Black-box Few-shot Knowledge Distillation

Dang Nguyen, Sunil Gupta, Kien Do, Svetha Venkatesh

Applied Artificial Intelligence Institute (A^2I^2), Deakin University, Geelong, Australia
{d.nguyen, sunil.gupta, k.do, svetha.venkatesh}@deakin.edu.au

Abstract. Knowledge distillation (KD) is an efficient approach to transfer the knowledge from a large "teacher" network to a smaller "student" network. Traditional KD methods require lots of *labeled* training samples and a *white-box* teacher (parameters are accessible) to train a good student. However, these resources are not always available in real-world applications. The distillation process often happens at an external party side where we do not have access to much data, and the teacher does not disclose its parameters due to security and privacy concerns. To overcome these challenges, we propose a black-box few-shot KD method to train the student with *few unlabeled* training samples and a *black-box* teacher. Our main idea is to expand the training set by generating a diverse set of out-of-distribution synthetic images using MixUp and a conditional variational auto-encoder. These synthetic images along with their labels obtained from the teacher are used to train the student. We conduct extensive experiments to show that our method significantly outperforms recent SOTA few/zero-shot KD methods on image classification tasks. The code and models are available at: https://github.com/nphdang/FS-BBT

1 Introduction

Despite achieving many great successes in real-world applications [11,34,43], deep neural networks often have millions of weights to train, thus require heavy computation and storage [31]. To make deep neural networks smaller and applicable to real-time devices, especially for edge devices with limited resources, knowledge distillation (KD) methods have been proposed [17,2,10].

The main goal of KD is to transfer the knowledge from a large pre-trained network (called *teacher*) to a smaller network (called *student*) so that the student can perform as well as the teacher [17,36]. Most of existing KD methods follow the idea introduced by Hinton et al. [17], which suggests to use both the ground-truth labels and the teacher's predictions as training signals for the student. The intuition behind this approach is that if the student network not only learns from its training data but also is guided by a powerful teacher network pre-trained on a large-scale data, then the student will improve its classification accuracy.

The success of existing KD methods relies on two strong assumptions. First, the student's training set must be *very large and labeled* (it is usually the same as the teacher's training set) [2,17,19,36]. Second, the teacher is a *white-box* model so that the student has access to the teacher's internal details (e.g. gradient,

parameters, feature maps, logits) [1,40,8,3]. However, these assumptions rarely hold in real-world applications. Typically, the distillation happens at an external party side where we can only access to few unlabeled samples. For example, DeepFace [35] developed by Facebook was trained on 4 million non-public facial images. For distilling a student network from DeepFace, an external party may not have access to the face database used by Facebook due to various reasons including privacy. Instead, its training set would typically comprise of a few thousands images that are accessible at the external party side. In some cases, the pre-trained teacher models are *black-box* i.e. they are released without disclosing their parameters, which is often the case with cloud-deployed machine learning web-services. For example, IBM Watson Speech-to-Text [32] only provides its APIs to end-users to convert audio and voice to written text.

To mitigate the demand of large training data, several few-shot KD methods were proposed for KD with few samples [3,20], but they still require a white-box teacher. To the best of our knowledge, there is only one method named BBKD [37] to train the student with few samples and a black-box teacher. BBKD uses MixUp to synthesize training images and active learning to select the most uncertain mixup images to query the teacher model. Although BBKD shows significant improvements over current SOTA few/zero-shot KD methods, it exhibits two notable limitations. First, it has to synthesize *a huge pool of candidate images*. For example, given $N = 1000$ original images, it constructs $C = 10^6$ candidate images, and selects $M = 20000$ synthetic images from C to train the student. Since the number of candidate images C is very large, it requires expensive computation and consumes large memory resource. Second, it has to train the student multiple times until a stopping criteria. Although the student network is smaller than the teacher network, it is still a deep neural network. Training the student multiple times must be avoided since it costs both resources and training time. *Therefore, few-shot KD with a black-box teacher in a resource- and time-efficient manner is an open problem.*

Our method. To solve the above problem, we propose a novel *unsupervised black-box few-shot* KD method i.e. training the student with only *few unlabeled* images and a *black-box* teacher. Our method offers a resource- and time-efficient KD process, which addresses the bottlenecks of BBKD. First, it does not need to create any pool of candidate images; instead it directly generates M synthetic images from N original images to train the student. Second, it only trains the student network in one-pass; no active learning is required and no multiple student models are repeatedly created.

Our method has three main steps. First, we generate synthetic images from a given *small* set of original images. Second, the synthetic images are sent to the teacher model to query their *soft-labels* (i.e. class probabilities). Finally, the original and synthetic images along with their soft-labels are used to train the student network. Our method is illustrated in Figure 1.

The key component in our method is the image generator, where we propose two approaches to generate synthetic images. First, we use the MixUp method [18,12,4] to synthesize a virtual image by a weighted combination of two original

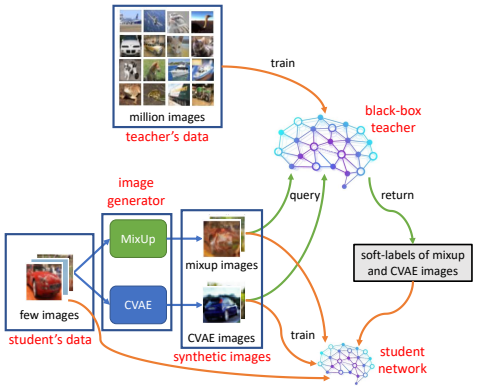

Fig. 1. Knowledge distillation with few samples and black-box teacher. Given a black-box teacher and a small set of original images, we propose to employ MixUp method and CVAE generative model to generate synthetic images to train the student network.

images. Mixup images help us to cover the manifold of natural images. Second, we use Conditional Variational Autoencoder (CVAE) [33] – a generative model to generate additional synthetic images. While MixUp is useful to some extent, mixup images too close to original images do not add much value to the training data. Such disqualified mixup images are replaced by images generated from CVAE. Using CVAE, we can generate interpolated images i.e. the output image semantically mixes characteristics from the original images. As a result, we can enrich the training set and improve the diversity of training images, which is very useful when training the student network.

