# An Introduction to Dataset Distillation and its Application

Jiyuan Shen

# Motivation

Compressed dataset can help model interpretability and efficient learning.

- How neural network compress and memorize large-scale dataset?
  - Project original data to feature maps.

$$\boldsymbol{x} \in \mathbb{R}^D \xrightarrow{f(\boldsymbol{x})} \boldsymbol{z} \in \mathbb{R}^d$$

  - Can be viewed as an implicit compression process.
- Whether can we explicitly re-construct the low-dimensional information from the neural network?
  - Not re-construct to equal size, but distill informative and representative data.

$$\boldsymbol{x} \in \mathbb{R}^D \xrightarrow{f(\boldsymbol{x})} \boldsymbol{z} \in \mathbb{R}^d \xrightarrow{g(\boldsymbol{z})} \widehat{\boldsymbol{x}} \in \mathbb{R}^D$$

  - Also offer a perspective of model interpretability.

- Do datasets have to be big?
- Can the information from a large dataset be distilled and integrated?
- Can training on “small data” be equally successful?

---

- By using the compressed dataset, model training, parameters tuning and testing process become easy-approaching.
- Under the data-centric AI, data-efficient learning is important.

# Definition: Dataset distillation (DD)

Synthesize a distilled dataset which performance is comparable to the real one.

- Dataset distillation is the task of **synthesizing a small dataset** such that models trained on it achieve high performance on the original large dataset. Condensed dataset can accelerate network training and reducing data storage.
- A good small distilled dataset is **not only useful in dataset understanding**, but has various applications (**e.g., continual learning, privacy, federated learning, neural architecture search, etc.**).
- *A dataset distillation algorithm takes as input a large real dataset to be distilled (training set), and outputs a small synthetic distilled dataset, which is evaluated via testing models trained on this distilled dataset on a separate real dataset (validation/test set).*
- This task was first introduced in the 2018 paper Dataset Distillation, along with a proposed algorithm using backpropagation through optimization steps. In recent years (2019-now), dataset distillation has gained increasing attention in the research community, across many institutes and labs.

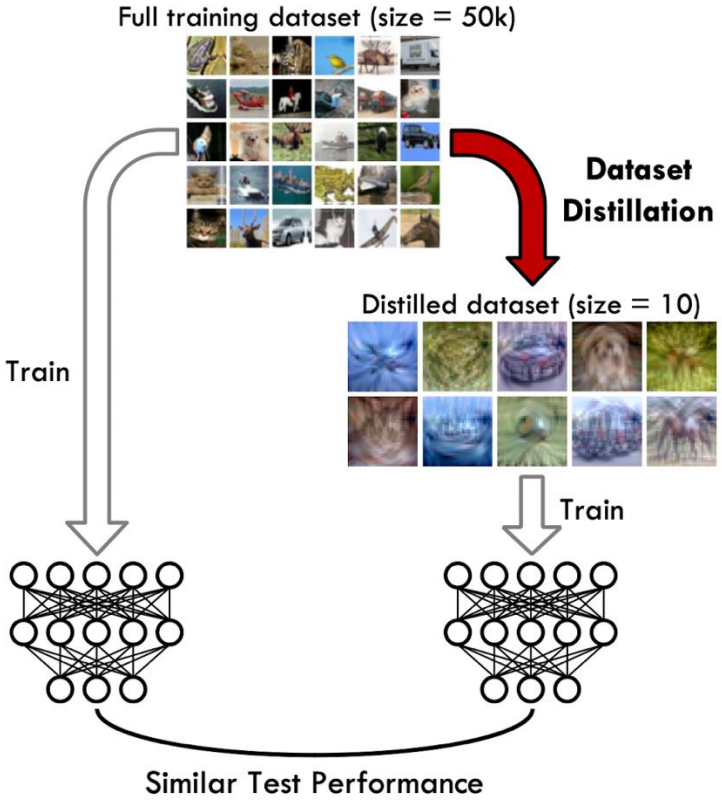



Figure 2. Dataset distillation aims to generate a small synthetic dataset for which a model trained on it can achieve a similar test performance as a model trained on the whole real train set.

Wang, Tongzhou, et al. "Dataset distillation." *arXiv preprint arXiv:1811.10959* (2018).

# Difference between Model-Agnostic Meta-Learning (MAML)

DD try to optimize the input space.

**Algorithm 1** Dataset Distillation

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

**Algorithm 1** Model-Agnostic Meta-Learning

**Require:** $p(\mathcal{T})$: distribution over tasks
**Require:** $\alpha$, $\beta$: step size hyperparameters
randomly initialize $\theta$
**while** not done **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  **for all** $\mathcal{T}_i$ **do**
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to $K$ examples
    Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
  **end for**
  Update $\theta \leftarrow \theta - \beta\nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
**end while**

- Both algorithms look similar. They all select some samples and average on the inner loop's results to update the optimized objective. However, DD and MAML choose different optimized variation. DD chooses input dataset and learning rate while MAML chooses model weight.
- Outer-loop optimization using Truncated Back-Propagation Through Time (TBPTT), i.e., unroll a limited number of inner-loop optimization steps while optimizing the outer-loop.

Wang, Tongzhou, et al. "Dataset distillation." *arXiv preprint arXiv:1811.10959* (2018).

# Dataset Condensation with Gradient Matching (ICLR 21)

## Gradient Matching Surrogate Objective: solve the unstable and cross-architecture problem.

**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 $P_{\theta_0}$, 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**

  Initialize $\boldsymbol{\theta}_0 \sim P_{\theta_0}$

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

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

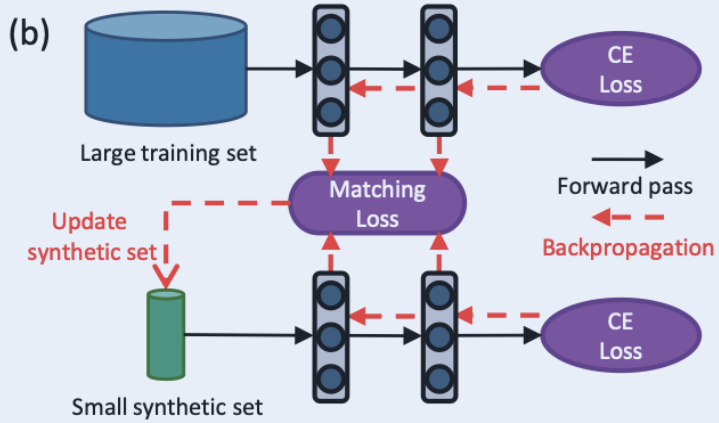
      Sample a minibatch pair $B_c^{\mathcal{T}} \sim \mathcal{T}$ and $B_c^{\mathcal{S}} \sim \mathcal{S}$ ▷ $B_c^{\mathcal{T}}$ and $B_c^{\mathcal{S}}$ are of the same class $c$.

      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:** $\mathcal{S}$




DD

**Algorithm 1** Improved Deep Leakage from Gradients (iDLG)

**Require:**

$F(\mathbf{x}; \mathbf{W})$: Differentiable learning model, $\mathbf{W}$: Model parameters, $\nabla\mathbf{W}$: Gradients produced by private training datum $(\mathbf{x}, c)$, $N$: maximum number of iterations. $\eta$: learning rate.

**Ensure:**

$(\mathbf{x}', c')$: Dummy datum and label.

1: $c' \leftarrow i$ s.t. $\nabla\mathbf{W}_L^{i\,T} \cdot \nabla\mathbf{W}_L^j \leq 0, \quad \forall j \neq i$ ▷ Extract the ground-truth label.

2: $\mathbf{x}' \leftarrow \mathcal{N}(0, 1)$ ▷ Initialize the dummy datum.

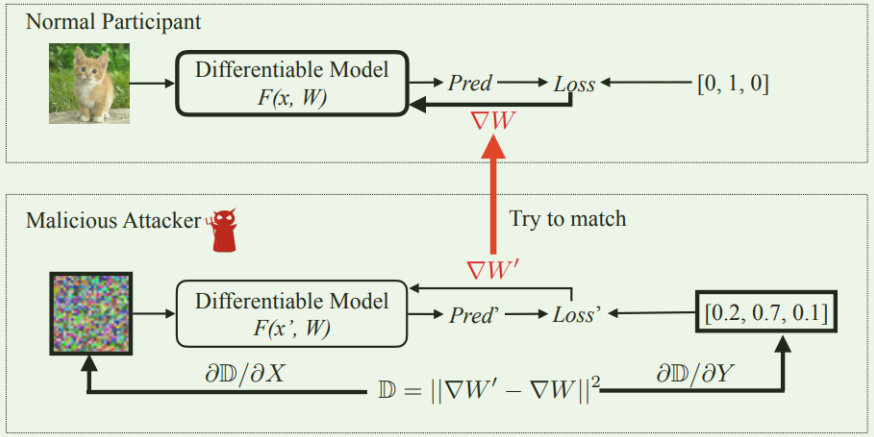
3: **for** $i \leftarrow 1$ to $N$ **do**

4: $\quad \nabla\mathbf{W}' \leftarrow \partial l(F(\mathbf{x}'; \mathbf{W}), c')/\partial\mathbf{W}$ ▷ Calculate the dummy gradients.

5: $\quad L_G = \|\nabla\mathbf{W}' - \nabla\mathbf{W}\|_F^2$ ▷ Calculate the loss (difference between gradients).

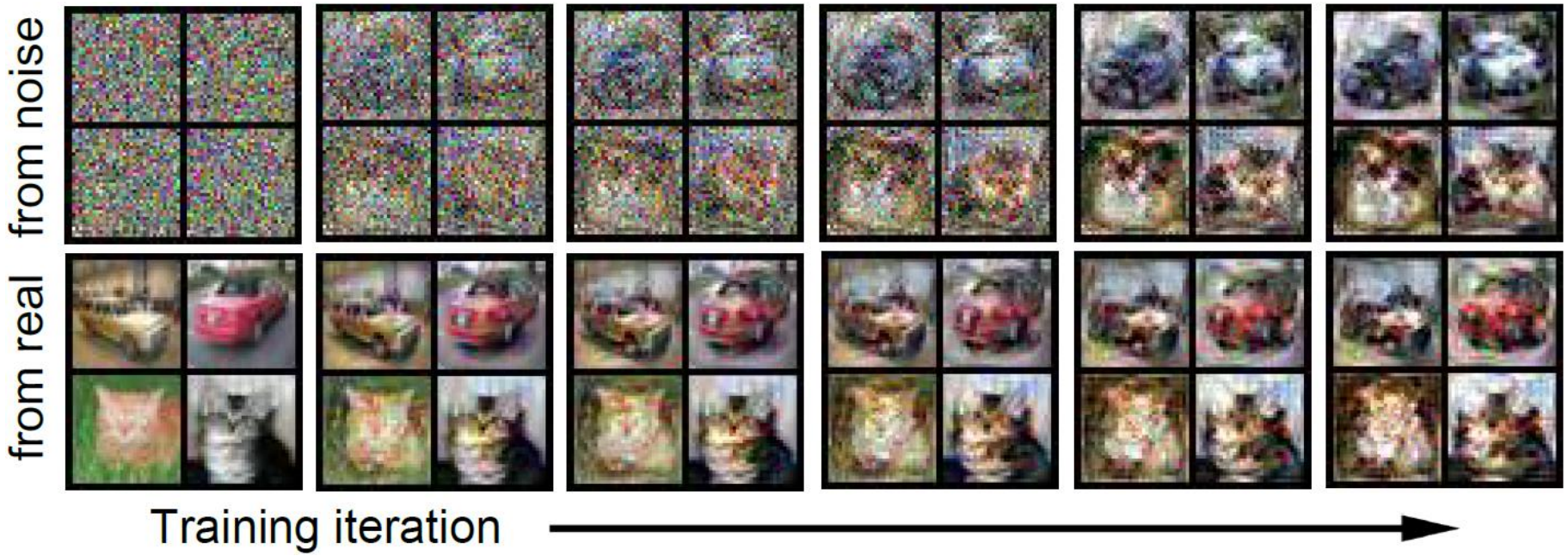
6: $\quad \mathbf{x}' \leftarrow \mathbf{x}' - \eta\nabla_{\mathbf{x}'} L_G$ ▷ Update the dummy datum.

7: **end for**




Deep Leakage from Gradient

Using $d(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^{\text{out}} \left(1 - \frac{\mathbf{A_i} \cdot \mathbf{B_i}}{\|\mathbf{A_i}\| \|\mathbf{B_i}\|}\right)$ to update the parameters of network that synthesize distilled dataset.



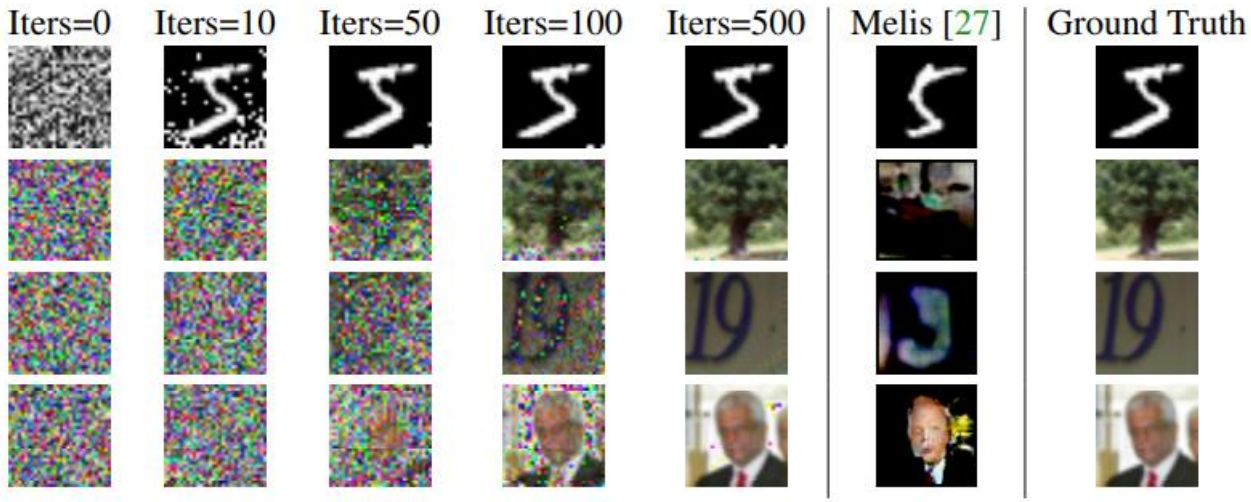
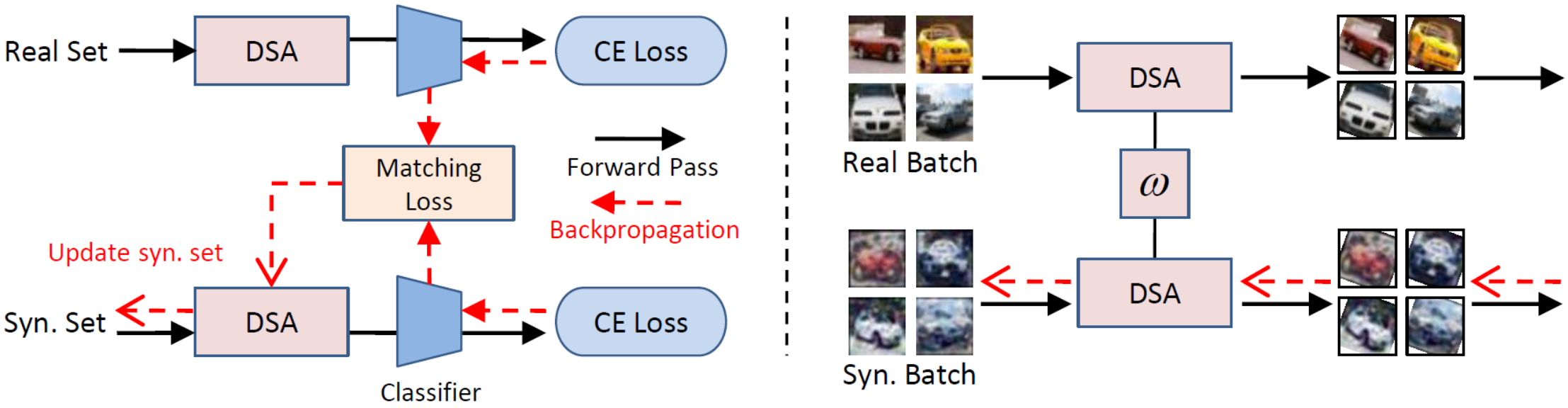




Figure 3: The visualization showing the deep leakage on images from MNIST [22], CIFAR-100 [21], SVHN [28] and LFW [14] respectively. Our algorithm fully recovers the four images while previous work only succeeds on simple images with clean backgrounds.

# Dataset Condensation with Differentiable Siamese Augmentation (ICML 21)

Apply data augmentation to both real and synthetic dataset.



