Preventing Catastrophic Forgetting and Distribution Mismatch in Knowledge Distillation via Synthetic Data

Kuluhan Binici*†, Nam Trung Pham*, Tulika Mitra†, Karianto Leman*
*Institute for Infocomm Research, A*STAR, Singapore
†School of Computing, National University of Singapore, Singapore
{kuluhan, tulika}@comp.nus.edu.sg, {ntpham, karianto}@i2r.a-star.edu.sg

Abstract

With the increasing popularity of deep learning on edge devices, compressing large neural networks to meet the hardware requirements of resource-constrained devices became a significant research direction. Numerous compression methodologies are currently being used to reduce the memory sizes and energy consumption of neural networks. Knowledge distillation (KD) is among such methodologies and it functions by using data samples to transfer the knowledge captured by a large model (teacher) to a smaller one (student). However, due to various reasons, the original training data might not be accessible at the compression stage. Therefore, data-free model compression is an ongoing research problem that has been addressed by various works. In this paper, we point out that catastrophic forgetting is a problem that can potentially be observed in existing data-free distillation methods. Moreover, the sample generation strategies in some of these methods could result in a mismatch between the synthetic and real data distributions. To prevent such problems, we propose a data-free KD framework that maintains a dynamic collection of generated samples over time. Additionally, we add the constraint of matching the real data distribution in sample generation strategies that target maximum information gain. Our experiments demonstrate that we can improve the accuracy of the student models obtained via KD when compared with state-of-the-art approaches on the SVHN, Fashion MNIST and CIFAR100 datasets.

1. Introduction

Large-scale deep learning models have achieved overwhelming success over many different learning tasks [25, 17, 24, 3]. However, the huge computational complexities and massive storage requirements make it challenging to deploy them on edge devices with low power and storage capacities such as mobile phones. Knowledge Distillation [12] is a popular method used in compressing complex models and has been proven to be successful in maintaining model performance after compression [2, 26, 27]. The process can be defined as extracting knowledge from a complex pre-trained neural network, called the teacher, to train a relatively more compact student network. This extraction is generally done by forcing the student to imitate the responses of the teacher against the training data. Unfortunately, this data dependency can cause problems when the compression process is carried out by groups other than the model developers. In such scenarios, the group that attempts compression might not have access to the training data due to privacy-related issues or they might not have the means to relocate and store the dataset if it is large in volume. Hence, the whole process could become infeasible.

Recognition of this problematic coupling of KD with data has recently attracted attention from the scientific community [19, 5, 18]. A common way to address this problem is to use synthetic samples for distillation. This task is often referred to as *data-free knowledge distillation* in the literature. Most of the existing works utilize a neural network (generator) to generate synthetic samples. The network parameters are updated throughout distillation via feedback from the teacher network and/or teacher-student performance gap (see Section 2.2). Although the reported results are promising, they still need improvement for data-free methods to become the standard way of compressing image models. There are two aspects of the current data synthesis strategies that we identified as performance bottlenecks. The first one is that, using only newly generated samples to train the student after each time generator's weights are updated [4, 6, 10, 18, 19], could cause the student network to forget the knowledge it acquired in the earlier steps. The reason is that, unlike in regular KD, the synthetic data distribution changes over time as the generator is updated in data-free methods. Therefore, performing KD only with a freshly generated batch, without having stored samples from earlier iterations, could cast the student vulnerable against such distribution shifts. Secondly, targeting the generation of samples over which the student and the

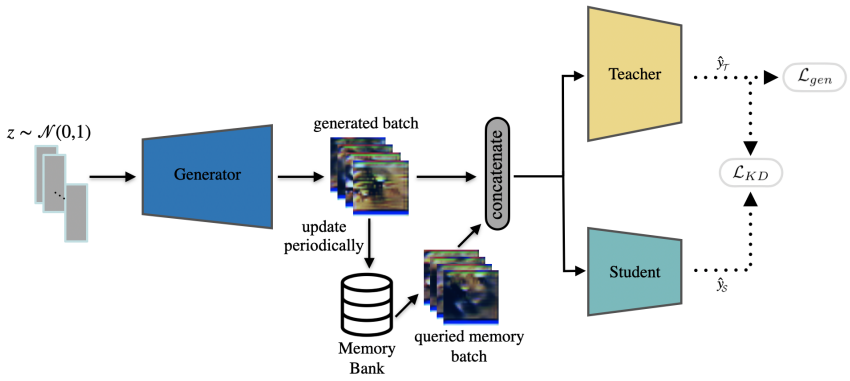


Figure 1: Proposed Data-Free Knowledge Distillation framework. At each iteration, the student is frozen and the generator is trained to produce samples that are confidently classified by the teacher but not by the student. Later the generator is frozen and the student is trained by a combination of samples inferred from the generator and retrieved from the memory bank. The bank is updated with the newly generated samples at a pre-determined frequency.

teacher have maximum disagreement [6, 19], could yield a student that is optimal for a different distribution than the original data.

