

# Coupling the Generator with Teacher for Effective Data-Free Knowledge Distillation

Xu Chen<sup>1</sup> Yang Li<sup>1</sup> Yahong Han<sup>1\*</sup> Guangquan Xu<sup>1</sup> Jialie Shen<sup>2</sup>

<sup>1</sup> College of Intelligence and Computing, Tianjin University, China

<sup>2</sup> Department of Computer Science, City St George's, University of London, UK

lanncx@gmail.com, {yahong, liyang1389, losin}@tju.edu.cn, jerry.shen@citystgeorges.ac.uk

## Abstract

*Data-Free Knowledge Distillation (DFKD) avoids accessing the original training data during knowledge transferring from a large model to a smaller one, possessing significant potential in ensuring the widespread promotion of industry-level applications while safeguarding user privacy and data security. Unfortunately, due to the lack of precise estimation of the original data distribution, existing DFKD methods often rely on manually induced priors to constrain the generator to produce samples that comply with the rules as much as possible. In this paper, we propose a novel method dubbed **Coupling Network (CPNet)** that constructs a generator to explicitly approximate the inverse transformation of the teacher model. Consequently, the two components can be integrated into an autoencoder specifically tailored for label information, where the generated images are treated as latent variables. Since real labels are typically uniformly distributed and the parameters of the teacher model are fixed, this enables our generator to produce images that closely approximate the true distribution. Besides, we transform real labels into feature-level constraints through the inverse transformation of a network classifier with fixed parameters, thereby converting the classification problem of generated images into an issue of distance measurement between features. We utilize this constraint for adversarial training and enhancing the diversity of produced images. Extensive experiments on three public benchmarks demonstrate that our proposed method achieves superior or competitive performance compared to previous state-of-the-art methods, while also exhibiting faster generation speed.*

## 1. Introduction

Knowledge Distillation (KD) refers to the process of transferring knowledge [10, 12, 16] from a large teacher model, which generally consumes substantial resources, to train a

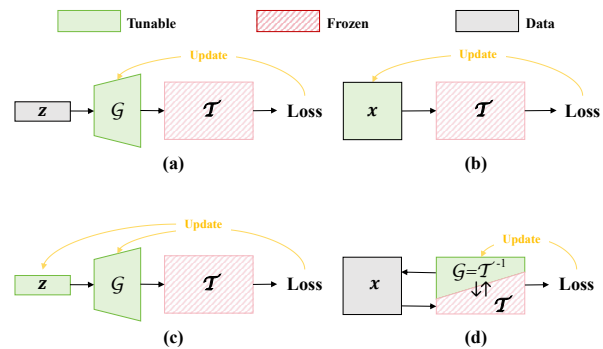


Figure 1. Comparison of different data-free knowledge distillation paradigms. (a) shows the generative methods [2, 3, 20]. (b) is the gradient-based method [19, 29], and (c) is the joint training method [7, 8] that optimize input image and generator together. Our method (d) designs the generator as the inverse transformation of the teacher model and couples them together, which achieves a better utilization of the teacher’s information.

smaller student model. Although this technique has been widely used in domains such as visual recognition and natural language processing [10, 28], traditional KD requires the original data from the pre-training phase of teacher, which has become a significant barrier to its further development. On one hand, the original training data is often difficult to fully access and train on local machines, which has become a common practice for large models (such as CLIP [23] and other LLMs [1, 6]) that have been released. On the other hand, there is an increasing emphasis on privacy, copyright, and security issues in the use of data. In recent years, Data-Free Knowledge Distillation (DFKD) has been proposed, which explores knowledge transfer without accessing any original training data, becoming a remedy for learning in data-scarce and sensitive environments.

Existing DFKD methods typically employ a generator to produce the training image  $\hat{x}$  from a noise vector  $z$ , and then use it for conventional distillation learning. To make the generated samples closer to the distribution of original data, the

\*Corresponding Author

generator needs to be optimized through the labels of teacher network and some prior knowledge about the original image data. As shown in Figure 1, in the training paradigm of previous methods, the teacher model  $\mathcal{T}$  and the generator  $\mathcal{G}$  are structurally independent. The  $\mathcal{T}$  mainly focuses on ensuring that the output samples of  $\mathcal{G}$  are consistent with the original data in terms of category. However, this constraint is too relaxed for estimating the distribution of the entire dataset. Some methods manually constrain the variance and  $l_2$  norm of  $\hat{x}$  to make it look more realistic. DeepInversion [29] further proposes to align the feature maps of  $\hat{x}$  after passing through each BatchNorm layer of the teacher model with the stored statistics (mean and variance) of training data. Due to its good performance, this approach has been used in a series of subsequent works [5, 8, 18, 25]. However, relying on the statistical information of the BatchNorm layers and manually induced constraints often only applies to smaller datasets, as the distribution characteristics of these samples are not very complex, and manually designed prior rules are sufficient to limit the generated results to their approximate range. However, these strategies easily fail on larger datasets, as the distribution space is larger and more complex, and manual priors are difficult to cover.

In this paper, we encapsulate the unified training paradigm for DFKD and identify that, owing to the uncertain distribution of  $\hat{x}$ , traditional generative techniques such as VAE [13] and GAN [9] are prone to failure during the training process. To alleviate this issue, we propose to couple the generator with teacher model. As depicted in Figure 1 (d), the derived architecture inherently constitutes an auto-encoder, which we have termed the Coupling Network (CPNet). As the intermediate hidden variable, although the distribution of  $\hat{x}$  is unknown, its label in standard public datasets is often uniformly distributed. Furthermore, as long as the generator is a inverse transformation of the teacher model, corresponding samples can be generated directly from the label information without modeling the distribution of  $\hat{x}$ . To achieve this, we divide the teacher model into several blocks with BatchNorm layers as separators. Each block is assigned a small inverse generator. The input and output feature maps of each block are reversed, and then normalized according to the mean and variance of the corresponding BatchNorm layer. The local generator are used to fit these features. Thus, connecting all continuous generators together forms the inverse transformation of the entire teacher model, which can be regarded as a flow-based implementation [4, 14] of the teacher’s inverse process. Meanwhile, we naturally feed the features that satisfy the distribution constraints on the BatchNorm layer to the generator’s intermediate layer, so that the prior knowledge of the teacher model about the training data can be explicitly coupled with the generator. It is worth mentioning that CPNet uses lower-resolution noise vectors as input compared to

previous methods, resulting in higher generation efficiency.

