DATASET DISTILLATION

Tongzhou Wang
Facebook AI Research, MIT CSAIL

Jun-Yan Zhu
MIT CSAIL

Antonio Torralba
MIT CSAIL

Alexei A. Efros
UC Berkeley

ABSTRACT

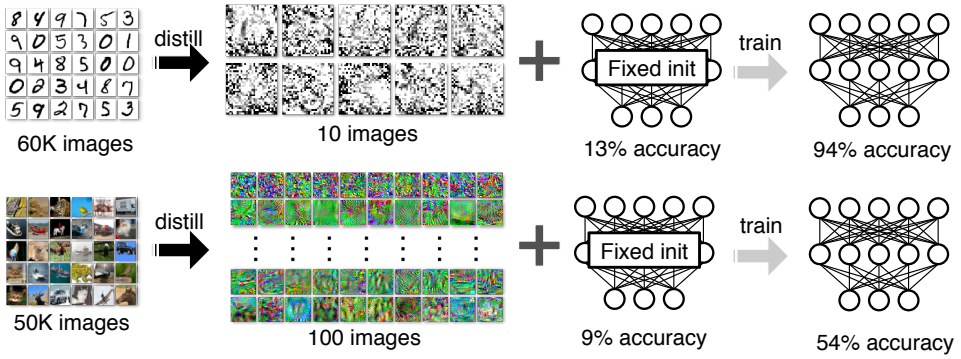
Model distillation aims to distill the knowledge of a complex model into a simpler one. In this paper, we consider an alternative formulation called *dataset distillation*: we keep the model fixed and instead attempt to distill the knowledge from a large training dataset into a small one. The idea is 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 original performance with only a few gradient descent steps, given a fixed network initialization. We evaluate our method in various initialization settings and with different learning objectives. Experiments on multiple datasets show the advantage of our approach compared to alternative methods.

1 INTRODUCTION

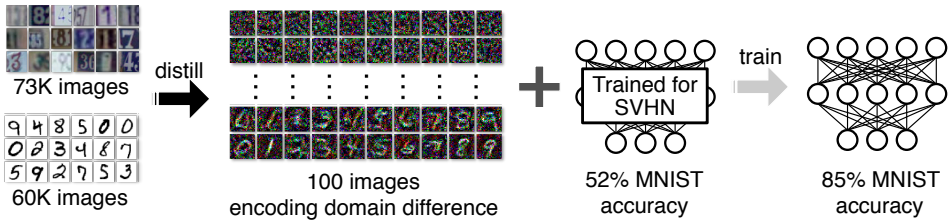
Hinton et al. (2015) proposed network distillation as a way to transfer the knowledge from an ensemble of many separately-trained networks into a single, typically compact network, performing a type of model compression. In this paper, we are considering a related but orthogonal task: rather than distilling 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 class), 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 80% with a few gradient descent steps. We name our method *Dataset Distillation* and these images *distilled images*.

But why is dataset distillation useful? There is the purely scientific question of how much data is encoded in a given training set and how compressible it is? Moreover, given a few distilled images, we can now "load up" a given network with an entire dataset-worth of knowledge much more efficiently, compared to traditional training that often uses tens of thousands of gradient descent steps.

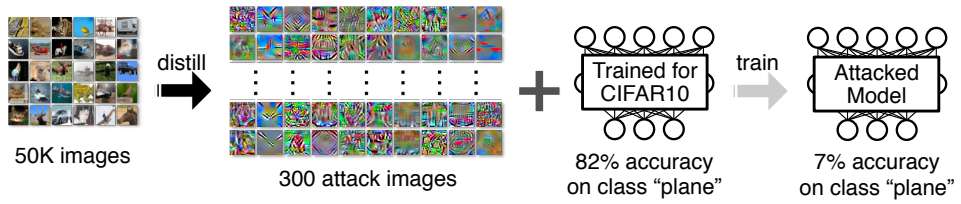
A key question is whether it is even possible to compress a dataset into a small set of synthetic data samples. For example, is it possible to train an image classification model on synthetic images out of the manifold of natural images? Conventional wisdom would suggest that the answer is no, as the 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 a new 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 in only a few gradient steps. To achieve our goal, we first derive the network weights as a 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



