Data-Free Adversarial Distillation

Gongfan Fang[1], Jie Song[1], Chengchao Shen[1], Xinchao Wang[2], Da Chen[3], Mingli Song[1]

[1]College of Computer Science and Technology, Zhejiang University, Hangzhou, China

[2]Department of Computer Science, Stevens Institute of Technology, New Jersey, United States

[3]Alibaba Group, Hangzhou, China

{fgf, sjie, chengchaoshen, brooksong}@zju.edu.cn,

xinchao.w@gmail.com, chen.cd@alibaba-inc.com

Abstract

Knowledge Distillation (KD) has made remarkable progress in the last few years and become a popular paradigm for model compression and knowledge transfer. However, almost all existing KD algorithms are data-driven, i.e., relying on a large amount of original training data or alternative data, which is usually unavailable in real-world scenarios. In this paper, we devote ourselves to this challenging problem and propose a novel adversarial distillation mechanism to craft a compact student model without any real-world data. We introduce a model discrepancy to quantificationally measure the difference between student and teacher models and construct an optimizable upper bound. In our work, the student and the teacher jointly act the role of the discriminator to reduce this discrepancy, when a generator adversarially produces some "hard samples" to enlarge it. Extensive experiments demonstrate that the proposed data-free method yields comparable performance to existing data-driven methods. More strikingly, our approach can be directly extended to semantic segmentation, which is more complicated than classification and our approach achieves state-of-the-art results. The code will be released.

1. Introduction

Deep learning has made unprecedented advances in a wide range of applications [21, 31, 8, 40] in recent years. Such great achievement largely attributes to several essential factors, including the availability of massive data, the rapid development of the computing hardware, and the more efficient optimization algorithms. Owing to the tremendous success of deep learning and the open-source spirit encouraged by the research fields, an enormous amount of pretrained deep networks can be obtained freely from the Internet nowadays.
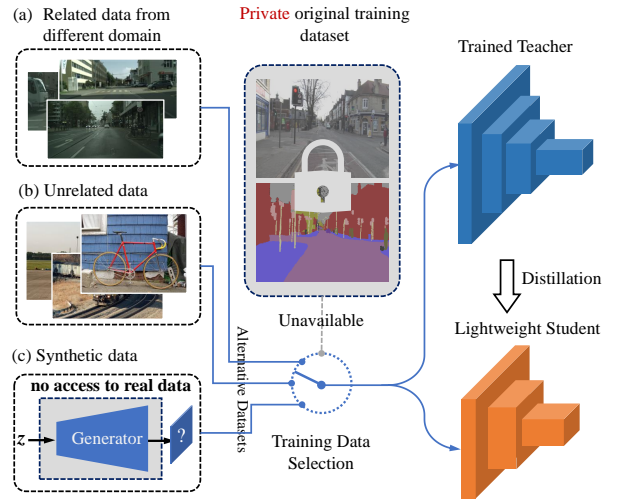

Figure 1: The original training data for pretrained models is usually unavailable to users. In this case, alternative data or synthetic data is used for model compression.

However, many problems may occur when we deploy these pretrained models into real-world scenarios. One prominent obstacle is that the pretrained deep model obtained online is usually large in volume, consuming expensive computing resources that we can not afford with the low-capacity edge devices. A large literature has been devoted to compressing the cumbersome deep models into a more lightweight one, from which Knowledge Distillation (KD) [14] is one of the most popular paradigms. In most existing KD methods, given the original training data or alternative data similar to the original one, a lightweight student model learns from the pretrained teacher by directly imitating its output. We term these methods data-driven KD.

Unfortunately, the training data of released pretrained models are often unavailable due to privacy, transmission, or legal issues, as seen in Figure 1. One strategy to deal with this problem is to use some alternative data [5], but it leads

to a new problem where users are utterly ignorant of the data domain, making it almost impossible to collect similar data. Meanwhile, even if the domain information is known, it is still onerous and expensive to collec a large amount of data. Another compromising strategy in this situation is using somewhat unrelated data for training. However, it drastically deteriorates the performance of the student due to the incurred data bias.

An effective way to avert the problems mentioned above is using synthetic samples, leading to the data-free knowledge distillation [26, 6, 27]. Data-free distillation is currently a new research area where traditional generation techniques such as GANS [11] and VAE [19] can not be directly applied due to the lack of real data. Nayak *et al*. [27] and Chen *et al*. [6] have made some pilot studies on this problem. In Nayak's work [27], some "Data Impressions" are constructed from the teacher model. Besides, in Chen's work [6], they also propose to generate some one-hot samples, which can highly activate the neurons of the teacher model. These exploratory researches achieve impressive results on classification tasks but still have several limitations. For example, their generation constraints are empirically designed based on assumption that an appropriate sample usually has a high degree of confidence in the teacher model. Actually, the model maps the samples from the data space to a very small output space, and a large amount of information is lost. It is difficult to construct samples with a fixed criterion on such a limited space. Besides, these existing data-free methods [6, 27] only take the fixed teacher model into account, ignoring the information from the student. It means that the generated samples can not be customized with the student model.

To avoid the one-sidedness of empirically designed constraints, we propose a data-free adversarial distillation framework to customize training samples for the student model and teacher model adaptively. In our work, a model discrepancy is introduced to demonstrates the functional difference between models. We construct an optimizable upper bound for the discrepancy so that it can be reduced to train the student model. The contributions of our proposed framework can be summarized as three points:

- We propose an adversarial training framework for data-free knowledge distillation. To our knowledge, it is the first approach that can be applied to semantic segmentation.

- We introduce a novel method to quantitatively measure the discrepancy between models without any real data.

- Extensive experiments demonstrate that the proposed method not only behaves significantly superior to data-free methods, and also yields comparable results to some data-driven approaches.

2. Related Work

2.1. Knowledge Distillation (KD)

Knowledge distillation [14] aims at learning a compact and comparable student model from pretrained teacher models. With a teacher-student training schema, it efficiently reduces the complexity and redundancy of the large teacher model. In order to extend the KD framework, researchers have proposed several techniques. According to the requirements of data, we divide those methods into two categories, which are data-driven knowledge distillation and data-free knowledge distillation.

2.1.1 Data-driven Knowledge Distillation

Data-driven knowledge distillation requires real data to extract the knowledge from teacher models. Bucilua *et al*. use a large-scale unlabeled dataset to get pseudo training labels from teacher models [5]. For generalization, Hinton *et al*. propose the concept of Knowledge Distillation (KD) [14]. In KD, the targets, softened by a temperature T, are obtained from the teacher model. The temperature T allows the student model to capture the similarities between different categories.

In order to learn more knowledge, some methods are proposed to utilize intermediate representation as supervision. For example, Romero *et al*. learn a student model by matching the aligned intermediate representation [32]. Moreover, Zagoruyko *et al*. add a constraint of attention matching to let the student network learn similar attention. [39]. In addition to classification tasks [14, 39, 10], knowledge distillation can also be applied to other tasks such as semantic segmentation [25, 17] and depth estimation [29]. Recently, it has also been extended to multitasking [38, 34]. By learning from multiple models, the student model can combine knowledge from different tasks to achieve better performance.

2.1.2 Data-free Knowledge Distillation

The data-driven methods mentioned above are difficult to practice if training data is not accessible. Intuitively, the parameters of a model are independent of its training data. It is possible to distill the knowledge out without real data with data-free methods.

