DATASET DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Model distillation aims to distill the knowledge of a complex model into a simpler one. In this paper, we consider an alternative formulation called *dataset distillation*: we keep the model fixed and instead attempt to distill the knowledge from a large training dataset into a small one. The idea is to *synthesize* a small number of data points that do not need to come from the correct data distribution, but will, when given to the learning algorithm as training data, approximate the model trained on the original data. For example, we show that it is possible to compress $60,000$ MNIST training images into just 10 synthetic *distilled images* (one per class) and achieve close to the original performance, given a fixed network initialization. We evaluate our method in various initialization settings. Experiments on multiple datasets, MNIST, CIFAR10, PASCAL-VOC, and CUB-200, demonstrate the advantage of our approach compared to alternative methods. Finally, we include a real-world application of dataset distillation to the continual learning setting: we show that storing distilled images as episodic memory of previous tasks can alleviate forgetting more effectively than real images.

1 INTRODUCTION

Hinton et al. (2015) proposed network distillation as a way to transfer the knowledge from an ensemble of many separately-trained networks into a single, typically compact network, performing a type of model compression. In this paper, we are considering a related but orthogonal task: rather than distilling the model, we propose to distill the dataset. Unlike network distillation, we keep the model fixed but encapsulate the knowledge of the entire training dataset, which typically contains thousands to millions of images, into a small number of synthetic training images. We show that we can go as low as *one* synthetic image per category, training the same model to reach surprisingly good performance on these synthetic images. For example, in Figure 1a, we compress $60,000$ training images of MNIST digit dataset into only 10 synthetic images (one per category), given a fixed network initialization. Training the standard LENET (LeCun et al., 1998) on these 10 images yields test-time MNIST recognition performance of 94%, compared to 99% for the original dataset. For networks with unknown random weights, 100 synthetic images train to 89%. We name our method *Dataset Distillation* and these images *distilled images*.

But why is dataset distillation interesting? First, there is the purely scientific question of how much data is encoded in a given training set and how compressible it is? Second, we wish to know whether it is possible to "load up" a given network with an entire dataset-worth of knowledge by a handful of images. This is in contrast to traditional training that often requires tens of thousands of data samples. Finally, on the practical side, dataset distillation enables applications that require compressing data with its task. We demonstrate that under the continual learning setting, storing distilled images as memory of past task and data can alleviate catastrophic forgetting (McCloskey and Cohen, 1989).

A key question is whether it is even possible to compress a dataset into a small set of synthetic data samples. For example, is it possible to train an image classification model on synthetic images that are not on the manifold of natural images? Conventional wisdom would suggest that the answer is no, as the synthetic training data may not follow the same distribution of the real test data. Yet, in this work, we show that this is indeed possible.

We present an optimization algorithm for synthesizing a small number of synthetic data samples not only capturing much of the original training data but also tailored explicitly for fast model training with only a few data point. To achieve our goal, we first derive the network weights as a

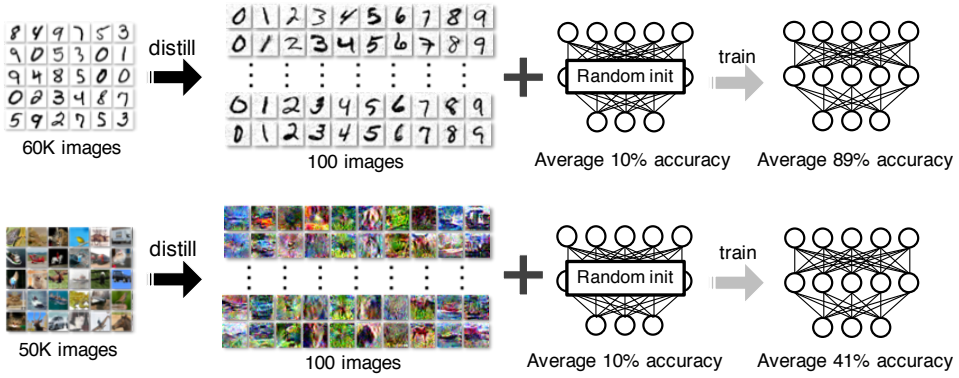


Figure 1: We distill the knowledge of tens of thousands of images into a few synthetic training images called distilled images. On MNIST, 100 distilled images can train a standard LENET with a random initialization to 89% test accuracy, compared to 99% when fully trained. On CIFAR10, 100 distilled images can train a network with a random initialization to 41% test accuracy, compared to 80% when fully trained. In Section 3.6, we show that these distilled images can efficiently store knowledge of previous tasks for continual learning.

differentiable function of our synthetic training data. Given this connection, instead of optimizing the network weights for a particular training objective, we optimize the pixel values of our distilled images. However, this formulation requires access to the initial weights of the network. To relax this assumption, we develop a method for generating distilled images for randomly initialized networks. To further boost performance, we propose an iterative version, where the same distilled images are reused over multiple gradient descent steps so that the knowledge can be fully transferred into the model. Finally, we study a simple linear model, deriving a lower bound on the size of distilled data required to achieve the same performance as training on the full dataset.

We demonstrate that a handful of distilled images can be used to train a model with a fixed initialization to achieve surprisingly high performance. For networks pre-trained on other tasks, our method can find distilled images for fast model fine-tuning. We test our method on several initialization settings: fixed initialization, random initialization, fixed pre-trained weights, and random pre-trained weights. Extensive experiments on four publicly available datasets, MNIST, CIFAR10, PASCAL-VOC, and CUB-200, show that our approach often outperforms existing methods. Finally, we demonstrate that for continual learning methods that store limited-size past data samples as episodic memory (Lopez-Paz and Ranzato, 2017; Kirkpatrick et al., 2017), storing our distilled data instead is much more effective. Our distilled images contain richer information about the past data and tasks, and we show experimental evidence on standard continual learning benchmarks. Our code, data, and models will be available upon publication.

2 RELATED WORK

Knowledge distillation. The main inspiration for this paper is network distillation (Hinton et al., 2015), a widely used technique in ensemble learning (Radosavovic et al., 2018) and model compression (Ba and Caruana, 2014; Romero et al., 2015; Howard et al., 2017). While network distillation aims to distill the knowledge of multiple networks into a single model, our goal is to compress the knowledge of an entire dataset into a few synthetic data. Our method is also related to the theoretical concept of teaching dimension, which specifies the minimal size of data needed to teach a target model to a learner (Shinohara and Miyano, 1991; Goldman and Kearns, 1995). However, methods (Zhu, 2013; 2015) inspired by this concept require the existence of target models, which our method does not.

Dataset pruning, core-set construction, and instance selection. Another way to distill knowledge is to summarize the entire dataset by a small subset, either by only using the "valuable" data for model training (Angelova et al., 2005; Felzenszwalb et al., 2010; Lapedriza et al., 2013) or by only labeling the "valuable" data via active learning (Cohn et al., 1996; Tong and Koller, 2001). Similarly, core-set construction (Tsang et al., 2005; Har-Peled and Kushal, 2007; Bachem et al., 2017; Sener and Savarese, 2018) and instance selection (Olvera-López et al., 2010) methods aim to select a

subset of the entire training data, such that models trained on the subset will perform as well as the model trained on the full dataset. For example, solutions to many classical linear learning algorithms, e.g., Perceptron (Rosenblatt, 1957) and SVMs (Hearst et al., 1998), are weighted sums of subsets of training examples, which can be viewed as core-sets. However, algorithms constructing these subsets require many more training examples per category than we do, in part because their "valuable" images have to be real, whereas our distilled images are exempt from this constraint.

