Foundations of Data-efficient Learning

Baharan Mirzasoleiman, Siddharth Joshi

UCLA

BigML

Outline

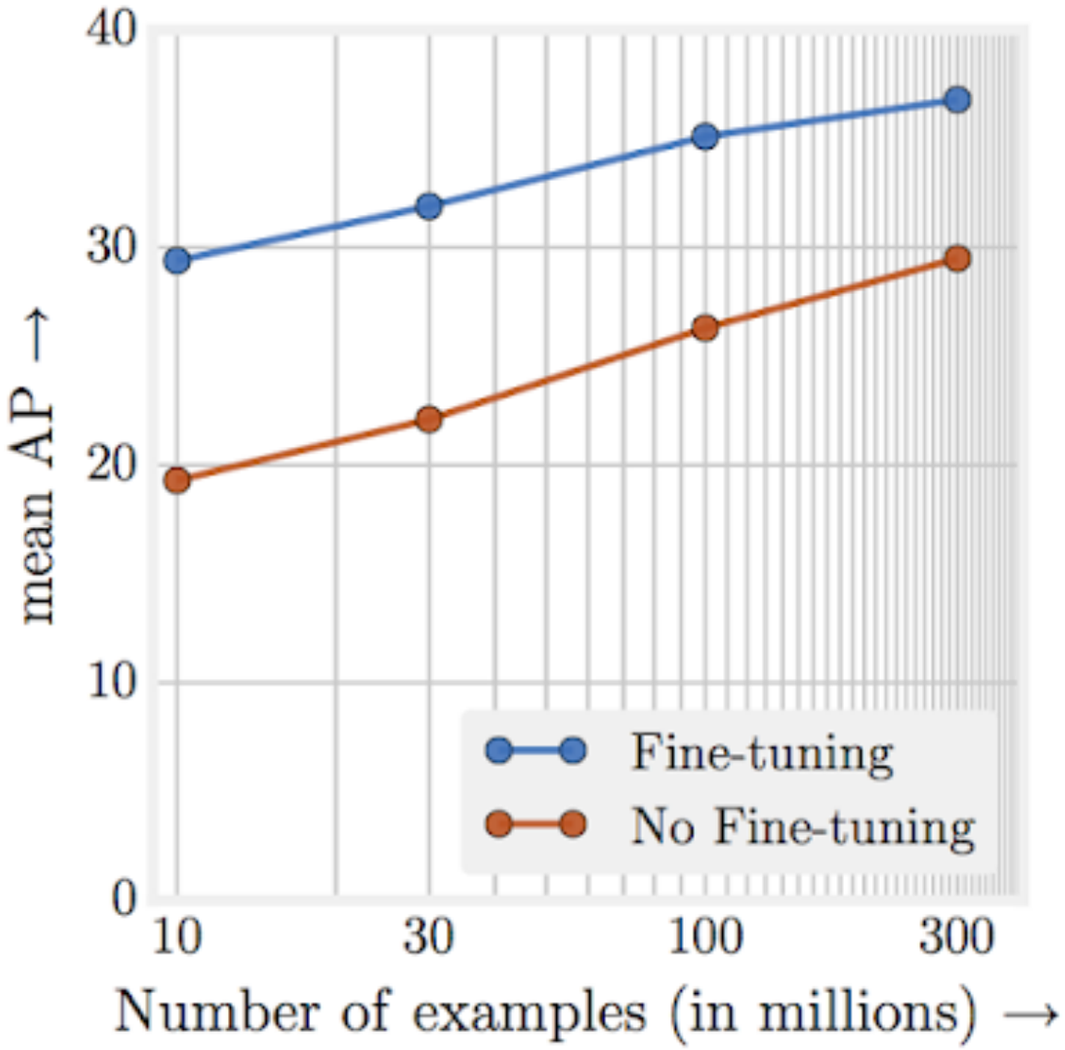
- Motivation: why is data-efficiency important?
- Part 1: Data-efficient Supervised Learning
  10 min break
- Part 2: Data-efficient self-supervised Contrastive Pretraining
  10 min break
- Part 3: Foundation Models
  - 3a: Data-efficient Contrastive Language-Image Pretraining
  - 3b: Data-efficient Training of Large Language Models

