Improving Knowledge Distillation via Regularizing Feature Direction and Norm

Yuzhu Wang[1], Lechao Cheng[2✉], Manni Duan[1], Yongheng Wang[1], Zunlei Feng[3], and Shu Kong[4,5,6]

[1]Zhejiang Lab [2]Hefei University of Technology [3]Zhejiang University
[4]University of Macau [5]Institute of Collaborative Innovation [6]Texas A&M University
https://github.com/WangYZ1608/Knowledge-Distillation-via-ND

Abstract. Knowledge distillation (KD) is a particular technique of model compression that exploits a large well-trained `teacher` neural network to train a small `student` network . Treating `teacher`'s feature as knowledge, prevailing methods train `student` by aligning its features with the `teacher`'s, e.g., by minimizing the KL-divergence or L2-distance between their (logits) features. While it is natural to assume that better feature alignment helps distill `teacher`'s knowledge, simply forcing this alignment does not directly contribute to the `student`'s performance, e.g., classification accuracy. For example, minimizing the L2 distance between the penultimate-layer features (used to compute logits for classification) does not necessarily help learn a better `student` classifier. We are motivated to regularize `student` features at the penultimate layer using `teacher` towards training a better `student` classifier. Specifically, we present a rather simple method that uses `teacher`'s class-mean features to align `student` features w.r.t their *direction*. Experiments show that this significantly improves KD performance. Moreover, we empirically find that `student` produces features that have notably smaller norms than `teacher`'s, motivating us to regularize `student` to produce large-norm features. Experiments show that doing so also yields better performance. Finally, we present a simple loss as our main technical contribution that regularizes `student` by simultaneously (1) aligning the *direction* of its features with the `teacher` class-mean feature, and (2) encouraging it to produce large-*norm* features. Experiments on standard benchmarks demonstrate that adopting our technique remarkably improves existing KD methods, achieving the state-of-the-art KD performance through the lens of image classification (on ImageNet and CIFAR100 datasets) and object detection (on the COCO dataset).

Keywords: knowledge distillation · large-norm · feature direction

1 Introduction

Knowledge distillation (KD) is a specific technique for model compression that aims to train a smaller model (called `student`) by distilling knowledge learned

✉ Corresponding author: chenglc@hfut.edu.cn

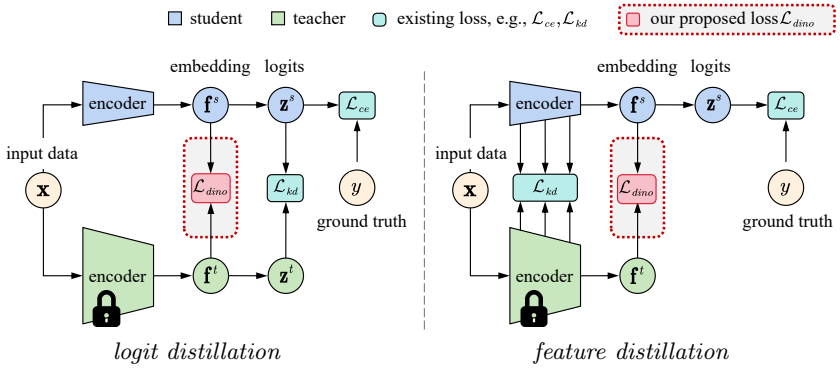


Fig. 1: Our main contribution is a simple loss, termed $\mathcal{L}_{dino}$, that regularizes the ***di***rection and ***no***rm of the `student` features (details in Sec. 3.3). $\mathcal{L}_{dino}$ is applicable to different KD methods which can be categorized into two types in the context of classification: (left) logit distillation that regularizes logits or softmax scores (e.g., KD [15] and DKD [48]), and (right) feature distillation that regularizes features other than logits (e.g., ReviewKD [4]). In this work, we apply $\mathcal{L}_{dino}$ to the embedding feature particularly at the penultimate layer (before logits). Experiments show that learning with $\mathcal{L}_{dino}$ improves existing KD methods, achieving the state-of-the-art benchmarking results for image classification (Tab. 1) and object detection (Tab. 3).

by a larger `teacher` model [15]. Deploying the small `student` model reduces inference computation (e.g., running time and memory use) compared against the original large model. Different from other model compression methods such as pruning [44] and quantization [10], KD has the flexibility of using different architectures of the `student`, which is preferred by certain real-world applications.

Status quo. Treating `teacher` features as knowledge, KD distills such knowledge to train `student` by encouraging its features to be similar to the `teacher`'s. Through the lens of image classification, prevailing methods can be categorized into two types: logit distillation (Fig. 1-left), and feature distillation (Fig. 1-right). Logit distillation trains the `student` by minimizing the KL divergence between its logits and the `teacher`'s [15, 48]. It assumes that, if `student` can produce logits more similar to `teacher`'s, it should achieve better performance that approaches `teacher`'s performance. However, logit distillation considers only the logit layer but not other intermediate layers. To exploit such, feature distillation trains `student` by encouraging its intermediate-layer features to be similar to the `teacher`'s, e.g., by minimizing the L2 distance between their features [4, 46].

Motivation. Despite the promising results of logit distillation and feature distillation methods, forcing the `student` to produce similar logits or features to the `teacher`'s does not directly serve the final task, e.g., classification. For example, minimizing the L2 distance between the penultimate-layer features (used to compute logits for classification) does not necessarily help learn a better `student`-classifier. Rather, `student` features are better regularized by the `teacher` to facilitate learning a better `student` classifier. Therefore, we are

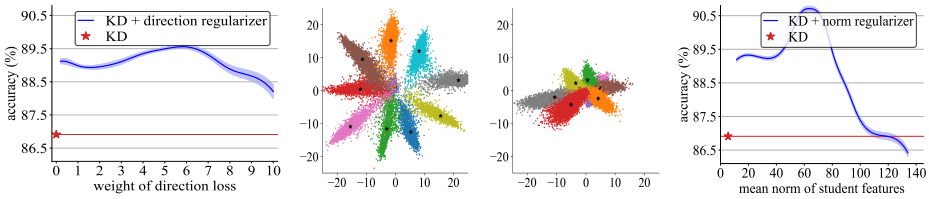
motivated to exploit `teacher`'s classifier to train better `student` towards better performance. Moreover, we empirically find that encouraging the `student` to produce large-norm features yields better performance (Fig. 2). Other lines of work such as domain adaptation [40] and pruning [44] also find the benefit of learning large-norm features, motivating us to train the `student` to produce large-norm features.