In addition, the diversity of images generated by DFKD methods is also a key issue for ensuring the final performance. Some methods [29] introduce adversarial learning to make the generator produce different outputs from the student model after each iteration. However, this approach mainly targets differences at the logits-level. We further extend the adversarial loss to the feature-level. Specifically, for a frozen teacher model, we apply the inverse transformation of Softmax and the fully-connected layer. This allows us to convert the logits of the ground-truth labels into the output features appear before the classifier head. We found that the result of a fixed transformation of this feature vector has a strict collinear relationship with the parameter matrix of the classification head. Hence, we integrate this relationship into the adversarial loss by promoting the feature space of the generated data to be orthogonal to that of the ground truth.

Our main contributions can be summarized as follows:

- We introduce CPNet, a coupled architectural designed to ensure that the generator functions as the inverse transformation of the teacher model, thereby generating synthetic images that exhibit a distribution closely aligned with that of the original dataset.
- To enhance the diversity of the synthetic images, we propose a novel adversarial loss function that converts pseudo labels into feature constraints through the inverse transformation of the fixed classifier.
- We conducted extensive experiments on CIFAR10 [15], CIFAR100 [15], and TinyImageNet [22] to validate the performance of CPNet and the effectiveness of its components. The results indicate that CPNet achieves superior performance compared to all existing methods across most architectures, while requiring significantly less time ( $1.2\times \sim 7.48\times$  faster).

## 2. Related Work

Early methods mainly focus on the distribution estimation problem. Lopes *et al.* [19] first tried the DFKD problem and proposed to compress the trained deep network to a fraction of its size by leveraging some metadata. They assumed that the activation records of pre-trained networks is available and reconstructed the original data by optimizing the input noise to match these activation records. However, this assumption is too strict in real application. Micaelli *et al.* [20] generalized DFKD to a more common setting, *i.e.*, the training data and metadata are unavailable. The generator and student is trained alternatively, and images that student poorly matches the teacher will be searched as adversarial samples. It encourages the generator to explore different region of the input space at each iteration. Inspired by the generative adversarial networks (GANs) [9], Chen *et al.* [2] regarded the teacher as a discriminator and trained a generator to produce training samples for knowledge distillation.

Since the output values of teacher network represent the probabilities of images belonging to each category, they took the class with highest probability as the pseudo ground-truth labels to compute the `CrossEntropy` loss for the generator. Different from general GAN models, the pseudo-label is not completely equal to  $P(\mathbf{y}|\mathbf{z})$ , so there will be deviations. In DeepInversion [29], the input image is directly optimized by the total loss, which is also equivalent to  $\mathbf{z} = \mathbf{1}$  and  $\mathcal{G} = \theta_z$ , where the shape of  $\theta_z$  is the same as the input image. Obviously, compared with the subsequent methods, the generator of this method is too simple to synthesize good images efficiently. However, another method proposed in this paper has a great impact on this domain, which is to force the mean and variance of the input data of each `BatchNorm` layer in the teacher network to be close to the mean and variance retained by the model. This is an effective way to constrain the generated image to the center point of the original training data. Based on the idea of model inversion, Fang *et al.* [7] further introduced a contrastive learning objective to explicitly improve the data diversity by increasing discrimination of instances at each iteration. To cover the large distribution shift during the adversarial training for generator and student model, MAD [5] introduced an EMA to make the update of  $\mathcal{G}$  more smoother, so that the gap between consecutive images is not too large. To further improve the representation of synthetic data, Yu *et al.* [30] proposed a channel-level feature exchange module and a multi-scale spatial activation region consistency loss. The simple operations on features allows more diverse output space for the generator while keeping a slight modification and avoiding deeper exploration for the complex knowledge of teacher model. In the aforementioned methods, the input is always a noise vector adhering to the Gaussian distribution. For the generator, these vectors indicate the information of image labels, but most of them are too weak to reconstruct the whole image. Thus, Tran *et al.* [25] proposed a noisy layer to generate meaningful constant label-text embedding (LTE) as the input of generator instead of random noise. Recently, more methods [8, 18] focus on the efficiency issue of DFKD. FastDFKD [8] achieved a significant speedup for DFKD methods by reusing the common features with a fast meta-synthesizer. Liu *et al.* [18] reduced the data scale by introducing a modulation function to select proper samples for the distillation learning. They carefully controlled the size of sample buffer using strategies like dynamic replay, reinforcement learning, and priority sampling function. Finally they remain a extremely small scale training data that is  $10\times$  less than original training data scale.

Despite significant advancements in current DFKD methods, the challenge of estimating the distribution of generated data remains substantial. Beyond leveraging the statistical information from `BatchNorm` layers and final classification outputs, the knowledge encapsulated within the teacher

model is not fully exploited, making the generation of realistic and adequate samples particularly difficult. In contrast, our approach naturally couples the generator with teacher model to form an auto-encoder, rendering the generator as the inverse transformation of the teacher model. This design allows for a more direct utilization of the teacher model’s knowledge and circumvents the need for a precise distribution of generated images within the generator model.

### 3. Proposed Method

#### 3.1. Preliminary

Given a teacher model  $\mathcal{T}$  pre-trained on dataset  $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^m$ , where each image  $\mathbf{x}_i \in \mathbb{R}^{H \times W \times 3}$  has the spatial shape of  $H \times W$  and  $\mathbf{y}_i \in \mathbb{R}^K$  represents the corresponding one-hot label,  $K$  is the number of categories. Data-free knowledge distillation aims to train the student model  $\mathcal{S}$  without accessing  $\mathcal{D}$ . Current methods approach this target by introducing a generator  $\mathcal{G}$  with parameter  $\theta_g$  to synthesis samples that satisfy the distribution of original training data  $\mathcal{D}$ :

$$\hat{\mathbf{x}} = \mathcal{G}(\mathbf{z}; \theta_g), \quad (1)$$

where  $\mathbf{z} \in \mathbb{R}^D$  is a random noise vector from the normal distribution  $\mathcal{N}(0, \mathbf{I})$ , and each dimension of  $\mathbf{z}$  represents an attribute. Then we can get its prediction by  $\hat{\mathbf{y}} = \mathcal{T}(\hat{\mathbf{x}}; \theta_t)$ , where  $\theta_t$  is the weight of teacher and it is frozen during training. Furthermore, the optimization of  $\mathcal{G}$  can be described mathematically by maximizing the likelihood as:

$$\arg \max_{\theta_g} \mathbb{E}_{\mathbf{z}} \log P_{\theta}(\hat{\mathbf{y}}) = \mathbb{E}_{\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})} [\log P_{\theta_t}(\hat{\mathbf{y}}|\hat{\mathbf{x}}) + \log P_{\theta_g}(\hat{\mathbf{x}})]. \quad (2)$$

