Data Efficiency

EECE695D: Efficient ML Systems

Spring 2025

Training cost

- Roughly, the training cost is:

$$f(\text{model size}, \text{dataset size}, \cdots)$$

- Example.

$$\text{Compute} = (\#\text{data}) \times (\#\text{epochs}) \times (\text{Fwd FLOPs} + \text{Bwd FLOPs})$$

$$\text{Duration} = (\#\text{data}) \times (\#\text{epochs}) \times (\text{Processing Time/sample})$$

Paradigm shift

- In the past, the number of usable data was scarce
    - **Why?** Labeling cost was expensive
    - **Strategy.** Increase the epochs and see data many times

- <u>Example.</u>
    - ResNet. 90 epochs (later works extend it to 600 epochs)
    - DeiT. 300 epochs
    - BERT. 40 epochs

Paradigm shift

- Nowadays, one can utilize **much more data**
  - **Why?** Self-supervised pre-training techniques
  - **Strategy.** Reduce the data redundancy by reducing epochs
    - Computation is the new bottleneck
- Example. GPT-3 uses 0.8 epoch, on average

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Observation

- Some data have notably higher quality than others
- Example. Textbooks are all you need (2023)
  - Textbook-quality samples enable training powerful models with smaller model size and dataset
  - Used GPT-4 as a filter for telling the quality

Date	Model	Model size (Parameters)	Dataset size (Tokens)	HumanEval (Pass@1)	MBPP (Pass@1)
2021 Jul	Codex-300M [CTJ+21]	300M	100B	13.2%	-
2021 Jul	Codex-12B [CTJ+21]	12B	100B	28.8%	-
2022 Mar	CodeGen-Mono-350M [NPH+23]	350M	577B	12.8%	-
2022 Mar	CodeGen-Mono-16.1B [NPH+23]	16.1B	577B	29.3%	35.3%
2022 Apr	PaLM-Coder [CND+22]	540B	780B	35.9%	47.0%
2022 Sep	CodeGeeX [ZXZ+23]	13B	850B	22.9%	24.4%
2022 Nov	GPT-3.5 [Ope23]	175B	N.A.	47%	-
2022 Dec	SantaCoder [ALK+23]	1.1B	236B	14.0%	35.0%
2023 Mar	GPT-4 [Ope23]	N.A.	N.A.	67%	-
2023 Apr	Replit [Rep23]	2.7B	525B	21.9%	-
2023 Apr	Replit-Finetuned [Rep23]	2.7B	525B	30.5%	-
2023 May	CodeGen2-1B [NHX+23]	1B	N.A.	10.3%	-
2023 May	CodeGen2-7B [NHX+23]	7B	N.A.	19.1%	-
2023 May	StarCoder [LAZ+23]	15.5B	1T	33.6%	52.7%
2023 May	StarCoder-Prompted [LAZ+23]	15.5B	1T	40.8%	49.5%
2023 May	PaLM 2-S [ADF+23]	N.A.	N.A.	37.6%	50.0%
2023 May	CodeT5+ [WLG+23]	2B	52B	24.2%	-
2023 May	CodeT5+ [WLG+23]	16B	52B	30.9%	-
2023 May	InstructCodeT5+ [WLG+23]	16B	52B	35.0%	-
2023 Jun	WizardCoder [LXZ+23]	16B	1T	57.3%	51.8%
2023 Jun	**phi-1**	1.3B	7B	50.6%	55.5%

Educational values deemed by the filter

High educational value

```python
import torch
import torch.nn.functional as F

def normalize(x, axis=-1):
    """Performs L2-Norm."""
    num = x
    denom = torch.norm(x, 2, axis, keepdim=True)
    .expand_as(x) + 1e-12
    return num / denom

def euclidean_dist(x, y):
    """Computes Euclidean distance."""
    m, n = x.size(0), y.size(0)
    xx = torch.pow(x, 2).sum(1, keepdim=True).
    expand(m, n)
    yy = torch.pow(x, 2).sum(1, keepdim=True).
    expand(m, m).t()
    dist = xx + yy - 2 * torch.matmul(x, y.t())
    dist = dist.clamp(min=1e-12).sqrt()
    return dist

def cosine_dist(x, y):
    """Computes Cosine Distance."""
    x = F.normalize(x, dim=1)
    y = F.normalize(y, dim=1)
    dist = 2 - 2 * torch.mm(x, y.t())
    return dist
```

Low educational value

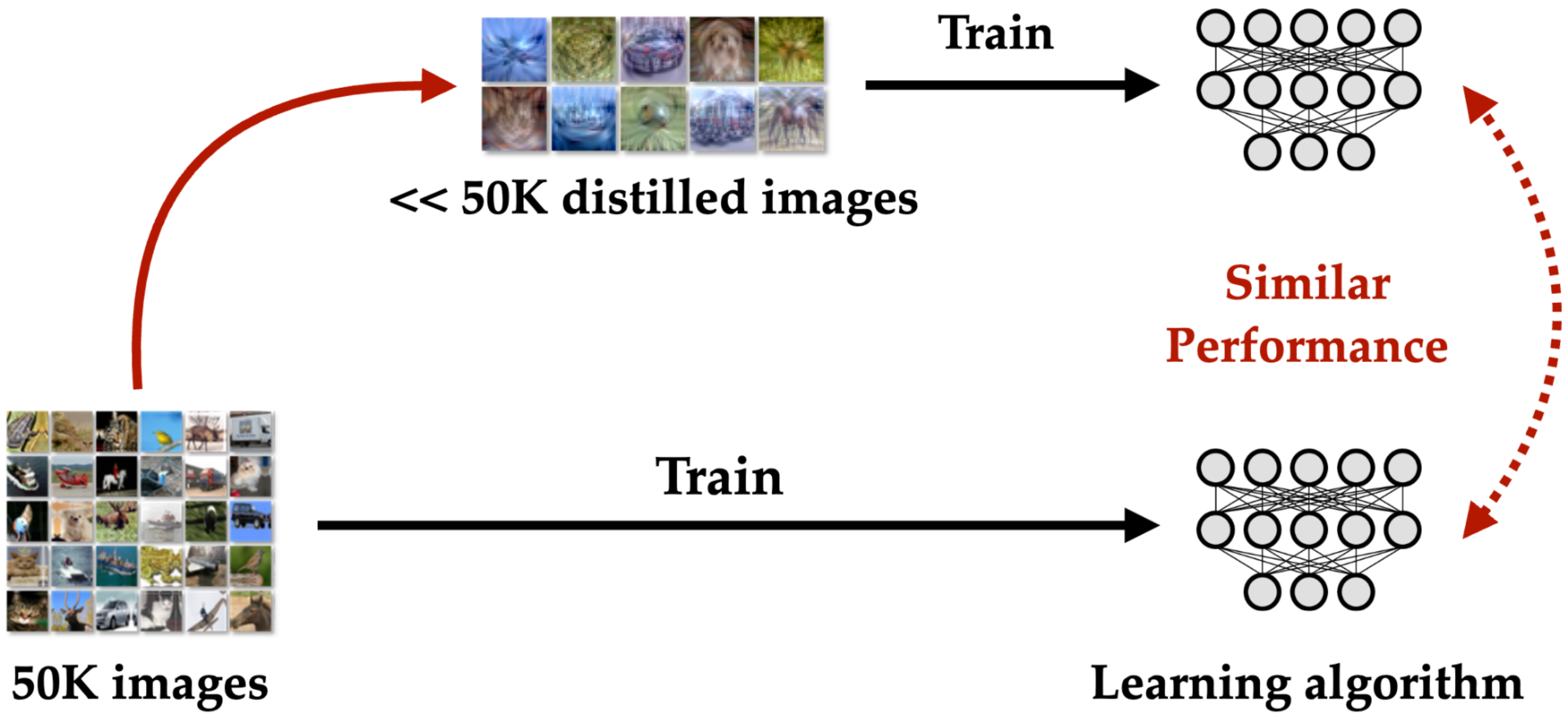
```python
import re
import typing
...

class Default(object):
    def __init__(self, vim: Nvim) -> None:
        self._vim = vim
        self._denite: typing.Optional[SyncParent]
    = None
        self._selected_candidates: typing.List[int
    ] = []
        self._candidates: Candidates = []
        self._cursor = 0
        self._entire_len = 0
        self._result: typing.List[typing.Any] = []
        self._context: UserContext = {}
        self._bufnr = -1
        self._winid = -1
        self._winrestcmd = ''
        self._initialized = False
        self._winheight = 0
        self._winwidth = 0
        self._winminheight = -1
        self._is_multi = False
        self._is_async = False
        self._matched_pattern = ''
        ...
```

Gunasekar et al., “Textbooks are all you need,” arXiv 2023

Key questions

- Given a large dataset, how can we automatically construct a **new dataset**, so that training with the dataset ensures **high quality of the trained model**?
    - Can we construct new data in a scalable way?
    - Distributional shift?
    - Synthesize or not?
    - Pick samples, or set?