Contributions. We make three major contributions. First, we take a novel perspective to improve KD by regularizing `student` to produce features that (1) are aligned with class-means features computed by the `teacher`, and (2) have sufficiently large *norm*s. Second, we study multiple baseline methods to achieve such regularizations. We show that when incorporating either or both, existing KD methods yields better performance, e.g., classification accuracy and object detection precision by the `student`. Third, we propose a novel and simple loss that simultaneously regularizes feature **di**rection and **no**rm, termed *dino-loss*. Experiments demonstrate that additionally adopting our dino-loss helps existing KD methods achieve better performance. For example, on the standard benchmark ImageNet [5], applying dino-loss to KD [15] achieves 72.49% classification accuracy (Fig. 5), better than the original KD (71.35%), with ResNet-18 and ResNet-50 architectures for `student` and `teacher`, respectively. This outperforms recent methods ReviewKD [4] (71.09%) and DKD [48] (71.85%).

2 Related Work

Knowledge distillation. (KD) aims to train a small `student` model by distilling knowledge of a well-trained large `teacher` model. The knowledge is delivered by features produced by the `teacher` for training data. Therefore, the key to KD is to align `student` features to the `teacher`'s. The seminal KD method [15] propose to train `student` by aligning its logits with the `teacher`'s, i.e., minimizing the Kullback-Leibler divergence (KL) between logits. Other works improve KD by decoupling the KL loss into multiple terms [48] or consider logits rankings [16]. Distilling logit knowledge alone may not be sufficient as this does not exploit intermediate-layer features. Hence, feature distillation propose to align more features at other layers [1, 4, 14, 25, 26, 30, 34, 41, 45, 46]. Another line of research adopts network architecture search towards finding small `student` for KD [7, 19, 20], achieving the state-of-the-art at a cost of extra computation for architecture search. In this work, we take a different and orthogonal perspective to improve KD with simple methods, by encouraging the `student` to produce features to be aligned with the direction of `teacher` classifier and of large norms.

Constructing classifiers using off-the-shelf features. Off-the-shelf features extracted from a well-trained model can be used to construct strong classifiers. One simple classifier is to compute class-mean of training examples in the feature space, and uses such as the classifier [6, 17, 33]. On the other hand, recent literature of pretrained large models [28] shows that using off-the-shelf features and cosine similarity is a powerful classifier for zero-shot recognition. In this work, we propose to regularize `student` features using class-mean of `teacher`



(a) regularize feature direction **(b)** `teacher` feature **(c)** `student` feature **(d)** regularize feature norm

Fig. 2: Regularizing feature direction and norm helps knowledge distillation and improves `student`'s performance. We train a ResNet-56 `teacher` and use the classic KD method [15] to train a ResNet-8 `student` on the CIFAR10 dataset. **(a)** We propose to regularize `student` by aligning its feature direction with the class-mean features computed by `teacher`. To do so, we adopt a simple cosine loss term on the penultimate-layer features (Sec. 3.2). Results show that regularizing feature direction improves `student`'s performance. **(b-c)** We train `teacher` and `student` to produce 2D features at the penultimate layer from scratch (without using method KD [15]). We visualize them as 2D points, colored by class labels, and mark the class center by $\star$. Notably, the large `teacher` model produces large-norm features (b), while the small `student` model produces small-norm features (c). **(d)** This motivates us to regularize `student` by encouraging it to produce large-norm features (Sec. 3.2). To do so, we use the method SIFN [40]. Results show that properly regularizing feature norms improves `student`'s performance. Our technical contribution is a simple loss that simultaneously regularize `student` feature ***di***rection and ***no***rm, so we call it ***dino***-loss.

features, as well as other different methods for feature alignment. We hypothesize that doing so helps learn better `student` classifiers. Indeed, our experiments justify this hypothesis (Tab. 4).

Learning large-norm features. Multiple lines of work find it important to learn large-norm features or weight parameters. For example, domain adaptation [40] reveals that the erratic discrimination of the target domain mainly stems from its much smaller feature norms w.r.t that of the source domain, and adopting a larger-norm constraint helps adapt a pretrained model (in the source domain) to a new target domain. Moreover, model pruning finds that features with smaller norms play a less informative role during the inference [44], so it is safe to remove weight parameters that produce small-norm features without causing notable performance drop. In our work, we also empirically find that a small-capacity model produces features that tend to collide in the small-norm region (Fig. 2c). Therefore, we are motivated to train `student` to produce large-norm features, hypothesizing that doing so improves `student` performance. Our experiments justify this hypothesis (Fig. 2d, Tab. 4).

Feature regularization w.r.t norm and direction. The literature has multiple works studying feature regularization w.r.t its norm and direction in KD. We present them and explain how our method stands out. PKD [2] normalizes `student` and `teacher` features to have zero mean and unit variances to eliminate the gap in feature norms between them, and applies MSE loss between the normalized features. Our method encourages learning large-norm features of the

`student` and additionally regularizes features using `teacher` classifier for better classification. Guo [8] transforms `student` logits to have the same norm of the `teacher` logits and performs KL-divergence on the transformed logits, whereas we encourage learning large-norm `student` features which can have feature norm larger than the `teacher`'s (Fig. 4). FNKD [39] also advocates learning large-norm features but chooses to tune a temperature (shared by all classes) to enlarge logit values before softmax; differently, our method unifies norm regularization and feature direction regularization into a simple loss. Both SimKD [3] and SRRL [42] use MSE loss to align the `student` features with the `teacher`'s without enlarging feature norms; Wang [35] uses an L2 loss to align features of `teacher` and `student` models and further adopts a loss based on locality-sensitive hashing to encourage the direction of `student` features to be aligned with `teacher`'s. These methods do not necessarily learn a `student` to have feature norms that can be larger than the `teacher`'s. In sum, our simple dino-loss not only aligns directions of `student` and `teacher` features but also explicitly encourages learning `student` to produce large-norm features.

3 Improving Knowledge Distillation by Regularizing Feature Direction & Norm

We first describe notations and motivate our study of regularizing feature direction and norm to improve KD. Then, we introduce baselines, followed by our proposed *dino-loss*.

