Dataset Distillation

Yufang Liu

East China Normal University

Background

- Knowledge Distillation
  - large model -> small model
- Datsetset Distillation
  - large dataset -> small dataset
  - coreset & synthetic

DATASET DISTILLATION

Tongzhou Wang
Facebook AI Research, MIT CSAIL

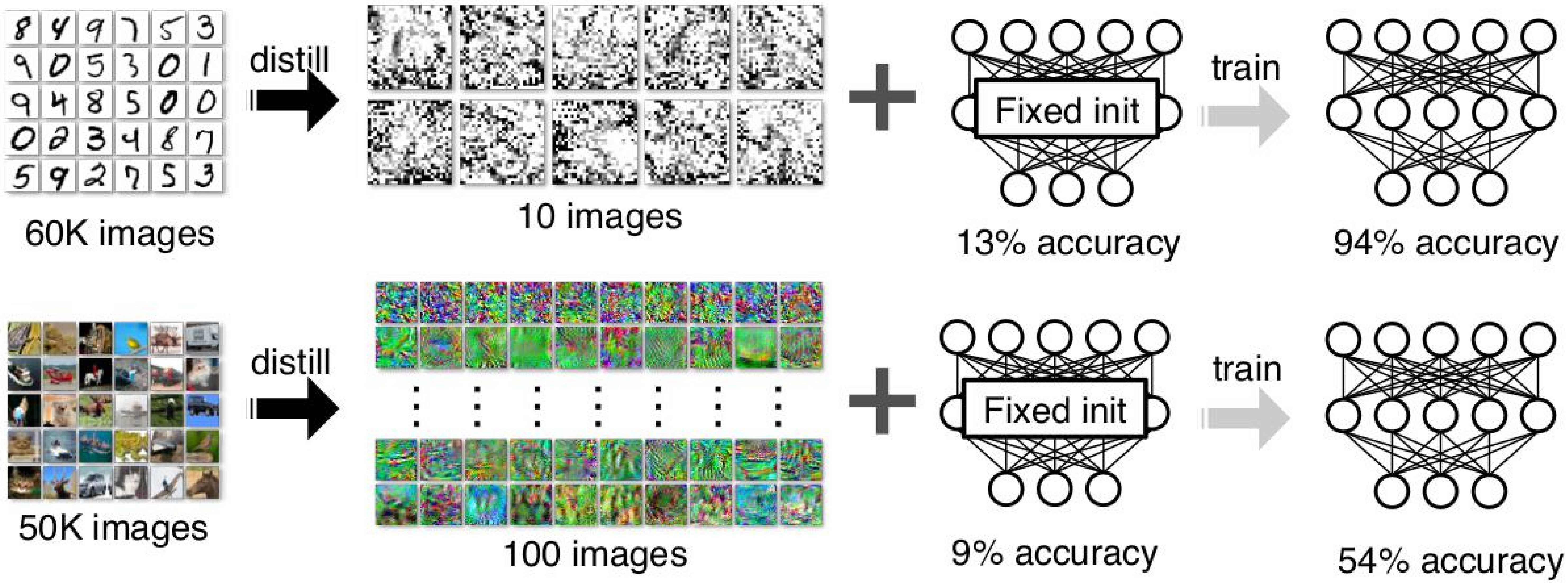
Jun-Yan Zhu
MIT CSAIL

Antonio Torralba
MIT CSAIL

Alexei A. Efros
UC Berkeley

2018

Motivation



(a) Dataset distillation on MNIST and CIFAR10

Methods

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

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

SGD update on syntactic data $$\theta_1 = \theta_0 - \tilde{\eta} \nabla_{\theta_0} \ell(\tilde{\mathbf{x}}, \theta_0) \qquad \tilde{\mathbf{x}} = \{\tilde{x}_i\}_{i=1}^{M} \text{ with } M \ll N$$

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

dependent on θ_0

Methods

- Distillation for random initializations

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

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 $\theta_0^{(j)} \sim p(\theta_0)$
5: **for each** sampled $\theta_0^{(j)}$ **do**
6: Compute updated parameter with GD: $\theta_1^{(j)} = \theta_0^{(j)} - \tilde{\eta} \nabla_{\theta_0^{(j)}} \ell(\tilde{\mathbf{x}}, \theta_0^{(j)})$
7: Evaluate the objective function on real training data: $\mathcal{L}^{(j)} = \ell(\mathbf{x}_t, \theta_1^{(j)})$
8: **end for**
9: Update $\tilde{\mathbf{x}} \leftarrow \tilde{\mathbf{x}} - \alpha \nabla_{\tilde{\mathbf{x}}} \sum_j \mathcal{L}^{(j)}$, and $\tilde{\eta} \leftarrow \tilde{\eta} - \alpha \nabla_{\tilde{\eta}} \sum_j \mathcal{L}^{(j)}$
10: **end for**

Output: distilled data $\tilde{\mathbf{x}}$ and optimized learning rate $\tilde{\eta}$

Analysis

- origin loss

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

d-> N×D

- DD loss

$$\ell(\mathbf{x},\theta_0-\tilde{\eta}\nabla_{\theta_0}\ell(\tilde{\mathbf{x}},\theta_0))$$

$\tilde{d}$ ->M×D

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

- $\mathbf{d}^T\mathbf{d}\theta^*=\mathbf{d}^T\mathbf{t}$ for global minimum

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

- Mild assumption: columns of d are independent (i.e. d^Td has full rank)