To achieve it, Lopes *et al*. propose to store some metadata during training and reconstruct the training samples during distillation [26]. However, this method still requires metadata during distillation, so it is not completely data-free. Furthermore, Nayak *et al*. propose to craft Data Impressions (DI) as training data from random noisy images [27]. They model the softmax space as a Dirichlet distribution and update random noise images to obtain training data. Another kine of method for data-free distillation

is to synthesize training samples with a generator directly. Chen *et al*. propose DAFL [6], in which the teacher model is fixed as a discriminator [11]. They utilize the generator to construct some training samples, which enable the teacher network to produce highly activated intermediate representations and one-hot predictions.

2.2. Generative Adversarial Networks (GANs)

GANs demonstrate powerful capabilities in image generation [11, 31, 2] for the past few years. It setups a min-max game between a discriminator and a generator. The discriminator aims to distinguish generated data from real ones when the generator is dedicated to generating more realistic and indistinguishable samples to fool the discriminator. Through the adversarial training, GANs can implicitly measure the difference between two distributions. However, GANs are also facing some problems such as training instability and mode collapse [1, 12]. Arjovsky *et al*. propose Wasserstein GAN (WGAN) to make training more stable. WGAN replaces traditional adversarial loss [11] with an approximated Wasserstein distance under 1-Lipschitz constraints so that the gradients of generator will be more stable. Similarly, Qi *et al*. propose to regularize the adversarial loss with Lipschitz regularization [30]. In practical applications, GANs are highly scalable and can be extended to many tasks such as image-to-image translation [40, 16], image super-resolution [37, 24] and domain adaptation [36, 22]. The powerful capabilities theoretically are qualified for sample generation for data-free knowledge distillation.

3. Method

Harnessing the learned knowledge of a pretrained teacher model $\mathcal{T}(x, \theta^t)$, our goal is to craft a more lightweight student model $\mathcal{S}(x, \theta^s)$ without any access to real-world data. To achieve this, we approximates the model $\mathcal{T}$ with a parameterized $\mathcal{S}$ by minimizing the model discrepancy $\mathcal{D}(\mathcal{T}, \mathcal{S})$, which indicates the differences between the teacher $\mathcal{T}$ and the student $\mathcal{S}$. With the discrepancy, an optimal student model can be expressed as follows:

$$\mathcal{S}^* = \min_{\mathcal{S}} \mathcal{D}(\mathcal{T}, \mathcal{S}) \qquad (1)$$

In vanilla data-driven distillation, we design a loss function, e.g., Mean Square Error, and optimize it with real data. The loss function in this procedure can be seen as a specific measurement of the model discrepancy. However, the measurement becomes intractable when the original training data is unavailable. To tackle this problem, we introduce our data-free adversarial distillation (DFAD) framework to approximately estimate the discrepancy so that it can be optimized to achieve data-free distillation.

3.1. Discrepancy Estimation

Given a teacher model $\mathcal{T}(x, \theta^t)$, a student model $\mathcal{S}(x, \theta^s)$ and a specific data distribution p, we firstly define a data-driven model discrepancy $\mathcal{D}(\mathcal{T}, \mathcal{S}; p)$:

$$\mathcal{D}(\mathcal{T}, \mathcal{S}; p) = \mathbb{E}_{x \sim p(x)}[\frac{1}{n}\|\mathcal{T}(x) - \mathcal{S}(x)\|_1] \qquad (2)$$

The constant factor n in Eqn. 2 indicates the number of elements in model output. This discrepancy simply measures the Mean Absolute Error (MAE) of model output across all data points. Note that $\mathcal{S}$ is functionally identical to $\mathcal{T}$ if and only if they produce the same output for any input x. Therefore, if p is a uniform distribution p_u covering the whole data space, we can obtain the true model discrepancy $\mathcal{D}^*$. Optimizing such a discrepancy is equivalent to training with random inputs sampled from the whole data space, which is obviously impossible due to the curse of dimensionality. To avert estimating the intractable $\mathcal{D}^*$, we introduce a generator network $\mathcal{G}(z, \theta^g)$ to control the data distribution. Like in GANs, the generator accepts a random variable z from a distribution p_z and generate a fake sample x. Then the discrepancy can be evaluated with the generator:

$$\mathcal{D}(\mathcal{T}, \mathcal{S}; \mathcal{G}) = \mathbb{E}_{z \sim p_z(z)}[\frac{1}{n}\|\mathcal{T}(\mathcal{G}(z)) - \mathcal{S}(\mathcal{G}(z))\|_1] \quad (3)$$

The key idea of our framework is to approximate $\mathcal{D}^*$ with $\mathcal{D}(\mathcal{T}, \mathcal{S}; \mathcal{G})$. In other words, we estimate the true discrepancy between the teacher model and student with a limited number of generated samples. In this work, we divide the generated samples into two types: "hard sample" and "easy sample". The hard sample is able to produce a relatively larger output differences with the model $\mathcal{T}$ and model $\mathcal{S}$, while the easy sample corresponds to small differences. Suppose that we have a generator $\mathcal{G}^h$ that can always generate hard samples, according to Eqn. 3, we can obtain a "hard sample discrepancy" $\mathcal{D}^h$. Since hard samples always cause large output differences, it is clear the following inequality is true:

$$\mathcal{D}^h = \mathcal{D}(\mathcal{T}, \mathcal{S}; \mathcal{G}^h) \geq \mathcal{D}(\mathcal{T}, \mathcal{S}; p_u) = \mathcal{D}^* \qquad (4)$$

In this inequality, p_u is the uniform distribution covering the whole data space, which comprises a large amount of hard samples and easy samples. Those easy samples make $\mathcal{D}^*$ numerically lower than $\mathcal{D}^h$ that is estimated on hard samples. The inequality is always established when the generated samples are guaranteed to be hard samples. Under this constant, $\mathcal{D}^h$ provides an upper bound for the real model discrepancy $\mathcal{D}^*$. Note that our goal is to optimize the true model discrepancy $\mathcal{D}^*$, which can be achieved by optimizing its upper bound $\mathcal{D}^h$.

However, in the process of training the student model $\mathcal{S}$, hard samples will be mastered by the student and converted
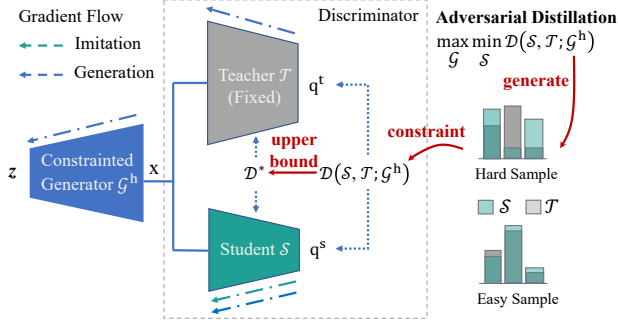
Figure 2: Framework of Data-Free Adversarial Distillation. We construct an upper bound for model discrepancy, under hard sample constraint.

into easy samples. Hence we need a mechanism to push the generator to continuously generate hard samples, which can be achieved by adversarial distillation.

3.2. Advserarial Distillation

In order to maintain the constraints of generating hard samples, we introduce a two-stage adversarial training in this section. Similar to GANs, there is also a generator and a discriminator in our framework. The generator $\mathcal{G}(z, \theta^g)$, as aforementioned, is used to generate hard samples. The student model $\mathcal{S}(x, \theta^s)$, together with the teacher model $\mathcal{T}(x, \theta^t)$ are jointly viewed as the discriminator to measure the hard sample discrepancy $\mathcal{D}(\mathcal{T}, \mathcal{S}; \mathcal{G})$. The adversarial training process consists of two stages: the imitation stage that minimizes the discrepancy and the generation stage that maximize the discrepancy, as shown in Fig. 2.