3.1 Notations and Background

Notations. Without losing generality, we think of a classification neural network as two modules: a feature extractor $f(\cdot;\Theta)$, and a classifier $g(\cdot;\mathbf{w})$, which are parameterized by Θ and $\mathbf{w}$, respectively. For the `teacher`, given input data $\mathbf{x}$, we denote its embedding feature as $\mathbf{f}^t = f^t(\mathbf{x};\Theta^t)$, and the logits as $\mathbf{z}^t = g^t(\mathbf{f}^t;\mathbf{w}^t)$. Similarly, the `student` outputs the embedding features for $\mathbf{x}$ as $\mathbf{f}^s = f^s(\mathbf{x};\Theta^s)$ and logits as $\mathbf{z}^s = g^s(\mathbf{f}^s;\mathbf{w}^s)$. We compute softmax scores in a vector $\mathbf{q}^t = \text{softmax}(\mathbf{z}^t;\tau)$, where τ is a temperature (default value as 1). Given N training examples from C classes, $\mathbf{x}_i$ and its label y_i (where $i = 1,\ldots,N$), one can train a classification model (e.g., the `teacher`) by minimizing the cross-entropy (CE) loss $\mathcal{L}_{ce}$ on all the training data.

Logit distillation trains the `student` by transferring the `teacher` knowledge using both the CE loss $\mathcal{L}_{ce}$ and a KD loss $\mathcal{L}_{kd}$. The seminal work of KD [15] uses KL divergence as the KD loss $\mathcal{L}_{kd}$, i.e., $\mathcal{L}_{kd} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{KL}(\mathbf{q}_i^t, \mathbf{q}_i^s)$.

Feature distillation distills `teacher` knowledge by minimizing the difference of intermediate features at more layers other than the logits [4, 45, 46]. A typical loss term is the L2 distance $\mathcal{L}_2$ between `student` and `teacher` features.[1] For

[1] When `student` and `teacher` have different features dimensions, one can learn extra layers in `student` to project its features to the same dimension as `teacher`'s [4].

example, over the embedding features at the penultimate layer (before logits), it applies the L2 loss $\mathcal{L}_{kd} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_2(\mathbf{f}_i^s, \mathbf{f}_i^t)$ in addition to the CE loss $\mathcal{L}_{ce}$.

The final loss is $\mathcal{L} = \mathcal{L}_{ce} + \alpha\mathcal{L}_{kd}$, where α controls the significance of the KD loss $\mathcal{L}_{kd}$ depending on distillation choice: logit distillation or feature distillation.

3.2 Baseline Methods

Recall that we are motivated to regularize `student` features during training: aligning their direction with `teacher` class-mean features, and encouraging them to be large in norm. In the main paper, we focus on the embedding features $\mathbf{f}^s$ at the penultimate layer, which are the direct input to a classifier; we also validate its effectiveness at other layers (Tab. 5). We compute the class-mean of the k^{th} class as $\mathbf{c}_k = \frac{1}{|\mathcal{I}_k|}\sum_{j\in\mathcal{I}_k}\mathbf{f}_j^t$, where $\mathcal{I}_k$ is the set of indices of training examples belonging to class-k. We now introduce simple techniques to regularize `student` features using $\mathbf{c}_k$ w.r.t feature direction and norm.

Feature Direction Regularization. We present two simple baseline methods below to regularize `student` w.r.t feature direction.

Baseline-1: cosine similarity. We use a simple cosine similarity based loss term to regularize the feature direction of $\mathbf{f}_i^s$ according to the mean feature $\mathbf{c}_k$ of the corresponding class-k:

$$\mathcal{L}_d = \frac{1}{C}\sum_{k=1}^{C}\frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k}(1-\cos(\mathbf{f}_i^s, \mathbf{c}_k)) \tag{1}$$

Baseline-2: InfoNCE. Using the cosine similarity loss Eq. 1 considers only paired examples and their corresponding class-mean. Inspired by InfoNCE [24], we also consider inter-class examples and class-means. Therefore, we train `student` by minimizing:

$$\mathcal{L}_d = \frac{1}{C}\sum_{k=1}^{C}\frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k} -\log\frac{\exp\left(\cos(\mathbf{f}_i^s, \mathbf{c}_k)\right)}{\sum_{j=1}^{C}\exp\left(\cos(\mathbf{f}_i^s, \mathbf{c}_j)\right)} \tag{2}$$

Feature Norm Regularization. We present two baseline methods below to regularize `student` towards producing large-norm features.

Baseline-1: $\mathcal{L}_2$ distance. As shown by Fig. 2c, the small-capacity `student` model produces features that have notably smaller norm than the `teacher`'s. To train the `student` to produce larger-norm features, perhaps a naive method is to increase `student` feature norm towards `teacher`'s. To this end, we minimize the L2 distance between features of `student` and `teacher`:

$$\mathcal{L}_n = \frac{1}{C}\sum_{k=1}^{C}\frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k}\|\mathbf{f}_i^s - \mathbf{f}_i^t\|_2^2 \tag{3}$$

Minimizing Eq. 3 is a common practice in feature distillation [1, 4, 14]. It implicitly trains `student` to produce features with norms approaching the corresponding larger-norm `teacher` features.

Baseline-2: Stepwise increasing feature norms (SIFN). We now describe a loss to explicitly increase the norm of the `student` features. Inspired by [40], we gradually increase the feature norm by minimizing:

$$\mathcal{L}_n = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_2\left(f^s(\mathbf{x}_i;\Theta^s_{previous}) + r, f^s(\mathbf{x}_i;\Theta^s_{current})\right), \tag{4}$$

where $\Theta^s_{previous}$ and $\Theta^s_{current}$ are parameters of an early checkpoint and the current model being optimized, respectively; r is a step size to increase the norm of `student` features during training.

3.3 The Proposed dino Loss

We now present the proposed dino-loss that simultaneously achieves the above two goals: (1) aligning `student` features to `teacher`'s w.r.t direction, and (2) learning large-norm `student` features. For simplicity, we drop the subscript (i.e., the index of a training example or class ID). Let $\mathbf{f}^s$ and $\mathbf{f}^t$ be the embedding features of an input $\mathbf{x}$ computed by `student` and `teacher`, respectively. Based on $\mathbf{x}$'s ground-truth label y, we have its corresponding class-mean $\mathbf{c}$. We denote the unit vector $\mathbf{e} = \mathbf{c}/\|\mathbf{c}\|_2$, and $\mathbf{p}^t = \mathbf{e}\|\mathbf{f}^t\|_2$. We compute the projection of $\mathbf{f}^s$ along $\mathbf{c}$'s direction: $\mathbf{p}^s = \mathbf{e}\|\mathbf{f}^s\|_2\cos(\mathbf{f}^s, \mathbf{c})$. Fig. 3 shows the geometric meaning.