# Accelerating Dataset Distillation via Model Augmentation (CVPR 23)

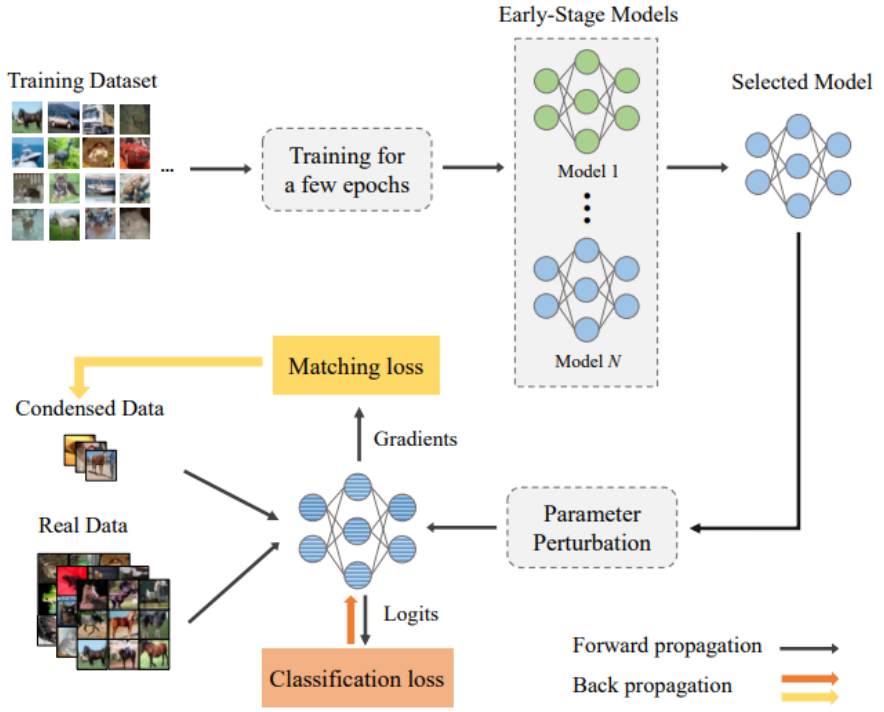



Figure 2. The illustration of our proposed fast dataset distillation method. We perform early-stage pretraining and parameter perturbation on models in dataset distillation.

**Algorithm 1:** Efficient Dataset Distillation

**Input:** Training data $\mathcal{T}$, loss function $l$, number of classes $C$, number of model $N$, magnitude $\alpha$, augmentation function $\mathcal{A}$, multi-information function $f$, deep neural network $\psi_\theta$ parameterized with $\theta$

**Output:** Condensed dataset $\mathcal{S}$

**Definition:** $D\left(B, B^{\prime}; \theta\right)=\left\|\nabla_\theta \ell(\theta ; B)-\nabla_\theta \ell\left(\theta ; B^{\prime}\right)\right\|$

`/* Early-Stage Pre-train */`

Randomly initialize $N$ networks $\{\tau_1 ... \tau_N\}$;
**for** $n \leftarrow 1$ **to** $N$ **do**
  Update network $\tau_n$ on real data $\mathcal{T}$:
  **for** $p \leftarrow 1$ **to** $P$ **do**
    $\tau_{n,p+1} \leftarrow \tau_{n,p}-\eta \nabla_{\tau_{n,p}} \ell\left(\tau_{n,p} ; \mathcal{A}(\mathcal{T})\right)$
  **end**
**end**
Initialize condensed dataset $\mathcal{S}$
**for** $t \leftarrow 0$ **to** $T$ **do**
  Randomly load one checkpoint from $\{\tau_1 ... \tau_N\}$ to initialize $\psi_\theta$ ;
  `/* Parameter Perturbation */`
  Sample vector $\mathbf{d}$ from Gaussian distribution
  Parameter perturbation on $\psi_\theta$: $\theta \leftarrow \theta+\alpha \cdot \mathbf{d}$
  **for** $m \leftarrow 0$ **to** $M$ **do**
    **for** $c \leftarrow 0$ **to** $C$ **do**
      Sample an intra-class mini-batch $T_c \sim \mathcal{T}, S_c \sim \mathcal{S}$
      Update synthetic data $\mathcal{S}_c$:
      $S_c \leftarrow S_c-\lambda \nabla_{S_c} D\left(\mathcal{A}\left(f\left(S_c\right)\right), \mathcal{A}\left(T_c\right)\right)$
    **end**
    Sample a mini-batch $T \sim \mathcal{T}$
    Update network $\psi_\theta$ w.r.t classification loss:
    $\theta_{m+1} \leftarrow \theta_m-\eta \nabla_\theta \ell\left(\theta_m ; \mathcal{A}(T)\right)$
  **end**
**end**

# Dataset Distillation with Distribution Matching (WACV 23)

## Distribution Matching Surrogate Objective: solve the intrinsic problem of gradient descending.

- Firstly, it samples mini-batch of real and synthetic data and then embeds them with the randomly sampled deep neural networks.
- The synthetic data is learned by minimizing the **distribution discrepancy** between real synthetic data in the sampled embedding spaces.
- Use $\min_{\mathcal{S}} E_{v\sim P_v} \left\| \frac{1}{|\mathcal{T}|}\sum_{i=1}^{|\mathcal{T}|}\psi_v(x_i) - \frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|}\psi_v(x_i)\right\|^2$ to match the distribution.

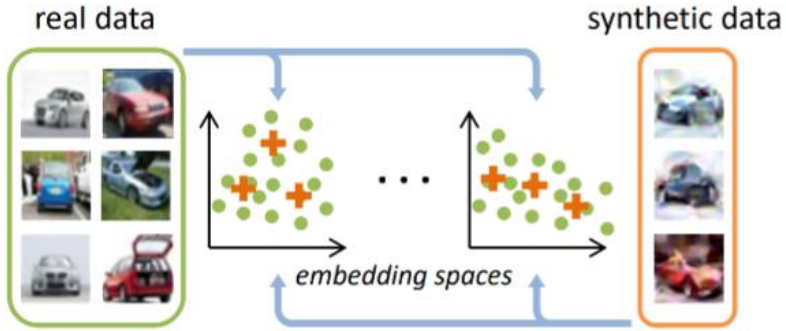



Figure 2. Dataset Condensation with Distribution Matching.

**Algorithm 1:** Dataset condensation with distribution matching

**Input:** Training set $\mathcal{T}$

**Required**: Randomly initialized set of synthetic samples $\mathcal{S}$ for $C$ classes, deep neural network $\psi_{\vartheta}$ parameterized with $\vartheta$, probability distribution over parameters $P_{\vartheta}$, differentiable augmentation $\mathcal{A}_{\omega}$ parameterized with $\omega$, augmentation parameter distribution $\Omega$, training iterations $K$, learning rate $\eta$.

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

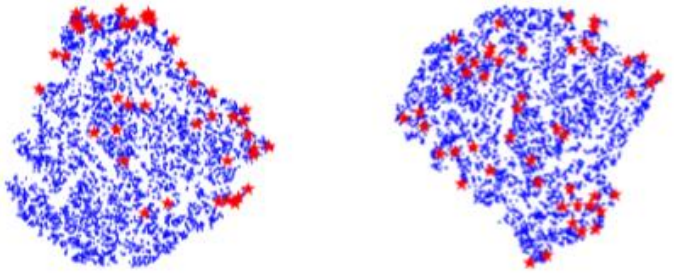
  Sample $\vartheta \sim P_{\vartheta}$

  Sample mini-batch pairs $B_c^{\mathcal{T}} \sim \mathcal{T}$ and $B_c^{\mathcal{S}} \sim \mathcal{S}$ and $\omega_c \sim \Omega$ for every class $c$

  Compute $\mathcal{L} = \sum_{c=0}^{C-1} \| \frac{1}{|B_c^{\mathcal{T}}|}\sum_{(\boldsymbol{x},y)\in B_c^{\mathcal{T}}} \psi_{\vartheta}(\mathcal{A}_{\omega_c}(\boldsymbol{x})) - \frac{1}{|B_c^{\mathcal{S}}|}\sum_{(\boldsymbol{s},y)\in B_c^{\mathcal{S}}} \psi_{\vartheta}(\mathcal{A}_{\omega_c}(\boldsymbol{s}))\|^2$

  Update $\mathcal{S} \leftarrow \mathcal{S} - \eta\nabla_{\mathcal{S}}\mathcal{L}$

**Output:** $\mathcal{S}$



(a) Gradient Matching (b) Distribution Matching

Figure 1. Synthetic dataset distribution with IPC 50.

# CAFE: Learning to Condense Dataset by Aligning Features (CVPR 22)

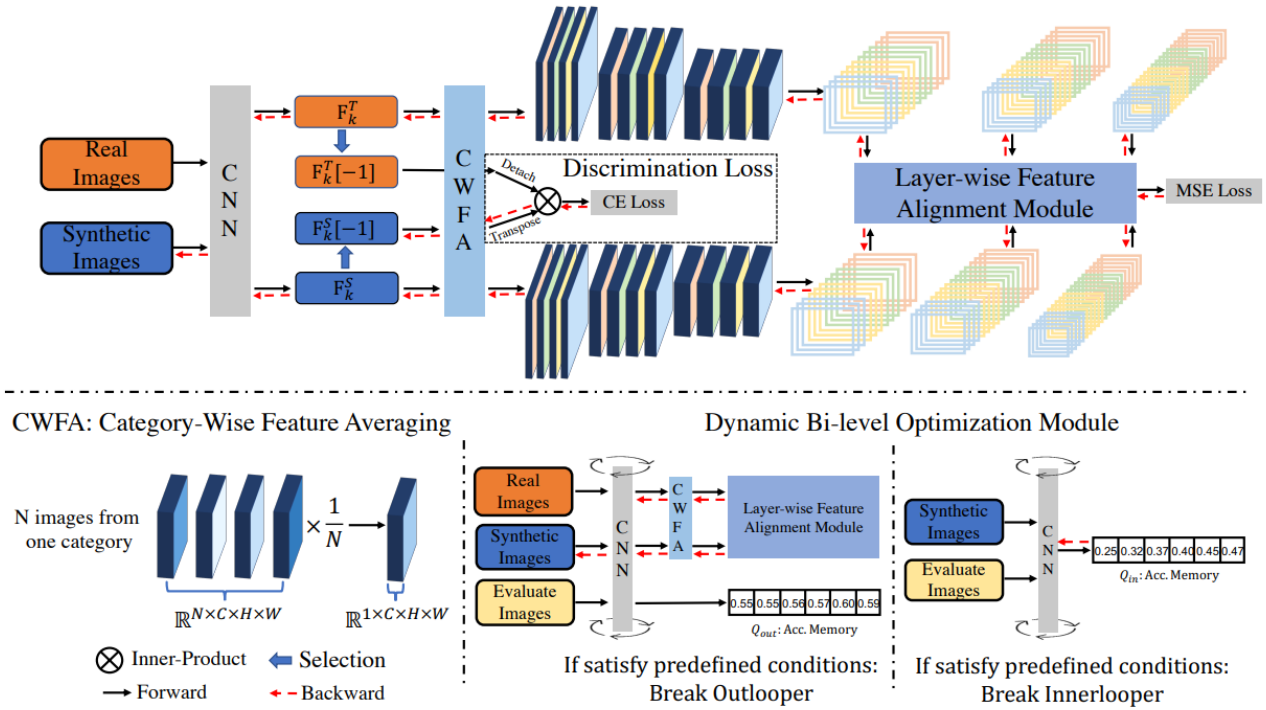
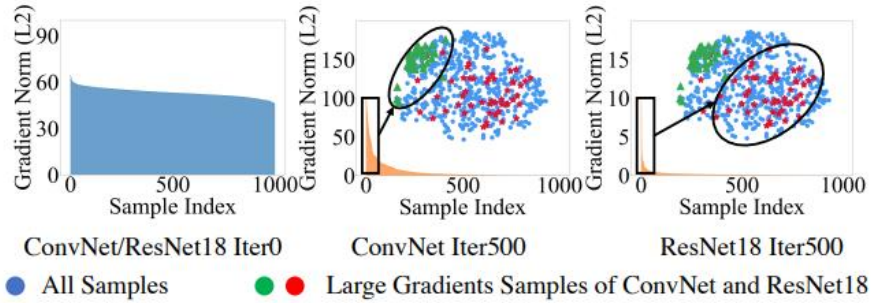
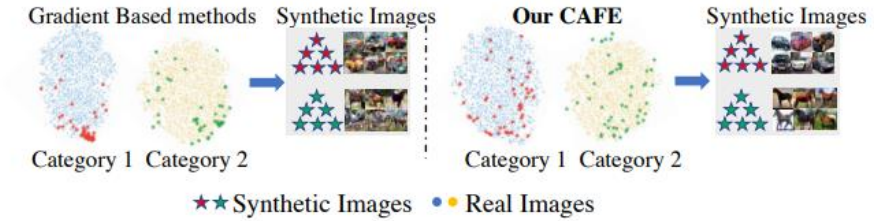



Figure 2: Illustration of the proposed CAFE method. The CAFE consists of a layer-wise feature alignment module to capture the accurate distribution of the original large-scale dataset, a discrimination loss for mining the discriminate samples from real dataset, and a dynamic bi-level optimization module to reduce the influence of under- and over-fitting on synthetic images.




(a) The gradient distribution changes from a uniform to long-tailed distribution during the training. Meanwhile, the overlap of large-gradient samples are small among different architectures.




(b) The visualization of synthetic images and their distributions generated by **gradient matching** and **CAFE**. ConvNet is used.

# DataDAM: Efficient Dataset Distillation with Attention Matching (ICCV 23)

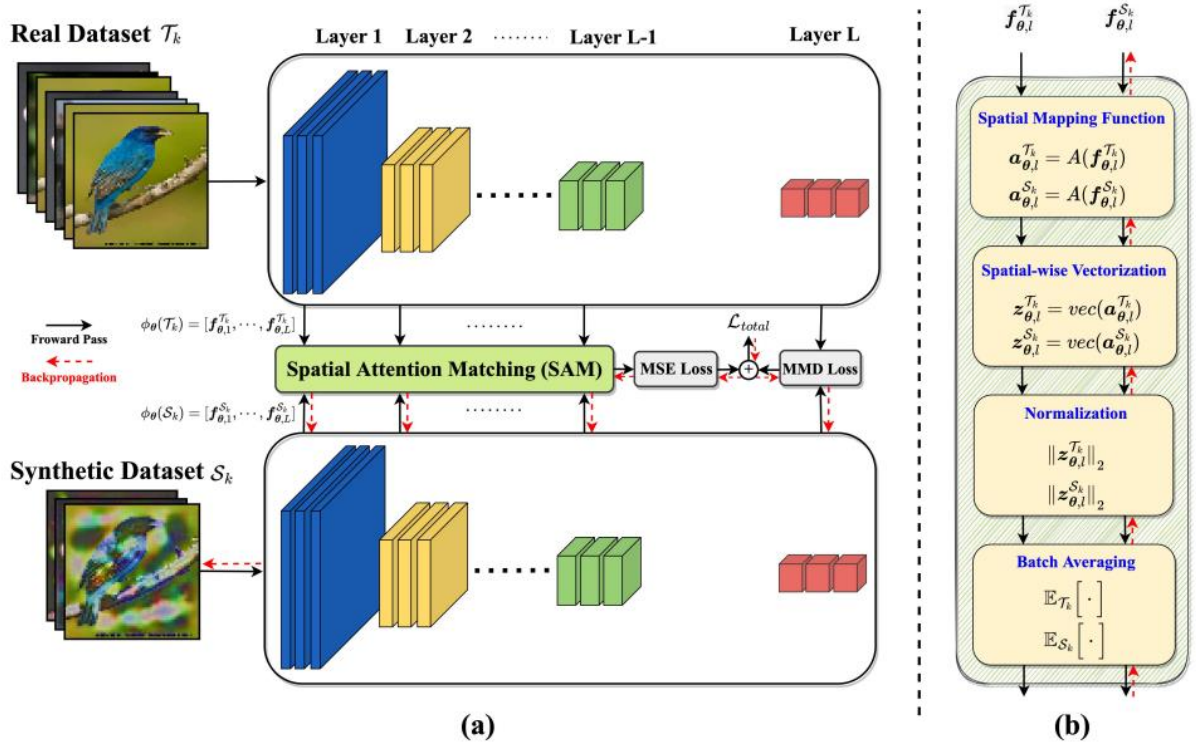
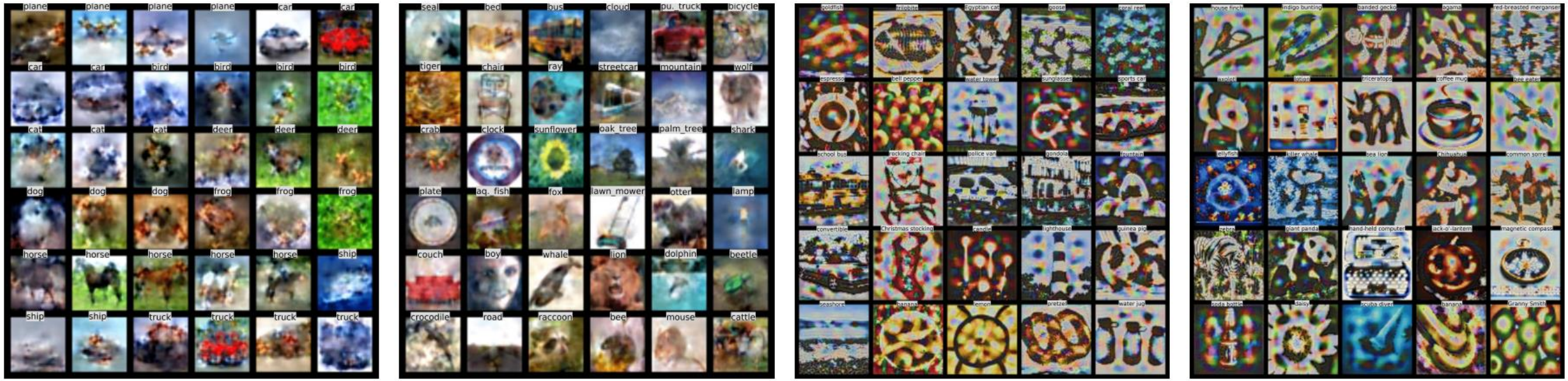



Figure 2: (a) Illustration of the proposed DataDAM method. DataDAM includes a Spatial Attention Matching (SAM) module to capture the dataset's distribution and a complementary loss for matching the feature distributions in the last layer of the encoder network. (b) The internal architecture of the SAM module.


