
Flexible Dataset Distillation: Learn Labels Instead of Images

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 We study the problem of dataset distillation – creating a small set of synthetic
2 examples capable of training a good model. In particular, we study the problem
3 of *label* distillation – creating synthetic labels for a small set of real images,
4 and show it to be more effective than the prior *image*-based approach to dataset
5 distillation. Methodologically, we introduce a more robust and flexible meta-
6 learning algorithm for distillation, as well as an effective first-order strategy based
7 on convex optimization layers. Distilling labels with our new algorithm leads to
8 improved results over prior image-based distillation. More importantly, it leads to
9 clear improvements in flexibility of the distilled dataset in terms of compatibility
10 with off-the-shelf optimizers and diverse neural architectures. Interestingly, label
11 distillation can be applied across datasets, for example enabling learning Japanese
12 character recognition by training only on synthetically labeled English letters.

13 1 Introduction

14 Distillation is a topical area of neural network research that initially began with the goal of extracting
15 the knowledge of a large pre-trained model and compiling it into a smaller model, while retaining
16 similar performance [11]. The notion of distillation has since found numerous applications and uses
17 including the possibility of *dataset* distillation [34]: extracting the knowledge of a large dataset
18 and compiling it into a small set of carefully crafted examples, such that a model trained on the
19 small dataset alone achieves good performance. This is of scientific interest as a tool to study neural
20 network generalisation under small sample conditions. More practically, it has the potential to address
21 the large and growing logistical and energy hurdle of neural network training, if adequate neural
22 networks can be quickly trained on small distilled datasets rather than massive raw datasets.

23 Nevertheless, progress towards the vision of dataset distillation has been limited as the performance of
24 existing methods [34, 30] trained from random initialization is far from that of full dataset supervised
25 learning. More fundamentally, existing approaches are relatively *inflexible* in terms of the distilled
26 data being over-fitted to the training conditions under which it was generated. While there is some
27 robustness to choice of initialization weights [34], the distilled dataset is largely specific to the
28 architecture used to train it (preventing its use to accelerate neural architecture search, for example),
29 and must use a highly-customized learner (a specific image visitation sequence, a specific sequence
30 of carefully chosen meta-learned learning rates, and a specific number of learning steps). Altogether
31 these constraints mean that existing distilled datasets are not general purpose enough to be useful in
32 practice, e.g. with off-the-shelf learning algorithms. We propose a more *flexible* approach to dataset
33 distillation underpinned by both algorithmic improvements and changes to the problem definition.

34 Rather than creating synthetic images [34] for arbitrary labels, or a combination of synthetic images
35 and soft labels [30], we focus on crafting synthetic labels for arbitrarily chosen standard images.
36 Compared to these prior approaches focused on synthetic images, label distillation benefits from

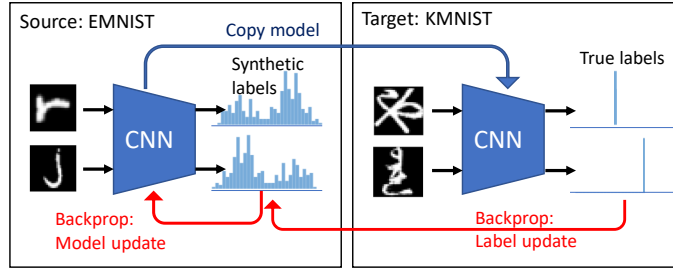


Figure 1: Label distillation enables training a model that can classify Japanese characters after being trained only on English letters and synthetic labels. Labels are updated only during meta-training, after which a new model is trained, using only the few source examples and their synthetic labels.

exploiting the data statistics of natural images and the lower-dimensionality of labels compared to images as parameters for meta-learning. Practically, this leads to improved performance compared to prior image distillation approaches. As a byproduct, this enables a new kind of cross-dataset knowledge distillation (Figure 1). One can learn solely on a source dataset (such as English characters) with synthetic distilled labels, and apply the learned model to recognise concepts in a disjoint target dataset (such as Japanese characters). Surprisingly, it turns out that models can make progress on learning to recognise Japanese only through exposure to English characters with synthetic labels.

Methodologically, we define a new meta-learning algorithm for distillation that does not require costly evaluation of multiple inner-loop (model-training) steps for each iteration of distillation. More importantly our algorithm leads to a more flexible distilled dataset that is better transferable across optimizers, architectures, learning iterations, etc. Furthermore, where existing dataset distillation algorithms rely on second-order gradients, we introduce an alternative learning strategy based on convex optimization layers that avoids high-order gradients and provides better optimization, thus improving the quality of the distilled dataset.

In summary, we contribute: (1) A dataset distillation method that produces flexible distilled datasets that exhibit transferability across learning algorithms. This brings us one step closer to producing useful general-purpose distilled datasets. (2) Our distilled datasets can be used to train higher performance models than those prior work. (3) We introduce the novel concept of cross-dataset distillation, and demonstrate proofs of concept, such as English→Japanese letter recognition.