Our contribution. To summarize, we make the following contributions.

1. We propose **FS-BBT** (*knowledge distillation with Few Samples and Black-Box Teacher*), a novel method offers a successful KD process even with few unlabeled training samples and a black-box teacher model.
2. We develop an efficient approach to train the student network in resource- and time-efficient manner, where we do not need to create a large pool of candidate images and only train the student network one time.
3. We empirically validate our proposed method on several image classification tasks, comparing it with both standard and SOTA few/zero-shot KD methods. The experimental results show that our method significantly outperforms competing baselines.

2 Related Works

Knowledge distillation. Knowledge distillation (KD) has become popular since Hinton et al. introduced its concept in their teacher-student framework [17]. The main goal of KD is to train a compact student network by mimicking the softmax output of a high-capacity teacher network. Many KD methods

have been proposed, and they can be categorized into three groups: *relation-based*, *feature-based*, and *response-based* methods. Relation-based methods not only use the teacher's output but also explore the relationships between different layers of teacher when training the student network. Examples include [40,23,30]. Feature-based methods leverage both the teacher's output at the last layer and the intermediate layers when training student network [1,19,30]. Response-based methods directly mimic the final prediction of the teacher network [17,7,26,29].

Knowledge distillation with limited data. To successfully train the student network, most KD methods assume that both the student's training data and the teacher's training data are identical. For example, [17,6] pointed out that the student only achieved its best accuracy when it had accessed to the teacher's training data. Similarly, [27] mentioned the typical setting in KD methods was the student network trained on the teacher's training data. Recent SOTA methods [19,2,36] also trained both teacher and student networks on the same dataset. In practice, the teacher's training data could be unavailable due to transmission limitation or privacy while we could only collect few samples for the student's training data. Several few/zero-shot KD methods were developed to deal with this situation [3,28,8,41,20]. However, all of these methods require a *white-box* teacher to access to its internal details (e.g. gradient information, weights, feature maps, logits...) to generate synthetic training samples. As far as we know, only BBKD [37] requires few training samples and zero knowledge of the teacher (i.e. *black-box* teacher). However, it is computation and resource intensive as it requires a large pool of candidate images and extensive iterative training.

3 Framework

3.1 Problem definition

Given a small set of *unlabeled* images $\mathcal{X} = \{x_i\}_{i=1}^N$ and a black-box teacher T, our goal is to train a student S on $\mathcal{X}$ s.t. S's performance is comparable to T's.

A direct solution for the above problem is to apply the standard KD method [17]. We first query the teacher to obtain the *hard-label* (i.e. one-hot encoding) y_i for each sample $x_i \in \mathcal{X}$, and then create a labeled training set $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$. Finally, we train the student network with the standard KD loss function:

$$\mathcal{L} = \sum_{(x_i,y_i)\in\mathcal{D}} (1-\omega)\mathcal{L}_{CE}(y_{x_i}^S, y_i) + \omega\mathcal{L}_{KL}(y_{x_i}^S, y_{x_i}^T), \tag{1}$$

where $y_{x_i}^S$, $y_{x_i}^T$, y_i are the student's softmax output, the teacher's softmax output, and the hard-label of a sample x_i, $\mathcal{L}_{CE}$ is the cross-entropy loss, $\mathcal{L}_{KL}$ is the Kullback–Leibler divergence loss, and ω is a trade-off factor to balance the two loss terms. Equation (1) does not use the *temperature* factor as in Hinton's KD method [17] since this requires access to the pre-softmax activations (logits) of teacher, which violates our assumption of "black-box" teacher.

Although training the student network via Equation (1) is a possible way, it is not a good solution as $\mathcal{X}$ only contains very few samples while standard KD methods typically require lots of training samples [17,19,2,36].

3.2 Proposed method FS-BBT

We propose a novel method to solve the above problem, which has three main steps: (1) we generate mixup images from original images contained in $\mathcal{X}$, (2) we replace disqualified mixup images by images generated from CVAE, and (3) we train the student with a combination of original, mixup, and CVAE images.

Generating mixup images. Our idea is to use MixUp [18] – one of recently proposed data augmentation techniques to expand the training set $\mathcal{X}$.

Inspired by BBKD [37], we generate M mixup images from N original images (typically, $N \ll M$). Given two original images $x_i, x_j \in \mathcal{X}$, we use MixUp to generate a synthetic image by a weighted combination between x_i and x_j:

$$x_{mu}(\lambda) = \lambda x_i + (1 - \lambda) x_j, \tag{2}$$

where the coefficient $\lambda \in [0, 1]$ is sampled from a Beta distribution.