Satify for any θ_0 $\longrightarrow$ $\mathbf{I}-\frac{\tilde{\eta}}{M}\tilde{\mathbf{d}}^T\tilde{\mathbf{d}}=\mathbf{0},$ $\longrightarrow$ $\tilde{\mathbf{d}}^T\tilde{\mathbf{d}}$ has full rank and $M\geq D$

any small number of distilled data fail to generalize to arbitrary initial θ_0

Methods

- Multiple GD steps and epochs

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 $\tilde{\eta}$

```
Initialize x̃ = {x̃_i}_{i=1}^M randomly, η̃ ← η̃_0
for each training step t = 1 to T do
    Get a minibatch of real training data x_t = {x_{t,j}}_{j=1}^n
    Sample a batch of initial weights θ_0^(j) ~ p(θ_0)
    for each sampled θ_0^(j) do
        Compute updated parameter with GD: θ_1^(j) = θ_0^(j) − η̃ ∇_{θ_0^(j)} ℓ(x̃, θ_0^(j))
        Evaluate the objective function on real training data: L^(j) = ℓ(x_t, θ_1^(j))
    end for
    Update x̃ ← x̃ − α ∇_x̃ Σ_j L^(j), and η̃ ← η̃ − α ∇_η̃ Σ_j L^(j)
end for
```

Output: distilled data $\tilde{\mathbf{x}}$ and optimized learning rate $\tilde{\eta}$

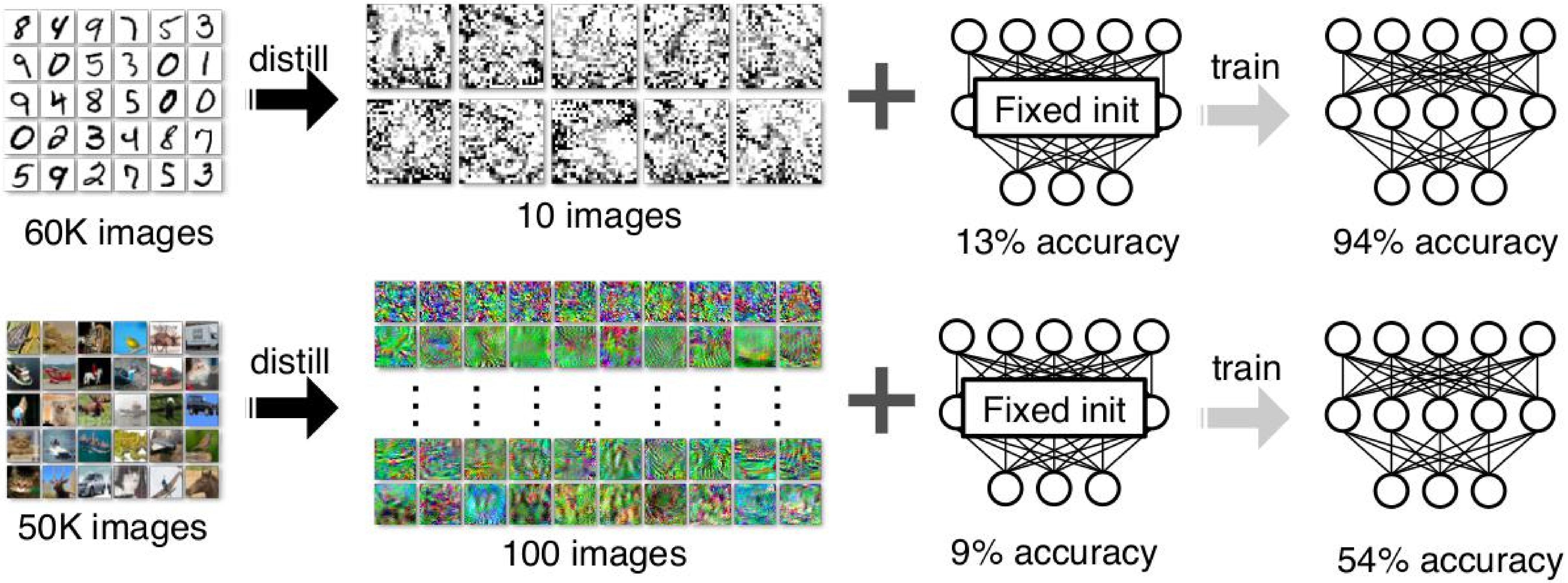
multiple epochs over the same sequence of distilled data