In Eq. (2),  $P_{\theta}$  represents the probability distribution parameterized by  $\theta$ ,  $P_{\theta_t}(\hat{\mathbf{y}}|\hat{\mathbf{x}})$  can be naturally defined as the output of teacher  $\mathcal{T}$ . Let  $\mathbf{y}'$  as the predefined ground truth of  $\mathbf{z}$  and  $\hat{\mathbf{x}}$ , maximizing  $\mathbb{E}_{\mathbf{z}} \log P_{\theta_t}(\hat{\mathbf{y}}|\hat{\mathbf{x}})$  can be simply approximated via minimizing `CrossEntropy`( $\hat{\mathbf{y}}, \mathbf{y}'$ ). This term ensures that the category of synthesized image  $\hat{\mathbf{x}}$  is consistent with the predefined label. Meanwhile, maximizing  $\log P_{\theta_g}(\hat{\mathbf{x}})$  ensures the synthesized images have a close distribution with  $\mathcal{D}$ . However, it is intractable to compute  $\mathbb{E}_{\mathbf{z}} \log P_{\theta_g}(\hat{\mathbf{x}})$  since we can not estimate the distribution of  $P(\hat{\mathbf{x}}|\mathbf{z})$  directly. Previous methods simply ignore this term and implicitly set  $P_{\theta_g}(\hat{\mathbf{x}})$  to a constant 1 by default, which makes it more difficult to generate realistic images. DeepInversion [29] introduces a feature distribution regularization term to achieve the similar function:

$$\mathcal{L}_{bn} = \sum_l \|\mu_l(\hat{\mathbf{x}}) - \mathbb{E}(\mu_l)\|_2 + \|\sigma_l^2(\hat{\mathbf{x}}) - \mathbb{E}(\sigma_l^2)\|_2, \quad (3)$$

where  $\mu_l(\hat{\mathbf{x}})$  and  $\sigma_l^2(\hat{\mathbf{x}})$  denote the batch-wise mean and variance of feature maps at the  $l$ -th convolutional layer of  $\mathcal{T}$ .

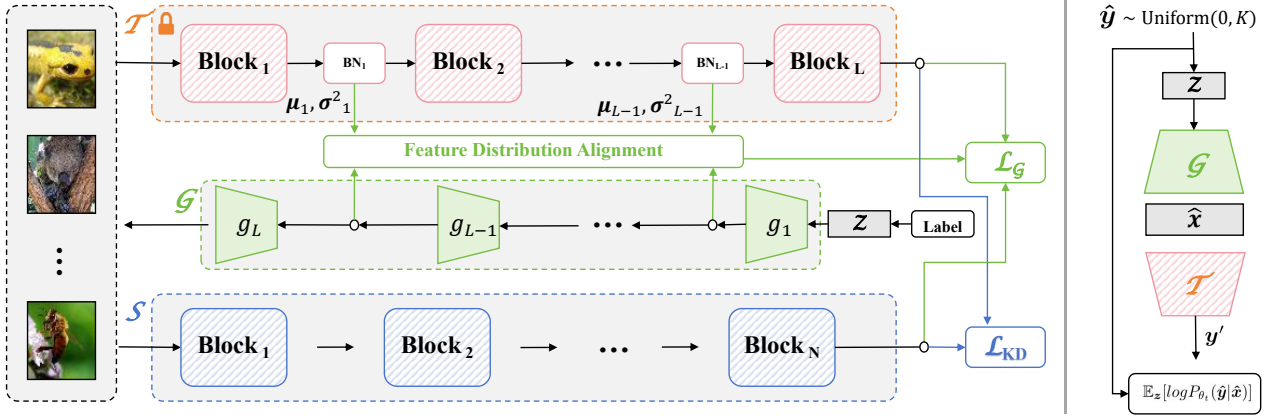


Figure 2. The pipeline of our CPNet framework (left). It consists of the training training of generator  $\mathcal{G}$  (green) and student model  $\mathcal{S}$  (blue). The right part shows the equivalent auto-encoder structure of  $\mathcal{G}$  and  $\mathcal{T}$  in CPNet.

Their expected values  $\mathbb{E}(\mu_l)$  and  $\mathbb{E}(\sigma_l^2)$  can be estimated by the statistics saved in the corresponding `BatchNorm` layer. However, such regularization and other manually designed methods [7, 21] of  $\hat{x}$  are essentially to indirectly constrain  $\mathcal{G}$  by adding prior knowledge, which is hard to generalize to large-scale dataset that the distribution characteristics of samples are complicated.

### 3.2. Coupled Generator

During the training of generator, to fully utilize the information from  $\mathcal{T}$ , instead of concluding priors and encoding them into optimization, we first propose to couple the generator with teacher model and explicitly construct the  $\mathcal{G} = \mathcal{T}^{-1}$  transformation from each  $\mathcal{T}$ . Thus, the derived structure is equivalent to an auto-encoder (left of Figure 2). The term in Eq. (2) can be eliminated directly because the total loss of this auto-encoder is reconstructing input label information and the distribution of hidden variable  $\hat{x}$  can be arbitrary and it does not impact our optimization. Fortunately, since the label distribution of most datasets conforms to a uniform distribution, we can simply compute the total loss using the `CrossEntropy` loss.