Sachdeva and McAuley, "Data Distillation: A Survey," TMLR 2023

Basic ideas

Formalism

- Suppose that we have a dataset $D = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$
- We use a learning algorithm $A(\cdot)$ which finds a parameter given the dataset

$$\hat{\theta} = A(D)$$

- **Goal.** Find another dataset $D' = \{\mathbf{z}'_1, \ldots, \mathbf{z}'_n\}$ such that
    - $n \ll N$
    - $L(A(D)) \approx L(A(D'))$

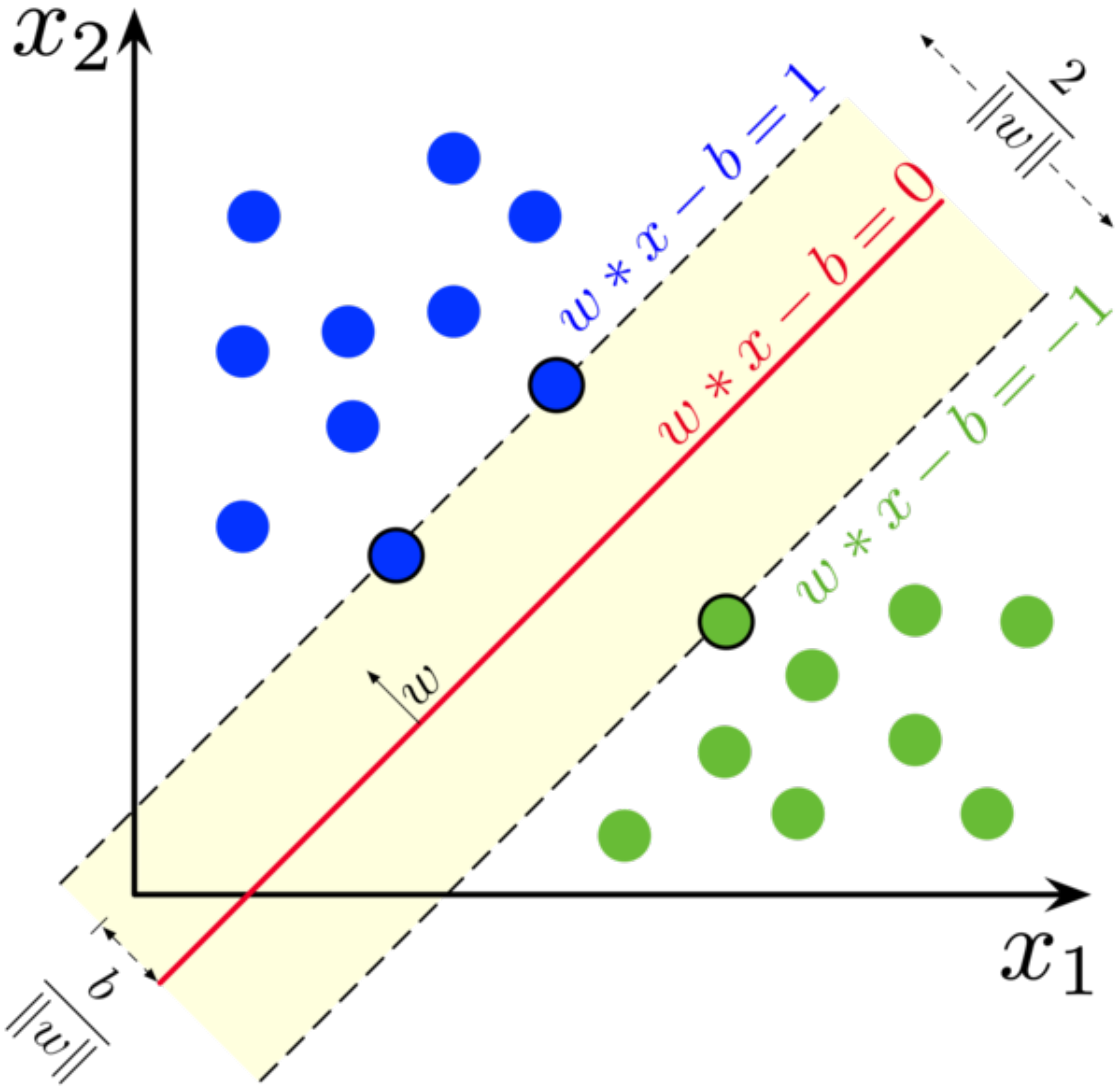
Terminologies

- **Data pruning.** Select a subset, i.e., $D' \subseteq D$
- **Data curation.** Same, but involves human judgement
- **Dataset distillation.** Allows data to be synthetic, thus $D' \nsubseteq D$
    - Also called "dataset condensation"
- **Data valuation.** Measures the importance of each $d \in D$
    - Can be used for data pruning, via top-k

(theoreticians might call these "coresets")

Proof of Concept

- Recall the **support vector machine (SVM)**
  - Margin maximizer
  - Determined by support vectors,
    i.e., samples on the margin
  - Can keep only difficult samples
    to perfectly reconstruct the classifier
  - <u>Note</u>. Not in deep learning, as we need
    samples for feature learning



Algorithms

- Data valuation
  - Leave-one-out, Influence function, Data Shapley
- Data pruning
  - Difficulty-based pruning
- Dataset distillation
  - Meta-Learning, Gradient Matching, Trajectory Matching, Distribution Matching

Data valuation

Data valuation

- Measure how much a sample affects the training

- For instance, consider the **leave-one-out (LOO)** error

$$v(\mathbf{z}; D) = L(A(D \backslash \mathbf{z})) - L(A(D))$$

  - Expensive to measure
    - Requires at least $(N+1)$ full training
  - Requires some easy-to-compute proxy…

Influence function

- Assume that we are using ERM algorithm, with the loss

$$L(D;\theta) = \sum_{\mathbf{z}\in D} L(\mathbf{z};\theta)$$

- **Question.** What if some $\mathbf{z} \in D$ has been upweighted by ϵ?
  - Then, we get the parameter

$$\hat{\theta}_{\mathbf{z},\epsilon} = \mathrm{argmin}_{\theta} \left(L(D;\theta) + \epsilon L(\mathbf{z};\theta)\right)$$

  instead of the original parameter $\hat{\theta} = \hat{\theta}_{\mathbf{z},0}$.

Koh and Liang, "Understanding black-box predictions via influence function," ICML 2017

Influence function

- **Definition.** The **influence function** of the sample $\mathbf{z}$ on parameter is:

$$I_{\text{param}}(\mathbf{z}) = \lim_{\epsilon \to 0^+} \frac{\hat{\theta}_{\mathbf{z},\epsilon} - \hat{\theta}}{\epsilon}$$

  - Using the fact that $\hat{\theta}$ is the argmin, we get

$$I_{\text{param}}(\mathbf{z}) = -H_{\hat{\theta}}^{-1} \nabla_\theta L(\mathbf{z}; \hat{\theta})$$

Koh and Liang, "Understanding black-box predictions via influence function," ICML 2017

Influence function

- Similarly, we have influence function on the loss as:

$$I_{\text{loss}}(\mathbf{z}, \mathbf{z}_{\text{test}}) = \lim_{\epsilon \to 0^+} \frac{L(\mathbf{z}_{\text{test}}; \hat{\theta}_{\mathbf{z},\epsilon}) - L(\mathbf{z}_{\text{test}}; \hat{\theta})}{\epsilon}$$