Let $X = [x_1, x_2, ..., x_N]$ be the vector of original images. We first sample two M-length vectors $X^1 = [x_1^1, x_2^1, ..., x_M^1]$ and $X^2 = [x_1^2, x_2^2, ..., x_M^2]$, where $x_i^1, x_i^2 \sim X$. We then sample a vector $\lambda = [\lambda_1, \lambda_2, ..., \lambda_M]$ from a Beta distribution, and mixup each pair of two images in X^1 and X^2 using Equation (2):

$$X_{mu} = \begin{bmatrix} \lambda_1 x_1^1 + (1 - \lambda_1) x_1^2 \\ \lambda_2 x_2^1 + (1 - \lambda_2) x_2^2 \\ ... \\ \lambda_M x_M^1 + (1 - \lambda_M) x_M^2 \end{bmatrix} \tag{3}$$

The goal of mixing up original images is to expand the initial set of training images $\mathcal{X}$ as much as possible to cover the manifold of natural images.

However, when mixing up two original images, there is a case that the mixup image is very similar to one of two original images, making it useless. This problem happens when $\lambda_i \approx 0$ or $\lambda_i \approx 1$. Figure 2 shows two examples of desirable vs. disqualified mixup images.

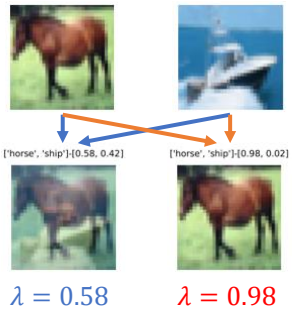

Fig. 2. Desirable vs. disqualified mixup images. At $\lambda = 0.58$, the mixup image shows a good combination between two original images "horse" and "ship" but at $\lambda = 0.98$, it looks almost the same as "horse".

To remove disqualified mixup images, we set a threshold $\alpha \in [0, 0.5]$, and discard mixup images generated with coefficient $\lambda_i \leq \alpha$ or $\lambda_i \geq (1-\alpha)$.

Let M_1 be the number of remaining mixup images after we filter out the disqualified ones. Our next step is to generate $M_2 = M - M_1$ synthetic images from CVAE (we call them CVAE images).

Generating CVAE images. We first query the teacher model to obtain the hard-label y_i for each sample $x_i \in \mathcal{X}$

to create a labeled training set $\mathcal{D} = \{x_i, y_i\}_{i=1}^{N}$. We then train a Conditional Variational Autoencoder (CVAE) model [33] using $\mathcal{D}$ to learn the distribution of the latent variable $z \in \mathbb{R}^d$, where d is the dimension of z. CVAE is a generative model consisting of an encoder and a decoder. We use the encoder network to map an image along with its label $(x, y) \in \mathcal{D}$ to a latent vector z that follows $P(z \mid y)$. From the latent vector z conditioned on the label y, we use the decoder network to reconstruct the input image x. Following [33], we train CVAE by maximizing the variational lower bound objective:

$$\log P(x \mid y) \geq \mathbb{E}(\log P(x \mid z, y)) - \mathrm{KL}(Q(z \mid x, y), P(z \mid y)), \tag{4}$$

where $Q(z \mid x, y)$ is parameterized by the encoder network that maps input image x and its label y to the latent vector z, $P(x \mid z, y)$ is parameterized by the decoder network that reconstructs input image x from the latent vector z and label y, $\mathbb{E}(\log P(x \mid z, y))$ is the expected likelihood, which is implemented by a cross-entropy loss between the input image and the reconstructed image, and $P(z \mid y) \equiv \mathcal{N}(0, I)$ is the prior distribution of z conditioned on y.

After the CVAE model is trained, we can generate images via $G(z, y)$, where $z \sim \mathcal{N}(0, I)$, y is a label, and G is the trained decoder network.

Covering both in-distribution and out-of-distribution samples. To generate M_2 CVAE images, we sample $(\frac{M_2}{2})$-length vector $z^{\mathcal{N}}$ from the normal distribution $\mathcal{N}(0, I)$ and $(\frac{M_2}{2})$-length vector $z^{\mathcal{U}}$ from the uniform distribution $\mathcal{U}([-3, 3]^d)$ (we choose the range $[-3, 3]$ following [16,13]). We create vector $z = z^{\mathcal{N}} \oplus z^{\mathcal{U}}$, where $\oplus$ is the concatenation operator. We manually define a M_2-length vector y_{cvae}, which contains the classes of generated images such that the number of generated images for each class is equivalent. Finally, we generate CVAE images $x_{cvae} = G(z, y_{cave})$.

The intuition behind our generation process is that: (1) Generating images from $z^{\mathcal{N}} \sim \mathcal{N}(0, I)$ will provide *synthetic images within the distribution* of $\mathcal{X}$. These images are interpolated versions of original images. (2) Generating images from $z^{\mathcal{U}} \sim \mathcal{U}([-3, 3]^d)$ will provide *synthetic images out-of the distribution* of $\mathcal{X}$. These images are far way from the original ones, but they are expected to better cover *unseen images,* which improves the student's generalization.

Discussion. One can sample $\lambda_i \in [\alpha, 1 - \alpha]$ to generate M_1 qualified mixup images, then generate M_2 CVAE images. This way requires two hyper-parameters M_1 and M_2. While this is definitely possible, for simplicity we choose to aggregate these two hyper-parameters into a single hyper-parameter M that controls the total number of synthetic images. In experiments, we set the same values for M as those in other few-shot KD methods [3,37] while M_1 and M_2 are automatically computed based on M and α.