Gradient-based hyperparameter optimization. Our work bears similarity with gradient-based hyperparameter optimization techniques, which compute the gradient of hyperparameter w.r.t. the final validation loss by reversing the entire training procedure (Bengio, 2000; Domke, 2012; Maclaurin et al., 2015; Pedregosa, 2016). We also backpropagate errors through optimization steps. However, we use only training set data and focus more heavily on learning synthetic training data rather than tuning hyperparameters. To our knowledge, this direction has only been slightly touched on previously (Maclaurin et al., 2015). We explore it in greater depth and demonstrate the idea of dataset distillation in various settings. More crucially, our distilled images work well across random initialization weights, not possible by prior work.

Understanding datasets. Researchers have presented various approaches for understanding and visualizing learned models (Zeiler and Fergus, 2014; Zhou et al., 2015; Mahendran and Vedaldi, 2015; Bau et al., 2017; Koh and Liang, 2017). Unlike these approaches, we are interested in understanding the intrinsic properties of the training data rather than a specific trained model. Analyzing training datasets has, in the past, been mainly focused on the investigation of bias in datasets (Ponce et al., 2006; Torralba and Efros, 2011). For example, Torralba and Efros (2011) proposed to quantify the "value" of dataset samples using cross-dataset generalization. Our method offers a different perspective for understanding datasets by distilling full datasets into a few synthetic samples.

3 FORMULATION

Given a model and a dataset, we aim to obtain a new, much-reduced *synthetic* dataset which performs almost as well as the original dataset. We first present our main optimization algorithm for training a network with a fixed initialization with one gradient descent (GD) step (Section 3.1). In Section 3.2, we derive the resolution to a more challenging case, where initial weights are random rather than fixed. In Section 3.3, we further study a linear network case to help readers understand both the properties and limitations of our method. We also discuss the distribution of initial weights with which our method can work well. In Section 3.4, we extend our approach to reuse the same distilled images over $2,000$ gradient descent steps and largely improve the performance. Finally, Section 3.5 discusses dataset distillation for different initialization distributions. Finally, in Section 3.6, we show that our distilled images can be used as effective episodic memory for continual learning tasks.

Consider a training dataset $\mathbf{x} = \{x_i\}_{i=1}^N$, we parameterize our neural network as θ and denote $\ell(x_i, \theta)$ as the loss function that represents the loss of this network on a data point x_i. Our task is to find the minimizer of the empirical error over entire training data:

$$\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \ell(x_i, \theta) \triangleq \arg\min_\theta \ell(\mathbf{x}, \theta), \tag{1}$$

where for notation simplicity we overload the $\ell(\cdot)$ notation so that $\ell(\mathbf{x}, \theta)$ represents the average error of θ over the entire dataset. We make the mild assumption that ℓ is twice-differentiable, which holds true for the majority of modern machine learning models and tasks.

3.1 OPTIMIZING DISTILLED DATA

Standard training usually applies minibatch stochastic gradient descent or its variants. At each step t, a minibatch of training data $\mathbf{x}_t = \{x_{t,j}\}_{j=1}^n$ is sampled to update the current parameters as

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} \ell(\mathbf{x}_t, \theta_t),$$

where η is the learning rate. Such a training process often takes tens of thousands or even millions of update steps to converge. Instead, we learn a tiny set of synthetic distilled training data $\tilde{\mathbf{x}} = \{\tilde{x}_i\}_{i=1}^M$ with $M \ll N$ and a corresponding learning rate $\tilde{\eta}$ so that a single GD step such as

$$\theta_1 = \theta_0 - \tilde{\eta} \nabla_{\theta_0} \ell(\tilde{\mathbf{x}}, \theta_0) \tag{2}$$

Algorithm 1 Dataset Distillation

Input: $p(\theta_0)$: distribution of initial weights; M: the number of distilled data
Input: α: step size; n: batch size; T: the number of optimization iterations; $\tilde{\eta}_0$: initial value for $\tilde{\eta}$
1: Initialize $\tilde{\mathbf{x}} = \{\tilde{x}_i\}_{i=1}^{M}$ either from $\mathcal{N}(0, I)$ or from real training images. Initialize $\tilde{\eta} \leftarrow \tilde{\eta}_0$
2: **for each** training step $t = 1$ to T **do**
3: Get a minibatch of real training data $\mathbf{x}_t = \{x_{t,j}\}_{j=1}^{n}$
4: Sample a batch of initial weights $\theta_0^{(j)} \sim p(\theta_0)$
5: **for each** sampled $\theta_0^{(j)}$ **do**
6: Compute updated parameter with GD: $\theta_1^{(j)} = \theta_0^{(j)} - \tilde{\eta}\nabla_{\theta_0^{(j)}}\ell(\tilde{\mathbf{x}}, \theta_0^{(j)})$
7: Evaluate the objective function on real training data: $\mathcal{L}^{(j)} = \ell(\mathbf{x}_t, \theta_1^{(j)})$
8: **end for**
9: Update $\tilde{\mathbf{x}} \leftarrow \tilde{\mathbf{x}} - \alpha\nabla_{\tilde{\mathbf{x}}}\sum_j \mathcal{L}^{(j)}$, and $\tilde{\eta} \leftarrow \tilde{\eta} - \alpha\nabla_{\tilde{\eta}}\sum_j \mathcal{L}^{(j)}$
10: **end for**
Output: distilled data $\tilde{\mathbf{x}}$ and optimized learning rate $\tilde{\eta}$

using these learned synthetic data $\tilde{\mathbf{x}}$ can greatly boost the performance on the real test set. Given an initial θ_0, we obtain these synthetic data $\tilde{\mathbf{x}}$ and learning rate $\tilde{\eta}$ by minimizing the objective below $\mathcal{L}$:

$$\begin{aligned}\tilde{\mathbf{x}}^*, \tilde{\eta}^* &= \arg\min_{\tilde{\mathbf{x}}, \tilde{\eta}} \mathcal{L}(\tilde{\mathbf{x}}, \tilde{\eta}; \theta_0) \\ &\triangleq \arg\min_{\tilde{\mathbf{x}}, \tilde{\eta}} \ell(\mathbf{x}, \theta_1) \\ &= \arg\min_{\tilde{\mathbf{x}}, \tilde{\eta}} \ell(\mathbf{x}, \theta_0 - \tilde{\eta}\nabla_{\theta_0}\ell(\tilde{\mathbf{x}}, \theta_0)),\end{aligned} \tag{3}$$

where we derive the new weights θ_1 as a function of distilled data $\tilde{\mathbf{x}}$ and learning rate $\tilde{\eta}$ using Equation 2 and then evaluate the new weights over all the real training data $\mathbf{x}$. The loss $\mathcal{L}(\tilde{\mathbf{x}}, \tilde{\eta}; \theta_0)$ is differentiable w.r.t. $\tilde{\mathbf{x}}$ and $\tilde{\eta}$, and can thus be optimized using standard gradient-based methods. In many classification tasks, the data $\mathbf{x}$ may contain discrete parts, e.g., class labels in data-label pairs. For such cases, we fix the discrete parts rather than learn them.

3.2 Distillation for random initializations

Unfortunately, the above distilled data is optimized for a given initialization, and does not generalize well to other initializations, as it encodes the information of both the training dataset $\mathbf{x}$ and a particular network initialization θ_0. To address this issue, we turn to calculate a small number of distilled data that can work for networks with random initializations from a specific distribution. We formulate the optimization problem as follows:

$$\tilde{\mathbf{x}}^*, \tilde{\eta}^* = \arg\min_{\tilde{\mathbf{x}}, \tilde{\eta}} \mathbb{E}_{\theta_0 \sim p(\theta_0)}\mathcal{L}(\tilde{\mathbf{x}}, \tilde{\eta}; \theta_0), \tag{4}$$

where the network initialization θ_0 is randomly sampled from a distribution $p(\theta_0)$. During our optimization, the distilled data are optimized to work well for randomly initialized networks. In practice, we observe that the final distilled data generalize well to unseen initializations. In addition, these distilled images often look quite informative, encoding the discriminative features of each category (e.g., in Figure 2). Algorithm 1 illustrates our main method.