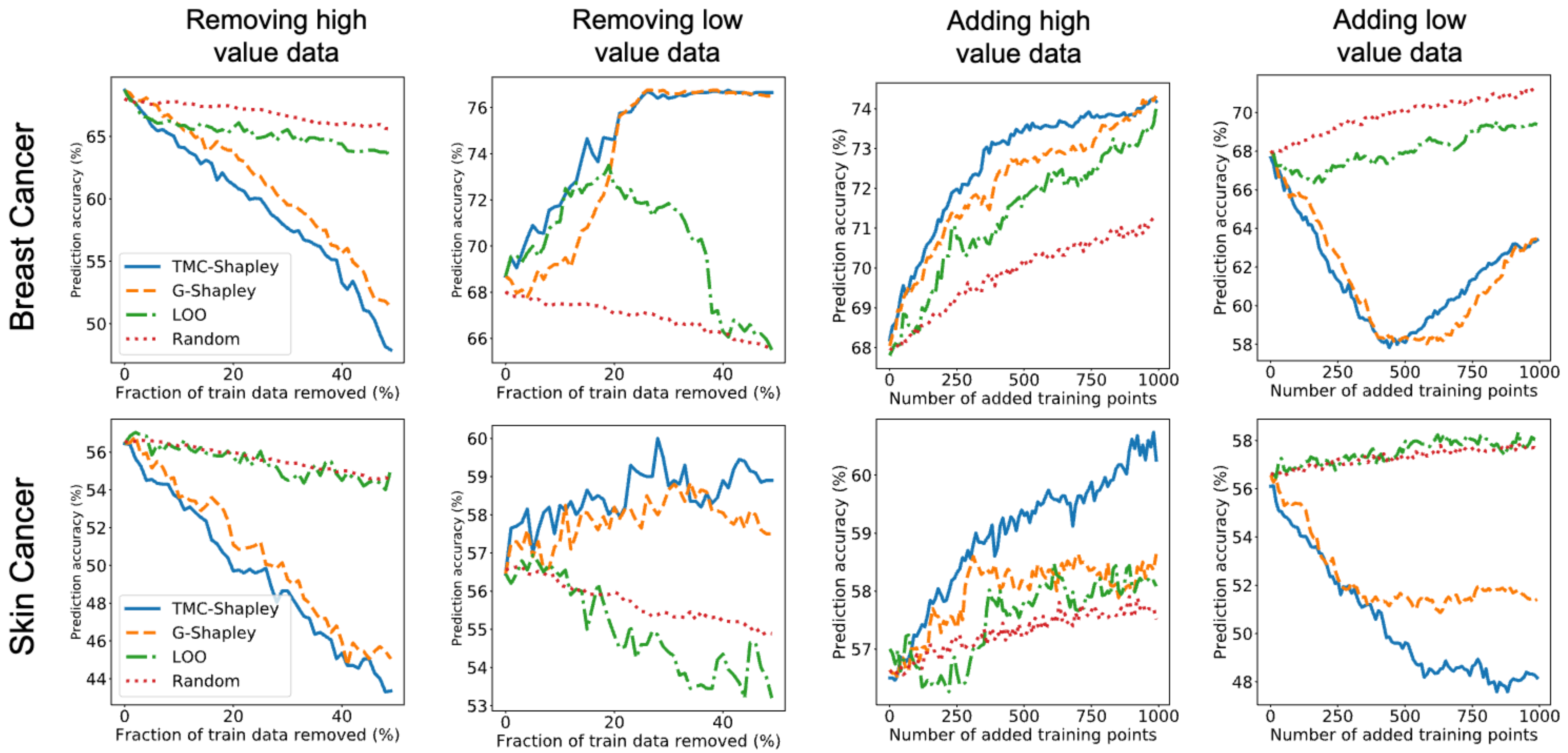
$$= -\nabla_\theta L(\mathbf{z}_{\text{test}}; \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta L(\mathbf{z}; \hat{\theta})$$

- Fortunately, this is much easier to compute

Koh and Liang, "Understanding black-box predictions via influence function," ICML 2017

Further readings

- Influence function is good for D, but maybe not for any $S \subseteq D$
  - **Data Shapley** addresses this problem
    - https://proceedings.mlr.press/v97/ghorbani19c/ghorbani19c.pdf
  - However, Data Shapley remains very costly to approximate

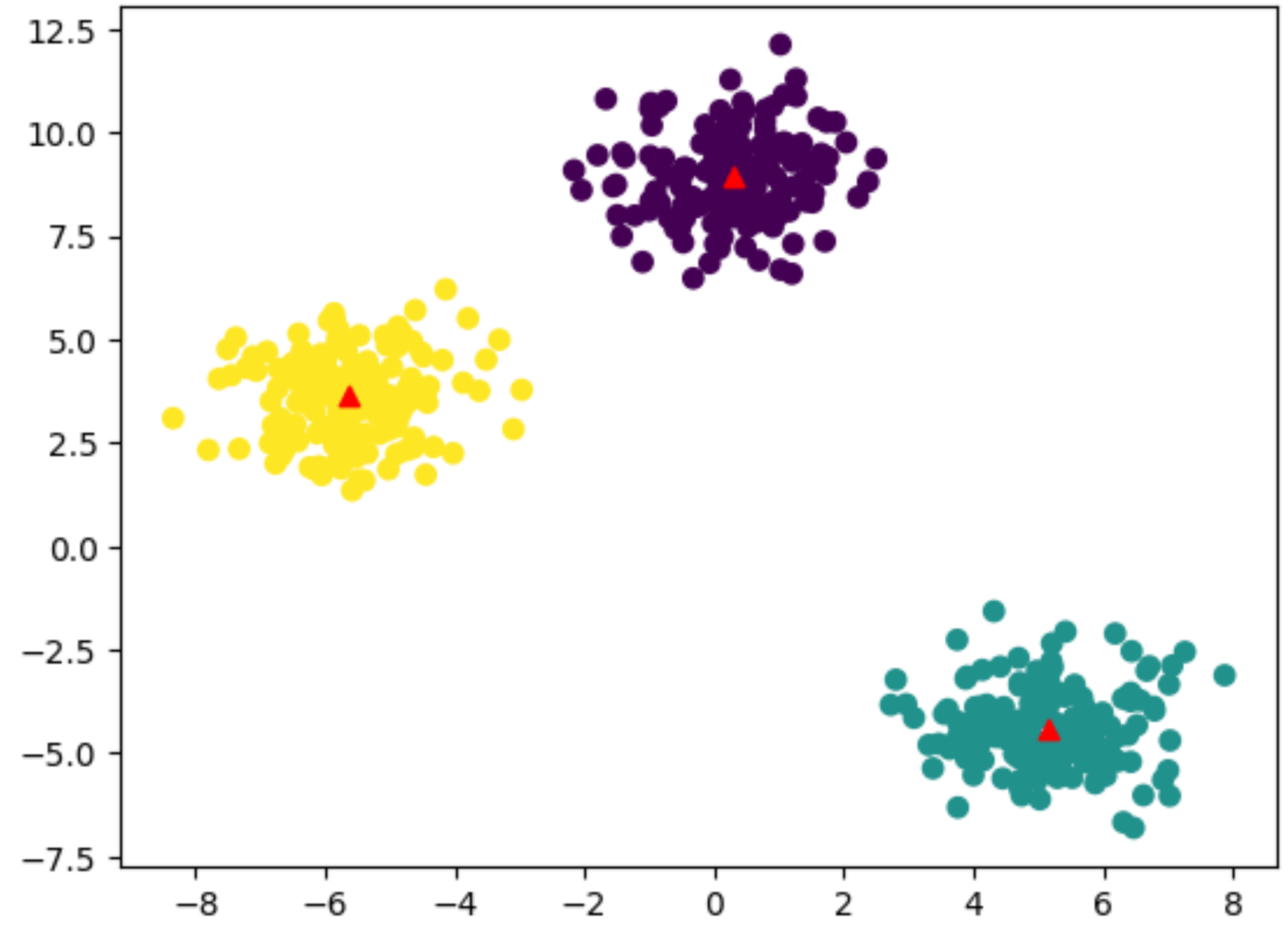


Ghobarni and Zou, “Data Shapley: Equitable Valuation of Data for Machine Learning,” ICML 2019