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

back propagate through all steps, using HVP to accelerate

Experiment

- Fixed initialization

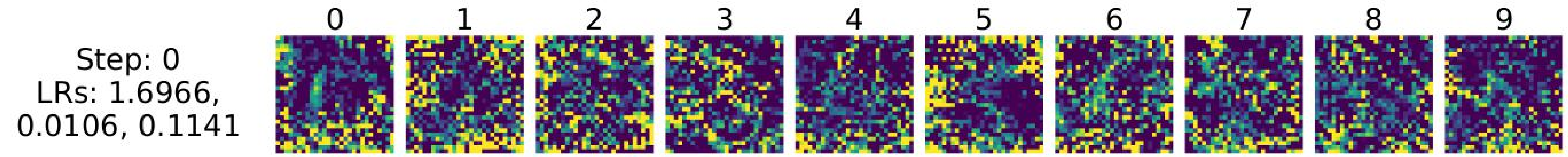


(a) Dataset distillation on MNIST and CIFAR10

Experiment

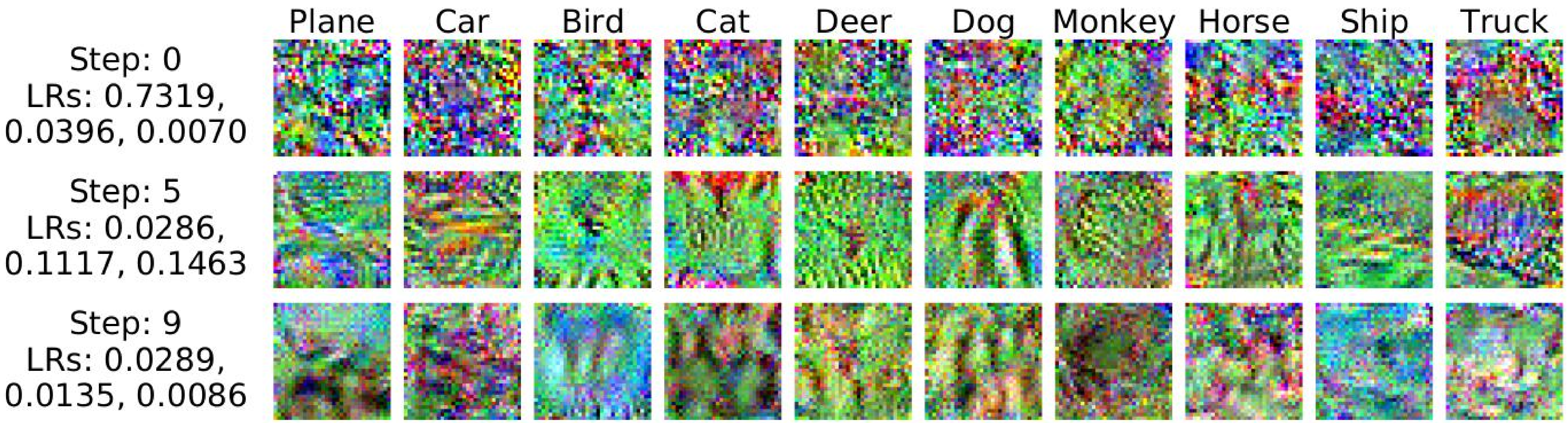
- Fixed initialization

1 GD step and
3 epochs
(10 imgs)



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

10 GD step
and 3 epochs
(100 imgs)



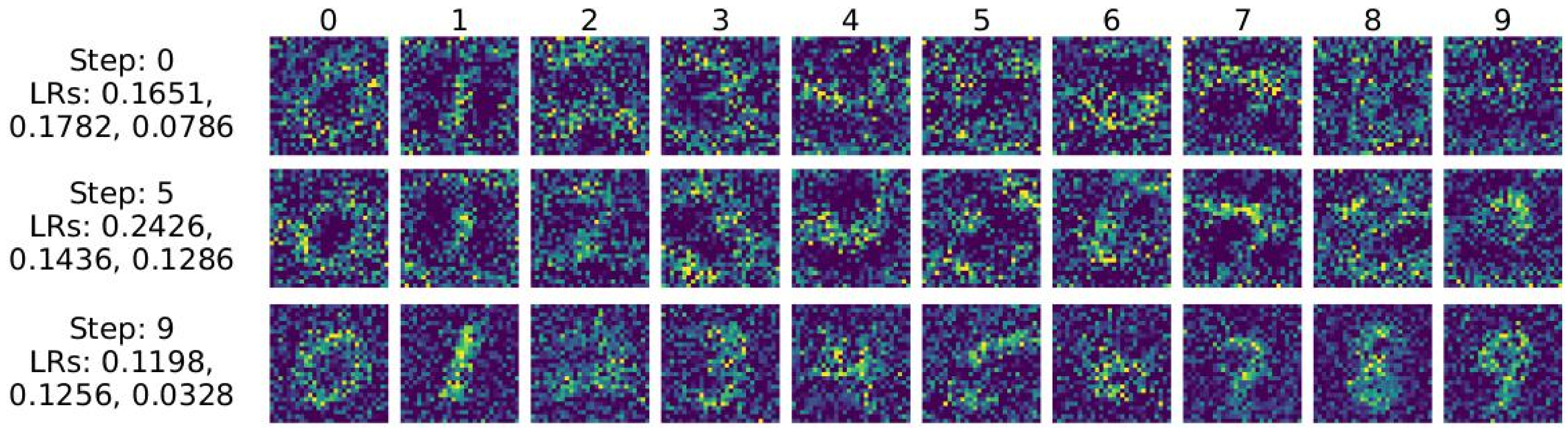
(b) CIFAR10. These distilled images train a fixed initialization from 8.82% test accuracy to 54.03%.

look like random noise as it encodes x and θ_0

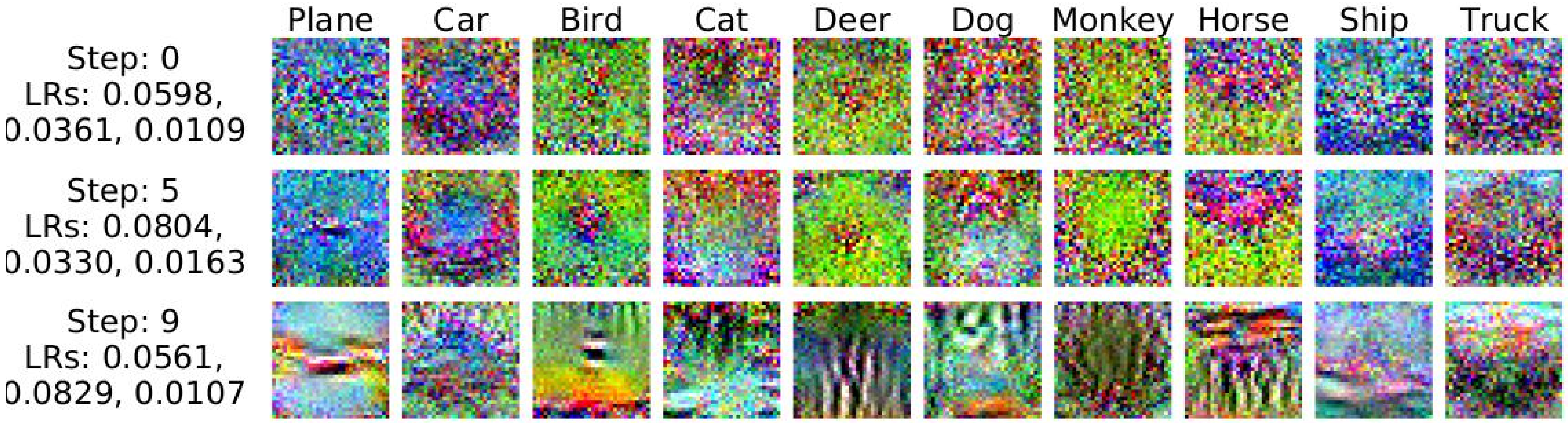
Experiment

- Random initialization (Xavier Initialization)

10 GD step and 3 epochs (100 imgs)



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



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

each step containing one image per category.