3.2.1 Imitation Stage

In this stage, we fix the generator $\mathcal{G}$ and only update the student $\mathcal{S}$ in the discriminator. we sample a batch of random noises z from Gaussian distribution and construct fake samples x with the generator $\mathcal{G}$. Then each sample x is fed to both the teacher and the student models to produce the output q^t and q^s. In classification tasks, q is a vector indicating the scores of different categories. In other tasks such as semantic segmentation, q can be a matrix.

Actually, there are several ways to define the discrepancy $\mathcal{D}$ to drive the student learning. Hinton *et al.* utilize the KD loss, which can be KullbackLeibler Divergence (KLD) or Mean Square Error (MSE), to train the student model. These loss functions are very effective in data-driven KD, yet problematic if directly applied to our framework. An important reason is that, when the student converges on the generated samples, these two loss function will produce decayed gradients, which will deactivate the learning of generator, resulting in a dying minmax game. Hence, the Mean Absolute Error (MAE) between q^t and q^s is used as the loss

funtion. Now we can define the loss function for imitation stage as follows:

$$
\begin{aligned}
\mathcal{L}_{IM} &= \mathcal{D}(\mathcal{T}, \mathcal{S}; \mathcal{G}^h) \\
&= \mathbb{E}_{z \sim p_z(z)}[\frac{1}{n}\|\mathcal{T}(\mathcal{G}^h(z)) - \mathcal{S}(\mathcal{G}^h(z))\|_1]
\end{aligned}
\tag{5}
$$

Given the output q_i^s q_i^t, The gradient of $|q_i^s - q_i^t|$ with respect to θ^s is shown in Eqn. 6. It simply multiply the gradients with the sign of $q_i^s - q_i^t$ when q^s is very close to q^t, which provides stable gradients for the generator so that the vanishing gradients can be alleviated.

$$
\nabla_{\theta^s}|q_i^s - q_i^t| = sign(q_i^s - q_i^t)\nabla_{\theta^s}q_i^s
\tag{6}
$$

Intuitively, this stage is very similar to KD, but the goals are slightly different. In KD, students can greedily learn from the soft targets produced by the teacher, as these targets are obtained from real data [14] and contain useful knowledge for the specific task. However, in our setting, we have no access to any real data. The fake samples synthesized by the generator are not guaranteed to be useful, especially at the beginning of training. As aforementioned, the generator is required to produce hard samples to measure the model discrepancy between teacher and student. Another essential purpose of the imitation stage, in addition to learning knowledge from the teacher, is to construct a better search space to force the generator to find new hard samples.

3.2.2 Generation Stage

The goal of the generation stage is to push the generation of hard samples and maintain the constraint for Formula 4. In this stage, we fix the discriminator and only update the generator. It is inspired by the human learning process, where basic knowledge is learned at the beginning, and then more advanced knowledge is mastered by solving more challenging problems. Therefore, in this stage, we encurge the generator to produce more confusing training samples. A straightforward way to achieve this goal is to simply take the negative MAE loss as the objective for optimizing the generator:

$$
\begin{aligned}
\mathcal{L}_{GEN} &= -\mathcal{L}_{IM} \\
&= -\mathbb{E}_{z \sim p_z(z)}[\frac{1}{n}\|\mathcal{T}(\mathcal{G}^h(z)) - \mathcal{S}(\mathcal{G}^h(z))\|_1]
\end{aligned}
\tag{7}
$$

With the generation loss, the error firstly back-propagates through the discriminator, i.e., teacher and the student model, then the generator, yielding the gradients for optimizing the generator. The gradient from the teacher model is indispensable at the beginning of adversarial training, because the randomly initialized student practically

Algorithm 1: Data Free Adversarial Distillation
___
Input: A pretrained teacher model, $\mathcal{T}(x; \theta^t)$
Output: A comparable student model $\mathcal{S}(x; \theta^s)$

1   Randomly initialize a student model $\mathcal{S}(x; \theta^s)$ and a generator $\mathcal{G}(z; \theta^g)$.

2   **for** *number of training iterations* **do**

3   **1. Imitation Stage**

4   **for** *k steps* **do**

5   Generate samples x from z with $\mathcal{G}(z; \theta^g)$;

6   Calculate model discrepancy with Eqn. 5;

7   Update θ^s to **minimize discrepancy** with
$$\nabla_{\theta^s} \mathcal{D}(\mathcal{T}, \mathcal{S}; \mathcal{G}^h)$$

8   **end**

9   **2. Generation Stage**

10   Generate samples x from z with $G(z; \theta^g)$;

11   Calculate negative discrepancy with Eqn. 7;

12   Update θ^g to **maximize discrepancy** with
$$\nabla_{\theta^g} - \mathcal{D}(\mathcal{T}, \mathcal{S}; \mathcal{G}^h)$$

13   **end**
___

provides no instructive information for exploring hard samples.

However, the training procedure with the objective shown in Eqn. 7 may be unstable if the student learning is relatively much slower. By minimizing the objective in Eqn. 7, the generator tends to generate "abnormal" training samples, which produce extremely different predictions when fed to the teacher and the student. It deteriorates the adversarial training process and makes the data distribution change drastically. Therefore, it is essential to ensure the generated samples to be normal. To this end, we propose to take the log value of MAE as an adaptive loss function for the generation stage:

$$\mathcal{L}_{GEN-ADA} = -log(\mathcal{L}_{IM} + 1) \qquad (8)$$

Different from $\mathcal{L}_{GEN}$ which always encourages the generator to produce hard samples with large discrepancy, in the proposed new objective in Eqn. 8, the gradients of the generator are gradually decayed to zero when discrepancy becomes large. It slow down the training of the generator and make training more stable. Without the log term, we have to carefully adjust the learning rate to make the training as stable as possible.

3.2.3 Optimization

Two-stage training. The whole distillation process is summarized in Algorithm 1. Our framework trains the student and the generator by repeating the two stages. It begins with the imitation stage to minimize $\mathcal{L}_{IM}$. Then in the generation stage, we update the generator to maximize $\mathcal{L}_{GEN}$.

Dataset	Model	REL	UNR
MNIST	LeNet-5	SVHN	FMNIST
CIFAR	ResNet	STL10	Cityscapes
Caltech101	ResNet	STL10	Cityscapes
CamVid	DeeplabV3	Cityscapes	VOC2012
NYUv2	DeeplabV3	SunRGBD	VOC2012

Table 1: The datasets and model architectures used in experiments. REL and UNR correspond to related alternative data and unrelated data respectively.

Based on the learning progress of student model, the generator crafts hard samples to further estimate the model discrepancy. The competition in this adversarial game drives the generator to discovers missing knowledge, leading to complete knowledge. After several steps of training, the system will ideally reach a balance point, at which the student model has mastered all hard samples, and the generator is not able to differentiate between the two models $\mathcal{S}$ and $\mathcal{T}$. In this case, $\mathcal{S}$ is functionally identical to $\mathcal{T}$.