When the norm of $\mathbf{f}^s$ is small, or its projection $\mathbf{p}^s$ has small norm, i.e., $\|\mathbf{p}^s\|_2 < \|\mathbf{f}^t\|_2$, we encourage the `student` to output larger-norm features and align them with the `teacher` class-mean by minimizing $\|\mathbf{p}^t - \mathbf{p}^s\|_2$.

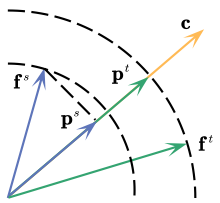


Fig. 3: Illustration of notations used in our dino-loss loss and its geometric meaning.

Because the feature norms of different examples can vary by an order of magnitude (see Fig. 2c), naively learning with the above can produce artificially large gradients from specific training data and negatively affect training. Thus, we divide the above by $\|\mathbf{f}^t\|_2$, which is equivalent to $\|\mathbf{p}^t\|_2$:

$$\mathcal{L}_{dino} = \frac{\|\mathbf{p}^t - \mathbf{p}^s\|_2}{\|\mathbf{f}^t\|_2} = \frac{\|\mathbf{p}^t\|_2 - \|\mathbf{p}^s\|_2}{\|\mathbf{f}^t\|_2} = 1 - \frac{\mathbf{f}^s\cdot\mathbf{e}}{\|\mathbf{f}^t\|_2} \tag{5}$$

Minimizing Eq. 5 amounts to simultaneously (1) increasing the norm of $\mathbf{f}^s$ and (2) reducing the angular distance between $\mathbf{f}^s$ and the class-mean $\mathbf{c}$.

When $\mathbf{f}^s$ has large norm, i.e., $\|\mathbf{f}^s\|_2 \geq \|\mathbf{f}^t\|_2$, we only minimize the angular distance between `student` feature and the class-mean defined by the `teacher`:

$$\mathcal{L}_{dino} = 1 - \frac{\mathbf{f}^s\cdot\mathbf{e}}{\|\mathbf{f}^s\|_2} \tag{6}$$

The above loss means that the feature norm of `student` is no longer explicitly required to reach a larger value if it is already large enough; yet it is still allowed

to increase freely during training. We merge Eq. 5 and 6 and average over all training examples as our dino-loss (dropping the constant 1):

$$\mathcal{L}_{dino} = -\frac{1}{C}\sum_{k=1}^{C}\frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k}\frac{\mathbf{f}_i^s\cdot\mathbf{e}_k}{\max\{\|\mathbf{f}_i^s\|_2,\|\mathbf{f}_i^t\|_2\}} \tag{7}$$

Compatible with existing KD methods, our dino-loss $\mathcal{L}_{dino}$ can be used altogether with CE loss $\mathcal{L}_{ce}$ and KD loss $\mathcal{L}_{kd}$ to train `student`:

$$\mathcal{L} = \mathcal{L}_{ce} + \alpha\mathcal{L}_{kd} + \beta\mathcal{L}_{dino} \tag{8}$$

α and β are the weights for $\mathcal{L}_{kd}$ and $\mathcal{L}_{dino}$, respectively. The definition of $\mathcal{L}_{kd}$ depends on the distillation method. Otherwise stated, we study $\mathcal{L}_{dino}$ with the seminal logit distillation method KD [15], so $\mathcal{L}_{kd} = \mathbf{KL}$.

Remark. The dino-loss simultaneously encourages `student` to output large-norm features (possibly larger than `teacher`'s, as shown in Fig. 4), and minimizes the angular distance between `student` features and the class-mean defined by the `teacher` during training. This is a desired property in terms of training the `student` to achieve better classification accuracy.

4 Experiments

4.1 Datasets and Settings

For fair comparisons, our implementation adheres to the previous methodologies outlined in [4, 16, 34, 48].

CIFAR-100 [18] contains 50k training images and 10k testing images. During training, we adopt random cropping and random left-right fliping augmentations. We randomly initialize weight parameters following [12] and train all `student` networks from scratch. The `teacher` models are publicly available that are adopted in prior art [34]. The `student` networks are trained using a batch size of 128 for 240 epochs (with a linear warm-up for the first 20 epochs), employing SGD with a weight decay of 5e-4 and momentum of 0.9. We set the initial learning rate of 0.1 for ResNet [13] and WRN [47] backbones, and 0.02 for MobileNet [32] and ShuffleNet [23] backbones, decaying it with a factor of 10 at 150th, 180th, and 210th epochs. The temperature is empirically set to 4.

ImageNet [31] has 1.28 million training images and 50,000 validation images spanning by 1,000 categories. We employ SGD with a batch size of 512 for a total of 100 epochs (with a linear warm-up for the first 5 epochs). The initial learning rate is set to 0.2 and is reduced by a factor of 10 every 30 epochs. Besides, the weight decay and momentum are set to 1e-4 and 0.9, respectively. The pre-trained weights for `teacher` come from PyTorch[2] and TIMM [38] for fair comparisons. The temperature for knowledge distillation is set to 1.

COCO 2017 [22] consists of 80 object categories with 118k training images and 5k validation images. We utilize Faster R-CNN [29] with FPN [21] as the

[2] https://pytorch.org/vision/stable/models.html

Table 1: Benchmarking results on CIFAR100. Methods are reported w.r.t top-1 accuracy (%) on the test set. ++ means that we apply the proposed dino-loss to existing methods. Clearly, doing so improves performance over the original KD methods and outperforms prior KD methods. We mark performance gains over corresponding methods using green superscripts.

	Homogeneous architectures			Heterogeneous architectures		
Methods	ResNet-56 ResNet-20	wide-ResNet-40-2 wide-ResNet-40-1	ResNet-32×4 ResNet-8×4	ResNet-50 MobileNet-V2	ResNet-32×4 ShuffleNet-V1	ResNet-32×4 ShuffleNet-V2
`teacher` (T)	72.34	75.61	79.42	79.34	79.42	79.42
`student` (S)	69.06	71.98	72.50	64.60	70.50	71.82
Feature distillation methods						
FitNet [30]	69.21	72.24	73.50	63.16	73.59	73.54
RKD [25]	69.61	72.22	71.90	64.43	72.28	73.21
PKT [27]	70.34	73.45	73.64	66.52	74.10	74.69
OFD [14]	70.98	74.33	74.95	69.04	75.98	76.82
CRD [34]	71.16	74.14	75.51	69.11	75.11	75.65
ReviewKD [4]	71.89	75.09	75.63	69.89	77.45	77.78
ReviewKD++	72.05$^{+0.16}$	**75.66**$^{+0.57}$	76.07$^{+0.44}$	70.45$^{+0.56}$	**77.68**$^{+0.23}$	**77.93**$^{+0.15}$
Logit distillation methods						
KD [15]	70.66	73.54	73.33	67.65	74.07	74.45
KD++	**72.53**$^{+1.87}$	74.59$^{+1.05}$	75.54$^{+2.21}$	70.10$^{+2.35}$	75.45$^{+1.38}$	76.42$^{+1.97}$
DIST [16]	71.78	74.42	75.79	69.17	75.23	76.08
DIST++	72.52$^{+0.74}$	75.00$^{+0.58}$	76.13$^{+0.34}$	69.80$^{+0.63}$	75.60$^{+0.37}$	76.64$^{+0.56}$
DKD [48]	71.97	74.81	75.44	70.35	76.45	77.07
DKD++	72.16$^{+0.19}$	75.02$^{+0.21}$	**76.28**$^{+0.84}$	**70.82**$^{+0.47}$	77.11$^{+0.66}$	77.49$^{+0.42}$