2 Related work

Dataset distillation Most closely related to our work is Dataset [34] and Soft-Label Dataset Distillation [30]. They focus on distilling a dataset or model [23] into a small number of example images, which are then used to train a new model. This can be seen as solving a meta-learning problem with respect to model’s training data [12]. The common approach is to initialise the distilled dataset randomly, use the distilled data to train a model, and then backpropagate through the model and its training updates to take gradient steps on the dataset. Since the ‘inner’ model training algorithm is gradient-based, this leads to high-order gradients. To make this process tractable, the original Dataset Distillation [34] uses only a few gradient steps in its inner loop (as per other famous meta-learners [10]). To ensure that sufficient learning progress is made with few updates, it also meta-learns a fixed set of optimal learning rates to use at each step. This balances tractability & efficacy, but causes the distilled dataset to be ‘locked in’ to the customized optimizer rather than serve as a general purpose dataset, which also prevents its use for NAS [29]. In this work we define an online meta-learning procedure that simultaneously learns the dataset and the base model. This enables us to tractably take more gradient steps and ultimately produce a performant yet flexible general purpose distilled dataset.

There are various motivations for dataset distillation, but the most practically intriguing is to summarize a dataset in compressed form to accelerate model training. In this sense it is related to dataset pruning [1, 9, 15], core-set construction [32, 2, 28] and instance selection [25] focusing on dataset summarization through a small number of examples. Summarization methods *select* a relatively large part of the data (e.g. at least 10%), while distillation extends down to using 10 images per category

($\approx 0.2\%$ of CIFAR-10 data) through example *synthesis*. We keep original data (like summarization methods), but synthesize labels (like distillation). This leads to a surprising observation – it is possible to synthesize labels for a few fixed examples so a model trained on these examples can directly (without any fine-tuning) solve a different problem with a different label space (Figure 1).

Meta-learning Meta-learning algorithms can often be grouped [12] into offline approaches (e.g. [34, 10]) that do inner optimization at each step of outer optimization; and online approaches that solve the base and meta-learning problem simultaneously (e.g. [3, 18]). Meta-learning relates to hyperparameter optimization, for example [22, 20] efficiently unroll through many steps of optimization like offline meta-learning, while [21] optimize hyperparameters and the base model like online meta-learning. Online approaches are typically faster, but optimize meta-parameters for a single problem. Offline approaches are slower and typically limit the length of the inner optimization for tractability, but can often find meta-parameters that solve a distribution of tasks (as different tasks are drawn in each outer-loop iteration). In dataset distillation, the notion of ‘distribution over tasks’ corresponds to finding a dataset that can successfully train a network in many settings, such as different initial conditions [34]. Our distillation algorithm is a novel hybrid of these two families. We efficiently solve the base and meta-tasks simultaneously like online approaches, and so are able to use more inner-loop steps. However, we also learn to solve many ‘tasks’ by detecting meta-overfitting and sampling a new ‘task’ when this occurs. This leads to a good combination of efficacy and efficiency. Finally, most gradient-based meta-learning algorithms rely on costly and often unstable higher-order gradients [12, 10, 34], or else make simple shortest-path first-order approximations [24]. Instability and large variance of higher-order gradients may make meta-learning less effective [19, 8], so we found inspiration in recent approaches in few-shot learning [4, 17] that avoid this issue through the use of convex optimization layers. We introduce the notion of a pseudo-gradient that enables this idea to scale beyond the few-shot setting to general meta-learning problems such as dataset distillation.

3 Methods

We aim to meta-learn synthetic labels for a fixed set of *base* examples that can be used to train a randomly initialized model. This corresponds to an objective such as (Eq. 1):

$$\tilde{\mathbf{y}}^* = \arg \min_{\tilde{\mathbf{y}}} \ell((\mathbf{x}_t, \mathbf{y}_t), \theta - \alpha \nabla_{\theta} \ell((\tilde{\mathbf{x}}, \tilde{\mathbf{y}}), \theta)), \quad (1)$$

where $\tilde{\mathbf{y}}$ are the distilled labels for base examples $\tilde{\mathbf{x}}$, $(\mathbf{x}_t, \mathbf{y}_t)$ are the target set examples and labels and θ is the neural network model learned with rate α and loss ℓ (we use cross-entropy). We learn soft labels, so the dimensionality of each label is the number of classes C to distill. One gradient step is shown above, but in general there may be multiple steps.

One option to achieve objective in (Eq. 1) would be to follow Wang et al. [34] and simulate the whole training procedure for multiple gradient steps ∇_{θ} within the inner loop. However, this requires back-propagating through a long inner loop, and ultimately requires a fixed training schedule with optimized learning rates. We aim to produce a dataset that can be used in a standard training pipeline downstream (e.g. Adam optimizer with the default parameters).