Specifically, we uniformly sample labels and employ the LTE in [25] to encode them for input. The advantage of this embedding-formed input lies in its ability to establish a determinate relationship between labels without disrupting the original distribution, while simultaneously preserving rich semantic information for each category. To construct the invertible transform of  $\mathcal{T}$ , we split  $\mathcal{T}$  into  $L$  sequential blocks by taking some `BatchNorm` layers as the separator, *i.e.*,  $\mathcal{T} = \text{blk}_1 \circ \text{blk}_2 \circ \dots \circ \text{blk}_L$ . For the  $l$ -th block  $\text{blk}_l$ , we assign a sub-generator  $g_l$  for it and the complete function generator can be expressed as:

$$\mathcal{G} = \mathcal{T}^{-1} = g_1 \circ g_2 \circ \dots \circ g_L, g_l = \text{blk}_l^{-1}, \quad (4)$$

where we suppose that each sub-generator  $g_l$  corresponding

to the inverse function  $l$ -th block. Like flow-based generative models [4, 14], we construct the whole generator by sequential sub-generators. To satisfy the invertibility of each sub-generator, a simply solution is using  $1 \times 1$  convolution like [14]. However, these invertible modules necessitate maintaining identical dimensions for both input and output features. Consequently, to accommodate the inclusion of upsampling and other more complex operations within the generator, we adopt a feature-level mimicry approach as:

$$\mathcal{L}_{align} = \sum_{l=1}^L \|\mathbf{X}_l - \mathbf{F}_l\|_2^2, \quad (5)$$

where  $\mathbf{F}_l$  denotes the output feature of  $\text{blk}_l$  and  $\mathbf{X}_l$  represents the normalized form of the output features of  $g_l$ , which is achieved by matching the distribution center and variance of the features with the channel-wise `running_mean` and `running_variance` stored in the corresponding `BatchNorm` layer of the  $\mathcal{T}$ :

$$\mathbb{E}(\mu(\mathbf{X}_l)) \rightarrow \text{BN}_l(\text{running\_mean}), \quad (6)$$

$$\mathbb{E}(\sigma^2(\mathbf{X}_l)) \rightarrow \text{BN}_l(\text{running\_variance}). \quad (7)$$

Previous methods have constrained the feature distribution of synthesized images after passing through the `BatchNorm` layer of the teacher model, as shown in Eq. (3). This ensures that the center of the feature distribution of  $\hat{x}$  after each `BatchNorm` layer of  $\mathcal{T}$  aligns with the statistical information of the original training set, effectively narrowing the output space of the generator to a certain range through a process of successive filtering. Building upon this, we further introduce this constraint into each sub-generator  $g_l$ , ensuring that it fits the inverse function of  $\text{blk}_l$  while equivalently incorporating the intermediate layer information of  $\mathcal{T}$  into the generation process of  $\hat{x}$ . This results in a deep coupling of the entire generator  $\mathcal{G}$  with the teacher model  $\mathcal{T}$ .

### 3.3. Adversarial Learning

We now focus on the adversarial training to improve the quality and diversity of synthetic images. To tackle this problem, prior works [8, 25, 29, 30] proposed a basic adversarial loss that encourages the disagreement between student and teacher:

$$\mathcal{L}_{adv} = -\mathbb{E}_{\hat{x} \sim \mathcal{G}(z)}[\mathcal{L}_{KL}(\mathcal{T}(\hat{x}), \mathcal{S}(\hat{x}))], \quad (8)$$

where  $\mathcal{L}_{KL}$  is the Kullback-Leibler Divergence. As we can observe, this term is designed upon final logits of student and teacher. However, as pointed by prior works on traditional knowledge distillation [26, 27], feature-level mimicking provides more effective information from the teacher feature. Thus, in this paper, we introduce an adversarial feature generation method to generate unaligned feature between teacher and student for data diversity. Different from general logit-level adversarial training in Eq. (8), we can not directly obtain the ground truth feature for reference. Thus, we transform the ground truth label  $\mathbf{y}'$  into the corresponding feature constraint for the penultimate layer of  $\mathcal{T}$  by computing the inverse transformation of the classification head.

In the forward pass, let  $\mathbf{X} \in \mathbb{R}^C$  denotes the output feature of penultimate layer in  $\mathcal{T}$ . Notice that we ignore the batch size dimension for easy understanding. Generally, the logit vector  $\mathbf{Y} = [y_1, y_2, \dots, y_K]$  derives from:

$$\mathbf{Y} = \mathbf{X}\mathbf{W}^\top + b, \quad (9)$$

where  $\mathbf{W} \in \mathbb{R}^{K \times C}$  and  $b$  refer to weight and bias of the classification head. Consecutively, we compute each prediction  $p_i$  via the `Softmax` operation, and the relationship between logits is:

$$e^{y_i} = p_i \sum_j^K e^{y_j}. \quad (10)$$

Let the prediction be the same as the ground truth  $\mathbf{y}'$ . Then we transform  $\mathbf{y}'_i = p_i, i = 1, 2, \dots, K$  into the corresponding logit in the backward path. Notice that if we ideally set  $p_j = 0$  for samples that do not belong to the  $j$ -th category, the corresponding logit  $y_j$  will be  $-\infty$ . To avoid numerical overflow, we apply label smoothing for the ground truth label and define that:

$$y_i = \begin{cases} u, & i = k \\ v, & i \neq k \end{cases}, \quad (11)$$

where  $k$  is the pseudo category of  $\hat{x}$  and  $u \gg v$ . By substituting Eq. (11) into Eq. (10), we have:

$$\begin{cases} e^u = p_k(K-1)e^v + p_k e^u, & i = k \\ e^v = (1-p_k)e^v + \frac{1-p_k}{K-1} e^u, & i \neq k \end{cases}. \quad (12)$$