feature extractor, and employ the dino-loss on the R-CNN head, wherein both `teacher` and `student` models adopt ResNet [13] as the backbone. In addition, MobileNet-V2 [32] is used as a heterogeneous `student` model. All `student` models are trained with 1x scheduler, following the public toolbox Detectron2.[3]

4.2 Comparisons with State-of-the-art Results

CIFAR-100 classification. Tab. 1 shows the KD performances on the CIFAR-100 dataset. In this context, spanning homogeneous and heterogeneous architectures, we undertake an extensive assessment over prominent *feature distillation methods* and *logits distillation methods*. The ++ signifies the integration of our novel dino-loss into the existing methods. A salient conclusion from Tab. 1 is that our proposed dino-loss empowers existing KD methods to achieve better classification accuracy. This suggests the effectiveness and compatibility of the dino-loss, which can serve as a plugin for existing KD methods, irrespective of the homogeneity or heterogeneity for network architectures.

ImageNet classification. We demonstrate the effectiveness of the proposed dino-loss on the well-established benchmark ImageNet. Tab. 2 lists detailed results. It is worth noting that with our simple dino-loss, the method **KD++** already outperforms most of previous methods (e.g., ReviewKD [4], DKD [48], and SRRL [43]). Moreover, incorporating our dino-loss, other methods such as ReviewKD++ and DKD++ also achieve notable improvements.

[3] https://github.com/facebookresearch/detectron2

Table 2: Comparisons with state-of-the-art methods on the **ImageNet validation set**. KD++ serves as a stronger baseline that introduces no more parameters and negligible training time over the vanilla KD [15] and outperforms most of previous methods (e.g., ReviewKD [4], SRRL [42], and MGD [43]). Methods marked by † adopts a searching strategy, demanding significant more computation cost to achieve the state-of-the-art performance. In contrast, our dino-loss is rather simple that helps existing KD methods to rival search-based methods.

	ResNet-34 → ResNet-18		ResNet-50 → MobileNet-V1	
Methods	Top-1	Top-5	Top-1	Top-5
`student`	69.76	89.43	68.87	88.76
`teacher`	73.31	91.59	76.16	92.86
CRD ICLR2020 [34]	71.17	90.13	71.37	90.42
SRRL ICLR2021 [42]	71.73	90.60	72.49	90.92
MGD ECCV2022 [43]	71.58	90.35	72.35	91.00
CAT-KD CVPR2023 [9]	71.26	90.45	72.24	91.13
KD NeurIPS2015 [15]	70.66	89.88	70.50	89.80
KD++	71.98 (+1.32)	90.53 (+0.65)	72.77 (+2.27)	91.14 (+1.34)
ReviewKD CVPR2021 [4]	71.62	90.51	72.56	91.00
ReviewKD++	71.64 (+0.02)	**90.61** (+0.10)	**72.96** (+0.40)	**91.16** (+0.16)
DKD CVPR2022 [48]	71.70	90.41	72.05	91.05
DKD++	**72.07** (+0.37)	90.59 (+0.18)	72.63 (+0.58)	91.07 (+0.02)
KD-Zero NeurIPS2023 [19]†	72.17	90.46	73.02	91.05
Auto-KD ICCV2023 [20]†	72.45	90.69	73.26	91.17

Table 3: Benchmarking KD methods for object detection on the **COCO** `val2017` dataset (mAP in %). All methods use the Faster R-CNN detector with various `teacher` and `student` architectures, including RestNet {18,50,101} amd MobileNet-V2 (MV2). Incorporating our dino-loss, simple methods KD++ and ReviewKD++ obtain notable performance gains (marks in green) over their original counterparts, achieving the state-of-the-art performance.

	R101→R18			R101→R50			R50→MV2		
Methods	mAP	AP^{50}	AP^{75}	mAP	AP^{50}	AP^{75}	mAP	AP^{50}	AP^{75}
`teacher`	42.04	62.48	45.88	42.04	62.48	45.88	40.22	61.02	43.81
`student`	33.26	53.61	35.26	37.93	58.84	41.05	29.47	48.87	30.90
KD [15]	33.97	54.66	36.62	38.35	59.41	41.71	30.13	50.28	31.35
FitNet [30]	34.13	54.16	36.71	38.76	59.62	41.80	30.20	49.80	31.69
FGFI [36]	35.44	55.51	38.17	39.44	60.27	43.04	31.16	50.68	32.92
DKD [48]	35.05	56.60	37.54	39.25	60.90	42.73	32.34	53.77	34.01
ReviewKD [4]	36.75	56.72	34.00	40.36	60.97	44.08	33.71	53.15	36.13
KD++	36.12 (+2.15)	56.81 (+2.15)	37.64 (+1.02)	39.86 (+1.51)	61.07 (+1.66)	43.57 (+1.86)	33.26 (+3.13)	53.71 (+3.43)	34.85 (+3.50)
ReviewKD++	**37.43** (+0.68)	**57.96** (+1.24)	**40.15** (+6.15)	**41.03** (+0.67)	**61.80** (+0.83)	**44.94** (+0.86)	**34.51** (+0.80)	**55.18** (+2.03)	**37.21** (+1.08)

We note that recent KD methods adopt a searching strategy for better results, e.g., KD-Zero [19] and Auto-KD [20]. These methods search for a distiller in a space of predefined modules, e.g., normalization methods, losses, transform operations, activation functions, etc. At the cost of significantly more computation time for searching, they achieve the best performance on the benchmark, slightly better than ours. Nevertheless, our dino-loss helps existing much simpler methods rival search-based approaches! Compared against search-based methods, applying our dino-loss, methods such as KD++ introduce no more parameters and negligible training time over the counterparts (e.g., vanilla KD [15]). This makes the use of our dino-loss a stronger baseline for knowledge distillation. Interestingly, current search-based methods do *not* consider our proposed regularization methods, i.e., feature norm regularizer and feature direction regularizer. That said, our method

Table 4: Analysis of feature direction and norm regularization. We train `teacher` (ResNet-50, ResNet-56) and `student` (MobileNet-V2, ResNet-20) models on the CIFAR100 dataset and report accuracy (%) on its test-set. We use KD [15] as the *baseline*, which is a logit distillation method. From (a-b), we see that applying either direction or norm regularization on `student` features improves KD as shown by the increased `student` accuracy. While combining both outperforms *baseline* (c), using dino-loss achieves the best (d).

