Distilling Knowledge by Mimicking Features

Guo-Hua Wang, Yifan Ge, and Jianxin Wu, *Member, IEEE*

Abstract—Knowledge distillation (KD) is a popular method to train efficient networks ("student") with the help of high-capacity networks ("teacher"). Traditional methods use the teacher's soft logits as extra supervision to train the student network. In this paper, we argue that it is more advantageous to make the student mimic the teacher's features in the penultimate layer. Not only the student can directly learn more effective information from the teacher feature, feature mimicking can also be applied for teachers trained without a softmax layer. Experiments show that it can achieve higher accuracy than traditional KD. To further facilitate feature mimicking, we decompose a feature vector into the magnitude and the direction. We argue that the teacher should give more freedom to the student feature's magnitude, and let the student pay more attention on mimicking the feature direction. To meet this requirement, we propose a loss term based on locality-sensitive hashing (LSH). With the help of this new loss, our method indeed mimics feature directions more accurately, relaxes constraints on feature magnitudes, and achieves state-of-the-art distillation accuracy. We provide theoretical analyses of how LSH facilitates feature direction mimicking, and further extend feature mimicking to multi-label recognition and object detection.

Index Terms—Convolutional neural networks, deep learning, knowledge distillation, image classification, object detection

1 Introduction

Recently, deep learning has achieved remarkable success in many visual recognition tasks. To deploy deep networks in devices with limited resources, more and more efficient networks have been proposed [1], [2]. Knowledge distillation (KD) [3] is a popular method to train these efficient networks (named "student") with the help of high-capacity networks (named "teacher").

Initial study of KD [3] used the softmax output of the teacher network as the extra supervisory information for training the student network. However, the output of a high-capacity network is not significantly different from ground-truth labels. And, due to the existence of the classifier layer, the softmax output contains less information compared with the representation in the penultimate layer. These issues hinder the performance of a student model. In addition, it is difficult for KD to distill teacher models trained by unsupervised or self-supervised learning [4], [5], [6], [7].

Feature distillation has received more and more attention in recent years [8], [9], [10], [11]. However, previous works only focused on distilling features in the middle layers [8] or transforming the features [9]. Few have addressed the problem of making the student directly mimic the teacher's feature in the penultimate layer. Distilling features in the middle layers suffers from the different architectures between teacher and student, while transforming the features may lose some information in the teacher. We believe it is a better way to *directly mimic the feature for knowledge distillation, in which we only mimic the feature in the penultimate layer*. Compared with KD, it does not need the student model to learn a classifier from the teacher. Feature mimicking can be applied to a teacher trained by unsupervised, metric or self-supervised learning, and can be easily used when the teacher and student have different architectures. Furthermore, if the student features are the same as the teacher's, the classification accuracy will surely be the same, too.

Some reasons may explain why feature mimicking has not yet been popular in the literature. First, previous work used the mean squared loss (ℓ_2 loss) to distill features. In this paper, we decompose a feature vector into the magnitude and the direction. The ℓ_2 loss focuses on both magnitude and direction. But due to the different capacities, the student cannot mimic the teacher in its entirety. In fact, only the direction affects the classification result while the magnitude mainly represents the confidence of prediction [12]. We find that different networks often have different feature magnitudes (cf. Table 2). That inspires us to give more freedom to the student feature's magnitude. One possible approach to tackle this problem is to distill the feature after ℓ_2-normalization [9]. However, it will lose all magnitude information about the teacher feature and make the optimization difficult [13]. In this paper, we propose a loss term which focuses on the feature direction *and gives more freedom to its magnitude*, which alleviates the shortcomings of the ℓ_2 loss (cf. Fig. 3).

Second, when teacher and student features have different dimensionalities, difficulty arises. To solve this problem, we split the final fully connected (FC) layer of the student network into two FC layers without non-linear activation in-between. The dimensionality of the first FC layer matches that of the teacher feature. The two FCs can be merged into one after training. Hence, no extra parameter or computation is added in the student's architecture during inference.

Third, even though the feature structure of the student is the same as that of the teacher, their feature space may

• *The authors are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China. E-mail: {wangguohua, geyf, wujx}@lamda.nju.edu.cn.*

Manuscript received 9 March 2021; revised 26 June 2021; accepted 8 August 2021. Date of publication 11 August 2021; date of current version 3 October 2022.
(Corresponding author: Jianxin Wu.)
Recommended for acceptance by T. Liu.
Digital Object Identifier no. 10.1109/TPAMI.2021.3103973

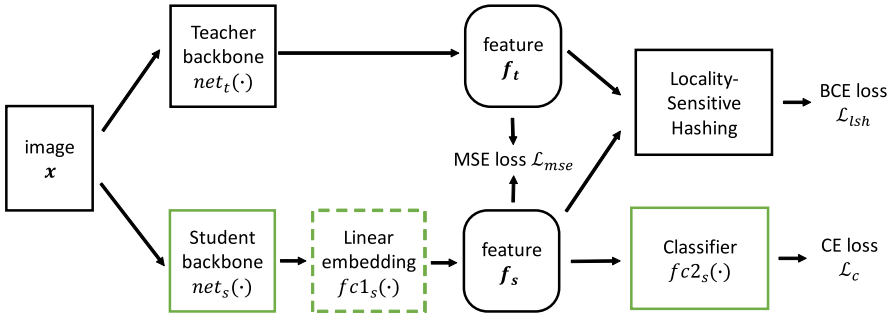


Fig. 1. The pipeline of our method. We use a linear embedding layer to make sure the dimensionality of student's feature is the same as that of the teacher's. But, this embedding layer will be absorbed post-training. (This figure is best viewed in color.).

misalign (cf. Fig. 2). If we have the freedom to rotate and rescale the student's feature space, it will align to the teacher's feature space better. Thanks to our two FC structure in the proposed feature mimicking method, we demonstrate that the first FC layer can transform the student's feature space and make feature mimicking easier, which is particularly important when the student network is initialized using a pretrained model (i.e., the student has formed a basic feature space to finetune rather than a random feature space).

Our contributions are as follows.

- We argue that directly mimicking features in the penultimate layer is advantageous for knowledge distillation. It produces better performance than distilling logits after log-softmax (as in [3]). It can be applied when the teacher and student have different architectures, while distilling features in the middle layers cannot.
- We claim that the feature's direction contains more effective information than its magnitude, and we should allow more freedom to the student feature's magnitude. We propose a loss term based on Locality-Sensitive Hashing (LSH) [14] to meet this requirement, and theoretically show why LSH fits this purpose.
- We propose a training strategy for mimicking features in transfer learning. With a pretrained student, we first transform its feature space to align to the teacher's, then finetune the student on the target dataset with our loss function. Our method is flexible and handles multi-label recognition well, while existing KD methods are difficult to apply to multi-label problems.

Our feature mimicking framework achieves state-of-the-art results on both single-label and multi-label recognition, and object detection tasks.

The rest of this paper is organized as follows. First, we review the related work in Section 2. Then, we introduce our method for feature mimicking in Section 3, and mathematically analyze the effectiveness of it in Section 4. Experimental results are reported and analyzed in Section 5. Finally, Section 6 concludes this paper.

2 Related Work

Knowledge distillation was first introduced in [3], which proposed to use the teacher's soft logits after log-softmax as extra supervision to train the student. FitNet [8] is the first work to distill the intermediate feature maps between teacher and student. Inspired by this, a variety of other feature-based knowledge distillation methods have been proposed. AT [15] transfers the teacher knowledge to student by the spatial attention maps. AB [16] proposes a knowledge transfer method via distillation of activation boundaries formed by hidden neurons. FitNet, AT and AB focus on activation maps of the middle layers, and it is difficult to apply them on cross-architecture settings. SP [17] considers pairwise similarities of different features instead of mimicking the teacher's representation space. FSP [18] computes the inner product between features from two layers and treats it as the extra information to teach student. FT [9] introduces a paraphraser to compress the teacher feature and uses the translator located at the student network to extract the student factors, then teaches the student by making student factors mimic teacher's compressed features. These methods transform the teacher's feature into other forms, which will lose some information in teacher features. In contrast, feature mimicking in the penultimate layer can apply on arbitrary teacher/student combinations and carry all information from the teacher.

Recently, CRD [10] and SSKD [11] take advantage of contrastive learning and transfer the structural knowledge of the teacher network to the student. In this paper, we argue that we can also achieve state-of-the-art by only mimicking features without explicitly considering the structural knowledge.

Object detection is a fundamental task in computer vision. Several previous works study knowledge distillation on the object detection task. ROI-mimic [19] mimics the features after ROI pooling. Fine-grained [20] uses the ground truth bounding box to generate the foreground mask and distill the foreground features on the feature map. PAD [21] introduces the adaptive sample weighting to improve these distillation methods. In this paper, we will show that mimicking features in the penultimate layer works better.

Locality-sensitive hashing (LSH) was first introduced in [22], [23]. With the help of p-stable distributions, [14] extended the algorithm to the ℓ_2 norm. With the rise of deep learning, hashing methods were widely used in image retrieval [24], [25], [26], [27]. Most of them focused on how to learn good hash functions to transform images into compact codes. Different from that, we utilized LSH to help the student network to learn from the teacher network. To the best of our knowledge, we are the first to propose the use of LSH in distilling knowledge.