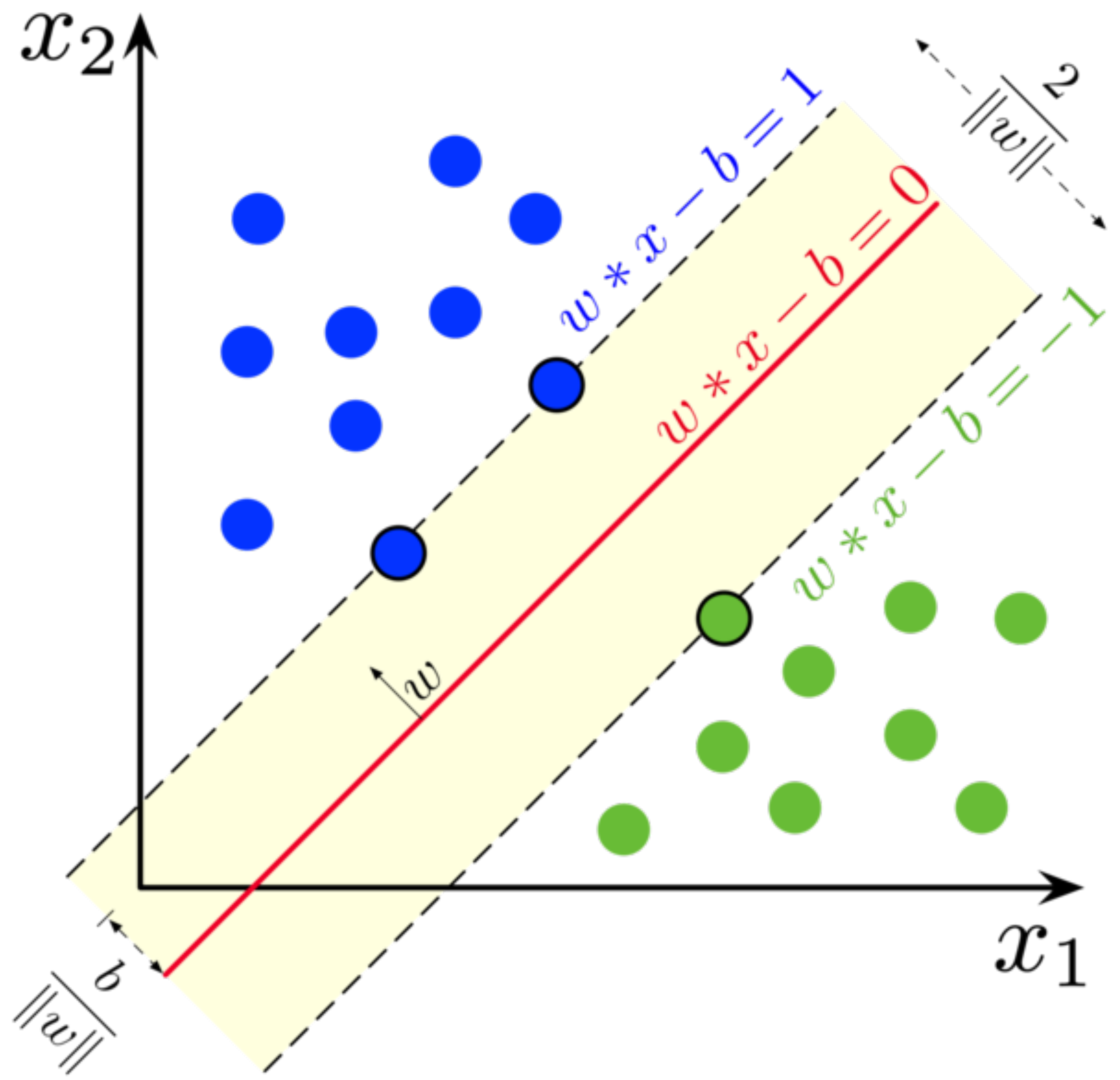
Data pruning

Data pruning

- Will only briefly discuss **difficulty-based** pruning
  - In particular, the results of Sorcher et al. (2022)



- Long-standing dispute:
  - Keep easy examples
    - Learning "prototype," e.g., K-Means
  - Keep hard examples
    - Like the case of SVM



Sorcher et al., "Beyond neural scaling laws: Beating power law scaling via data pruning," NeurIPS 2022

Data pruning

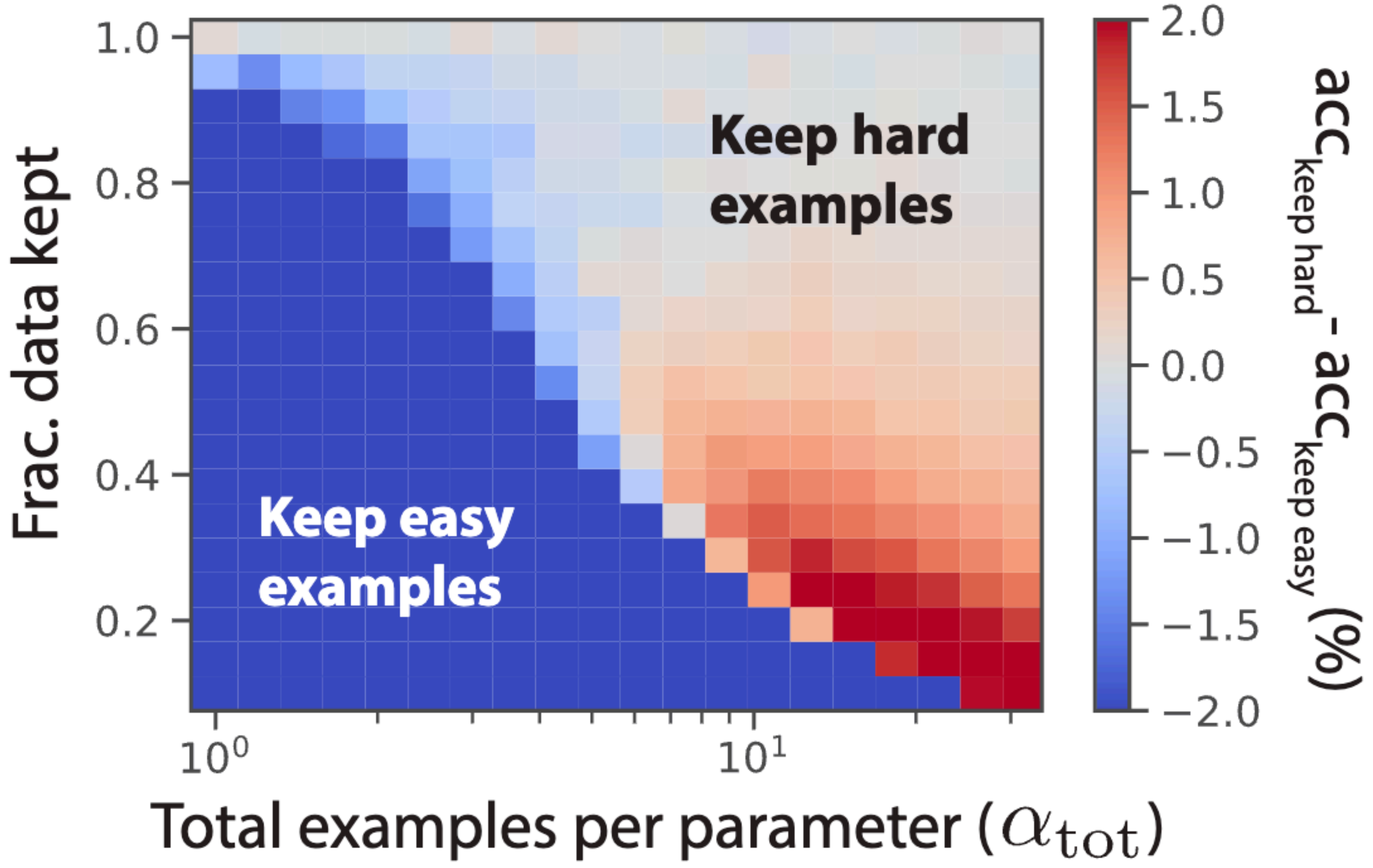
- Suppose that we have a self-supervised feature map $\Phi(\cdot)$.
    - e.g., SWaV
- We measure the sample difficulty by:
    - Conduct K-means clustering with $\Phi(\mathbf{z}_1), \ldots, \Phi(\mathbf{z}_N)$
    - Difficulty is the cosine distance to the centroid

Sorcher et al., "Beyond neural scaling laws: Beating power law scaling via data pruning," NeurIPS 2022

