Soft-Label Dataset Distillation and Text Dataset Distillation

Ilia Sucholutsky
Statistics and Actuarial Science
University of Waterloo
Waterloo, Canada
isucholu@uwaterloo.ca

Matthias Schonlau
Statistics and Actuarial Science
University of Waterloo
Waterloo, Canada
schonlau@uwaterloo.ca

2021 International Joint Conference on Neural Networks (IJCNN) | 978-1-6654-3900-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/IJCNN52387.2021.9533769

***Abstract*—Dataset distillation is a method for reducing dataset sizes by learning a small number of representative synthetic samples. This has several benefits such as speeding up model training, reducing energy consumption, and reducing required storage space. These benefits are especially crucial in settings like federated learning where initial overhead costs are justified by the speedup they enable. Currently, 1) each synthetic sample is assigned a single 'hard' label, and 2) dataset distillation can only be used with image data. We propose to simultaneously distill both images and their labels, thus assigning each synthetic sample a 'soft' label (a distribution of labels). Our algorithm increases accuracy by 2-4% for several image classification tasks. Using 'soft' labels also enables distilled datasets to consist of fewer samples than there are classes as each sample encodes information for multiple classes. For example, training a LeNet model with 10 distilled images (one per class) results in over 96% accuracy on MNIST, and almost 92% accuracy when trained on just 5 distilled images. We also extend the dataset distillation algorithm to distill text data. We demonstrate that text distillation outperforms other methods across multiple datasets. For example, models attain almost their original accuracy on the IMDB sentiment analysis task using just 20 distilled sentences. Our code can be found at https://github.com/ilia10000/dataset-distillation**

I. Introduction

Deep learning is computationally intensive and this poses several issues: high energy consumption [1], high financial cost, and long training times. This is particularly problematic for settings like federated learning where edge devices are computationally constrained and may not be able to work with large datasets [2]. One path for mitigating these issues is to reduce network sizes. [3] proposed knowledge distillation as a method for imbuing smaller, more efficient networks with all the knowledge of their larger counterparts. Instead of decreasing network size, a second path to efficiency may be to decrease dataset size. Dataset distillation (DD) has recently been proposed as an alternative formulation of knowledge distillation to do exactly that [4].

Dataset distillation is the process of creating a small number of synthetic samples that can quickly train a network to the same accuracy it would achieve if trained on the original dataset. It may seem counter-intuitive that training a model on a small number of synthetic images coming from a different distribution than the training data can result in comparable accuracy, but [4] have shown that for models with known initializations this is indeed feasible; they achieve

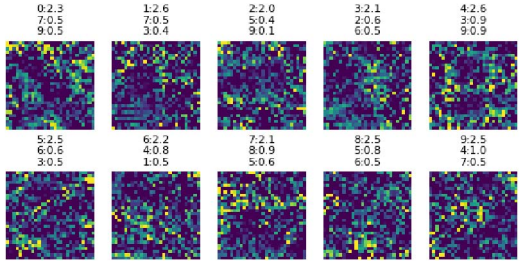

Fig. 1: 10 synthetic MNIST images learned by SLDD can train networks with fixed initializations from 11.13% accuracy to 96.13% ($r_{10} = 97.1$). Each image is labeled with its top 3 classes and their associated logits.

$$\begin{bmatrix}0\\1\\0\\0\\0\\0\\0\\0\\0\\0\end{bmatrix} \xleftarrow{\text{Set largest to 1 and rest to 0}} \begin{bmatrix}0.01\\0.69\\0.02\\0.02\\0.03\\0.05\\0.03\\0.1\\0.01\\0.04\end{bmatrix} \xleftarrow{\text{Apply softmax}} \begin{bmatrix}0.8\\5.1\\1.5\\1.5\\2\\2.5\\2\\3.2\\0.8\\2\end{bmatrix}$$

Fig. 2: **Left**: An example of a 'hard' label where the second class is selected. **Center**: An example of a 'soft' label restricted to being a valid probability distribution. The second class has the highest probability. **Right**: An example of an unrestricted 'soft' label. The second class has the highest weight. 'Hard' labels can be derived from unrestricted 'soft' labels by applying the softmax function and then setting the highest probability element to 1, and the rest to 0.

94% accuracy on MNIST, a hand-written digit recognition task [5], after training LeNet on just 10 synthetic images. We propose to improve their already impressive results by learning 'soft' labels as a part of the distillation process. The original DD algorithm uses fixed, or 'hard', labels for the synthetic samples (e.g. the ten synthetic MNIST images each have a

978-0-7381-3366-9/21/$31.00 ©2021 IEEE

label corresponding to a different digit). In other words, each label is a one-hot vector: a vector where all entries are set to zero aside from a single entry, corresponding to the correct class, which is set to one. We relax this one-hot restriction and make the distribution learnable for these synthetic labels. The resulting distilled labels are thus similar to those used for knowledge distillation as a single image can now correspond to multiple classes. An example comparing a 'hard' label to a 'soft' label is shown in Figure 2. A 'hard' label can be derived from a 'soft' label by applying the softmax function and setting the element with the highest probability to one, while the remaining elements are set to zero. Our soft-label dataset distillation (SLDD) not only achieves over 96% accuracy on MNIST when using ten distilled images (as seen in Figure 1), a 2% increase over the state-of-the-art (SOTA), but also achieves almost 92% accuracy with just five distilled images, which is less than one image per class. In addition to soft labels, we also extend dataset distillation to the natural language/sequence modeling domain and enable it to be used with several additional network architectures. For example, we show that Text Dataset Distillation (TDD) can train a custom convolutional neural network (CNN) [6] with known initialization up to 90% of its original accuracy on the IMDB sentiment classification task [7] using just two synthetic sentences.