Our first modification to the standard pipeline is to perform gradient descent iteratively on the model and the distilled labels, rather than performing many inner (model) steps for each outer (dataset) step. This increases efficiency significantly due to a shorter compute graph for backpropagation. Nevertheless, when there are very few training examples, the model converges quickly to an overfitted local minimum, likely within a few hundred iterations. To manage this, our innovation is to detect overfitting when it occurs, reset the model to a new random initialization and keep training. Specifically, we measure the moving average of target problem accuracy, and when it has not improved for set number of iterations, we reset the model. This periodic reset of the model after varying number of iterations is helpful for learning labels that are useful for all stages of training and thus less sensitive to the number of iterations used. To ensure scalability to any number of examples, we sample a minibatch of base examples and synthetic labels and use those to update the model, which also better aligns with standard training practice. Once label distillation is done, we train a new model from scratch using random initial weights, given the base examples and learned synthetic labels.

We propose and evaluate two label distillation algorithms: a second-order version that performs one update step within the inner loop; and a first-order version that uses a closed form solution of ridge regression to find optimal classifier weights for the base examples.

Algorithm 1 Label distillation with ridge regression (RR)

```
1: Input:  $\tilde{\mathbf{x}}$ :  $N$  unlabeled examples from the base dataset;  $(\mathbf{x}_t, \mathbf{y}_t)$ : labeled examples from the
   target dataset;  $\beta$ : step size;  $\alpha$ : pseudo-gradient step size;  $n_o, n_i$ : outer (resp. inner) loop batch
   size;  $C$  number of classes in the target dataset
2: Output: distilled labels  $\tilde{\mathbf{y}}$  and a reasonable number of training steps  $T_i$ 
3:  $\tilde{\mathbf{y}} \leftarrow 1/C$  //Uniformly initialize synthetic labels
4:  $\theta \sim p(\theta)$  //Randomly initialize feature extractor parameters
5:  $w \leftarrow 0$  // Initialize global RR classifier weights
6: while  $\tilde{\mathbf{y}}$  not converged do
7:    $(\tilde{\mathbf{x}}', \tilde{\mathbf{y}}') \sim (\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$  //Sample a minibatch of  $n_i$  base examples
8:    $(\mathbf{x}'_t, \mathbf{y}'_t) \sim (\mathbf{x}_t, \mathbf{y}_t)$  //Sample a minibatch of  $n_o$  target examples
9:    $w_l = \arg \min_{w_l} \mathcal{L}(f_{w_l}(f_\theta(\tilde{\mathbf{x}}')), \tilde{\mathbf{y}}')$  // Solve RR problem
10:   $w \leftarrow (1 - \alpha)w + \alpha w_l$  //Update global RR classifier weights
11:   $\tilde{\mathbf{y}} \leftarrow \tilde{\mathbf{y}} - \beta \nabla_{\tilde{\mathbf{y}}} \mathcal{L}(f_w(f_\theta(\mathbf{x}'_t)), \mathbf{y}'_t)$  //Update synthetic labels with target loss
12:   $\theta \leftarrow \theta - \beta \nabla_{\theta} \mathcal{L}(f_w(f_\theta(\tilde{\mathbf{x}}')), \tilde{\mathbf{y}}')$  //Update feature extractor
13:  if  $\mathcal{L}(f_w(f_\theta(\mathbf{x}_t)), \mathbf{y}_t)$  did not improve then
14:     $\theta \sim p(\theta)$  //Reset feature extractor
15:     $w \leftarrow 0$  // Reset global RR classifier weights
16:     $T_i \leftarrow$  iterations since previous reset //Record time to overfit
17:  end if
18: end while
```

128 **Vanilla second-order version** The training procedure includes both inner and outer loop. The
129 inner loop consists of one update of the model $\theta' = \theta - \alpha \nabla_{\theta} \mathcal{L}(f_{\theta}(\tilde{\mathbf{x}}'), \tilde{\mathbf{y}}')$, through which we then
130 backpropagate to update the synthetic labels $\tilde{\mathbf{y}} \leftarrow \tilde{\mathbf{y}} - \beta \nabla_{\tilde{\mathbf{y}}} \mathcal{L}(f_{\theta'}(\mathbf{x}'_t), \mathbf{y}'_t)$. We perform a standard
131 update of the model after updating the synthetic labels, which is more of a technical detail about how
132 it is implemented rather than needed in theory as it could be combined with the inner-loop update.
133 The method is otherwise similar to our first-order ridge regression method, which is summarized in
134 Algorithm 1, together with our notation. Examples from the target dataset are used when updating the
135 synthetic labels $\tilde{\mathbf{y}}$, but for updating the model θ we only use synthetic labels and the base examples.
136 After each update of the labels, we normalize them to represent a valid probability distribution. This
137 makes them interpretable and has led to improvements compared to unnormalized labels.