3 Feature Mimicking for Knowledge Distillation

Fig. 1 shows the pipeline of our method. Given an image $\boldsymbol{x}$, the teacher backbone network extracts feature $\boldsymbol{f}_t$, in which $\boldsymbol{f}_t \in \mathbb{R}^{D_t}$ is the penultimate layer feature (after the global average pooling and before the final classifier or detection head). The student backbone network extracts feature $\boldsymbol{f}_s$. To make the dimensionalities of $\boldsymbol{f}_s$ and $\boldsymbol{f}_t$ match, we add a linear embedding layer after the student backbone. Section 3.1 will introduce this module in detail.

Three losses are used. $\mathcal{L}_c$ is the regular cross-entropy loss between the student output and the ground truth label of $\boldsymbol{x}$. $\mathcal{L}_{mse}$ and $\mathcal{L}_{lsh}$ are used to make the student feature mimic the teacher's. More details about these two losses can be found in Section 3.2. More analyses are in Sections 3.3 and 3.4. During training, modules with green boxes (student backbone, linear embedding and classifier) in Fig. 1 need to be learned by back-propagation. Parameters in the teacher backbone and locality-sensitive hashing will *not* change after initialization. Finally, Section 3.5 discusses how to initialize our framework. We leave theoretical results for feature mimicking to Section 4.

3.1 The Linear Embedding Layer

When the dimensionality of the student's feature is different from that of the teacher's, we add a linear embedding layer before the student's classifier layer. Assume the dimensionality of student's features and teacher's are D_s and D_t, respectively, the embedding layer is defined as

$$fc1_s(\boldsymbol{f}) = \boldsymbol{W}_1^{\mathsf{T}}\boldsymbol{f} + \boldsymbol{b}_1\,, \tag{1}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{D_s \times D_t}$ and $\boldsymbol{b}_1 \in \mathbb{R}^{D_t}$. The main advantage of this approach is that the embedding layer can be merged into the classifier without adding parameters or computation post-training. Assume the classifier is defined as

$$fc2_s(\boldsymbol{f}) = \boldsymbol{W}_2^{\mathsf{T}}\boldsymbol{f} + \boldsymbol{b}_2\,, \tag{2}$$

where $\boldsymbol{W}_2 \in \mathbb{R}^{D_t \times C}$ and $\boldsymbol{b}_2 \in \mathbb{R}^{C}$. Then, the final classifier for student can be computed by

$$\begin{aligned} fc_s(\boldsymbol{f}) &= fc2_s(fc1_s(\boldsymbol{f})) && (3)\\ &= (\boldsymbol{W}_1\boldsymbol{W}_2)^{\mathsf{T}}\boldsymbol{f} + (\boldsymbol{W}_2^{\mathsf{T}}\boldsymbol{b}_1 + \boldsymbol{b}_2)\,, && (4)\end{aligned}$$

$fc1_s$ and $fc2_s$ can be merged by setting the weights and bias for the final classifier as $\boldsymbol{W}_1\boldsymbol{W}_2$ and $\boldsymbol{W}_2^{\mathsf{T}}\boldsymbol{b}_1 + \boldsymbol{b}_2$, respectively.

This linear embedding layer shares similar idea as FSKD [28]. FSKD adds a 1×1 conv at the end of each block of the student network and proves that the 1×1 conv can be merged into the previous convolution layer. However, FSKD requires the teacher and student to share similar architectures, and adds more parameters during training. Our method is more efficient and can be applied with different teacher/student architectures.

Even when the dimensionality of the student's feature is the same as that of the teacher's, the linear embedding layer may still be necessary. Because the teacher and the student may have significantly different network architectures, their feature spaces may be misaligned. The teacher and the student feature spaces, even when they encode the same semantic information, can still be subject to differences caused by transformations such as rotation and scaling. Fig. 2 illustrates the feature space misalignment issue. Assume the penultimate layer feature is denoted by $\boldsymbol{f}$ and the classifier's parameters are $\boldsymbol{W}$ and $\boldsymbol{b}$, respectively. The prediction can be computed by

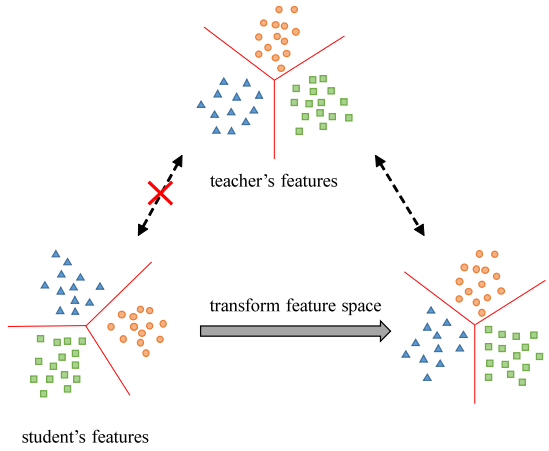


Fig. 2. An illustration of the feature space misalignment issue. The points denote the features, and different colors with different shapes represent different classes. The student's feature space needs to rotate to align to the teacher's. (This figure is best viewed in color.).

$$\boldsymbol{p} = \boldsymbol{W}^{\mathsf{T}}\boldsymbol{f} + \boldsymbol{b}\,. \tag{5}$$

Given any orthogonal matrix $\boldsymbol{R}$, we have

$$\begin{aligned} \boldsymbol{p} &= \boldsymbol{W}^{\mathsf{T}}\boldsymbol{R}^{\mathsf{T}}\boldsymbol{R}\boldsymbol{f} + \boldsymbol{b} && (6)\\ &= \boldsymbol{W}_*^{\mathsf{T}}\boldsymbol{f}_* + \boldsymbol{b}\,, && (7)\end{aligned}$$

where $\boldsymbol{W}_*$ and $\boldsymbol{f}_*$ are $\boldsymbol{R}\boldsymbol{W}$ and $\boldsymbol{R}\boldsymbol{f}$, respectively. That is, the feature space can be rotated without changing the prediction.

Our linear embedding layer $fc1_s$ can learn any linear transformation (such as the above rotation $\boldsymbol{R}$) to align the student's feature space to that of the teacher's. If we let the student mimic the teacher's features directly without aligning their feature spaces, the performance will be lower, especially when the student has been pretrained. Experimental validation of the importance of feature space alignment can be found in Section 5.3.

3.2 The LSH Module

To mimic the teacher's feature, $\mathcal{L}_{mse}$ and $\mathcal{L}_{lsh}$ are used in our framework. $\mathcal{L}_{mse}$ is defined as

$$\mathcal{L}_{mse} = \frac{1}{nD}\sum_{i=1}^{n} \|\boldsymbol{f}_t(\boldsymbol{x}_i) - \boldsymbol{f}_s(\boldsymbol{x}_i)\|_2^2\,, \tag{8}$$

where $\boldsymbol{f}_t(\boldsymbol{x}_i)$ and $\boldsymbol{f}_s(\boldsymbol{x}_i)$ represent the teacher and student features for the ith image in the training set, and D denotes the dimensionality of the feature (after the linear embedding $fc1_s$). Note that $\mathcal{L}_{mse}$ addresses both feature direction and magnitude. On the contrary, we propose to use locality-sensitive hashing (LSH) [14] to give the student more freedom with regard to its magnitude, but

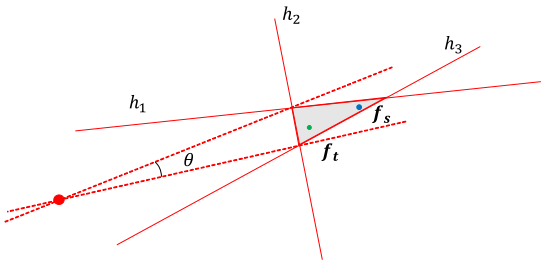


Fig. 3. An illustration of the LSH loss. f_s and f_t denote a student and a teacher feature vector for the same input image, respectively. h represents the hash function constraints. These constraints form a small polyhedron (the shaded region) and θ is the maximum angle between any two vectors in this polyhedron. Magnitudes of these features, however, can alter with greater freedom.

let the student concentrate more on mimicking the feature direction.

Fig. 3 shows an illustration for $\mathcal{L}_{lsh}$. In our LSH module, each hash function can be considered as a linear constraint. Many constraints will divide the feature space into a lot of polyhedra, and in general each polyhedron will be small. The LSH loss will encourage f_s and f_t to fall into the same polyhedron. Hence, more hash functions will result in smaller polyhedra, which in turn means that the angle between f_t and f_s will be small (upper bounded by θ in Fig. 3, which is small itself because the polyhedron is small compared to the feature magnitudes.) In short, the LSH module encourages f_t and f_s to have similar directions, but relaxes constraints on their magnitudes.

