DataDAM: Efficient Dataset Distillation with Attention Matching






Ahmad Sajedi*

Samir Khaki*

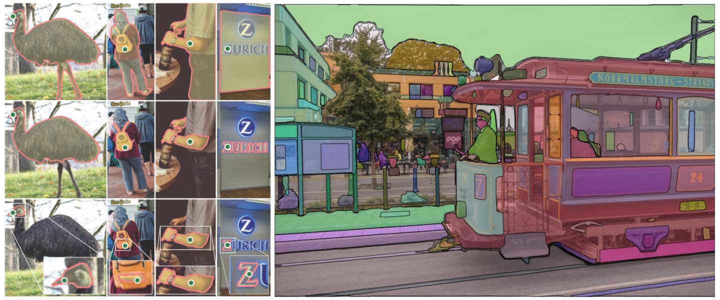
Ehsan Amjadian

Lucy Z. Liu

Yuri A. Lawryshyn

Kostas N. Plataniotis

Large-scaled Dataset



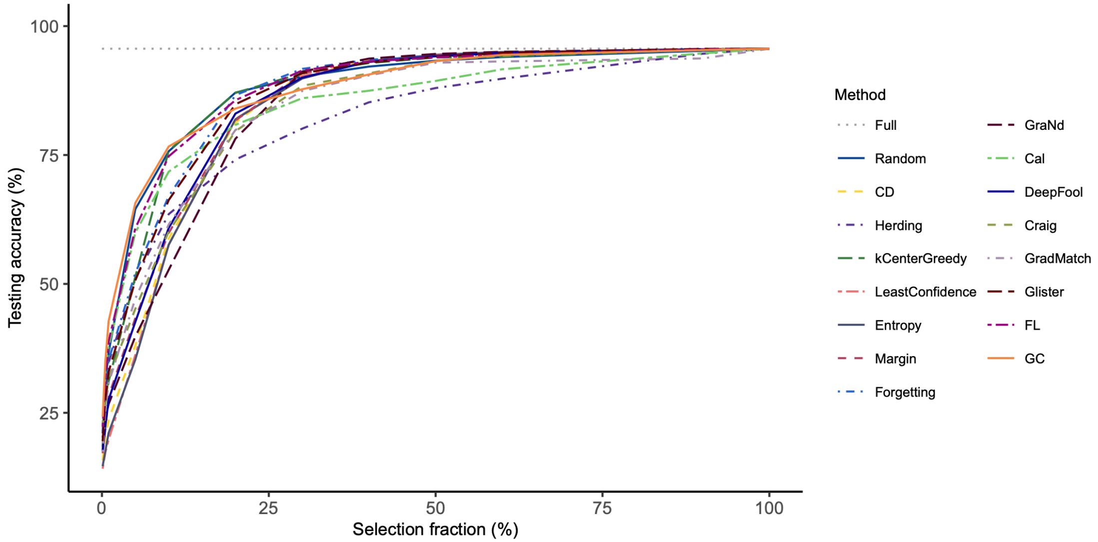
Facebook's SAM: SA-1B Dataset

- Data is growing at more than 20% per Year
- Larger datasets can offer increased performance at a cost of more human labor and training hours
- Privacy issues
- Large storage capacity required

How can we do better?

Can we transfer the information from large datasets into smaller ones?

Coreset Selection



	IPC	Ratio%	Resolution	Coreset Selection Random	Herding	K-Center	Forgetting
CIFAR-10	1	0.02	32	14.4 ± 2.0	21.5 ± 1.2	21.5 ± 1.3	13.5 ± 1.2
	10	0.2	32	26.0 ± 1.2	31.6 ± 0.7	14.7 ± 0.9	23.3 ± 1.0
	50	1	32	43.4 ± 1.0	40.4 ± 0.6	27.0 ± 1.4	23.3 ± 1.1
CIFAR-100	1	0.2	32	4.2 ± 0.3	8.3 ± 0.3	8.4 ± 0.3	4.5 ± 0.2
	10	2	32	14.6 ± 0.5	17.3 ± 0.3	17.3 ± 0.3	15.1 ± 0.3
	50	10	32	30.0 ± 0.4	33.7 ± 0.5	30.5 ± 0.3	-
Tiny ImageNet	1	0.2	64	1.4 ± 0.1	2.8 ± 0.2	1.6 ± 0.1	-
	10	2	64	5.0 ± 0.2	6.3 ± 0.2	5.1 ± 0.2	-
	50	10	64	15.0 ± 0.4	16.7 ± 0.3	15.0 ± 0.3	-