138 **Intuition** We analysed how the synthetic labels are meta-learned by the second-order algorithm
139 for a simple one-layer model θ , with a sigmoid output unit and two classes (details are in the
140 supplementary). The meta-gradients are $\nabla_{\tilde{\mathbf{y}}} \mathcal{L}(f_{\theta'}(\mathbf{x}'_t), \mathbf{y}'_t) = \alpha (\sigma(\theta'^T \mathbf{x}'_t) - \mathbf{y}'_t) \mathbf{x}'_t{}^T \tilde{\mathbf{x}}'$. This
141 shows synthetic labels are updated proportionally to the similarity of the target dataset examples \mathbf{x}'_t
142 and base examples $\tilde{\mathbf{x}}'$ as well as the difference between the predictions and true target set labels. Thus
143 synthetic labels capture the different degrees of similarity between a base example and examples
144 from different classes in the target dataset. For example, in cross-dataset, the KMNIST ‘Ya’  has no
145 corresponding English symbol, but could be learned by partially assigning its label to similar looking
146 English ‘X’s and ‘R’s.

147 **First-order version with ridge regression** To avoid second-order gradients, we propose a first-
148 order version that uses pseudo-gradient generated via a closed form solution to ridge regression (RR).
149 RR layers have previously been used for few-shot learning [4] *within* minibatch. We extend it to learn
150 global weights that persist across minibatches. Ridge regression can be defined and solved as:

$$\begin{aligned} W_l &= \arg \min_W \|XW - Y\|^2 + \lambda \|W\|^2 \\ &= (X^T X + \lambda I)^{-1} X^T Y, \end{aligned} \quad (2)$$

151 where X are the image features, Y are the labels, W are the ridge regression weights and λ is the
152 regularization parameter. Following Bertinetto et al. [4], we use Woodbury formula [27]:

$$W_l = X^T (X X^T + \lambda I)^{-1} Y, \quad (3)$$

153 which allows us to use matrix $X X^T$ with dimensionality depending on the square of the number of
154 inputs (minibatch size) rather than the square of the input embedding size. This makes the matrix

inversion significantly less costly in practice. While ridge regression is usually oriented at regression problems, it has been shown [4] to work well for classification when regressing label vectors.

Ridge regression with pseudo-gradients We use a RR layer as the final output layer of our base network – we decompose the original model θ used for the second-order version into feature extractor θ and RR classifier w . RR solves for the optimal (local) weights w_l that classify the features of the current minibatch examples. We exploit this local minibatch solution by taking a pseudo-gradient step that updates global weights w as $w \leftarrow (1 - \alpha)w + \alpha w_l$, with α being the pseudo-gradient step size. We can understand this as a pseudo-gradient as it corresponds to the step $w \leftarrow w - \alpha(w - w_l)$. We can then update the synthetic labels by back-propagating through local weights w_l . Subsequent feature extractor updates on θ avoid second-order gradients. The process is summarised in Algorithm 1.

4 Experiments

We perform two main types of experiments: (1) within-dataset distillation, when the base examples come from the target dataset and (2) cross-dataset distillation, when the base examples come from a different but related dataset. The dataset should be related because if there is a large shift in the domain (e.g. from characters to photos), then the feature extractor trained on the base examples would generalize poorly to the target dataset. We use MNIST, CIFAR-10 and CIFAR-100 for the task of within-dataset distillation, while for cross-dataset distillation we use EMNIST (“English letters”), KMNIST, Kuzushiji-49 (both “Japanese letters”), MNIST (digits), CUB (birds) and CIFAR-10 (general objects). Details of these datasets are in the supplementary.

4.1 Experimental settings

Monitoring overfitting We use parameters N_M, N_W to say over how many iterations to calculate the moving average and how many iterations to wait before reset since the best moving average value. We select $N_M = N_W$ and use a value of 50 steps in most cases, while we use 100 for CIFAR-100 and Kuzushiji-49, and 200 for other larger-scale experiments (more than 100 base examples). These parameters do not affect the total number of iterations.

Early stopping for learning synthetic labels We update the synthetic labels for a given number of epochs and then select the best labels to use based on the validation performance. For this, we train a new model from scratch using the current distilled labels and the associated base examples and then evaluate the model on the validation part of the target set. We randomly set aside about 10-15% (depending on the dataset) of the training data for validation.

Models We use LeNet [16] for MNIST and similar experiments, and AlexNet [14] for CIFAR-10, CIFAR-100 and CUB. Both models are identical to the ones used in [34]. In a fully supervised setting they achieve about 99% and 80% test accuracy on MNIST and CIFAR-10.

Selection of base examples The base examples are selected randomly, using a shared random seed for consistency across scenarios. Our baseline models use the same random seed as the distillation models, so they share base examples for fair comparison. For within-dataset label distillation, we create a balanced set of base examples, so each class has the same number of base examples. For cross-dataset label distillation, we do not consider the original classes of base examples. The size of the label space and the labels are different in the source and the target problem. Our additional analysis (Tables 6 and 7 in the supplementary) has shown that the specific random set of base examples does not have a significant impact on the success of label distillation.