LSH aims at hashing the points into bins by several hash functions to ensure that, for each function, near points will fall into the same bin with high probability. In our framework, we use the hash family based on the Gaussian distribution which is a 2-stable distribution, defined as

$$h_{\boldsymbol{w},b}(\boldsymbol{f}) = \left\lfloor \frac{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{f} + b}{r} \right\rfloor, \tag{9}$$

where $\boldsymbol{f} \in \mathbb{R}^D$ is the feature, $\boldsymbol{w} \in \mathbb{R}^D$ is a random vector whose entries are sampled from a Guassian distribution, b is a real number chosen uniformly from the range $[0, r]$, r is the length of each bin, and $\lfloor \cdot \rfloor$ is the floor function.

Our loss term $\mathcal{L}_{lsh}$ encourages the student feature to fall into the same bin as that of the teacher's. According to the theory of locality-sensitive hashing, for two vectors f_1, f_2, the probability of collision decreases monotonically with the distance between f_1 and f_2. Therefore, $h_{\boldsymbol{W},b}(\boldsymbol{f}_t) = h_{\boldsymbol{W},b}(\boldsymbol{f}_s)$ (which will result in a low value of $\mathcal{L}_{lsh}$) is a necessary condition for $\|\boldsymbol{f}_t - \boldsymbol{f}_s\|_2 = 0$. Hence, it is reasonable to force the student to mimic the teacher by minimizing $\mathcal{L}_{lsh}$.

In our framework, we use N hash functions with the form in Equation (9). The locality-sensitive hashing module will generate N hash codes for each feature. 0 is used as the threshold to chop the real line. Therefore, the LSH module can be implemented by a FC layer and a signum function

$$h_{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{f}) = \operatorname{sign}(\boldsymbol{W}^{\mathsf{T}}\boldsymbol{f} + \boldsymbol{b}), \tag{10}$$

$$\operatorname{sign}(x) = \begin{cases} 1, & \text{if } x > 0; \\ 0, & \text{otherwise}, \end{cases} \tag{11}$$

in which $\boldsymbol{f} \in \mathbb{R}^D$ is the feature, $\boldsymbol{W} \in \mathbb{R}^{D\times N}$ is the weights whose entries are sampled from a Guassian distribution and $\boldsymbol{b} \in \mathbb{R}^N$ is the bias. Equation (10) generates N binary codes for teacher feature $\boldsymbol{f}_t$, and the hash code for student feature is expected to be the same as teacher's. We *enforce this requirement by learning a classification problem*, and the binary cross entropy loss is to be minimized, i.e.,

$$\mathbf{h} = \operatorname{sign}(\boldsymbol{W}^{\mathsf{T}}\boldsymbol{f}_t + \boldsymbol{b}), \tag{12}$$

$$\mathbf{p} = \sigma(\boldsymbol{W}^{\mathsf{T}}\boldsymbol{f}_s + \boldsymbol{b}), \tag{13}$$

$$\mathcal{L}_{lsh} = -\frac{1}{nN}\sum_{i=1}^{n}\sum_{j=1}^{N}\left[h_j \log p_j + (1-h_j)\log(1-p_j)\right], \tag{14}$$

where σ is the sigmoid function $\sigma(x) = \frac{1}{1+\exp(-x)}$, h_j and p_j are the jth entry of $\mathbf{h}$ and $\mathbf{p}$, respectively.

Finally, inspired by [29], we only distill the features which the teacher classifies correctly. To reduce the effect of randomness in the locality-sensitive hashing module, the average of the last 10 epochs' models during training is used as our final model.

3.3 Experimental Analysis

We use experiments to demonstrate the advantage of giving the student more freedom to the feature magnitude and making it focus on mimicking the feature direction.

Table 1 shows the experimental results. The models vgg13 and vgg8 share similar architectures, while ResNet50 and MobileNetV2 have different architectures. "CE" denotes training the student by only the cross entropy loss without a teacher. We find that $\|\boldsymbol{f}_s\|_2$ is very different from $\|\boldsymbol{f}_t\|_2$. More statistics on $\|\boldsymbol{f}\|_2$ of different models can be found in Table 2. When knowledge distillation is not used, teacher and student features have very different directions as there are large angles between them, especially when their architectures are different.

When the ℓ_2 loss (ℓ_2 + CE) is used for feature mimicking, the student features are encouraged to be similar to the teacher features in *both magnitudes and angles*, and the student accuracy is higher.

The proposed LSH loss *gives the student more freedom to its feature magnitude*. With the LSH loss (LSH + CE), vgg8 gets a larger feature magnitude while MobileNetV2 gets a smaller feature magnitude than that of CE. For vgg8, although θ of LSH + CE is a little larger than that of ℓ_2 + CE, the accuracy of LSH + CE is higher, which shows the benefit of giving more freedom to the feature magnitude. For MobileNetV2, LSH + CE achieves both a smaller θ and better performance.

Finally, the LSH loss and the ℓ_2 loss can be combined to help each other, and result in both smaller θ (i.e., similar directions) and better accuracy rates.

In Section 4, we will analyze the LSH module theoretically.

3.4 Ensemble All Losses

The final loss consists of two terms, the classification and the feature mimicking losses. The regular cross-entropy loss

TABLE 1
The Difference Between Teacher Features and Student Features

	Teacher / Student / dataset	vgg13 / vgg8 train	vgg13 / vgg8 test	ResNet50 / MobileNetV2 train	ResNet50 / MobileNetV2 test
	$\|\boldsymbol{f}_t\|_2$	12.64	11.83	15.52	15.03
CE	$\|\boldsymbol{f}_s\|_2$	16.50	16.16	16.64	16.33
	θ	69.49°	68.00°	90.09°	90.07°
	Acc@1	99.19	70.72	90.31	64.36
ℓ_2 + CE	$\|\boldsymbol{f}_s\|_2$	12.53	12.06	13.82	13.59
	θ	26.86°	29.90°	32.04°	33.25°
	Acc@1	98.26	72.33	89.07	65.73
LSH + CE	$\|\boldsymbol{f}_s\|_2$	24.74	23.73	9.69	9.60
	θ	28.22°	31.00°	31.28°	32.29°
	Acc@1	97.60	72.69	84.11	67.02
LSH+ℓ_2+CE	$\|\boldsymbol{f}_s\|_2$	15.22	14.50	9.94	9.83
	θ	25.43°	28.99°	29.73°	30.80°
	Acc@1	97.72	73.68	85.76	68.99

The statistics were estimated average values on the training and testing sets of CIFAR-100. $\|\boldsymbol{f}_t\|_2$ and $\|\boldsymbol{f}_s\|_2$ denote the 2-norm of teacher and student features, respectively. θ represents the average angle between them. Acc@1 is the accuracy (%) of the student model.

TABLE 2
The Networks Used in Our Experiments

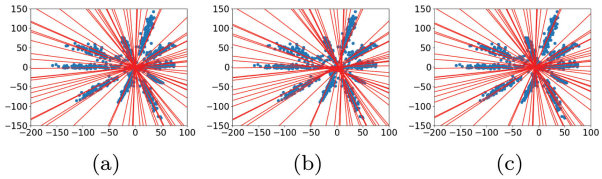
role	model	D	std	$\|\boldsymbol{f}\|_2$
Teacher	WRN-40-2	128	0.1713	13.64
	resnet56	64	0.2415	18.08
	resnet110	64	0.2262	20.13
	resnet32x4	256	0.1100	12.06
	vgg13	512	0.0749	12.64
	ResNet50	2048	0.0381	15.52
Student	WRN-16-2	128	0.2035	15.07
	WRN-40-1	64	0.2638	14.29
	resnet20	64	0.2704	14.47
	resnet32	64	0.2563	16.06
	resnet8x4	256	0.1573	16.92
	vgg8	512	0.0980	16.50
	MobileNetV2	640	0.0579	15.82
	ShuffleNetV1	960	0.0642	17.26
	ShuffleNetV2	1024	0.0625	15.70
Teacher	ResNet34	512	0.0640	30.75
Student	ResNet18	512	0.0695	29.58

ResNet34 and ResNet18 were used on ImageNet, while other models were used on CIFAR-100. D denotes the dimensionality of the feature before the final classifier. std represents the standard deviation of the final classifier's weight with vanilla training. $\|\boldsymbol{f}\|_2$ is the mean of the 2-norm of features in the training set.

$\mathcal{L}_c$ is used as the classification loss. We use both $\mathcal{L}_{mse}$ and $\mathcal{L}_{lsh}$ as the feature mimicking loss. Different from CRD [10] and SSKD [11], our method does *not* need the knowledge distillation loss [3] (KL-divergence between teacher and student logits with temperature). The final loss is

$$\mathcal{L} = \mathcal{L}_c + \beta(\mathcal{L}_{mse} + \mathcal{L}_{lsh}), \tag{15}$$

where β is the balancing weight. Therefore, if the mean square loss is already used in other researches (e.g., detection, segmentation), our LSH module can be added directly without introducing extra hyperparameter.



(a) (b) (c)

Fig. 4. Illustration of different initialization for the bias. Red lines denote the hash function constraints, while blue points represent teacher features. (a), (b), and (c) show the bias initialized by **0**, the median, or the mean of the teacher hash values, respectively. This figure is best viewed in color and zoomed in.