As the optimization (Equation 4) is highly non-linear and complex, the initialization of $\tilde{\mathbf{x}}$ plays a critical role in the final performance. We experiment with different initialization strategies and observe that using random real images as initialization often produces better distilled images compared to random initialization, e.g., $\mathcal{N}(0, I)$.

For a compact set distilled data to be properly learned, it turns out having only one GD step is far from sufficient. Next, we derive a lower bound on the size of distilled data needed for a simple model with arbitrary initial θ_0 in one GD step, and discuss its implications on our algorithm.

3.3 Analysis of a simple linear case

This section studies our formulation in a simple linear regression problem with quadratic loss. We derive a lower bound of the size of distilled data needed to achieve the same performance as training

on the full dataset for arbitrary initialization with one GD step. Consider a dataset $\mathbf{x}$ containing N data-target pairs $\{(d_i, t_i)\}_{i=1}^N$, where $d_i \in \mathbb{R}^D$ and $t_i \in \mathbb{R}$, which we represent as two matrices: an $N \times D$ data matrix $\mathbf{d}$ and an $N \times 1$ target matrix $\mathbf{t}$. Given the mean squared error metric and a $D \times 1$ weight matrix θ, we have

$$\ell(\mathbf{x}, \theta) = \ell((\mathbf{d}, \mathbf{t}), \theta) = \frac{1}{2N} \|\mathbf{d}\theta - \mathbf{t}\|^2. \tag{5}$$

We aim to learn M synthetic data-target pairs $\tilde{\mathbf{x}} = (\tilde{\mathbf{d}}, \tilde{\mathbf{t}})$, where $\tilde{\mathbf{d}}$ is an $M \times D$ matrix, $\tilde{\mathbf{t}}$ an $M \times 1$ matrix ($M \ll N$), and $\tilde{\eta}$ the learning rate, to minimize $\ell(\mathbf{x}, \theta_0 - \tilde{\eta}\nabla_{\theta_0}\ell(\tilde{\mathbf{x}}, \theta_0))$. The updated weight matrix after one GD step with these distilled data is

$$\begin{aligned}
\theta_1 &= \theta_0 - \tilde{\eta}\nabla_{\theta_0}\ell(\tilde{\mathbf{x}}, \theta_0) \\
&= \theta_0 - \frac{\tilde{\eta}}{M}\tilde{\mathbf{d}}^T(\tilde{\mathbf{d}}\theta_0 - \tilde{\mathbf{t}}) \\
&= (\mathbf{I} - \frac{\tilde{\eta}}{M}\tilde{\mathbf{d}}^T\tilde{\mathbf{d}})\theta_0 + \frac{\tilde{\eta}}{M}\tilde{\mathbf{d}}^T\tilde{\mathbf{t}}.
\end{aligned} \tag{6}$$

For the quadratic loss, there always exists distilled data $\tilde{\mathbf{x}}$ that can achieve the same performance as training on the full dataset $\mathbf{x}$ (i.e., attaining the global minimum) for *any* initialization θ_0. For example, given any global minimum solution θ^*, we can choose $\tilde{\mathbf{d}} = N \cdot \mathbf{I}$ and $\tilde{\mathbf{t}} = N \cdot \theta^*$. But how small can the size of the distilled data be? For such models, the global minimum is attained at any θ^* satisfying $\mathbf{d}^T\mathbf{d}\theta^* = \mathbf{d}^T\mathbf{t}$. Substituting Equation 6 in the condition above, we have

$$\mathbf{d}^T\mathbf{d}(\mathbf{I} - \frac{\tilde{\eta}}{M}\tilde{\mathbf{d}}^T\tilde{\mathbf{d}})\theta_0 + \frac{\tilde{\eta}}{M}\mathbf{d}^T\mathbf{d}\tilde{\mathbf{d}}^T\tilde{\mathbf{t}} = \mathbf{d}^T\mathbf{t}. \tag{7}$$

Here we make the mild assumption that the feature columns of the data matrix $\mathbf{d}$ are independent (i.e., $\mathbf{d}^T\mathbf{d}$ has full rank). For a $\tilde{\mathbf{x}} = (\tilde{\mathbf{d}}, \tilde{\mathbf{t}})$ to satisfy the above equation for any θ_0, we must have

$$\mathbf{I} - \frac{\tilde{\eta}}{M}\tilde{\mathbf{d}}^T\tilde{\mathbf{d}} = \mathbf{0}, \tag{8}$$

which implies that $\tilde{\mathbf{d}}^T\tilde{\mathbf{d}}$ has full rank and $M \geq D$.

Discussion. The analysis above only considers a simple case but suggests that any small number of distilled data fail to generalize to arbitrary initial θ_0. This is intuitively expected as the optimization target $\ell(\mathbf{x}, \theta_1) = \ell(\mathbf{x}, \theta_0 - \tilde{\eta}\nabla_{\theta_0}\ell(\tilde{\mathbf{x}}, \theta_0))$ depends on the local behavior of $\ell(\mathbf{x}, \cdot)$ around θ_0 (e.g., gradient magnitude), which can be drastically different across various initializations θ_0. The lower bound $M \geq D$ is a quite restricting one, considering that real datasets often have thousands to even hundreds of thousands of dimensions (e.g., images). This analysis motivates us to avoid the limitation of using one GD step by extending to multiple steps in the next section.

3.4 Multiple gradient descent steps

We extend Algorithm 1 to more than one gradient descent steps by changing Line 6 to multiple sequential GD steps on the same batch of distilled data, i.e., each step i performs

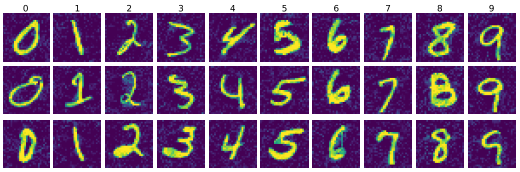
$$\theta_{i+1} = \theta_i - \tilde{\eta}_i\nabla_{\theta_i}\ell(\tilde{\mathbf{x}}, \theta_i), \tag{9}$$

and changing Line 9 to backpropagate through all steps. We do not share the same learning rates across steps as later steps often require lower learning rates.

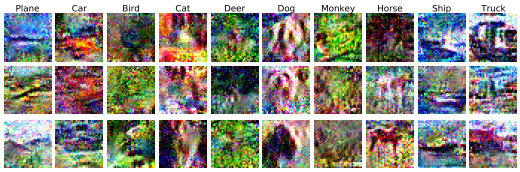
Naively computing gradients is memory and computationally intensive. Therefore, we exploit a recent technique called back-gradient optimization, which allows for significantly faster gradient calculation in reverse-mode differentiation (Domke, 2012; Maclaurin et al., 2015). Specifically, back-gradient optimization formulates the necessary second-order terms into efficient Hessian-vector products (Pearlmutter, 1994), which can be easily calculated with modern automatic differentiation systems such as PyTorch (Paszke et al., 2017).

3.5 Distribution of initial weights

There is freedom in choosing the distribution of initial weights $p(\theta_0)$. In this work, we explore the following four practical choices in the experiments:



(a) MNIST. These distilled images can train unknown random initializations to $88.51\% \pm 1.11\%$ test accuracy in 2000 GD steps.



(b) CIFAR10. These distilled images can train unknown random initializations to $41.23\% \pm 0.88\%$ test accuracy in 50 GD steps.

Figure 2: Distilled images trained for *random initialization* a batch of 100 distilled images (ten per class). Only 30 of 100 distilled images are shown here. Please see the appendix for the full result.