(a) Dataset distillation on MNIST and CIFAR10



(b) Dataset distillation can quickly fine-tune pre-trained networks on new datasets



(c) Dataset distillation can maliciously attack classifier networks

Figure 1: We distill the knowledge of tens of thousands of images into a few synthetic training images called distilled images. (a) On MNIST, 10 distilled images can train a standard LeNet with a fixed initialization to 94% test accuracy, compared to 99% when fully trained. On CIFAR10, 100 distilled images can train a network with a fixed initialization to 54% test accuracy, compared to 80% when fully trained. (b) We distill the domain difference between SVHN and MNIST into 100 images. These images can quickly fine-tune pre-trained SVHN networks to achieve high accuracy on MNIST. (c) Our formulation can create dataset poisoning images. After trained with these images for *one single* gradient step, networks will catastrophically misclassify one category.

images for randomly initialized networks. To further boost performance, we propose an iterative version, where we obtain a sequence of distilled images and these distilled images can be trained with multiple epochs (passes). 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, as well as two training objectives: image classification and malicious dataset poisoning attack. Extensive experiments on four publicly available datasets, MNIST, CIFAR10, PASCAL-VOC, and CUB-200, show that our approach often outperforms existing methods. Please check out our code and website for more details.

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 & 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 training images. Similar to our approach, data-free knowledge distillation also optimizes synthetic data samples, but with a different objective of matching activation statistics of a teacher model in knowledge distillation (Lopes et al., 2017). Our method is also related to the theoretical concept of teaching dimension, which specifies the size of dataset necessary to teach a target model to a learner (Shinohara & Miyano, 1991; Goldman & Kearns, 1995). However, methods (Zhu, 2013; 2015) inspired by this concept need the existence of target models, which our method does not require.

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 & Koller, 2001). Similarly, core-set construction (Tsang et al., 2005; Har-Peled & Kushal, 2007; Bachem et al., 2017; Sener & 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 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 a subset 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 & Fergus, 2014; Zhou et al., 2015; Mahendran & Vedaldi, 2015; Bau et al., 2017; Koh & 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 & Efros, 2011). For example, Torralba & Efros (2011) proposed to quantify the "value" of dataset samples using cross-dataset generalization. Our method offers a new perspective for understanding datasets by distilling full datasets into a few synthetic samples.

3 Approach

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 property and limitation of our method. We also discuss the initial weights distribution with which our method can work well. In Section 3.4, we extend our approach to more than one gradient descent steps and more than one epoch (pass). Finally, Section 3.5 and Section 3.6 demonstrate how to obtain distilled images with different initialization distributions and learning objectives.

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 aim to 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

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

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 $\mathcal{L}$:

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

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 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

固定初始化训练抽象数据，在训练模型时，必须采用同样的初始化，否则性能很差
可以随机初始化来训练抽象数据，但这样的性能不如固定初始化的

Unfortunately, the above distilled data optimized for a given initialization do not generalize well to other initializations. The distilled data often look like random noise (e.g., in Figure 2a) as it encodes the information of both 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. Algorithm 1 illustrates our main method. 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 3).

For distilled data to be properly learned, it turns out crucial for $\ell(\mathbf{x}, \cdot)$ to share similar local conditions (e.g., output values, gradient magnitudes) over initializations θ_0 sampled from $p(\theta_0)$. In the next section, we derive a lower bound on the size of distilled data needed for a simple model with arbitrary initial θ_0, and discuss its implications on choosing $p(\theta_0)$.