3.5 Model Initialization

The LSH module needs to be initialized before the end-to-end training. In the LSH module, the entries of W are sampled from a Guassian distribution. We always set 0 as its mean and treat the standard deviation (std_{hash}) as a hyperparameter. To find a good default value for std_{hash}, we collect statistics about the standard deviation (std) of the final classifier's weight ($\mathbf{W}'$) with vanilla training (cf. Table 2). Assume $\mathbf{W}' = [\mathbf{w}'_1, \mathbf{w}'_2, \ldots, \mathbf{w}'_c]^{\mathrm{T}}$, where $\mathbf{w}'_i \in \mathbb{R}^D$ and c is the number of categories, the expectation of $\|\mathbf{w}'\|_2$ can be roughly calculated by

$$E(\|\mathbf{w}'\|_2) = E\left(\sqrt{\mathbf{w}'^{\mathrm{T}}\mathbf{w}'}\right) \approx std \times \sqrt{D}, \tag{16}$$

where the last transition holds because we noticed that the mean of $\mathbf{W}'$ is roughly zero. There is a tendency that $E(\|\mathbf{w}'\|_2)$ does not change drastically, and std will become small when D is large. These phenomena inspire us to choose std_{hash} according to D. We also find that directly using the std of teacher's final classifier's weight is a good default value for std_{hash}.

$\boldsymbol{b}$ is the bias in the LSH module. As shown in Fig. 4, the bias can be initialized by **0**, the median, or the mean of the teacher hash values. Because BCE loss is applied, to make the binary classification problem balanced, we use the median of the teacher hash values as the bias in our LSH module. We also tried to use the mean of the teacher hash values or simply set $\boldsymbol{b} = \mathbf{0}$. Later we will exhibit in Tables 7 and 8 the experimental results when using different initialization for the bias. These results show that our method is not sensitive to the initialization of $\boldsymbol{b}$.

4 Theoretical Analyses

Now we will analyze why the LSH loss is sensitive to the feature's direction but not to the feature's magnitude. First, the following Claim 1 says that if the teacher features are scaled, $\mathcal{L}_{lsh}$ will not change, i.e., $\mathcal{L}_{lsh}$ is not sensitive to the teacher feature's magnitude.

Claim 1. *For a given scale $s > 0$, $\mathcal{L}_{lsh}(s\boldsymbol{f}_t, \boldsymbol{f}_s) = \mathcal{L}_{lsh}(\boldsymbol{f}_t, \boldsymbol{f}_s)$ for arbitrary $\boldsymbol{f}_s$.*

Next, the following Claim 2 states that when $\boldsymbol{f}_s$ and $\boldsymbol{f}_t$ have the same direction, $\mathcal{L}_{lsh}$ will encourage $\boldsymbol{f}_s$ to be longer.

Claim 2. *Assume the direction of $\boldsymbol{f}_s$ is the same as that of $\boldsymbol{f}_t$, and $\boldsymbol{b} = \mathbf{0}$ in LSH. For a given scale $s > 1$, then $\mathcal{L}_{lsh}(\boldsymbol{f}_t, s\boldsymbol{f}_s) \leq \mathcal{L}_{lsh}(\boldsymbol{f}_t, \boldsymbol{f}_s)$ always holds.*

Finally, the following Claims 3 and 4 are the most important conclusions, which explain why our LSH loss can help

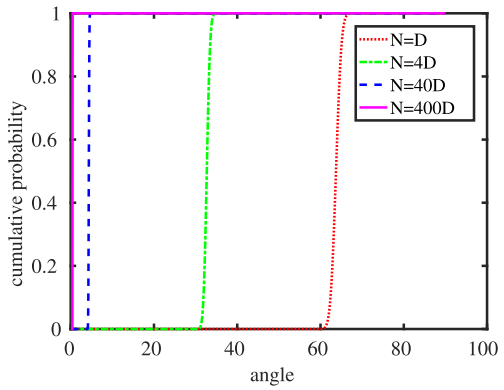


Fig. 5. The cumulative probability of the angle with $D = 2048$, where D and N denote the feature dimensionality and the number of hash functions, respectively. With N becoming larger, the angle between the student feature and the teacher feature will become smaller with high probability. This figure is best viewed in color and zoomed in.

the student to mimic the direction of teacher features. Claim 3 computes the probability of the LSH loss being small (less than $\log 2$) when we are given $\angle(\boldsymbol{f}_t, \boldsymbol{f}_s)$, the angle between teacher and student features. Hence, if the angle between $\boldsymbol{f}_s$ and $\boldsymbol{f}_t$ is smaller, the LSH loss will become small with higher probability.

Claim 4 gives the probability of $\angle(\boldsymbol{f}_t, \boldsymbol{f}_s) < \epsilon$ under the constraint that $\mathcal{L}_{lsh}$ is small. Using the probability formula in Claim 4, we can numerically calculate the cumulative probability of the angle when $\mathcal{L}_{lsh}$ meets the condition (cf. Fig. 5). From this figure, we can conclude that if more hashing functions are used, the direction of $\boldsymbol{f}_s$ will approach that of $\boldsymbol{f}_t$ with higher probability.

Claim 3. *Suppose $\boldsymbol{b} = \boldsymbol{0}$ in LSH, and $\boldsymbol{f}_s$ and $\boldsymbol{f}_t$ follow the standard Gaussian distribution. Then*

$$\Pr\left\{l_j < \log 2 \,|\, \angle(\boldsymbol{f}_t, \boldsymbol{f}_s) = \theta\right\} = 1 - \frac{\theta}{\pi}, \tag{17}$$

will hold, where $\angle(\boldsymbol{f}_t, \boldsymbol{f}_s)$ denotes the angle between $\boldsymbol{f}_t$ and $\boldsymbol{f}_s$, and

$$l_j \doteq -h_j \log\left(p_j\right) - \left(1 - h_j\right) \log\left(1 - p_j\right). \tag{18}$$

Claim 4. *Suppose $\boldsymbol{b} = \boldsymbol{0}$ in LSH, and $\boldsymbol{f}_s$ and $\boldsymbol{f}_t$ follow the standard Gaussian distribution. Then, for any $0 < \epsilon < \pi$, the equation*

$$\begin{aligned}&\Pr\left\{\angle(\boldsymbol{f}_t, \boldsymbol{f}_s) < \epsilon \,|\, \bigwedge_{j=1}^{N} \left(l_j < \log 2\right)\right\} \\ &= \frac{\int_0^{\epsilon} \left(\left(1 - \frac{\theta}{\pi}\right)^N \cdot \sin^{D-2}(\theta)\right) \mathrm{d}\theta}{\int_0^{\pi} \left(\left(1 - \frac{\theta}{\pi}\right)^N \cdot \sin^{D-2}(\theta)\right) \mathrm{d}\theta},\end{aligned} \tag{19}$$

will hold.

The proof of all Claims are provided in the appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2021.3103973, to this paper. Utilizing these results, we numerically calculate the probability in Claim 4 for different N values. As Fig. 5 shows, when the number of hash function N grows, the angle between teacher and student features indeed converge to 0. That is, our LSH loss is effective in mimicking the teacher feature's direction.

5 Experiments

In this section, we evaluate the proposed feature mimicking framework on single-label classification, multi-label recognition, and object detection. For single-label classification, we use the CIFAR-100 [30] and ImageNet [31] datasets, which are usually used as benchmarks for knowledge distillation. CIFAR-100 contains 32×32 natural images from 100 categories, which contains 50000 training images and 10000 testing images. ImageNet is a large-scale dataset with natural color images from 1000 categories. Each category typically has 1300 images for training and 50 for evaluation.

For CIFAR-100, we used the code provided by CRD [10].[1] For a fair comparison, we used the same hyperparameters of CRD in our experiments, such as the learning rate, batch size and epoch. For ImageNet, we followed the standard PyTorch example code and trained 100 epochs (following CRD).[2]

5.1 Ablation Studies

We first study the effects of the loss functions, hyperparameters in LSH, and model initialization.

5.1.1 The Loss Functions

First, we conduct ablation studies on the loss functions. Our final loss contains $\mathcal{L}_{mse}$ and $\mathcal{L}_{lsh}$. We will use only one of them to see their individual effects.

Tables 3 and 4 summarize the results. We used "KD" [3] as the baseline method. Note that all experiments used the classification loss $\mathcal{L}_c$. "ℓ_2 loss" denotes only using $\mathcal{L}_{mse}$, while "LSH loss" represent only using $\mathcal{L}_{lsh}$. "ℓ_2 loss + LSH loss" combines the $\mathcal{L}_{mse}$ and $\mathcal{L}_{lsh}$ as in Equation (15). To balance the feature mimicking loss and classification loss, β was set as 6. In these tables, we also show the relative improvement as a percentage. Accuracy of the student and the teacher are treated as 0 and 100 percent, respectively. For example, in the last column of Table 4, the student and teacher accuracy are 70.50 and 75.61, while the proposed "ℓ_2 loss + LSH loss" is 76.25, hence the relative improvement is $\frac{76.25-70.50}{75.61-70.50} = 113\%$.