- **Random initialization:** Distribution over random initial weights, e.g., He Initialization (He et al., 2015) and Xavier Initialization (Glorot and Bengio, 2010) for neural networks.
- **Fixed initialization:** A particular fixed network initialized by the method above.
- **Random pre-trained weights:** Distribution over models pre-trained on other tasks or datasets, e.g., ALEXNET (Krizhevsky et al., 2012) networks trained on ImageNet (Deng et al., 2009).
- **Fixed pre-trained weights:** A particular fixed network pre-trained on other tasks and datasets.

Distillation with pre-trained weights. Such learned distilled data essentially fine-tune weights pre-trained on one dataset to perform well for a new dataset, thus bridging the gap between the two domains. Domain mismatch and dataset bias represent a challenging problem in machine learning (Torralba and Efros, 2011; Daume III, 2007; Saenko et al., 2010). In this work, we characterize the domain mismatch via distilled data. In Section 4.1.2, we show that a small number of distilled images are sufficient to quickly adapt convolutional neural network (CNN) models to new datasets and tasks.

3.6 APPLICATION TO CONTINUAL LEARNING

To guard against domain shift, several continual learning methods store a subset of training samples in a small memory buffer, and restrict future updates to maintain reasonable performance on these stored samples (Rebuffi et al., 2017; Kirkpatrick et al., 2017; Lopez-Paz and Ranzato, 2017; Nguyen et al., 2018). As our distilled images contain rich information about the past training data and task, they could naturally serve as a compressed memory of the past.

To test this, we modify a recent continual learning method called Gradient Episodic Memory (GEM) (Lopez-Paz and Ranzato, 2017). GEM enforces inequality constraints such that the new model, after being trained on the new data and task, should perform at least as well as the old model on the previously stored data and tasks. Here, we store our distilled data for each task instead of randomly drawn training samples as used in GEM. We use the distilled data to construct inequality constraints, and solve the optimization using quadratic programming, same as in GEM. As shown in Section 4.2, our method compares favorably against several baselines that rely on real images.

4 EXPERIMENTS

In this section, we report experiments of regular image classifications on MNIST (LeCun, 1998) and CIFAR10 (Krizhevsky and Hinton, 2009), adaptation from ImageNet (Deng et al., 2009) to PASCAL-VOC (Everingham et al., 2010) and CUB-200 (Wah et al., 2011), and continual learning on permuted MNIST and CIFAR100.

Baselines. For each experiment, in addition to baselines specific to the setting, we generally compare our method against baselines trained with data derived or selected from real training images:

- **Random real images:** We randomly sample the same number of real images per category.
- **Optimized real images:** We sample different sets of random real images as above, and choose the top 20% best performing sets.
- **k-means++:** We apply k-means++ (Arthur and Vassilvitskii, 2007) clustering to each category, and extract the cluster centroids.

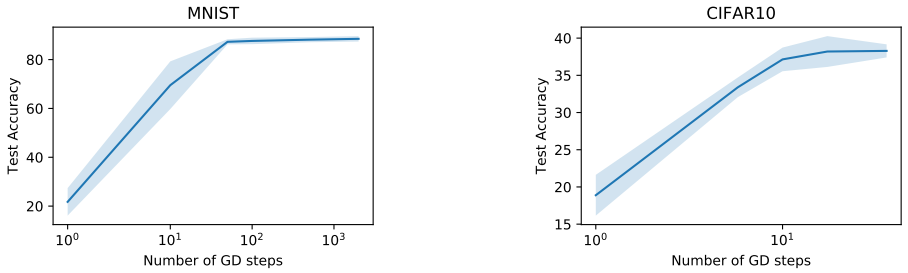


Figure 3: Distillation performance with varying numbers of GD steps and a fixed number of distilled images.

	Ours		Baselines					
	Fixed init.	Random init.	Used as training data in CNN				Used in KNN classification	
			Random real	Optimized real	k-means++	Average real	Random real	k-means++
MNIST	**94.4**	88.5 ± 1.1	82.8 ± 1.8	83.8 ± 2.1	86.7 ± 1.4	77.7 ± 2.7	71.5 ± 2.1	$\mathbf{92.4 \pm 0.2}$
CIFAR10	**45.2**	$\mathbf{41.2 \pm 0.9}$	24.8 ± 1.5	24.9 ± 1.4	26.7 ± 1.8	22.8 ± 0.8	18.8 ± 1.3	29.4 ± 0.4

Table 1: Comparison between our method and various baselines. All methods use ten images per category (100 in total), except for the average real images baseline, which reuses the same images in different GD steps. For MNIST, our method uses 2000 GD steps, and baselines use the best among #steps $\in \{1, 100, 500, 1000, 2000\}$. For CIFAR10, our method uses 50 GD steps, and baselines use the best among #steps $\in \{1, 5, 10, 20, 500\}$. In addition, we include a K-nearest neighbors (KNN) baseline, and report best results among all combinations of distance metric $\in \{l_1, l_2\}$ and one or three neighbors.

- **Average real images:** We compute the average image for each category.

Please see the appendix for more details about training and baselines, and additional results.

4.1 Dataset distillation

We first present experimental results on training classifiers either from scratch or adapting from pre-trained weights. For MNIST, the distilled images are trained with LeNet (LeCun et al., 1998), which achieves about 99% test accuracy if conventionally trained. For CIFAR10, we use a network architecture (Krizhevsky, 2012) that achieves around 80% test accuracy if conventionally trained. For ImageNet adaptations, we use an AlexNet (Krizhevsky et al., 2012). We use 2000 GD steps for MNIST and 50 GD steps for CIFAR10. For random initializations and random pre-trained weights, we report means and standard deviations over 200 held-out models, unless otherwise stated.

For baselines, we perform each evaluation on 200 held-out models using all possible combinations of learning rate $\in \{$distilled learning rates $\tilde{\eta}^*$, 1e-3, 3e-3, 1e-2, 3e-2, 1e-1, 3e-1$\}$ and several choices of numbers of training GD steps (see table captions for details), and report results with the best performing combination.

4.1.1 Distillation with weights sampled fro network initialization

Fixed initialization. With access to initial network weights, distilled images can directly train a fixed network to reach high performance. Experiment results show that just 10 distilled images (one per class) can boost the performance of a LeNet with an initial accuracy 8.25% to a final accuracy of 93.82% on MNIST in 2000 GD steps. Using 100 distilled images (ten per class) can raise the final accuracy can be raised to 94.41%, as shown in the first column of Table 1. Similarly, 100 distilled images can train a network with an initial accuracy 10.75% to test accuracy of 45.15% on CIFAR10 in 50 GD steps.

Random initialization. Figure 2 distilled images trained with randomly sampled initializations using Xavier Initialization (Glorot and Bengio, 2010). While the resulting average test accuracy from these images are not as high as those for fixed initialization, these distilled images crucially do not require a specific initial point, and thus could potentially generalize to a much wider range of starting points. In Section 4.2 below, we present preliminary results of achieving nontrivial gains from applying such distilled images to classifier networks during a continual learning training process.

	Ours w/ fixed pre-trained	Ours w/ random pre-trained	Random real	Optimized real	k-means++	Average real	Few-shot adaptation Motiian et al. (2017)	No adaptation	Train on **full** target dataset
$\mathcal{M} \rightarrow \mathcal{U}$	$\mathbf{97.9}$	$\mathbf{95.4 \pm 1.8}$	94.9 ± 0.8	95.2 ± 0.7	94.8 ± 0.7	93.9 ± 0.8	$\mathbf{96.7 \pm 0.5}$	90.4 ± 3.0	97.3 ± 0.3
$\mathcal{U} \rightarrow \mathcal{M}$	$\mathbf{93.2}$	$\mathbf{92.7 \pm 1.4}$	87.1 ± 2.9	87.6 ± 2.1	88.0 ± 2.2	78.4 ± 5.0	89.2 ± 2.4	67.5 ± 3.9	98.6 ± 0.5
$\mathcal{S} \rightarrow \mathcal{M}$	$\mathbf{96.2}$	85.2 ± 4.7	84.6 ± 2.1	85.2 ± 1.2	$\mathbf{86.5 \pm 1.2}$	74.9 ± 2.6	74.0 ± 1.5	51.6 ± 2.8	98.6 ± 0.5

Table 2: Adapting models among MNIST ($\mathcal{M}$), USPS ($\mathcal{U}$), and SVHN ($\mathcal{S}$) using 100 distilled images. Our method outperforms few-shot domain adaptation (Motiian et al., 2017) and other baselines in most settings. Due to computation limitations, the 100 distilled images are split into 10 minibatches applied in 10 sequential GD steps, and the entire set of 100 distilled images is iterated through 3 times (30 GD steps in total). For baselines, we train the model using the same number of images with $\{1, 3, 5\}$ times and report the best result.