Training Stability It is essential to maintain stability in adversarial training. In the imitation stage, we update the student model for k times so as to ensure its convergence. However, since the generating samples are not guaranteed to be useful for our tasks, the value of k cannot be set too large, as it leads to an extraordinarily biased student model. We find that setting k to 5 can make training stable. In addition, we suggest using adaptive loss $\mathcal{L}_{GEN-ADA}$ in dense prediction tasks, such as segmentation, in which each pixel will provide statistical information for adjusting the gradient. In classification tasks, only a few samples are used to calculate the generation loss and the statistical information is not accurate, hence the $\mathcal{L}_{GEN}$ is more prefered.

Sample Diversity Unlike GANs, our approach naturally maintains diversity of generated samples. When mode collapse occurs, it is easy for students to fit these duplicated samples in our framework, resulting in a very low model discrepancy. In this case, the generator is forced to generate different samples to enlarge the discrepancy.

4. Experiments

We conduct extensive experiments to verify the effectiveness of the proposed method, in which knowledge distillation on two types of models are explored: the classification models and the segmentation model.

4.1. Experimental Settings

4.1.1 Models and Datasets

We adopt the following six pretrained models to demonstrate the effectiveness of the proposed method: MNIST [23], CIFAR10 [20], CIFAR100 [20], Caltech101 for classification and CamVid [4, 3], NYUv2 [35] for se-

Method	MNIST		CIFAR10		CIFAR100		Caltech101	
	FLOPs	Accuracy	FLOPs	Accuracy	FLOPs	Accuracy	FLOPs	Accuracy
Teacher	433K	0.989	1.16G	0.955	1.16G	0.775	1.20G	0.766
KD-ORI	139K	0.988 ± 0.001	557M	0.939 ± 0.011	558M	0.733 ± 0.003	595M	0.775 ± 0.002
KD-REL	139K	0.960 ± 0.006	557M	0.912 ± 0.002	558M	0.690 ± 0.004	595M	0.748 ± 0.003
KD-UNR	139K	0.957 ± 0.007	557M	0.445 ± 0.012	558M	0.133 ± 0.003	595M	0.352 ± 0.015
RANDOM	139K	0.747 ± 0.033	557M	0.101 ± 0.002	558M	0.015 ± 0.001	595M	0.010 ± 0.000
DAFL	139K	0.981 ± 0.001	557M	0.885 ± 0.003	558M	0.614 ± 0.005	595M	FAILED
Ours	139K	$\mathbf{0.983 \pm 0.002}$	557M	$\mathbf{0.933 \pm 0.000}$	558M	$\mathbf{0.677 \pm 0.003}$	595M	$\mathbf{0.735 \pm 0.008}$

Table 2: Test accuracy of different distillation methods on several classification datasets.

mantic segmentation. Here the models are named after the corresponding training data.

MNIST. MNIST [23] is a simple image dataset for recognition of handwritten digits containing 60,000 training images and 10,000 test images from 10 categories. Following [26, 6], we use a LeNet-5 as the pretrained teacher model and use a LeNet-5-Half as the student model.
CIFAR10 and **CIFAR100**. CIFAR10 [20] and CIFAR100 both contain 60,000 RGB images. Among them, 50,000 images are used for training and 10,000 for testing. CIFAR10 contains 10 classes when CIFAR100 contains 100 classes. Due to the limitations of the small resolution, we use a modified ResNet-34 [13] as our teacher, which has only three downsample layers. We utilize a ResNet-18 as our student model.
Caltech101. Caltech101 [9] is a classification dataset. There are 101 categories, each of which contains at least 40 images. We randomly split the dataset into two parts: a training set with 6982 images and a test set with 1695 images. During training, the images are resized and cropped to 128×128. We use the standard ResNet-34 architecture as the teacher model and use ResNet-18 as the student model.
CamVid. Camvid [4, 3] is a road scene segmentation dataset, consisting of 367 training and 233 testing RGB images. There are 11 categories in CamVid, such as road, cars, poles, traffic lights, etc. The original resolution of images is 720×960. Due to the difficulty in generating high-resolution images, we resize the short side to 256 and train our teacher with 128×128 random crop. The teacher model is a DeepLabV3 [7] model with ResNet-50 [13] as backbone. For student model, we adopt a Mobilenet-V2 [15] as the backbone.
NYUv2. The NYUv2 [35] is collected for indoor scene parsing. It provides 1449 labeled RGB-D images with 13 categories and 407024 unlabeled images. We use 795 pixel-wise labeled images to train our teacher and use the left 654 images as the test set. Similar to CamVid, we also resize and crop the images to 128×128 blocks for training and use the DeeplabV3 as our model architecture.

4.1.2 Implementation Details and Evaluation Metrics

Our method is implemented with Pytorch [28] on an NVIDIA Titan Xp. For training, We use SGD with momentum 0.9 and weight decay 5e-4 to update student models. The generator in our method is trained with Adam [18]. During training, the learning rates of SGD and Adam are decayed by 0.1 for every 100 epochs. In order to measure function discrepancy, we use a large batch size for adversarial training. The batch size is set to 512 for MNIST, 256 for CIFAR, and 64 for other datasets. In our experiments, all models are randomly initialized except that, in semantic segmentation tasks, the backbone of teacher models are pretrained on ImageNet [21]. More detailed hyperparameter settings for different datasets can be found in supplementary materials.

To evaluate our methods, we take the accuracy of prediction as our metric for classification tasks. Furthermore, for semantic segmentation, we calculate Mean Intersection over Union (mIoU) on the whole test set.

4.1.3 Baselines

A bunch of baselines is compared to demonstrate the effectiveness of our proposed method, including both data-driven and data-free methods. The baselines are briefly described as follows.
Teacher: the given pretrained model which serves as the teacher in the distillation process.
KD-ORI: the student trained with the vanilla KD [14] method on original training data.
KD-REL: the student trained with the vanilla KD on an alternative data set which is similar to the original training data.
KD-UNR: the student trained with the vanilla KD on an alternative data set which is unrelated to the original training data.
RANDOM: the student trained with randomly generated noise images.
DAFL: the student trained with DAta-Free Learning [6] without any data without any real data.

4.2. KD in Classification Models

The testing accuracy of our methods and the compared baselines are provided in Table 2. In order to eliminate the effects of randomization, we repeat each experiment for 5 times and record average value and standard deviation of the highest accuracy. The first part of the tables gives the results on data-driven distillation methods. KD-ORI requires the original training data when KD-REL and KD-UNR use some unlabeled alternative data for training. In KD-REL, the training data should be similar to the original training data. However, the domain different between alternative and original data is unavoidable, which will result in incomplete knowledge. As shown in the table, the accuracy of KD-REL is slightly lower than KD-ORI. Note that in our experiments, the original training data is available, so that we can easily find some similar data for training. Nevertheless, in the real world, we are ignorant of the domain information, which makes it impossible to collect similar data. In this case, the blindly collected data may contain many unrelated samples, leading to the KD-UNR methods. The incurred data bias makes training very difficult and deteriorates the performance of the student model.

The second part of the table shows the results of data-free distillation methods. We compare our methods with DALF [6] using their released code. In our experiment, we set the batch size to 256 for CIFAR and 64 for Caltech101 train each model for 500 epochs. Our adversarial learning method achieves the highest accuracy among the data-free methods, and the performance is even comparable to those data-driven methods. Note that we set the batch size of Caltech101 to 64, DAFL methods failed in this case when our method is still able to learn a student model from the teacher. The influence of different batch sizes can be found in supplementary materials.