(a) Iteration 0 | (b) Iteration 600 | (c) Iteration 2000 | (d) Iteration 5000 | (e) Iteration 15000 | (f) Iteration 25000 | (g) Iteration 35000 | (h) Iteration 40000

Figure 10: The learning process of all classes in the CIFAR10 dataset (IPC 50) initialized from Gaussian noise. We take two random images for each class and visualize their progression over the 40,000 training epochs.




(a) CIFAR10 (b) CIFAR100 (c) Tiny ImageNet (d) ImageNet-1K

Figure 6: Example distilled images from 32x32 CIFAR10/100 (IPC10), 64x64 Tiny ImageNet (IPC1), and 64x64 ImageNet-1K (IPC1).

# Dataset Distillation by Matching Training Trajectories (CVPR 22)

## Trajectory (network parameters) Matching Surrogate Objective.

## (i) Long-range multiple-steps matching. (ii) Explore large-dataset usage.

**Algorithm 1** Dataset Distillation via Trajectory Matching

**Input:** $\{\tau_i^*\}$: set of expert parameter trajectories trained on $\mathcal{D}_{\text{real}}$.
**Input:** $M$: # of updates between starting and target expert params.
**Input:** $N$: # of updates to student network per distillation step.
**Input:** $\mathcal{A}$: Differentiable augmentation function.
**Input:** $T^+ < T$: Maximum start epoch.

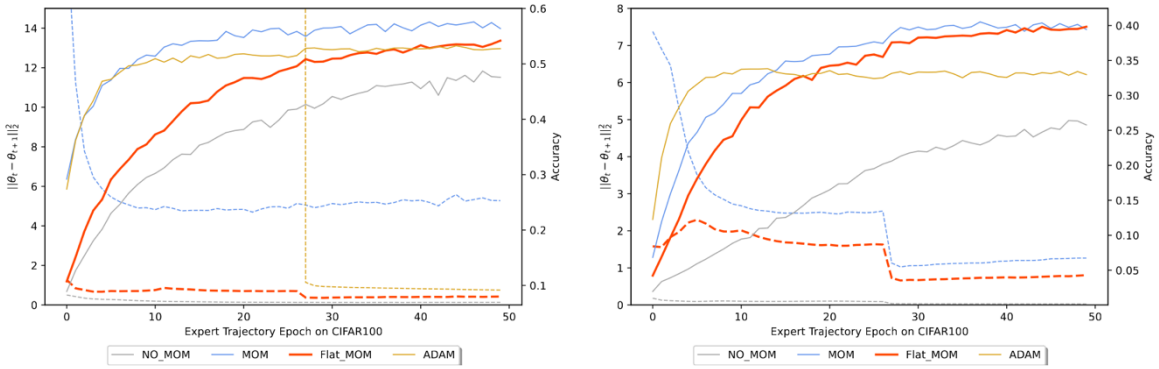
Initialize distilled data $\mathcal{D}_{\text{syn}} \sim \mathcal{D}_{\text{real}}$
Initialize trainable learning rate $\alpha := \alpha_0$ for apply $\mathcal{D}_{\text{syn}}$
**for each** distillation step... **do**
    ▷ Sample expert trajectory: $\tau^* \sim \{\tau_i^*\}$ with $\tau^* = \{\theta_t^*\}_0^T$
    ▷ Choose random start epoch, $t \leq T^+$
    ▷ Initialize student network with expert params:
        $\hat{\theta}_t := \theta_t^*$
    **for** $n = 0 \rightarrow N - 1$ **do**
        ▷ Sample a mini-batch of distilled images:
            $b_{t+n} \sim \mathcal{D}_{\text{syn}}$
        ▷ Update student network w.r.t. classification loss:
            $\hat{\theta}_{t+n+1} = \hat{\theta}_{t+n} - \alpha \nabla \ell(\mathcal{A}(b_{t+n}); \hat{\theta}_{t+n})$
    **end for**
    ▷ Compute loss between ending student and expert params:
        $\mathcal{L} = \|\hat{\theta}_{t+N} - \theta_{t+M}^*\|_2^2 \;/\; \|\theta_t^* - \theta_{t+M}^*\|_2^2$
    ▷ Update $\mathcal{D}_{\text{syn}}$ and $\alpha$ with respect to $\mathcal{L}$
**end for**

**Output:** distilled data $\mathcal{D}_{\text{syn}}$ and learning rate $\alpha$

Cazenavette G, Wang T, Torralba A, et al. Dataset distillation by matching training trajectories[C]

# AST: Effective Dataset Distillation through Alignment with Smooth and High-Quality Expert Trajectories

## We observe the dilemma between stronger expert and student alignment.




(a) CIFAR-100 (b) Tiny ImageNet

Fig. 9: $\|\theta_t - \theta_{t+1}\|_2^2$ refers to the change of the model weights on two consecutive iterations, shown by dash curve. Correspondingly, the solid curve refers to the metric of evaluation accuracy. NO_MOM refers to SGD without momentum. MOM means using SGD with momentum alone. Flat_MOM denotes smooth expert trajectories that apply gradient penalty and clipped loss under the usage of SGD with momentum. ADAM means using ADAM as an optimizer alone. We view the red curve as a much smoother and higher-quality expert trajectory. The $\|\theta_t - \theta_{t+1}\|_2^2$ of Adam is so huge that it cannot fully appear in both CIFAR-100 and Tiny ImageNet.

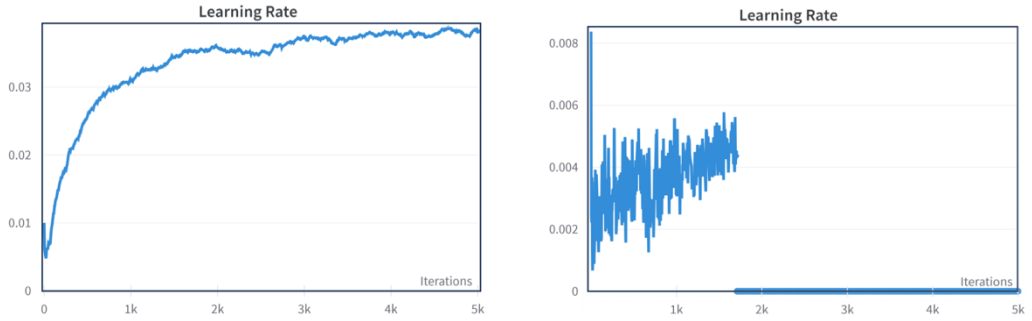
| Momentum | Gradient Penalty | Clipped Loss | Avg_Var ↓ | Acc. (Expert) ↑ | Acc. (Distill) ↑ |
|---|---|---|---|---|---|
| × | × | × | 0.1726 | 48.6 | 39.7 |
| ✓ | × | × | 7.9611 (×46) | 57.1 (+8.5) | 18.8 (−20.9) |
| ✓ | ✓ | × | 0.9331 (×5.4) | 54.1 (+5.5) | 41.7 (+2.0) |
| ✓ | ✓ | ✓ | **0.5899** (×3.4) | **54.4** (+5.8) | **42.0** (+2.3) |

(a) CIFAR-100

| Momentum | Gradient Penalty | Clipped Loss | Avg_Var ↓ | Acc. (Expert) ↑ | Acc. (Distill) ↑ |
|---|---|---|---|---|---|
| × | × | × | 0.0705 | 25.8 | 8.8 |
| ✓ | × | × | 2.2950 (×33) | 39.4 (+13.6) | 1.8 (−7.0) |
| ✓ | ✓ | × | 1.7358 (×24) | 39.0 (+13.2) | 10.0 (+1.2) |
| ✓ | ✓ | ✓ | **1.3066** (×18) | **39.5** (+13.7) | **10.8** (+2.0) |

(b) Tiny ImageNet

TABLE VII: Comparison between different buffer generation methods using SGD as base optimizer.




(a) Smooth expert trajectory (b) Non-smooth expert trajectory

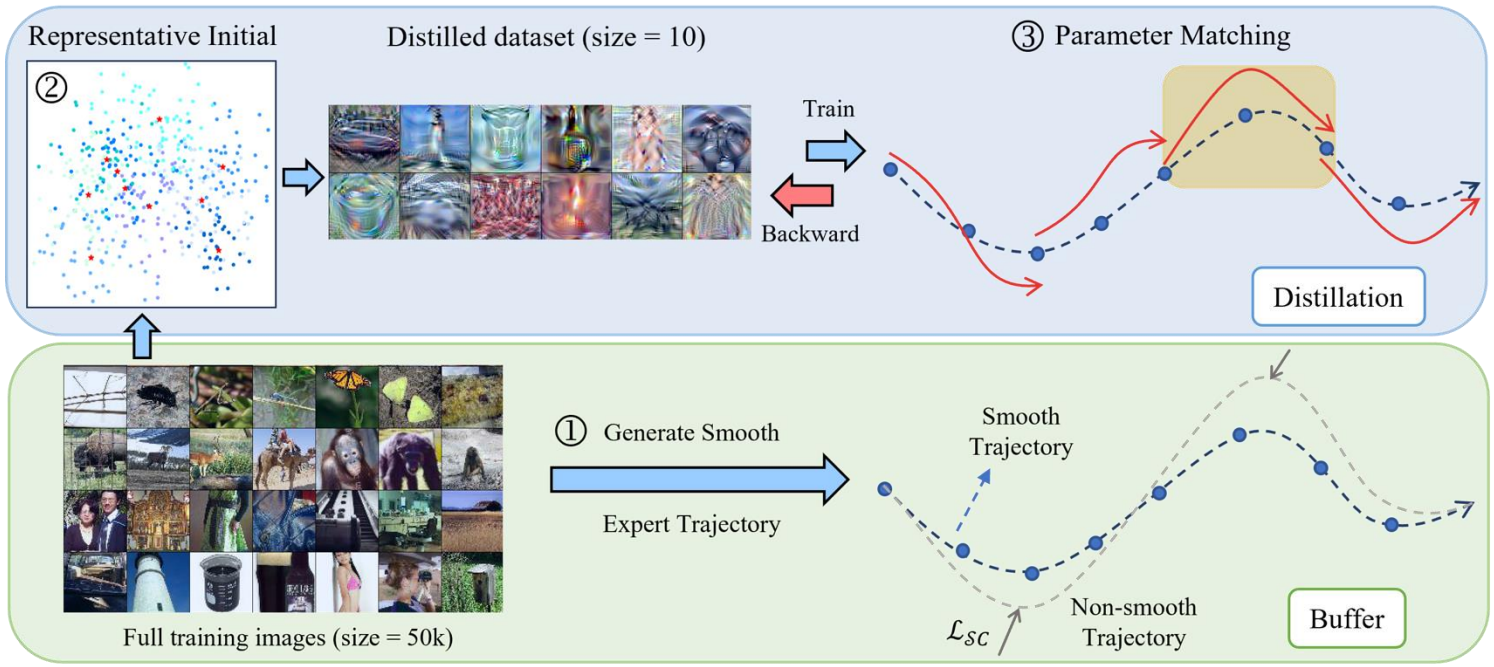
Fig. 8: Learning curve for student model learning rate. When applying a non-smooth expert trajectory, the output of the learning rate may encounter NaN which will lead to the collapse of the training process.

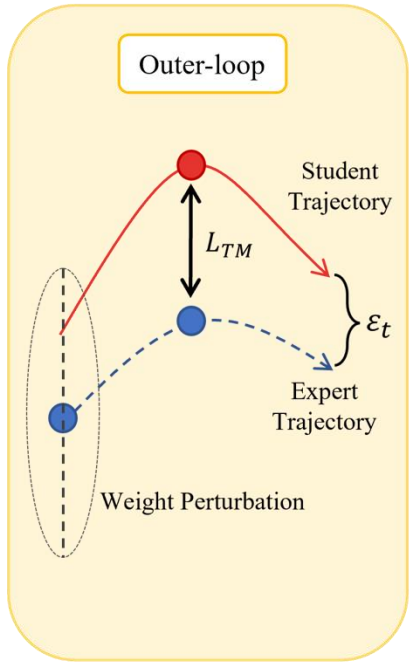
$$\mathcal{L}_{SC} = \underbrace{\lambda \log \frac{\exp\left(x_{i,y_i}\right)}{\sum_{c=1}^{C} \exp\left(x_{i,c}\right)}}_{\text{Clipped CELoss}} + \underbrace{\mu \mathop{\mathbb{E}}_{x_i \sim \mathbb{P}_x} \left[ \left( \|\nabla_{x_i} \mathcal{W}(x_i)\|_2 - \mathcal{K} \right)^2 \right]}_{\text{Gradient Penalty}}$$

**Smooth** expert trajectories is an important factor!

# AST: Effective Dataset Distillation through Alignment with Smooth and High-Quality Expert Trajectories

With refined expert trajectory, an enhanced parameter alignment method is proposed.




**for** ***each*** *distillation step* **do**
  Sample smooth expert trajectory $\tau \sim \{\tau_i\}$ with $\tau := \{\theta_t\}$;
  Choose random start epoch, $t \leqslant \mathcal{T}_+$;
  Perturb weight on initial expert model with $\tau^* := \{\theta_t^*\}$; // Method IV-D2
  Initialize student network with expert parameters $\hat{\theta}_t := \theta_t^*$;
  **for** *n = 1 to* $\mathcal{N}$ **do**
    Sample a mini-batch of distilled images: $b_{t+n} \sim \mathcal{D}_{syn}$;
    Compute the cross-entropy loss based on Eq. (21). ;
    Get $\nu$ based on Eq. (20) and balance $\ell_{IL}$: $\hat{\ell}_{BIL} = \nu \times \ell_{IL}$; // Method IV-C2
    Update student model: $\hat{\theta}_{t+n+1} = \hat{\theta}_{t+n} - \alpha \nabla \hat{\ell}_{BIL}$;
    **if** *n* ***in*** $\{\xi\}$ **then**
      Calculate intermediate matching loss $\mathcal{L}_i = \left\| \hat{\theta}_{t+n} - \theta^*_{t+\xi_i} \right\|_2^2 / \left\| \theta^*_t - \theta^*_{t+\xi_i} \right\|_2^2$; // Method IV-D1
    **end**
  **end**
  Get the final loss $\hat{\mathcal{L}} = \sum_{\xi_i} \beta_i \times \mathcal{L}_i$ ;
  Update $\mathcal{D}_{syn}$ and $\alpha$ with respect to $\hat{\mathcal{L}}$;
**end**

# Main Results

| IPC | CIFAR-10 | | | CIFAR-100 | | | Tiny ImageNet | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 10 | 50 | 1 | 10 | 50 | 1 | 10 |
| Full Dataset | | 84.8±0.1 | | | 56.2±0.3 | | 39.5±0.4 | |
| Random | 14.4±2.0 | 26.0±1.2 | 43.4±1.0 | 4.2±0.3 | 14.6±0.5 | 30.0±0.4 | 1.4±0.1 | 5.0±0.2 |
| Herding [8] | 21.5±1.2 | 31.6±0.7 | 40.4±0.6 | 8.4±0.3 | 17.3±0.3 | 33.7±0.5 | 2.8±0.2 | 6.3±0.2 |
| Forgetting [9] | 13.5±1.2 | 23.3±1.0 | 23.3±1.1 | 4.5±0.2 | 15.1±0.3 | 30.5±0.3 | 1.6±0.1 | 5.1±0.2 |
| DC [26] | 28.3±0.5 | 44.9±0.5 | 53.9±0.5 | 12.8±0.3 | 25.2±0.3 | - | - | - |
| DM [27] | 26.0±0.8 | 48.9±0.6 | 63.0±0.4 | 11.4±0.3 | 29.7±0.3 | 43.6±0.4 | 3.9±0.2 | 12.9±0.4 |
| DSA [36] | 28.8±0.7 | 52.1±0.5 | 60.6±0.5 | 13.9±0.3 | 32.3±0.3 | 42.8±0.4 | - | - |
| CAFE [39] | 30.3±1.1 | 46.3±0.6 | 55.5±0.6 | 12.9±0.3 | 27.8±0.3 | 37.9±0.3 | - | - |
| FRePo [33] | 45.1±0.5 | 59.1±0.3 | 69.6±0.4 | 25.9±0.1† | 40.9±0.1 | - | 13.5±0.1† | 20.4±0.1 |
| MTT [22] | 46.2±0.8 | 65.4±0.7 | 71.6±0.2 | 24.3±0.3 | 39.7±0.4 | 47.7±0.2 | 8.8±0.3 | 23.2±0.2 |
| TESLA [24] | 48.5±0.8† | 66.4±0.8 | 72.6±0.7 | 24.8±0.4 | 41.7±0.3 | 47.9±0.3 | 9.8±0.4 | 24.4±0.6 |
| FTD [23] | 46.8±0.3 | 66.6±0.3† | 73.8±0.2† | 25.2±0.2 | 43.4±0.3† | 50.7±0.3† | 10.4±0.3 | 24.5±0.2† |
| **Ours** | **48.8±0.9** | **67.1±0.4** | **74.6±0.5** | **26.6±0.4** | **44.4±0.6** | **51.7±0.7** | **13.7±1.4** | **25.7±1.1** |

TABLE III: Performance comparision trained with 128 width-ConvNet [49] to other state-of-the-art methods on the CIFAR and Tiny ImageNet. We cite the reported results from Sachdeva *et al.* [11] and Du *et al.* [23]. IPC: Images Per Class. Bold digits represent the best results and † refers to the second-best results among all the methods..

| | | Evaluation Model | | | |
|---|---|---|---|---|---|
| | | ConvNet | ResNet | VGG | AlexNet |
| Method | DC | 53.9±0.5 | 20.8±1.0 | 38.8±1.1 | 28.7±0.7 |
| | CAFE | 55.5±0.4 | 25.3±0.9 | 40.5±0.8 | 34.0±0.6 |
| | MTT | 71.6±0.2 | 61.9±0.7 | 55.4±0.8 | 48.2±1.0 |
| | FTD | 73.8±0.2 | 65.7±0.3 | 58.4±1.6 | 53.8±0.9 |
| | **Ours** | **74.6±0.5** | **67.3±0.4** | **60.3±0.5** | **56.7±0.3** |

TABLE IV: Generalization testing of different architectures on CIFAR-10 dataset with IPC 50.

| IPC | ImageNette | | ImageWoof | | ImageFruit | | ImageMeow | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 10 | 1 | 10 | 1 | 10 | 1 | 10 |
| Full dataset | 87.4±1.0 | | 67.0±1.3 | | 63.9±2.0 | | 66.7±1.1 | |
| MTT [22] | 47.7±0.9 | 63.0±1.3 | 28.6±0.8 | 35.8±1.8 | 26.6±0.8 | 40.3±1.3 | 30.7±1.6 | 40.4±2.2 |
| FTD [23] | 52.2±1.0 | 67.7±0.7 | 30.1±1.0 | 38.8±1.4 | 29.1±0.9 | 44.9±1.5 | 33.8±1.5 | 43.3±0.6 |
| **Ours** | **53.1±0.8** | **68.4±1.2** | **31.6±0.6** | **39.5±1.5** | **30.0±1.2** | **45.4±1.5** | **34.6±1.5** | **44.9±1.7** |

TABLE V: Applying our methods to 128×128 resolution ImageNet subsets. Bold digits represent the best results.