Coreset Selection performance on CIFAR10 with ResNet-18
(Chengcheng Guo, Bo Zhao, and Yanbing Bai. 2022. DeepCore)

- Coreset relies on a Heuristic estimate
- Often results in a sub optimal result compared to learned selection methods

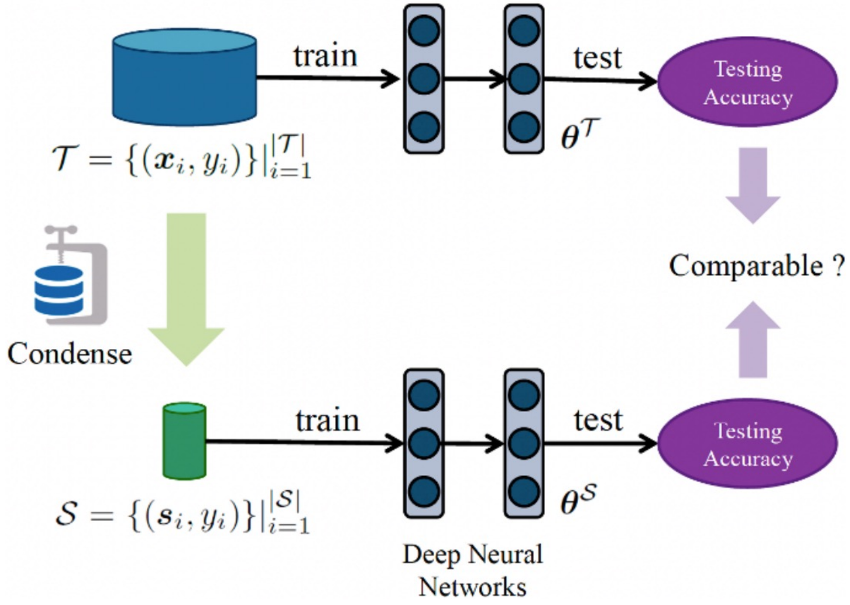
Dataset Distillation (DD)

- DD synthesizes a **small yet informative** dataset that approximates the large dataset!

$$\mathcal{T} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{|\mathcal{T}|}$$

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

$$\mathcal{S} = \{(\boldsymbol{s}_j, y_j)\}_{j=1}^{|\mathcal{S}|}$$



Dataset Condensation with Gradient Matching, ICLR, 2021 (Zhao et al.)

Goal: A model trained on the synthetic dataset should have a similar generalization performance to that trained on the original one.

Previous Works

Goal: To generate **synthetic dataset** that **approximate** the original dataset

How can we solve this?

(1) Performance Matching

$$\mathcal{L}(\mathcal{S},\mathcal{T}) = \mathbb{E}_{\theta^{(0)}\sim\Theta}[l(\mathcal{T};\theta^{(T)})],$$
$$\theta^{(t)} = \theta^{(t-1)} - \eta\nabla l(\mathcal{S};\theta^{(t-1)}),$$

(2) Label distillation

$$\tilde{\boldsymbol{Y}}_{\mathcal{S}}^{*} = \underset{\tilde{\boldsymbol{Y}}_{\mathcal{S}}}{\arg\min} \sum_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{T}} L\left(f_{\boldsymbol{\Theta}'}\left(\boldsymbol{x}\right),\boldsymbol{y}\right)$$

Shortcomings: Heavy computation !

(3) Gradient Matching

$$\mathcal{L}(\mathcal{S},\mathcal{T}) = \mathbb{E}_{\theta^{(0)}\sim\Theta}[\sum_{t=0}^{T}\mathcal{D}(\mathcal{S},\mathcal{T};\theta^{(t)})],$$
$$\theta^{(t)} = \theta^{(t-1)} - \eta\nabla l(\mathcal{S};\theta^{(t-1)}),$$
$$\mathcal{D}(\mathcal{S},\mathcal{T};\theta) = \sum_{c=0}^{C-1} d(\nabla l(\mathcal{S}_c;\theta),\nabla l(\mathcal{T}_c;\theta)),$$
$$d(\mathbf{A},\mathbf{B}) = \sum_{i=1}^{L}\sum_{j=1}^{J_i}(1 - \frac{\mathbf{A_j^{(i)}}\cdot\mathbf{B_j^{(i)}}}{\|\mathbf{A_j^{(i)}}\|\|\mathbf{B_j^{(i)}}\|}),$$

Shortcomings: Heavy computation
Biased Samples

Previous Works (2)