This is the inverse function of the `Softmax` and it depicts the relationship between logits before we derive category

probability in general process. By solving the system of equations in Eq. (12), we find that regardless of the category of input image, there exists a constant margin between the logit of its corresponding category and other logits. We define it as follows:

$$\Delta = u - v = \log\left(\frac{p_k(K-1)}{1-p_k}\right). \quad (13)$$

Now we further compute the inverse transformation of the linear layer (classification head). For simplification, we rewrite the logit vector  $\mathbf{Y}$  as  $\hat{\mathbf{Y}} \cdot \Delta + v$ , where  $\hat{\mathbf{Y}}_i = 1, i = k$  and  $\hat{\mathbf{Y}}_i = 0, i \neq k$ , and then substitute it into Eq. (9). Thus, we can obtain the following simple equation:

$$\mathbf{X} - (\hat{\mathbf{Y}} \cdot \Delta - b)\mathbf{W}^{\top*} = v\mathbf{W}^{\top*}, \quad (14)$$

where  $\mathbf{W}^{\top*} \in \mathbb{R}^{K \times C}$  is the Moore-Penrose inverse of  $\mathbf{W}^\top$ , which satisfies that  $\mathbf{W}^{\top*}\mathbf{W}^\top = \mathbf{I}$  and it is computed using the standard SVD decomposition. Let  $\mathbf{A} = \mathbf{X} - (\hat{\mathbf{Y}} \cdot \Delta - b)\mathbf{W}^{\top*}$  and  $\mathbf{B} = v\mathbf{W}^{\top*}$ . According to Eq. (14), we can easily observe that  $\mathbf{A} \parallel \mathbf{B}$ . For now, we have established that the vector resulting from a linear transformation of the ground truth label and its penultimate layer features aligns collinearly with the weights of the classification layer. We introduce this constraint into the adversarial training. Specifically, we substitute the labels output by the student model in the last iteration as the ground truth labels and the penultimate layer features of the generated samples into Eq. (14). If the resulting  $\mathbf{A}$  and  $\mathbf{B}$  are closer to a perpendicular relationship, it indicates that the distance between them is greater. Therefore, Eq. (8) can be rewritten by:

$$\mathcal{L}_{adv} = -\mathbb{E}_{\hat{x} \sim \mathcal{G}(z)}[\mathcal{L}_{KL}(\mathcal{T}(\hat{x}), \mathcal{S}(\hat{x})) + \beta \cos(A, B)], \quad (15)$$

where  $\cos(\cdot)$  denotes the cosine similarity function that encourage  $\mathbf{A}$  and  $\mathbf{B}$  to be orthogonal,  $\beta$  represent its weight.

### 3.4. Overall Optimization

The whole training pipeline of our CPNet is shown in Figure 2, it keeps two lines to alternatively update the generator and student. The total loss to optimize the generator is:

$$\mathcal{L}_{\mathcal{G}} = \alpha_{cls}\mathcal{L}_{cls} + \alpha_{adv}\mathcal{L}_{adv} + \alpha_{bn}\mathcal{L}_{bn} + \mathcal{L}_{align}, \quad (16)$$

where the  $\mathcal{L}_{cls}$  is the `CrossEntropy` loss, which ensures that the predicted label  $\hat{y} = \mathcal{T}(\hat{x})$  for the image  $\hat{x}$  generated by the generator is consistent with the predefined label  $\mathbf{y}'$ , and  $\alpha_{cls}$ ,  $\alpha_{bn}$ , and  $\alpha_{adv}$  are coefficients to balance three loss terms. During training the student, the distillation loss can be expressed as:

$$\mathcal{L}_{KD} = \mathbb{E}_{\hat{x} \sim \mathcal{G}(z)}[\mathcal{L}_{KL}(\mathcal{T}(\hat{x}), \mathcal{S}(\hat{x}))] + \text{MSE}(\mathbf{F}_{\mathcal{T}}, \mathbf{F}_{\mathcal{S}}), \quad (17)$$

---

**Algorithm 1:** CPNet

---

**Input** : Pre-trained teacher  $\mathcal{T}_{\theta_t}$ , student  $\mathcal{S}_{\theta_s}$ , generator  $\mathcal{G}_{\theta_g}$ ;

**Output** : An optimized student  $\mathcal{S}_{\theta_s}$ .

```
1 Initialize  $\theta_g, \theta_s$  and synthetic dataset  $\mathcal{M} = \emptyset$ ;  
2 for  $M$  iterations do  
3   Sample pseudo label  $\mathbf{y}' \sim \text{Uniform}(0, K)$ ;  
4   Generate embedding  $\mathbf{z}$  according  $\mathbf{y}'$ ;  
5   Re-initialize  $\theta_g$  periodically;  
6   for  $\text{max}_g$  steps do  
7      $\hat{\mathbf{x}} = \mathcal{G}(\mathbf{z})$ ;  
8      $\mathcal{L}_{\mathcal{G}} \leftarrow \alpha_{cls} \mathcal{L}_{cls} + \alpha_{bn} \mathcal{L}_{bn} + \mathcal{L}_{align}$ ;  
9     Update  $\theta_g$  by minimizing  $\mathcal{L}_{\mathcal{G}}$ ;  
10  end  
11   $\mathcal{M} \leftarrow \mathcal{M} \cup \hat{\mathbf{x}}$ ;  
12 end  
13 for  $N$  iterations do  
14   Sample pseudo label  $\mathbf{y}' \sim \text{Uniform}(0, K)$ ;  
15   Generate embedding  $\mathbf{z}$  according  $\mathbf{y}'$ ;  
16   Re-initialize  $\theta_g$  periodically;  
17   for  $\text{max}_g$  steps do  
18      $\hat{\mathbf{x}} = \mathcal{G}(\mathbf{z})$ ;  
19     Update  $\theta_g$  by minizing  $\mathcal{L}_{\mathcal{G}}$  in Eq. (16);  
20   end  
21    $\mathcal{M} \leftarrow \mathcal{M} \cup \hat{\mathbf{x}}$ ;  
22   for  $\text{max}_{kd}$  steps do  
23      $\hat{\mathbf{x}} \sim \mathcal{M}$ ;  
24     Update  $\theta_s$  by minimizing  $\mathcal{L}_{KD}$  in Eq. (17);  
25   end  
26 end
```

---

where MSE denotes the Mean Squared Error and  $\mathbf{F}_{\mathcal{T}}, \mathbf{F}_{\mathcal{S}}$  are penultimate layer features of teacher and student, respectively. As the same as adversarial learning, we also consider the feature-level alignment for a better representation.

The whole algorithm is depicted in Alg 1. To pursuit a stable generation, we first warm up  $\mathcal{G}$  for  $M$  iterations. In this stage, we using  $\mathcal{L}_{\mathcal{G}} = \alpha_{cls} \mathcal{L}_{cls} + \alpha_{bn} \mathcal{L}_{bn} + \mathcal{L}_{align}$  for training since the student has not been updated. After warm up, we conduct an alternative training for the generator and student model. Specifically, we generate synthetic images and update  $\mathcal{G}$  using the loss in Eq. (16) to complete generative and adversarial learning. To avoid over-fitting problem during generation, we periodically re-initialize the weight of  $\mathcal{G}$  to improve diversity.