Images used in early steps tend to look noisier

Experiment

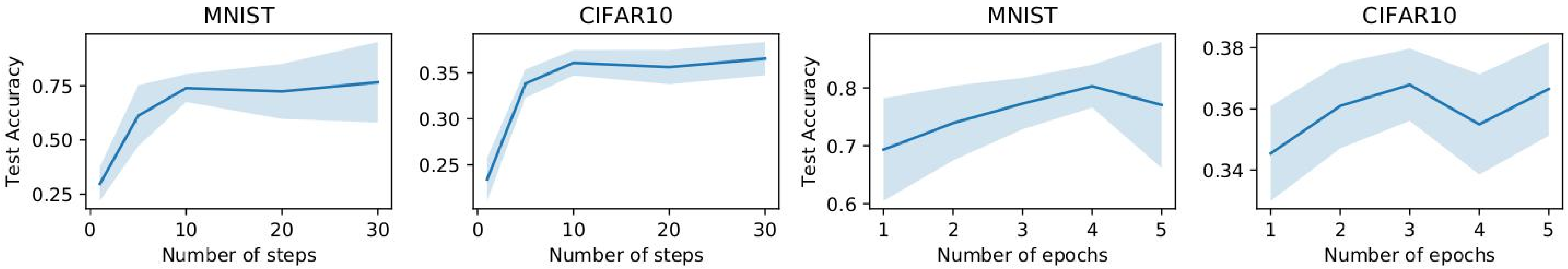
- Compare with baseline (Lenet)

		Ours		Baselines					
		Fixed init.	Random init.	Used as training data in same number of GD steps				Used in K-NN classification	
				Random real	Optimized real	k-means	Average real	Random real	k-means
99%	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	$\mathbf{92.2 \pm 0.1}$
80%	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

- Random real images
- Optimized real images -> top 20% from random sets
- k-means -> use cluster centroids for each category
- Average real images -> average image
- All methods use ten images per category (100 in total) except average real

Experiment

- Hyper-parameter sensitivity studies



(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.

Experiment

- Few-shot domain adaptation

	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
$\mathcal{M} \rightarrow \mathcal{U}$	$\mathbf{97.9}$	$\mathbf{95.4 \pm 1.8}$	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
$\mathcal{U} \rightarrow \mathcal{M}$	$\mathbf{93.2}$	$\mathbf{92.7 \pm 1.4}$	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
$\mathcal{S} \rightarrow \mathcal{M}$	$\mathbf{96.2}$	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.

DATASET CONDENSATION WITH GRADIENT MATCHING

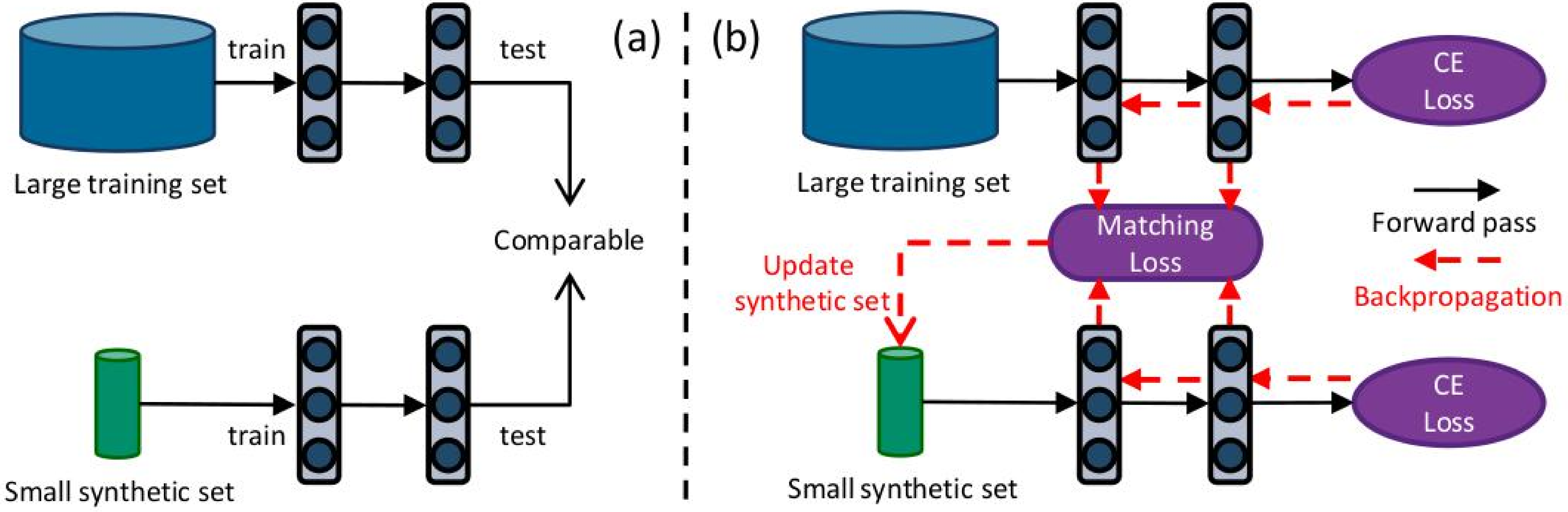
Bo Zhao, Konda Reddy Mopuri, Hakan Bilen
School of Informatics, The University of Edinburgh
{bo.zhao, kmopuri, hbilen}@ed.ac.uk

ICLR 2021

Motivation

- Coreset construction
  - continual learning, active learning ...
  - compactness, diversity, forgetfulness ...
  - not guarantee for downstream tasks and presence of representative samples
- Datset Distillation
  - synthesize informative samples by optimization

Methods



learning a synthetic set such that a deep network trained on it and the large set produces similar gradients w.r.t. its weights.