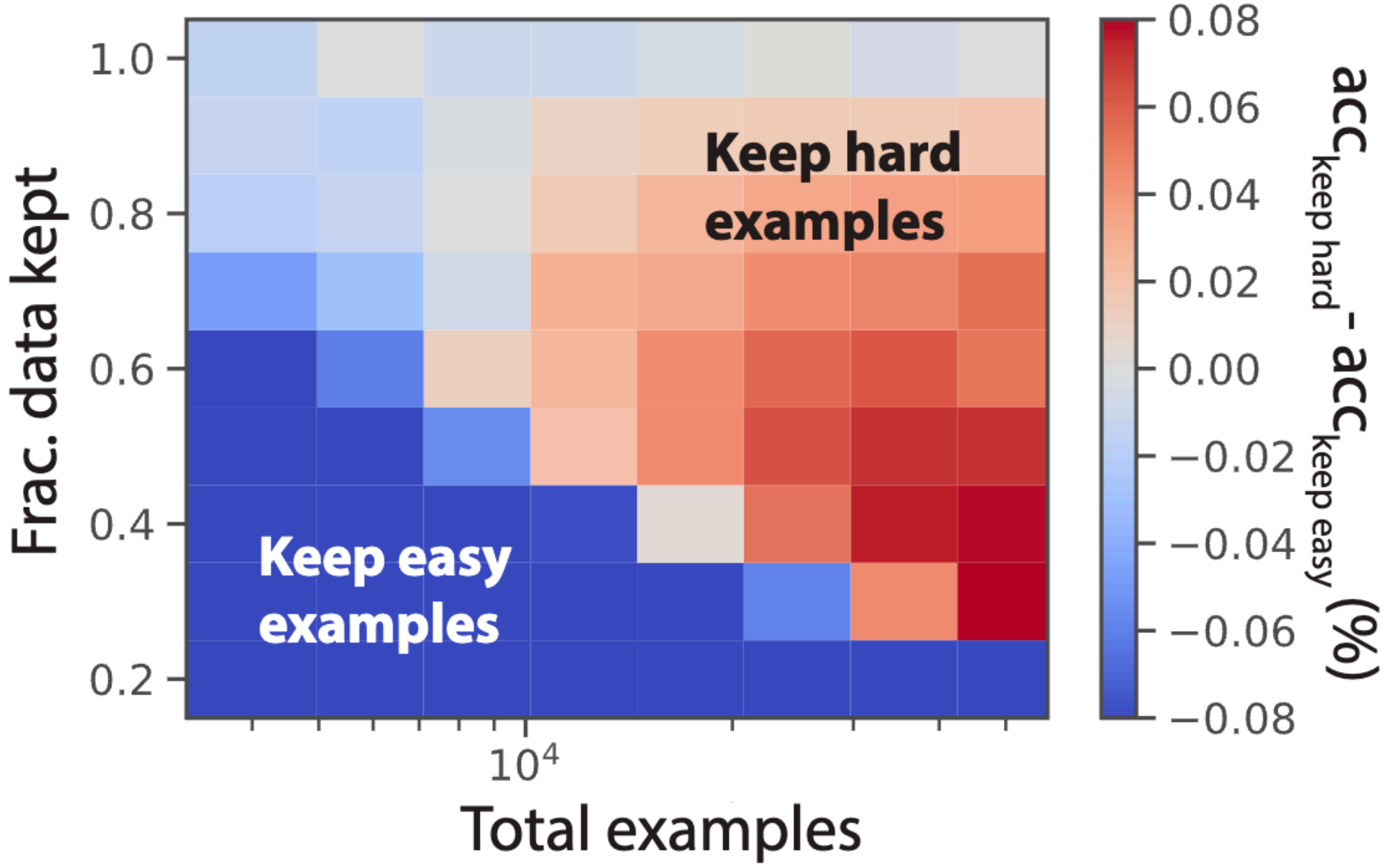
Data pruning

- **Observation.** A clear phase-transition (with some theory in paper)
  - <u>Abundant data, small model, or low sparsity</u>. Keep hard examples
  - <u>Scarce data, large model, or high sparsity</u>. Keep easy examples

Perceptron in teacher-student setting



ResNet18 on CIFAR-10



Sorcher et al., "Beyond neural scaling laws: Beating power law scaling via data pruning," NeurIPS 2022

Dataset distillation

Approaches

- Allows data to be synthetic, i.e., $D' \not\subseteq D$

- Meta-learning
- Gradient matching
- Trajectory matching
- Distribution matching

Meta-learning

- **Idea.** Use the full dataset as the validation set
  - By training on some synthetic set D', we wish to minimize the loss on the original dataset:

$$\min_{D'} L(A(D'); D)$$

    - e.g., update pixels of randomly initialized images in D'
  - Solvable via MAML-like bi-level optimization algorithms

Wang et al., "Dataset distillation," arXiv 2018

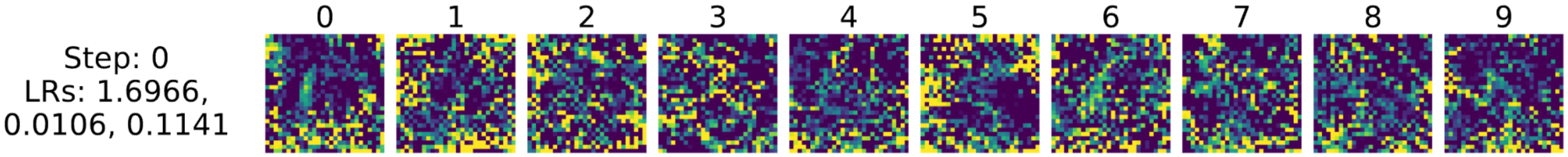
Meta-learning

- Initialize $D' = \{\mathbf{z}'_i\}_{i=1}^n$
- **Outer loop:**
  - Sample a batch of original data $B = \{\mathbf{z}_j\}$
  - Sample a batch of initial weights $\theta_0^{(k)}$
  - **Inner loop:** for each initial weight $\theta_0^{(k)}$
    - Update one step with D'
    - Evaluate loss on B
  - Update compressed dataset, with the loss summed over j

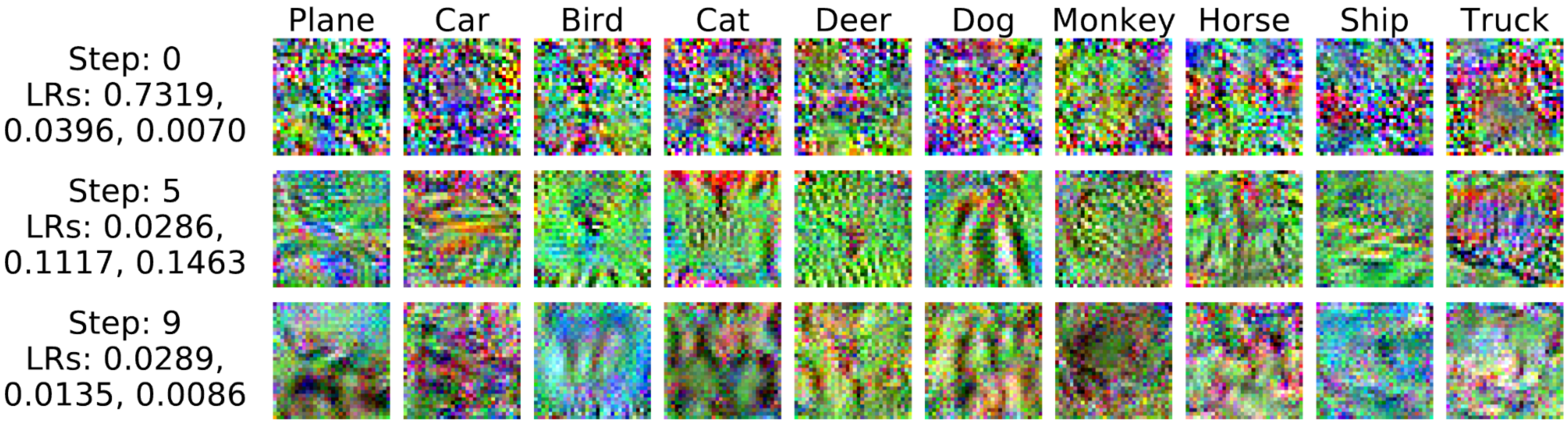
Wang et al., "Dataset distillation," arXiv 2018