(4) Distribution Matching (DM)

$$\mathcal{L}(\mathcal{S},\mathcal{T}) = \mathbb{E}_{\theta\in\Theta}[\mathcal{D}(\mathcal{S},\mathcal{T};\theta)]$$

$$\mathcal{D}(\mathcal{S},\mathcal{T};\theta) = \sum_{c=0}^{C-1} \|\mu_{\theta,s,c} - \mu_{\theta,t,c}\|^2,$$

$$\mu_{\theta,s,c} = \frac{1}{M_c}\sum_{j=1}^{M_c} f_\theta^{(i)}(X_{s,c}^{(j)}), \quad \mu_{\theta,t,c} = \frac{1}{N_c}\sum_{j=1}^{N_c} f_\theta^{(i)}(X_{t,c}^{(j)}),$$

- **Avoid** expensive computation stemming from bi-level optimization
- Performance is **lower** than SOTA

(5) Matching Train Trajectory (MTT)

$$\mathcal{L} = \frac{\|\hat{\theta}_{t+N} - \theta^*_{t+M}\|_2^2}{\|\theta^*_t - \theta^*_{t+M}\|_2^2},$$

- **Incurs** expensive computation stemming from training multiple experts.

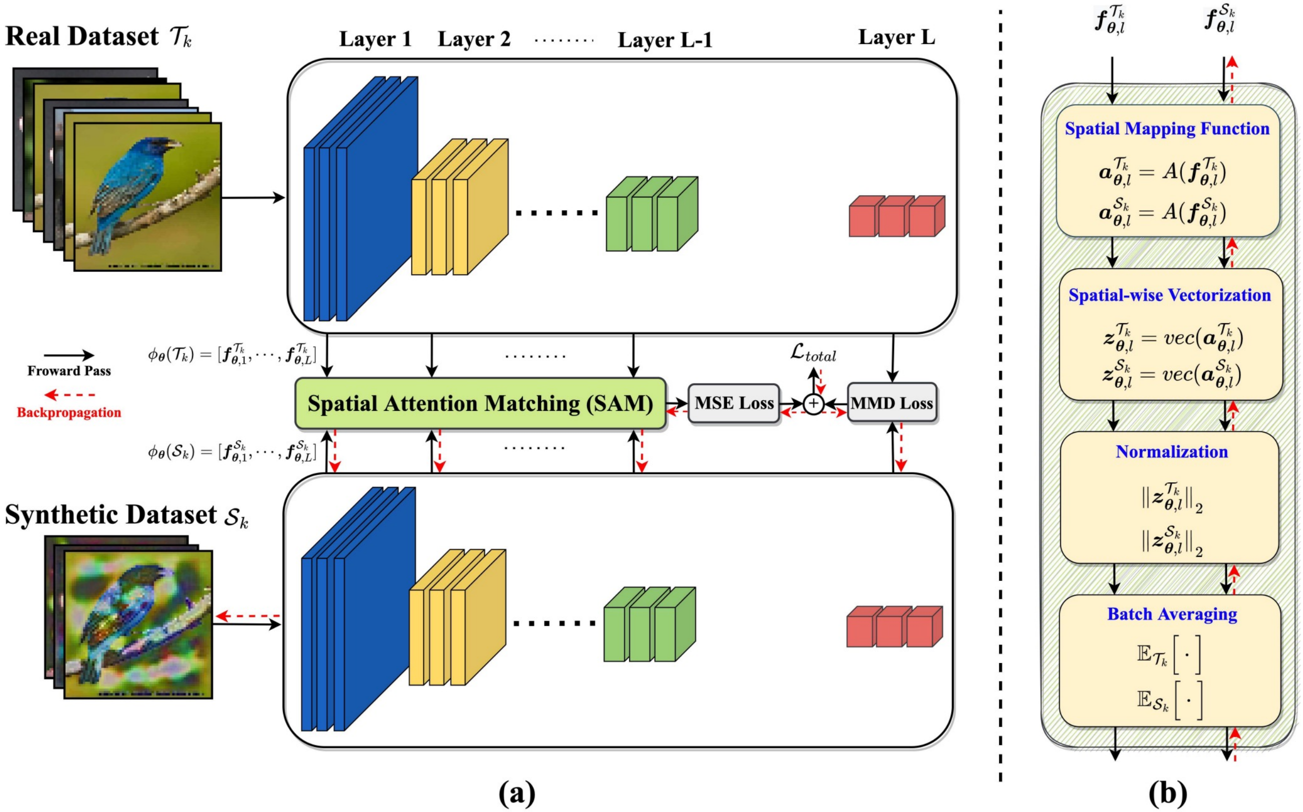
Previous Works (3)

To summarize:

- Performance matching is **short-sighted** and is **difficult** to optimize.
- Gradient matching generates **biased** samples.
- While distribution matching alleviate this, its performance is **lower.**
- Trajectory matching takes a **long** time and is quite **compute intensive.**

Less Efficient ...