(a) Regularizing feature direction only. *Cosine* is more robust than *InfoNCE*.

case	R50→MV2	R56→R20
baseline	67.65	70.66
cosine	69.18 (+1.53)	71.75 (+1.09)
InfoNCE	69.06 (+1.41)	70.73 (+0.07)

(b) Regularizing feature norm only. *SIFN* is more robust than $\mathcal{L}_2$.

case	R50→MV2	R56→R20
baseline	67.65	70.66
$\mathcal{L}_2$	69.05 (+1.40)	71.85 (+1.19)
SIFN	69.32 (+1.67)	72.13 (+1.47)

(c) Regularizing both feature norm and direction. Simple addition does not yield better results (e.g., cosine + $\mathcal{L}_2$ *vs.* cosine *vs.* $\mathcal{L}_2$).

case	R50→MV2	R56→R20
cosine + $\mathcal{L}_2$	68.62 (+0.97)	71.58 (+0.92)
cosine + SIFN	69.07 (+1.42)	71.72 (+1.06)
InfoNCE + $\mathcal{L}_2$	68.47 (+0.82)	70.81 (+0.15)
InfoNCE + SIFN	68.71 (+1.06)	71.04 (+0.38)

(d) The proposed dino-loss can perfectly integrate the advantages of regularization feature norm and direction, and perform best.

case	R50→MV2	R56→R20
CE + KL (*baseline*)	67.65	70.66
CE + dino	68.78 (+1.13)	71.96 (+1.30)
KL + dino	68.68 (+1.03)	72.04 (+1.38)
CE + KL + dino	**70.10** (+2.45)	**72.53** (+1.87)

is orthogonal to existing KD methods and can potentially serve as modules for searching, suggesting the novelty of our work.

COCO Object Detection. We verify the efficacy of the proposed dino-loss for knowledge distillation through the lens of object detection on the COCO dataset. Tab. 3 lists detailed results compared to prior works. Specifically, the ReviewKD++ yields a significant improvement in performance, outperforming state-of-the-art results with a remarkable margin. Moreover, this experiment and the above demonstrate the applicability of our dino-loss across tasks (i.e., image classification and object detection).

4.3 Ablation Study

In this subsection, we first conduct an ablation study on feature norm and direction regularization with the CIFAR-100 dataset. Subsequently, we perform a visual analysis of the impact before and after applying dino-loss. Finally, we conduct intriguing experiments on ImageNet to validate the benefits of using our approach that encourages learning large-norm features of the `student` model.

The effectiveness feature direction and norm regularization. Recall the baseline methods in Sec. 3.2 and 3.2 for feature direction and norm regularization. We present results for $\mathcal{L}_2$ (Eq. 3) and SIFN (Eq. 4) on CIFAR-100 in Tab. 4b. Yet additional offline experiments substantiate that SIFN outperforms

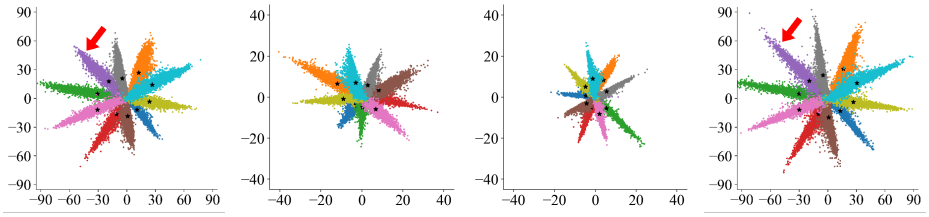
Table 5: Comparison of using `teacher`'s classifier weights (dubbed "w/ weights") versus per-class mean features (dubbed "w/ class-mean") in our dino-loss. We study them with the KD method [15] on CIFAR-100. Results show that using per-class mean features outperforms classifier weights. Moreover, we apply our dino-loss at more layers other than the penultimate layer (cf. the bottom row block). Here, layer-x means that the dino-loss is applied to all the x layers right before the penultimate layer. In particular, layer-0 means that the dino-loss is applied only at the penultimate layer (i.e., "w/ class-mean" in this table). Results show that applying dino-loss at other layers also leads to improvements but doing so at the penultimate layer achieves the best performance.

	Homogeneous architectures			Heterogeneous architectures		
	ResNet-56	wide-ResNet-40-2	ResNet-32×4	ResNet-50	ResNet-32×4	ResNet-32×4
Methods	ResNet-20	wide-ResNet-40-1	ResNet-8×4	MobileNet-V2	ShuffleNet-V1	ShuffleNet-V2
`teacher`	72.34	75.61	79.42	79.34	79.42	79.42
`student`	69.06	71.98	72.50	64.60	70.50	71.82
KD [15]	70.66	73.54	73.33	67.65	74.07	74.45
L2 of cls weights	70.54	73.61	73.76	66.81	73.62	74.13
w/ weights	71.73	73.97	75.06	69.76	75.24	75.61
w/ class-mean	**72.53**	**74.59**	**75.54**	**70.10**	**75.45**	**76.42**
apply dino-loss to other layers						
layer-1	71.30	73.74	74.23	69.44	75.01	76.26
layer-2	71.96	73.81	74.12	69.10	**75.72**	**76.53**
layer-3	71.19	73.61	73.92	68.87	74.93	76.10

$\mathcal{L}_2$ regularization in terms of performance and consistently affirm that large `student` feature norms encapsulate more `teacher` knowledge. Similarly, Tab. 4a shows superior gains of cosine similarity-based regularizer (Eq. 1) over InfoNCE (Eq. 2); both underscore the significance of feature direction regularization.