Training the student network. After the above steps, we obtain two types of synthetic images – mixup and CVAE images. We send them to the teacher model to obtain their softmax outputs (i.e. their class probabilities) as the *soft-labels* for the images. We train the student network with the original

and synthetic images along with their soft-labels using the following loss:

$$\mathcal{L} = \sum_{x_i \in \mathcal{X} \cup \mathcal{X}_{mu} \cup \mathcal{X}_{cvae}} \mathcal{L}_{CE}(y^S_{x_i}, y^T_{x_i}), \tag{5}$$

where $y^S_{x_i}$, $y^T_{x_i}$ are the student's and the teacher's softmax output, $\mathcal{X}$, $\mathcal{X}_{mu}$, $\mathcal{X}_{cvae}$ are the set of original, mixup, and CVAE images, and $\mathcal{L}_{CE}$ is the cross-entropy loss. Although we train the student by matching the teacher's softmax outputs, our loss function is still applicable in case the teacher only returns top-1 labels [39]. Algorithm 1 summarizes our proposed method **FS-BBT**.

Algorithm 1: The proposed **FS-BBT** algorithm.

Input: T: pre-trained *black-box* teacher network
Input: $\mathcal{X} = \{x_i\}_{i=1}^N$: *unlabeled* training set
Input: M: number of synthetic images
Input: α: threshold to select mixup images
Output: S: student network

begin
- query teacher T to obtain hard-label y_i for each $x_i \in \mathcal{X}$;
- train CVAE model using $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$;
- sample $\lambda = [\lambda_1, ..., \lambda_M]$ from a Beta distribution;
- select M_1 instances of λ_i s.t. $\alpha < \lambda_i < 1 - \alpha$;
- generate M_1 mixup images $\mathcal{X}_{mu}$ using Eq. (3);
- compute $M_2 = M - M_1$;
- sample $(\frac{M_2}{2})$-length vector $z^{\mathcal{N}} \sim \mathcal{N}(0, I)$;
- sample $(\frac{M_2}{2})$-length vector $z^{\mathcal{U}} \sim \mathcal{U}([-3, 3]^d)$;
- create vector $z = z^{\mathcal{N}} \oplus z^{\mathcal{U}}$;
- design M_2-length vector y_{cvae} with class balance;
- generate M_2 CVAE images $\mathcal{X}_{cvae} = G(z, y_{cvae})$;
- query teacher T to obtain soft-labels for $\mathcal{X}$, $\mathcal{X}_{mu}$, $\mathcal{X}_{cvae}$;
- train student S with $\mathcal{X}, \mathcal{X}_{mu}, \mathcal{X}_{cvae}$ and their soft-labels using Eq. (5);

4 Experiments and Discussions

We conduct extensive experiments on five benchmark image datasets to evaluate the classification performance of our method, comparing it with SOTA baselines. Our main goal is to show that with the same number of original and synthetic images, our method is much better than existing few/zero-shot KD methods.

4.1 Datasets

We use five image datasets, namely MNIST, Fashion-MNIST, CIFAR-10, CIFAR-100, and Tiny-ImageNet. These datasets were often used to evaluate the classification performance of KD methods [17,3,8,28,37].

4.2 Baselines

We compare our method **FS-BBT** with the following baselines:

- *Student-Alone*: the student network is trained on the student's training data $\mathcal{D}$ from scratch.
- *Standard-KD*: the student network is trained with the standard KD loss in Equation (1). We choose the trade-off factor $\omega = 0.9$, which is a common value used in KD methods [17,36,42,25].
- *FSKD* [3]: this is a few-shot KD method, which generates synthetic training images using adversarial technique. It requires a *white-box* teacher model to generate adversarial samples to train the student network.
- *WaGe* [20]: this few-shot KD method integrates a Wasserstein-based loss with the standard KD loss to improve the student's generalization.
- *BBKD* [37]: this method uses few original images and a *black-box* teacher model to train the student model. Its main idea is to use MixUp and active learning to generate synthetic images. Since this is the closest work to ours, we consider BBKD as our main competitor.

To have a fair comparison, we use the same teacher-student network architecture, the same number of original and synthetic images N and M as in FSKD and BBKD. We also set the same hyper-parameters (e.g. batch size and the number of epochs) for Student-Alone, Standard-KD, and our **FS-BBT**. We use threshold $\alpha = 0.05$ to select qualified mixup images across all experiments. In an ablation study in Section 4.7, we will investigate how different values for α affect our method's performance. We repeat each experiment five times with random seeds, and report the averaged accuracy. For the baselines FSKD, WaGe, and BBKD, we obtain their accuracy from the papers [20,37][1]. We also compare with several well-known zero-shot KD methods in Section 4.6.

4.3 Results on MNIST and Fashion-MNIST

Experiment settings. Following [3,28], we use the LeNet5 architecture [22] for the teacher and LeNet5-Half (a modified version with half number of channels per layer) for the student. We train the teacher network with a batch size of 64 and 20 epochs. As shown in Table 1, our teacher model achieves comparable accuracy with that reported by BBKD in [37] (99.18% vs. 99.29% for MNIST and 90.15% vs. 90.80% for Fashion-MNIST). We train the student network with a batch size of 64 and 50 epochs. We train the CVAE with feed-forward neural networks for both encoder and decoder, using a latent dimension of 2, a batch size of 256, and 100 (200) epochs for MNIST (Fashion-MNIST). Following FSKD [3] and BBKD [37], we set $N = 2000$ and $M = 24000$ for MNIST and $N = 2000$ and $M = 48000$ for Fashion-MNIST.

The MNIST and Fashion-MNIST datasets have 60K training images and 10K testing images from 10 classes ($[0, 1, ..., 9]$).