Meta-learning

- **Result.** One can train a model, even with one image per class:
- When starting from a fixed initialization



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

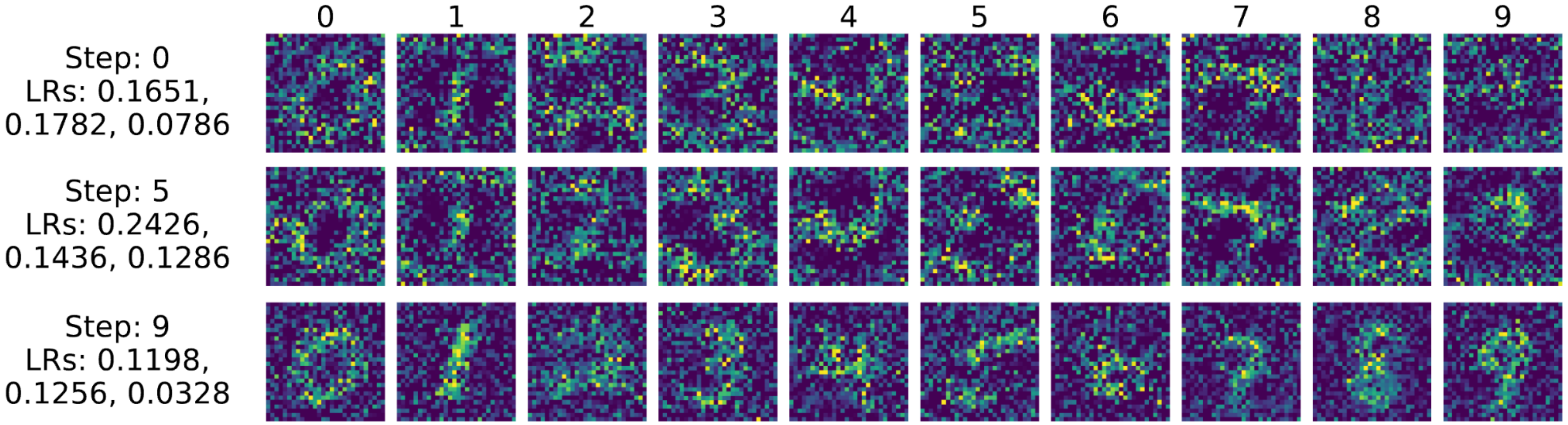


(b) CIFAR10. These distilled images train a fixed initialization from 8.82% test accuracy to 54.03%.

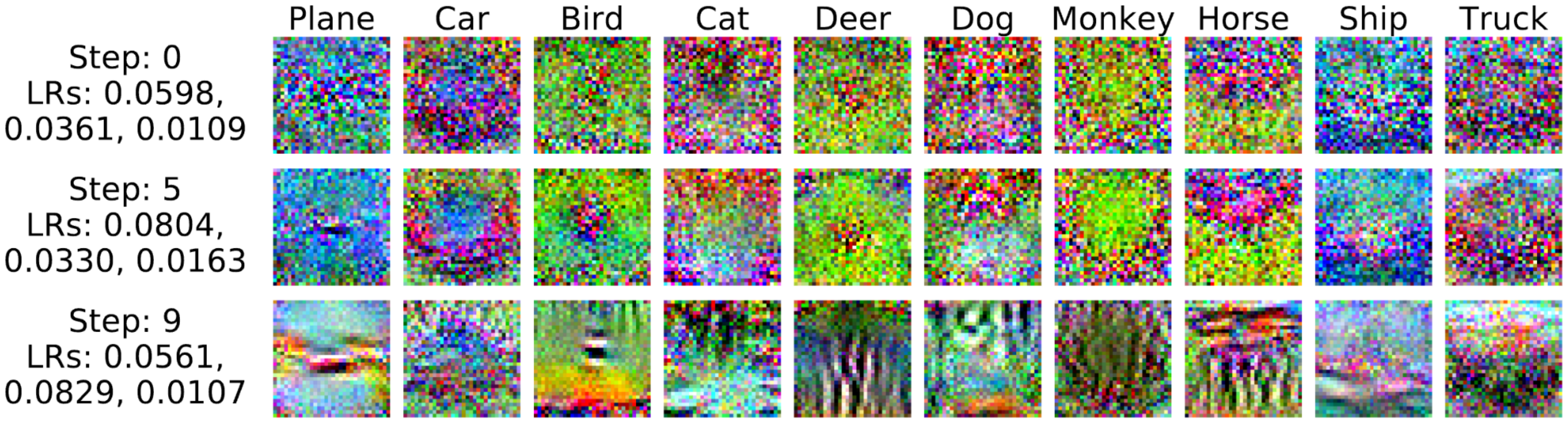
Wang et al., "Dataset distillation," arXiv 2018

Meta-learning

- When starting from a random initialization
  - A bit more semantic, but lower accuracy



(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.

Wang et al., "Dataset distillation," arXiv 2018

Further readings

- Combining data augmentation
  - https://proceedings.mlr.press/v139/zhao21a.html
- Shared information between classes
  - https://arxiv.org/abs/2206.02916
- NTK kernel for Meta-learning
  - https://arxiv.org/abs/2011.00050

Wang et al., "Dataset distillation," arXiv 2018

Gradient matching

- **Idea.** Gradient from D' should be similar to gradient from D

$$\nabla_\theta L(\theta; D) \approx \nabla_\theta L(\theta; D')$$

  - Needs to hold for all θ in the learning trajectory (when training with D'):

$$\min_{D'} \mathbb{E}\Big[\sum_{t=0}^{T} \mathrm{dist}\Big(\nabla_\theta L(A_t(D'); D), \nabla_\theta L(A_t(D'); D')\Big)\Big]$$

    - $\mathrm{dist}(\cdot, \cdot)$ can be some distance metric
    - A_t denotes the t-step updated version
    - Gradient is measured class-wise

Zhao et al., "Dataset condensation with gradient matching," ICLR 2021

Gradient matching

- Initialize D'
- **Outer loop:**
  - Initialize the model weight
  - **Inner loop:** For $t = 0, \ldots, T$
    - For each class,
      - Sample original data batch B and synthetic data batch B'
      - Compute gradients g and g'
      - Update synthetic data based on $\mathrm{dist}(g, g')$
    - Update model weight

Zhao et al., "Dataset condensation with gradient matching," ICLR 2021

Gradient matching

- **Result.** Interestingly, very semantically aligned

Zhao et al., "Dataset condensation with gradient matching," ICLR 2021

Gradient matching

- Also very transferable between architectures

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

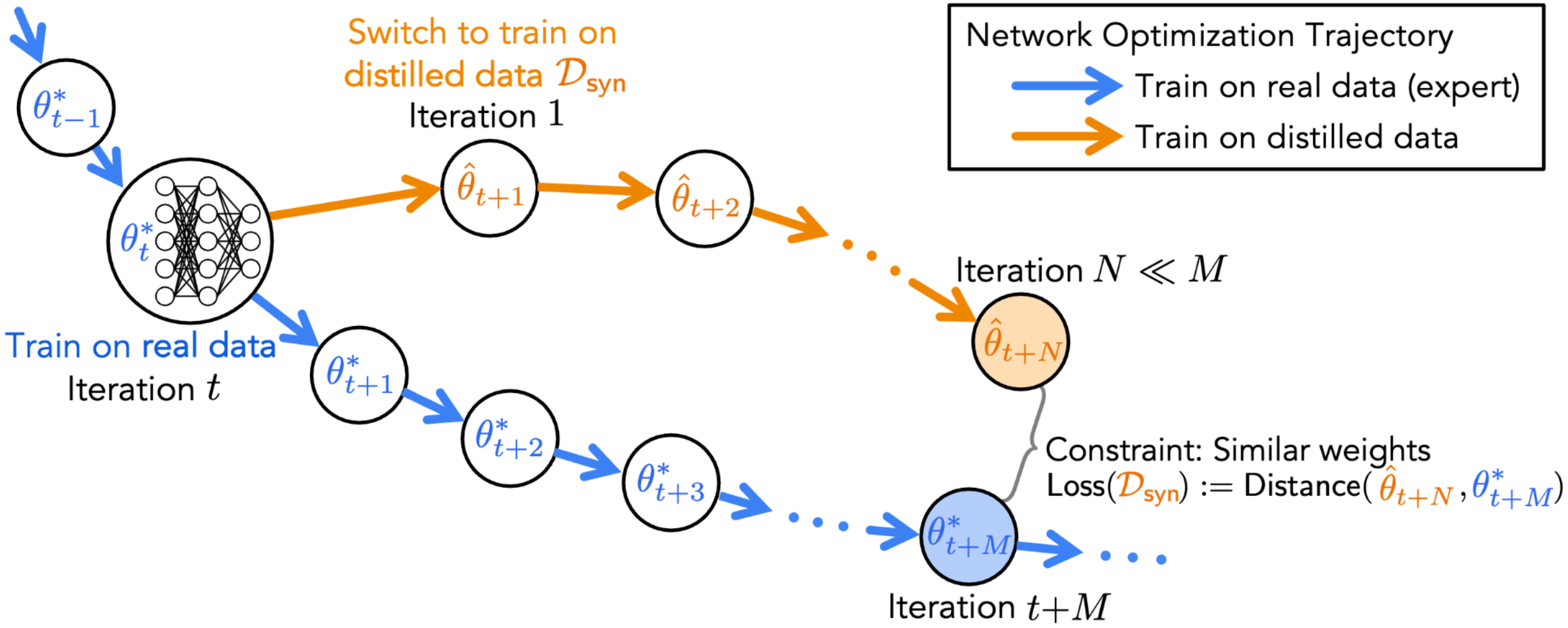
Zhao et al., "Dataset condensation with gradient matching," ICLR 2021