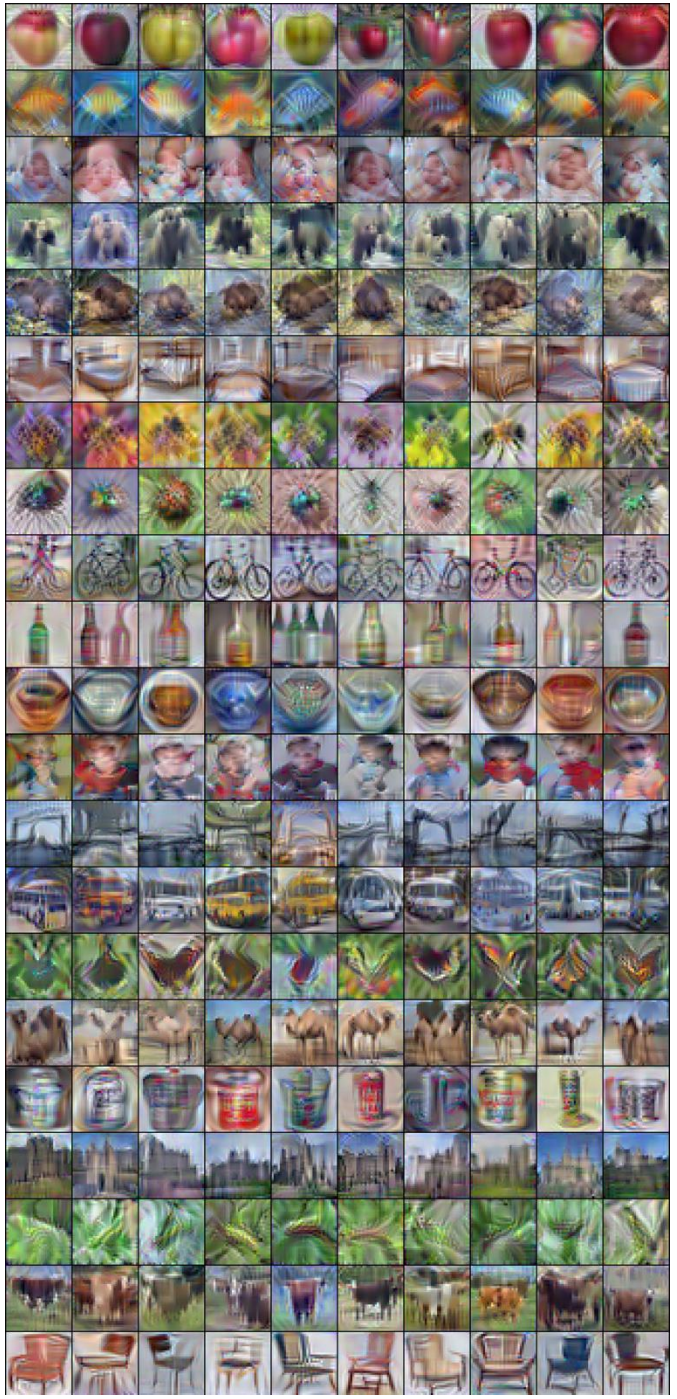
| Dataset | Image per class | 1 Iter. (sec) | 1k Iter. (min) | 5k Iter. (min) |
|---|---|---|---|---|
| CIFAR-10 | 1 | 0.5 | 8.3 | 41 |
| | 10 | 0.6 | 11 | 50 |
| | 50 | 0.9 | 15 | 75 |
| CIFAR-100 | 1 | 0.6 | 11 | 50 |
| | 10 | 0.85 | 14 | 70 |
| | 50 | 1.97 | 33 | 163 |
| Tiny ImageNet | 1 | 1.15 | 20 | 95 |
| | 10 | 2.42 | 40 | 200 |

TABLE VI: Distillation time for each dataset and support IPC.


0: apple
1:aquarium_fish
2: baby
3: bear
4: beaver
5: bed
6: bee
7: beetle
8: bicycle
9: bottle
10: bowl
11: boy
12: bridge
13: bus
14: butterfly
15: camel
16: can
17: castle
18: caterpillar
19: cattle
20: chair

=>

Distill



# Extremely high compression: 1 image per class

# Towards Lossless Dataset Distillation via Difficulty-Aligned Trajectory Matching (ICLR 24)

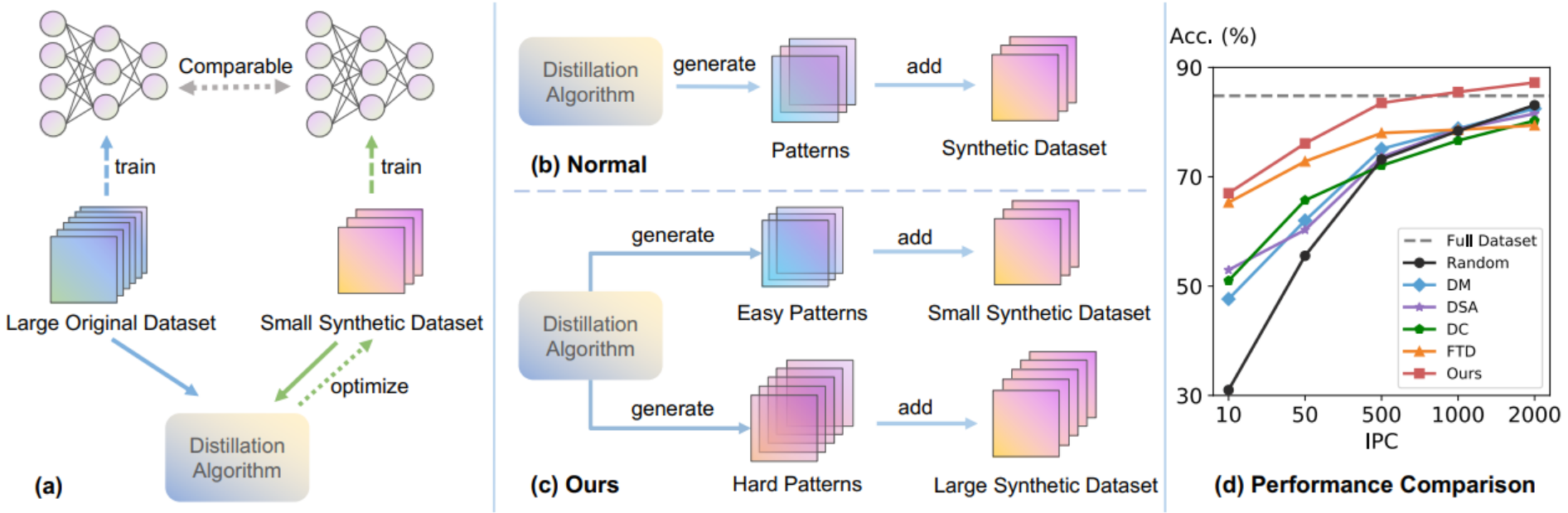



Figure 1: (a) Illustration of the objective of dataset distillation. (b) The optimization in dataset distillation can be viewed as the process of generating informative patterns on the synthetic dataset. (c) We align the difficulty of the synthetic patterns with the size of the distilled dataset, to enable our method to perform well in both small and large IPC regimes. (d) Comparison of the performance of multiple dataset distillation methods on CIFAR-10 with different IPC. As IPC increases, the performance of previous methods becomes worse than random selection.

# Vision-Language Dataset Distillation (arxiv 24)

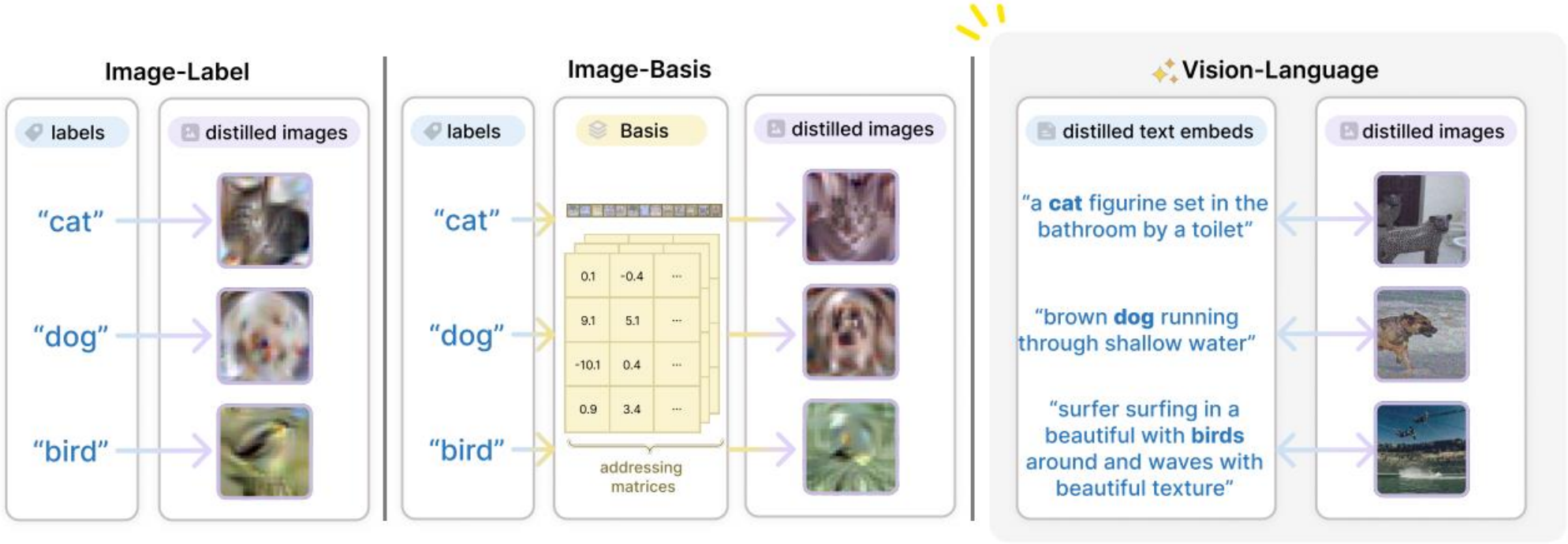


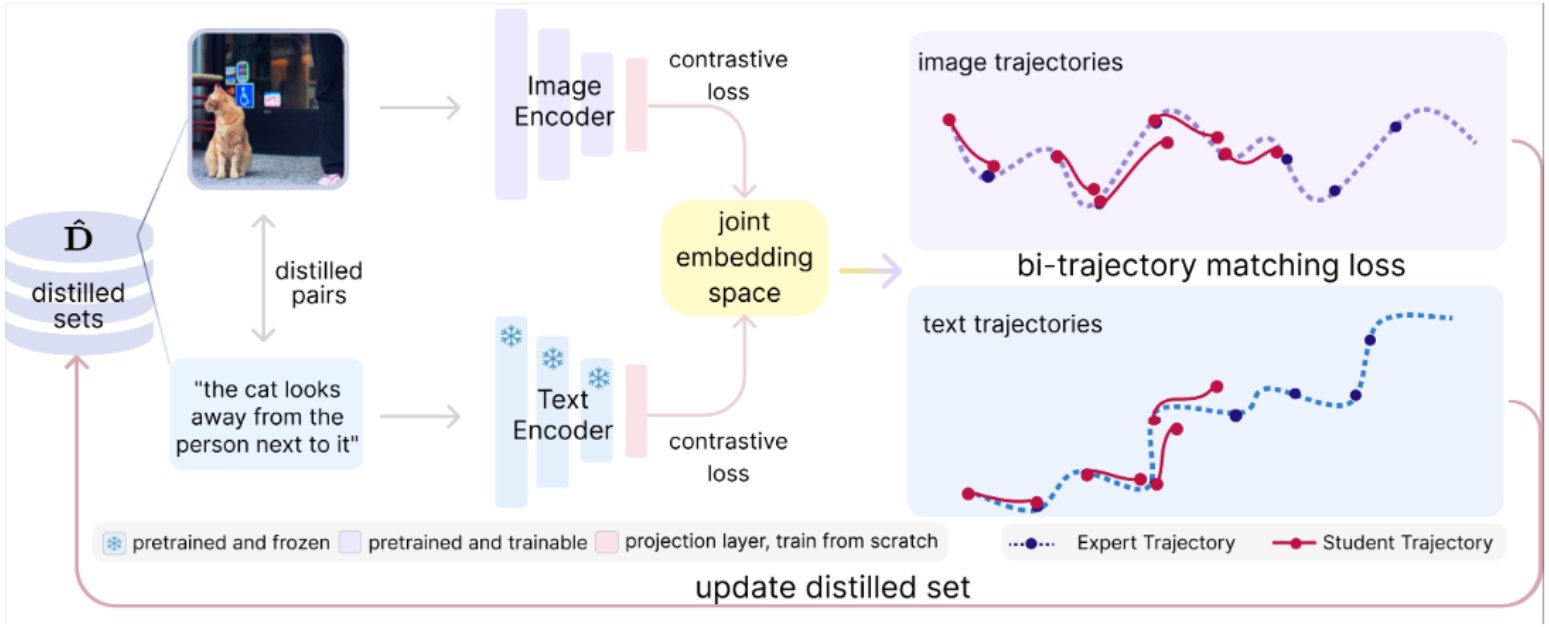

*Figure 1.* **Dataset Distillation Comparison.** (*Left*) Prior dataset distillation methods (Wang et al., 2018; Cazenavette et al., 2022; Nguyen et al., 2020) are class-specific: they distill the key information for each individual discrete class. (*Center*) Even the recently developed method (Deng & Russakovsky, 2022), which enables information sharing between classes through learned bases, still assumes a discrete set of classes. (*Right*) In contrast, we set out to distill vision-language datasets with no discrete classes; we do so via a novel method which jointly distills the images and texts.

# Vision-Language Dataset Distillation (arxiv 24)

## Bi-Trajectory-Guided Co-Distillation

The approach consists of two stages:

1. Obtaining the expert training trajectories $\{\tau^*\}$, with each trajectory $\tau^* = \{\theta_t^*\}_{t=0}^{T}$, by training multiple models for $T$ epochs on the full dataset $\mathbf{D}$. For our multimodal setting, the models are trained using **bidirectional contrastive loss**.
2. Training a set of student models on the current distilled dataset $\hat{\mathbf{D}}$ using the same bidirectional contrastive loss, and then updating the distilled dataset $\hat{\mathbf{D}}$ based on the **multimodal trajectory matching loss** of the student models' parameters and the optimal $\theta^*$.