The remainder of this paper is divided into four sections. In the second section, we discuss related work in knowledge distillation and dataset reduction. In the third section, we propose improvements and extensions to dataset distillation and associated theory. In the fourth section, we empirically validate SLDD and TDD in a wide range of experiments. In the last section, we discuss the significance of SLDD and TDD, and our outlook for the future.

II. Related Work

A. Knowledge Distillation

Dataset distillation was originally inspired by network distillation [3] which is a form of knowledge distillation or model compression [8]. Network distillation has been studied in various contexts including when working with sequential data [9]. Network distillation aims to distill the knowledge of large, or even multiple, networks into a smaller network. Similarly, dataset distillation aims to distill the knowledge of large, or even multiple, datasets into a small number of synthetic samples. 'Soft' labels were recently proposed as an effective way of distilling networks by feeding the output probabilities of a larger network directly to a smaller network [3], and have previously been studied in the context of different machine learning algorithms [10]. Our SLDD algorithm also uses 'soft' labels but these are persistent and learned over the training phase of a network rather than being produced during the inference phase as in the case of network distillation.

B. Dataset Reduction and Prototype Methods

There are many methods that aim to reduce the size of a dataset with varying objectives. Active learning aims to reduce the required size of the labeled portion of a dataset by only labeling examples that are determined to be the most important [11], [12]. Several methods aim to 'prune' a dataset, or create a 'core-set', by leaving in only examples that are determined to be useful [13]–[16]. In the context of nearest-neighbor classification, prototype selection (PS) and prototype generation (PG) are studied extensively as methods of reducing storage requirements and improving the efficiency of nearest-neighbor classification [17], [18]. In general, all of these methods aside from PG, use samples from the true distribution, typically subsets of the original training set. By lifting this restriction and, instead, learning synthetic samples, DD requires far fewer samples to distill the same amount of knowledge. PG methods typically create samples that are not found in the training data; however, these methods are designed specifically for use with nearest-neighbor classification algorithms. In addition, most of the methods listed above use fixed labels. SLDD removes this restriction and allows the label distribution to be optimized simultaneously with the samples (or prototypes) themselves.

C. Sample Efficiency in Deep Learning

Deep learning models are notoriously data-hungry. In their landmark paper introducing GPT-3, [19] write that a 'limitation broadly shared by language models is poor sample efficiency during pre-training'. Sample efficiency is also a widely discussed problem in reinforcement learning [20]–[22]. Few-shot learning aims to improve deep learning sample efficiency to the point where models can learn a new class given only a few examples [23]–[25]. Dataset distillation allows us to probe the limits of a given model's sample efficiency by creating synthetic samples that are tailored specifically for efficiently training this exact model.

D. Federated Learning and Privacy Preservation

DD-like algorithms are naturally well-suited for federated learning: they provide the ability to transmit small, synthetic, privacy-preserving training datasets while still maintaining high performance of the networks trained on them. Increased initial overhead is a small price to pay for smaller transmission sizes, faster training times, and privacy preservation. We initially released the code for SLDD and TDD in 2019. Since then, there has been significant work building on our research that demonstrates the applications of SLDD to federated learning [26]–[28] as well as to the transmission of private data [28], [29]. There has also been an increasing amount of literature discussing using the original DD algorithm in the context of federated learning [30]–[33].

III. Extending Dataset Distillation

A. Motivation

As mentioned above, nearest-neighbors classification often involves accuracy-preserving data reduction techniques known as prototype selection (PS) and prototype generation (PG). We can use these concepts, along with the k-Nearest Neighbors (kNN) classification algorithm, to visualize the difference

between classical dataset reduction methods, DD, and SLDD. When we fit a kNN model, we essentially divide the entire space into classes based on the location of points in the training set. However, the cost of fitting a kNN model increases with the number of training points.

PS methods use subsets of the training set to construct a reduced training set. In the first column of Figure 3, we visualize reduction of the training set for a three-class problem, the Iris flower dataset [34], by selecting one point from each class and then fitting the kNN on these three selected points. Only being able to use a subset of the original points limits the ability to finely tune the shape of the resulting partitions. PG methods create synthetic points whose placement can be optimized to improve kNN performance. DD for neural networks works analogously. In the second column of Figure 3, we generate one point for each class and optimize the location of one of them. Using synthetic points allows for better control of the resulting landscape.