3.3 Analysis of a Simple Linear Case

This section studies our formulation in a simple linear regression problem with quadratic loss. We derive the 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 mean squared error 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

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 $\tilde{\mathbf{x}} = \{\tilde{x}_i\}_{i=1}^{M}$ randomly, $\tilde{\eta} \leftarrow \tilde{\eta}_0$
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 $\theta_0^{(j)} \sim p(\theta_0)$ 也可以使用fixed
5: **for each** sampled $\theta_0^{(j)}$ **do** 取多个随机的初始权重 为了减少权重初始化的影响
6: Compute updated parameter with GD: $\theta_1^{(j)} = \theta_0^{(j)} - \tilde{\eta}\nabla_{\theta_0^{(j)}}\ell(\tilde{\mathbf{x}}, \theta_0^{(j)})$ 利用抽象数据训练权重
7: Evaluate the objective function on real training data: $\mathcal{L}^{(j)} = \ell(\mathbf{x}_t, \theta_1^{(j)})$ 利用新权重和真实数据算loss
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 $\tilde{\eta}$

matrix after one GD step with these distilled data is

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

For the quadratic loss, there always exists learned 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 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, 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 focus on $p(\theta_0)$ distributions that yield similar local conditions. Section 3.5 explores several practical choices. To address the limitation of using a single GD step, we extend our method to multiple GD steps in the next section.

3.4 Multiple Gradient Descent Steps and Multiple Epochs

We extend Algorithm 1 to more than one gradient descent steps by changing Line 6 to multiple sequential GD steps each on a different batch of distilled data and learning rates, i.e., each step i is

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

and changing Line 9 to backpropagate through all steps. However, naively computing gradients is memory and computationally expensive. Therefore, we exploit a recent technique called back-gradient optimization, which allows for significantly faster gradient calculation of such updates in reverse-mode differentiation. 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). For further algorithmic details, we refer readers to prior work (Domke, 2012; Maclaurin et al., 2015).

Multiple epochs. To further improve the performance, we train the network for multiple epochs (passes) over the same sequence of distilled data. In other words, for each epoch, our method cycles

through all GD steps, where each step is associated with a batch of distilled data. We do not tie the trained learning rates across epochs as later epochs often use smaller learning rates. In Section 4.1, we find that using multiple steps and multiple epochs is more effective than using just one on neural networks, with the total amount of distilled data fixed.

3.5 Distillation with Different Initializations

Inspired by the analysis of the simple linear case in Section 3.3, we aim to focus on initial weights distributions $p(\theta)$ that yield similar local conditions over the data distribution. In this work, we focus on the following four practical choices:

- **Random initialization:** Distribution over random initial weights, e.g., He Initialization (He et al., 2015) and Xavier Initialization (Glorot & 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 today (Torralba & Efros, 2011). Extensive prior work has been proposed to adapt models to new tasks and datasets (Daume III, 2007; Saenko et al., 2010). In this work, we characterize the domain mismatch via distilled data. In Section 4.2, we show that a small number of distilled images are sufficient to quickly adapt CNN models to new datasets and tasks.

3.6 Distillation with Different Objectives

Distilled data learned with different learning objectives can train models to exhibit different desired behaviors. We have already mentioned image classification as one of the applications, where distilled images help to train accurate classifiers. Below, we introduce a different learning objective to further demonstrate the flexibility of our method.

Distillation for malicious data poisoning. For example, our approach can be used to construct a new form of data poisoning attack. To illustrate this idea, we consider the following scenario. When a single GD step is applied with our synthetic adversarial data, a well-behaved image classifier catastrophically forgets one category but still maintains high accuracy on other categories.

Formally, given an attacked category K and a target category T, we minimize a new objective $\ell_{K\to T}(\mathbf{x}, \theta_1)$, which is a classification loss that encourages θ_1 to misclassify images of category K as category T while correctly predicting other images, e.g., a cross entropy loss with target labels of K modified to T. Then, we can obtain the malicious distilled images by optimizing

$$\tilde{\mathbf{x}}^*, \tilde{\eta}^* = \arg\min_{\tilde{\mathbf{x}}, \tilde{\eta}} \mathbb{E}_{\theta_0 \sim p(\theta_0)} \mathcal{L}_{K\to T}(\tilde{\mathbf{x}}, \tilde{\eta}; \theta_0), \tag{10}$$

where $p(\theta_0)$ is the distribution over *random weights* of well-optimized classifiers. Trained on a distribution of such classifiers, the distilled images *do not* require access to the exact model weights and thus can generalize to unseen models. In our experiments, the malicious distilled images are trained on 2000 well-optimized models and evaluated on 200 held-out ones.