# Vision-Language Dataset Distillation (arxiv 24)

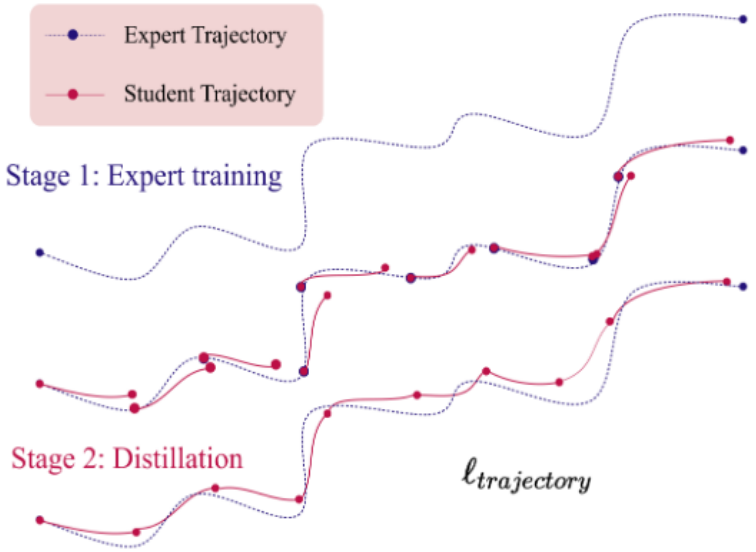
## Vision-Language Bi-Trajectory Matching

Following the MTT formulation, we randomly sample $M$ image-text pairs from $\mathbf{D}$ to initialize the distilled dataset $\hat{\mathbf{D}}$ (more details can be found elsewhere). We sample an expert trajectory (i.e., the trajectory of a model trained on the full dataset) $\tau^* = \{\theta_t^*\}_{t=0}^{T}$ and a random starting epoch $s$ to initialize $\hat{\theta}_s = \theta_s^*$.

We train the student model on the distilled dataset for $\hat{R}$ steps to obtain $\hat{\theta}_{s+\hat{R}}$. We then update the distilled dataset based on multimodal trajectory matching loss $\ell_{trajectory}$ computed on the accumulated difference between student trajectory and expert trajectory:

$$\ell_{trajectory} = \frac{\|\hat{\theta}_{img,s+\hat{R}} - \theta^*_{img,s+R}\|_2^2}{\|\theta^*_{img,s} - \theta^*_{img,s+R}\|_2^2} + \frac{\|\hat{\theta}_{txt,s+\hat{R}} - \theta^*_{txt,s+R}\|_2^2}{\|\theta^*_{txt,s} - \theta^*_{txt,s+R}\|_2^2}.$$

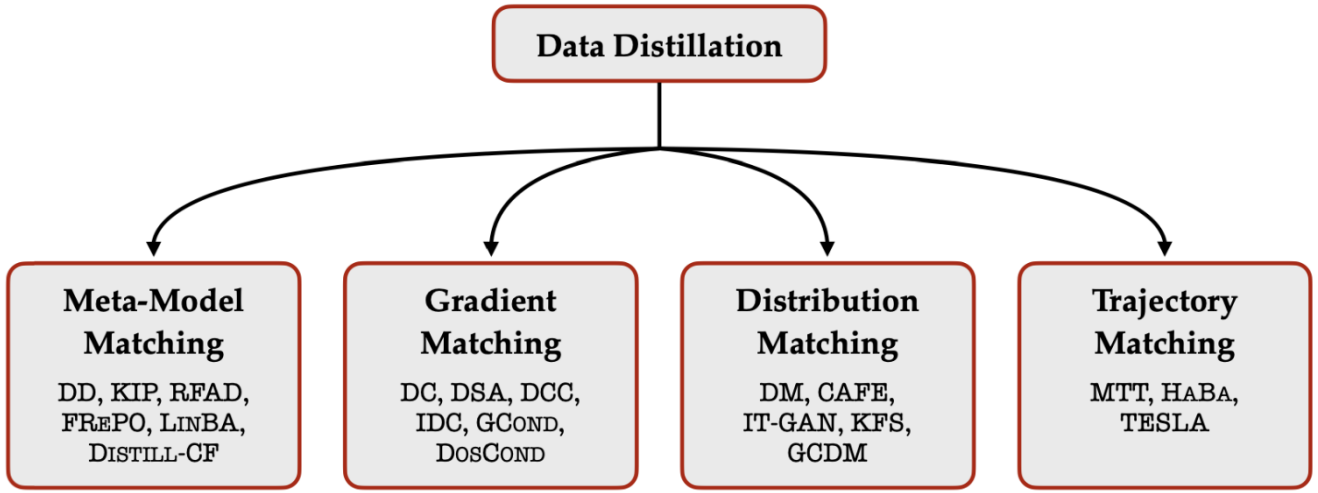
We update the distilled dataset by back-propagating through multiple ( $\hat{R}$) gradient descent updates to the $\hat{\mathbf{D}}$, specifically, image pixel space and text embedding space. We initialize the continuous sentence embeddings using a pretrained BERT model and update the distilled text in the continuous embedding space. For the distilled image optimization, we directly update the pixel values of the distilled images.



# Taxonomy of existing approaches

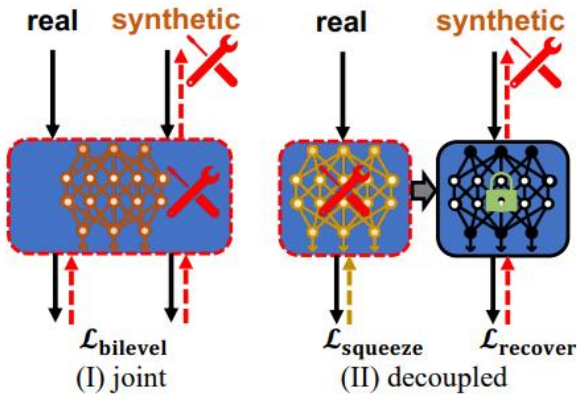
A bilevel optimization problem.

- Nested loop: The inner-loop typically optimizes a representative learning algorithm on the data summary, and using the optimized learning algorithm, the outer-loop optimizes a tractable proxy of Minimum.
- Existing approaches:
  - Meta-model matching
  - Gradient matching
  - Distribution matching
  - Trajectory matching



# Squeeze, Recover and Relabel: Dataset Condensation at ImageNet Scale From A New Perspective (NeurIPS 23)

## Decouple the bilevel optimization process: pretrained model assisted DD



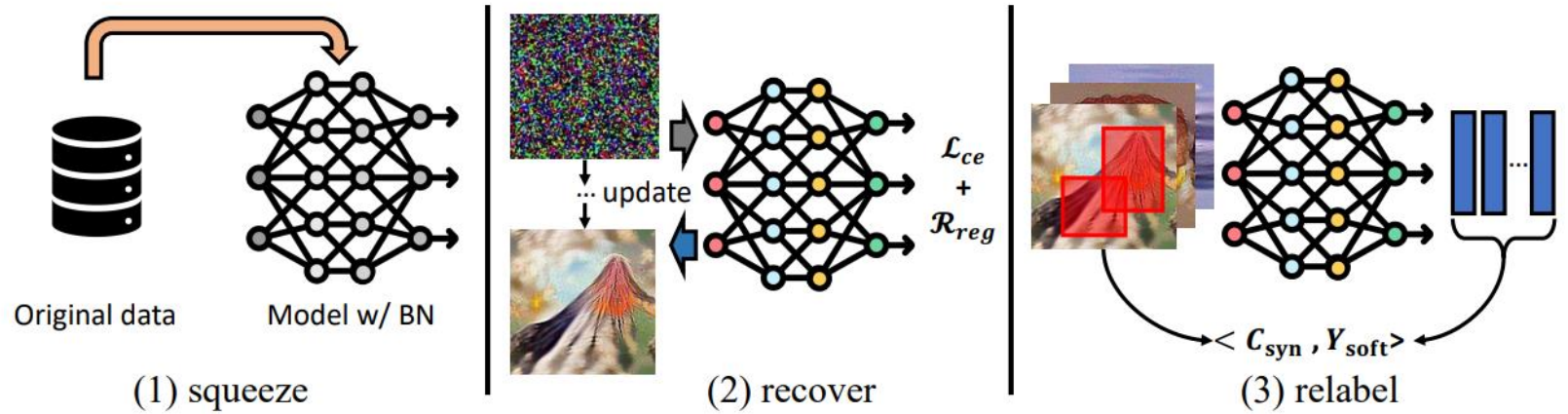
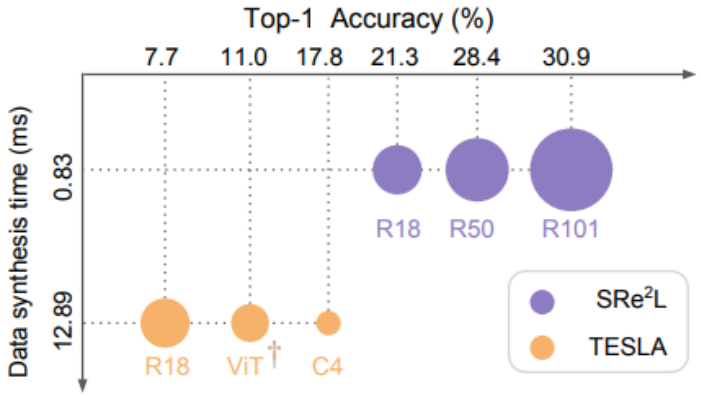




Figure 2: Overview of our framework. It consists of three stages: in the first stage, a model is trained from scratch to accommodate most of the crucial information from the original dataset. In the second stage, a recovery process is performed to synthesize the target data from the Gaussian noise. In the third stage, we relabel the synthetic data in a crop-level scheme to reflect the true label of the data.

1. Larger scales and higher resolution of datasets, e.g. ImageNet-1K
2. Cross model architectures.
3. Utilize many off-the-shelf pre-trained large models



# Squeeze, Recover and Relabel: Dataset Condensation at ImageNet Scale From A New Perspective (NeurIPS 23)

## An inversion process towards neural network. Borrowed from DeepInversion [1].

- How neural network compress and memorize large-scale dataset?
  - Project original data to feature maps.

$$\boldsymbol{x} \in \mathbb{R}^D \xrightarrow{f(\boldsymbol{x})} \boldsymbol{z} \in \mathbb{R}^d$$

  - Can be viewed as an implicit compression process.
- Whether can we explicitly re-construct the low-dimensional information from the neural network?

$$\boldsymbol{x} \in \mathbb{R}^D \xrightarrow{f(\boldsymbol{x})} \boldsymbol{z} \in \mathbb{R}^d \xrightarrow{g(\boldsymbol{z})} \widehat{\boldsymbol{x}} \in \mathbb{R}^D$$

  - Don't use the surrogate like before.
  - Instead by **aligning the BN statistics** (mean and variance), syn data are compelled to encapsulate a portion of original image distribution.

running mean

running var

$$\mathcal{R}_{\text{BN}}(\widetilde{\boldsymbol{x}}) = \sum_l \|\mu_l(\widetilde{\boldsymbol{x}}) - \mathbb{E}\left(\mu_l \mid \mathcal{T}\right)\|_2 + \sum_l \|\sigma_l^2(\widetilde{\boldsymbol{x}}) - \mathbb{E}\left(\sigma_l^2 \mid \mathcal{T}\right)\|_2$$

$$\approx \sum_l \|\mu_l(\widetilde{\boldsymbol{x}}) - \mathbf{BN}_l^{\text{RM}}\|_2 + \sum_l \|\sigma_l^2(\widetilde{\boldsymbol{x}}) - \mathbf{BN}_l^{\text{RV}}\|_2 \quad (7)$$

batch mean

batch var

where $l$ is the index of BN layer, $\mu_l(\widetilde{\boldsymbol{x}})$ and $\sigma_l^2(\widetilde{\boldsymbol{x}})$ are mean and variance. $\mathbf{BN}_l^{\text{RM}}$ and $\mathbf{BN}_l^{\text{RV}}$ are running mean and running variance in the pre-trained model at $l$-th layer, which are globally counted.

[1] Yin, Hongxu, et al. "Dreaming to distill: Data-free knowledge transfer via deepinversion." *CVPR*. 2020.

# Squeeze, Recover and Relabel: Dataset Condensation at ImageNet Scale From A New Perspective (NeurIPS 23)

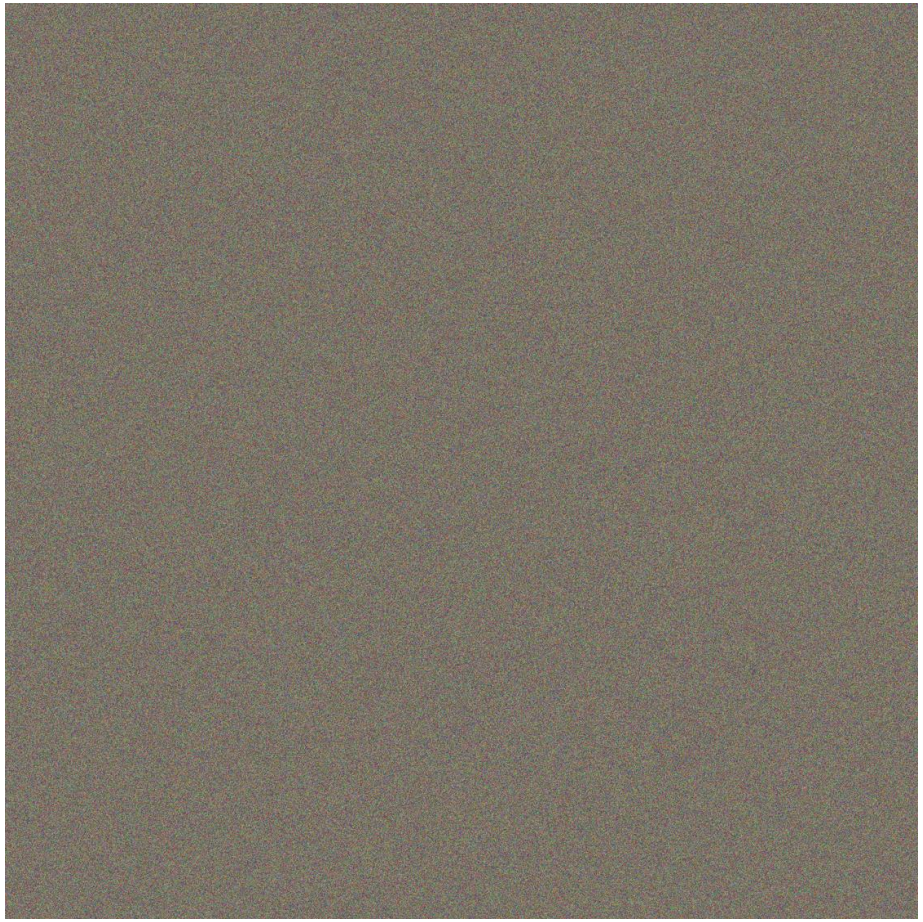
Main results.

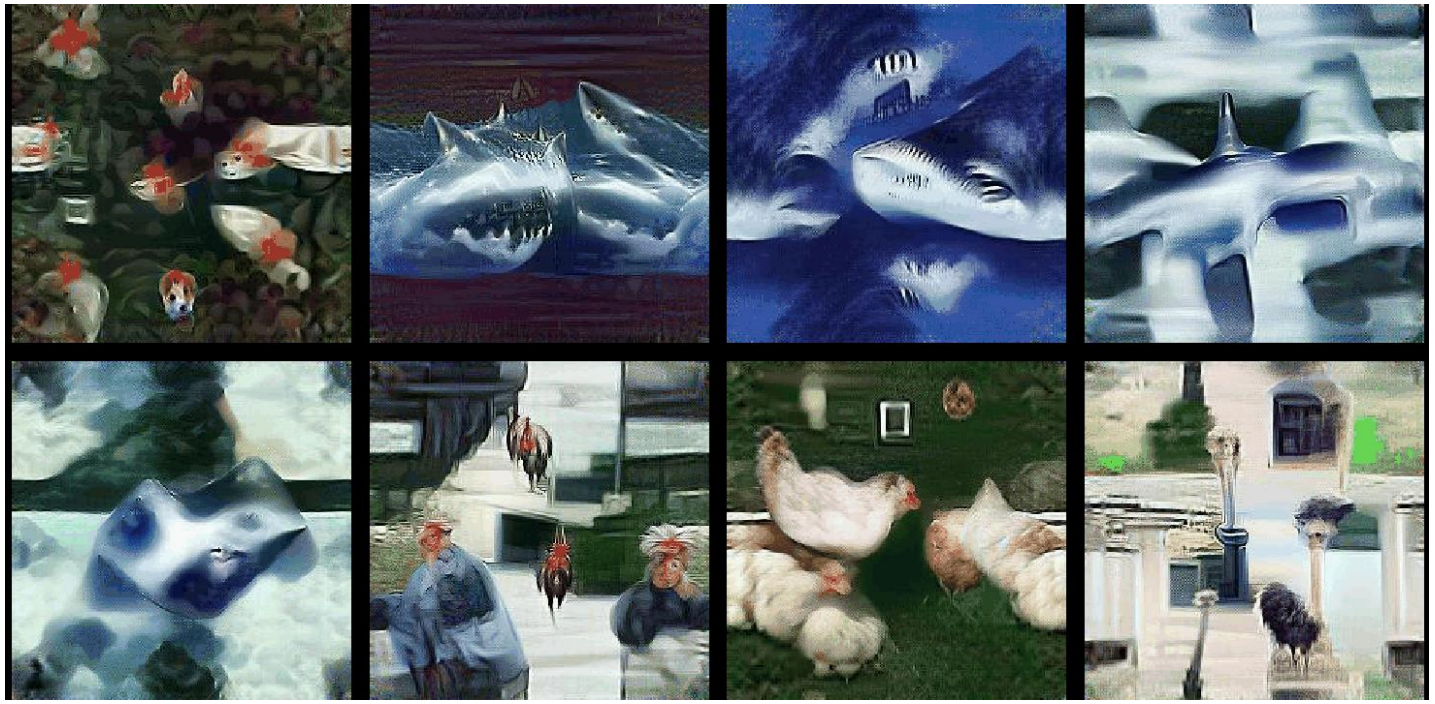
| Dataset | IPC | MTT [1] | TESLA [11] | TESLA [11] (ViT) | TESLA [11] (R18) | SRe$^2$L (R18) | SRe$^2$L (R50) | SRe$^2$L (R101) |
|---|---|---|---|---|---|---|---|---|
| Tiny-IN | 50 | 28.0±0.3 | - | - | - | 41.1±0.4 | 42.2±0.5 | 42.5±0.2 |
| | 100 | - | - | - | - | 49.7±0.3 | 51.2±0.4 | 51.5±0.3 |
| IN-1K | 10 | 64.0±1.3† | 17.8±1.3 | 11.0±0.2 | 7.7±0.1 | 21.3±0.6 | 28.4±0.1 | 30.9±0.1 |
| | 50 | - | 27.9±1.2 | - | - | 46.8±0.2 | 55.6±0.3 | 60.8±0.5 |
| | 100 | - | - | - | - | 52.8±0.3 | 61.0±0.4 | 65.8±0.2 |
| | 200 | - | - | - | - | 57.0±0.4 | 64.6±0.3 | 65.9±0.3 |

Table 4: Comparison with baseline models. † indicates the ImageNette dataset, which contains only 10 classes. TESLA [11] uses the downsampled ImageNet-1K dataset. Our results are derived from the full ImageNet-1K, which is more challenging on computation and memory, meanwhile, presenting greater applicability potential in real-world scenarios. The recovery model used is R18.