When the teacher and student share similar architectures, only using the ℓ_2 loss can surpass the standard KD significantly, which demonstrates the advantage of feature mimicking for knowledge distillation. When only applying the LSH loss we proposed, the performance of most teacher/student combinations are better than that of the ℓ_2 loss, showing the benefit of giving the student more freedom to the feature magnitude and letting it focus on mimicking the feature direction. Combining ℓ_2 and LSH losses can boost the performance. We believe it is because the LSH loss can alleviate the shortcomings of the ℓ_2 loss, and the LSH loss can also benefit from the ℓ_2 loss.

1. https://github.com/HobbitLong/RepDistiller
2. https://github.com/pytorch/examples/tree/master/imagenet

TABLE 3
Test Accuracy (%) of the Student Network on CIFAR-100

Teacher Student	WRN-40-2 WRN-16-2	WRN-40-2 WRN-40-1	resnet56 resnet20	resnet110 resnet20	resnet110 resnet32	resnet32x4 resnet8x4	vgg13 vgg8
Teacher	75.61	75.61	72.34	74.31	74.31	79.42	74.64
Student	73.26	71.98	69.06	69.06	71.14	72.50	70.36
KD	74.92 (71% ↑)	73.54 (43% ↑)	70.66 (49% ↑)	70.67 (31% ↑)	73.08 (61% ↑)	73.33 (12% ↑)	72.98 (61% ↑)
ℓ_2 loss	75.53 (97% ↑)	74.33 (65% ↑)	71.42 (72% ↑)	71.30 (43% ↑)	73.81 (84% ↑)	74.01 (22% ↑)	72.33 (46% ↑)
LSH loss	75.61 (100% ↑)	74.20 (61% ↑)	71.51 (75% ↑)	71.73 (51% ↑)	73.69 (80% ↑)	73.49 (14% ↑)	72.69 (54% ↑)
ℓ_2 loss + LSH loss	75.62 (100% ↑)	74.54 (71% ↑)	71.65 (79% ↑)	71.39 (44% ↑)	73.99 (90% ↑)	73.37 (13% ↑)	73.68 (78% ↑)

The teacher and the student share similar architectures.

TABLE 4
Test Accuracy (%) of the Student Network on CIFAR-100

Teacher Student	vgg13 MobileNetV2	ResNet50 MobileNetV2	ResNet50 vgg8	resnet32x4 ShuffleNetV1	resnet32x4 ShuffleNetV2	WRN-40-2 ShuffleNetV1
Teacher	74.64	79.34	79.34	79.42	79.42	75.61
Student	64.60	64.60	70.36	70.50	71.82	70.50
KD	67.37 (28% ↑)	67.35 (19% ↑)	73.81 (38% ↑)	74.07 (40% ↑)	74.45 (35% ↑)	74.83 (85% ↑)
ℓ_2 loss	66.98 (24% ↑)	65.73 (8% ↑)	71.90 (17% ↑)	74.65 (47% ↑)	75.73 (51% ↑)	75.37 (95% ↑)
LSH loss	67.48 (29% ↑)	67.02 (16% ↑)	74.15 (42% ↑)	75.49 (56% ↑)	75.56 (49% ↑)	75.89 (105% ↑)
ℓ_2 loss + LSH loss	67.16 (25% ↑)	68.99 (30% ↑)	74.89 (50% ↑)	75.36 (54% ↑)	76.70 (64% ↑)	76.25 (113% ↑)

The teacher and the student use different architectures.

When the teacher and student use different architectures, the difference of their accuracy is larger than that in the similar-architecture settings, and their features are more different. Due to the limited capacity of student networks, it is difficult for the student to mimic both features' directions and magnitudes. The experimental results in Table 4 show that only using $\mathcal{L}_{lsh}$ outperforms using $\mathcal{L}_{mse}$ in most cases, which justifies that feature directions have more effective information to boost the student performance, and that we should make the student pay more attention to the feature direction. Combining ℓ_2 and LSH losses is consistently better than only applying the ℓ_2 loss. It demonstrates that feature mimicking indeed benefits from giving more freedom to the student feature's magnitude.

Furthermore, by comparing the relative improvement numbers in Tables 3 and 4, it is obvious that knowledge distillation across different network architectures is a more challenging task than distilling between similar-architecture networks. Hence, it is not surprising that differences among the ℓ_2 loss, the proposed LSH loss, and the "ℓ_2 loss + LSH loss" are relatively small in Table 3. On the other hand, Table 4 confirms that the proposed LSH loss is supervisor to the ℓ_2 loss, which also shows that the combination of these two are complementary in feature mimicking. For example, when we distill knowledge from ResNet50 to MobileNetV2, the combined relative improvement (30 percent) is even higher than the sum of both (8% + 16%).

5.1.2 Hyperparameters in the LSH Module

Next, we study the effect of hyperparameters in the LSH loss. There are three hyperparameters in locality-sensitive hashing. N denotes the number of hashing functions. std_{hash} represents the standard deviation of the Gaussian sampler. Note that we always use 0 as the mean of the Gaussian sampler. β is the balancing weight for both $\mathcal{L}_{lsh}$ and $\mathcal{L}_{mse}$.

Tables 5 and 6 summarize the results. First, when $std_{hash} = 1$ and $N = 2048$, different teacher/student combinations achieve the best results with different β. So it is better to use a validation set to tune this hyperparameter. Limited by computation resources, we simply used $\beta = 6$ for all experiments on CIFAR-100. Second, the value of std_{hash} also affect the performance. But we find that it is less sensitive than β. Third, a larger N may reduce the randomness in LSH. Experiments show that setting $N = 2048$ is good enough. Overall, if applying our method to other problems, we suggest that $N = 2048$ or $N = 4D_t$, $std_{hash} = 1$ or $std_{hash} = std_t$, and finally using a validation set to tune β.

5.1.3 Different Model Initialization

We study different initialization for bias in the LSH module. By default, the bias is initialized as the median of teacher hashing codes to balance the binary classification problem. We also tried to use the mean of teacher hashing codes or $\mathbf{0}$ to initialize the bias. Tables 7 and 8 present the results. We find that knowledge distillation is not sensitive to the initialization of bias. When apply our method on large-scale datasets (like ImageNet), we used $\mathbf{0}$ to initialize the bias because it is difficult to compute the median.

5.2 Single-Label Classification

Tables 9 and 10 compare our method with other knowledge distillation approaches on the CIFAR-100 benchmark. We simply set $\beta = 6$, $std_{hash} = 1$ and $N = 2048$ for all experiments. And for a fair comparison, we used the same teacher networks as CRD [10]. Different from SSKD [11], we only used self-supervised learning [5] to train student networks and got the backbone weights to initialize our framework.

TABLE 5
Test Accuracy (%) of the Student Network on CIFAR-100 Using Different Hyperparameters (β, std_{hash}, N)

Teacher Student	WRN-40-2 WRN-16-2	WRN-40-2 WRN-40-1	resnet56 resnet20	resnet110 resnet20	resnet110 resnet32	resnet32x4 resnet8x4	vgg13 vgg8
std_t	0.17	0.17	0.24	0.23	0.23	0.11	0.07
std_s	0.20	0.26	0.27	0.27	0.26	0.16	0.10
D_t	128	128	64	64	64	256	512
D_s	128	64	64	64	64	256	512
$(1, 1, 2048)$	75.33	73.50	71.25	71.17	73.37	73.64	73.06
$(3, 1, 2048)$	75.47	74.16	71.70	71.68	73.87	74.11	72.91
$(5, 1, 2048)$	75.99	74.43	71.41	71.66	73.32	73.66	73.77
$(6, 1, 2048)$	75.62	74.54	71.65	71.39	73.99	73.37	73.68
$(7, 1, 2048)$	76.34	74.36	71.18	71.78	73.96	73.70	73.89
$(6, std_t, 2048)$	76.11	74.42	70.96	71.75	74.00	73.91	73.57
$(6, std_s, 2048)$	75.53	74.25	71.43	71.60	74.19	73.82	73.61
$(6, std_t, 4 \times D_t)$	76.43	74.15	71.27	71.13	73.55	74.13	73.57
$(6, std_t, 32 \times D_t)$	75.84	74.51	70.96	71.75	74.00	73.77	73.25

The teacher and the student share similar architectures.

TABLE 6
Test Accuracy (%) of the Student Network on CIFAR-100 Using Different Hyperparameters (β, std_{hash}, N)

Teacher Student	vgg13 MobileNetV2	ResNet50 MobileNetV2	ResNet50 vgg8	resnet32x4 ShuffleNetV1	resnet32x4 ShuffleNetV2	WRN-40-2 ShuffleNetV1
std_t	0.07	0.04	0.04	0.11	0.11	0.17
std_s	0.06	0.06	0.10	0.06	0.06	0.06
D_t	512	2048	2048	256	256	128
D_s	640	640	512	960	1024	960
$(1, 1, 2048)$	66.82	65.79	72.12	75.23	75.42	74.98
$(3, 1, 2048)$	67.95	67.33	73.47	74.94	76.12	76.17
$(5, 1, 2048)$	68.01	67.60	74.64	75.38	75.56	76.06
$(6, 1, 2048)$	67.16	68.99	74.89	75.36	76.70	76.25
$(7, 1, 2048)$	67.88	69.20	74.43	75.25	76.70	76.35
$(6, std_t, 2048)$	68.12	67.57	72.89	75.22	76.52	75.63
$(6, std_s, 2048)$	67.77	67.47	73.68	74.93	76.27	75.70
$(6, std_t, 4 \times D_t)$	68.12	67.95	72.80	75.02	76.46	76.36
$(6, std_t, 32 \times D_t)$	67.78	67.33	72.76	75.36	76.25	75.83

The teacher and the student use different architectures.