[1] This is possible because we use benchmark datasets, and the training and test splits are fixed.

Quantitative results. From Table 1, we can see that our method **FS-BBT** outperforms Student-Alone and Standard-KD on both MNIST and Fashion-MNIST. **FS-BBT** achieves 98.42% (MNIST) and 84.73% (Fashion-MNIST), which is much better than Student-Alone achieving 95.97% and 81.37%. With a support from the teacher model, Standard-KD is always better than Student-Alone, for example, 83.87% vs. 81.37% on Fashion-MNIST.

Table 1. Classification results on MNIST and Fashion-MNIST. "Teacher" indicates the accuracy of the teacher network on the test set. "Model" indicates whether the teacher network is a *black-box* model. "N" shows the number of original images used by each method. "Accuracy" is the accuracy of the student network on the test set. The results of FSKD, WaGe, and BBKD* are obtained from [20,37]. "$\star$" means the BBKD* and **FS-BBT*** methods use the same architecture (LeNet5) for both teacher and student networks.

Dataset	**Method**	**Teacher**	**Model**	N	**Accuracy**
MNIST	Student-Alone	-	-	2,000	95.97%
	Standard-KD	99.18%	Black	2,000	95.99%
	FSKD [3]	99.29%	White	2,000	80.43%
	BBKD* [37]	99.29%	Black	2,000	98.74%
	FS-BBT (Ours)	99.18%	Black	2,000	98.42%
	FS-BBT* (Ours)	99.18%	Black	2,000	**98.91%**
Fashion-MNIST	Student-Alone	-	-	2,000	81.37%
	Standard-KD	90.15%	Black	2,000	83.87%
	FSKD [3]	90.80%	White	2,000	68.64%
	WaGe [20]	92.00%	White	1,000	85.18%
	BBKD* [37]	90.80%	Black	2,000	80.90%
	FS-BBT (Ours)	90.15%	Black	2,000	84.73%
	FS-BBT* (Ours)	90.15%	Black	2,000	**86.53%**

Compared with FSKD and WaGe, **FS-BBT** significantly outperforms FSKD on both MNIST and Fashion-MNIST while **FS-BBT** is similar with WaGe on Fashion-MNIST.

Compared with BBKD, **FS-BBT** achieves a comparable accuracy with BBKD on MNIST while **FS-BBT** outperforms BBKD by a large margin on Fashion-MNIST, where our accuracy improvement is around 4%. Since BBKD uses the same architecture LeNet5 for both teacher and student networks, we also report the accuracy of our method with this setting, indicated by **FS-BBT***. With LeNet5 for the student network, we further achieve 2% gain (i.e. an improvement of 6% over BBKD) on Fashion-MNIST.

4.4 Results on CIFAR-10 and CIFAR-100

Experiment settings. Following [3,28], we use AlexNet [21] and AlexNet-Half (50% filters are removed) for teacher and student networks on CIFAR-10. We

train the teacher network with a batch size of 512 and 50 epochs. Our teacher model achieves a comparable accuracy with that reported by BBKD in [37] (84.07% vs. 83.07%). We train the student network with a batch size of 128 and 100 epochs. We use ResNet-32 [15] for the teacher and ResNet-20 for the student on CIFAR-100. We train student and teacher networks with a batch size of 16/32 and 200 epochs. For both CIFAR-10 and CIFAR-100, we train the CVAE model with convolutional neural networks for both encoder and decoder, using a latent dimension of 2, a batch size of 64, and 600 epochs. Like BBKD [37] and WaGe [20], we set $N = 2000$ for CIFAR-10, $N = 5000$ for CIFAR-100, and $M = 40000$ for both datasets.

CIFAR-10 is set of RGB images with 10 classes, 50K training images, and 10K testing images while CIFAR-100 is with 100 classes, and each class contains 500 training images and 100 testing images. Since neither the accuracy reference nor the source code is available for BBKD on CIFAR-100, we implement BBKD by ourselves, and use the same teacher as in our method for a fair comparison.

Table 2. Classification results on CIFAR-10 and CIFAR-100. "N" shows the number of original images used by each method. The results of FSKD, WaGe, and BBKD* are obtained from [20,37]. "$\star$" means the BBKD* and **FS-BBT*** methods use the same architecture (AlexNet) for both teacher and student networks. "†" means the result is based on our own implementation.

Dataset	**Method**	**Teacher**	**Model**	N	**Accuracy**
CIFAR-10	Student-Alone	-	-	2,000	54.59%
	Standard-KD	84.07%	Black	2,000	58.96%
	FSKD [3]	83.07%	White	2,000	40.58%
	WaGe [20]	89.00%	White	5,000	73.08%
	BBKD* [37]	83.07%	Black	2,000	74.60%
	FS-BBT (Ours)	84.07%	Black	2,000	74.10%
	FS-BBT* (Ours)	84.07%	Black	2,000	**76.17%**
CIFAR-100	Student-Alone	-	-	5,000	32.85%
	Standard-KD	69.08%	Black	5,000	36.79%
	WaGe [20]	47.00%	White	5,000	20.32%
	BBKD† [37]	69.08%	Black	5,000	53.41%
	FS-BBT (Ours)	69.08%	Black	5,000	**56.28%**

Quantitative results. From Table 2 we observe the similar results as in MNIST and Fashion-MNIST. Student-Alone does not have a good accuracy. Standard-KD improves 4% of accuracy over Student-Alone with the knowledge transferred from the teacher.