# Squeeze, Recover and Relabel: Dataset Condensation at ImageNet Scale From A New Perspective (NeurIPS 23)

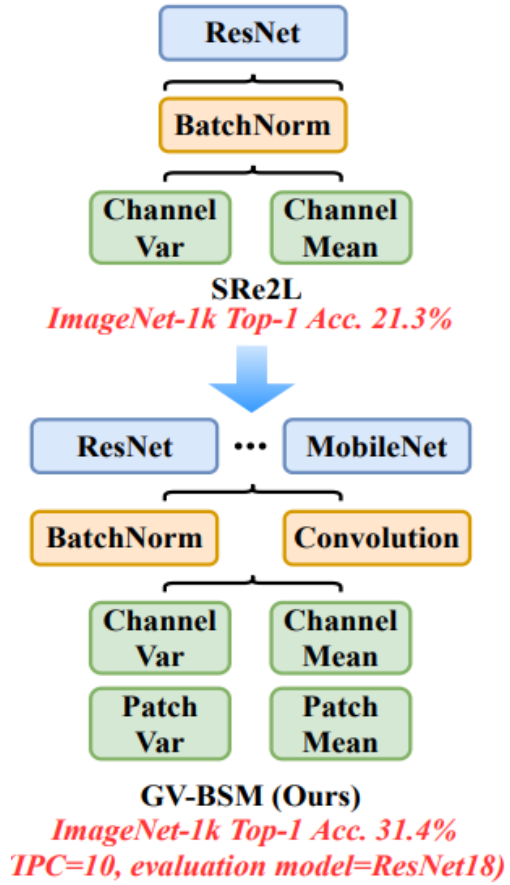
Distillation animation.





# Generalized Large-Scale Data Condensation via Various Backbone and Statistical Matching (CVPR 24)

Augment recover process: Propose generalized matching to preserves the rich and diverse information within the synthetic data.



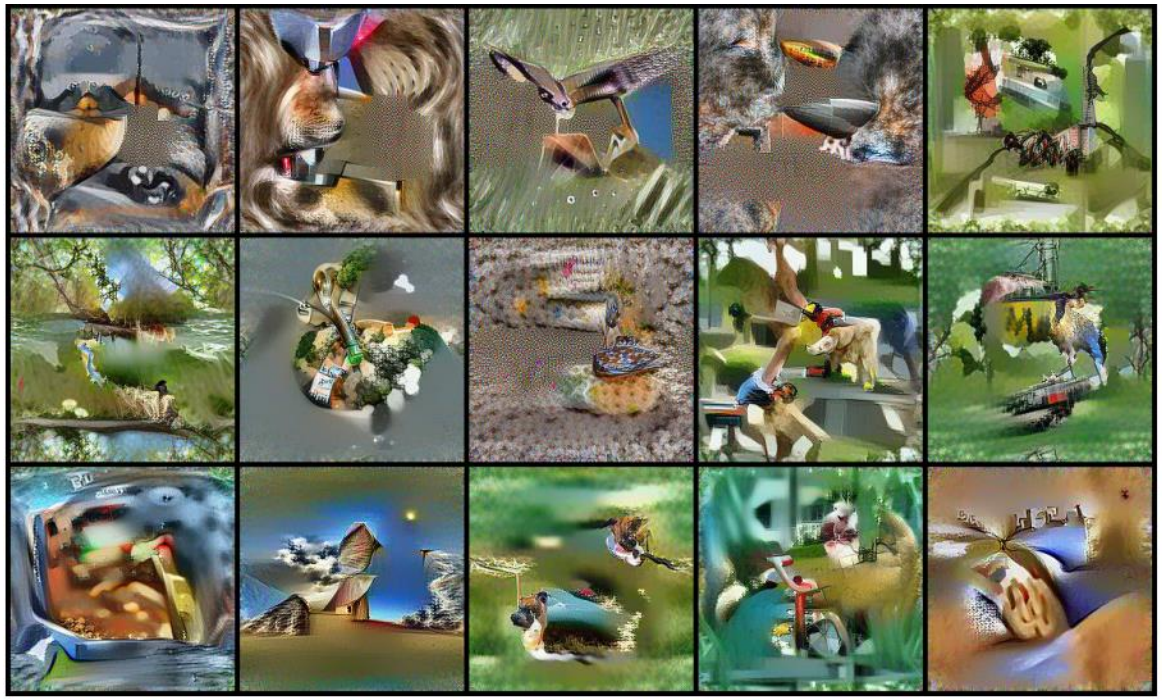



Figure 11. Synthetic data visualization on ImageNet-1k randomly selected from G-VBSM.

# Generalized Large-Scale Data Condensation via Various Backbone and Statistical Matching (CVPR 24)

Augment recover process: Propose generalized matching to preserves the rich and diverse information within the synthetic data.

| Dataset | IPC | MTT [1] (CW128) | DataDAM [20] (CW128) | TESLA [2] (R18) | SRe2L (R18) | SRe2L (R50) | SRe2L (R101) | G-VBSM (R18) | G-VBSM (R50) | G-VBSM (R101) |
|---|---|---|---|---|---|---|---|---|---|---|
| Tiny-ImageNet | 50 | 28.0±0.3 | 28.7±0.3 | - | 41.1±0.4 | 42.2±0.5 | 42.5±0.2 | 47.6±0.3 | 48.7±0.2 | 48.8±0.4 |
| | 100 | - | - | - | 49.7±0.3 | 51.2±0.4 | 51.5±0.3 | 51.0±0.4 | 52.1±0.3 | 52.3±0.1 |
| ImageNet-1k | 10 | 64.0±1.3† | 6.3±0.0 | 7.7±0.1 | 21.3±0.6 | 28.4±0.1 | 30.9±0.1 | 31.4±0.5 | 35.4±0.8 | 38.2±0.4 |
| | 50 | - | 15.5±0.2 | - | 46.8±0.2 | 55.6±0.3 | 60.8±0.5 | 51.8±0.4 | 58.7±0.3 | 61.0±0.4 |
| | 100 | - | - | - | 52.8±0.3 | 61.0±0.4 | 62.8±0.2 | 55.7±0.4 | 62.2±0.3 | 63.7±0.2 |

Table 4. Comparison with baseline models in Tiny-ImageNet and ImageNet-1k. † indicates the ImageNette dataset, which contains only 10 classes. DataDAM [20] and TESLA [2] use the downsampled 64×64 ImageNet-1k. We cite the experimental results from SRe2L [34].

| Dataset | IPC | Coreset Selection | | | | Training Set Synthesis (CW128) | | | | | | | Training Set Synthesis (R18) | | Whole Dataset |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Random | Herding | K-Center | Forgetting | DC [42] | DM [41] | CAFE [31] | KIP [15] | MTT [1] | DataDAM[20] | G-VBSM | SRe2L | G-VBSM | (CW128) |
| CIFAR-10 | 10 | 26.0±1.2 | 31.6±0.7 | 14.7±0.9 | 23.3±1.0 | 44.9±0.5 | 48.9±0.6 | 50.9±0.5 | 46.1±0.7 | 65.3±0.7 | 54.2±0.8 | 46.5±0.7 | 27.2±0.5 | 53.5±0.6 | 84.8±0.1 |
| | 50 | 43.4±1.0 | 40.4±0.6 | 27.0±1.4 | 23.3±1.1 | 53.9±0.5 | 63.0±0.4 | 62.3±0.4 | 53.2±0.7 | 71.6±0.2 | 67.0±0.4 | 54.3±0.3 | 47.5±0.6 | 59.2±0.4 | |
| CIFAR-100 | 1 | 4.2±0.3 | 8.3±0.3 | 8.4±0.3 | 4.5±0.2 | 12.8±0.3 | 11.4±0.3 | 14.0±0.3 | 12.0±0.2 | 24.3±0.3 | 14.5±0.5 | 16.4±0.7 | 2.0±0.2 | 25.9±0.5 | 56.2±0.3 |
| | 10 | 14.6±0.5 | 17.3±0.3 | 17.3±0.3 | 15.1±0.3 | 25.2±0.3 | 29.7±0.3 | 31.5±0.2 | 40.1±0.4 | 33.1±0.4 | 34.8±0.5 | 38.7±0.2 | 31.6±0.5 | 59.5±0.4 | |
| | 50 | 30.0±0.4 | 33.7±0.5 | 30.5±0.3 | - | 30.6±0.6 | 43.6±0.4 | 47.7±0.2 | - | 42.9±0.3 | 49.4±0.3 | 45.7±0.4 | 49.5±0.3 | 65.0±0.5 | |

Table 5. Comparison with baseline models on CIFAR-10/100. All methods, except for SRe2L and G-VBSM, use a 128-width ConvNet (CW128) for data synthesis and evaluation. G-VBSM utilizes {CW128, WRN-16-2, ResNet18 (R18), ShuffleNetV2-0.5, MobileNetV2-0.5} for data synthesis and {CW128, R18} for evaluation. We cite the experimental results, except for SRe2L's, from DataDAM [20].

# Dataset Distillation in Large Data Era (arxiv 23)

Augment relabel process: Apply Curriculum Data Augmentation to distill larger dataset.

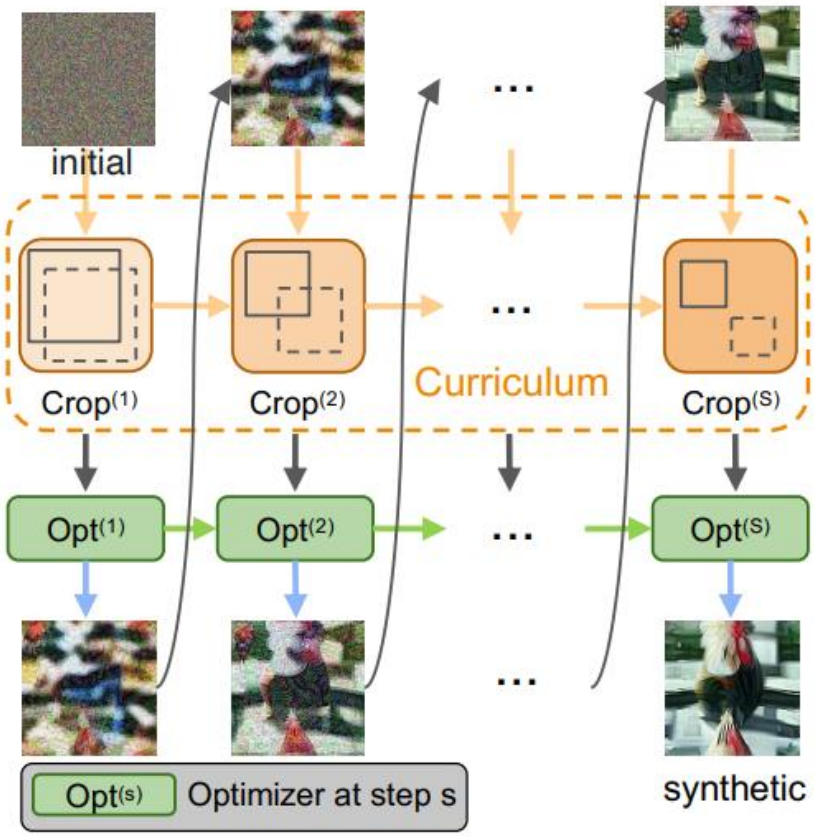



Figure 3: Curriculum data synthesis.

**Algorithm 1:** Our Curriculum Data Augmentation via *RandomResizedCrop*

**Input:** squeezed model $\phi_\theta$, recovery iteration $S$, curriculum milestone $T$, target label $\boldsymbol{y}$, default lower and upper bounds of crop scale $\beta_l$ and $\beta_u$ in *RandomResizedCrop*, decay of lower scale bound $\gamma$

**Output:** synthetic image $\boldsymbol{x}$

**Initialize:** $x_0$ from a standard normal distribution

**for** step $s$ from 0 to $S$-1 **do**

  **if** $s \leq T$ **then**

    $\alpha \leftarrow$

$$\begin{cases} \beta_u & \text{if } \texttt{step} \\ \beta_l + \gamma * (\beta_u - s/T) & \text{if } \texttt{linear} \\ \beta_l + \gamma * (\beta_u + \cos(\pi * s/T))/2 & \text{if } \texttt{cosine} \end{cases}$$

  **else**

    $\alpha \leftarrow \beta_l$

  **end**

  $\boldsymbol{x}_\mathcal{T} \leftarrow \mathit{RandomResizedCrop}(\boldsymbol{x}_s, \texttt{min\_crop} = \alpha, \texttt{max\_crop} = \beta_u)$

  $\boldsymbol{x}'_\mathcal{T} \leftarrow \boldsymbol{x}_\mathcal{T}$ is optimized w.r.t $\phi_\theta$ and $\boldsymbol{y}$ in Eq. 6.

  $\boldsymbol{x}_{s+1} \leftarrow \mathit{ReverseRandomResizedCrop}(\boldsymbol{x}_s, \boldsymbol{x}'_\mathcal{T})$

**end**

**return** $\boldsymbol{x} \leftarrow \boldsymbol{x}_S$

# Generalizing Dataset Distillation via Deep Generative Prior (CVPR 23)

Use the learned prior from pre-trained deep generative models.

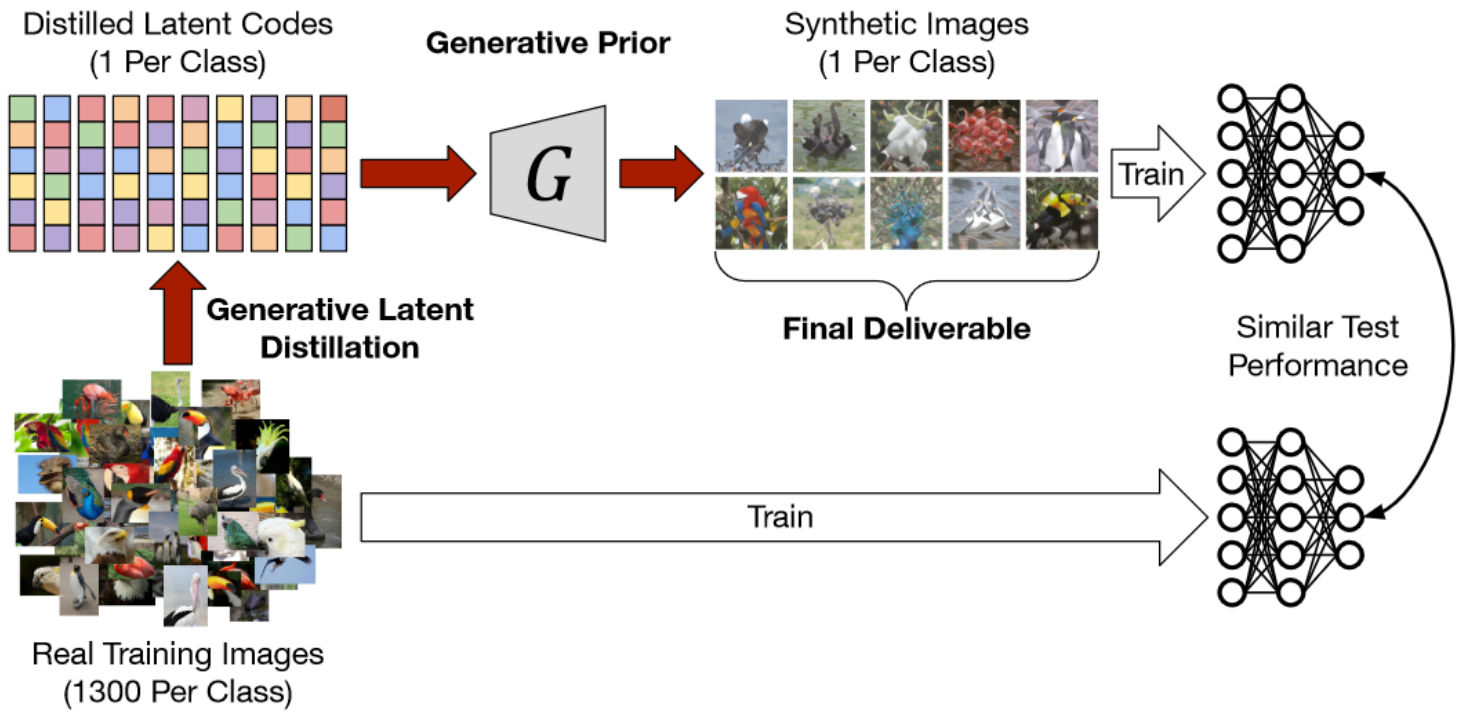



Figure 2. Rather than directly distilling a dataset into synthetic pixels (like all previous methods), our new method GLaD instead distills into the latent space of a *deep generative prior*. This enforces a tuneable amount of coherence in the output synthetic images, leading to far better generalization to new architectures.