Methods

$$\boldsymbol{\theta}^{\mathcal{T}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{T}}(\boldsymbol{\theta})$$

$$\mathcal{L}^{\mathcal{T}}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{T}|}\sum_{(\boldsymbol{x},y)\in\mathcal{T}} \ell(\phi_{\boldsymbol{\theta}}(\boldsymbol{x}), y)$$

$$|\mathcal{S}| \ll |\mathcal{T}|$$

$$\boldsymbol{\theta}^{\mathcal{S}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta})$$

$$\mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{S}|}\sum_{(\boldsymbol{s},y)\in\mathcal{S}} \ell(\phi_{\boldsymbol{\theta}}(\boldsymbol{s}), y)$$

- Discussion

$$\mathcal{S}^* = \arg\min_{\mathcal{S}} \mathcal{L}^{\mathcal{T}}(\boldsymbol{\theta}^{\mathcal{S}}(\mathcal{S})) \qquad \text{subject to} \qquad \boldsymbol{\theta}^{\mathcal{S}}(\mathcal{S}) = \arg\min_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}).$$

  - nested loop optimization
  - unrolling the recursive computation graph over multiple steps

Methods

- Optimization
  - comparable performance && similar in parameter space $\boldsymbol{\theta}^{\mathcal{S}} \approx \boldsymbol{\theta}^{\mathcal{T}}$
  - assume similar weights imply similar performance

$$\min_{\mathcal{S}} D(\boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\theta}^{\mathcal{T}}) \quad \text{subject to} \quad \boldsymbol{\theta}^{\mathcal{S}}(\mathcal{S}) = \arg\min_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta})$$

  - $\boldsymbol{\theta}^{\mathcal{T}}$ depends on $\boldsymbol{\theta}_0$

$$\min_{\mathcal{S}} \mathrm{E}_{\boldsymbol{\theta}_0 \sim P_{\boldsymbol{\theta}_0}}[D(\boldsymbol{\theta}^{\mathcal{S}}(\boldsymbol{\theta}_0), \boldsymbol{\theta}^{\mathcal{T}}(\boldsymbol{\theta}_0))] \quad \text{subject to} \quad \boldsymbol{\theta}^{\mathcal{S}}(\mathcal{S}) = \arg\min_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}(\boldsymbol{\theta}_0))$$

  - back-optimization approach (fixed steps (ς))

$$\boldsymbol{\theta}^{\mathcal{S}}(\mathcal{S}) = \texttt{opt-alg}_{\boldsymbol{\theta}}(\mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}), \varsigma)$$

distance can be large and fixed steps may be not enough

Methods

- Curriculum based approach
  - follow a similar path throughout the optimization

$$\min_{\mathcal{S}} \mathrm{E}_{\boldsymbol{\theta}_0 \sim P_{\boldsymbol{\theta}_0}} [\sum_{t=0}^{T-1} D(\boldsymbol{\theta}_t^{\mathcal{S}}, \boldsymbol{\theta}_t^{\mathcal{T}})] \quad \text{subject to}$$

$$\boldsymbol{\theta}_{t+1}^{\mathcal{S}}(\mathcal{S}) = \texttt{opt-alg}_{\boldsymbol{\theta}}(\mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}_t^{\mathcal{S}}), \varsigma^{\mathcal{S}}) \quad \text{and} \quad \boldsymbol{\theta}_{t+1}^{\mathcal{T}} = \texttt{opt-alg}_{\boldsymbol{\theta}}(\mathcal{L}^{\mathcal{T}}(\boldsymbol{\theta}_t^{\mathcal{T}}), \varsigma^{\mathcal{T}})$$

  - In preliminary experiments $D(\boldsymbol{\theta}_t^{\mathcal{S}}, \boldsymbol{\theta}_t^{\mathcal{T}})$ -> 0

$$\boldsymbol{\theta}_{t+1}^{\mathcal{S}} \leftarrow \boldsymbol{\theta}_t^{\mathcal{S}} - \eta_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}_t^{\mathcal{S}}) \quad \text{and} \quad \boldsymbol{\theta}_{t+1}^{\mathcal{T}} \leftarrow \boldsymbol{\theta}_t^{\mathcal{T}} - \eta_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{T}}(\boldsymbol{\theta}_t^{\mathcal{T}}),$$

$$\min_{\mathcal{S}} \mathrm{E}_{\boldsymbol{\theta}_0 \sim P_{\boldsymbol{\theta}_0}} [\sum_{t=0}^{T-1} D(\nabla_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}_t), \nabla_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{T}}(\boldsymbol{\theta}_t))].$$

does not require the expensive unrolling

Methods

Algorithm 1: Dataset condensation with gradient matching

Input: Training set $\mathcal{T}$

Required: Randomly initialized set of synthetic samples $\mathcal{S}$ for C classes, probability distribution over randomly initialized weights $P_{\boldsymbol{\theta}_0}$, deep neural network $\phi_{\boldsymbol{\theta}}$, number of outer-loop steps K, number of inner-loop steps T, number of steps for updating weights $\varsigma_{\boldsymbol{\theta}}$ and synthetic samples $\varsigma_{\mathcal{S}}$ in each inner-loop step respectively, learning rates for updating weights $\eta_{\boldsymbol{\theta}}$ and synthetic samples $\eta_{\mathcal{S}}$.

for $k = 0, \cdots, K-1$ **do** Loop for initialization

  Initialize $\boldsymbol{\theta}_0 \sim P_{\boldsymbol{\theta}_0}$

  for $t = 0, \cdots, T-1$ **do** steps

    for $c = 0, \cdots, C-1$ **do** class

      Sample a minibatch pair $B_c^{\mathcal{T}} \sim \mathcal{T}$ and $B_c^{\mathcal{S}} \sim \mathcal{S}$ ▷ $B_c^{\mathcal{T}}$ and $B_c^{\mathcal{S}}$ are of the same class c.