Further details Outer loop minibatch size is $n_o = 1024$ examples, while the inner minibatch size n_i depends on the number of base examples used. When using 100 or more base examples, we select a minibatch of 50 examples, except for CIFAR-100 for which we use 100 examples. For 10, 20 and 50 base examples our minibatch sizes are 10, 10 and 25. We optimize the synthetic labels and the model itself using Adam optimizer with the default parameters ($\beta = 0.001$). Most of our models are trained for 400 epochs, while larger-scale models (more than 100 base examples and CIFAR-100) are trained for 800 epochs. Smaller-scale Kuzushiji-49 experiments are trained for 100 epochs, while larger-scale ones use 200 epochs. Epochs are calculated based on the number of target set examples, rather than base examples. In the second-order version, we perform one inner-loop step update, using a learning rate of $\alpha = 0.01$. We back-propagate through the inner-loop update when updating the synthetic labels (meta-knowledge), but not when subsequently updating the model θ with Adam

Table 1: Within-dataset distillation recognition accuracy (%). Our label distillation (LD) outperforms prior Dataset Distillation [34] (DD) and SLDD [30], and scales to synthesizing more examples.

	Base examples	10	20	50	100	200	500
MNIST	LD	60.89 \pm 3.20	74.37 \pm 1.27	82.26 \pm 0.88	87.27 \pm 0.69	91.47 \pm 0.53	93.30 \pm 0.31
	Baseline	48.35 \pm 3.03	62.60 \pm 3.33	75.07 \pm 2.40	82.06 \pm 1.75	85.95 \pm 0.98	92.10 \pm 0.43
	Baseline LS	51.22 \pm 3.18	64.14 \pm 2.57	77.94 \pm 1.26	85.95 \pm 1.09	90.10 \pm 0.60	94.75 \pm 0.29
	LD RR	64.57 \pm 2.67	75.98 \pm 1.00	82.49 \pm 0.93	87.85 \pm 0.43	88.88 \pm 0.28	89.91 \pm 0.33
	Baseline RR	52.53 \pm 2.61	60.44 \pm 1.97	74.85 \pm 2.37	81.40 \pm 2.11	87.03 \pm 0.69	92.10 \pm 0.80
	Baseline RR LS	51.53 \pm 2.42	60.91 \pm 1.78	76.26 \pm 1.80	83.13 \pm 1.41	87.94 \pm 0.67	93.48 \pm 0.61
	DD [34]				79.5 \pm 8.1		
	SLDD [30]				82.7 \pm 2.8		
CIFAR-10	LD	25.69 \pm 0.72	30.00 \pm 0.86	35.36 \pm 0.64	38.33 \pm 0.44	41.05 \pm 0.71	42.45 \pm 0.40
	Baseline	14.29 \pm 1.40	16.80 \pm 0.72	20.75 \pm 1.05	25.76 \pm 1.04	31.53 \pm 1.02	38.33 \pm 0.75
	Baseline LS	13.22 \pm 1.22	18.36 \pm 0.65	22.81 \pm 0.71	27.27 \pm 0.68	33.62 \pm 0.81	39.22 \pm 1.12
	LD RR	25.07 \pm 0.69	29.83 \pm 0.46	35.23 \pm 0.64	37.94 \pm 1.22	41.17 \pm 0.33	43.16 \pm 0.47
	Baseline RR	13.37 \pm 0.79	17.08 \pm 0.31	19.85 \pm 0.51	24.65 \pm 0.47	28.97 \pm 0.74	36.31 \pm 0.49
	Baseline RR LS	13.82 \pm 0.85	16.95 \pm 0.52	20.00 \pm 0.57	24.84 \pm 0.60	29.28 \pm 0.56	35.73 \pm 1.02
	DD [34]				36.8 \pm 1.2		
	SLDD [30]				39.8 \pm 0.8		

optimizer. In the RR version, we use a pseudo-gradient step size of 0.01 and λ of 1.0. We calibrate the regression weights by scaling them with a value learned during training with the specific set of base examples and distilled labels. Our tables report the mean test accuracy and standard deviation (%) across 20 models trained from scratch using the base examples and synthetic labels.

4.2 Within-dataset distillation

We compare our label distillation (LD) to previous dataset distillation (DD) and soft-label dataset distillation (SLDD) on MNIST and CIFAR-10. We also establish new baselines that take true labels from the target dataset and otherwise are trained in the same way as LD models. RR baselines use RR and pseudo-gradient for consistency with LD RR (overall network architecture remains the same as in the second-order approach). In addition, we include baselines that use label smoothing (LS) [31] with a smoothing parameter of 0.1 as suggested in [26]. The number of training steps for our baselines is optimized using the validation set, by training a model for various numbers of steps between 10 and 1000 and measuring the validation set accuracy (up to 1700 steps are used for cases with more than 100 base examples). The results in Table 1 show that LD significantly outperforms previous work on MNIST. This is in part due to LD enabling the use of more steps (LD estimates $T_i \approx 200 - 300$ steps vs fixed 3 epochs of 10 steps in LD and SLDD). Our improved baselines are also competitive, and outperform the prior baselines in [34] due to taking more steps. However, since approaches from Wang et al. [34], Sucholutsky and Schonlau [30] do not scale to more steps, this is a reasonable comparison. The standard uniform label smoothing baseline works well on MNIST for a large number of base examples, where the problem anyway approaches one of conventional supervised learning. However, this strategy has not shown to be effective enough for CIFAR-10, where synthetic labels are the best. Importantly, in the most intensively distilled regime of 10 examples, LD clearly outperforms all competitors. We provide an analysis of the labels learned by our method in Section 4.5. For CIFAR-10 our results also improve on the original DD result. In this experiment, our second-order algorithm performs similarly to our RR pseudo-gradient strategy.