Methodology (DataDAM)



(a) *Illustration of DataDAM method. DataDAM includes a Spatial Attention Matching (SAM) module and a complementary MMD loss to capture the dataset's distribution. (b) The internal architecture of the SAM module.*

Idea: Use attention to extract and match meaningful information from the intermediate features

Methodology (DataDAM)

Algorithm 1 Dataset Distillation with Attention Matching

Input: Real training dataset $\mathcal{T} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{|\mathcal{T}|}$
Required: Initialized synthetic samples for K classes, Deep neural network $\phi_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$, Probability distribution over randomly initialized weights $P_{\boldsymbol{\theta}}$, Learning rate $\eta_{\mathcal{S}}$, Task balance parameter λ, Number of training iterations I.

```
Initialize synthetic dataset S
for i = 1, 2, ··· , I do
    Sample θ from P_θ
    Sample mini-batch pairs B_k^T and B_k^S from the real
    and synthetic sets for each class k
    Compute L_SAM and L_MMD using Equations 2 and 3
    Calculate L = L_SAM + λ L_MMD
    Update the synthetic dataset using S ← S − η_S ∇_S L
end for
```

- Initialize synthetic dataset $\mathcal{S}$
- **for** $i = 1, 2, \cdots, I$ **do**
  - Sample $\boldsymbol{\theta}$ from $P_{\boldsymbol{\theta}}$
  - Sample mini-batch pairs $B_k^{\mathcal{T}}$ and $B_k^{\mathcal{S}}$ from the real and synthetic sets for each class k
  - Compute $\mathcal{L}_{\text{SAM}}$ and $\mathcal{L}_{\text{MMD}}$ using Equations 2 and 3
  - Calculate $\mathcal{L} = \mathcal{L}_{\text{SAM}} + \lambda\mathcal{L}_{\text{MMD}}$
  - Update the synthetic dataset using $\mathcal{S} \leftarrow \mathcal{S} - \eta_{\mathcal{S}}\nabla_{\mathcal{S}}\mathcal{L}$
- **end for**

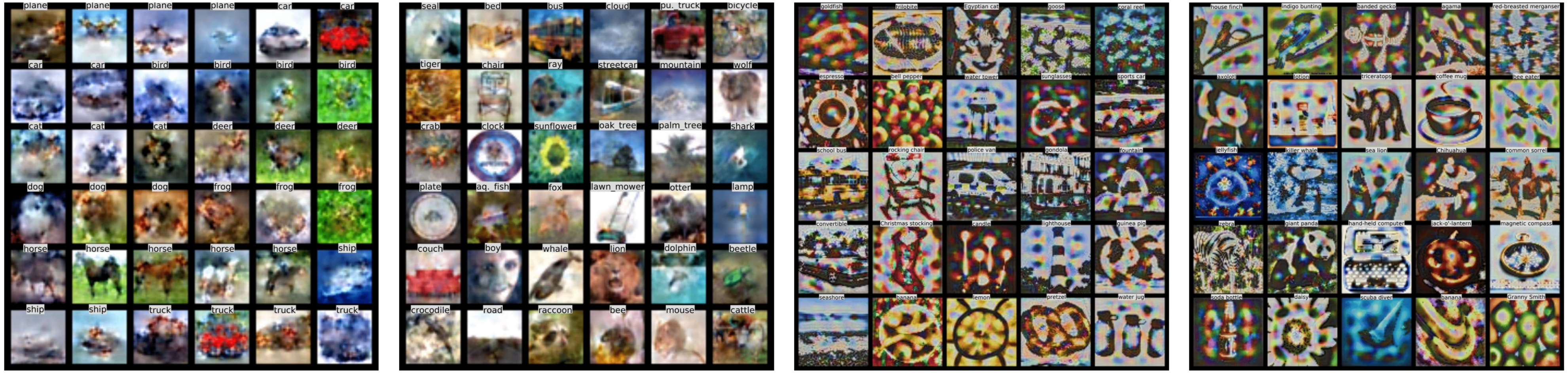
Output: Synthetic dataset $\mathcal{S} = \{(\boldsymbol{s}_i, y_i)\}_{i=1}^{|\mathcal{S}|}$