      Compute $\mathcal{L}_c^{\mathcal{T}} = \frac{1}{|B_c^{\mathcal{T}}|} \sum_{(\boldsymbol{x},y)\in B_c^{\mathcal{T}}} \ell(\phi_{\boldsymbol{\theta}_t}(\boldsymbol{x}), y)$ and $\mathcal{L}_c^{\mathcal{S}} = \frac{1}{|B_c^{\mathcal{S}}|} \sum_{(\boldsymbol{s},y)\in B_c^{\mathcal{S}}} \ell(\phi_{\boldsymbol{\theta}_t}(\boldsymbol{s}), y)$

      Update $\mathcal{S}_c \leftarrow \texttt{opt-alg}_{\mathcal{S}}(D(\nabla_{\boldsymbol{\theta}}\mathcal{L}_c^{\mathcal{S}}(\boldsymbol{\theta}_t), \nabla_{\boldsymbol{\theta}}\mathcal{L}_c^{\mathcal{T}}(\boldsymbol{\theta}_t)), \varsigma_{\mathcal{S}}, \eta_{\mathcal{S}})$

    Update $\boldsymbol{\theta}_{t+1} \leftarrow \texttt{opt-alg}_{\boldsymbol{\theta}}(\mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}_t), \varsigma_{\boldsymbol{\theta}}, \eta_{\boldsymbol{\theta}})$ ▷ Use the whole $\mathcal{S}$

Output: $\mathcal{S}$

Methods

- Matching loss D(·, ·)
  - different layers -> different parameter shape
  - MLP: 2D (out×in), CNN: 4D (out×in×h×w)

$$D(\nabla_{\boldsymbol{\theta}}\mathcal{L}^{\mathcal{S}}, \nabla_{\boldsymbol{\theta}}\mathcal{L}^{\mathcal{T}}) = \sum_{l=1}^{L} d(\nabla_{\boldsymbol{\theta}^{(l)}}\mathcal{L}^{\mathcal{S}}, \nabla_{\boldsymbol{\theta}^{(l)}}\mathcal{L}^{\mathcal{T}})$$

$$d(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^{\text{out}} \left(1 - \frac{\mathbf{A_{i\cdot}} \cdot \mathbf{B_{i\cdot}}}{\|\mathbf{A_{i\cdot}}\| \|\mathbf{B_{i\cdot}}\|}\right)$$

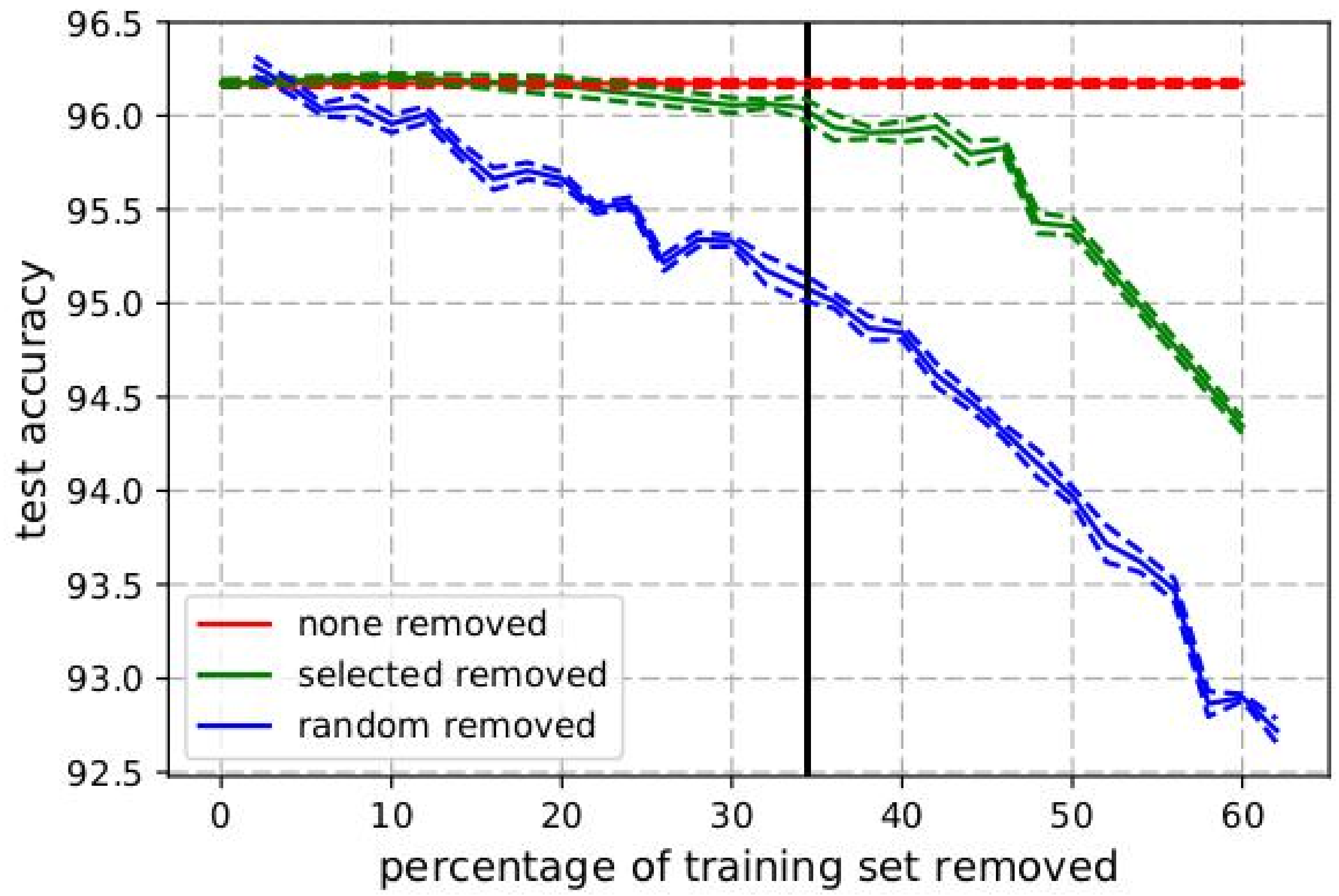
cosine similarity

$\mathbf{A}_{i\cdot}$ and $\mathbf{B}_{i\cdot}$ are flattened gradients corresponding to each output node i

Experiment

- Baseline
  - random, DD
  - Herding baseline-> nearest to cluster center
  - K-Center-> multiple center points
  - Forgetting method [1]

the unforgettable examples contain less information



[1] An Empirical Study of Example Forgetting during Deep Neural Network Learning, Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, Geoffrey J. Gordon, ICLR 2019.