On CIFAR-10, WaGe and BBKD greatly outperform FSKD, and our **FS-BBT** is comparable with WaGe and BBKD. When we use the same architecture AlexNet for both teacher and student as in BBKD, our variant **FS-BBT*** is the

best method, where it outperforms BBKD (the second best method) by around 2%. **FS-BBT*** outperforms WaGe by around 3% even though WaGe uses much more original training samples than ours (5K vs. 2K), and more powerful teacher (89% vs. 84%).

On CIFAR-100, Student-Alone achieves low accuracy at around 32%. Standard-KD is better than Student-Alone around 4% thanks to the knowledge transferred from the teacher. Interestingly, WaGe works very poorly (only 20.32% of accuracy), becoming the worst method. Its unsatisfactory performance can be a consequence of distilling from a low-accuracy teacher. BBKD is significantly better than other methods with an improvement around 20-30%. Using the same number of original and synthetic images, our method **FS-BBT** achieves 3% gains over BBKD thanks to the CVAE images generated in Section 3.2.

The above results suggest that replacing disqualified mixup images by synthetic images generated from CVAE is an effective solution to improve the robustness and generalization of the student network on the unseen testing samples, as we discussed in Section 3.2.

4.5 Results on Tiny-ImageNet

Experiment settings. We use ResNet-32 and ResNet-20 for the teacher and student. We train teacher and student networks with a batch size of 32 and 100 epochs. Our teacher model achieves a similar accuracy with literature [5] (52.02% vs. 48.26%). We train CVAE in the same way as in CIFAR-100. We set $N = 10000$ and $M = 50000$. Tiny-ImageNet has 100K training images, 10K testing images, and 200 classes.

Table 3. Classification results on Tiny-ImageNet. "N" shows the number of original images used by each method. "†" means the result is based on our own implementation.

Dataset	**Method**	**Teacher**	**Model**	N	**Accuracy**
Tiny-ImageNet	Student-Alone (full)	-	-	100,000	48.81%
	Student-Alone	-	-	10,000	23.19%
	Standard-KD	52.02%	Black	10,000	35.81%
	BBKD† [37]	52.02%	Black	10,000	40.01%
	FS-BBT (Ours)	52.02%	Black	10,000	**43.29%**

Quantitative results. Table 3 shows that Student-Alone reaches a very low accuracy due to a large number of classes presented in this dataset. Standard-KD is significantly better than Student-Alone with an improvement more than 12%. Our method **FS-BBT** achieves 3% gains over BBKD (the second-best baseline).

We also train Student-Alone with full 100K original images and their soft-labels provided by the teacher. This can be considered as an upper bound of all few-shot KD methods as it uses the full set of training images. **FS-BBT** drops

only 5% accuracy from Student-Alone with full training data although it requires only 10% of training data. This proves the efficacy of our proposed framework.

4.6 Comparison with zero-shot (or data-free) KD methods

We also compare with several popular zero-shot KD methods, including *Meta-KD* [24], *ZSKD* [28], *DAFL* [8], *DFKD* [38], and *ZSDB3KD* [39].

Table 4 reports the classification accuracy on MNIST, Fashion-MNIST, and CIFAR-10. Our method is much better than other methods on Fashion-MNIST and CIFAR-10 while it is comparable on MNIST.

Table 4. Classification comparison with zero-shot KD methods. The results of baselines are obtained from [39].

Method	**Model**	**MNIST**	**Fashion-MNIST**	**CIFAR-10**
Meta-KD [24]	White	92.47%	-	-
ZSKD [28]	White	98.77%	79.62%	69.56%
DAFL [8]	White	98.20%	-	66.38%
DFKD [38]	White	**99.08%**	-	73.91%
ZSDB3KD [39]	Black	96.54%	72.31%	59.46%
FS-BBT (Ours)	Black	98.91%	**86.53%**	**76.17%**

4.7 Ablation study

As there are several components and a hyper-parameter α in our method, we further conduct some ablation experiments to analyze how each of them affects to our overall classification accuracy. We select CIFAR-10 for this analysis.

Different types of synthetic images. As described in Section 3.2, we generate three types of synthetic images to train the student network. First, we generate *mixup images*. Second, we sample $z^{\mathcal{N}} \sim \mathcal{N}(0, I)$ to generate CVAE images within the distribution of the original images (we call them *CVAE-WD images*). Finally, we sample $z^{\mathcal{U}} \sim \mathcal{U}([-3, 3]^d)$ to generate CVAE images out-of the distribution of the original images (we call them *CVAE-OOD images*).

Figure 3 shows original images and three types of synthetic images for four true classes "car", "deer", "ship", and "dog". Our synthetic images have good quality, where the objects are clearly recognized and visualized. These synthetic images provide a comprehensive coverage of real images in the test set, resulting in the great improvement of the student network trained on them.

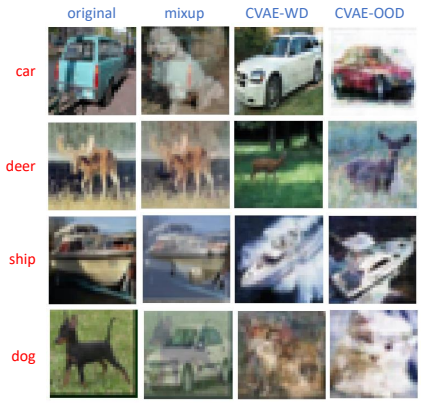

Fig. 3. Original images (1st column) and three types of synthetic images: mixup images (2nd column), CVAE-WD images (3rd column), and CVAE-OOD images (4th column). The text on the left indicates the true labels of original images.