# Latent Dataset Distillation with Diffusion Models (arxiv 24)

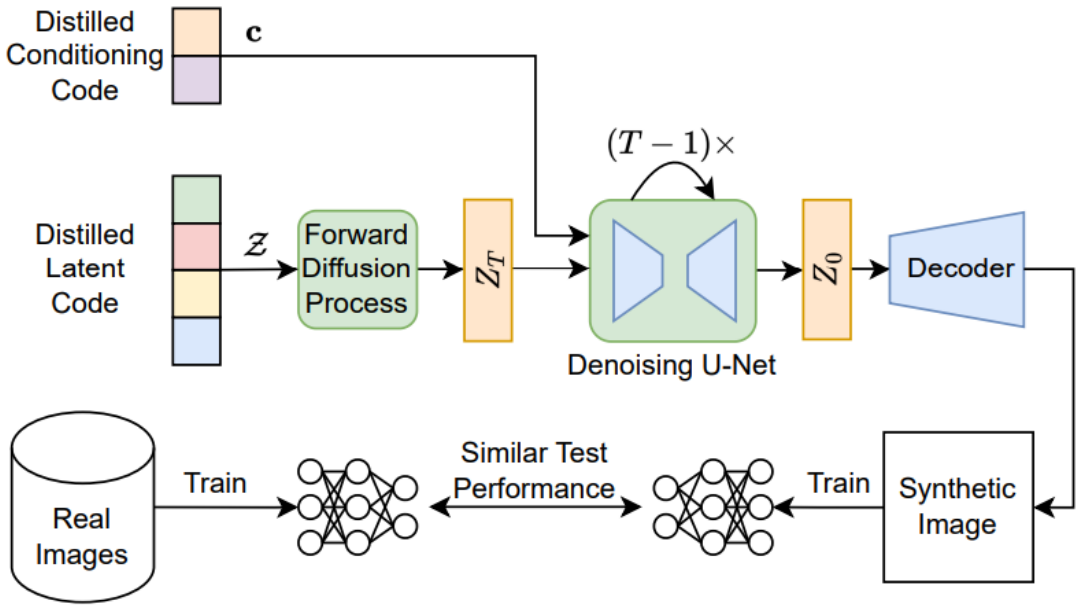



**Fig. 1:** Overview of the LD3M framework. Two components influence the generation of the synthetic images: The distilled latent codes and the distilled conditioning codes. The distilled latent codes are perturbated via Gaussian noise to the initial state $\mathbf{z}_T$. Next, it is iteratively denoised $(T-1)$ times with the pre-trained denoising U-Net of the LDM. Within each computation of the intermediate state $\mathbf{z}_t$, we add a linearly decreasing influence of $\mathbf{z}_T$ to allow an enhanced gradient flow to the distilled latent codes while making the conditioning also learnable. The pre-trained decoder translates the final latent code $\mathbf{z}_0$ back to pixel space. Note that LD3M can be used with any existing distillation algorithm, e.g., DC, DM, or MTT.

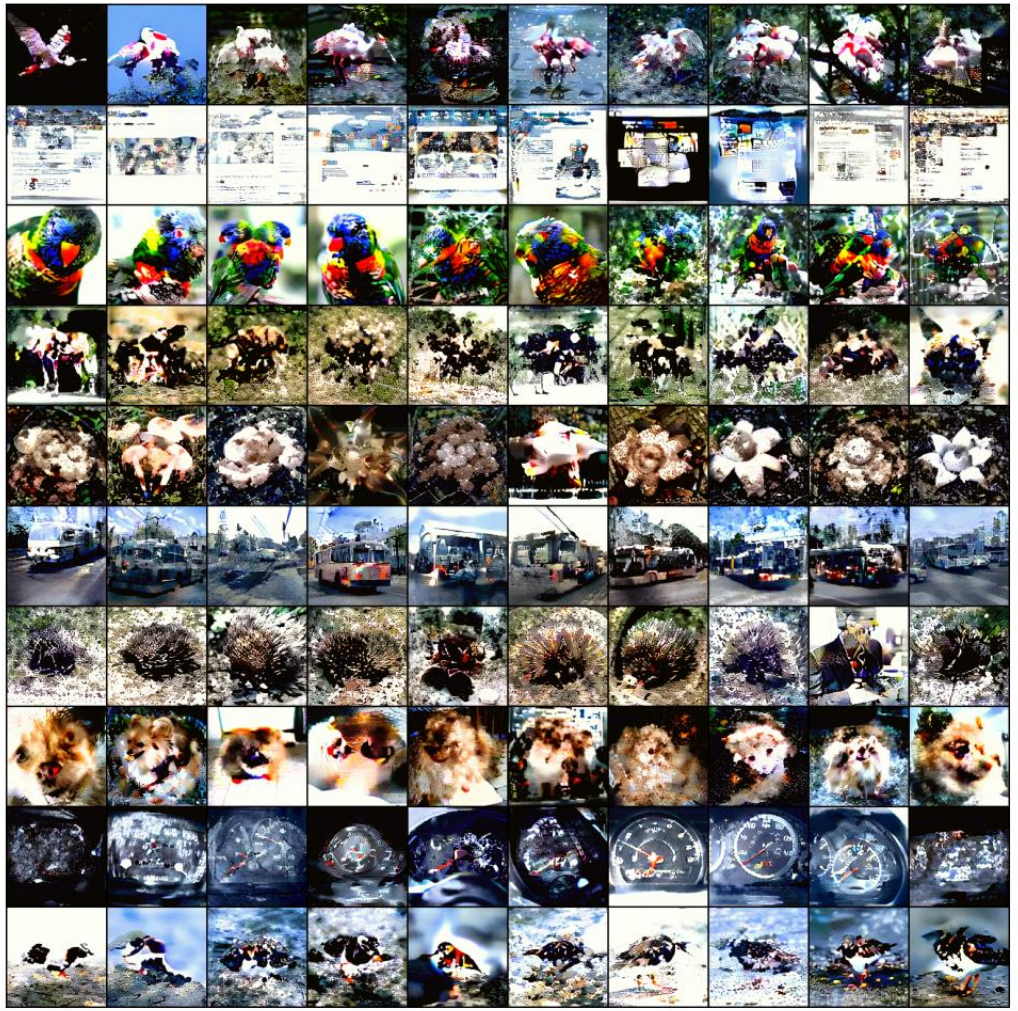


**Fig. 10:** Images distilled by DC in LD3M for IPC=10 and ImageNet-B.

# Application

# Meta Knowledge Condensation for Federated Learning (ICLR 23)

## One-shot FL.

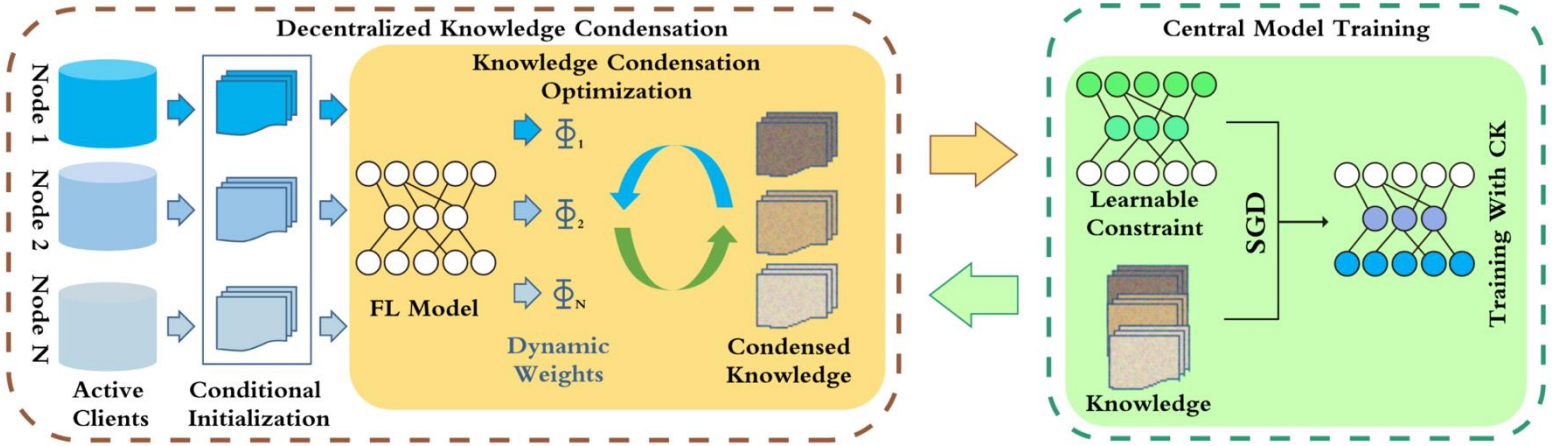



Figure 1: Illustration of our pipeline, in which only three active clients are shown. In our method, the local clients conduct meta knowledge condensation from local private data, and the server utilizes the uploaded meta knowledge for training a global model. The local meta knowledge condensation and central model training are conducted in an iterative manner. For meta knowledge extraction on clients, we design two mechanisms, *i.e.*, meta knowledge sharing, and dynamic weight assignment. For server-side central model training, we introduce a learnable constraint.

**Algorithm 1:** FedMK

**Input:** Original data $\mathcal{D}$; global parameters $\mathbf{W}_G$; generator parameter $\mathbf{w}^{\mathcal{G}}$; the communication budget .
**Output:** Optimal $\mathbf{W}_G^*$

**while** *not over the communication budget* **do**
  the server selects active clients $C$ uniformly at random, broadcasts $\mathbf{W}_G$ to the selected clients $C$.
  ▷Federated Meta Knowledge Extraction on selected clients $C$:
  **for** *all user* $c \in C$ *in parallel* **do**
    $\mathbf{w}^c \leftarrow \mathbf{W}_G$;
    **for** $t$ = *1, ..., #Round* **do**
      conduct the conditional initialization: $\hat{\mathcal{D}}^c_{ini} \leftarrow \hat{\mathcal{D}}^{c'}_{t-1}, c' \sim randint[1, C], c' \neq c$;
      calculate dynamic weights by Eq. 6;
      generate $\hat{\mathcal{D}}^c$ by Eq. 3;
    **end**
    send the $\hat{\mathcal{D}}^c$ to the server.
  **end**
  ▷Global Model Training on the server:
  update generator parameter $\mathbf{w}^{\mathcal{G}}$ by Eq. 9;
  generate $\hat{\mathcal{D}}^{pseu}$ by the updated generator $\mathcal{G}$;
  update global parameter $\mathbf{W}_G$ by Eq. 10.
**end**
return $\mathbf{W}_G$ as $\mathbf{W}_G^*$;

# FedDM: Iterative Distribution Matching for Communication-Efficient Federated Learning (CVPR 23)

## Able to guarantee (ε, δ)-differential privacy with the Gaussian mechanism.

**Algorithm 1** FedDM: Federated Learning with Iterative Distribution Matching

**Input:** Training set $\mathcal{D}$, set of synthetic samples $\mathcal{S}$, deep neural network parameterized with $w$, probability distribution over parameters $\mathcal{P}_w$, training iterations of distribution matching $T$, learning rate $\eta_c$ and $\eta_s$.
**Server executes:**
**for** each round $r = 1, \ldots, R$ **do**
  **for** client $k = 1, \ldots, K$ **do**
    $\mathcal{S}_k \leftarrow \text{ClientUpdate}(k, w_r)$
    Transmit $\mathcal{S}_k$ to the server
  **end for**
  Aggregate synthesized data from each client and build the surrogate function by Equation 9
  Update weights to $w_{r+1}$ on $\mathcal{S}$ by SGD with the learning rate $\eta_s$
**end for**
**ClientUpdate**$(k, w_r)$:
Initialize $\mathcal{S}_k$ from random noise or real examples.
**for** $t = 0, \cdots, T-1$ **do**
  Sample $w \sim P_w(w_r)$
  Sample mini-batch pairs $B_c^{\mathcal{D}_k} \sim \mathcal{D}_k$ and $B_c^{\mathcal{S}_k} \sim \mathcal{S}_k$ for each class $c$
  Compute $\mathcal{L}_c$ based on Equation 8, $\mathcal{L} \leftarrow \sum_{c=0}^{C-1} \mathcal{L}_c$
  Update $\mathcal{S}_k \leftarrow \mathcal{S}_k - \eta_c \nabla_{\mathcal{S}_k} \mathcal{L}$
**end for**

Figure 14: Synthesized images when no noise is added.

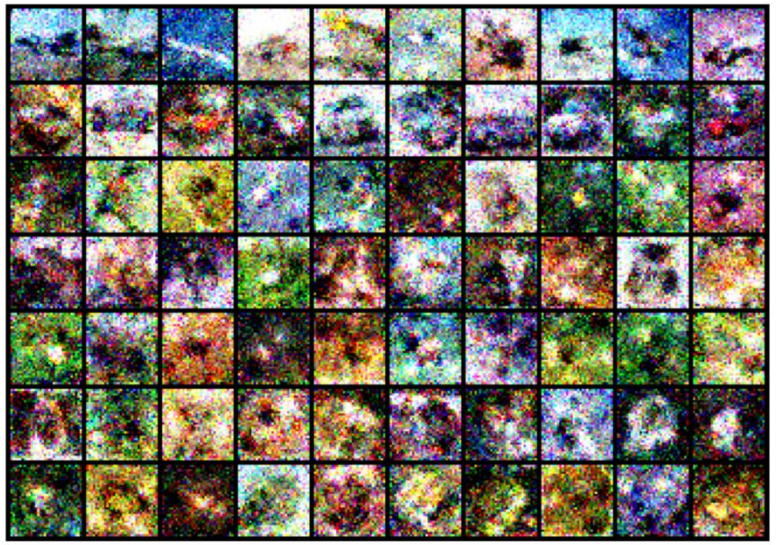


Figure 15: Synthesized images with $\sigma = 2$

# Privacy for Free: How does Dataset Condensation Help Privacy? (ICML 22)

## A milestone for data-efficient and privacy-preserving machine learning.

- They identify the privacy benefit of DC and propose to use DC for efficient and privacy-preserving data generation in machine learning pipeline.
- The synthetic dataset generated from random initialization is much more robustness against Membership inference attacks (MIA).
- The privacy is for free due to the dataset distillation. DC- synthesized data are perceptually **irreversible to original data** in terms of similarity metrics of L2 and LPIPS.

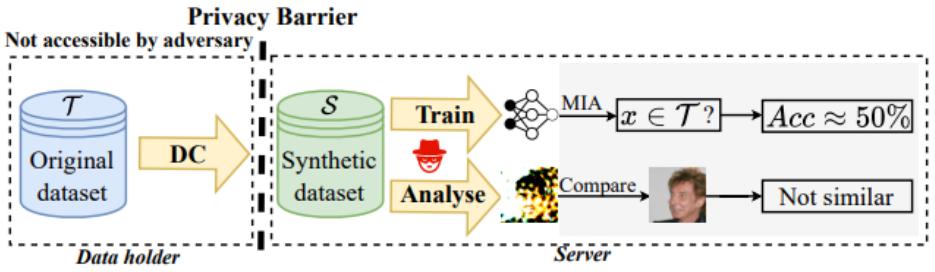



*Figure 1.* DC-synthesized data can be used for privacy-preserving model training and cannot be recovered through MIA and visual comparison analysis.

**Membership privacy** (Loss-based MIA, DC with **random initialization**)

Advantage (%) $= 2 \times \left(\frac{|\cup_x\{x|x\in\mathcal{T}_{mem}, l(x)<\tau\}|+|\cup_x\{x|x\in\mathcal{T}^C_{mem}, l(x)\geq\tau\}|}{|\mathcal{T}_{mem}|+|\mathcal{T}^C_{mem}|} - 50\%\right)$, where $|\mathcal{T}_{mem}| = |\mathcal{T}^C_{mem}|$, $\mathcal{T}_{mem} \cap \mathcal{T}^C_{mem} = \emptyset$ and $\mathcal{T}_{mem} \cup \mathcal{T}^C_{mem} = \mathcal{T}$

cGAN model can still leak privacy (Chen et al., 2020). Not private!