We further show in Table 2 that our distillation approach scales to a significantly larger number of classes than 10 by application to the CIFAR-100 benchmark. As before, we establish new baseline results that use the original labels (or their smoother alternatives) of the same images as those used as base examples in distillation. We use the validation set to choose a suitable number of steps for training a baseline, allowing up to 1000 steps, which is significantly more than what is typically used by LD. The results show our distillation method leads to clear improvements over the baseline.

Table 2: CIFAR-100 within-dataset distillation. One example per class.

LD	11.46 \pm 0.39
Baseline	3.51 \pm 0.31
Baseline LS	4.07 \pm 0.23
LD RR	10.80 \pm 2.36
Baseline RR	3.00 \pm 0.39
Baseline RR LS	3.65 \pm 0.28

Table 3: Cross-dataset distillation recognition accuracy (%). Datasets: E = EMNIST, M = MNIST, K = KMNIST, B = CUB, C = CIFAR-10, K-49 = Kuzushiji-49.

Base examples	10	20	50	100	200	500
E → M (LD)	36.13 ± 5.51	57.82 ± 1.91	69.21 ± 1.82	77.09 ± 1.66	83.67 ± 1.43	86.02 ± 1.00
E → M (LD RR)	56.08 ± 2.88	67.62 ± 3.03	80.80 ± 1.44	82.70 ± 1.33	84.44 ± 1.18	86.79 ± 0.76
E → K (LD)	31.90 ± 2.82	39.88 ± 1.98	47.93 ± 1.38	51.91 ± 0.85	57.46 ± 1.37	59.84 ± 0.80
E → K (LD RR)	34.35 ± 3.37	46.67 ± 1.66	53.13 ± 1.88	57.02 ± 1.24	58.01 ± 1.28	63.77 ± 0.71
B → C (LD)	26.86 ± 0.97	28.63 ± 0.86	31.21 ± 0.74	34.02 ± 0.66	38.39 ± 0.56	38.12 ± 0.41
B → C (LD RR)	26.50 ± 0.54	28.95 ± 0.47	32.23 ± 3.59	32.19 ± 7.89	36.55 ± 6.32	38.46 ± 6.97
E → K-49 (LD)	7.37 ± 1.01	9.79 ± 1.23	17.80 ± 0.78	19.17 ± 1.27	22.78 ± 0.98	23.99 ± 0.81
E → K-49 (LD RR)	10.48 ± 1.26	14.84 ± 1.83	21.59 ± 1.87	20.86 ± 1.81	24.59 ± 2.26	24.72 ± 1.78

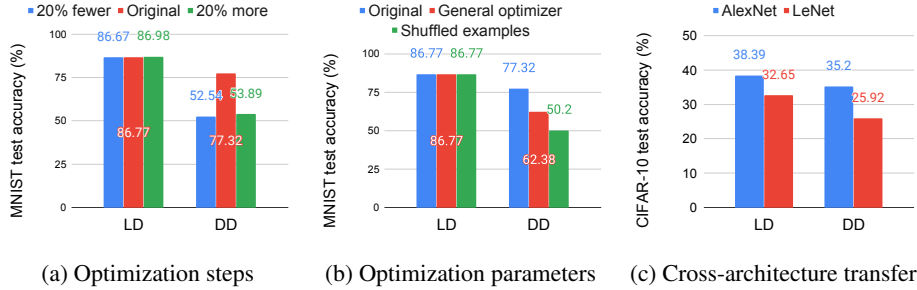


Figure 2: Datasets distilled with LD are more flexible than those distilled with DD.

4.3 Cross-dataset task

For cross-dataset distillation, we considered four scenarios: from EMNIST letters to MNIST digits, from EMNIST letters (“English”) to Kuzushiji-MNIST or Kuzushiji-49 characters (“Japanese”) and from CUB bird species to CIFAR-10 general categories. The results in Table 3 show we are able to distill labels on examples of a different source dataset and achieve surprisingly good performance on the target problem, given no target data is used when training these models. In contrast, directly applying a trained source-task model to the target without distillation unsurprisingly leads to chance performance (about 2% test accuracy for Kuzushiji-49 and 10% for all other cases). These results show we can indeed distill the knowledge of one dataset into base examples from a different but related dataset through crafting synthetic labels. Furthermore, our RR approach surpasses the second-order method in most cases, confirming its value. When using 10 and 20 base examples for Kuzushiji-49, the number of training examples is smaller than the number of classes (49), providing a novel example of *less-than-one-shot learning* where there are fewer examples than classes.