Further readings

- Class contrastive signals
    - https://arxiv.org/abs/2202.02916
- Less storage budget, by considering data regularity
    - https://arxiv.org/abs/2205.14959

Trajectory matching

- **Idea.** Match the trajectory itself, rather than gradients
  - Start at some model trained on original data for some steps:
    - Train on D for M steps
    - Train on D' for N steps



Cazenavette et al., "Dataset distillation by matching training trajectories," CVPR 2022

Trajectory matching

- More concretely, minimize the **normalized distance**:

$$\min_{D'} \mathbb{E}\left[\sum_{t=0}^{T-M} \frac{\mathrm{dist}(A_{t+M}(D), A_{t+N}(D'))}{\mathrm{dist}(A_{t+M}(D), A_t(D))}\right]$$

  - Can consider much longer horizon than previous approaches
  - Can utilize pre-computed trajectories for original data

Cazenavette et al., "Dataset distillation by matching training trajectories," CVPR 2022

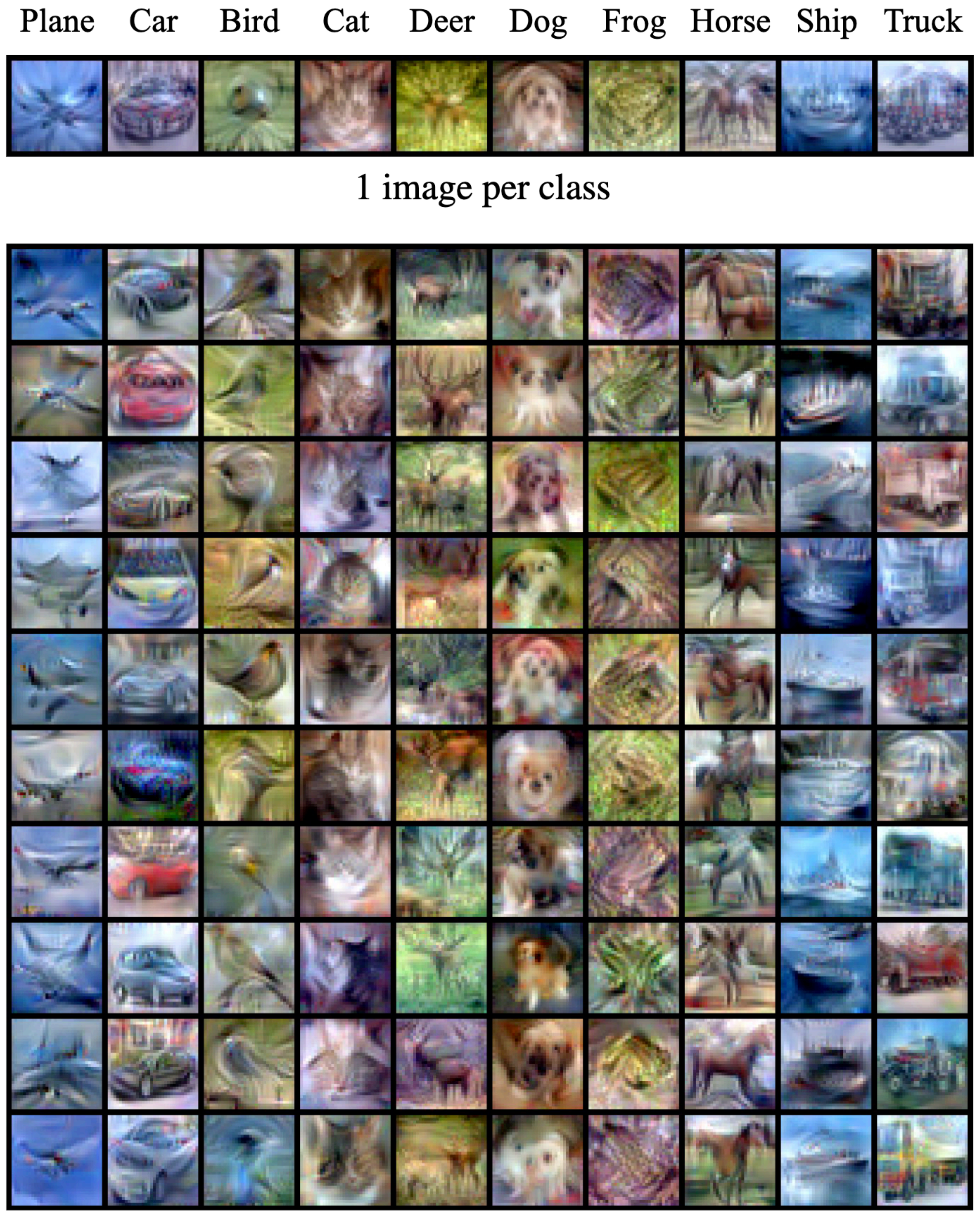
Trajectory matching

- **Result.** Much more visually appealing
  - Example. ImageNet dataset

Cazenavette et al., "Dataset distillation by matching training trajectories," CVPR 2022

Trajectory matching

- Example. CIFAR-10 dataset



10 images per class

Cazenavette et al., "Dataset distillation by matching training trajectories," CVPR 2022

Trajectory matching

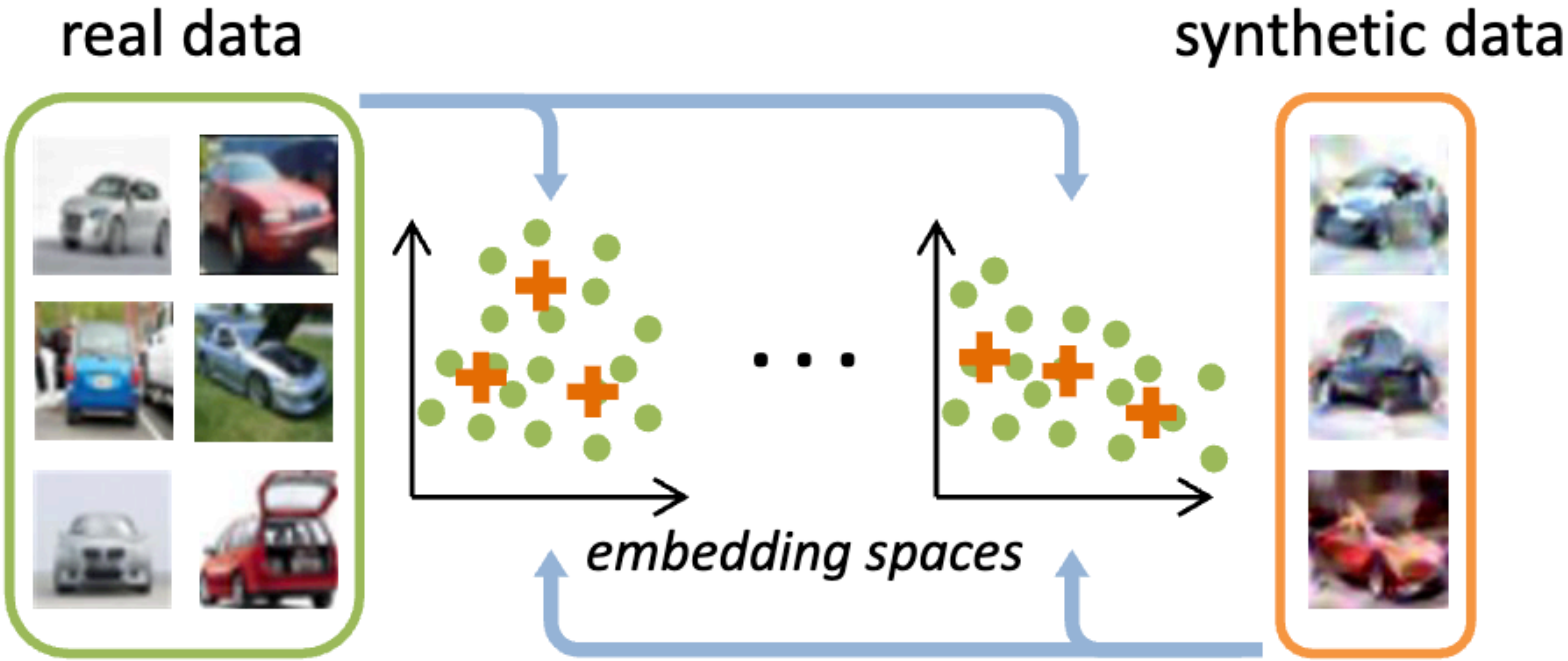
- Much better model accuracy as well
  - But still much worse than full data