In the third column of Figure 3, we propose that the points have optimizable distributions of labels. In order to visualize the effect of changing a point's label distribution, we create one point per class, but then change the label distribution of one of these points, increasingly making it a mixture of the other two classes. In the final column of Figure 3, we combine the PG method with our soft-label modification, to visualize the effect of simultaneously changing a point's location and label distribution. This last case is the kNN counterpart to our proposed SLDD algorithm. The visualization illustrates that this method provides the most control over the resulting landscape. In fact, Figure 4 shows that we can even separate three classes using just two points when using soft labels with the kNN classifier.

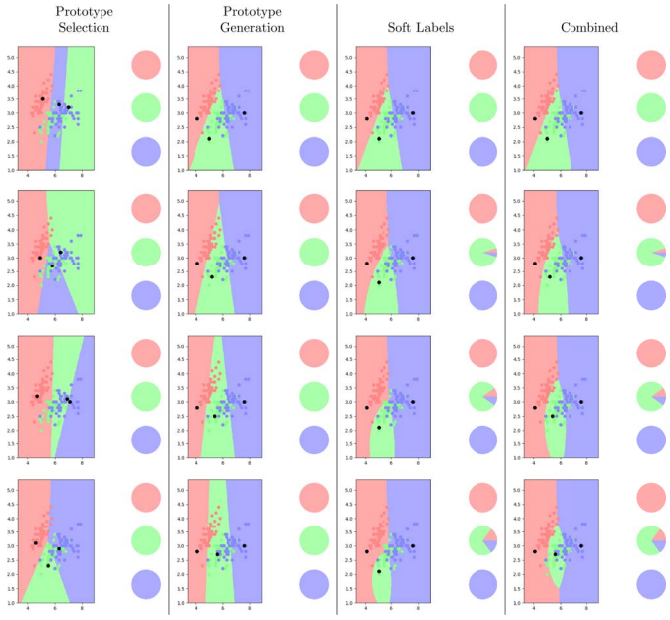

Fig. 3: kNN models are fitted on 3 points obtained using four methods: PS, PG, soft labels, and PG combined with soft labels. Each column contains 4 steps of the associated method used to update the 3 points. The pie charts represent the label distributions assigned to each of the 3 points. **PS:** A different random point from each class is chosen to represent its class in each of the steps. **PG:** The model can select and adjust synthetic points to represent each class. In this case, the middle point associated with the 'green' label is moved diagonally in each step. **Soft Labels:** The label distribution of the middle point is changed each step to contain a larger proportion of both other classes. **Combined:** The middle point is simultaneously moved and has its label distribution updated in each step.

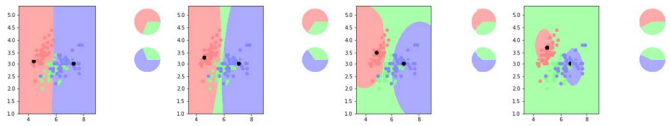

Fig. 4: A kNN model fitted on 2 points obtained using the 'Combined' method. The pie charts represent the label distributions assigned to each of the 2 points. From the left plot to the right plot, the locations of the 2 points slightly shift and the 'green' portions of their label distributions are increased. By modifying the soft labels of the 2 points, the space can still be separated into 3 classes.

B. Basic Approach

In this section, we summarize the approach by [4] in a slightly modified way to explicitly show the labels of the distilled dataset. In the next section, we expand this approach to work with soft labels.

Given a training dataset $\mathbf{d} = \{x_i, y_i\}_{i=1}^{N}$, a neural network with parameters θ, and a twice-differentiable loss function $\ell\left(x_i, y_i, \theta\right)$, our objective is to find

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \ell\left(x_i, y_i, \theta\right) \triangleq \arg\min_{\theta} \ell(\mathbf{x}, \mathbf{y}, \theta) \ . \quad (1)$$

In general, training with involves repeatedly sampling mini-batches of training data and updating network parameters by their error gradient scaled by learning rate η.

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} \ell\left(\mathbf{x}_t, \mathbf{y}_t, \theta_t\right) \quad (2)$$

With DD, the goal is to perform just one such step while still achieving the same accuracy. We do this by learning a small number of synthetic samples $\tilde{\mathbf{x}}$ that minimize $\mathcal{L}$, a one-step loss objective, for $\theta_1 = \theta_0 - \tilde{\eta} \nabla_{\theta_0} \ell\left(\tilde{\mathbf{x}}, \theta_0\right)$.

$$\mathcal{L}\left(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \tilde{\eta}; \theta_0\right) := \ell\left(\mathbf{x}, \mathbf{y}, \theta_0 - \tilde{\eta} \nabla_{\theta_0} \ell\left(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \theta_0\right)\right) \quad (3)$$

$$\tilde{\mathbf{x}}^*, \tilde{\eta}^* = \arg\min_{\tilde{\mathbf{x}}, \tilde{\eta}} \mathcal{L}\left(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \tilde{\eta}; \theta_0\right) \quad (4)$$

We minimize this objective, or 'learn the distilled samples', by using SGD. Here we are optimizing over $\tilde{\mathbf{x}}$ and $\tilde{\eta}$, but not $\tilde{\mathbf{y}}$, as the labels are fixed for the original DD algorithm.