Visualization of Generated Samples. The generated samples and real samples are shown in Figure 3. The images in the first row are produced by the generator during adversarial learning, and the real images are listed in the second row. Although those generated samples are not recognizable by humans, they can be used to craft a comparable student model. It means that using realistic samples are not the only way for knowledge distillation. Comparing the generated samples on CIFAR10 and CIFAR100, we can find that the generator on CIFAR100 produces more complicated samples than on CIFAR10. As the difficulty of classification increases, the teacher model becomes more knowledgeable so that, in adversarial learning, the generator can recover more complicated images. As mentioned above, the diversity of generated samples are guaranteed by the adversarial loss. In our results, the generator does maintain a perfect image diversity, and almost every generated image is different.
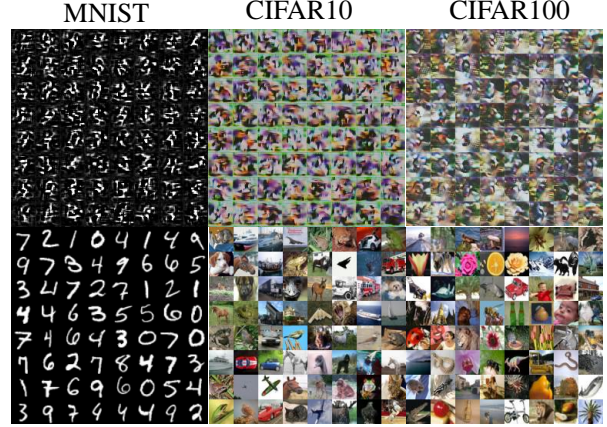

Figure 3: Generated samples on MNIST, CIFAR10 and CIFAR100. The images in the second row are sampled from real data.
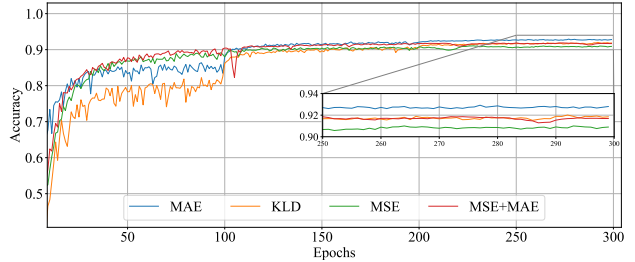

Figure 4: The accuracy curve of different loss functions on CIFAR10. MAE achieves the best performance among those loss candidates.

Comparison between Loss Functions. It is essential to keep the balance of the adversarial game. An appropriate adversarial loss should provide stable gradients during training. In this experiment, four candidates are explored, which are MAE, MSE, KLD, and MSE+MAE. By comparing the accuracy curves of the different loss function, we find that MAE indeed provides the best results owing to its stable gradient for generator.

4.3. KD in Segmentation Models

Our method can be naturally extended to semantic segmentation tasks. In this experiment, we adopt ImageNet-pretrained ResNet-50 to initialize the teacher model and train all student models from scratch. All Models in data-driven methods are trained with 128×128 cropped images. For data-free methods, 128×128 images are directly generated for training. Table 3 shows the performance of the student model obtained with different methods. We can see that, on CamVid, our method obtains a competitive student model even compared with KD-ORI, which requires the original training data. On NYUv2, our approach goes beyond KD-UNR and all data-free methods, although not

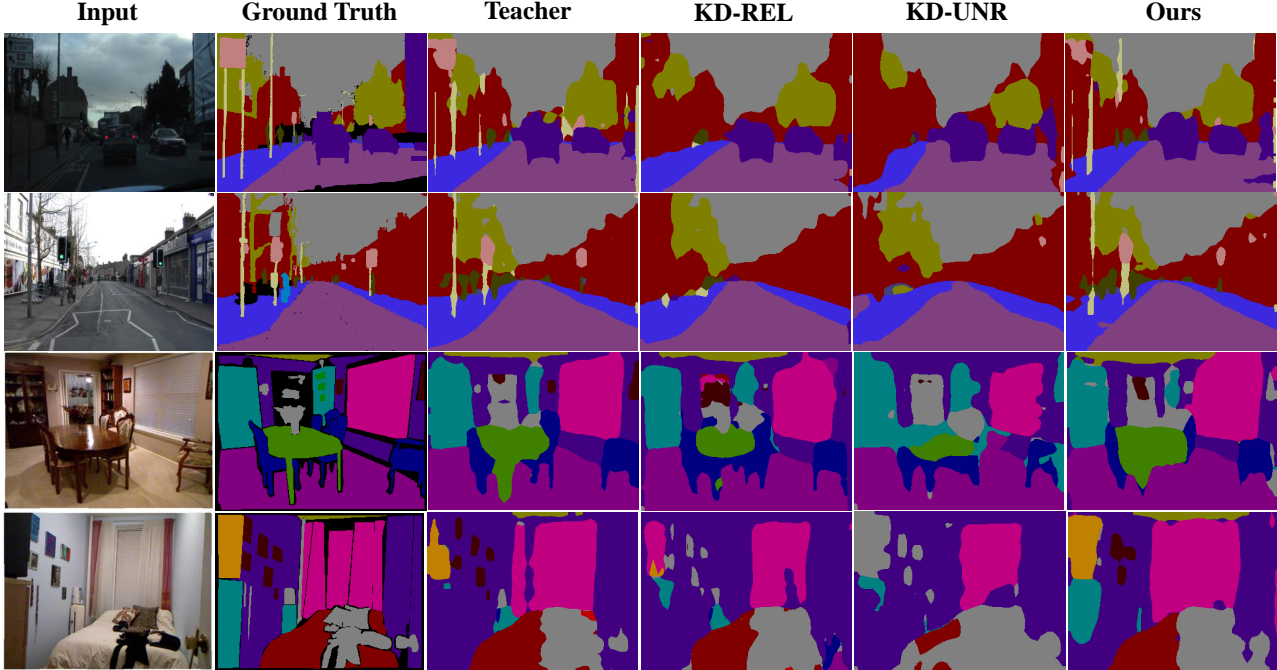| Input | Ground Truth | Teacher | KD-REL | KD-UNR | Ours |

Figure 5: Segmentation results on CamVid and NYUv2. All baseline methods in the figure are data-driven and our framework achieves the best performance when the original training data is not available.

	CamVid		NYUv2	
Method	**FLOPs**	**mIoU**	**FLOPs**	**mIoU**
Teacher	41.0G	0.594	41.0G	0.517
KD-ORI	5.54G	0.535	5.54G	0.380
KD-REL	5.54G	0.475	5.54G	0.396
KD-UNR	5.54G	0.406	5.54G	0.265
RANDOM	5.54G	0.018	5.54G	0.021
DAFL	5.54G	0.010	5.54G	0.105
Ours	5.54G	**0.535**	5.54G	**0.364**

Table 3: The mIoU of DeepLabv3 model on CamVid and NYUv2. The teacher model are pretrained on ImageNet when the student are randomly initialized.

comparable to KD-ORI. In fact, our method is the first data-free distillation method proposed to work on segmentation tasks.