Experiment

	Img/Cls	Ratio %	Coreset Selection Random	Herding	K-Center	Forgetting	Ours	Whole Dataset
MNIST	1	0.017	64.9±3.5	89.2±1.6	89.3±1.5	35.5±5.6	**91.7±0.5**	99.6±0.0
	10	0.17	95.1±0.9	93.7±0.3	84.4±1.7	68.1±3.3	**97.4±0.2**	
	50	0.83	97.9±0.2	94.9±0.2	97.4±0.3	88.2±1.2	**98.8±0.2**	
FashionMNIST	1	0.017	51.4±3.8	67.0±1.9	66.9±1.8	42.0±5.5	**70.5±0.6**	93.5±0.1
	10	0.17	73.8±0.7	71.1±0.7	54.7±1.5	53.9±2.0	**82.3±0.4**	
	50	0.83	82.5±0.7	71.9±0.8	68.3±0.8	55.0±1.1	**83.6±0.4**	
SVHN	1	0.014	14.6±1.6	20.9±1.3	21.0±1.5	12.1±1.7	**31.2±1.4**	95.4±0.1
	10	0.14	35.1±4.1	50.5±3.3	14.0±1.3	16.8±1.2	**76.1±0.6**	
	50	0.7	70.9±0.9	72.6±0.8	20.1±1.4	27.2±1.5	**82.3±0.3**	
CIFAR10	1	0.02	14.4±2.0	21.5±1.2	21.5±1.3	13.5±1.2	**28.3±0.5**	84.8±0.1
	10	0.2	26.0±1.2	31.6±0.7	14.7±0.9	23.3±1.0	**44.9±0.5**	
	50	1	43.4±1.0	40.4±0.6	27.0±1.4	23.3±1.1	**53.9±0.5**	

ConvNet is used for training and testing, commonly used in fewshot learning

Experiment

- Comapre with DD

Dataset	Img/Cls	DD	Ours	Whole Dataset
MNIST	1	-	85.0±1.6	99.5±0.0
	10	79.5±8.1	**93.9±0.6**	
CIFAR10	1	-	24.2±0.9	83.1±0.2
	10	36.8±1.2	**39.1±1.2**	

Table 3: Comparison to DD (Wang et al., 2018) in terms of testing accuracy (%).

2 times faster and more consistent results over multiple runs

Experiment

- Visualization

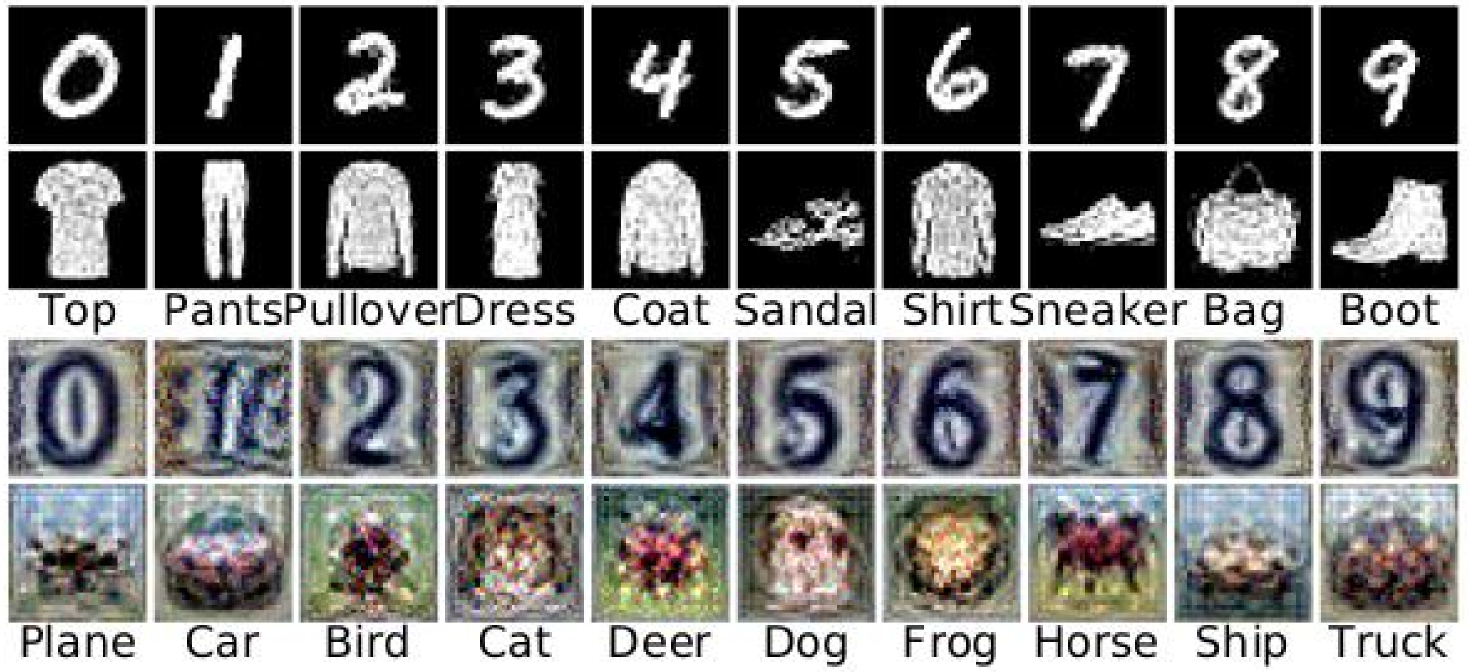


Figure 2: Visualization of condensed 1 image/class with ConvNet for MNIST, Fashion-MNIST, SVHN and CIFAR10.