To address the above-mentioned performance bottlenecks, we propose a memory mechanism to store generated samples over iterations and a generator objective that targets samples that both maximize teacher-student disagreement and approximate real data distribution. Our contributions in this work have three folds and can be summarized as:

- Identification of the catastrophic forgetting problem as a performance bottleneck for available KD methods using synthetic data.
- A data-free KD framework that mitigates catastrophic forgetting by keeping memory of generated samples over iterations.
- Preventing possible mismatches between the generated and the original data distributions with an enhanced sample generation strategy that improves upon the state-of-the-art.

2. Related Works

2.1. Catastrophic Forgetting

The term *Catastrophic Forgetting* was defined by French [7] to describe the loss of previously learned information observed in neural networks when they are sequentially trained to learn new information. Unlike natural cognitive systems, neural networks cannot retain the knowledge they had previously acquired while they keep learning from new data. To preserve such knowledge, they should also be exposed to old data at every update. Goodfellow et al. [9] investigated the extent to which catastrophic forgetting affects neural networks. To alleviate these effects, Kirkpatrick et al. [14] proposed to selectively slow down the learning rates of the weights associated with the previously learned information.

2.2. Data-Free Knowledge Distillation

There are two main synthetic data generation approaches in existing data-free KD works.

The first one is called *model inversion* and can be described as inverting the information flow in a neural network to reconstruct appropriate input samples based on the imposed constraints. Nayak et al. [21] proposed to account for the correlations between target classes while generating samples. They used these correlations to construct soft target labels that describe how synthetic samples should be. Yin et al. [29] used one-hot target labels and applied the Jensen-Shannon divergence between the model-to-be-inverted (teacher) and the student to diversify the synthetic samples. Haroush et al. [11] leveraged the batch normalization statistics to make the synthetic data better mimic the real training data. One weakness of the model inversion-based methods is that they require a vast amount of time to create sufficiently large datasets for distillation [29]. Another one is that some methods include loss terms that constrain the synthetic images to be realistic in the learning objective. The synthesis of realistic images could violate the goal of preserving privacy related to the training data.

The second approach is to use a generator that, once trained, can produce synthetic data suitable for KD. Yoo et al. [30] published an early work in this category and proposed a Variational Auto Encoder (VAE) [13] to generate images that can be recognized by the teacher model. Later, these images are used to distill the knowledge from the teacher to the student. However, by separating the image generation process from KD, this work does not directly attempt to optimize distillation performance. Chen et al. [4]

used a data-free GAN [8] setup and included the distance between the teacher and student responses as a term to be minimized in the objective function. Micaelli and Storkey. [19], and Fang et al. [6] targeted synthetic samples that maximize the information gain for the student. Throughout the paper, we will refer to these samples as "novel" samples. The disagreement between the teacher and the student is used as a proxy to quantify the information gain.

3. Method Description

Our framework contains a generator that produces synthetic samples to distill knowledge from the teacher to the student. We train the generator and the student in two separate stages, alternatively. During the training stage of the generator, the target is to produce samples that are both novel to the student and can be classified by the teacher with high confidence. Later, when the turn comes to the student, it is trained to imitate the responses of the teacher against the generated data. We also include a fixed-sized memory mechanism to remind the student of the information presented to it over time. In Section 3.1, we discuss how the generator loss used in [6] could cause catastrophic forgetting. Later in Section 3.2 we elaborate on our memory mechanism to mitigate catastrophic forgetting. Moreover, we use a loss function that conditions the generator to produce novel samples that also approximate the real data distribution. In Section 3.3, we construct our proposed loss function by combining the ideas introduced in the papers [6] and [4]. The overview of our framework is given in Figure 1 and procedural details are specified in Algorithm 2.

3.1. Catastrophic Forgetting in Data-Free KD Methods Targeting Novel Synthetic Samples

In [6] the optimal generator produces samples that cause the maximum teacher-student disagreement. Formally, the optimal samples at epoch t ($\hat{x}^*_{(t)}$), could be obtained by the Equation 1.

$$\hat{x}^*_{(t)} = \underset{\hat{x}}{argmax}\, D(\mathcal{T}(\hat{x}), \mathcal{S}_{(t)}(\hat{x})) \tag{1}$$

where D is a function that measures the distance between the outputs of the teacher ($\mathcal{T}$) and the student at epoch t ($\mathcal{S}_{(t)}$). Assuming that $\mathcal{S}_{(t)}$ is optimized for $\hat{x}^*_{(t)}$ after epoch t based on,

$$\mathcal{S}^*_{(t+1)} = \underset{\mathcal{S}}{argmin}\, D(\mathcal{T}(\hat{x}^*_{(t)}), \mathcal{S}_{(t)}(\hat{x}^*_{(t)})) \tag{2}$$

the $\hat{x}^*_{(t)}$ will yield minimum teacher-student discrepancy in the next epoch $t+1$. Therefore,

$$\hat{x}^*_{(t)} \cap \hat{x}^*_{(t+1)} = \emptyset \tag{3}$$

$$\hat{x}^*_{(t+1)} = \underset{\hat{x}}{argmax}\, D(\mathcal{T}(\hat{x}), \mathcal{S}^*_{(t+1)}(\hat{x})) \tag{4}$$