Compared to prior data poisoning attacks (Biggio et al., 2012; Li et al., 2016; Muñoz-González et al., 2017; Koh & Liang, 2017), our approach crucially *does not* require the poisoned training data to be stored and trained on repeatedly. Instead, our method attacks the model training in one iteration and with only a few data. This advantage makes our method potentially effective for online training algorithms and useful for the case where malicious users hijack the data feeding pipeline for only one gradient step (e.g., one network transmission). In Section 4.2, we show that a single batch of distilled data applied in one step can successfully attack well-optimized neural network models. This setting can be viewed as distilling the knowledge of a specific category into data.

4 EXPERIMENTS

We report image classification results on MNIST (LeCun, 1998) and CIFAR10 (Krizhevsky & Hinton, 2009). For MNIST, the distilled images are trained with LENET (LeCun et al., 1998), which achieves about 99% test accuracy if fully trained. For CIFAR10, we use a network architecture (Krizhevsky, 2012) that achieves around 80% test accuracy if fully trained. For random initializations and random pre-trained weights, we report means and standard deviations over 200 held-out models, unless otherwise specified. The code and full results can be found at our website.

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 clustering to each category, and use the cluster centroids as training images.
- **Average real images:** We compute the average image for each category, which is reused in different GD steps.

For these baselines, we perform each evaluation on 200 held-out models with all combinations of learning rate $\in \{\text{learned learning rate with our method}, 0.001, 0.003, 0.01, 0.03, 0.1, 0.3\}$ and #epochs $\in \{1, 3, 5\}$, and report results using the best performing combination. Please see the appendix Section S-1 for more details about training and baselines.

4.1 DATASET DISTILLATION

Fixed initialization. With access to initial network weights, distilled images can directly train a fixed network to reach high performance. For example, 10 distilled images can boost the performance of a neural network with an initial accuracy 12.90% to a final accuracy 93.76% on MNIST (Figure 2a). Similarly, 100 images can train a network with an initial accuracy 8.82% to 54.03% test accuracy on CIFAR10 (Figure 2b). This result suggests that even only a few distilled images have enough capacity to distill part of the dataset.

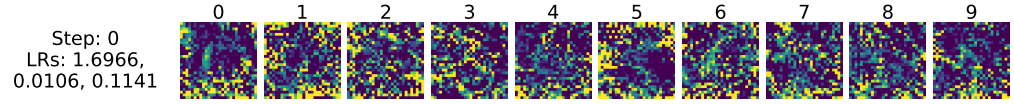
Random initialization. Trained with randomly sampled initializations using Xavier Initialization (Glorot & Bengio, 2010), the learned distilled images do not need to encode information tailored for a particular starting point and thus can represent meaningful content independent of network initializations. In Figure 3, we see that such distilled images reveal the discriminative features of corresponding categories: e.g., the `ship` image in Figure 3b. These 100 images can train randomly initialized networks to 36.79% average test accuracy on CIFAR10. Similarly, for MNIST, the 100 distilled images shown in Figure 3a can train randomly initialized networks to 79.50% test accuracy.

Multiple gradient descent steps and multiple epochs. In Figure 3, distilled images are learned for 10 GD steps applied in 3 epochs, leading to a total of 100 images (with each step containing one image per category). Images used in early steps tend to look noisier. However, in later steps, the distilled images gradually look like real data and share the discriminative features for these categories. Figure 4a shows that using more steps significantly improves the results. Figure 4b shows a similar but slower trend as the number of epochs increases. We observe that longer training (i.e., more epochs) can help the model learn all the knowledge from the distilled images, but the performance is eventually limited by the total number of images. Alternatively, we can train the model with one GD step but a big batch size. Section 3.3 has shown theoretical limitations of using only one step in a simple linear case. In Figure 5, we observe that using multiple steps drastically outperforms the single step method, given the same number of distilled images.