Table 5 reports the accuracy of various types of our synthetic images. The standard KD method achieves only 58.96% of accuracy. By utilizing mixup images, our method achieves up to 71.67% of accuracy. However, using solely mixup images has disadvantages as we discussed in Section 3.2. By combining mixup images with CVAE-WD images or CVAE-OOD images, our method further improves its accuracy up to 72.60% and 73.25% of accuracy respectively. Finally, when combining all three types of synthetic images, our method achieves the best performance at 74.10% of accuracy.

Table 5. Effectiveness of different types of synthetic images on our method **FS-BBT**.

	KD	**FS-BBT** (Ours)						
mixup images		✓			✓	✓		✓
CVAE-WD images			✓		✓		✓	✓
CVAE-OOD images				✓		✓	✓	✓
Accuracy	58.96%	71.67%	70.26%	69.42%	72.60%	73.25%	70.63%	**74.10%**

The ablation experiments suggest that each type of synthetic images in our method is meaningful, where it greatly improves the student's classification performance compared to the standard KD method. By leveraging all three types of synthetic images, our method improves the generalization and diversity of the training set, which is very effective for the training of the student network.

Hyper-parameter analysis. Our method **FS-BBT** has one hyper-parameter, that is, the threshold α to determine disqualified mixup images and replace them

by CVAE images (see Section 3.2). We examine how the different choices of α affect our classification.

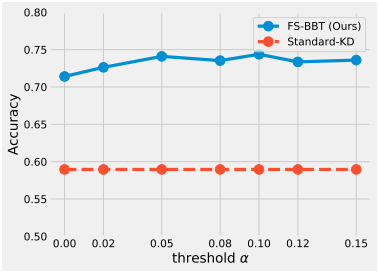

Fig. 4. FS-BBT's accuracy vs. threshold α on CIFAR-10.

As shown in Figure 4, **FS-BBT** is always better than the standard KD method regardless of α values. More importantly, it is stable with $\alpha \in [0.05, 0.10]$, where its accuracy just slightly changes. When α is too small (i.e. $\alpha < 0.05$), most of mixup images will be considered qualified although many of them are very similar to the original images, leading to few extra meaningful training samples added. The performance of **FS-BBT** is decreased as expected. When α is too large (i.e. $\alpha > 0.10$), **FS-BBT** also slightly reduces its accuracy. This is because many mixup images may become cluttered and semantically meaningless due to a large proportion of two original images blended together, making them difficult for the teacher network to label.

5 Conclusion

Existing standard and few/zero-shot KD methods require lots of original training data or a white-box teacher, which are not realistic in some cases. We present **FS-BBT** – a novel KD method, which is effective even with few training samples and a black-box teacher. **FS-BBT** uses MixUp and CVAE to generate synthetic images to train the student network. Although neither of them is new, combining them is a novel solution to address the problem of black-box KD with few samples. As **FS-BBT** is *unsupervised*, which does not require any ground-truth labels, it can be directly applied to domains where labeled images are difficult to obtain e.g. medical images. We demonstrate the benefits of **FS-BBT** on five benchmark image datasets, where it significantly outperforms SOTA baselines. Our work can cheaply create a white-box proxy of a black-box model, which allows algorithmic assurance [9,14] to verify its behavior along various aspects e.g. robustness, fairness, safety, etc.

Acknowledgment: This research was fully supported by the Australian Government through the Australian Research Council's Discovery Projects funding scheme (project DP210102798). The views expressed herein are those of the authors and are not necessarily those of the Australian Government or Australian Research Council.

References

1. Adriana, R., Nicolas, B., Ebrahimi, S., Antoine, C., Carlo, G., Yoshua, B.: Fitnets: Hints for thin deep nets. In: ICLR (2015)