No Bi-level Optimization

Regularizing the Attention Matching

Visual Results

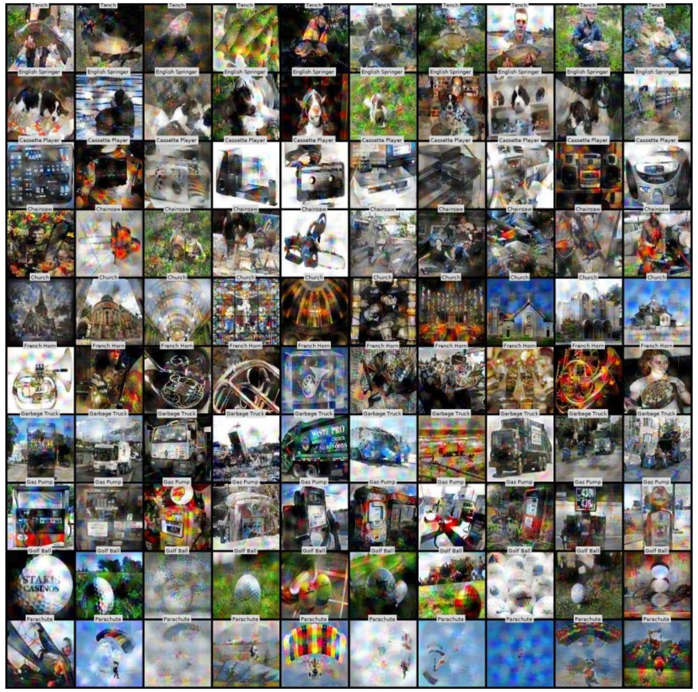
- Experiments were conducted in
  - CIFAR-10/100 (IPC10; Resolution 32x32)
  - Tiny-ImageNet and ImageNet (IPC1; Resolution 64x64)
- ConvNet was employed, which consists of several blocks, each containing 3 x 3 / 128 kernels, ReLU, 2 x 2 Average Pooling.

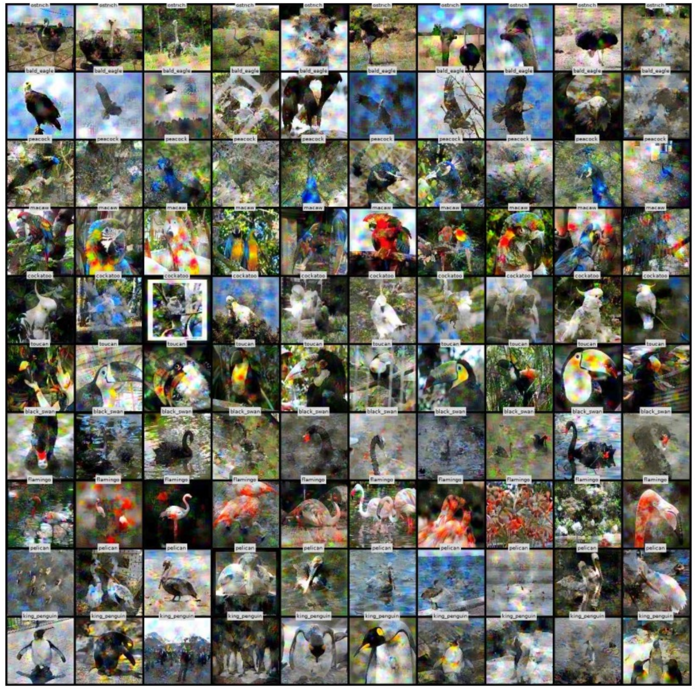


(a) CIFAR10 (b) CIFAR100 (c) Tiny ImageNet (d) ImageNet-1K

Visual Results (2)

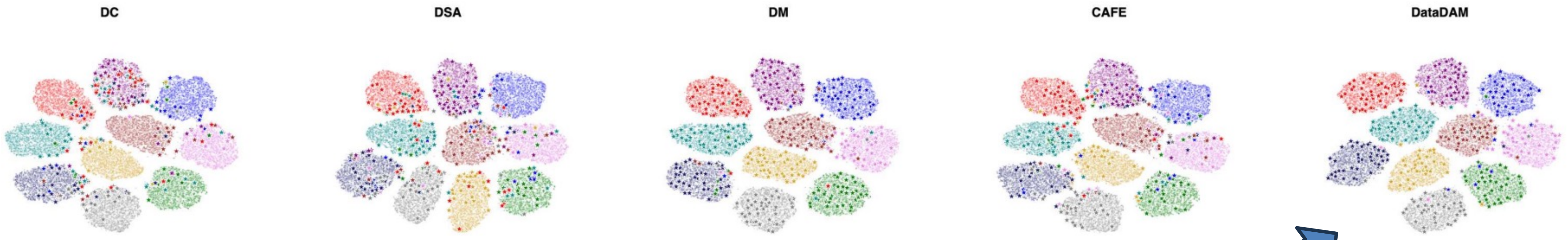
- Experiments were conducted in
  - ImageNet subsets (High Resolution 128x128)
- ImageNette (Left), ImageWoof (Center), ImageSquawk (Right) - IPC 10