Target dataset	Ours	Random real	Optimized real	Average real	Fine-tune on **full** target dataset
PASCAL-VOC	$\mathbf{70.75}$	19.41 ± 3.73	23.82 ± 3.66	9.94	75.57 ± 0.18
CUB-200	$\mathbf{38.76}$	7.11 ± 0.66	7.23 ± 0.78	2.88	41.21 ± 0.51

Table 3: Adapting an ALEXNET pre-trained on ImageNet to PASCAL-VOC and CUB-200. We use one distilled image per category, repeatedly applied via three GD steps. Our method significantly outperforms the baselines. For baselines, we train the model with $\{1, 3, 5\}$ GD steps and report the best. Results are over 10 runs.

Multiple gradient descent steps. Section 3.3 has shown theoretical limitations of using only one step in a simple linear case. In Figure 3, we empirically verify for deep networks that using multiple steps drastically outperforms the single step method, given the same number of distilled images. Table 1 summarizes the results of our method and all baselines. Our method with both fixed and random initializations outperforms all the baselines on CIFAR10 and most of the baselines on MNIST.

4.1.2 DISTILLATION WITH PRE-TRAINED INITIAL WEIGHTS

Next, we show the extended setting of our algorithm discussed in Section 3.5, where the weights are not randomly initialized but pre-trained on a particular dataset. In this section, for random initial weights, we train the distilled images on 2000 pre-trained models and evaluate them on 200 *unseen* models.

Fixed and random pre-trained weights on digits. As shown in Section 3.5, we can optimize distilled images to quickly fine-tune pre-trained models on a new dataset. Table 2 shows that our method is more effective than various baselines on adaptation between three digits datasets: MNIST, USPS (Hull, 1994), and SVHN (Netzer et al., 2011). We also compare our method against a state-of-the-art few-shot domain adaptation method (Motiian et al., 2017). Although our method uses the entire training set to compute the distilled images, both methods use the same number of images to distill the knowledge of target dataset. Prior work (Motiian et al., 2017) is outperformed by our method with fixed pre-trained weights on all the tasks, and by our method with random pre-trained weights on two of the three tasks. This result shows that our distilled images effectively compress the information of target datasets.

Fixed pre-trained ALEXNET to PASCAL-VOC and CUB-200. In Table 3, we adapt a widely used ALEXNET model pre-trained on ImageNet to image classification on PASCAL-VOC and CUB-200 datasets. Given only one distilled image per category, our method outperforms various baselines significantly. Our method is on par with fine-tuning on the full datasets with thousands of images.

4.2 APPLICATION TO CONTINUAL LEARNING

We modify Gradient Episodic Memory (GEM) (Lopez-Paz and Ranzato, 2017) to store distilled data for each task rather than real training images. Experiments in Lopez-Paz and Ranzato (2017) use large memory buffers, up to 25% of the training set. Instead, we focus on a more realistic scenario where the buffer is rather small ($\leq 1\%$ of the training set). Following the experiment settings and architecture choices from Lopez-Paz and Ranzato (2017), we consider two continual learning tasks:

		Permuted MNIST	CIFAR100
Memory size per task = 10	iCaRL (Rebuffi et al., 2017)	–	42.4
	GEM (Lopez-Paz and Ranzato, 2017)	67.4	43.8
	GEM + Ours	**75.6**	**52.8**
Memory size per task = 40	iCaRL	–	45.8
	GEM	75.3	51.6
Memory size per task = 50	iCaRL	–	46.9
	GEM	75.8	52.4
No memory buffer	EWC (Kirkpatrick et al., 2017)	63.5	45.6

Table 4: Continual learning results. Distilled images are trained with random Xavier Initialization distribution. For permuted MNIST, they are trained with 2000 GD steps. For CIFAR100, they are trained for 200 GD steps.

- **Permuted MNIST**: 20 classification tasks each formed by using a different permutation to arrange pixels from MNIST images. Each task contains $1,000$ training images. The classifier used has 2 hidden layers each with 100 neurons.
- **CIFAR100**: 20 classification tasks formed by splitting the 100 classes into 20 equal subsets of 5 classes. Each task contains $2,500$ training images. The classifier used is RESNET18 (He et al., 2016).

Table 4 shows that using distilled data drastically improves final overall accuracy on all tasks, and reduces buffer size by up to $5\times$ compared to the original GEM that uses real images. We only report the basic iCaRL (Rebuffi et al., 2017) setting on CIFAR100 because it requires similar input distributions across all tasks, and it is unclear how to properly inject distilled images into its specialized examplar selection procedure.

The appendix details the hyper-parameters tested for each continual learning algorithm.

5 DISCUSSION AND LIMITATIONS

In this paper, we have presented dataset distillation for compressing the knowledge of entire training data into a few synthetic training images. We demonstrate how to train a network to reach surprisingly good performance with only a small number of distilled images. Finally, the distilled images can efficiently store the memory of previous tasks in the continual learning setting.

Many challenges remain for knowledge distillation of data. Although our method generalizes well to random initializations, it is still limited to a particular network architecture. Since loss surfaces for different architectures might be drastically different, a more flexible method of applying the distilled data may overcome this difficulty. Another limitation is the increasing computation and memory requirements for finding the distilled data as the number of images and steps increases. To compress large-scale datasets such as ImageNet, we may need first-order gradient approximations to make the optimization computationally feasible.

Nonetheless, we are encouraged by the findings in this paper on the possibilities of training large models with a few distilled data, leading to potential applications such as accelerating network evaluation in neural architecture search (Zoph and Le, 2017). We believe that the ideas developed in this work might give new insights into the quantity and type of data that deep networks are able to process, and hopefully inspire others to think along this direction.

REFERENCES

A. Angelova, Y. Abu-Mostafam, and P. Perona. Pruning training sets for learning of object categories. In *CVPR*, 2005.

D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007.

J. Ba and R. Caruana. Do deep nets really need to be deep? In *NIPS*, 2014.

O. Bachem, M. Lucic, and A. Krause. Practical coreset constructions for machine learning. *arXiv preprint arXiv:1703.06476*, 2017.

D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *CVPR*, 2017.

Y. Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900, 2000.

D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996.

H. Daume III. Frustratingly easy domain adaptation. In *ACL*, 2007.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

J. Domke. Generic methods for optimization-based modeling. In *Artificial Intelligence and Statistics*, pages 318–326, 2012.

M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010.

P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32(9):1627–1645, 2010.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, 2010.

S. A. Goldman and M. J. Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50 (1):20–31, 1995.

S. Har-Peled and A. Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19, 2007.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. In *CVPR*, 2017.

J. J. Hull. A database for handwritten text recognition research. *PAMI*, 16(5):550–554, 1994.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICLR*, 2015.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 114(13): 3521–3526, 2017.

P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *ICML*, 2017.

A. Krizhevsky. cuda-convnet: High-performance c++/cuda implementation of convolutional neural networks. *https://github.com/akrizhevsky/cuda-convnet2*, 2012.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

A. Lapedriza, H. Pirsiavash, Z. Bylinskii, and A. Torralba. Are all training examples equally valuable? *arXiv preprint arXiv:1311.6510*, 2013.