C. Learnable Labels

Soft labels exploit the fact that each training image contains information about more than one class (e.g. the digit '3' looks

a lot like a '3' but also like an '8'). Using soft labels allows us to convey more information about each image.

The original dataset distillation algorithm was restricted to 'hard' labels for the distilled data; each distilled image has to be associated with just a single class. We relax this restriction and allow distilled labels to take on any real value. Since the distilled labels are now continuous variables, we can modify the distillation algorithm in order to make the distilled labels learnable using the same method as for the distilled images: a combination of backpropagation and gradient descent. With our modified notation, we simply need to change Equation 4 to also minimize over $\tilde{\mathbf{y}}$.

$$\tilde{\mathbf{x}}^*, \tilde{\mathbf{y}}^*, \tilde{\eta}^* = \arg\min_{\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \tilde{\eta}} \mathcal{L}\left(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \tilde{\eta}; \theta_0\right) \tag{5}$$

$$= \arg\min_{\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \tilde{\eta}} \ell\left(\mathbf{x}, \mathbf{y}, \theta_0 - \tilde{\eta} \nabla_{\theta_0} \ell\left(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \theta_0\right)\right) \tag{6}$$

In our experiments, we generally initialize $\tilde{y}$ with the one-hot values that 'hard' labels would have. We found that this tends to increase accuracy when compared to random initialization, perhaps because it encourages more differentiation between classes early on in the distillation process.

D. Text and Other Sequences

It is difficult to use gradient methods directly on text data as they are discrete. In order to use SLDD with text data, we first embed the text data into a continuous space using pre-trained GloVe embeddings [35]. This is a common practice when working with many modern natural language processing models, though the embedding method itself can vary greatly [36]–[38]. Distilling embedded text is analogous to image distillation. If all sentences are padded/truncated to some pre-determined length, then each sentence can be viewed as a one-channel image of size [length]$*$[embedding dimension], though this is not required when working with RNNs. Only sentences coming from the true dataset need to be embedded; the distilled samples are learned directly as embedded representations. As a result, finding the nearest sentences that correspond to the distilled embeddings may help with interpretability. To compute the nearest sentence to a distilled embedding matrix, for every column vector in the matrix, the nearest embedding vector from the original dictionary must be found. These embedding vectors must then be converted back into their corresponding words, and those words joined into a sentence. To differentiate this modified procedure from SLDD, we refer to it as Text Dataset Distillation (TDD).

E. Random initializations and multiple steps

The procedures we described above make one important assumption: network initialization θ_0 is fixed. The samples created this way do not lead to high accuracies when the network is re-trained on them with a different initialization as they contain information not only about the dataset but also about θ_0. In Figures 1 and 5, this can be seen as what looks like a lot of random noise. [4] propose a generalization that works with network initializations randomly sampled from some restricted distribution.

$$\tilde{\mathbf{x}}^*, \tilde{\mathbf{y}}^*, \tilde{\eta}^* = \arg\min_{\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \tilde{\eta}} \mathbb{E}_{\theta_0 \sim p(\theta_0)} \mathcal{L}\left(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \tilde{\eta}; \theta_0\right) \tag{7}$$

The resulting images, especially for MNIST, appear to have clearer patterns and less random noise. Our experimental results suggest that this method generalizes fairly well to other randomly sampled initializations from the same distribution.

Additionally, [4] suggest that the above methods can work with multiple gradient descent (GD) steps and with multiple epochs over the distilled data. The experimental results suggest that multiple steps and multiple epochs improve distillation performance for both image and text data, particularly when using random network initializations.

IV. Experiments

A. Metrics

The simplest metric for gauging distillation performance is to train a model on distilled samples and then test it on real samples. We refer to the accuracy achieved on real samples as the 'distillation accuracy'. However, several of the models we use in our experiments do not achieve SOTA accuracy on the datasets they are paired with, so it is useful to construct a relative metric that compares distillation accuracy to original accuracy. We define 'distillation ratio' as the ratio of distillation accuracy to original accuracy. The distillation ratio is heavily dependent on the number of distilled samples so the notation we use is $r_M = 100\% * \frac{\text{[distillation accuracy]}}{\text{[original accuracy]}}, M = \text{[number of distilled samples]}$. We may refer to this as the 'M-sample distillation ratio' when clarification is needed. It may also be of interest to find the minimum number of distilled images required to achieve a certain distillation ratio. We call this the '$A\%$ distillation size', and we write $d_A = M$ where M is the minimum number of distilled samples required to achieve a distillation ratio of $A\%$.

B. Image Data

While LeNet achieves 99% accuracy on MNIST, AlexCifarNet only achieves 80% on CIFAR10 so it is helpful to use the relative metrics when describing this set of results.

Baselines. We use the same baselines as [4].

Random real images: We randomly sample the same number of real images per class from the training data. These images are used for two baselines: training neural networks and training K-Nearest Neighbors classifiers.

Optimized real images: We sample several sets of random real images as above, but now we choose the 20% of these sets that have the best performance on training data. These images are used for one baseline: training neural networks.

k-means: We use k-means to find centroid images for each class. These images are used for two baselines: training neural networks and training K-Nearest Neighbors classifiers.