Visual Results (3)

- Experiments were conducted on CIFAR-10 (IPC50):
  - TSNE visualization of Synthetic Data distribution (stars) dispersed over the original dataset



Better coverage of class embeddings
when using our Synthetic Dataset

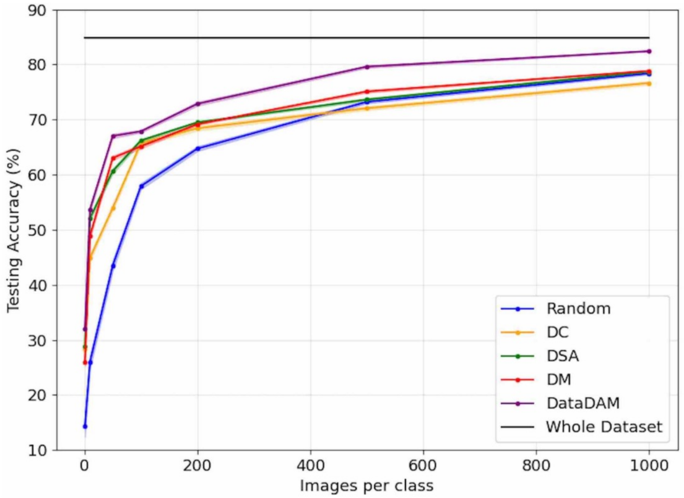
Experimental Results

Dataset	IPC	Coreset Selection: Random	Coreset Selection: K-Center	Training Set Synthesis: DD	DC	DSA	DM	CAFE	KIP	MTT	DataDAM	Whole Dataset
CIFAR-10	1	$14.4_{\pm2.0}$	$21.5_{\pm1.3}$	-	$28.3_{\pm0.5}$	$28.8_{\pm0.7}$	$26.0_{\pm0.8}$	$31.6_{\pm0.8}$	$29.8_{\pm1.0}$	$31.9_{\pm1.2}$	$\mathbf{32.0_{\pm1.2}}$	$84.8_{\pm0.1}$
	10	$26.0_{\pm1.2}$	$14.7_{\pm0.9}$	$36.8_{\pm1.2}$	$44.9_{\pm0.5}$	$52.1_{\pm0.5}$	$48.9_{\pm0.6}$	$50.9_{\pm0.5}$	$46.1_{\pm0.7}$	$\mathbf{56.4_{\pm0.7}}$	$54.2_{\pm0.8}$	
	50	$43.4_{\pm1.0}$	$27.0_{\pm1.4}$	-	$53.9_{\pm0.5}$	$60.6_{\pm0.5}$	$63.0_{\pm0.4}$	$62.3_{\pm0.4}$	$53.2_{\pm0.7}$	$65.9_{\pm0.6}$	$\mathbf{67.0_{\pm0.4}}$	
CIFAR-100	1	$4.2_{\pm0.3}$	$8.4_{\pm0.3}$	-	$12.8_{\pm0.3}$	$13.9_{\pm0.3}$	$11.4_{\pm0.3}$	$14.0_{\pm0.3}$	$12.0_{\pm0.2}$	$13.8_{\pm0.6}$	$\mathbf{14.5_{\pm0.5}}$	$56.2_{\pm0.3}$
	10	$14.6_{\pm0.5}$	$17.3_{\pm0.3}$	-	$25.2_{\pm0.3}$	$32.3_{\pm0.3}$	$29.7_{\pm0.3}$	$31.5_{\pm0.2}$	$29.0_{\pm0.3}$	$33.1_{\pm0.4}$	$\mathbf{34.8_{\pm0.5}}$	
	50	$30.0_{\pm0.4}$	$30.5_{\pm0.3}$	-	$30.6_{\pm0.6}$	$42.8_{\pm0.4}$	$43.6_{\pm0.4}$	$42.9_{\pm0.2}$	-	$42.9_{\pm0.3}$	$\mathbf{49.4_{\pm0.3}}$	
Tiny ImageNet	1	$1.4_{\pm0.1}$	$1.6_{\pm0.1}$	-	$5.3_{\pm0.1}$	$5.7_{\pm0.1}$	$3.9_{\pm0.2}$	-	-	$6.2_{\pm0.4}$	$\mathbf{8.3_{\pm0.4}}$	$37.6_{\pm0.4}$
	10	$5.0_{\pm0.2}$	$5.1_{\pm0.2}$	-	$12.9_{\pm0.1}$	$16.3_{\pm0.2}$	$12.9_{\pm0.4}$	-	-	$17.3_{\pm0.2}$	$\mathbf{18.7_{\pm0.3}}$	
	50	$15.0_{\pm0.4}$	$15.0_{\pm0.3}$	-	$12.7_{\pm0.4}$	$5.1_{\pm0.2}$	$25.3_{\pm0.2}$	-	-	$26.5_{\pm0.3}$	$\mathbf{28.7_{\pm0.3}}$	