Here we emphasize that, once the student is optimized to imitate the teacher for a set of novel samples, the teacher-student disagreement over those samples will diminish. Therefore, in the next turn, the generator will attempt to produce different samples for which the student still can not imitate the teacher's responses. This can result in catastrophic forgetting as the distribution of our synthetic samples change over iterations causing the student to forget the information gained in earlier time steps. An example case is given in Figure 2. In the example, the teacher is trained to classify 2-dimensional data points into 3 classes, denoted by red, green, and blue. Later, a student model is trained by the synthetic samples, marked with yellow crosses, at each epoch. The synthetic samples change over epochs and there is no common sample between synthetic sets across epochs. It can be observed that when we train the student with the synthetic samples generated at the 3rd epoch, the knowledge learned earlier about the green class is forgotten. This example visualizes catastrophic forgetting caused by the above-mentioned data synthesis strategy, in an observable space.

3.2. Mitigating Catastrophic Forgetting

To prevent such problem, we propose to store some of the samples generated throughout iterations, in a memory bank. Since the number of accumulated memory samples could increase linearly with the arbitrarily selected number of training epochs, we use a fix-sized list to maintain our bank. The bank is updated periodically at a desired rate by inserting generated samples. If the sample list has reached maximum size, we replace a randomly selected batch with the batch of new samples. The procedure is described in Algorithm 1.

Algorithm 1 Memory Bank Update

INPUT: A list structure SL that keeps track of recorded samples, batch size B.
if SL is full **then**
  $out\_samples \leftarrow SL.remove\_random(B)$
  $REMOVE(out\_samples)$
end if
sample B vectors (z) from $\mathcal{N}(0, 1)$
$in\_samples \leftarrow \mathcal{G}(z)$
$SL.append(in\_samples)$
$SAVE(in\_samples)$

During distillation, if the stored sample list is not empty, a batch of memory samples is selected randomly and combined with freshly generated samples to train the student.

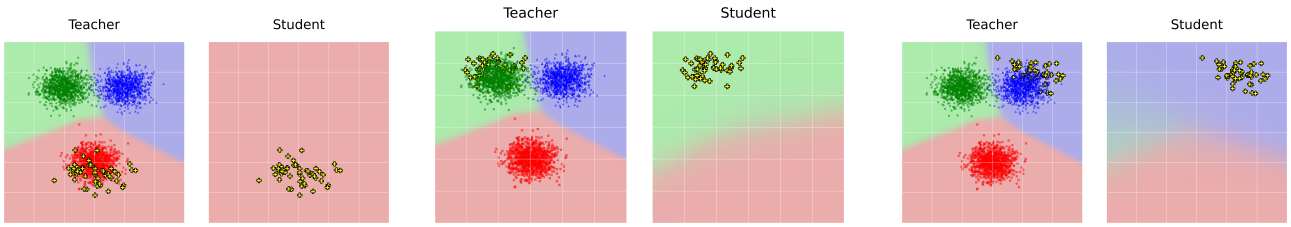


Figure 2: Example of catastrophic forgetting during data-free distillation. The teacher and student decision boundaries are plotted for 3 epochs. Real data samples are displayed in the teacher's decision space and the synthetic samples are marked with yellow crosses.

3.3. Generator Loss to Prevent Mismatch Between Synthetic and Original Data Distributions

As mentioned in Section 2.2, targeting the generation of novel samples has demonstrated promising results on several benchmarks [6, 19]. However, this approach fails to maintain high-quality distillation across different benchmarks. We observed this issue after conducting experiments with various new datasets and student architectures. Throughout our experiments, the training/testing performances of the student models were unstable and highly impacted by the hyper-parameter choices. We attribute this to the lack of constraint to generate samples that come from a similar distribution as the training data. Without such constraint, the distillation can result in a student network that is optimized to imitate the teacher for a certain sample subspace which is different from the original training data. Thus, the student might fail to behave like the teacher when presented with samples from the original data distribution. Even if during the distillation, the novel samples coincidentally correspond with real samples, once the student is optimized for them, the next batches of novel samples will be different. If the distillation continues after such correspondence, the student will forget the decision boundaries related to the real data distribution, causing its accuracy on the real test set to degrade.

To prevent this issue, we propose to generate samples that both induce high discrepancy between teacher and student predictions, and stimulate similar responses from the teacher as the real data. Our optimization objective while training the generator, has two main components and can be denoted as,

$$\mathcal{L}_{gen} = \mathcal{L}_{match} + \mathcal{L}_{d} \tag{5}$$

We give the details of these components in the following sections.