Our dino-loss yields better results. Tab. 4c shows that feature direction regularization (cosine, InfoNCE) and feature norm regularization ($\mathcal{L}_2$, SIFN) improve KD. While combining them helps KD (e.g., cosine + $\mathcal{L}_2$, cosine + SIFN), our dino-loss performs the best at 70.10% accuracy (Tab. 4d). This convincingly shows the superiority of our dino-loss over others.

Class-mean *vs.* classifier weights in regularizing `student` feature direction. We perform a quantitative analysis of using the `teacher`'s classifier weights and per-class feature mean to regularize `student`'s features. We adopt the classifier weights that are derived from `teacher` as centers in our dino-loss to train `student` models (dubbed "w/ weights"). In comparison, we utilize per-class feature means, denoted as "w/ class-mean" (which is our proposed method). The results in Tab. 5 clearly demonstrate that the utilization of per-class feature mean performs better than the method of using `teacher`'s classifier weights for feature direction regularization. Additionally, we have implemented an alternative approach that employs an L2 loss to guide the `student` to output classifier weights similar to those of the `teacher` (referred to as "L2 of cls weight"). However, this approach consistently underperforms our "w/ class-mean" method and even lags behind the baseline KD method [15] in most experimental settings. These findings provide strong evidence for the superiority of employing class-mean features over the `teacher`'s classifier weights.



(a) `teacher` **(b)** naive small model **(c)** `student` by KD **(d)** `student` by KD++

Fig. 4: Visualization of 2D embedding features. (a) Features computed by `teacher` (ResNet-50) are well separated at class label; note the purple class pointed by red arrow. **(b)** A small-capacity model (ResNet-18) fails to separate this purple class, which is occluded by others. **(c)** Even using KD [15] to train ResNet-18 `student` cannot reveal this purple class. **(d)** Using our dino-loss along with KD, i.e., KD++, achieves better separation of the points and reveals purple class. This attributes to the feature direction regularization using `teacher` class-means. Moreover, `student` features in (d) have larger-norms than the `teacher` in (a) because our dino-loss explicitly encourages learning large-norm featurs of the `student`.

Applying the dino-loss to other layers. In the above experiment, we apply the dino-loss to the penultimate layer only. Here, we study applying it to other layers and evaluate on CIFAR-100 in bottom of Tab. 5. Results show that our dino-loss applied at other layers also improves over the baseline KD but performs the best when applied at the penultimate layer.

KD++ is a strong baseline. Tab. 4d illustrates the impacts of different losses in KD. By incorporating dino-loss into conventional KD framework [15], KD++ (i.e., CE+KL+dino) achieves a significant improvement: KD (67.65%)→ KD++ (70.10%). Further, combining dino-loss with CE or KL can also boost the accuracy by ∼1% compared to the original KD method. It is worth noting that KD++ introduces no additional parameters and negligible computation overhead, making it a strong baseline for KD (cf. more results in Tab. 1, 2, & 3).

In addition, following [37], we visually examine the `student` feature in Fig. 4. First, as indicated in Fig. 4d, KD++ demonstrates notably amplified feature norms, surpassing even those of the `teacher` depicted in Fig. 4a. Furthermore, the direction in KD++ align well with the `teacher` (Fig. 4a *vs.* Fig. 4d, thereby maintaining consistent relative margins among categories. Another observation is that both the naive `student` (Fig. 4b) and the conventional KD (Fig. 4c) exhibit direct failures in classifying the purple category, whereas our approach, KD++ (Fig. 4d), effectively learns `student` features that can discriminate all classes.

The benefit from larger teacher models. Now we investigate whether our approach exhibits monotonically incremental performance gains when exploiting a larger `teacher`. We first examine the classic KD [15] framework, and two other recently proposed methods, namely ReviewKD [4] and DKD [48].

As shown in Fig. 5, it is clear that for KD [15], DKD [48] and ReviewKD [4] show a degradation or fluctuation when scaling up the `teacher` from ResNet-34 to ResNet-152, with ResNet-18 as `student`. The main obstacle is the huge capacity

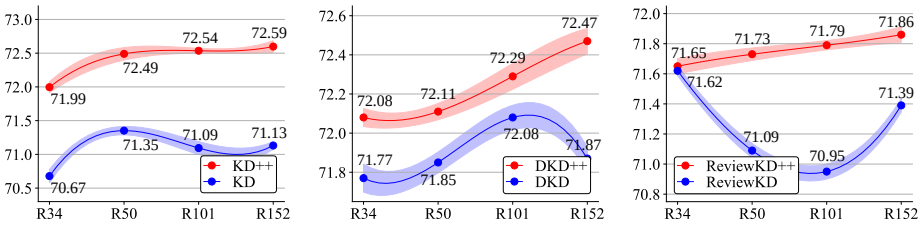


Fig. 5: Using our dino-loss can benefit from larger `teacher` models. The `student` is ResNet-18, with scaling up the `teacher` from ResNet-34 to ResNet-152 (denoted in the x-axis). We report the top-1 accuracy (%) on the ImageNet validation set. All results are the average accuracy over five runs. With the `teacher` capacity increasing, our methods, KD++, DKD++ and ReviewKD++ (**red**) achieve better knowledge distillation results, even though the original distillation methods (**blue**) suffer from performance degradation with large `teacher` models (e.g., ResNet-152).

Table 6: A larger `student` gets better knowledge distillation performance. The `teacher` is ResNet-152 (top-1 acc, 78.31%). We report the top-1 accuracy (%) on the ImageNet validation set. Model marked with † adopts a stronger recipe [11]. Results show that our dino-loss achieves consistent improvement, outperforming the model trained from scratch (cf. the first row).

`student`	ResNet-18	ResNet-34	ResNet-50	ResNet-101	ViT-S	ViT-B	ViT-B †
trained from scratch	69.76	73.31	76.16	77.37	74.64	78.00	81.80
KD [15]	70.66	74.84	76.93	78.04	75.77	78.51	82.39
KD++	**71.98** (+1.32)	**75.53** (+0.69)	**77.48** (+0.55)	**79.15** (+1.11)	**76.53** (+0.76)	**79.85** (+1.34)	**82.92** (+0.53)

gap between `student` and `teacher` [1, 16, 34]. Surprisingly, upon incorporating our dino-loss, distillation methods yield significant improvements compared to the original methods (e.g., KD++ *vs*. KD, ReviewKD++ *vs*. ReviewKD). More importantly, they show a consistent improvement with scaling up the `teacher` size (e.g., **KD++**: 71.99% → 72.49% → 72.54% → 72.59%).