4.4 Flexibility of distilled datasets

We conduct experiments to verify the flexibility and general applicability of our label-distilled dataset compared to image-distilled alternative by Wang et al. [34]. Namely, we looked at (1) How the number of steps used during meta-testing affects the performance of learning with the distilled data, and in particular sensitivity to deviation from the number of steps used during meta-training of DD and LD. (2) Sensitivity of the models to changes in optimization parameters between meta-training + meta-testing. (3) How well the distilled datasets transfer to neural architectures different to those used for training. We used the DD implementation of Wang et al. [34] for fair evaluation. Figure 2 summarizes the key points, and detailed tables are in the supplementary.

Sensitivity to the number of meta-testing steps (Figure 2a) A key feature of our method is that it is relatively insensitive to the number of steps used to train a model on the distilled data. Our results from Table 9 show that even if we do 50 steps less or 100 steps more than the number estimated during training (≈ 300), the testing performance does not change significantly. This is in contrast with previous DD and SLDD methods that must be trained for a specific number of steps with optimized learning rates, making them rather inflexible. If the number of steps changes even by as little as 20%, they incur a significant cost in accuracy. Table 10 provides a further analysis of sensitivity of DD.

Sensitivity to optimization parameters (Figure 2b) DD uses step-wise meta-optimized learning rates to maximize accuracy. Our analysis in Table 11 shows that if we use an average of the optimized learning rate rather than the specific value in each step (more general optimizer), the performance decreases significantly. Further, original DD relies on training the distilled data in a fixed sequence, and our analysis in Table 12 shows changing the order of examples leads to a large decrease in accuracy. Our LD method by design does not depend on the specific order of examples, and it can be used with off-the-shelf learning optimizers such as Adam to train the model.

Transferability of labels across architectures (Figure 2c) We study the impact of distilling the labels using AlexNet and then training AlexNet, LeNet and ResNet-18 using the distilled labels. We establish new baselines for the additional architectures, using a number of steps chosen based on the validation set. Results in Tables 13 and 14 suggest our labels are helpful in both within and across dataset distillation scenarios. Even if we relabel a dataset with one label space to solve a different problem, the labels generalize to a new architecture and lead to improvements over the baselines. As shown in Table 15, there is a large decrease in accuracy for the original DD, but the results are better than their reported baselines [34]. This suggests their synthetic images are somewhat transferable if we train the model with the specific order of examples and optimized learning rates. However, our LD method incurs a smaller decrease in accuracy, suggesting better transferability across architectures.

4.5 Further analysis

Analysis of synthetic labels We have analysed to what extent the synthetic labels learn the true label and how this differs between our second-order and RR methods (Figures 6 and 7 in the supplementary). The results for a MNIST experiment with 100 base examples show the second-order method recovers the true value to about 84% on average, so it could be viewed as non-uniform label smoothing with meta-learned weights on labels specific to the examples. For the same scenario, RR recovers the true value to about 63% on average, suggesting RR leads to more complex labels that may better capture the different amounts of similarity of base examples to examples from different classes in the target dataset. This is also supported by visually similar digits such as 4 and 9 receiving more weight compared to other combinations. For CIFAR-10, the labels are significantly more complex, indicating non-trivial information is embedded into the synthetic labels. For cross-dataset LD (EMNIST base examples with KMNIST and MNIST target respectively), Figure 9 suggests LD can be understood as learning labels such that the base examples’ label-weighted combination resembles the images from the target class. Example synthetic labels are included in the supplementary.

Pseudo-gradient analysis To understand the superior performance of our RR method, we analysed the variances of meta-gradients obtained by our second-order and pseudo-gradient methods. Results in Figure 3 show our pseudo-gradient RR method obtains a significantly lower variance of meta-knowledge gradients, leading to more stable and effective training.

Discussion LD provides a more effective and more flexible distillation approach than prior work. This brings us one step closer to the vision of leveraging distilled data to accelerate model training, or design – such as architecture search [7]. Currently we only explicitly randomize over network initializations during training. In future work we believe our strategy of multi-step training and reset at convergence could be used with other factors, such as randomly selected network architectures to further improve cross-network generalisation performance.

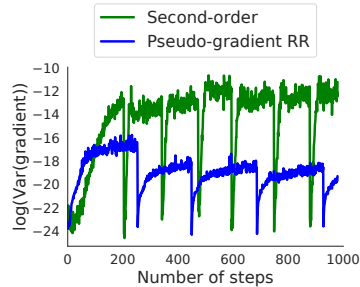


Figure 3: RR-based LD reduces synthetic label meta-gradient variance $\text{Var}[\nabla_{\tilde{y}}\mathcal{L}]$.

5 Conclusion

We have introduced a new label distillation algorithm for distilling the knowledge of a large dataset into synthetic labels of a few base examples from the same or a different dataset. Our method improves on prior dataset distillation results, scales better to larger problems, and enables novel settings such as cross-dataset distillation. Most importantly, it is significantly more flexible in terms of distilling general purpose datasets that can be used downstream with off-the-shelf optimizers.