Y. LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *NIPS*, 2017.

D. Maclaurin, D. Duvenaud, and R. Adams. Gradient-based hyperparameter optimization through reversible learning. In *ICML*, 2015.

A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, 2015.

M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. 1989.

S. Motiian, Q. Jones, S. Iranmanesh, and G. Doretto. Few-shot adversarial domain adaptation. In *NIPS*, 2017.

Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop*, 2011.

C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner. Variational continual learning. In *ICLR*, 2018.

J. A. Olvera-López, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, and J. Kittler. A review of instance selection methods. *Artificial Intelligence Review*, 34(2):133–143, 2010.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *ICLR Workshop*, 2017.

B. A. Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.

F. Pedregosa. Hyperparameter optimization with approximate gradient. In *ICML*, 2016.

J. Ponce, T. L. Berg, M. Everingham, D. A. Forsyth, M. Hebert, S. Lazebnik, M. Marszalek, C. Schmid, B. C. Russell, A. Torralba, et al. Dataset issues in object recognition. In *Toward category-level object recognition*. 2006.

I. Radosavovic, P. Dollár, R. Girshick, G. Gkioxari, and K. He. Data distillation: Towards omni-supervised learning. In *CVPR*, 2018.

S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017.

A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

F. Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *ECCV*, 2010.

O. Sener and S. Savarese. Active learning for convolutional neural networks: A core-set approach. In *ICLR*, 2018.

A. Shinohara and S. Miyano. Teachability in computational learning. *New Generation Computing*, 8(4):337–347, 1991.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *JMLR*, 2 (Nov):45–66, 2001.

A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR*. IEEE, 2011.

I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast svm training on very large data sets. *JMLR*, 6(Apr):363–392, 2005.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Y. Wu and K. He. Group normalization. In *ECCV*, 2018.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. In *ICLR*, 2015.

X. Zhu. Machine teaching for bayesian learners in the exponential family. In *NIPS*, 2013.

X. Zhu. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *AAAI*, 2015.

B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

S-1 EXPERIMENT DETAILS

In our experiments, we disable dropout layers in the networks due to the randomness and computational cost they introduce in distillation. Moreover, we initialize the distilled learning rates with a constant between 0.001 and 0.02 depending on the task, and use the Adam solver (Kingma and Ba, 2015) with a learning rate of 0.001. For random initialization and random pre-trained weights, we sample 4 to 16 initial weights in each optimization step. We run all the experiments on NVIDIA 1080 Ti, 2080 Ti, Titan Xp, and V100 GPUs. We use one GPU for fixed initial weights and up to four GPUs for random initial weights. Each training typically takes 1 to 6 hours.

Below we describe the details of our baselines using real training images.

- **Random real images:** We randomly sample the same number of real images per category. We evaluate the performance over 10 randomly sampled sets.
- **Optimized real images:** We sample 50 sets of real images using the procedure above, pick 10 sets that achieve the best performance on 20 held-out models and 1024 randomly chosen training images, and evaluate the performance of these 10 sets.
- **k-means++:** For each category, we use k-means++ (Arthur and Vassilvitskii, 2007) clustering to extract the same number of cluster centroids as the number of distilled images in our method. We evaluate the method over 10 runs.
- **Average real images:** We compute the average image of all the images in each category, which is repeated to match the same total number of images. We evaluate the model only once because average images are deterministic.

To enforce our optimized learning rate to be positive, we apply `softplus` to a scalar trained parameter. For continual learning experiment on CIFAR10 dataset, to compare with GEM (Lopez-Paz and Ranzato, 2017), we replace the Batch normalization (Wu and He, 2018) with Group normalization (Ioffe and Szegedy, 2015) in RESNET18 (He et al., 2016), as it is difficult to run back-gradient optimization through batch norm running statistics. For a fair comparison, we use the same architecture for our method and other baselines.

For dataset distillation experiments with pre-trained initial weights, distilled images are initialized with $\mathcal{N}(0, 1)$ at the beginning of training. For other experiments, distilled images are initialized with random real samples, unless otherwise stated.

S-1.1 CONTINUAL LEARNING EXPERIMENT DETAILS

For the compared continual learning methods, we report the best report from the following combinations of hyper-parameters:

- GEM:
  - $\gamma \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$.
  - learning rate $= 0.1$.
- iCARL:
  - regularization $\in \{0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1.0\}$.
  - learning rate $= 0.1$.

S-2 ADDITIONAL EXPERIMENTS RESULTS

- Figures 4 and 5 show distilled images trained for random initializations on MNIST and CIFAR10.
- Figures 6, 7, and 8 show distilled images trained for adapting random pre-trained models on digits datasets including MNIST, USPS, and SVHN.

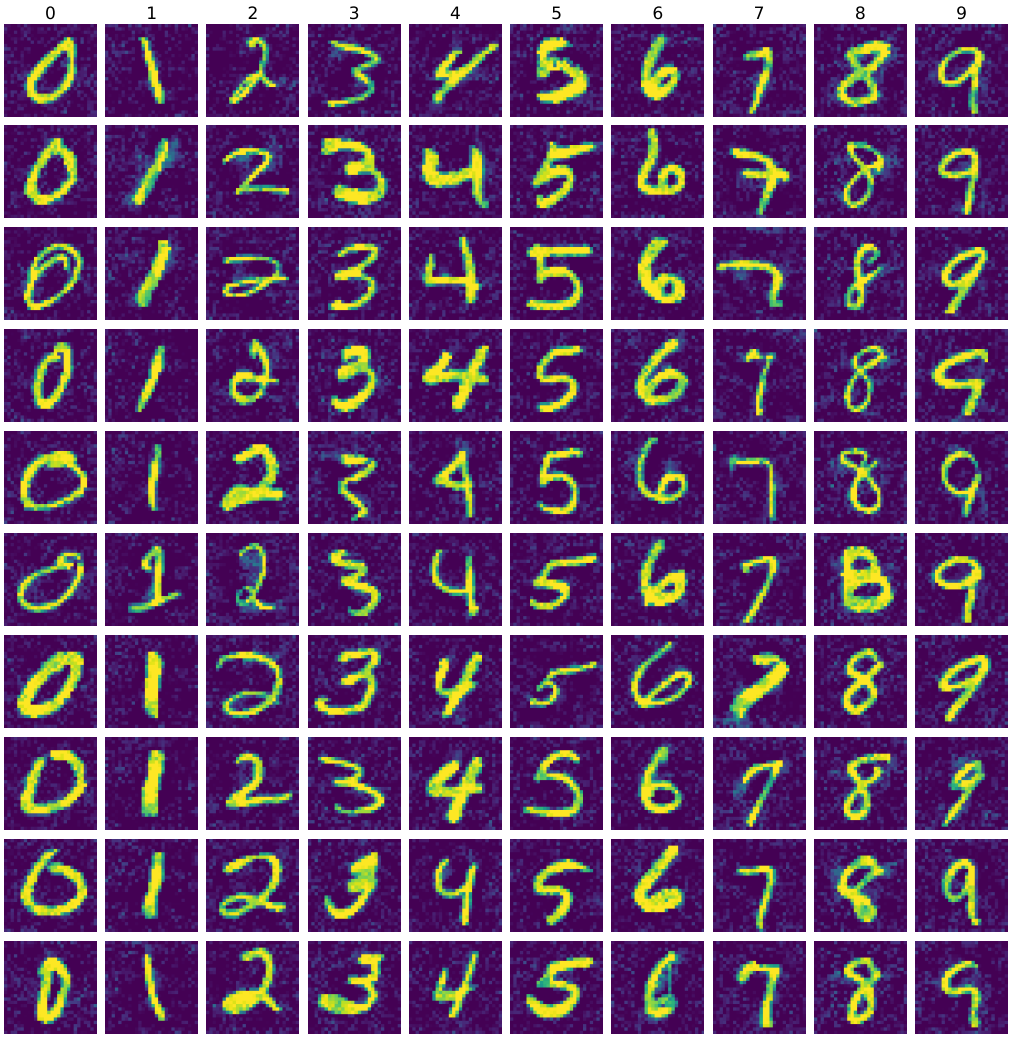


Figure 4: Dataset distillation for random initializations on MNIST. This batch of 100 distilled images are repeatedly applied in 2000 GD steps.. These images train average test accuracy to $88.51\% \pm 1.11\%$.