Matching Training Data Distribution To generate samples that match the distribution of the original training data, we adopt the optimization objective described in [4] as

$$\mathcal{L}_{match} = \mathcal{L}_{oh} + \alpha\mathcal{L}_{a} + \beta\mathcal{L}_{ie}, \tag{6}$$

where α and β are coefficients to adjust the weighted contributions of the participating terms. $\mathcal{L}_{oh}$ is the one-hot loss that causes the teacher outputs to be one-hot vectors when minimized. It can be defined as the the cross-entropy between the teacher's softmax outputs ($\hat{y}^i_{\mathcal{T}} = \mathcal{T}(\hat{x}^i)$)) and the predicted one-hot labels ($t^i_{\mathcal{T}} = argmax(\hat{y}^i_{\mathcal{T}})$) when image $\hat{x}^i$ is the input.

$$\mathcal{L}_{oh} = \frac{1}{n}\sum_i t^i_{\mathcal{T}} log(\hat{y}^i_{\mathcal{T}}) \tag{7}$$

$\mathcal{L}_a$ stands for the activation loss and is motivated by the observation of meaningful inputs causing higher valued activation maps in a trained network. The term is defined as

$$\mathcal{L}_{a} = -\frac{1}{n}\sum_i \|f^i_{\mathcal{T}}\|_1 \tag{8}$$

where $f^i_{\mathcal{T}}$ denotes the activation values observed at all of the selected layers i of the teacher ($\mathcal{T}$).

Lastly, $\mathcal{L}_{ie}$ is the term that imposes the generator to produce an equal amount of images from each category. When minimized, the entropy of the number of images generated per category gets maximized causing each category to have a similar amount of samples. If we define the probability distribution of class predictions as $p(\hat{y}_{\mathcal{T}}) = \frac{1}{n}\sum_i \hat{y}^i_{\mathcal{T}}$, where $y^i_{\mathcal{T}}$ is the softmax output vector for the sample i, then the loss term can be denoted as

$$\mathcal{L}_{ie} = -\sum_j p(\hat{y}_{\mathcal{T}})^j log(p(\hat{y}_{\mathcal{T}})^j) \tag{9}$$

Here $p(\hat{y}_{\mathcal{T}})^j$ is the averaged occurrence frequency of the class indexed by j among the generated samples.

Maximum Teacher-Student Discrepancy Commonly, L1 distance or Kullback-Leibler (KL) divergence between

the teacher and student responses are used to quantify disagreement in the loss function. Instead we choose Jensen-Shannon (JS) divergence (Equation 12), inspired by [29]. The loss term can be denoted by

$$M = \frac{1}{2}(\mathcal{T}(\hat{x}) + \mathcal{S}(\hat{x})) \tag{10}$$

$$JS(\mathcal{T}(\hat{x}), \mathcal{S}(\hat{x})) = \tfrac{1}{2}(KL(\mathcal{T}(\hat{x}) \parallel M) + KL(\mathcal{S}(\hat{x}) \parallel M)) \tag{11}$$

$$\mathcal{L}_d = 1 - JS(\mathcal{T}(\hat{x}), \mathcal{S}(\hat{x})) \tag{12}$$

where KL stands for the KL divergence. Our motivation to choose JS divergence to represent the discrepancy between the predictions of the teacher and student, is discussed in section 4.2.

3.4. Knowledge Distillation

In the knowledge distillation phase, the main objective is to get the student responses as close to those of the teacher as possible. To do so, we employ the L1 distance between the responses of the teacher and student networks (Equation 13). An analysis on the impacts of other possible distance terms was given in [6].

$$\mathcal{L}_{KD} = \|t_{\mathcal{T}} - t_{\mathcal{S}}\|_1 \tag{13}$$

4. Experiments

In this section, we share the experiments we have conducted to demonstrate the effectiveness of our proposed method on four image classification datasets. Our results are given in Table 1 together with those of the baselines, for comparison. Moreover, we discuss how the inclusion of each technique we propose and hyper-parameter choices related to them, affect distillation performance. We consider DAFL [4], DFAD [6], EATSKD [20], DeGAN [1] and RDSKD [10] as our baselines for comparison using the same student and teacher models. It is noted, to test the baselines on the benchmarks that they had not reported results on, we used the code on their GitHub repositories.

4.1. Datasets

In our experiments, we have used MNIST [16], SVHN [22], CIFAR-10 [15], CIFAR-100 [15], and Fashion MNIST [28] datasets as benchmarks. In all experiments we use the DCGAN [23] inspired generator adopted from [4].