Low and Medium Resolution Performance

Dataset	IPC	Random	DM	DataDAM	Whole Dataset
ImageNet-1K	1	$0.5_{\pm0.1}$	$1.3_{\pm0.1}$	$\mathbf{2.0_{\pm0.1}}$	$33.8_{\pm0.3}$
	2	$0.9_{\pm0.1}$	$1.6_{\pm0.1}$	$\mathbf{2.2_{\pm0.1}}$	
	10	$3.1_{\pm0.2}$	$5.7_{\pm0.1}$	$\mathbf{6.3_{\pm0.0}}$	
	50	$7.6_{\pm1.2}$	$11.4_{\pm0.9}$	$\mathbf{15.5_{\pm0.2}}$	
ImageNette	1	$23.5_{\pm4.8}$	$32.8_{\pm0.5}$	$\mathbf{34.7_{\pm0.9}}$	$87.4_{\pm1.0}$
	10	$47.7_{\pm2.4}$	$58.1_{\pm0.3}$	$\mathbf{59.4_{\pm0.4}}$	
ImageWoof	1	$14.2_{\pm0.9}$	$21.1_{\pm1.2}$	$\mathbf{24.2_{\pm0.5}}$	$67.0_{\pm1.3}$
	10	$27.0_{\pm1.9}$	$31.4_{\pm0.5}$	$\mathbf{34.4_{\pm0.4}}$	
ImageSquawk	1	$21.8_{\pm0.5}$	$31.2_{\pm0.7}$	$\mathbf{36.4_{\pm0.8}}$	$87.5_{\pm0.3}$
	10	$40.2_{\pm0.4}$	$50.4_{\pm1.2}$	$\mathbf{55.4_{\pm0.9}}$	

High Resolution Performance



Performance on CIFAR-10 over varying IPC

Experimental Results (2)

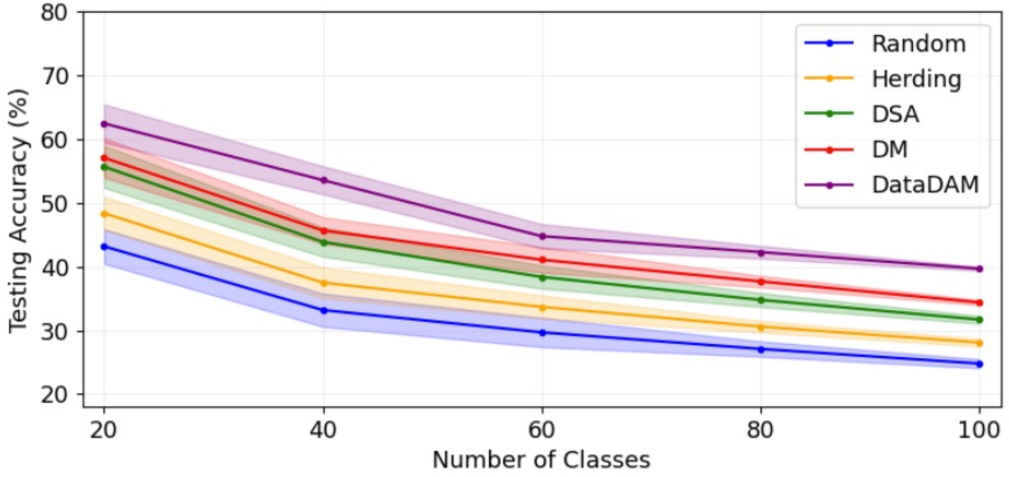
- Experiments were conducted on CIFAR-10 :
  - Cross-Architecture generalizations
  - Computational costs