Average real images: We compute the average image for each class and use it for training. These images are used for one baseline: training neural networks.

	SLDD accuracy		DD accuracy		Used as training data in same # of GD steps				Used in K-NN	
	Fix	Rand	Fix	Rand	Rand real	Optim real	k-means	Avg real	Rand real	k-means
MNIST	**98.6**	82.7 $\pm$ 2.8	96.6	79.5 $\pm$ 8.1	68.6 $\pm$ 9.8	73.0 $\pm$ 7.6	76.4 $\pm$ 9.5	77.1 $\pm$ 2.7	71.5 $\pm$ 2.1	**92.2 $\pm$ 0.1**
CIFAR	**60.0**	**39.8 $\pm$ 0.8**	54.0	36.8 $\pm$ 1.2	21.3 $\pm$ 1.5	23.4 $\pm$ 1.3	22.5 $\pm$ 3.1	22.3 $\pm$ 0.3	18.8 $\pm$ 1.3	29.4 $\pm$ 0.3

TABLE I: Means and standard deviations of SLDD, DD, and baseline accuracies (as detailed in [4]) on MNIST and CIFAR10 datasets. All values are percentages. The first four baselines produce reduced datasets that are used to train the same neural network as in the distillation experiments. The last two baselines produce reduced datasets that are used to train a K-Nearest Neighbors classifier. Experiments with random initializations have their results listed in the form [mean $\pm$ standard deviation] and are based on the resulting performance of 200 randomly initialized networks.

Each of these baseline methods produces a small set of images that can be used to train models. All four baseline methods are used to train and test LeNet and AlexCifarNet on their respective datasets. Two of the baseline methods are used to also train K-Nearest Neighbor classifiers to compare performance against neural networks. The results for these six baselines, as determined by [4], are shown in Table I.

Fixed initialization. When the network initialization is fixed between the distillation and training phases, synthetic images produced by dataset distillation result in high distillation accuracies. The SLDD algorithm produces images that result in equal or higher accuracies when compared to the original DD algorithm. For example, DD can produce 10 distilled images that train a LeNet model up to 93.76% accuracy on MNIST [4]. Meanwhile, SLDD can produce 10 distilled images that train the same model up to 96.13% accuracy (Figure 1). SLDD can even produce a tiny set of just 5 distilled images that train LeNet to 91.56% accuracy. As can be seen in Figure 6, the 90% distillation size (i.e. the minimum number of images needed to achieve 90% of the original accuracy) of MNIST with fixed initializations is $d_A = 5$, and while adding more distilled images typically increases distillation accuracy, this begins to plateau after five images. Similarly, SLDD provides a 7.5% increase in 100-sample distillation ratio (6% increase in distillation accuracy) on CIFAR10 over DD. Based on these results detailed in Table I, it appears that SLDD is more effective than DD at distilling image data into a small number of samples.

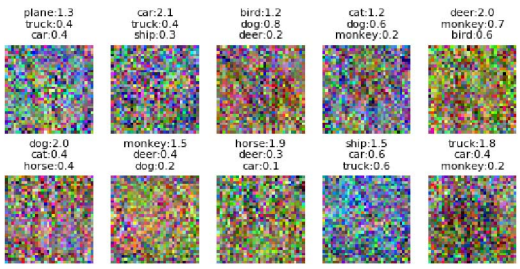

Fig. 5: SLDD can learn 100 distilled CIFAR10 images (10 steps with 10 images each) that train networks with fixed initializations from 12.9% distillation accuracy to 60.0% ($r_{100} = 75.0$). Each image is labeled with its top 3 classes and their logits. Only the last step is shown.

Random initialization. It is also of interest to know whether distilled data are robust to network initialization. Specifically, we aim to identify if distilled samples store information only about the network initializations, or whether they can store information contained within the training data. To this end, we perform experiments by sampling random network initializations generated using the Xavier Initialization [39]. The distilled images produced in this way are more representative of the training data but generally result in lower accuracies when models are trained on them. On both datasets, distilled images produced by SLDD lead to higher distillation accuracies than those from DD when the number of distilled images is held constant. These results are detailed in Table I. It is also interesting to note that the actual distilled images, as seen in Figures 7 and 8, appear to have clearer patterns emerging than in the fixed initialization case. These results suggest that DD, and even more so SLDD, can be generalized to work with random initializations and distill knowledge about the dataset itself when they are trained this way.

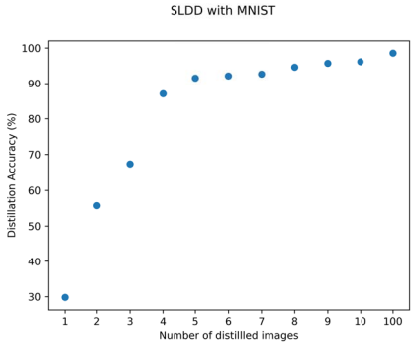

Fig. 6: Distilled dataset size and MNIST accuracy

C. Text Data

We use the IMDB sentiment dataset, the Stanford Sentiment Treebank 5-class task (SST5) [40], and the Text Retrieval Conference question classification tasks with 6 (TREC6) and 50 (TREC50) classes [41]. Text experiments are performed with three architectures: a shallow but wide CNN (TextConvNet), a bi-directional RNN (Bi-RNN) [42], and a bi-directional Long Short-Term Memory network (Bi-LSTM) [43]. The accuracies on full datasets are detailed in Table III and distillation ratios in this section are calculated based on them.