MNIST is an image classification dataset composed of 70,000 grey-scale images of handwritten digits from 0 to 9. Training set consists of 60,000 images while the testing set consists of 10,000. This benchmark is simple and therefore student-teacher gaps in most methods are very small. For the experiments on MNIST, we selected LeNet-5 [16] as the teacher and LeNet-5-Half as the student. The teacher model was achieving 98.9% accuracy on the test set. Our

Algorithm 2 Data-Free Knowledge Distillation

INPUT: A trained teacher network $\mathcal{T}(x;\theta_{\mathcal{T}})$, batch size B

Initialize the more compact student $\mathcal{S}(x;\theta_{\mathcal{S}})$ network and the generator $\mathcal{G}(z;\theta_{\mathcal{G}})$
for number of epochs **do**
  for k steps **do**
    sample B vectors (z) from $\mathcal{N}(0,1)$
    $\hat{x} \leftarrow \mathcal{G}(z)$
    if memory bank is not empty **then**
      sample batch $\hat{x}_M$ from memory
      $\hat{x} \leftarrow CONCAT(\hat{x}, \hat{x}_M)$
    end if
    $\hat{y}_{\mathcal{T}} \leftarrow \mathcal{T}(\hat{x})$
    $\hat{y}_{\mathcal{S}} \leftarrow \mathcal{S}(\hat{x})$
    Calculate the loss term $\mathcal{L}_{KD}$ given $\hat{y}_{\mathcal{T}}$ and $\hat{y}_{\mathcal{S}}$
    $\theta_{\mathcal{S}} \leftarrow optimizer.step(backward(\mathcal{L}_{KD}, \theta_{\mathcal{S}}), \theta_{\mathcal{S}})$
  end for
  sample B vectors (z) from $\mathcal{N}(0,1)$
  $\hat{x} \leftarrow \mathcal{G}(z)$
  $\hat{y}_{\mathcal{T}}, f_{\mathcal{T}} \leftarrow \mathcal{T}(\hat{x})$
  $\hat{y}_{\mathcal{S}} \leftarrow \mathcal{S}(\hat{x})$
  Calculate the loss terms $\mathcal{L}_{oh}$, $\mathcal{L}_a$, $\mathcal{L}_{ie}$ and $\mathcal{L}_{JS}$ given $\hat{x}$, $\hat{y}_{\mathcal{T}}$, $\hat{y}_{\mathcal{S}}$, $f_{\mathcal{T}}$
  $\mathcal{L}_{gen} \leftarrow \mathcal{L}_{oh} + \alpha\mathcal{L}_a + \beta\mathcal{L}_{ie} + \mathcal{L}_{JS}$
  $\theta_{\mathcal{G}} \leftarrow optimizer.step(backward(\mathcal{L}_{gen}, \theta_{\mathcal{G}}), \theta_{\mathcal{G}})$
  if epoch % update_frequency == 0 **then**
    update memory bank
  end if
end for

method performed comparably with the DAFL and DFAD baselines while outperforming RDSKD. Our experiments revealed that when we set the memory size as 10 batches and update frequency as 1 per 5 epochs, the accuracy increases more consistently (see Figures 4a and 4b). However, we achieved the best results with the update rate of once in every 1 epoch and the memory bank size of 10 batches. Additionally, we plot the learning curves of the student obtained by our method and the baselines, during distillation. The Figures 3a and 3b demonstrate that our method can reduce the variance among student accuracies achieved throughout epochs. This is desirable in real-life applications since, if there is no evaluation data during the distillation process, it is best if the student model performs at its peak when the process ends. Therefore, maintaining high student accuracy over iterations is as important as achieving it. Based on the plots, our method accomplished these better than the baselines DAFL and DFAD.

SVHN is a colored digit classification dataset consisting of over 600,000 labeled images. This dataset is similar to MNIST with the difference of having RGB channels there-

	MNIST $\mathcal{T}$:LeNet5 $\mathcal{S}$:LeNet-half		SVHN $\mathcal{T}$:ResNet34 $\mathcal{S}$:ResNet18		CIFAR10 $\mathcal{T}$:ResNet34 $\mathcal{S}$:ResNet18		Fashion MNIST $\mathcal{T}$:ResNet34 $\mathcal{S}$:ResNet18	
Method	**FLOPs**	**Accuracy**	**FLOPs**	**Accuracy**	**FLOPs**	**Accuracy**	**FLOPs**	**Accuracy**
Teacher	433K	98.9%	1.16G	96.3%	1.16G	95.4%	1.16G	94.1%
Train with data	139K	98.6%	558M	96.0%	558M	93.9%	558M	94.0%
RDSKD	139K	97.6%	558M	94.6%	558M	90.8%	558M	-
DAFL	139K	98.2%	558M	94.5%*	558M	92.2%	558M	90.4%*
DFAD	139K	**98.3%**	558M	94.7%*	558M	**93.3%**	558M	70.0%*
Ours	139K	98.2%	558M	**95.4%**	558M	91.3 %	558M	92.3%
Ours (w/ memory bank)	139K	**98.3%**	558M	95.0%	558M	92.4 %	558M	**92.9%**

Table 1: Accuracy results of student networks obtained by different data-free distillation strategies on several datasets. The (*) mark indicates that the results are produced by running the original implementations on previously untested datasets.

fore the benchmark is again relatively simple. For the experiments on SVHN, we selected ResNet-34 [16] as the teacher and ResNet-18 as the student. The teacher model was achieving 96.3% accuracy on the test set. Our method outperformed all baselines. While testing the inclusion of the memory bank, we used an update period of 5 epochs and a memory size of 10 batches. For the remaining benchmarks, we used the same memory bank settings.