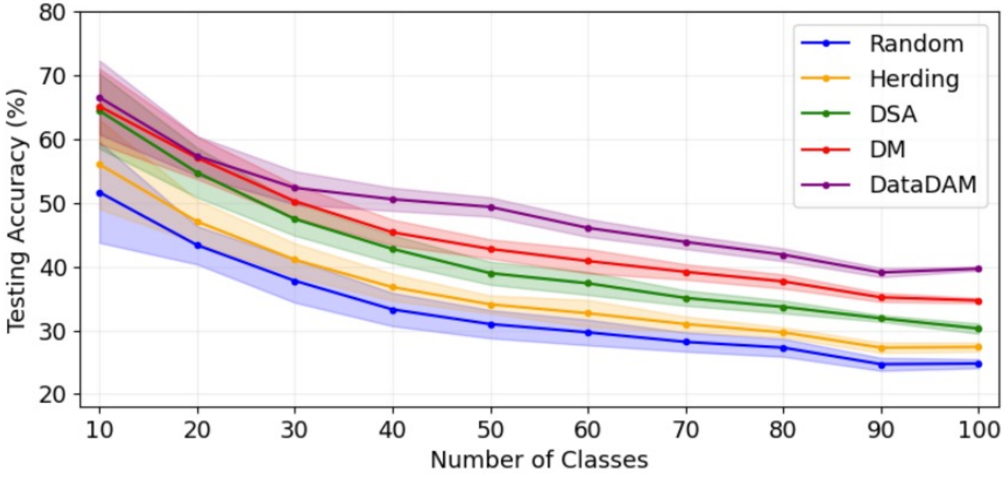
	T\E	AlexNet	VGG-11	ResNet-18
DC	ConvNet	$28.8_{\pm 0.7}$	$38.8_{\pm 1.1}$	$20.9_{\pm 1.0}$
CAFE	ConvNet	$43.2_{\pm 0.4}$	$48.8_{\pm 0.5}$	$43.3_{\pm 0.7}$
DSA	ConvNet	$53.7_{\pm 0.6}$	$51.4_{\pm 1.0}$	$47.8_{\pm 0.9}$
DM	ConvNet	$60.1_{\pm 0.5}$	$57.4_{\pm 0.8}$	$52.9_{\pm 0.4}$
MTT	ConvNet	$43.9_{\pm 0.9}$	$48.7_{\pm 1.3}$	$60.0_{\pm 0.7}$
DataDAM	ConvNet	$\mathbf{63.9}_{\pm \mathbf{0.9}}$	$\mathbf{64.8}_{\pm \mathbf{0.5}}$	$\mathbf{60.2}_{\pm \mathbf{0.7}}$

Method	run time(sec)			GPU memory(MB)		
	IPC1	IPC10	IPC50	IPC1	IPC10	IPC50
DC	$0.16_{\pm 0.0}$	$3.31_{\pm 0.0}$	$15.74_{\pm 0.1}$	3515	3621	4527
DSA	$0.22_{\pm 0.0}$	$4.47_{\pm 0.1}$	$20.13_{\pm 0.6}$	3513	3639	4539
DM	$\mathbf{0.08}_{\pm \mathbf{0.0}}$	$\mathbf{0.08}_{\pm \mathbf{0.0}}$	$\mathbf{0.08}_{\pm \mathbf{0.0}}$	3323	3455	**3605**
MTT	$0.36_{\pm 0.2}$	$0.40_{\pm 0.2}$	OOM	**2711**	8049	OOM
DataDAM	$\mathbf{0.09}_{\pm \mathbf{0.0}}$	$\mathbf{0.08}_{\pm \mathbf{0.0}}$	$0.16_{\pm 0.0}$	3452	**3561**	3724

Applications

- Experiments were conducted on CIFAR-10 (IPC50):
  - Applications in **Continual Learning**: 5-step (Left) 10-step (Right)





Applications (2)

- Experiments were conducted on CIFAR-10 (IPC50):
  - Applications in **Neural Architecture Search**
    - (Left) CIFAR-10 using full search-space
    - (Right) CIFAR1-10 using Top 20% of search space

	Random	DM	CAFE	Ours	Early-stopping	Whole Dataset
Performance	88.9	87.2	83.6	**89.0**	88.9	89.2
Correlation	0.70	0.71	0.59	**0.72**	0.69	1.00
Time cost (min)	206.4	206.6	206.4	206.4	206.2	5168.9
Storage (imgs)	**500**	**500**	**500**	**500**	5×10^4	5×10^4

	Random	DM	CAFE	Ours	Early-stopping	Whole Dataset
Performance	88.9	87.2	83.6	**89.0**	88.9	89.2
Correlation	0.44	0.51	0.36	**0.69**	0.64	1.00
Time cost (min)	33.0	32.2	**30.7**	34.8	37.1	5168.9
Storage (imgs)	**500**	**500**	**500**	**500**	5×10^4	5×10^4

Thank You for watching!