The main difficulty DAFL encounters is that the one-hot constraint is detrimental to segmentation tasks, in which each pixel has strong correlations with its neighboring ones. In our framework, the generator are encouraged to produce complicated patterns by combining multiple pixels to make the game more challenging. As shown in Figure 6, the generator for CamVid indeed catches the co-occurrence of traffic lights and poles with reasonable spatial correlations. To further study these generated samples, we also conduct an experiment to train a student model with a fixed generator obtained from adversarial distillation and train a student

| Generated Semantics | Real Semantics |

Figure 6: The generated samples on CamVid, as well as their semantics predicted by the teacher model. The co-occurrence of traffic lights and poles, as in real samples, is captured by generator.

model with mIoU of 0.460. It demonstrates that the generator indeed learns "what should be generated."

5. Conclusions

This paper introduces a data-free adversarial distillation framework for model compression. We propose a novel method to estimate the optimizable upper bound of the intractable model discrepancy between the teacher and the student. Without any access to real data, we successfully reduce the discrepancy by optimizing the upper bound and obtain a comparable student model. Our experiments on classification and segmentation demonstrate that our frame-

work is highly scalable and can be effectively applied to different network architectures. To the best of our knowledge, it is also the first effective data-free method for semantic segmentation. However, it is still very difficult to generate complicated samples. We believe that introducing human priori can effectively improve the generator by avoding useless search space. In the future, we will explore the impact of different prior information on the proposed adversarial distillation framework.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223, 2017. 3

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 3, 11

[3] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009. 5, 6

[4] Gabriel J Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *European conference on computer vision*, pages 44–57. Springer, 2008. 5, 6

[5] Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM, 2006. 1, 2

[6] 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. *arXiv preprint arXiv:1904.01186*, 2019. 2, 3, 6, 7

[7] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 6, 11

[8] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014. 1

[9] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006. 6

[10] Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. *arXiv preprint arXiv:1805.04770*, 2018. 2

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 2, 3

[12] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017. 3

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6, 11

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1, 2, 4, 6

[15] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 6

[16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 3

[17] Jianbo Jiao, Yunchao Wei, Zequn Jie, Honghui Shi, Rynson WH Lau, and Thomas S Huang. Geometry-aware distillation for indoor semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2869–2878, 2019. 2

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6, 11

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 5, 6

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1, 6

[22] Vinod Kumar Kurmi, Shanu Kumar, and Vinay P Namboodiri. Attending to discriminative certainty for domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 491–500, 2019. 3

[23] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998. 5, 6

[24] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017. 3

[25] Yifan Liu, Ke Chen, Chris Liu, Zengchang Qin, Zhenbo Luo, and Jingdong Wang. Structured knowledge distillation for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2604–2613, 2019. 2

[26] Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*, 2017. 2, 6

[27] Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, R Venkatesh Babu, and Anirban Chakraborty. Zero-shot knowledge distillation in deep networks. *arXiv preprint arXiv:1905.08114*, 2019. 2

[28] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 6

[29] Andrea Pilzer, Stephane Lathuiliere, Nicu Sebe, and Elisa Ricci. Refine and distill: Exploiting cycle-inconsistency and knowledge distillation for unsupervised monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9768–9777, 2019. 2

[30] Guo-Jun Qi. Loss-sensitive generative adversarial networks on lipschitz densities. *arXiv preprint arXiv:1701.06264*, 2017. 3

[31] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 1, 3, 11

[32] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014. 2

[33] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 11

[34] Chengchao Shen, Mengqi Xue, Xinchao Wang, Jie Song, Li Sun, and Mingli Song. Customizing student networks from heterogeneous teachers via adaptive knowledge amalgamation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3504–3513, 2019. 2

[35] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012. 5, 6

[36] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017. 3

[37] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018. 3

[38] Jingwen Ye, Yixin Ji, Xinchao Wang, Kairi Ou, Dapeng Tao, and Mingli Song. Student becoming the master: Knowledge amalgamation for joint scene parsing, depth estimation, and more. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2829–2838, 2019. 2

[39] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016. 2

[40] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 1, 3

Supplementary Material

This supplementary material is organized as follows: Sec. A provides model architectures and implementation details for each dataset; Sec. B presents the influence of different batch sizes on our proposed method. Sec. C visualizes more generated samples and segmentation results.

A. Model Architectures and Hyperparameters

Table 7 summarizes the basic configurations for each dataset. In our experiments, teacher models are obtained from labeled data, when student models and generators are trained without access to real-world data. We validate our models every 50 iterations. For the sake of simplicity, we regard such a period as an **"epoch"**.

A.1. Generators

As illustrated in Fig. 4, two kinds of vanilla generator architectures are aodpted in our experiments. The first generator, denoted as **"Generator-A"**, uses Nearest Neighbor Interpolation for upsampling. The second one, denoted as **"Generator-B"**, is isomorphic to the generator proposed by DCGAN [31], which replaces the interpolations with deconvolutions. We use the Generator-A for MNIST and CIFAR, and apply the more powerful Generator-B to other datasets. The slope of LeakyReLU is set to 0.2 for more stable gradients. In distillation, all generators are optimized with Adam [18] with a learning rate of 1e-3. The betas are set to their default values, which are 0.9 and 0.999.

A.2. Teachers and Students

MNIST. Table 5 provides detailed information about the architectures of LeNet-5 and LeNet-5-Half. In distillation, we use SGD with a fixed learning rate of 0.01 to train the student model for 40 epochs.
CIFAR10 and CIFAR100. The modified ResNet [13] architectures with $8\times$ downsampling for CIFAR10 and CIFAR100 are presented in Table 6. The learning rate starts from 0.1 and is divided by 10 at 100 epochs and 200 epochs. We apply a weight decay of 5e-4 and train the student model for 500 epochs.
Caltech101. The standard ResNet [13] architectures with $32\times$ downsampling are adopted for Caltech101. During training, the learning rate of SGD is assigned as 0.05 and is decayed every 100 epochs. The student model and generator are optimized for 300 epochs with a weight decay of 5e-4.
CamVid. We use DeepLabV3 [7] with dilated convolutions to tackle this segmentation problem. In distillation, we construct a MobileNet-V2 [33] student model and optimize it for 300 epochs using SGD with a learning rate of 0.1 and a weight decay of 5e-4. The learning rate of SGD and Adam is decayed every 100 epochs.

Generator-A	Generator-B
FC, Reshape, BN	FC, Reshape, BN
Upsample $\uparrow_{2\times}$	3×3 512 Deconv $\uparrow_{2\times}$, BN, LReLU
3×3 128 Conv, BN, LReLU	3×3 256 Deconv $\uparrow_{2\times}$, BN, LReLU
Upsample $\uparrow_{2\times}$	3×3 128 Deconv $\uparrow_{2\times}$, BN, LReLU
3×3 64 Conv, BN, LReLU	3×3 64 Deconv $\uparrow_{2\times}$, BN, LReLU
3×3 3 Conv, Tanh, BN	3×3 3 Conv, Tanh

Table 4: Generator Architectures. The vector input is firstly projected to a feature maps [31] and then upsampled to the required size.

Output Size	LeNet-5	LeNet-5-Half
14×14	5×5 6 Conv, ReLU maxpool $\downarrow_{2\times}$	5×5 3 Conv, ReLU maxpool $\downarrow_{2\times}$
5×5	5×5 16 Conv, ReLU maxpool $\downarrow_{2\times}$	5×5 8 Conv, ReLU maxpool $\downarrow_{2\times}$
1×1	5×5 120 Conv, ReLU	5×5 60 Conv, ReLU
1×1	84 FC	42 FC
1×1	Output FC	

Table 5: LeNet-5 architectures for MNIST.