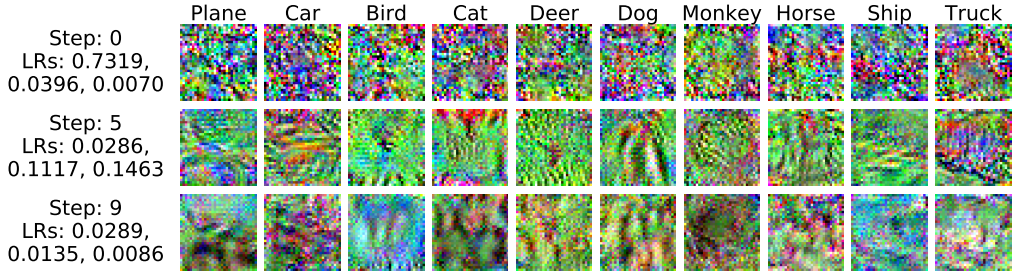
Table 1 summarizes the results of our method and all baselines. Our method with both fixed and random initializations outperforms all the baselines on CIFAR10 and most of the baselines on MNIST.

4.2 DISTILLATION WITH DIFFERENT INITIALIZATIONS AND OBJECTIVES

Next, we show two extended settings of our main algorithm discussed in Section 3.5 and Section 3.6. Both cases assume that the initial weights are random but pre-trained on the same dataset. We train the distilled images on 2000 random pre-trained models and evaluate them on *unseen* models.

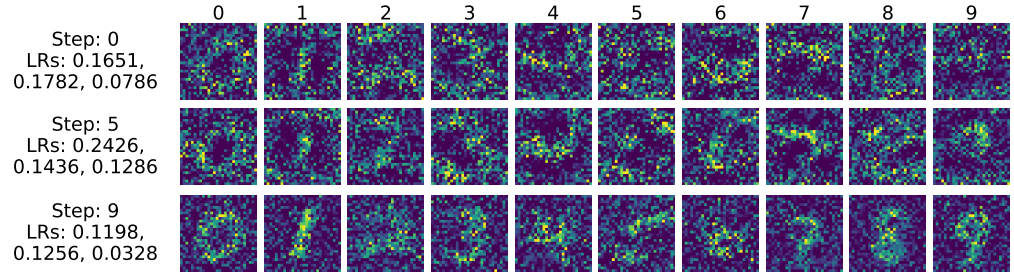


(a) MNIST. Theses distilled images train a fixed initializations from 12.90% test accuracy to 93.76%.

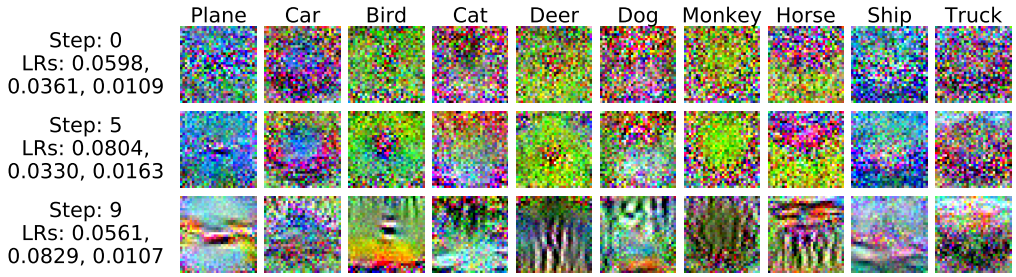


(b) CIFAR10. These distilled images train a fixed initialization from 8.82% test accuracy to 54.03%.

Figure 2: Dataset distillation for fixed initializations: MNIST distilled images use one GD step and three epochs (10 images in total). CIFAR10 distilled images use ten GD steps and three epochs (100 images in total). For CIFAR10, only selected steps are shown. At left, we report the corresponding learning rates for all three epochs.