- Cross-architecture performance

C\T	MLP	ConvNet	LeNet	AlexNet	VGG	ResNet
MLP	70.5±1.2	63.9±6.5	77.3±5.8	70.9±11.6	53.2±7.0	80.9±3.6
ConvNet	69.6±1.6	**91.7±0.5**	85.3±1.8	85.1±3.0	**83.4±1.8**	**90.0±0.8**
LeNet	71.0±1.6	90.3±1.2	85.0±1.7	84.7±2.4	80.3±2.7	89.0±0.8
AlexNet	72.1±1.7	87.5±1.6	84.0±2.8	82.7±2.9	81.2±3.0	88.9±1.1
VGG	70.3±1.6	90.1±0.7	83.9±2.7	83.4±3.7	81.7±2.6	89.1±0.9
ResNet	**73.6±1.2**	91.6±0.5	**86.4±1.5**	**85.4±1.9**	**83.4±2.4**	89.4±0.9

Table 2: Cross-architecture performance in testing accuracy (%) for condensed 1 image/class in MNIST.

mlp generated pics performs badly for cnn, while cnn generated pics perform well for other mlp

Experiment

- Continual Learning

testing accuracies are computed by averaging performance after each stage (SVHN→MNIST→USPS)

10 images/class
KD: regularize output vs previous

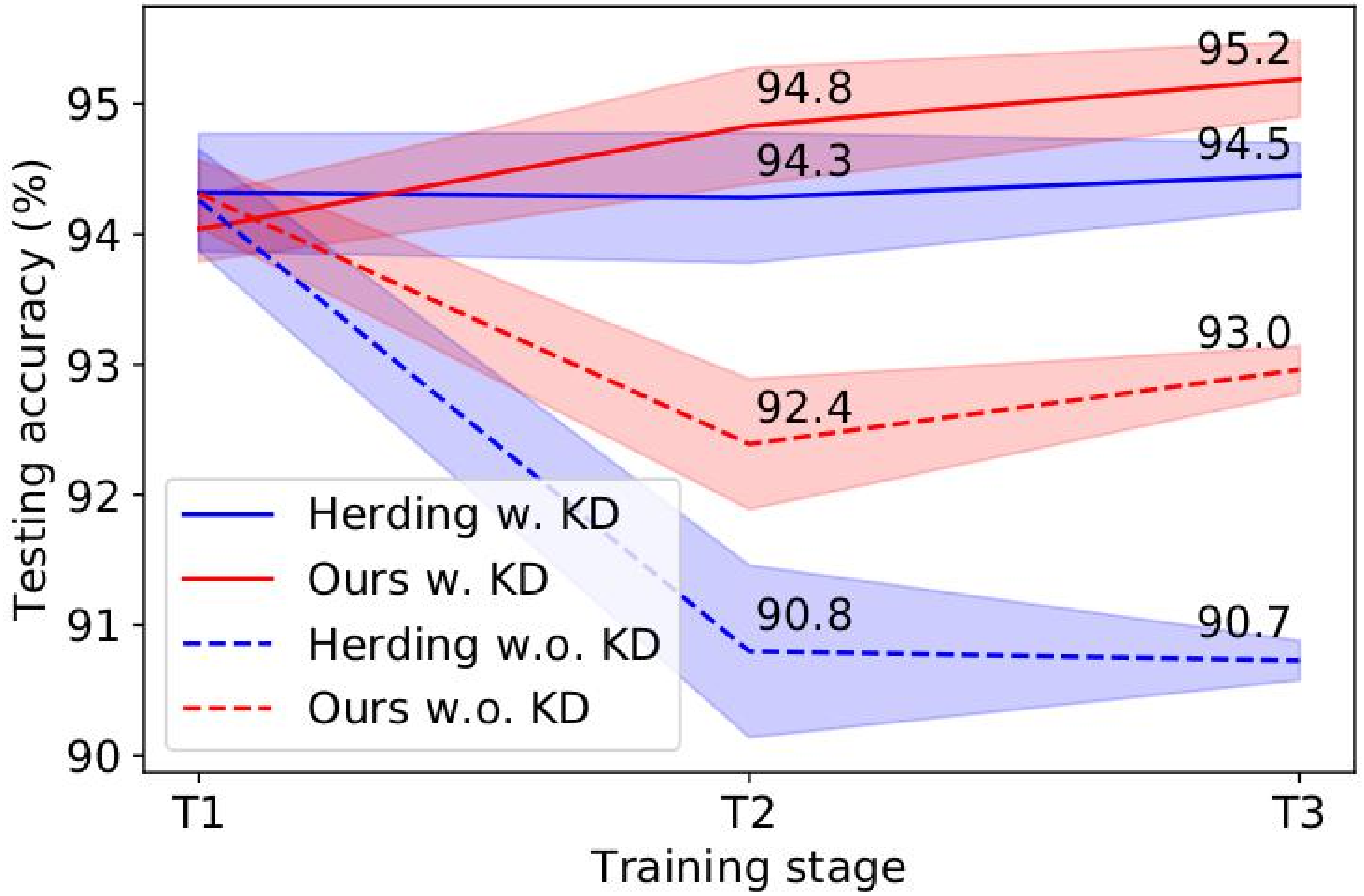


Figure 4: Continual learning performance in accuracy (%). Herding denotes the original E2E (Castro et al., 2018). T1, T2, T3 are three learning stages. The performance at each stage is the mean testing accuracy on all learned tasks.