Baselines. We consider the same six baselines as for image

	TDD accuracy		Used as training data in 10 GD steps				Used in K-NN	
	Fixed	Random	Rand. real	Optim. real	k-means	Avg. real	Rand. real	k-means
IMDB	**75.0**	**73.4 ± 3.3**	49.7 ± 0.9	49.9 ± 0.8	49.9 ± 0.6	50.0 ± 0.1	50.0 ± 0.1	50.0 ± 0.0
SST5	**37.5**	**36.3 ± 1.5**	21.2 ± 4.9	24.6 ± 2.6	19.6 ± 4.5	21.3 ± 4.1	23.1 ± 0.0	20.9 ± 2.1
TREC6	**79.2**	**77.3 ± 2.9**	37.5 ± 10.1	44.6 ± 7.5	34.4 ± 13.0	28.0 ± 9.5	31.5 ± 9.9	50.5 ± 6.8
TREC50	67.4	42.1 ± 2.1	8.2 ± 6.0	9.9 ± 6.6	14.7 ± 5.5	12.5 ± 6.4	15.4 ± 5.1	**45.1 ± 6.6**

TABLE II: Means and standard deviations of TDD and baseline accuracies on text data using TextConvNet. All values are percentages. The first four baselines are used to train the same neural network as in the distillation experiments. The last two baselines are used to train a K-Nearest Neighbors classifier. Each result uses 10 GD steps aside from IMDB with k-means (2 GD steps) and TREC50 (5 GD steps, 4 images per class) which had to be done with fewer steps due to GPU memory constraints and also insufficient training samples for some classes in TREC50. Experiments with random initializations have their results listed in the form [mean ± standard deviation] and are based on the resulting performance of 200 randomly initialized networks.

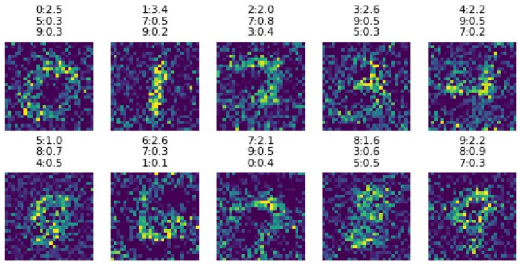

Fig. 7: SLDD can learn 100 distilled MNIST images (10 steps with 10 images each) that train networks with random initializations from $10.09\% \pm 2.54\%$ distillation accuracy to $82.75\% \pm 2.75\%$ ($r_{100} = 83.6$). Each image is labeled with its top 3 classes and their logits. Only the last step is shown.

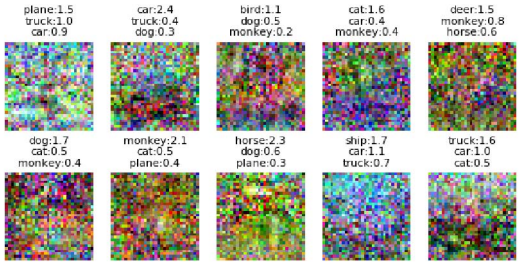

Fig. 8: SLDD can learn 100 distilled CIFAR10 images (10 steps with 10 images each) that train networks with random initializations from $10.17\% \pm 1.23\%$ distillation accuracy to $39.82\% \pm 0.83\%$ ($r_{100} = 49.8$). Each image is labeled with its top 3 classes and their logits. Only the last step is shown.

data but modify them slightly so that they work with text data (e.g. padding/truncating the sampled sentences). Each of the baseline methods produces a small set of sentences, or sentence embeddings, that can be used to train models. All four of the baseline methods are used to train and test our TextConvNet on each of the text datasets. Additionally, two of the baseline methods are used to also train K-Nearest Neighbor classifiers to compare performance against neural networks.

Model	Dataset	# of Classes	Accuracy
TextConvNet	IMDB	2	87.1%
Bi-RNN	SST5	5	41.0%
Bi-LSTM	TREC6	6	89.4%
TextConvNet	TREC50	50	84.4%

TABLE III: Model accuracies when trained on full text datasets.

			Distillation Ratio (r_M)	
Model	Dataset	M	Fixed	Random (Mean ± SD)
TextConvNet	IMDB	2	89.9	80.0 ± 6.3
TextConvNet	IMDB	20	91.5	85.2 ± 3.2
Bi-RNN	SST5	5	87.7	57.0 ± 5.7
Bi-RNN	SST5	100	89.8	66.8 ± 5.4
Bi-LSTM	TREC6	6	97.8	69.3 ± 9.8
Bi-LSTM	TREC6	120	98.2	78.9 ±6.3
TextConvNet	TREC50	500	57.6	11.0 ± 0.0
TextConvNet	TREC50	1000	67.4	42.1 ± 2.1

TABLE IV: Distillation ratios for text datasets and their associated neural networks. The number of distilled sentences, M, is specified ahead of time. Experiments with random initializations have their results listed in the form [mean ± standard deviation] and are based on the resulting performance of 200 randomly initialized networks.