CIFAR10 is a colored image classification dataset consisting of over 60,000 labeled images from 10 categories, each category containing 6000 samples. The number of samples ratio for training and testing sets is 5/1. This benchmark contains less salient samples for classification and therefore is more challenging than MNIST and SVHN. For the experiments on CIFAR10, we selected ResNet-34 [16] as the teacher and ResNet-18 as the student. The ResNet-34 teacher model was achieving 95.4% accuracy on the test set. Our method containing the memory bank, outperformed DAFL and RDSKD baselines while performing lower than DFAD.

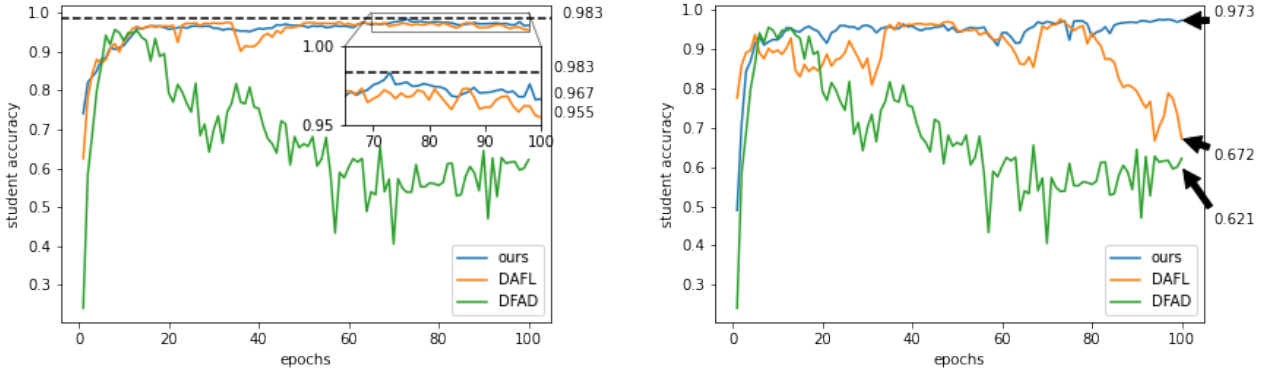
Fashion MNIST is a dataset of 70,000 grey-scale cloth images from 10 different categories. The number of samples in the training and testing sets are 60,000 to 10,000 respectively. To compare our work with DAFL and DFAD, we selected ResNet-34 [16] as the teacher and ResNet-18 as the student. The ResNet-34 teacher model was achieving 94.1% accuracy on the test set. Our method outperformed both baselines. The improvement our method achieves over Fashion MNIST is much clearer than it achieves over other benchmarks. This could be due to a particular organization of the teacher decision space, tightening the performance bottleneck of the baselines caused by their above-mentioned weaknesses. We note that the results of our baselines we report are obtained after running repeated experiments with different random initializations. Even after numerous trials, DFAD could not maintain the distillation quality it achieved in other benchmarks. We relate this to its frailty against non-optimal hyper-parameter choices, which is also recognized by [10].

CIFAR100 is a more extensive version of CIFAR10 with 100 object categories. There are 600 colored images for each type of object. These total 60,000 samples are split into training and testing sets with a 5 to 1 ratio. For the experiments on CIFAR100, we selected ResNet-34 [16] as the teacher and ResNet-18 as the student. The ResNet-34 teacher model was achieving 77.94% accuracy on the test set. Our method containing the memory bank outperformed all baselines.

	CIFAR100 $\mathcal{T}$:ResNet34 $\mathcal{S}$:ResNet18	
Method	**FLOPs**	**Accuracy**
Teacher	1.16G	77.94%
Train with data	558M	76.53%
DeGAN	558M	65.25%
EATSKD	558M	67.18%
DFAD	558M	67.70%
DAFL	558M	74.47%
Ours	558M	**75.35%**

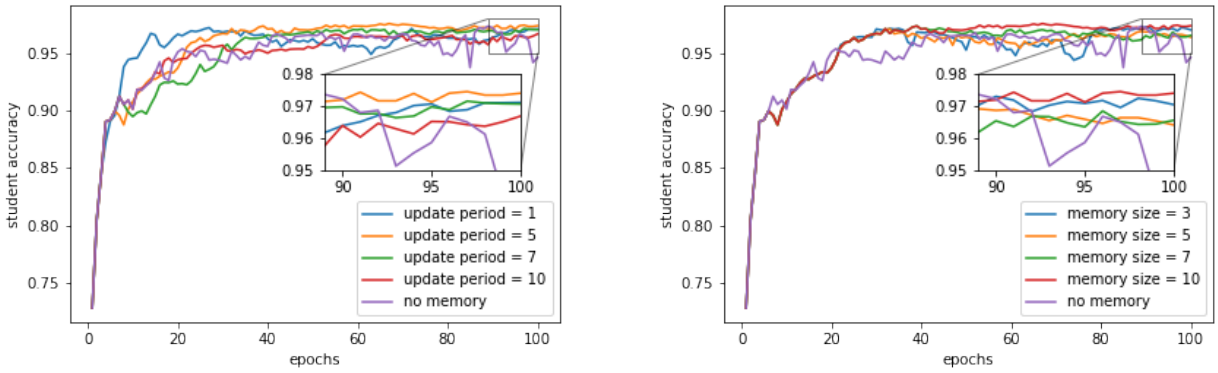
Table 2: Accuracy results of student networks obtained by different data-free distillation strategies on CIFAR100 dataset.