## 4. Experiments

### 4.1. Experimental Settings

**Dataset and Metrics.** We evaluate CPNet and other DFKD methods on CIFAR10, CIFAR100 [15] and TinyImageNet

[22] due to their balanced size, diverse content, and practical relevance. CIFAR-10 and CIFAR-100 contain 10 and 100 categories, respectively. Both of them comprise 60,000  $32 \times 32$  color images, of which 50,000 images are used for training, and 10,000 images are used for testing. TinyImageNet consists of 200 classes with 5000 images per class, amounting to 100,000 images. The images are  $64 \times 64$  pixels in size, providing a middle ground between CIFAR10 and CIFAR100 in terms of complexity.

**Training Setups.** For a fair comparison, following previous DFKD researches [8, 25, 30], we conduct data-free training on 5 different teacher-student model pairs across various backbone networks including ResNet [11], WideResNet (WRN) [31] and VGG [24]. We set the warm up iteration as 20. For the comparison with SOTA methods, the maximal training epoch is 300, and it will be set to 100 during ablation studies for fast evaluation. If not specified, we use  $\alpha_{bn} = 10, \alpha_{adv} = 1.33, \alpha_{cls} = 0.5$  by default. The re-initialization period of generator is 10 and . All experiments are performed on one Nvidia V100(32 GB) using 16 cores of Intel Xeon Gold 6130R CPU. Our codebase is implemented with the Pytorch library (ver 1.10) and CUDA (ver11.3).

## 4.2. Results and Analysis

### 4.2.1. Comparison with SOTA

As shown in Table 1, we report the top-1 accuracy (%) of our CPNet and other DFKD methods like DeepInversion [29], DAFL [2], ZSKT [20], and CMI [7]. All the listed methods use the same teacher/student network for a fair comparison. Among them, DeepInversion [29] adopt the image-based manner that generates samples by optimizing the noise image. The entire learning process relies on the classification gradients of the generated images after passing through the teacher network and the constraints on the features of the BatchNorm layer. Although the strategy of adversarial learning has been added, due to its fewer tunable parameters, the performance is relatively lower compared to subsequent methods, and it requires a longer training time. Other methods construct a generator separately to produce training samples, and their differences are mainly reflected in the design of the loss function. For example, CMI [7] and DDA [17] employ contrastive learning to improve the diversity of generated samples. However, previous approaches typically decouple the generator and teacher model, thereby missing the opportunity to leverage additional information from the teacher model. Thanks to our coupled generator serves as the inverse transformation of teacher, CPNet surpasses all SOTA methods except the results of ‘R34-R18’ and ‘V11-R18’ pairs on CIFAR10. More surprisingly, we find that CPNet even achieves better performance than the method trained on the original dataset for WideResNet16-2, WideResNet16-1, WideResNet40-1 on CIFAR10, and ResNet18 on CIFAR100.

Table 1. Performance comparison of existing DFKD methods and CPNet. The best results are highlighted in **bold** and the runner-up is underlined. The results of previous methods derives from their original papers. Following [25], ‘R’ indicates ResNet, ‘W’ corresponds to WideResNet, and ‘V’ stands for VGG in this paper. ‘-’ denotes the result is not reported.

Method	CIFAR10					CIFAR100				TinyImageNet	
	R34 R18	W40-2 W16-2	W40-2 W16-1	W40-2 W40-1	V11 R18	R34 R18	W40-2 W16-2	W40-2 W16-1	W40-2 W40-1	V11 R18	R34 R18
Teacher	95.7	94.87	94.87	94.87	92.25	77.94	75.83	75.83	75.83	71.32	66.44
Student	95.2	93.95	91.12	93.94	95.2	77.1	73.56	65.31	72.19	77.1	64.87
DeepInversion [29]	93.26	89.72	83.04	86.85	90.36	61.32	61.34	53.77	68.58	54.13	-
DAFL [2]	92.22	81.55	65.71	81.33	81.1	74.47	43.7	20.88	42.83	54.16	-
DFQ [3]	94.61	92.01	86.14	91.69	90.84	77.01	64.79	51.27	54.43	66.21	-
ZSKT [20]	93.32	89.66	83.74	86.07	89.46	67.74	54.59	36.6	53.6	54.31	-
CMI [7]	94.84	92.52	90.01	92.78	91.13	77.04	68.75	57.91	68.88	70.56	64.01
FastDFKD [8]	94.05	92.45	89.29	92.51	90.53	74.34	65.12	54.02	63.91	67.44	-
MAD [5]	94.9	92.64	-	-	-	77.31	64.05	-	-	-	62.32
Spaceship [30]	<u>95.39</u>	93.25	90.38	93.56	92.27	77.41	69.95	58.06	68.78	71.41	64.04
DDA [17]	<b>95.64</b>	93.51	90.92	93.63	<b>93.02</b>	<u>77.56</u>	70.27	58.96	69.31	72.04	64.13
SSD-KD [18]	94.26	93.11	89.96	93.23	90.67	<u>75.16</u>	65.28	55.61	64.57	68.77	-
NAYER-100 [25]	94.03	93.48	91.12	93.57	91.34	76.29	70.2	59.26	69.89	71.1	61.71
NAYER-300 [25]	95.21	<u>94.07</u>	<u>91.94</u>	<u>94.15</u>	<u>92.37</u>	77.54	<u>71.72</u>	<u>62.23</u>	<u>71.8</u>	<u>71.75</u>	<u>64.17</u>
CPNet-100	94.66	93.51	91.31	93.54	91.5	76.61	70.82	60.98	70.53	70.26	62.45
CPNet-300	95.26	<b>94.09</b>	<b>92</b>	<b>94.22</b>	92.18	<b>77.68</b>	<b>71.9</b>	<b>63.23</b>	<b>71.98</b>	<b>71.8</b>	<b>64.91</b>

Table 2. The training time in GPU hours for compared DFKD methods on CIFAR10 and CIFAR100 with the teacher/student models WRN40-2/WRN16-2. We use the time consumption data closest to our settings in the original paper of SSD-KD since the source code is unavailable up to the time of writing.

