Huan Jin, Jiantao Gao, Zhen Li, Lihui Jiang, Wei Zhang, Hongbo Zhang, Dengxin Dai, and Bingbing Liu. Forging vision foundation models for autonomous driving: Challenges, methodologies, and opportunities, 2024. 1, 2

[56] Jiyeon Yu, Angélica de Antonio, and Elena Villalba Mora.

Deep learning (cnn, rnn) applications for smart homes: A systematic review. *Computers*, 2022. 2

[57] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, Ce Liu, Mengchen Liu, Zicheng Liu, Yumao Lu, Yu Shi, Lijuan Wang, Jianfeng Wang, Bin Xiao, Zhen Xiao, Jianwei Yang, Michael Zeng, Luowei Zhou, and Pengchuan Zhang. Florence: A new foundation model for computer vision, 2021. 2

[58] Mengyang Yuan, Bo Lang, and Fengnan Quan. Student-friendly knowledge distillation, 2023. 2

[59] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *BMVC*, 2016. 3, 5, 6, 7

[60] Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. 2021. 6

[61] Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In *CVPR*, 2022. 1, 2, 3, 5, 6

[62] Mingkai Zheng, Shan You, Lang Huang, Fei Wang, Chen Qian, and Chang Xu. Simmatch: Semi-supervised learning with similarity matching. In *CVPR*, 2022. 5, 6

[63] Yichen Zhu and Yi Wang. Student customized knowledge distillation: Bridging the gap between student and teacher. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5057–5066, 2021. 2, 7

[64] Yongchun Zhu, Fuzhen Zhuang, Jindong Wang, Guolin Ke, Jingwu Chen, Jiang Bian, Hui Xiong, and Qing He. Deep subdomain adaptation network for image classification. 2021. 5