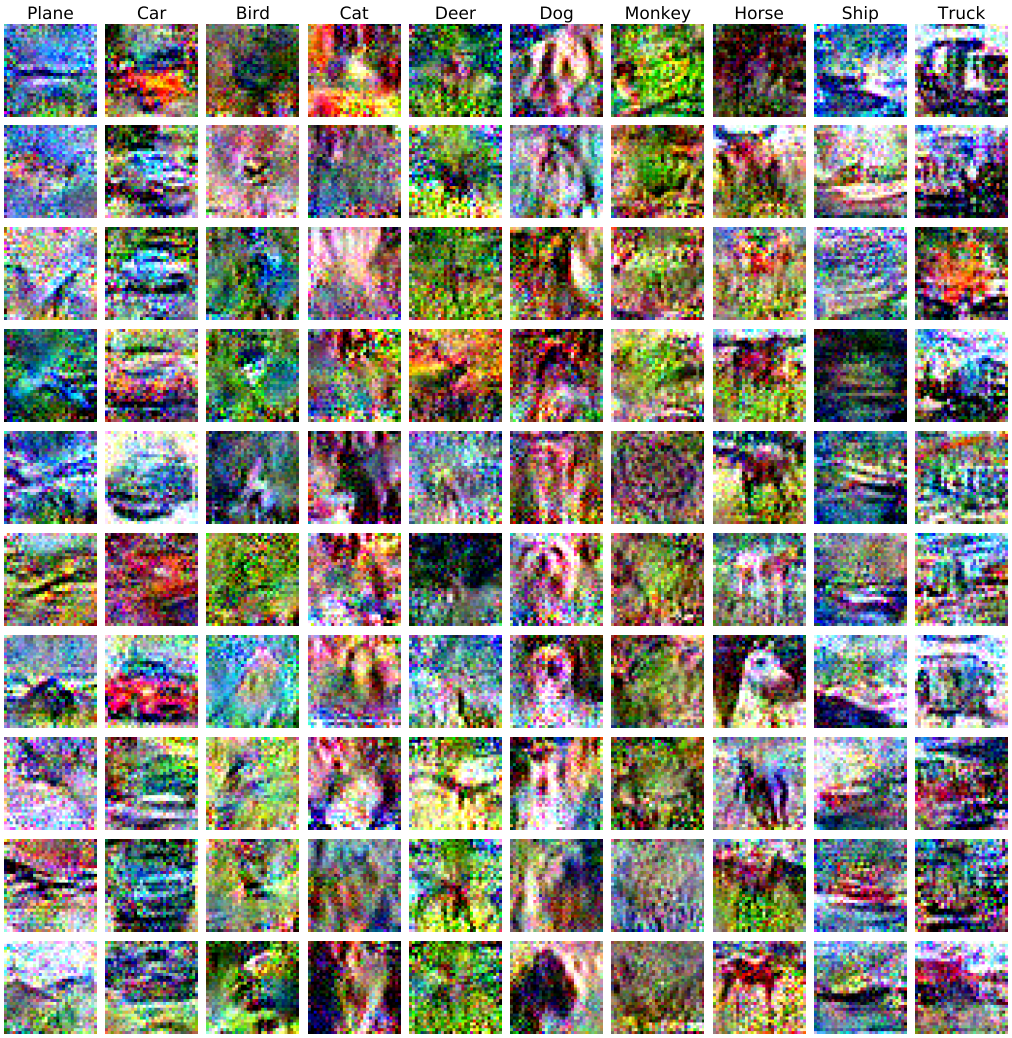


Figure 5: Dataset distillation for random initializations on CIFAR10. This batch of 100 distilled images are repeatedly applied in 50 GD steps. These images train average test accuracy to $41.23\% \pm 0.88\%$.

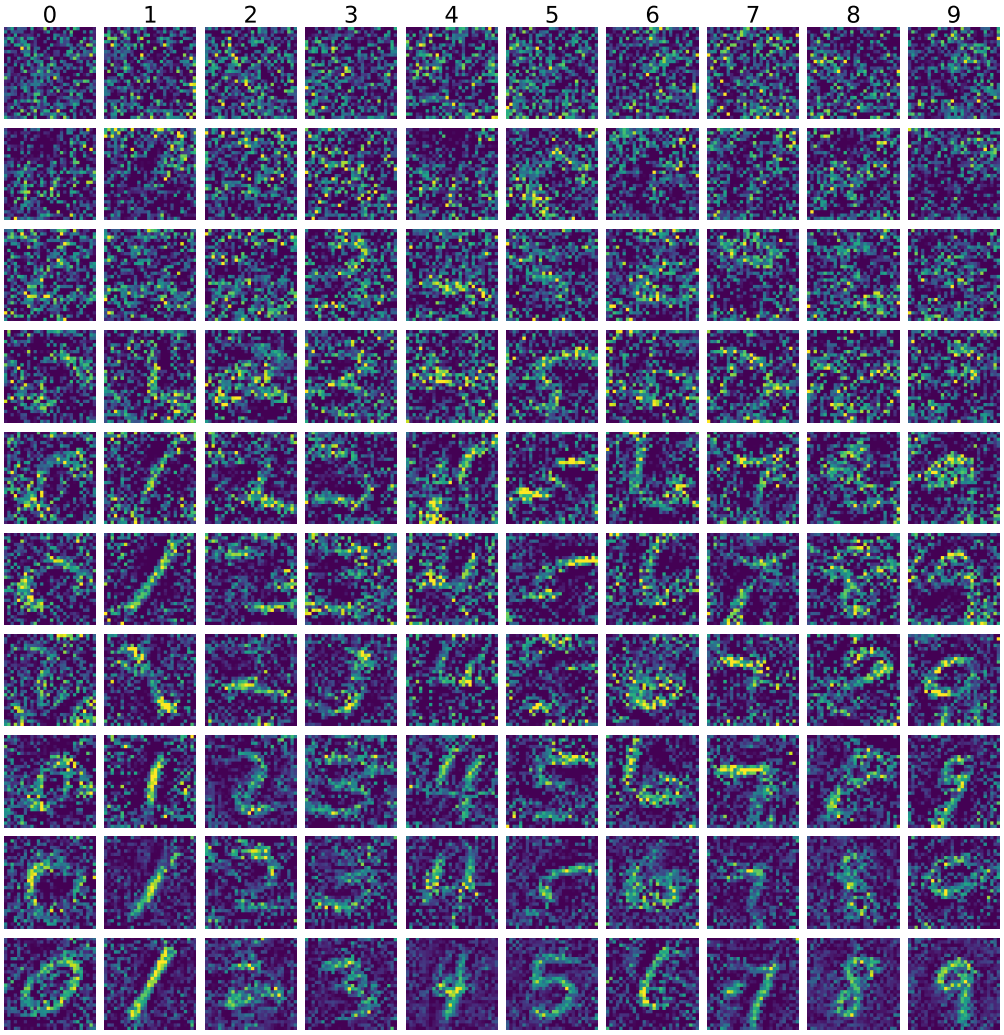


Figure 6: Dataset distillation for adapting random pretrained models from USPS to MNIST. 100 distilled images are split into 10 GD steps, shown as 10 rows here. Top row is the earliest GD step, and bottom row is the last. The 10 steps are iterated over three times to finish adaptation, leading to a total of 30 GD steps. These images train average test accuracy on 200 held out models from $67.54\% \pm 3.91\%$ to $92.74\% \pm 1.38\%$.