Output Size	ResNet-18-8x	ResNet-34-8x
32×32	3×3 64 Conv, BN, ReLU	
32×32	[3×3 64 Conv, BN, ReLU; 3×3 64 Conv, BN, ReLU] ×2	[3×3 64 Conv, BN, ReLU; 3×3 64 Conv, BN, ReLU] ×3
16×16	[3×3 128 Conv, BN, ReLU; 3×3 128 Conv, BN, ReLU] ×2	[3×3 128 Conv, BN, ReLU; 3×3 128 Conv, BN, ReLU] ×4
8×8	[3×3 256 Conv, BN, ReLU; 3×3 256 Conv, BN, ReLU] ×2	[3×3 256 Conv, BN, ReLU; 3×3 256 Conv, BN, ReLU] ×6
4×4	[3×3 512 Conv, BN, ReLU; 3×3 512 Conv, BN, ReLU] ×2	[3×3 512 Conv, BN, ReLU; 3×3 512 Conv, BN, ReLU] ×3
1×1	Average Pool	
1×1	Output FC	

Table 6: ResNet architectures with $8\times$ downsampling for CIFAR10 and CIFAR100.

NYUv2. We adopt the same architectures and hyperparameters as NYUv2 for NYUv2, except that the learning rate of SGD is modified to 0.05 and the weight decay is reduced to 5e-5. We train the student model and the generator for 300 epochs and multiply the learning rate by 0.3 at 150 epochs and 250 epochs.

B. Influence of Different Batch Sizes

In our method, a large batch size is required to train the generator [2] and ensure the accuracy of the discrepancy estimation. To explore the influence of different batch sizes, we conduct several experiments on classification and semantic segmentation datasets. As illustrated in Fig. 7, a small batch size injures the performance of student models. We also found that increasing batch size can bring tremen-

Dataset	Teacher	Student	Generator	Input Size	Batch Size	lr$_S$	lr$_G$	wd
MNIST	LeNet-5	LeNet-5-Half	Generator-A	32×32	512	0.01		-
CIFAR10	ResNet34-8x	ResNet18-8x	Generator-A	32×32	256	0.1		5e-4
CIFAR100	ResNet34-8x	ResNet18-8x	Generator-A	32×32	256	0.1	1e-3	5e-4
Caltech101	ResNet34	ResNet18	Generator-B	128×128	64	0.05		5e-4
CamVid	DeepLabv3-ResNet50	DeepLabv3-MobileNetV2	Generator-B	256×341	64	0.1		5e-4
NYUv2	DeepLabv3-ResNet50	DeepLabv3-MobileNetV2	Generator-B	256×341	64	0.1		5e-5

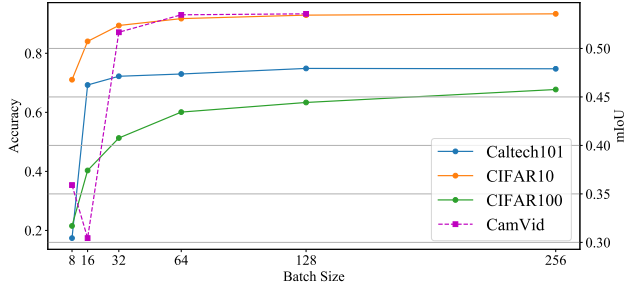Table 7: Configurations and hyperparameters for different datasets.

Figure 7: The influence of different batch sizes on our method.

dous benefits to our method. An important reason for the phenomenon is that the large batch size can provide sufficient statistical information for hard sample generation and make the training more stable.

C. More Visualization

We provide more visualization results in this section. Fig. 8 and Fig. 9 compare the generated samples with real samples on classification datasets. Those fake samples can not be recognized by humans, but indeed contain sufficient knowledge for their tasks. Fig. 10 provides some generated samples on segmentation datasets, as well as the predictions produced by teacher models. To further demonstrate the effectiveness of our method, we provide more segmentation results in Fig. 11 and Fig. 12. As in the main paper, our data-free method is compared with several data-driven baselines, such as KD-REL and KD-UNR. KD-REL requires related data and KD-UNR uses some unrelated data.

Figure 8: Generated samples (left) and real samples (right) from MNIST, CIFAR10 and CIFAR100.
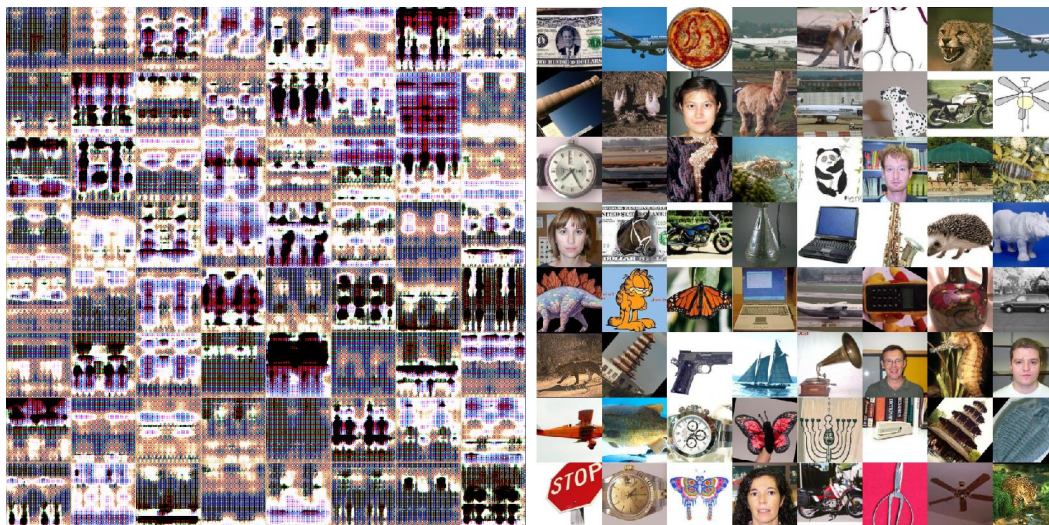
Figure 9: Generated samples (left) and real samples (right) from Caltech101.
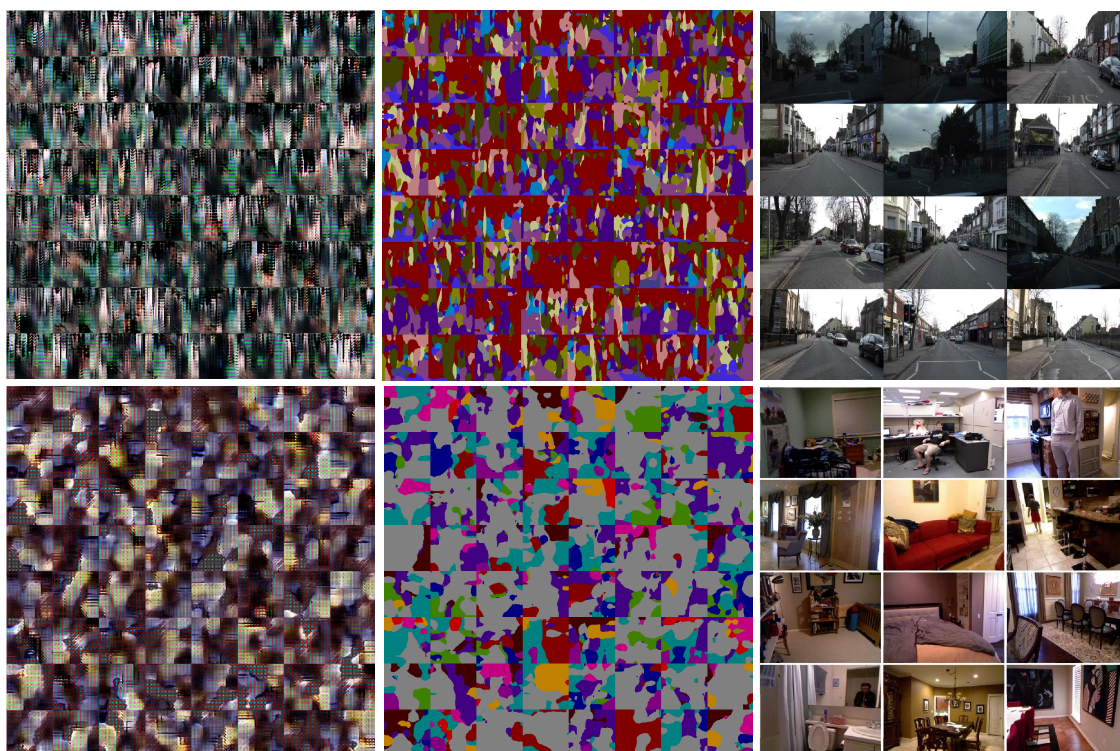