2. Ahn, S., Hu, X., Damianou, A., Lawrence, N., Dai, Z.: Variational information distillation for knowledge transfer. In: CVPR. pp. 9163–9171 (2019)
3. Akisato, K., Zoubin, G., Koh, T., Tomoharu, I., Naonori, U.: Few-shot learning of neural networks from scratch by pseudo example optimization. In: British Machine Vision Conference (BMVC). p. 105 (2018)
4. Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., Raffel, C.: MixMatch: A Holistic Approach to Semi-Supervised Learning. In: NIPS. vol. 32 (2019)
5. Bhat, P., Arani, E., Zonooz, B.: Distill on the go: Online knowledge distillation in self-supervised learning. In: CVPR. pp. 2678–2687 (2021)
6. Chawla, A., Yin, H., Molchanov, P., Alvarez, J.: Data-Free Knowledge Distillation for Object Detection. In: CVPR. pp. 3289–3298 (2021)
7. Chen, G., Choi, W., Yu, X., Han, T., Chandraker, M.: Learning efficient object detection models with knowledge distillation. In: NIPS. pp. 742–751 (2017)
8. Chen, H., Wang, Y., Xu, C., Yang, Z., Liu, C., Shi, B., Xu, C., Xu, C., Tian, Q.: Data-free learning of student networks. In: ICCV. pp. 3514–3522 (2019)
9. Gopakumar, S., Gupta, S., Rana, S., Nguyen, V., Venkatesh, S.: Algorithmic assurance: An active approach to algorithmic testing using bayesian optimisation. NIPS **31** (2018)
10. Gou, J., Yu, B., Maybank, S., Tao, D.: Knowledge distillation: A survey. International Journal of Computer Vision **129**(6), 1789–1819 (2021)
11. Guo, G., Zhang, N.: A survey on deep learning based face recognition. Computer Vision and Image Understanding **189**, 102805 (2019)
12. Guo, H., Mao, Y., Zhang, R.: Mixup as locally linear out-of-manifold regularization. In: AAAI. vol. 33, pp. 3714–3722 (2019)
13. Gyawali, K.: Semi-Supervised Learning by Disentangling and Self-Ensembling Over Stochastic Latent Space. arXiv preprint arXiv:1907.09607 (2019)
14. Ha, H., Gupta, S., Rana, S., Venkatesh, S.: High Dimensional Level Set Estimation with Bayesian Neural Network. In: AAAI. vol. 35, pp. 12095–12103 (2021)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
16. Higgins, I., et al.: beta-vae: Learning basic visual concepts with a constrained variational framework. In: ICLR (2017)
17. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
18. Hongyi, Z., Moustapha, C., Yann, D., David, L.P.: mixup: Beyond empirical risk minimization. In: ICLR (2018)
19. Kim, J., Park, S., Kwak, N.: Paraphrasing complex network: Network compression via factor transfer. In: NIPS. pp. 2760–2769 (2018)
20. Kong, S., Guo, T., You, S., Xu, C.: Learning student networks with few data. In: AAAI. vol. 34, pp. 4469–4476 (2020)
21. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. NIPS **25**, 1097–1105 (2012)
22. LeCun, Y., et al.: LeNet-5: convolutional neural networks. URL: http://yann. lecun. com/exdb/lenet **20**(5), 14 (2015)
23. Lee, S., Song, B.C.: Graph-based knowledge distillation by multi-head attention network. arXiv preprint arXiv:1907.02226 (2019)
24. Lopes, R.G., Fenu, S., Starner, T.: Data-free knowledge distillation for deep neural networks. arXiv preprint arXiv:1710.07535 (2017)
25. Ma, H., Chen, T., Hu, T.K., You, C., Xie, X., Wang, Z.: Undistillable: Making a nasty teacher that cannot teach students. In: ICLR (2021)

26. Meng, Z., Li, J., Zhao, Y., Gong, Y.: Conditional teacher-student learning. In: ICASSP. pp. 6445–6449. IEEE (2019)
27. Nayak, G.K., Mopuri, K.R., Chakraborty, A.: Effectiveness of Arbitrary Transfer Sets for Data-free Knowledge Distillation. In: CVPR. pp. 1430–1438 (2021)
28. Nayak, K., Mopuri, R., Shaj, V., Radhakrishnan, B., Chakraborty, A.: Zero-shot knowledge distillation in deep networks. In: ICML. pp. 4743–4751 (2019)
29. Nguyen, D., Gupta, S., Nguyen, T., Rana, S., Nguyen, P., Tran, T., Le, K., Ryan, S., Venkatesh, S.: Knowledge distillation with distribution mismatch. In: ECML-PKDD. pp. 250–265 (2021)
30. Passalis, N., Tzelepi, M., Tefas, A.: Heterogeneous knowledge distillation using information flow modeling. In: CVPR. pp. 2339–2348 (2020)
31. Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, P., Shyu, M.L., Chen, S.C., Iyengar, S.: A survey on deep learning: Algorithms, techniques, and applications. ACM Computing Surveys **51**(5), 1–36 (2018)
32. Santiago, F., Singh, P., Sri, L., et al.: Building Cognitive Applications with IBM Watson Services: Volume 6 Speech to Text and Text to Speech. IBM Redbooks (2017)
33. Sohn, K., Lee, H., Yan, X.: Learning structured output representation using deep conditional generative models. NIPS pp. 3483–3491 (2015)
34. Sreenu, G., Durai, S.: Intelligent video surveillance: a review through deep learning techniques for crowd analysis. Journal of Big Data **6**(1), 1–27 (2019)
35. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Deepface: Closing the gap to human-level performance in face verification. In: CVPR. pp. 1701–1708 (2014)
36. Tian, Y., Krishnan, D., Isola, P.: Contrastive representation distillation. In: ICLR (2020)
37. Wang, D., Li, Y., Wang, L., Gong, B.: Neural Networks Are More Productive Teachers Than Human Raters: Active Mixup for Data-Efficient Knowledge Distillation from a Blackbox Model. In: CVPR. pp. 1498–1507 (2020)
38. Wang, Z.: Data-free knowledge distillation with soft targeted transfer set synthesis. In: AAAI. vol. 35, pp. 10245–10253 (2021)
39. Wang, Z.: Zero-Shot Knowledge Distillation from a Decision-Based Black-Box Model. In: ICML (2021)
40. Yim, J., Joo, D., Bae, J., Kim, J.: A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In: CVPR. pp. 4133–4141 (2017)
41. Yin, H., Molchanov, P., Alvarez, J., Li, Z., Mallya, A., Hoiem, D., Jha, N., Kautz, J.: Dreaming to distill: Data-free knowledge transfer via deepinversion. In: CVPR. pp. 8715–8724 (2020)
42. Yuan, L., Tay, F., Li, G., Wang, T., Feng, J.: Revisiting knowledge distillation via label smoothing regularization. In: CVPR. pp. 3903–3911 (2020)
43. Zhang, S., Yao, L., Sun, A., Tay, Y.: Deep learning based recommender system: A survey and new perspectives. ACM Computing Surveys **52**(1), 1–38 (2019)