(a) MNIST. These distilled images unknown random initializations to $79.50\% \pm 8.08\%$ test accuracy.

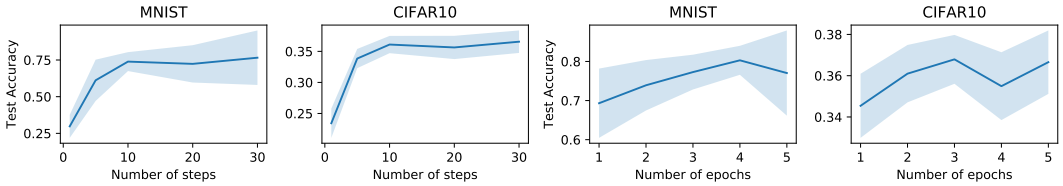


(b) CIFAR10. These distilled images unknown random initializations to $36.79\% \pm 1.18\%$ test accuracy.

Figure 3: Distilled images trained for *random initialization* with ten GD steps and three epochs (100 images in total). We show images from selected GD steps and the corresponding learning rates for all three epochs.

	Ours		Baselines					
			Used as training data in same number of GD steps				Used in K-NN classification	
	Fixed init.	Random init.	Random real	Optimized real	k-means	Average real	Random real	k-means
MNIST	**96.6**	79.5 ± 8.1	68.6 ± 9.8	73.0 ± 7.6	76.4 ± 9.5	77.1 ± 2.7	71.5 ± 2.1	$\mathbf{92.2 \pm 0.1}$
CIFAR10	**54.0**	$\mathbf{36.8 \pm 1.2}$	21.3 ± 1.5	23.4 ± 1.3	22.5 ± 3.1	22.3 ± 0.7	18.8 ± 1.3	29.4 ± 0.3

Table 1: Comparison between our method trained for ten GD steps and three epochs and various baselines. For baselines using K-Nearest Neighbor (K-NN), best result among all combinations of distance metric $\in \{l_1, l_2\}$ and $\mathtt{K} \in \{1, 3\}$ is reported. In K-NN and k-means, K and k can have different values. 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.



(a) Test accuracy w.r.t. the number of steps (b) Test accuracy w.r.t. the number of epochs

Figure 4: Hyper-parameter sensitivity studies: we evaluate models in *random initialization* settings. (a) Average test accuracy w.r.t. the number of GD steps. We use two epochs. (b) Average test accuracy w.r.t. the number of epochs. We use 10 steps, with each step containing 10 images.

(a) Test accuracy w.r.t. the number of steps (b) Test accuracy w.r.t. the number of epochs

Figure 5: Comparison between one and multiple GD steps given the same number of images. We evaluate models in *random initialization* settings and use one epoch. N denotes the total number of images per category. For multiple steps runs, each step uses 10 images in total (one per category).

	Ours w/ fixed pre-trained	Ours w/ random pre-trained	Random real	Optimized real	k-means	Average real	Adaptation Motiian et al. (2017)	No adaptation	Train on **full** target dataset
$\mathcal{M} \rightarrow \mathcal{U}$	**97.9**	$\mathbf{95.4 \pm 1.8}$	94.9 ± 0.8	95.2 ± 0.7	92.2 ± 1.6	93.9 ± 0.8	$\mathbf{96.7 \pm 0.5}$	90.4 ± 3.0	97.3 ± 0.3
$\mathcal{U} \rightarrow \mathcal{M}$	**93.2**	$\mathbf{92.7 \pm 1.4}$	87.1 ± 2.9	87.6 ± 2.1	85.6 ± 3.1	78.4 ± 5.0	89.2 ± 2.4	67.5 ± 3.9	98.6 ± 0.5
$\mathcal{S} \rightarrow \mathcal{M}$	**96.2**	85.2 ± 4.7	84.6 ± 2.1	85.2 ± 1.2	$\mathbf{85.8 \pm 1.2}$	74.9 ± 2.6	74.0 ± 1.5	51.6 ± 2.8	98.6 ± 0.5