**Takeaway: The advantage of loss-based MIA is close to 0, indicating the attack cannot effectively infer data membership privacy.**

| Methods | $r_{ipc}$ | FashionMNST | CIFAR-10 | CelebA |
|---|---|---|---|---|
| cGAN (baseline, non-private) | 0.002 | $0.29 \pm 0.89$ | $-0.44 \pm 1.88$ | $-0.57 \pm 0.97$ |
| | 0.01 | $0.18 \pm 1.21$ | $-0.58 \pm 2.09$ | $-0.81 \pm 0.95$ |
| | 0.02 | $0.04 \pm 0.70$ | $-0.77 \pm 1.59$ | $-0.47 \pm 1.22$ |
| DM | 0.002 | $-0.34 \pm 0.42$ | $0.31 \pm 1.93$ | $-0.66 \pm 1.44$ |
| | 0.01 | $-0.29 \pm 0.48$ | $1.06 \pm 1.20$ | $-0.56 \pm 1.52$ |
| | 0.02 | $0.18 \pm 0.53$ | $0.72 \pm 0.70$ | $-0.67 \pm 1.18$ |
| DSA | 0.002 | $0.09 \pm 0.51$ | $0.39 \pm 1.04$ | $-0.39 \pm 1.90$ |
| | 0.01 | $0.52 \pm 0.55$ | $1.27 \pm 1.71$ | $-1.16 \pm 0.90$ |
| KIP (w/o zca) | 0.002 | $-1.13 \pm 1.84$ | $0.25 \pm 1.20$ | $-0.56 \pm 1.07$ |
| | 0.01 | $-0.95 \pm 0.96$ | $0.25 \pm 1.80$ | $-1.51 \pm 0.69$ |
| KIP (w/ zca) | 0.002 | $-0.56 \pm 2.02$ | $-0.64 \pm 1.86$ | $-1.06 \pm 1.10$ |
| | 0.01 | $-1.69 \pm 1.96$ | $-0.22 \pm 1.27$ | $-1.80 \pm 1.91$ |

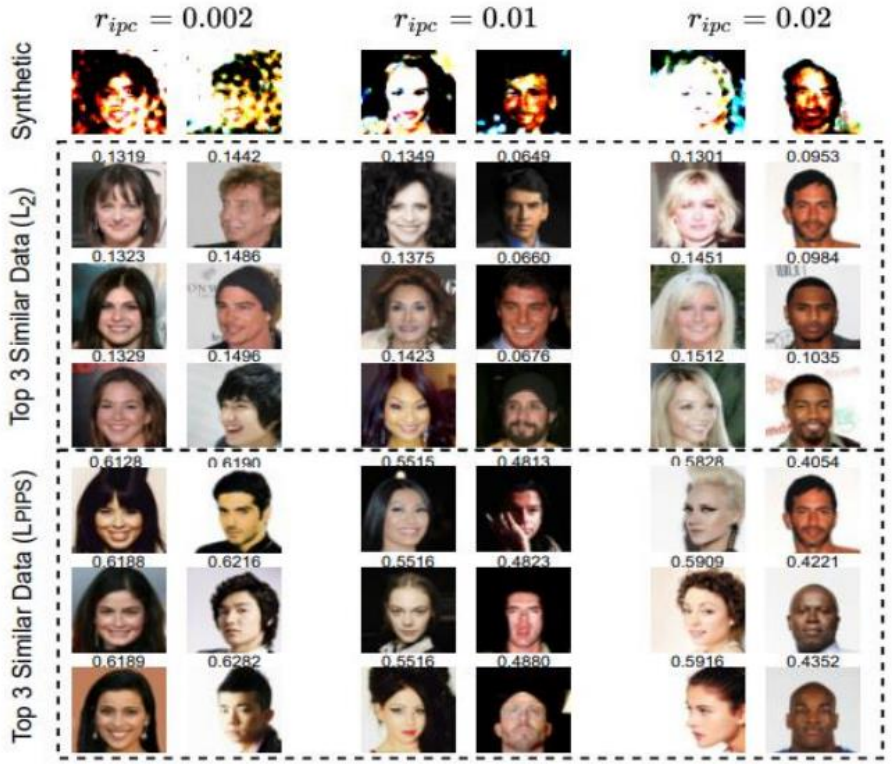
Dong T, Zhao B, Lyu L. Privacy for Free: How does Dataset Condensation Help Privacy?[J]. arXiv preprint arXiv:2206.00240, 2022.

# Privacy for Free: How does Dataset Condensation Help Privacy? (ICML 22)

**Visual privacy**:
Find the top 3 most similar images via comparison ($L_2$ norm & LPIPS).

Compression ratio
(ratio of images per class):

$$r_{ipc} = \frac{|\mathcal{S}|}{|\mathcal{T}|}$$




**Takeaway: The adversary cannot recover raw data from synthetic data by visual comparison.**

# Backdoor Attacks Against Dataset Distillation (NDSS 23)

Naïve backdoor attacks don't have much effect.

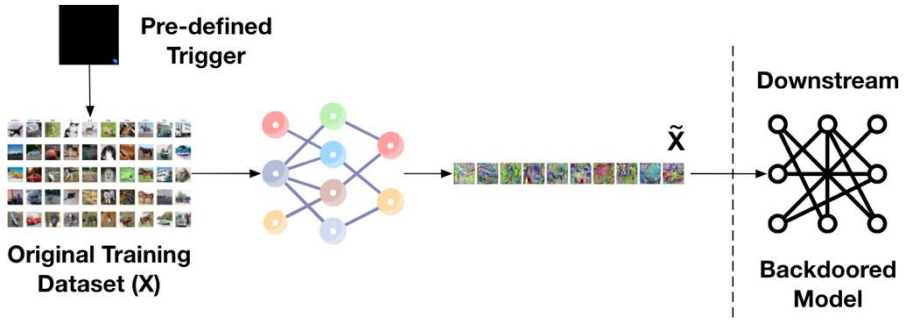
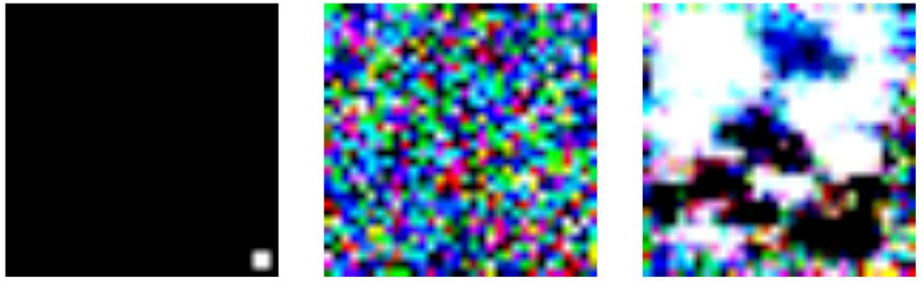



**Fig. 2:** Trigger insertion via NAIVEATTACK.



**(a)** Trigger image **(b)** DD distilled image **(c)** DC distilled image

**Fig. 3:** Illustration of pre-defined trigger used by the NAIVEATTACK and samples of distilled images by DD and DC models. We use airplane class from CIFAR10 for DD model to generate Figure 3b and DC model to generate Figure 3c.

Inject directly.

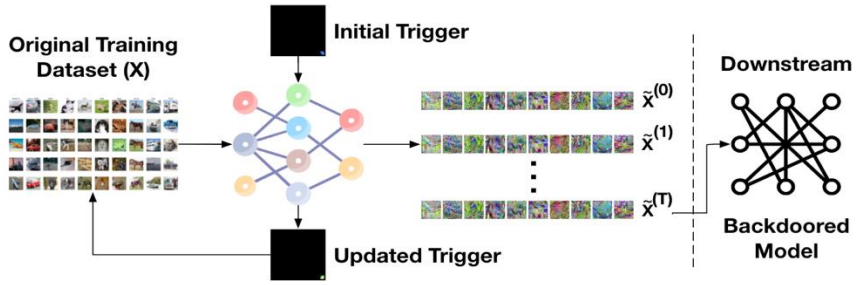
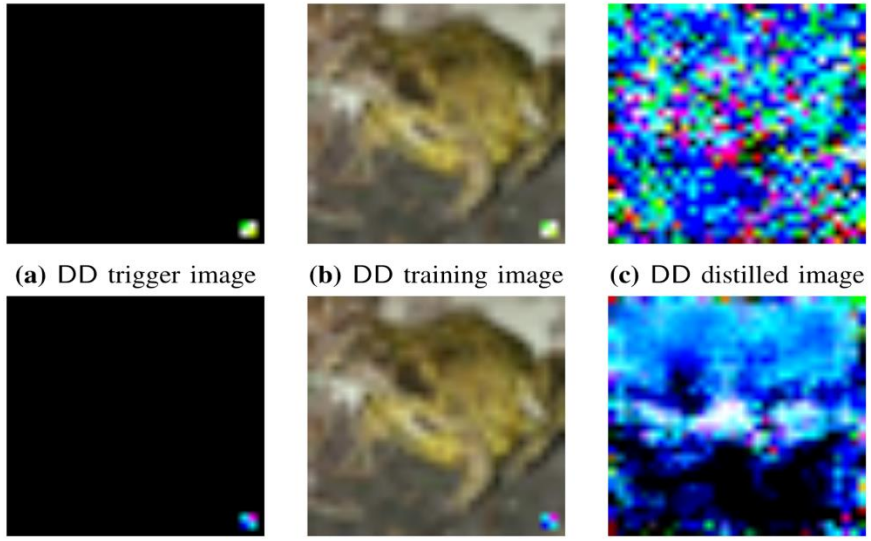



**Fig. 4:** Trigger updating on DOORPING.



**(a)** DD trigger image **(b)** DD training image **(c)** DD distilled image

**(d)** DC trigger image **(e)** DC training image **(f)** DC distilled image

**Fig. 5:** Illustration of the optimized trigger by DOORPING attack and samples of distilled images by DD and DC models. We use the *airplane* class from CIFAR10 as our target backdoor class when employing DOORPING.

Finetune trigger.

# Backdoor Attacks Against Dataset Distillation (NDSS 23)

ASR results. On the other hands, prove that DD is robust to normal backdoor attacks.

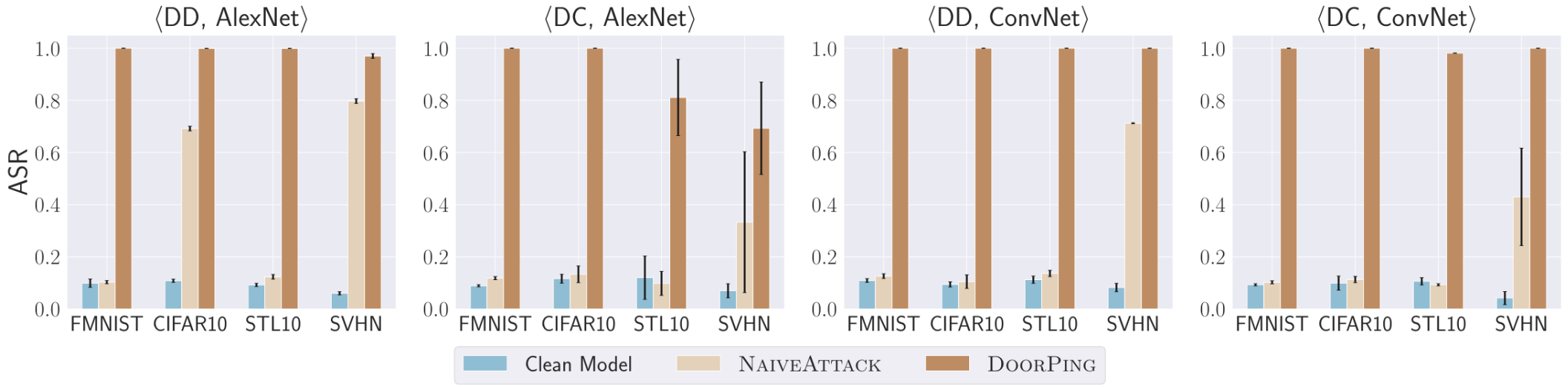



**Fig. 6:** *ASR* of clean model, NAIVEATTACK and DOORPING for different distillation algorithms and different model architectures.

# Potential direction

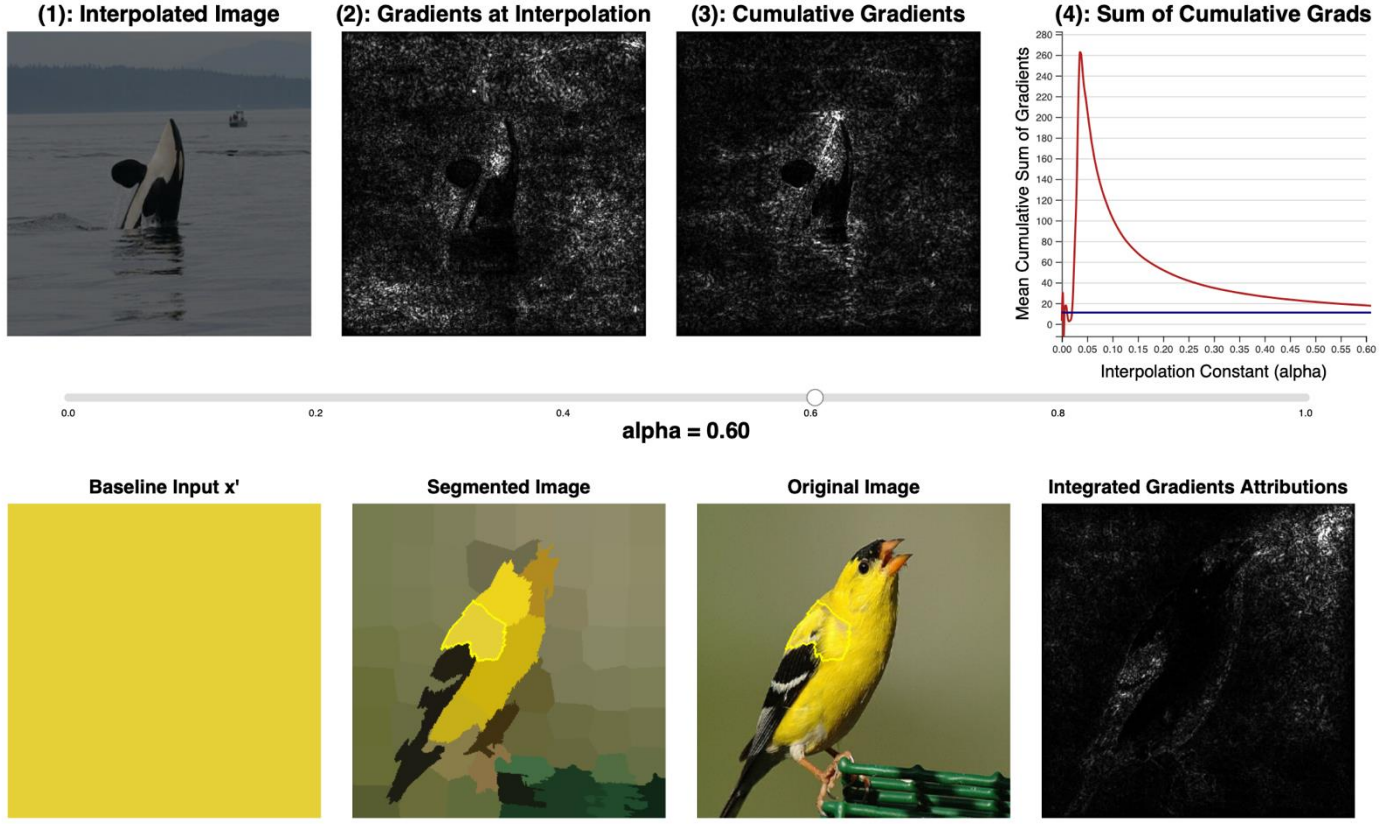
1. Vision-language dataset distillation.
    - Use reconstruction method, such as BN statistic, conv channel statistic.
    - Improve the efficiency of DD process.
2. Improve the cross-architecture ability and generalization capacity for various scale dataset.
3. Inverse the process: distill informative dataset from LM and use this to efficiently train small model.
4. Back to reconstruction attack:
    - We see the potential of this DD techniques and know the origin of DD from DLG. How about back to reconstruction?
    - E.g. Understanding Reconstruction Attacks with the Neural Tangent Kernel and Dataset Distillation (Noel Loo et al., ICLR 2024)
5. In FL scenario, is it really safe to transmit meta-knowledge? If safe, can have better method to prove? If not, privacy problems can be explored.
6. Connected with Explainable AI.
    - E.g. Help explain fine-grained classification problem.

# Integrated Gradients (IG) Method

## What's IG: using integrated gradients and comparing with the baseline to attribute feature.

- Integrated gradients is inspired by the Aumann-Shapley value, which provides a theoretically grounded way to determine **how much different groups of participants** contribute to the system.
- Focus on the **missingness** that contributes to the final result.
- Steps:
  - Select a baseline
  - Interpolate the image between baseline and original image
  - Calculate the gradients towards the image
  - Cumulative the gradients
- Problems:
  - Hard to **well-defined a notion of missingness** towards a DL models.
  - Like 'Why doesn't the attribution for "killer whale" highlight the black parts of the killer whale?'

$$\phi_i^{IG}(f, x, x') = \overbrace{(x_i - x'_i)}^{\text{Difference from baseline}} \times \underbrace{\int_{\alpha=0}^{1}}_{\text{From baseline to input...}} \overbrace{\frac{\delta f(x' + \alpha(x - x'))}{\delta x_i} d\alpha}^{\text{...accumulate local gradients}}$$



# Usage in the XAI: DD + IG

Use distilled images as baseline and explain why network can achieve fined-grained classification.

- Assumed steps:
  - Find a hierarchical dataset, which labels contain a coarse and fined annotation.
  - Distill the dataset using coarse labels and get distilled dataset (image per class=1).
  - Train a neural network to classify the fined annotated dataset.
  - View distilled image as baseline and then interpolate the image between baseline and original image.
  - Use IG method to explain why network can make the proper prediction.
- Potential advantages:
  - Distilled images are highly abstract, helping people understand why local and tiny features can make model do fined-grained task.
  - Serve well as a notion of missingness in IG because DD is a machine learning method and focuses on the coarse dataset.