The benefit from larger student models. We use KD++ to study the effect of increasing the size of the `student` for knowledge distillation, and set the `teacher` as ResNet-152. The results in Tab. 6 demonstrate that increasing the capacity of the `student` significantly improves distillation results.

5 Discussion and Conclusion

Broader Impacts As our work falls in the area of knowledge distillation, we do not see any new potential societal impacts other than those already known, e.g., `student` models might learn bias and unfairness delievered by the `teacher`.

Conclusion. We study feature regularization w.r.t norm and direction when training `student` models for better knowledge distillation (KD). Indeed, sufficient experiments and ablation studies demonstrate that doing so with our proposed dino-loss helps existing KD methods to achieve better performance.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was supported by the National Natural Science Foundation of China (U22A2032), by the Zhejiang Provincial Science and Technology Plan Project (2024SSYS0013), and in part by the National Natural Science Foundation of China under Grant No. (62106235). Shu Kong acknowledges the support by the University of Macau (SRG2023-00044-FST).

References

1. Beyer, L., Zhai, X., Royer, A., Markeeva, L., Anil, R., Kolesnikov, A.: Knowledge distillation: A good teacher is patient and consistent. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10925–10934 (2022)
2. Cao, W., Zhang, Y., Gao, J., Cheng, A., Cheng, K., Cheng, J.: Pkd: General distillation framework for object detectors via pearson correlation coefficient. Advances in Neural Information Processing Systems **35**, 15394–15406 (2022)
3. Chen, D., Mei, J.P., Zhang, H., Wang, C., Feng, Y., Chen, C.: Knowledge distillation with the reused teacher classifier. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11933–11942 (2022)
4. Chen, P., Liu, S., Zhao, H., Jia, J.: Distilling knowledge via knowledge review. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5008–5017 (2021)
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009)
6. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. In: International conference on machine learning. pp. 647–655. PMLR (2014)
7. Dong, P., Li, L., Wei, Z.: Diswot: Student architecture search for distillation without training. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11898–11908 (2023)
8. Guo, J., Chen, M., Zhao, Y., Pan, B., Zhu, C., Hu, Y., Wang, H., He, X., Cai, D.: Reducing the capacity gap via spherical knowledge distillation (2022)
9. Guo, Z., Yan, H., Li, H., Lin, X.: Class attention transfer based knowledge distillation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11868–11877 (June 2023)
10. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149 (2015)
11. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16000–16009 (2022)
12. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)

14. Heo, B., Kim, J., Yun, S., Park, H., Kwak, N., Choi, J.Y.: A comprehensive overhaul of feature distillation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1921–1930 (2019)
15. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
16. Huang, T., You, S., Wang, F., Qian, C., Xu, C.: Knowledge distillation from a stronger teacher. arXiv preprint arXiv:2205.10536 (2022)
17. Kong, S., Ramanan, D.: Opengan: Open-set recognition via open data generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 813–822 (2021)
18. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
19. Li, L., Dong, P., Li, A., Wei, Z., Ya, Y.: Kd-zero: Evolving knowledge distiller for any teacher-student pairs. In: Thirty-seventh Conference on Neural Information Processing Systems (2023)
20. Li, L., Dong, P., Wei, Z., Yang, Y.: Automated knowledge distillation via monte carlo tree search. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 17413–17424 (October 2023)
21. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117–2125 (2017)
22. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13. pp. 740–755. Springer (2014)
23. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the European conference on computer vision (ECCV). pp. 116–131 (2018)
24. Oord, A.v.d., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 (2018)
25. Park, W., Kim, D., Lu, Y., Cho, M.: Relational knowledge distillation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3967–3976 (2019)
26. Passalis, N., Tefas, A.: Learning deep representations with probabilistic knowledge transfer. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 268–284 (2018)
27. Passalis, N., Tefas, A.: Probabilistic knowledge transfer for deep representation learning. CoRR, abs/1803.10837 **1**(2), 5 (2018)
28. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)
29. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems **28** (2015)
30. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: Fitnets: Hints for thin deep nets. arXiv preprint arXiv:1412.6550 (2014)
31. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**, 211–252 (2015)

32. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4510–4520 (2018)
33. Sharif Razavian, A., Azizpour, H., Sullivan, J., Carlsson, S.: Cnn features off-the-shelf: an astounding baseline for recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. pp. 806–813 (2014)
34. Tian, Y., Krishnan, D., Isola, P.: Contrastive representation distillation. arXiv preprint arXiv:1910.10699 (2019)
35. Wang, G.H., Ge, Y., Wu, J.: Distilling knowledge by mimicking features. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(11), 8183–8195 (2021)
36. Wang, T., Yuan, L., Zhang, X., Feng, J.: Distilling object detectors with fine-grained feature imitation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4933–4942 (2019)
37. Wen, Y., Zhang, K., Li, Z., Qiao, Y.: A discriminative feature learning approach for deep face recognition. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14. pp. 499–515. Springer (2016)
38. Wightman, R.: Pytorch image models. `https://github.com/rwightman/pytorch-image-models` (2019). `https://doi.org/10.5281/zenodo.4414861`
39. Xu, K., Rui, L., Li, Y., Gu, L.: Feature normalized knowledge distillation for image classification. In: European conference on computer vision. pp. 664–680. Springer (2020)
40. Xu, R., Li, G., Yang, J., Lin, L.: Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1426–1435 (2019)
41. Yang, C., Zhou, H., An, Z., Jiang, X., Xu, Y., Zhang, Q.: Cross-image relational knowledge distillation for semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12319–12328 (2022)
42. Yang, J., Martinez, B., Bulat, A., Tzimiropoulos, G., et al.: Knowledge distillation via softmax regression representation learning. International Conference on Learning Representations (ICLR) (2021)
43. Yang, Z., Li, Z., Shao, M., Shi, D., Yuan, Z., Yuan, C.: Masked generative distillation. In: European Conference on Computer Vision. pp. 53–69. Springer (2022)
44. Ye, J., Lu, X., Lin, Z., Wang, J.Z.: Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. arXiv preprint arXiv:1802.00124 (2018)
45. Yim, J., Joo, D., Bae, J., Kim, J.: A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4133–4141 (2017)
46. Zagoruyko, S., Komodakis, N.: Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. arXiv preprint arXiv:1612.03928 (2016)
47. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)
48. Zhao, B., Cui, Q., Song, R., Qiu, Y., Liang, J.: Decoupled knowledge distillation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11953–11962 (2022)