TABLE 7
Test Accuracy (%) of the Student Network on CIFAR-100 With Different Initializations of Bias in the LSH Module

Teacher Student	WRN-40-2 WRN-16-2	WRN-40-2 WRN-40-1	resnet56 resnet20	resnet110 resnet20	resnet110 resnet32	resnet32x4 resnet8x4	vgg13 vgg8
KD	74.92	73.54	70.66	70.67	73.08	73.33	72.98
0	**76.04**	74.46	71.16	**71.79**	**74.18**	**73.70**	73.92
mean	75.39	74.11	71.52	70.95	73.85	73.64	**73.98**
median	75.62	**74.54**	**71.65**	71.39	73.99	73.37	73.68

The teacher and the student share similar architectures. Bold *denotes the best results.*

Table 9 presents the results when the teacher and student share similar architecture. Note that "Ours (1FC)" removed the linear embedding layer, which is possible because teacher and student features have the same dimensionality. Our method surpasses CRD+KD [10] on most teacher/student combinations. Note that our method did not use the original KD [3] loss, and is thus more flexible. Compared with SSKD [11], our method outperforms on five teacher/student combinations. And our method can be combined with SSKD ("Ours + SSKD"), which consistently outperforms SSKD. We simply set $\beta = 0.01, std_{hash} = 1, N = 2048$ and added our loss terms into the SSKD framework.

Table 10 summarizes the results when the architectures of teacher and student are different. Our method outperformed CRD+KD [10] on the majority of teacher/student combinations, but slightly worse than SSKD [11]. These results suggest that with different teacher/student architectures, self-supervised learning is critical for KD (because SSKD outperformed other methods). However, note that our method can be combined with SSKD, which consistently outperforms

TABLE 8
Test Accuracy (%) of the Student Network on CIFAR-100 With Different Initializations of Bias in the LSH Module

Teacher Student	vgg13 MobileNetV2	ResNet50 MobileNetV2	ResNet50 vgg8	resnet32x4 ShuffleNetV1	resnet32x4 ShuffleNetV2	WRN-40-2 ShuffleNetV1
KD	67.37	67.35	73.81	74.07	74.45	74.83
0	67.14	68.64	74.25	**75.57**	**76.71**	75.76
mean	**68.16**	68.07	74.54	75.55	75.32	75.99
median	67.16	**68.99**	**74.89**	75.36	76.70	**76.25**

The teacher and student use different architectures. Bold *denotes the best results.*

TABLE 9
Test Accuracy (%) of the Student Network on CIFAR-100

Teacher Student	WRN-40-2 WRN-16-2	WRN-40-2 WRN-40-1	resnet56 resnet20	resnet110 resnet20	resnet110 resnet32	resnet32x4 resnet8x4	vgg13 vgg8
Teacher	75.61	75.61	72.34	74.31	74.31	79.42	74.64
Student	73.26	71.98	69.06	69.06	71.14	72.50	70.36
KD [3]	74.92	73.54	70.66	70.67	73.08	73.33	72.98
FitNet [8]	73.58	72.24	69.21	68.99	71.06	73.50	71.02
AT [15]	74.08	72.77	70.55	70.22	72.31	73.44	71.43
SP [17]	73.83	72.43	69.67	70.04	72.69	72.94	72.68
AB [16]	72.50	72.38	69.47	69.53	70.98	73.17	70.94
FT [9]	73.25	71.59	69.84	70.22	72.37	72.86	70.58
FSP [18]	72.91	n/a	69.65	70.11	71.89	72.62	70.23
CRD [10]	75.48	74.14	71.16	71.46	73.48	75.51	73.94
CRD+KD [10]	75.64	74.38	**71.63**	71.56	73.75	75.46	74.29
Ours (1FC)	75.99	–	71.39	**71.64**	**73.90**	73.40	73.78
Ours	**76.41**	74.64	71.44	71.48	73.59	76.75	74.63
SSKD* [11]	75.55	75.50	71.00	71.27	73.60	76.13	74.90
Ours + SSKD	75.89	**75.72**	71.29	71.34	73.68	**76.95**	**75.19**

*The teacher and the student share similar architectures. We denote by * methods where we re-run three times using author-provided code. And the results of our method were run by five times.* Bold *denotes the best results.*

TABLE 10
Test Accuracy (%) of the Student Network on CIFAR-100

Teacher Student	vgg13 MobileNetV2	ResNet50 MobileNetV2	ResNet50 vgg8	resnet32x4 ShuffleNetV1	resnet32x4 ShuffleNetV2	WRN-40-2 ShuffleNetV1
Teacher	74.64	79.34	79.34	79.42	79.42	75.61
Student	64.60	64.60	70.36	70.50	71.82	70.50
KD [3]	67.37	67.35	73.81	74.07	74.45	74.83
FitNet [8]	64.14	63.16	70.69	73.59	73.54	73.73
AT [15]	59.40	58.58	71.84	71.73	72.73	73.32
SP [17]	66.30	68.08	73.34	73.48	74.56	74.52
AB [16]	66.06	67.20	70.65	73.55	74.31	73.34
FT [9]	61.78	60.99	70.29	71.75	72.50	72.03
CRD [10]	69.73	69.11	74.30	75.11	75.65	76.05
CRD+KD [10]	69.94	69.54	74.58	75.12	76.05	76.27
Ours	69.42	69.64	74.74	77.06	77.08	**77.57**
SSKD* [11]	71.24	71.81	75.71	78.18	78.75	77.30
Ours+SSKD	**71.77**	**72.38**	**76.13**	**78.32**	**79.01**	77.46

*The architectures of teacher and student are different. We denote by * methods where we re-run three times using author-provided code. And the results of our method were run by five times.* Bold *denotes the best results.*

SSKD. Same as that on similar architecture, we simply set $\beta = 0.01, std_{hash} = 1, N = 2048$ and added our loss terms into the SSKD framework.

Table 11 summarizes the results on ImageNet. The hyperparameters in our method are $\beta = 5$, $std_{hash} = std_t$ and $N = 2048$. Note that different from $CRD + KD$ and $SSKD$, we did not use the standard KD loss [3] to boost the performance. Only using the ℓ_2 loss to force the student features to mimic the teacher features outperforms CRD, which once again supports the validity of our proposed feature mimicking. Combining the ℓ_2 and LSH losses further boosts the performance by a significant margin and achieves the

TABLE 11
Top-1 and Top-5 Error Rates (%) on the ImageNet Validation Set

	Teacher	Student	CC [32]	SP [17]	Online-KD [33]	KD [3]	AT [15]	CRD [10]	CRD+KD	SSKD [11]	Ours (ℓ_2)	Ours (ℓ_2 + LSH)
Top-1	26.70	30.25	30.04	29.38	29.45	29.34	29.30	28.83	28.62	28.38	28.61	**28.28**
Top-5	8.58	10.93	10.83	10.20	10.41	10.12	10.00	9.87	9.51	**9.33**	9.61	9.59

The teacher and student are ResNet-34 and ResNet-18, respectively. Bold *denotes the best results.*

state-of-the-art performance, which further supports the proposed LSH loss.

5.3 Multi-Label Classification

We consider two typical multi-label classification tasks, i.e., VOC2007 [34] and MS-COCO [35]. VOC2007 contains a train-val set of 5011 images and a test set of 4952 images. And MS-COCO contains 82081 images in the training set and 40137 images for validation. We resize all images into a fixed size (448×448) to train the networks. And the data augmentation consist of random horizontal flips and color jittering. The backbone networks contain MobileNetV2, ResNet18, ResNet34, ResNet50 and ResNet101. The networks are all pre-trained on ImageNet and finetuned on the multi-label classification dataset with stochastic gradient descent (SGD) for 60 epochs in total. The binary cross entropy (BCE) loss is used to finetune the network. We employ the mean average precision (mAP) to evaluate all the methods. Note that multi-label recognition is *not a typical application of KD because existing KD methods rely on the soft logits, which do not exist in multi-label scenarios*. The proposed feature mimicking method, however, is *flexible and handles multi-label distillation well*.

First, we conduct experiments on VOC2007 using ResNet34 as teacher and ResNet18 as student to demonstrate that feature space alignment is necessary and important. The teacher ResNet34 is first trained on ImageNet and then finetuned on VOC2007. It achieves 91.69% mAP as in Table 12. The student ResNet18 achieves 89.15% mAP. And in Section 5.2, we have trained ResNet18 supervised by ResNet34 on ImageNet. This model is denoted as "ResNet18 (pretrained by KD)" and achieves 89.88% mAP. When finetuned on VOC2007 supervised by the teacher with the ℓ_2 loss, ResNet18 achieves a worse performance (88.75 percent) than baseline, which we believe is because the feature spaces of the teacher and student do not align well. If we use ResNet18 pretrained by KD whose feature space aligns to the teacher's, the student can be improved to 90.89 percent. With the 2FC structure, the first linear layer can transform the student feature space to align to the teacher's. It alleviates the feature space misalignment issue and achieve a better performance (89.98 percent) than baseline. ResNet18 pretrained by KD with 2FC achieves a worse performance (90.77 percent) than that with 1FC. That demonstrates it does not need the first linear layer to transform the feature space.