In summary, Tables 1 and 2 show that our method improved the highest accuracy reached by distilled student networks for SVHN, Fashion MNIST and CIFAR100 datasets. Moreover, although it performs lower than DFAD on CIFAR10, our method consistently maintained high performance across different benchmarks. Considering these, we can claim that our efforts in mitigating catastrophic forgetting and avoiding mismatch between synthetic and real data, can improve the overall quality of data-free KD.



(a) (b)

Figure 3: Progression of student accuracy throughout distillation achieved by our method in comparison with DAFL and DFAD.



(a) (b)

Figure 4: Impacts of update rate and memory size on the student training accuracy.

4.2. Ablation Study

To demonstrate how each component in our proposed method impacts the distillation performance, we reported the performances of our method after the inclusion of each component (see Table 1). The addition of the constraint to generate samples that match the original data distribution improved accuracy compared to prior methods for SVHN and Fashion MNIST benchmarks. Later, the inclusion of the memory bank, boosted our performance in all benchmarks except SVHN, yielding comparable results with DFAD on MNIST and better performance than DAFL on CIFAR10.

Additionally, in Table 3, we compare the results obtained by different choices of divergence terms that force the generator to produce novel samples for the student. For this experiment, we used our method without the memory bank. Choosing JS divergence over other alternatives caused a slight improvement in the performance. We attribute this to the symmetric property of JS divergence as opposed to the asymmetric KL divergence. While the KL divergence quantifies the distance of two distributions from the reference point of one, JS divergence considers both reference points. This is due to the definition of JS divergence given in Equation 11 which contains a combination of both the KL divergence of the student from the teacher and that of the teacher from the student. Such way to represent the teacher-student discrepancy could be more suitable than the asymmetric KL divergence.

	JS	L1	KLD
Student Accuracy	92.3%	92.1%	92.1%

Table 3: Effectiveness of different distance terms used to produce novel samples for the student.

4.3. Memory Bank Update Frequency

The update frequency for the memory bank is a hyper-parameter that needs to be pre-determined and set before distillation. To find the optimal value, we compared the behaviors of different update rates for MNIST dataset distillation. Since the randomness in the parameter initialization could have affected our comparison, we set the random seeds manually. The memory bank size is limited to contain at most 10 batches. We shared the outcome of our experiments in Figure 4a. From the plot, it can be observed that regardless of the choice of update rate, the usage of memory bank improved the peak and final values of student accuracy while also causing fewer fluctuations. Moreover, among the update frequencies we tested, 1 update per 5 epochs performed better than others by resulting in a more smooth learning curve and higher student accuracy. Meanwhile, updating the stored samples every epoch, caused larger fluctuations relative to other choices. We attribute this to the rapid refresh rate unable to store samples for enough iterations. On the other hand, the less frequent update periods such as 7 and 10 could be causing failure to memorize some informative samples generated in between updates.

4.4. Memory Bank Size

Selection of the maximum number of batches to be stored in the memory bank is another hyper-parameter choice. To observe the isolated impacts of such choice, we fix the update period to be 5 epochs and plot the learning curves obtained by different memory sizes. We again used manual random seed for a fair comparison. Figure 4b shows that storing at most 10 batches at any step throughout distillation resulted both in a higher final student accuracy and a smoother increase in accuracy. Additionally, we observed a correlation between memory size and smoothness of the curve, as keeping fewer samples in memory caused larger fluctuations. We note that any choice of memory size yielded a better performing student than not storing samples.

4.5. Distillation Among Different Architecture Types

To the best of our knowledge, all prior work had reported results on teacher-student pairs that contain the same type of architectural blocks. For the LeNet5 teacher, the student is typically picked as LeNet-half and for the ResNet34 teacher, ResNet18 is selected as the student. In this work, we also practice distillation among networks with different architectural blocks such as ResNet and MobileNetV2. We used the same ResNet34 teacher trained on Fashion MNIST as in Table 1 while changing the student to be MobileNetV2. From Table 4, it can be observed that our method achieved higher performance than DAFL even when we change the student network architecture. The results we report of our methods were obtained by including both the memory bank and the proposed generator loss in our KD framework.