The baseline results are shown in Table II.

Fixed initialization. TDD achieves impressive performance in fixed initialization experiments that are detailed in Tables II and IV. For example, TDD can produce 2 distilled sentences that train the TextConvNet up to a distillation ratio of almost 90% on the IMDB dataset. However, TDD does require a larger number of distilled samples M for more complicated datasets like TREC50. In Table V, we list four of the decoded sentences corresponding to the distilled embeddings from the six-sentence TREC6 experiments along with their respective label distributions. These sentences can contain any tokens found in the TREC6 dataset, including punctuation, numbers, abbreviations, etc. The sentences do not have much overlap which is consistent with the distilled labels suggesting that each sentence corresponds strongly to a different class. It

Distilled Sentence	Label Logits for each Class					
	ENT	HUM	DES	NUM	LOC	ABB
allan milk banned yellow planted successfully introduced bombay 1936 grass mines iron delhi 1942 male heir throne oath clouds 7th	**2.72**	-0.48	-0.07	-0.62	-0.53	-0.27
whom engineer grandfather joan officer entered victoria 1940s taxi romania motorcycle italian businessman photographer powerful driving u brilliant affect princess	-0.05	**3.21**	-1.15	-0.79	-0.71	-0.64
necessarily factors pronounced pronounced define bow destroying belonged balls 1923 storms buildings 1925 victorian sank dragged reputation sailed nn occurs	-0.67	-0.66	**3.28**	-0.27	-0.77	0.47
accommodate accommodate peak 2.5 adults thin teenagers hike aged nurse policeman admit aged median philippines define baghdad libya ambassador admit	-0.98	-0.14	-1.12	**5.57**	-0.85	-1.85

TABLE V: TDD can learn 6 distilled sentences of length 30 that train networks with fixed initializations from 12.6% to 87.4% ($r_6 = 97.8$). Each row contains the first 20 (out of 30) words of the nearest decoding for 4 of these sentences accompanied by their label logits. Classes are denoted by their abbreviations: ENT - Entities, HUM - Human beings, DES - Description/abstract concepts, NUM - Numeric values, LOC - Locations, ABB - Abbreviations

Distilled Sentence	Label
editor panoramic brewing swat regency medley arts fleet attained hilary novelist rugged medal who abbot has sweden new ensemble member understands alba archaeologist operatic intercontinental martian marshal paste smooth titular english-language adaptation songwriter historian enlightenment royal gaelic ceo author macarthur skipper honored excellence distance most endings collaborations his mythological polanski mayer choreographer grisham eminent brooke sympathies modelling vitality dictionary dedication farewell enjoys energetic jordan equality lectures sophia elijah maureen novelist	2.53
dump misled speculate bait substandard uncovered lump corpses 911 punched whopping discarded ref dollar were dough sided brink unconscious tomatoes locking trash burying punched diversion grenade overboard cashier wards agreeing brent prematurely knife randomly stupid buster wipe virus pap waste inflated loan patient peg eliminates nudity worms ?? rotten shoddy strangled substandard boil whopping tunnels steep unjust dummy satisfactory mistakenly adopt tightened bloated hacked misled dump 3-4 untrue contaminated bureaucracy waste	-2.21

TABLE VI: TDD can learn 2 distilled sentences of length 400 that train networks with random initializations from 50.0% to $69.6\% \pm 5.5\%$ ($r_2 = 79.96$). Each sentence is accompanied by its associated soft label logit. Only a segment of 70 (out of 400) words is shown for each sentence. The first sentence corresponds to positive sentiment, and the second to negative.

appears that TDD encourages class separation; however, some mixing still occurs and some words do not seem to match the associated class with the highest logit.

Random initialization. TDD experiments with random initialization are detailed in Tables II and IV. On IMDB, TDD has only a slight performance drop compared to fixed initialization, but a larger drop between for recurrent networks and the more difficult TREC50 task. We show a sample of the two decoded sentences from the IMDB experiment with $M = 2$ in Table VI. As this is a binary classification task, each label is a scalar. TDD produced one sentence with a positive sentiment and one with a negative sentiment. The model appears to have overcome the challenge of having to describe long sentences with a single scalar by using duplication. For example in the negative sentence, words like 'contaminated' and 'misled' are repeated. In such a way, the algorithm may be assigning lower sentiment scores to individual words.

V. Conclusion

By introducing learnable distilled labels, we have increased distillation accuracy across multiple datasets by up to 6%. By enabling text distillation, we have also greatly increased the types of datasets and architectures with which distillation can be used. In fact, as long as a network has a twice-differentiable loss function and the gradient can be back-propagated to the inputs, then that network is compatible with dataset distillation. However, there are still some limitations to dataset distillation. The network initializations come from the same distribution, and no testing has yet been done on whether a single distilled dataset can be used to train networks with different architectures.