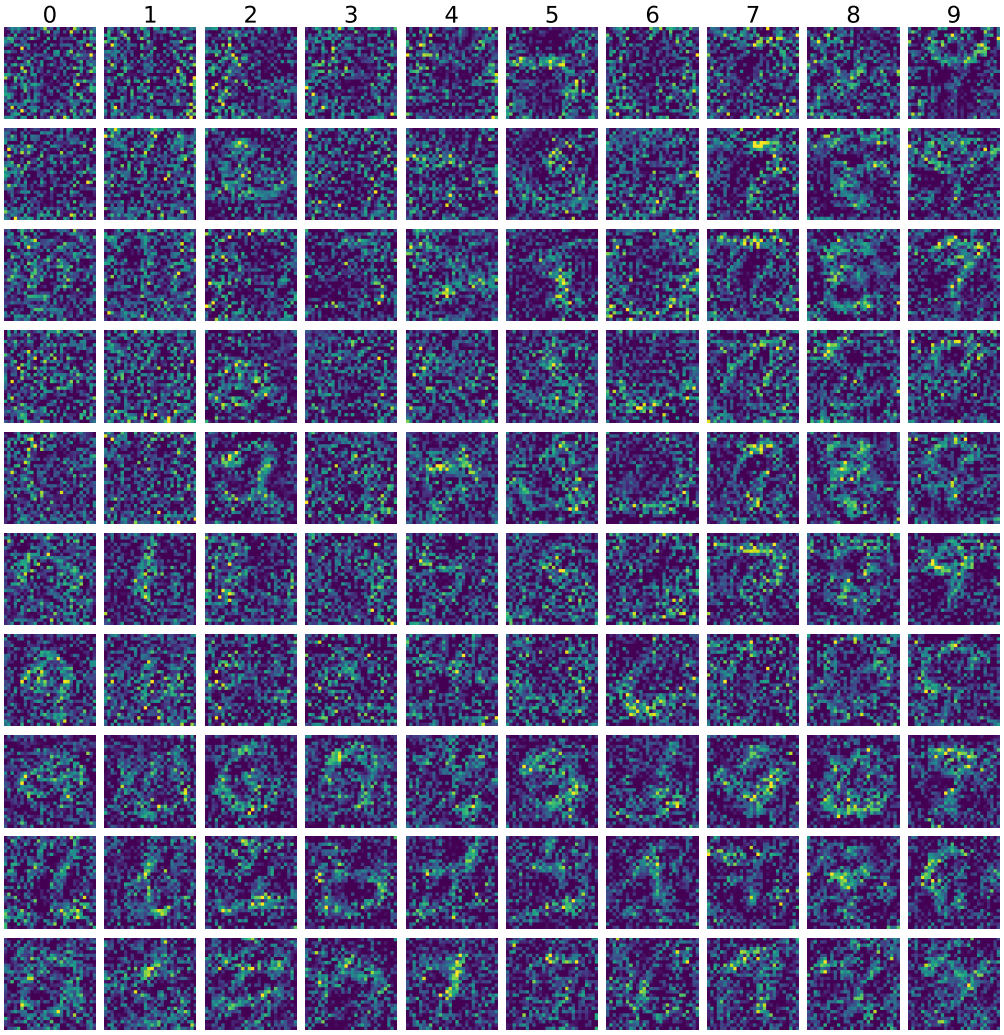


Figure 7: Dataset distillation for adapting random pretrained models from MNIST to USPS. 100 distilled images are split into 10 GD steps, shown as 10 rows here. Top row is the earliest GD step, and bottom row is the last. The 10 steps are iterated over three times to finish adaptation, leading to a total of 30 GD steps. These images train average test accuracy on 200 held out models from $90.43\% \pm 2.97\%$ to $95.38\% \pm 1.81\%$.

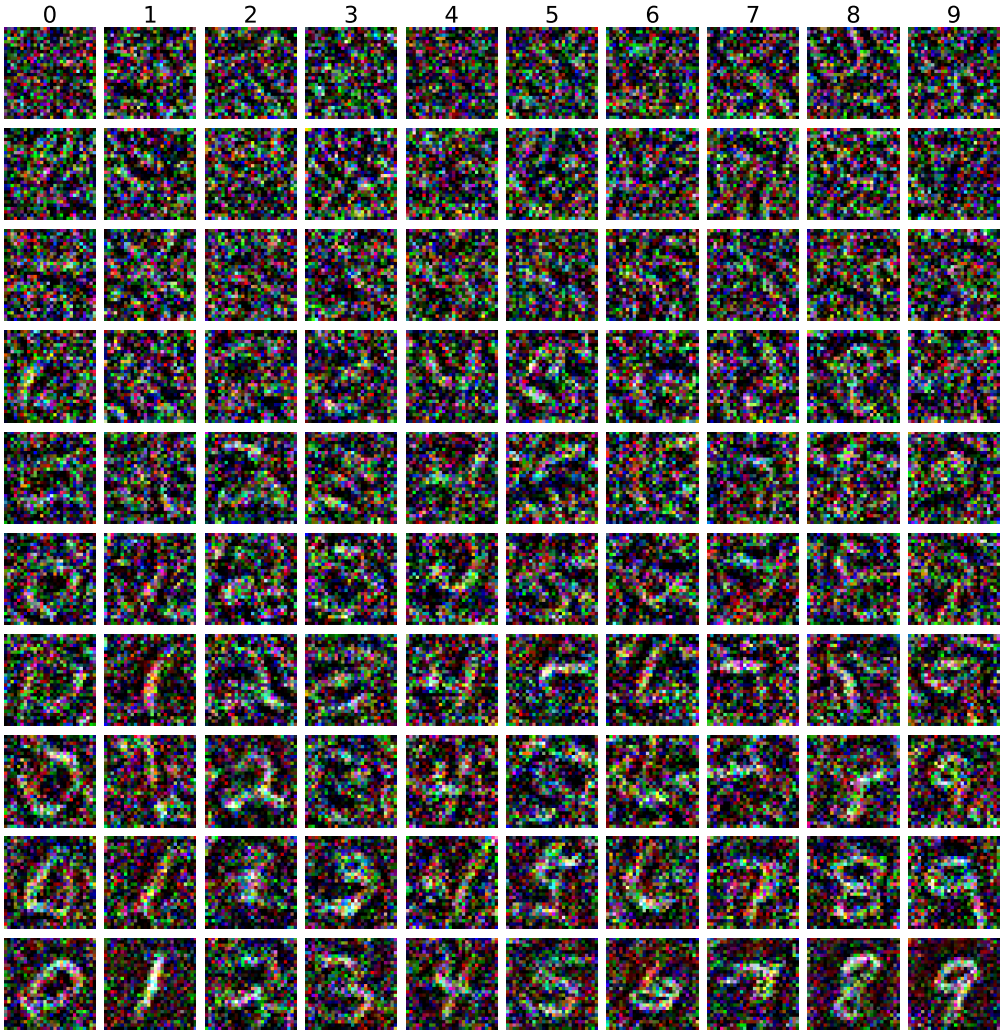


Figure 8: Dataset distillation for adapting random pretrained models from SVHN to MNIST. 100 distilled images are split into 10 GD steps, shown as 10 rows here. Top row is the earliest GD step, and bottom row is the last. The 10 steps are iterated over three times to finish adaptation, leading to a total of 30 GD steps. These images train average test accuracy on 200 held out models from $51.64\% \pm 2.77\%$ to $85.21\% \pm 4.73\%$.