Although the backbone pretrained by KD on a large scale dataset will transfer better and easily mimic the teacher's features during finetuning, it is expensive to pretrain the student on a large scale dataset in many cases. Hence, we propose a simple but effective approach to alleviate the feature space misalignment problem. We finetune the student by two stages. In the first stage, we fix the weights in the student backbone and only optimize the linear embedding layer with the feature mimicking loss functions. This stage aims at transform the student feature space to align to the teacher's. In the second stage, we add the classifier on top of the linear embedding layer and optimize all parameters in the student with the supervision of both the groundtruth labels and the teacher. Table 13 summarizes the results. With this two-stage training, the student can be improved by a large margin, compared with 89.98% mAP when training the student by one stage. We find that the feature mimicking loss chosen in the first stage is important, and the LSHL2 ($\mathcal{L}_{mse} + \mathcal{L}_{lsh}$) loss is consistently better than the ℓ_2 loss.

We conduct experiments on VOC2007 and MS-COCO and adopt two settings, i.e., using ResNet101 to teach ResNet50 and MobileNetV2, respectively. The student is finetuned with the two-stage strategy, and the LSHL2 loss is used in the first stage based on the above findings. Table 14 presents the results on VOC2007. LSHL2 $\rightarrow$ L2 denotes using the LSHL2 loss in the first stage and the ℓ_2 loss in the second stage. The hyperparameters are set as $\beta = 0.5$, $std_{hash} = std_t$, and $N = 4D_t$ in all experiments. LSHL2 $\rightarrow$ L2 achieves the best performance. ResNet50 is improved by 0.41 percent and MobileNetV2 is improved by 0.61 percent. Experimental results of MS-COCO are showed in Table 15. And we use $\beta = 3$, $std_{hash} = std_t$, and $N = 4D_t$ in all experiments. LSHL2 $\rightarrow$ LSHL2 achieves the best performance.

A common trick in the multi-label classification task is replacing the global average pooling (GAP) with the global maximum pooling (GMP). So we evaluate the backbone network with GMP on the MS-COCO. Table 16 presents the

TABLE 12
Test mAP (%) on Pascal VOC2007

Teacher Student	ResNet34 ResNet18	ResNet34 ResNet18 (pretrained by KD)
Teacher	91.69	91.69
Student	89.15	89.88
KD	89.26 (4% ↑)	89.85 (2% ↓)
ℓ_2 (1FC)	88.75 (20% ↓)	90.89 (56% ↑)
ℓ_2 (2FC)	89.98 (33% ↑)	90.77 (49% ↑)

TABLE 13
Test mAP (%) of the Student Network on Pascal VOC07

2nd stage \ 1st stage	L2	LSH	LSHL2
L2	90.40 (49% ↑)	90.21 (42% ↑)	**90.59** (57% ↑)
LSH	90.29 (45% ↑)	90.11 (38% ↑)	90.30 (45% ↑)
LSHL2	90.57 (56% ↑)	90.37 (48% ↑)	**90.59** (57% ↑)

Bold *denotes the best results.*

TABLE 14
Test mAP (%) of the Student Network on Pascal VOC2007

Teacher Student	ResNet101 ResNet50	ResNet101 MobileNetV2
Teacher Student	93.27 92.76	93.27 89.53
LSHL2 → KD	92.69 (14% ↓)	89.64 (3% ↑)
LSHL2 → L2	**93.17** (80% ↑)	**90.14** (16% ↑)
LSHL2 → LSH	92.40 (71% ↓)	89.91 (10% ↑)
LSHL2 → LSHL2	92.85 (18% ↑)	89.90 (10% ↑)

Bold *denotes the best results.*

TABLE 15
Test mAP (%) of the Student Network on MS-COCO

Teacher Student	ResNet101 ResNet50	ResNet101 MobileNetV2
Teacher Student	77.67 75.54	77.67 71.06
LSHL2 → KD	75.14 (19% ↓)	71.47 (6% ↑)
LSHL2 → L2	77.04 (70% ↑)	73.28 (34% ↑)
LSHL2 → LSH	76.59 (49% ↑)	73.73 (40% ↑)
LSHL2 → LSHL2	**77.16** (76% ↑)	**73.73** (40% ↑)

Bold *denotes the best results.*

results. As previously mentioned, we use ResNet101 (GMP) to teach MobileNetV2 (GMP) and ResNet50 (GMP). In addition, we also evaluate the performance of self-distillation, i.e., using ResNet101 (GMP) to teach ResNet101 (GMP). Our method achieves better performances than baselines. We compared our method with MCAR [36], which employs a complex training pipeline designed for multi-label classification and is the state-of-the-art method on multi-label classification. Our MobileNetV2 surprisingly surpassed that in MCAR, which demonstrates the advantage of our method.

5.4 Detection

We evaluate our method on the object detection task. Following previous work [21], we conduct experiments on the Pascal VOC dataset [34]. The training set consists of the VOC2007 trainval set and the VOC2012 trainval set, and in total 21K images. The testing set is the VOC2007 test set of 5K images. We use mAP@0.5 as the metric to compare the performance of different methods. The detection frameworks we adopted are both two-stage (Faster-RCNN [37]) and one-stage (RetinaNet [38]). And we use four networks (ResNet50, ResNet101, VGG11, VGG16) pretrained on ImageNet as the backbone. FPN [39] layers are adopted in all experiments. All models are finetuned on VOC with 24 epochs. The hyperparameters in feature mimicking loss are set as $\beta = 7$, $std_{hash} = std_t$, $N = 4D_t$ and $bias = 0$ in all experiments. We have released our code.[3]

Fig. 6a shows our feature mimicking framework with Faster-RCNN. As in classification, we want the student to mimic features in the penultimate layer. In the object detection framework, two linear layers are applied on the penultimate layer to generate the classification and bounding box predictions, respectively. Given one image, the backbone and FPN produce the feature pyramid, and the region proposal network (RPN) generates proposals to indicate the localities that objects may appear. Hence, many features are extracted according to the proposals. To make sure the student will mimic teacher's features in the same locations, the teacher uses the proposals produced by the student. When training this framework, we only add the proposed loss to the original loss and apply the traditional training strategy. Our proposed loss is applied on the entire detection network, and it affects the optimization of the backbone network, FPN, RPN and MLP.

TABLE 16
Test mAP (%) on MS-COCO

Model	MobileNetV2	ResNet50	ResNet101
Baseline	73.90	77.20	79.57
MCAR [36]	75.0	**82.1**	**83.8**
Ours	**76.03**	79.55	81.24

The backbone networks use the global maximum pooling (GMP) to aggregate features. Bold *denotes the best results.*

The experimental results are presented in Table 17. First, we use ResNet101 to teach ResNet50. The performances of these two baseline networks are 83.6 and 82.0 percent, respectively. The teacher is higher than student by 1.6 percent. All experimental results of ROI-mimic, PAD-ROI-mimic, Fine-grained and PAD-Fine-grained are cited from PAD [21]. They improve the student by at most 0.5 percent. With our feature mimicking framework, i.e., mimicking the features in the penultimate layer, simply using the ℓ_2 loss as the feature mimicking loss can improve the student by 1 percent. That shows the benefit of feature mimicking on object detection. Combining the LSH loss and the ℓ_2 loss, the student is improved by 1.1 percent. When using VGG16 to teach VGG11, the ℓ_2 loss can improve the student by 1.8 percent. With the LSH loss, the student is improved by 2.1 percent.

Fig. 6b shows our feature mimicking framework with RetinaNet. Different from Faster-RCNN, RetinaNet produces features on all positions of the feature pyramid, and each position will consider several anchors. With the groundtruth bounding boxes, only a few of positions are considered as positive and sent to the classification loss. We force the student to mimic the features on these positive positions and ignore the features on negative positions. RetinaNet uses class subnet and box subnet to generate class feature and box feature, respectively. We find that it is better to only mimic the class feature and ignore the box feature. So our proposed feature mimicking loss affects the optimization of the backbone network, FPN and the class subnet. Table 18 shows the experimental results. Similar to Faster-RCNN, our feature mimicking framework can improve the student with a large margin. ResNet50 is improved by 0.5 percent whose performance is comparable to the teacher performance. And VGG11 is also improved by 2 percent.