As mentioned above, the initialization of distilled labels appears to affect performance. It is not clear whether it is better to separate similar classes (e.g. '3' and '8' in MNIST) thereby encouraging the network to discern between them, or to instead keep those classes together thus allowing soft-label information to be shared between them. It may be interesting to explore the dynamics of distillation when using a variety of label initialization methods. The pre-specified number of distilled samples also appears to have a large effect. Generally, we found that adding more distilled samples results in higher performance, but we quickly ran into issues such as overfitting and decreasing returns as we experimented with larger numbers of distilled samples. Meanwhile, reducing the number of distilled images for MNIST revealed that soft labels allow us to represent the dataset using less than one sample per class.

More broadly, distilled datasets enable faster training which is particularly useful for federated learning, as well as for other computationally intensive meta-algorithms like neural architecture search. When distilled datasets are a good proxy for performance evaluation, they may reduce training times by multiple orders of magnitude.

REFERENCES

[1] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019. [Online]. Available: http://dx.doi.org/10.18653/v1/P19-1355

[2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[3] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[4] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros, "Dataset distillation," *arXiv preprint arXiv:1811.10959*, 2018.

[5] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, *Object Recognition with Gradient-Based Learning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 319–345. [Online]. Available: https://doi.org/10.1007/3-540-46805-6_19

[7] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150.

[8] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2006, pp. 535–541.

[9] Y. Kim and A. M. Rush, "Sequence-level knowledge distillation," *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016. [Online]. Available: http://dx.doi.org/10.18653/v1/D16-1139

[10] N. El Gayar, F. Schwenker, and G. Palm, "A study of the robustness of knn classifiers trained using soft labels," in *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*. Springer, 2006, pp. 67–80.

[11] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of Artificial Intelligence Research*, vol. 4, pp. 129–145, 1996.

[12] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *Journal of Machine Learning Research*, vol. 2, no. Nov, pp. 45–66, 2001.

[13] A. Angelova, Y. Abu-Mostafam, and P. Perona, "Pruning training sets for learning of object categories," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 494–501.

[14] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core vector machines: Fast SVM training on very large data sets," *Journal of Machine Learning Research*, vol. 6, no. Apr, pp. 363–392, 2005.

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

[16] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," *arXiv preprint arXiv:1708.00489*, 2017.

[17] I. Triguero, J. Derrac, S. Garcia, and F. Herrera, "A taxonomy and experimental study on prototype generation for nearest neighbor classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 1, pp. 86–100, 2011.

[18] S. Garcia, J. Derrac, J. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 3, pp. 417–435, 2012.

[19] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[20] R. Munos, T. Stepleton, A. Harutyunyan, and M. Bellemare, "Safe and efficient off-policy reinforcement learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 1054–1062.

[21] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 3303–3313.

[22] J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee, "Sample-efficient reinforcement learning with stochastic ensemble value expansion," in *Advances in Neural Information Processing Systems*, 2018, pp. 8224–8234.

[23] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[24] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=rJY0-Kcll

[25] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in neural information processing systems*, 2017, pp. 4077–4087.

[26] J. Goetz and A. Tewari, "Federated learning via synthetic data," *arXiv preprint arXiv:2008.04489*, 2020.

[27] Y. Zhou, G. Pu, X. Ma, X. Li, and D. Wu, "Distilled one-shot federated learning," *arXiv preprint arXiv:2009.07999*, 2020.

[28] I. Sucholutsky and M. Schonlau, "Secdd: Efficient and secure method for remotely training neural networks," *arXiv preprint arXiv:2009.09155*, 2020.

[29] G. Li, R. Togo, T. Ogawa, and M. Haseyama, "Soft-label anonymous gastric x-ray image distillation," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 305–309.

[30] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[31] S. Song, Y. Shao, and J. Li, "Loosely coupled federated learning over generative models," *arXiv preprint arXiv:2009.12999*, 2020.

[32] T. Shen, J. Zhang, X. Jia, F. Zhang, G. Huang, P. Zhou, F. Wu, and C. Wu, "Federated mutual learning," *arXiv preprint arXiv:2006.16765*, 2020.

[33] M. M. Amiri, D. Gunduz, S. R. Kulkarni, and H. V. Poor, "Federated learning with quantized global model updates," *arXiv preprint arXiv:2006.10672*, 2020.

[34] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x

[35] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[36] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016. [Online]. Available: http://dx.doi.org/10.18653/v1/P16-1101

[37] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *Proceedings of the 2019 Conference of the North*, 2019. [Online]. Available: http://dx.doi.org/10.18653/v1/N19-1423

[38] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018. [Online]. Available: http://dx.doi.org/10.18653/v1/N18-1202

[39] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. PMLR, 2010, pp. 249–256.

[40] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1631–1642.

[41] E. M. Voorhees *et al.*, "The TREC-8 question answering track report," in *the Proceedings of the Eighth Text Retrieval Conference (TREC-8)*, vol. 99, 1999, pp. 77–82.

[42] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[43] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.