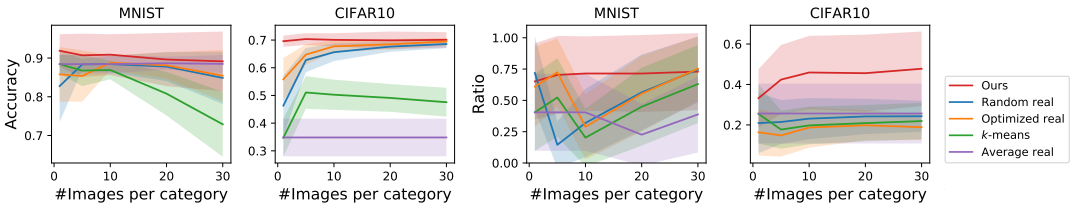
Table 2: Performance of our method and baselines in adapting models among MNIST ($\mathcal{M}$), USPS ($\mathcal{U}$), and SVHN ($\mathcal{S}$). 100 distilled images are trained for ten GD steps and three epochs. Our method outperforms few-shot domain adaptation (Motiian et al., 2017) and other baselines in most settings.

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

Table 3: Performance of our method and baselines in adapting an AlexNet pre-trained on ImageNet to PASCAL-VOC and CUB-200. We use one distilled image per category, with one GD step repeated for three epochs. Our method significantly outperforms the baselines. We report results over 10 runs.

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 for 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 weights on ImageNet. In Table 3, we adapt a widely used AlexNet model (Krizhevsky et al., 2012) pre-trained on ImageNet (Deng et al., 2009) to image classification on PASCAL-VOC (Everingham et al., 2010) and CUB-200 (Wah et al., 2011) datasets. Using



(a) Overall accuracy w.r.t. modified labels (b) Ratio of attacked category misclassified as target

Figure 6: Performance for our method and baselines with *random pre-trained weights* and *a malicious objective*. All the methods are trained for one GD step. (a) Our method slightly outperforms the best baseline in accuracy w.r.t. modified labels. (b) Our method performs similarly with some baselines in changing the prediction of the attacked category on MNIST, but is significantly better than all baselines on CIFAR10.

only one distilled image per category, our method outperforms baselines significantly. Our method is on par with fine-tuning on the full datasets with thousands of images.

Random pre-trained weights and a malicious data-poisoning objective. Section 3.6 shows that our method can construct a new type of data poisoning, where an attacker can apply just one GD step with a few malicious data to manipulate a well-trained model. We train distilled images to make well-optimized neural networks to misclassify an attacked category as another target category *within only one GD step*. Our method requires *no* access to the exact weights of the model. In Figure 6b, we evaluate our method on 200 held-out models, against various baselines using data derived from real images and incorrect labels. For baselines, we apply one GD step using the same numbers of images with modified labels (i.e., the attacked category images are labeled as target category) and report the highest overall accuracy w.r.t. the modified labels while misclassifying $\geq 10\%$ attacked category as target category. This avoids results with learning rates too low to change model behavior at all. While some baselines perform similarly well as our method on MNIST, our method significantly outperforms all the baselines on CIFAR10.

5 Discussion

In this paper, we have presented dataset distillation for compressing the knowledge of entire training data into a few synthetic training images. We can train a network to reach high performance with a small number of distilled images and several gradient descent steps. Finally, we demonstrate two extended settings including adapting pre-trained models to new datasets and performing a malicious data-poisoning attack. In the future, we plan to extend our method to compressing large-scale visual datasets such as ImageNet and other types of data (e.g., audio and text). Also, our current method is sensitive to the distribution of initializations. We would like to investigate other initialization strategies, with which dataset distillation can work well.

Acknowledgments This work was supported in part by NSF 1524817 on Advancing Visual Recognition with Feature Visualizations, NSF IIS-1633310, and Berkeley Deep Drive.