	DeepInv	CMI	DAFL	DFQ	ZSKT	Fast5	Fast10	SpaceshipNet	SSD-KD	NAYER-100	CPNet-100
<b>CIFAR-10</b>	89.72	92.52	81.55	92.01	89.66	91.96	92.45	93.25	93.45	93.48	<b>93.51</b>
<b>Time(hours)</b>	11.67	10.58	18.52	59.40	118.82	1.89	2.05	26.15	1.78	1.80	<b>1.39</b>
<b>CIFAR-100</b>	61.34	68.75	43.70	64.79	54.59	63.83	65.12	69.95	65.28	70.20	<b>70.82</b>
<b>Time(hours)</b>	11.68	10.53	18.67	59.48	118.84	1.86	1.95	26.23	<b>1.78</b>	1.82	<b>1.78</b>
<b>Average Speed Up</b>	1×	1.11×	0.63×	0.20×	0.10×	6.23×	5.84×	0.45×	6.56×	6.45×	<b>7.48×</b>

#### 4.2.2. Time Comparison

Some most methods in Table 1 also focus on the generation speed problem, like FastDFKD [8] and SSD-KD [18]. Therefore, we further compared the efficiency of CPNet with them. Specifically, we use the teacher/student combination of WideResNet40-2/WideResNet16-2 and conduct time consumption tests on some typical and efficient DFKD methods, as shown in the Table 2. For rapid verification, we adopted the settings from NAYER and trained for only 100 epochs. The results indicate that CPNet is more efficient than other methods, improving image generation speed by 1.2% to 7.48%. This is also attributed to the architecture of our generator. It is strictly aligned with the intermediate features of the teacher model, making the entire generation process easier. Furthermore, compared to most methods, it requires a smaller size of the input noise embedding.

#### 4.3. Ablation

**Effect of different generators.** We compare our coupled generator with other general used generators, the results is shown in Table 3. The generator used in CMI [7] is consists of two convolution layers with  $3 \times 3$  kernel and two

upsampling layers. Since we only use the generator of CMI, the reported number is lower than the result in Tabel 1. Based on this architecture, the noise layer generator in NAYER [25] further adds more linear layers to transform the input noise to a embedding. In contrast to these approaches, our model adaptively builds convolutional layers and upsampling operations based on the structure of the teacher model, which makes it more efficient and achieves better performance.

Table 3. Comparison of different generators on CIFAR-10 and CIFAR-100 using ResNet-34 as the teacher and ResNet-18 as the student.

Generator	CIFAR10		CIFAR100	
	Teacher: 95.7%	Student: 95.2%	Teacher: 77.94%	Student: 77.1%
CMI [7]	93.84		76.14	
Noisy Layer [25]	94.03		76.29	
Coupled Generator	94.48		76.45	

**Effect of loss terms.** In Table 4, we also ablate each loss term to study their impact on the final performance. From the results, it is evident that feature-level alignment significantly aids our generator in mimicking the inverse transformation of the teacher model, thereby substantially enhancing performance. Additionally, by comparing (c) with (b), it can be concluded that for the image generation, feature-level

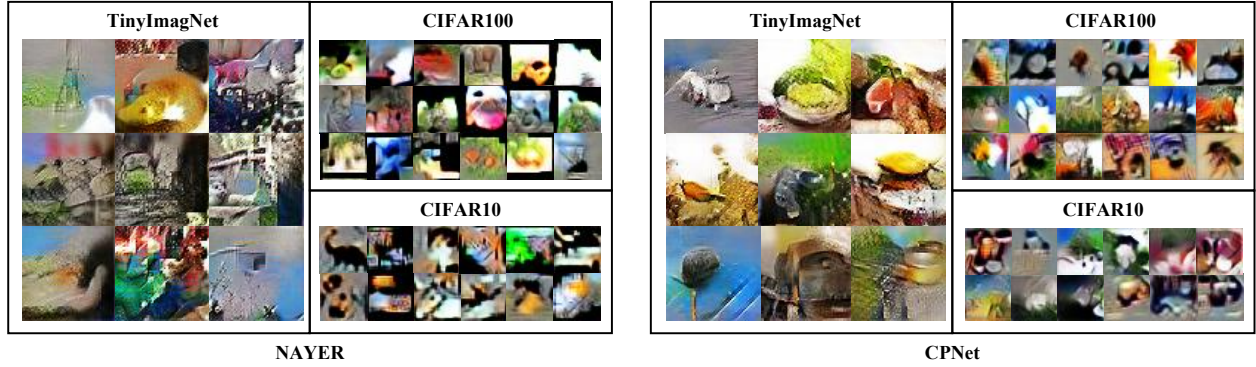


Figure 3. Visualization examples of synthetic images generated by NAYER and our CPNet. Zoom in for more details.

adversarial learning achieves better results compared to targeting only the logits. This is attributed to the theoretically stringent constraints on feature directions. Furthermore, we incorporated consistency between the teacher and student at the penultimate layer features during phase (d), which has been proven effective in previous distillation learning methods, although it was not as pronounced in our experiments.

Table 4. Ablation experiments about different loss terms in our method.

	Generation Loss	Distillation Loss	CIFAR100	
			Teacher: 95.7%	Teacher: 77.94%
(a)	$\mathcal{L}_{bn}, \mathcal{L}_{cls}$	$\mathcal{L}_{KL}$	93.26	61.32
(b)	$\mathcal{L}_{bn}, \mathcal{L}_{cls}, \mathcal{L}_{align}$	$\mathcal{L}_{KL}$	94.56	76.56
(c)	$\mathcal{L}_{bn}, \mathcal{L}_{cls}, \mathcal{L}_{adv}, \mathcal{L}_{align}$	$\mathcal{L}_{KL}$	94.63	76.59
(d)	$\mathcal{L}_{bn}, \mathcal{L}_{cls}, \mathcal{L}_{adv}, \mathcal{L}_{align}$	$\mathcal{L}_{KL}, \text{MSE}$	94.66	76.61

Table 5. Effect of the block count selection.

#Block	1	2	4	6	8
CIFAR10	87.26	91.30	<b>94.66</b>	92.35	90.01
CIFAR100	69.23	72.69	<b>76.61</b>	75.83	71.12

**Sensitivity of the chosen block count.** We show the analysis about the number of chosen block (teacher) in Table 5. The results clearly show that the accuracy of model decrease when the number of chosen blocks is either too small or too large, with the highest accuracy achieved at 4.