References

- [1] Angelova, A., Abu-Mostafa, Y., and Perona, P. (2005). Pruning training sets for learning of object categories. In *CVPR*.
- [2] Bachem, O., Lucic, M., and Krause, A. (2017). Practical coreset constructions for machine learning. In *arXiv*.
- [3] Balaji, Y., Sankaranarayanan, S., and Chellappa, R. (2018). MetaReg: towards domain generalization using meta-regularization. In *NeurIPS*.
- [4] Bertinetto, L., Henriques, J., Torr, P. H. S., and Vedaldi, A. (2019). Meta-learning with differentiable closed-form solvers. In *ICLR*.
- [5] Clanuwat, T., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. (2018). Deep learning for classical Japanese literature. In *NeurIPS Workshop on Machine Learning for Creativity and Design*.
- [6] Cohen, G., Afshar, S., Tapson, J., and van Schaik, A. (2017). EMNIST: an extension of MNIST to handwritten letters. In *arXiv*.
- [7] Elsken, T., Metzen, J. H., and Hutter, F. (2019). Neural architecture search: a survey. *Journal of Machine Learning Research*, 20:1–21.
- [8] Farquhar, G., Whiteson, S., and Foerster, J. (2019). Loaded DiCE: Trading off Bias and Variance in Any-Order Score Function Estimators for Reinforcement Learning. In *NeurIPS*.
- [9] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645.
- [10] Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.
- [11] Hinton, G., Vinyals, O., and Dean, J. (2014). Distilling the knowledge in a neural network. In *NIPS*.
- [12] Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. (2020). Meta-learning in neural networks: a survey. In *arXiv*.
- [13] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report.
- [14] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *NIPS*.
- [15] Lapedriza, A., Pirsiavash, H., Bylinskii, Z., and Torralba, A. (2013). Are all training examples equally valuable? In *arXiv*.
- [16] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- [17] Lee, K., Maji, S., Ravichandran, A., and Soatto, S. (2019). Meta-learning with differentiable convex optimization. In *CVPR*.
- [18] Li, Y., Yang, Y., Zhou, W., and Hospedales, T. M. (2019). Feature-critic networks for heterogeneous domain generalization. In *ICML*.
- [19] Liu, H., Socher, R., and Xiong, C. (2019). Taming MAML: Efficient Unbiased Meta-Reinforcement Learning. In *ICML*.
- [20] Lorraine, J., Vicol, P., and Duvenaud, D. (2020). Optimizing Millions of Hyperparameters by Implicit Differentiation. In *AISTATS*.
- [21] Luketina, J., Berglund, M., Klaus Greff, A., and Raiko, T. (2016). Scalable Gradient-Based Tuning of Continuous Regularization Hyperparameters. In *ICML*.

- 369 [22] Maclaurin, D., Duvenaud, D., and Adams, R. P. (2015). Gradient-based Hyperparameter
370 Optimization through Reversible Learning. In *ICML*.
- 371 [23] Micaelli, P. and Storkey, A. (2019). Zero-shot knowledge transfer via adversarial belief matching.
372 In *NeurIPS*.
- 373 [24] Nichol, A., Achiam, J., and Schulman, J. (2018). On first-order meta-learning algorithms. In
374 *arXiv*.
- 375 [25] Olvera-López, J. A., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F., and Kittler, J. (2010). A
376 review of instance selection methods. *Artificial Intelligence Review*, 34(2):133–143.
- 377 [26] Pereyra, G., Tucker, G., Chorowski, J., Kaiser, L., and Hinton, G. (2017). Regularizing neural
378 networks by penalizing confident output distributions. In *arXiv*.
- 379 [27] Petersen, K. B. and Pedersen, M. S. (2012). The matrix cookbook. Technical report.
- 380 [28] Sener, O. and Savarese, S. (2018). Active learning for convolutional neural networks: A core-set
381 approach. In *ICLR*.
- 382 [29] Shleifer, S. and Prokop, E. (2019). Proxy Datasets for Training Convolutional Neural Networks.
383 In *arXiv*.
- 384 [30] Sucholutsky, I. and Schonlau, M. (2019). Soft-label dataset distillation and text dataset distilla-
385 tion. In *arXiv*.
- 386 [31] Szegedy, C., Vanhoucke, V., Ioffe, S., and Shlens, J. (2016). Rethinking the Inception architec-
387 ture for computer vision. In *CVPR*.
- 388 [32] Tsang, I. W., Kwok, J. T., and Cheung, P.-M. (2005). Core vector machines: fast SVM training
389 on very large data sets. *Journal of Machine Learning Research*, 6:363–392.
- 390 [33] Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. (2011). The Caltech-UCSD
391 Birds-200-2011 dataset. Technical report.
- 392 [34] Wang, T., Zhu, J.-Y., Torralba, A., and Efros, A. A. (2018). Dataset distillation. In *arXiv*.