	Fashion MNIST			
	$\mathcal{T}$:ResNet34 $\mathcal{S}$:MobileNetV2		$\mathcal{T}$:ResNet34 $\mathcal{S}$:ResNet18	
Method	**FLOPs**	**Acc.**	**FLOPs**	**Acc.**
Teacher	1.16G	94.1%	1.16G	94.1%
Train w/ data	16M	92.4%	558M	94.0%
DAFL	16M	85.7%*	558M	90.4%*
Ours	16M	**91.3%**	558M	**92.9%**

Table 4: Accuracy results of MobileNetV2 student networks obtained by different data-free distillation strategies on Fashion MNIST dataset. The (*) mark indicates that the results are produced by running the original implementations on previously untested settings.

5. Conclusion & Future Work

In this paper, we identify two problems that impair data-free knowledge distillation performance and propose methods to solve them. These are catastrophic forgetting and, mismatch between synthetic and real data distributions.

Addressing the first problem, we proposed a memory system where we keep the history of generated samples over iterations. While this simple approach manages to suppress the negative effects of catastrophic forgetting on student accuracy, it also increases the secondary memory overhead and slows down distillation. Therefore, we believe future efforts could be directed to find more efficient ways to prevent catastrophic forgetting in data-free KD. Some possible improvements could be achieved by selectively storing samples and/or by storing compressed versions of the samples.

To remedy the second problem, we propose a data-free KD strategy that constrains the generated novel samples to match the distribution of the original data. This way, we ensure that the synthetic samples distinguishing the teacher from the student are not irrelevant to the original training data. Additionally, to produce such distinguishing samples we adopt Jensen-Shannon divergence as a loss term and compare it with other alternatives.

Experimental results show that our framework not only improves data-free knowledge distillation performance for certain datasets, but also maintains high student accuracy throughout the entire process.

Acknowledgement

This research is partially supported by the National Research Foundation, Singapore under its Competitive Research Programme Award NRF-CRP23-2019-0003. We thank Cihan Acar for helpful comments on the manuscript.

References

[1] Sravanti Addepalli, Gaurav Kumar Nayak, Anirban Chakraborty, and Venkatesh Babu Radhakrishnan. Degan: Data-enriching gan for retrieving representative samples from a trained classifier. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3130–3137, 2020.

[2] Taichi Asami, Ryo Masumura, Yoshikazu Yamaguchi, Hirokazu Masataki, and Yushi Aono. Domain adaptation of dnn acoustic models using knowledge distillation. In *ICASSP*, pages 5185–5189. IEEE, 2017.

[3] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, pages 6299–6308, 2017.

[4] Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Data-free learning of student networks. In *ICCV*, pages 3514–3522, 2019.

[5] Yoojin Choi, Jihwan Choi, Mostafa El-Khamy, and Jungwon Lee. Data-free network quantization with adversarial knowledge distillation. In *CVPR*, pages 710–711, 2020.

[6] Gongfan Fang, Jie Song, Chengchao Shen, Xinchao Wang, Da Chen, and Mingli Song. Data-free adversarial distillation. *arXiv preprint arXiv:1912.11006*, 2019.

[7] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *NIPS*, 27:2672–2680, 2014.

[9] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[10] Pengchao Han, Jihong Park, Shiqiang Wang, and Yejun Liu. Robustness and diversity seeking data-free knowledge distillation. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2740–2744. IEEE, 2021.

[11] Matan Haroush, Itay Hubara, Elad Hoffer, and Daniel Soudry. The knowledge within: Methods for data-free model compression. In *CVPR*, pages 8494–8502, 2020.

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

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

[14] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[15] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[16] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[17] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015.

[18] Liangchen Luo, Mark Sandler, Zi Lin, Andrey Zhmoginov, and Andrew Howard. Large-scale generative data-free distillation. *arXiv preprint arXiv:2012.05578*, 2020.

[19] Paul Micaelli and Amos J Storkey. Zero-shot knowledge transfer via adversarial belief matching. In *NIPS*, pages 9551–9561, 2019.

[20] Gaurav Kumar Nayak, Konda Reddy Mopuri, and Anirban Chakraborty. Effectiveness of arbitrary transfer sets for data-free knowledge distillation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1430–1438, 2021.

[21] Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, Venkatesh Babu Radhakrishnan, and Anirban Chakraborty. Zero-shot knowledge distillation in deep networks. In *ICML*, pages 4743–4751. PMLR, 2019.

[22] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

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

[24] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[25] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.

[26] Yangyang Shi, Mei-Yuh Hwang, Xin Lei, and Haoyu Sheng. Knowledge distillation for recurrent neural network language modeling with trust regularization. In *ICASSP*, pages 7230–7234. IEEE, 2019.

[27] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*, 2019.

[28] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[29] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *CVPR*, pages 8715–8724, 2020.

[30] Jaemin Yoo, Minyong Cho, Taebum Kim, and U Kang. Knowledge extraction with no observable data. In *NIPS*, pages 2705–2714, 2019.