Overall, these object detection experimental results demonstrate the advantages of our method. The LSHL2 loss is consistently better than the ℓ_2 loss in all experiments. Note

3. https://git.nju.edu.cn/wanggh/detection.vision

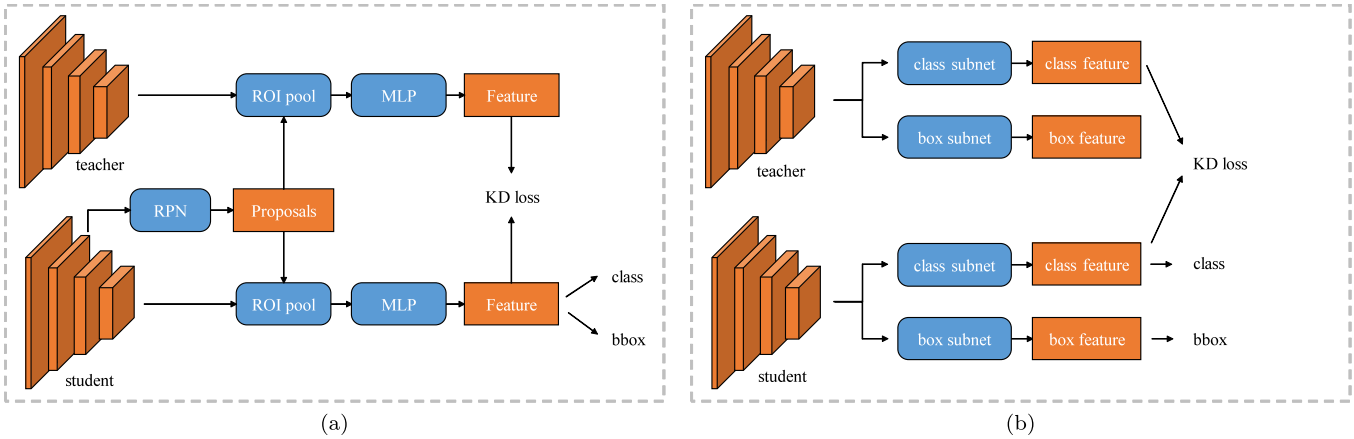


Fig. 6. The pipeline of our method on the object detection task. (a) and (b) show our feature mimicking framework with Faster-RCNN and RetinaNet, respectively. This figure is best viewed in color and zoomed in.

that the difference between the teacher and the student is smaller when compared to the differences in recognition tasks. However, the high relative improvement numbers and the consistent improvements across different experiments both verify our proposed method is effective. In this paper, we only focus on mimicking the final features and leave mimicking proposals as the future work. However, only using feature mimicking has already improved the student by a large margin, and the RetinaNet with ResNet50 backbone is even comparable to the teacher performance.

Compared with multi-label classification, we find it does not need the two stage training strategy on object detection.

TABLE 17
Test mAP@0.5 (%) of the Student Network on Pascal VOC0712

Teacher Student	ResNet101 ResNet50	VGG16 VGG11
Teacher	83.6	79.0
Student	82.0	75.1
ROI-mimic [19]	82.3 (19% ↑)	75.0 (3% ↓)
PAD-ROI-mimic [21]	82.5 (31% ↑)	75.8 (18% ↑)
Fine-grained [20]	82.0 (0% ↑)	74.6 (13% ↓)
PAD-Fine-grained [21]	82.3 (19% ↑)	75.2 (3% ↑)
Ours (L2)	83.0 (63% ↑)	76.9 (46% ↑)
Ours (LSHL2)	**83.1** (69% ↑)	**77.2** (54% ↑)

The detector is Faster R-CNN with different backbones. Bold *denotes the best results.*

TABLE 18
Test mAP@0.5 (%) of the Student Network on Pascal VOC0712

Teacher Student	ResNet101 ResNet50	VGG16 VGG11
Teacher	83.0	76.6
Student	82.5	73.2
Fine-grained [20]	81.5 (200% ↓)	72.0 (35% ↓)
PAD-Fine-grained [21]	81.9 (120% ↓)	73.2 (0% ↓)
Ours (L2)	82.6 (20% ↑)	74.8 (47% ↑)
Ours (LSHL2)	**83.0** (**100**% ↑)	**75.2** (59% ↑)

The detector is RetinaNet with different backbones. Bold *denotes the best results.*

We guess it may be due to the MLP layer and the subnet in Faster-RCNN and RetinaNet, respectively. These layers are randomly initialized before finetuning on the detection dataset. The feature space alignment will be learned implicitly in these layers.

6 Conclusion

In this paper, we proposed a flexible and effective knowledge distillation method. We argued that mimicking feature in the penultimate layer is more advantageous than distilling the teacher's soft logits [3]. And to make the student learn the more effective information from the teacher, it needs to give the student more freedom to its feature magnitude, but let it focus on mimicking the feature direction. We proposed a loss term based on Locality-Sensitive Hashing (LSH) [14] to fulfill this objective. Our algorithm was evaluated on single-label classification, multi-label classification and object detection. Experiments showed the effectiveness of the proposed method.

Future work could explore how to improve our method, such as reducing the randomness in the LSH module, and aligning feature spaces efficiently and even if without training data. Applying our method to other problems is also interesting. It is promising to combine our method with self-supervised learning. And we will also consider how to deploy our method to knowledge distillation under a data free setting.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61772256 and Grant 61921006.

References

[1] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.
[2] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis.* 2018, pp. 116–131.
[3] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.
[4] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–16.

[5] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 10709–10719.
[6] X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning," 2020, *arXiv:2003.04297*.
[7] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 9726–9735.
[8] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for thin deep nets," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–13.
[9] J. Kim, S. Park, and N. Kwak, "Paraphrasing complex network: Network compression via factor transfer," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 2760–2769.
[10] Y. Tian, D. Krishnan, and P. Isola, "Contrastive representation distillation," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–14.
[11] G. Xu, Z. Liu, X. Li, and C. C. Loy, "Knowledge distillation meets self-supervision," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 588–604.
[12] K. Xu, L. Rui, Y. Li, and L. Gu, "Feature normalized knowledge distillation for image classification," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 664–680.
[13] F. Wang, X. Xiang, J. Cheng, and A. L. Yuille, "NormFace: L2 hypersphere embedding for face verification," in *Proc. 25th ACM Int. Conf. Multimedia*, 2017, pp. 1041–1049.
[14] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on P-stable distributions," in *Proc. 20th ACM Symp. Comput. Geometry*, 2004, pp. 253–262.
[15] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–13.
[16] B. Heo, M. Lee, S. Yun, and J. Y. Choi, "Knowledge transfer via distillation of activation boundaries formed by hidden neurons," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, pp. 3779–3787.
[17] F. Tung and G. Mori, "Similarity-preserving knowledge distillation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 1365–1374.
[18] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 7130–7138.
[19] Q. Li, S. Jin, and J. Yan, "Mimicking very efficient network for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 7341–7349.
[20] T. Wang, L. Yuan, X. Zhang, and J. Feng, "Distilling object detectors with fine-grained feature imitation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4928–4937.
[21] Y. Zhang *et al.*, "Prime-aware adaptive distillation," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 658–674.
[22] A. Gionis, P. Indyk, R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. 25th Int. Conf. Very Large Data Bases*, 1999, pp. 518–529.
[23] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. 30th Annu. ACM Symp. Theory Comput.*, 1998, pp. 604–613.
[24] Z. Cao, M. Long, J. Wang, and P. S. Yu, "HashNet: Deep learning to hash by continuation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 5609–5618.
[25] Y. Cao, M. Long, B. Liu, and J. Wang, "Deep Cauchy hashing for Hamming space retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1229–1237.
[26] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2064–2072.
[27] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *Proc. AAAI Conf. Artif. Intell.*, 2014, pp. 2156–2162.
[28] T. Li, J. Li, Z. Liu, and C. Zhang, "Few sample knowledge distillation for efficient network compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 14627–14635.
[29] T. Wen, S. Lai, and X. Qian, "Preparing lessons: Improve knowledge distillation with better supervision," 2019, *arXiv:1911.07471*.
[30] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, *Tech. Rep.*, 2009.
[31] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.
[32] B. Peng *et al.*, "Correlation congruence for knowledge distillation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 5006–5015.
[33] X. Lan, X. Zhu, and S. Gong, "Knowledge distillation by on-the-fly native ensemble," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7517–7527.
[34] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, pp. 303–338, 2009.
[35] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.
[36] B. Gao and H. Zhou, "Multi-label image recognition with multi-class attentional regions," 2020, *arXiv:2007.01755*.
[37] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 36, pp. 1137–1149, Jun. 2017.
[38] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2999–3007.
[39] T.-Y. Lin, P. Dollar, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 936–944.

Guo-Hua Wang received the BS degree from the School of Management and Engineering, Nanjing University, China. He is currently working toward the PhD degree with the Department of Computer Science and Technology, Nanjing University. His research interests include computer vision and machine learning.

Yifan Ge received his BS degree in the Kuang Yaming Honors School from Nanjing University in 2019. He is currently a graduate student in the School of Artificial Intelligence at Nanjing University, China. His research interests include computer vision and machine learning.

Jianxin Wu (Member, IEEE) received the BS and MS degrees from Nanjing University, and the PhD degree from the Georgia Institute of Technology, all in computer science. He is currently a professor with the Department of Computer Science and Technology and School of Artificial Intelligence, Nanjing University, China, and associated with the State Key Laboratory for Novel Software Technology, China. His research interests include computer vision and machine learning. He was the (senior) area chair of CVPR, ICCV, ECCV, AAAI, and IJCAI and an associate editor for the *IEEE Transactions on Pattern Analysis and Machine Intelligence*.