REFERENCES

Anelia Angelova, Yaser Abu-Mostafam, and Pietro Perona. Pruning training sets for learning of object categories. In *CVPR*, 2005. 3

Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *NIPS*, 2014. 2

Olivier Bachem, Mario Lucic, and Andreas Krause. Practical coreset constructions for machine learning. *arXiv preprint arXiv:1703.06476*, 2017. 3

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *CVPR*, 2017. 3

Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900, 2000. 3

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *ICML*, 2012. 6

David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996. 3

Hal Daume III. Frustratingly easy domain adaptation. In *ACL*, 2007. 6

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 6, 9

Justin Domke. Generic methods for optimization-based modeling. In *Artificial Intelligence and Statistics*, pp. 318–326, 2012. 3, 5

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010. 9

Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32(9):1627–1645, 2010. 3

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, 2010. 6, 7

Sally A Goldman and Michael J Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995. 3

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 6

Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998. 3

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. 1, 2

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. In *CVPR*, 2017. 2

Jonathan J. Hull. A database for handwritten text recognition research. *PAMI*, 16(5):550–554, 1994. 9

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 14

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *ICML*, 2017. 3, 6

Alex Krizhevsky. cuda-convnet: High-performance c++/cuda implementation of convolutional neural networks. *https://github.com/akrizhevsky/cuda-convnet2*, 2012. 7

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 7

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 6, 9

Agata Lapedriza, Hamed Pirsiavash, Zoya Bylinskii, and Antonio Torralba. Are all training examples equally valuable? *arXiv preprint arXiv:1311.6510*, 2013. 3

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998. 7

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. 1, 7

Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. In *NIPS*, 2016. 6

Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*, 2017. 3

Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *ICML*, 2015. 3, 5

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, 2015. 3

Saeid Motiian, Quinn Jones, Seyed Iranmanesh, and Gianfranco Doretto. Few-shot adversarial domain adaptation. In *NIPS*, 2017. 9

Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 27–38, 2017. 6

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop*, 2011. 9

J Arturo Olvera-López, J Ariel Carrasco-Ochoa, J Francisco Martínez-Trinidad, and Josef Kittler. A review of instance selection methods. *Artificial Intelligence Review*, 34(2):133–143, 2010. 3

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *ICLR Workshop*, 2017. 5

Barak A Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994. 5

Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *ICML*, 2016. 3

Jean Ponce, Tamara L Berg, Mark Everingham, David A Forsyth, Martial Hebert, Svetlana Lazebnik, Marcin Marszalek, Cordelia Schmid, Bryan C Russell, Antonio Torralba, et al. Dataset issues in object recognition. In *Toward category-level object recognition*, pp. 29–48. 2006. 3

Ilija Radosavovic, Piotr Dollár, Ross Girshick, Georgia Gkioxari, and Kaiming He. Data distillation: Towards omni-supervised learning. In *CVPR*, 2018. 2

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015. 2

Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957. 3

Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *ECCV*, 2010. 6

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *ICLR*, 2018. 3

Ayumi Shinohara and Satoru Miyano. Teachability in computational learning. *New Generation Computing*, 8(4): 337–347, 1991. 3

Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *JMLR*, 2(Nov):45–66, 2001. 3

Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR*, pp. 1521–1528. IEEE, 2011. 3, 6

Ivor W Tsang, James T Kwok, and Pak-Ming Cheung. Core vector machines: Fast svm training on very large data sets. *JMLR*, 6(Apr):363–392, 2005. 3

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. 9

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 3

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. In *ICLR*, 2015. 3

Xiaojin Zhu. Machine teaching for bayesian learners in the exponential family. In *NIPS*, 2013. 3

Xiaojin Zhu. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *AAAI*, 2015. 3

S-1 SUPPLEMENTARY MATERIAL

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 & 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 Titan Xp and V100 GPUs. We use one GPU for fixed initial weights and four GPUs for random initial weights. Each training typically takes 1 to 4 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 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 reused in different GD steps. 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.