	Img/Cls	Ratio %	Coreset Selection: Random	Herding	Forgetting	Training Set Synthesis: DD† [44]	LD† [2]	DC [47]	DSA [45]	DM [46]	CAFE [43]	CAFE+DSA [43]	Ours	Full Dataset
CIFAR-10	1	0.02	14.4 ± 2.0	21.5 ± 1.2	13.5 ± 1.2	-	25.7 ± 0.7	28.3 ± 0.5	28.8 ± 0.7	26.0 ± 0.8	30.3 ± 1.1	31.6 ± 0.8	$\mathbf{46.3 \pm 0.8}^*$	84.8 ± 0.1
	10	0.2	26.0 ± 1.2	31.6 ± 0.7	23.3 ± 1.0	36.8 ± 1.2	38.3 ± 0.4	44.9 ± 0.5	52.1 ± 0.5	48.9 ± 0.6	46.3 ± 0.6	50.9 ± 0.5	$\mathbf{65.3 \pm 0.7}^*$	
	50	1	43.4 ± 1.0	40.4 ± 0.6	23.3 ± 1.1	-	42.5 ± 0.4	53.9 ± 0.5	60.6 ± 0.5	63.0 ± 0.4	55.5 ± 0.6	62.3 ± 0.4	$\mathbf{71.6 \pm 0.2}$	
CIFAR-100	1	0.2	4.2 ± 0.3	8.4 ± 0.3	4.5 ± 0.2	-	11.5 ± 0.4	12.8 ± 0.3	13.9 ± 0.3	11.4 ± 0.3	12.9 ± 0.3	14.0 ± 0.3	$\mathbf{24.3 \pm 0.3}^*$	56.2 ± 0.3
	10	2	14.6 ± 0.5	17.3 ± 0.3	15.1 ± 0.3	-	-	25.2 ± 0.3	32.3 ± 0.3	29.7 ± 0.3	27.8 ± 0.3	31.5 ± 0.2	$\mathbf{40.1 \pm 0.4}$	
	50	10	30.0 ± 0.4	33.7 ± 0.5	30.5 ± 0.3	-	-	-	42.8 ± 0.4	43.6 ± 0.4	37.9 ± 0.3	42.9 ± 0.2	$\mathbf{47.7 \pm 0.2}^*$	
Tiny ImageNet	1	0.2	1.4 ± 0.1	2.8 ± 0.2	1.6 ± 0.1	-	-	-	-	3.9 ± 0.2	-	-	$\mathbf{8.8 \pm 0.3}$	37.6 ± 0.4
	10	2	5.0 ± 0.2	6.3 ± 0.2	5.1 ± 0.2	-	-	-	-	12.9 ± 0.4	-	-	$\mathbf{23.2 \pm 0.2}$	
	50	10	15.0 ± 0.4	16.7 ± 0.3	15.0 ± 0.3	-	-	-	-	24.1 ± 0.3	-	-	$\mathbf{28.0 \pm 0.3}$	

Cazenavette et al., “Dataset distillation by matching training trajectories,” CVPR 2022

Distribution matching

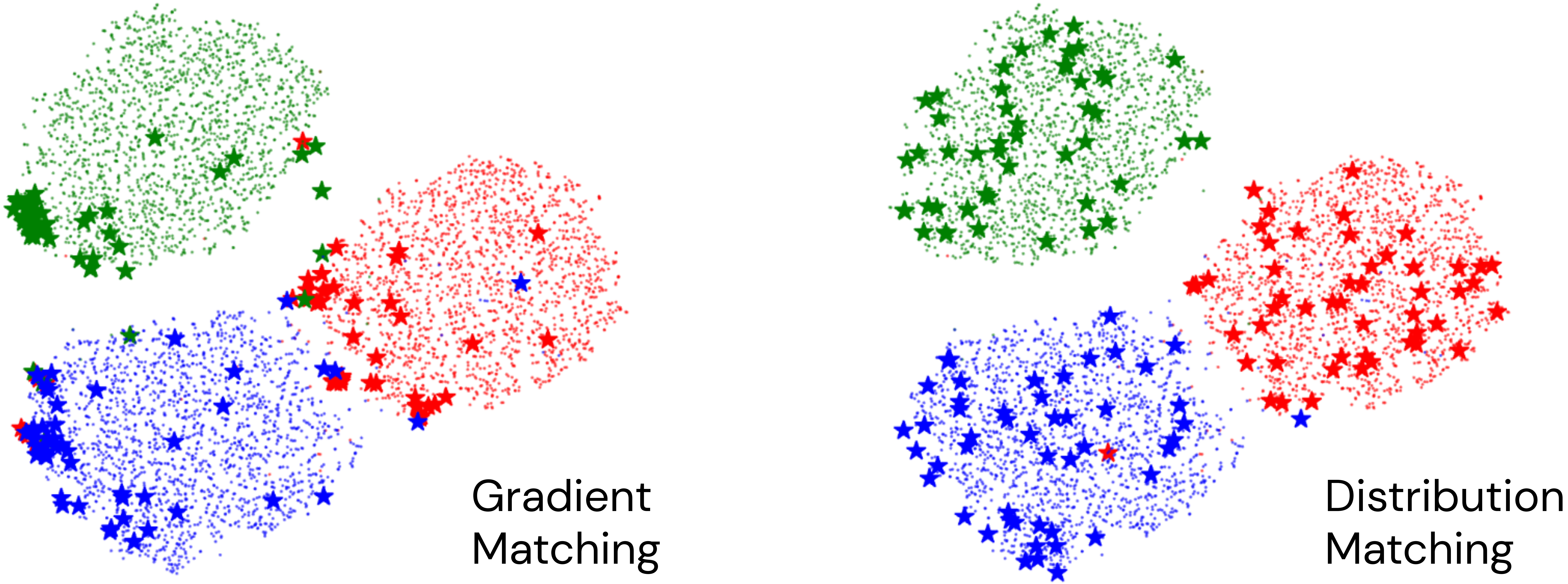
- **Idea.** D and D' should have similar distributions
  - Use some random embedding $g(\,\cdot\,)$ (e.g., randomly initialized net)
  - Common to measure MMD as the distance



Zhao and Bilen, "Dataset condensation with distribution matching," WACV 2023
Wang et al., "CAFE: Learning to condense dataset by aligning features," CVPR 2022.

Distribution matching

- Tend to provide a more wholesome summary of the original distribution



Zhao and Bilen, “Dataset condensation with distribution matching,” WACV 2023
Wang et al., “CAFE: Learning to condense dataset by aligning features,” CVPR 2022.

Wrapping up

- Selecting only the useful data is crucial for more efficient training
    - However, still far from low-cost automation

That’s it for today 🙌