Figure 10: Generated samples (left), teacher predictions (middle) and real samples (right) from CamVid and NYUv2.
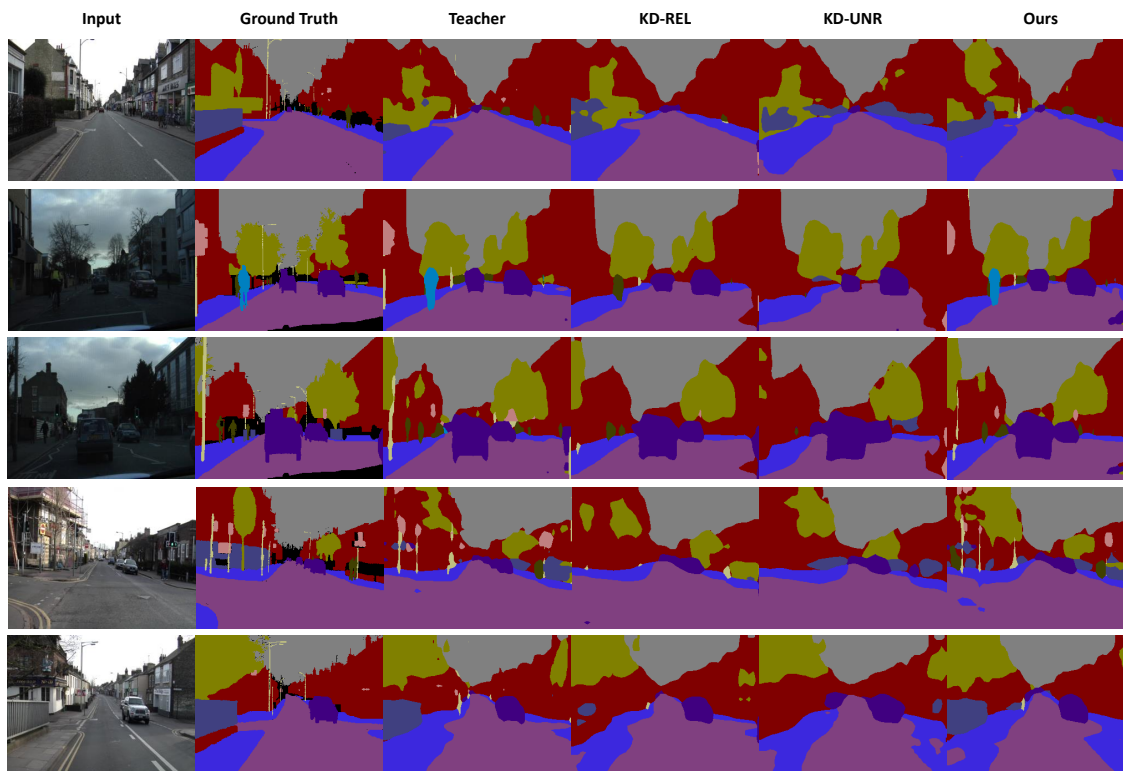
Figure 11: Segmentation results on CamVid. KD-REL uses Cityscapes as training data and KD-UNR adopts VOC2012 as an alternative.
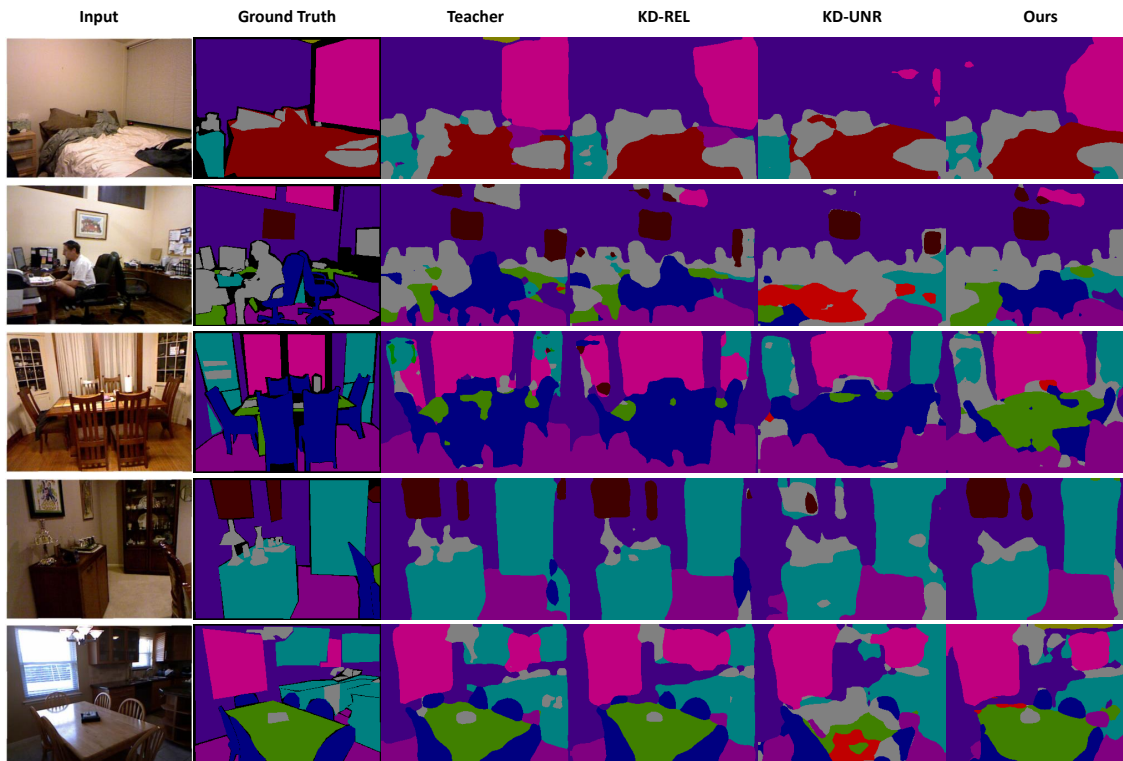

Figure 12: Segmentation results on NYUv2. KD-REL uses SunRGBD as training data and KD-UNR adopts VOC2012 as an alternative.