**Effect of hyperparameter.** To investigate the importance of the adversarial feature learning term in Eq. (15), we adjust its weight in  $\mathcal{L}_{adv}$  on CIFAR10, the results are shown in Figure 4. As we can observe, for all network architectures, the best performance is achieved when the weight is distributed between 0.7 and 0.8.

#### 4.4. Visualization

Figure 3 shows examples of synthetic images derives from NAYER and our CPNet for the TinyImageNet dataset. Compared to NAYER, our method produces images with more realistic textures and overall quality. It is worth emphasizing

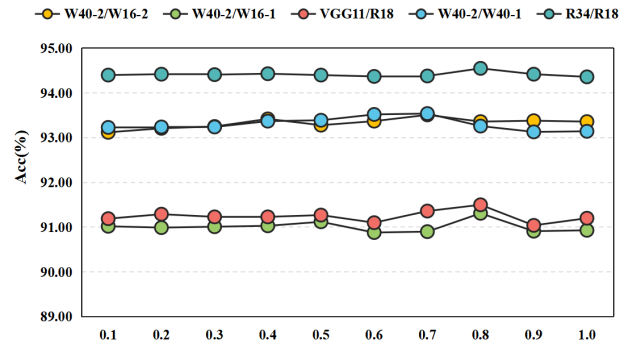


Figure 4. Comparison of different values for the weight  $\beta$  of adversarial feature in  $\mathcal{L}_{adv}$ .

that the primary goal of CPNet is to generate semantically meaningful features for distillation, not purely photorealistic images. Despite the blurriness of synthetic images, they enable student models to achieve results comparable to or even better than those trained on real images.

## 5. Conclusion

We present CPNet, a novel method that addresses the lack of precise data distribution estimation by constructing a generator that explicitly approximates the inverse transformation of the teacher model. Together with the feature-level adversarial learning, it not only achieves superior or competitive performance compared to previous state-of-the-art DFKD methods but also boasts a faster generation speed. This integration allows for the creation of an auto-encoder specifically designed for label information.

## Acknowledgements

This work is supported in part by National Key R&D Program of China (under Grant No.2022YFB3102100) and the National Natural Science Foundation of China (under Grants U22B2027 and 62376186).

## References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 1
- [2] Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Data-free learning of student networks. In *ICCV*, pages 3514–3522, 2019. 1, 2, 6, 7
- [3] Yoojin Choi, Jihwan Choi, Mostafa El-Khamy, and Jungwon Lee. Data-free network quantization with adversarial knowledge distillation. In *CVPRW*, pages 710–711, 2020. 1, 7
- [4] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 2, 4
- [5] Kien Do, Thai Hung Le, Dung Nguyen, Dang Nguyen, HariPriya Harikumar, Truyen Tran, Santu Rana, and Svetha Venkatesh. Momentum adversarial distillation: Handling large distribution shifts in data-free knowledge distillation. *NeurIPS*, 35:10055–10067, 2022. 2, 3, 7
- [6] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 1
- [7] Gongfan Fang, Jie Song, Xinchao Wang, Chengchao Shen, Xingen Wang, and Mingli Song. Contrastive model inversion for data-free knowledge distillation. *arXiv preprint arXiv:2105.08584*, 2021. 1, 3, 4, 6, 7
- [8] Gongfan Fang, Kanya Mo, Xinchao Wang, Jie Song, Shitao Bei, Haofei Zhang, and Mingli Song. Up to 100x faster data-free knowledge distillation. In *AAAI*, pages 6597–6604, 2022. 1, 2, 3, 5, 6, 7
- [9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 2
- [10] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *IJCV*, 129(6):1789–1819, 2021. 1
- [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6
- [12] Geoffrey Hinton. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1
- [13] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2
- [14] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *NeurIPS*, 31, 2018. 2, 4
- [15] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2, 6
- [16] Deng Li, Peng Li, Aming Wu, and Yahong Han. Prototype-guided cross-task knowledge distillation. *Frontiers of Information Technology & Electronic Engineering*, 26(6):912–929, 2025. 1
- [17] Muquan Li, Dongyang Zhang, Tao He, Xiurui Xie, Yuanfang Li, and Ke Qin. Towards effective data-free knowledge distillation via diverse diffusion augmentation. In *ACM MM*, pages 4416–4425, 2024. 6, 7
- [18] He Liu, Yikai Wang, Huaping Liu, Fuchun Sun, and Anbang Yao. Small scale data-free knowledge distillation. In *CVPR*, pages 6008–6016, 2024. 2, 3, 7
- [19] Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*, 2017. 1, 2
- [20] Paul Micaelli and Amos J Storkey. Zero-shot knowledge transfer via adversarial belief matching. *NeurIPS*, 32, 2019. 1, 2, 6, 7
- [21] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. *Google research blog*, 20(14):5, 2015. 4
- [22] Hadi Pouransari and Saman Ghili. Tiny imagenet visual recognition challenge. *CS 231N*, 7, 2014. 2, 6
- [23] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763. PMLR, 2021. 1
- [24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 6
- [25] Minh-Tuan Tran, Trung Le, Xuan-May Le, Mehrtash Harandi, Quan Hung Tran, and Dinh Phung. Nayer: Noisy layer data generation for efficient and effective data-free knowledge distillation. In *CVPR*, pages 23860–23869, 2024. 2, 3, 4, 5, 6, 7
- [26] Guo-Hua Wang, Yifan Ge, and Jianxin Wu. Distilling knowledge by mimicking features. *IEEE TPAMI*, 44(11):8183–8195, 2021. 5
- [27] Jiabao Wang, Yuming Chen, Zhaohui Zheng, Xiang Li, Ming-Ming Cheng, and Qibin Hou. Crosskd: Cross-head knowledge distillation for object detection. In *CVPR*, pages 16520–16530, 2024. 5
- [28] Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE TPAMI*, 44(6):3048–3068, 2021. 1
- [29] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deep-inversion. In *CVPR*, pages 8715–8724, 2020. 1, 2, 3, 5, 6, 7
- [30] Shikang Yu, Jiachen Chen, Hu Han, and Shuqiang Jiang. Data-free knowledge distillation via feature exchange and activation region constraint. In *CVPR*, pages 24266–24275, 2023. 3, 5, 6, 7
- [31] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 6