Outline

- Motivation: why is data-efficiency important?
- Part 1: Data-efficient Supervised Learning
- Part 2: Data-efficient self-supervised Contrastive Pretraining
- Part 3: Foundation Models
  - 3a: Data-efficient Contrastive Language-Image Pretraining
  - 3b: Data-efficient Training of Large Language Models

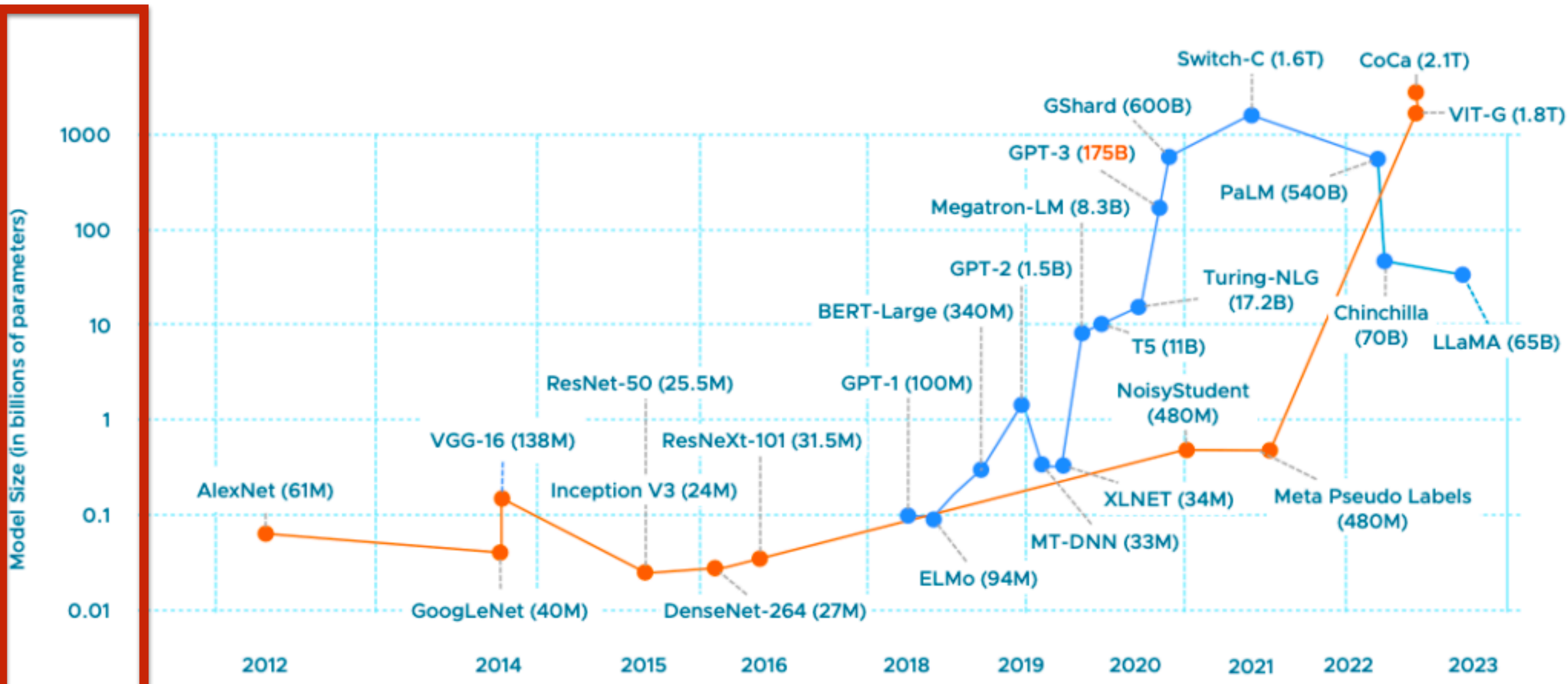
Data is the Fuel for Machine Learning

Example: object detection



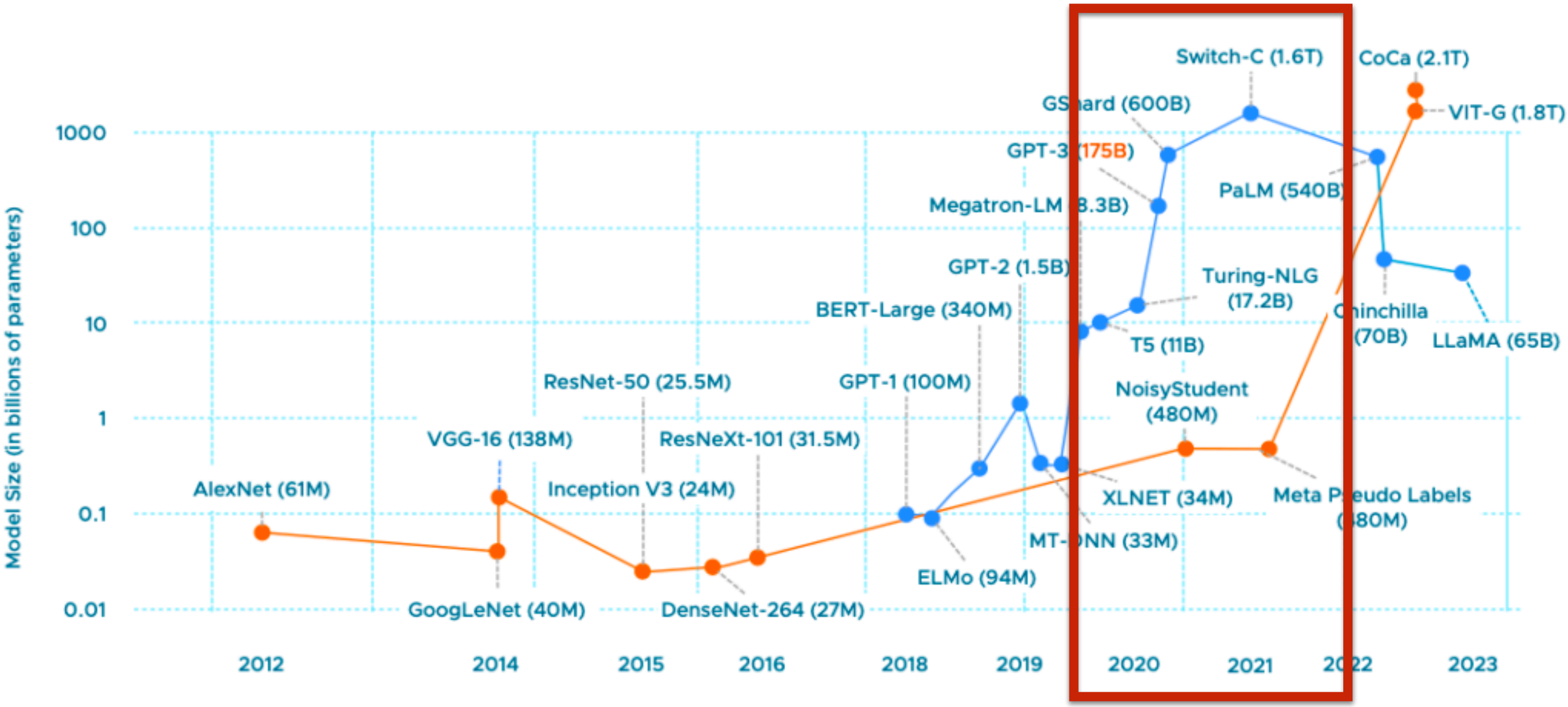
Object detection performance in mAP@[.5,.95] on COCO minival [Google AI, 2017]

How Big Are We Now?



frankdenneman.nl

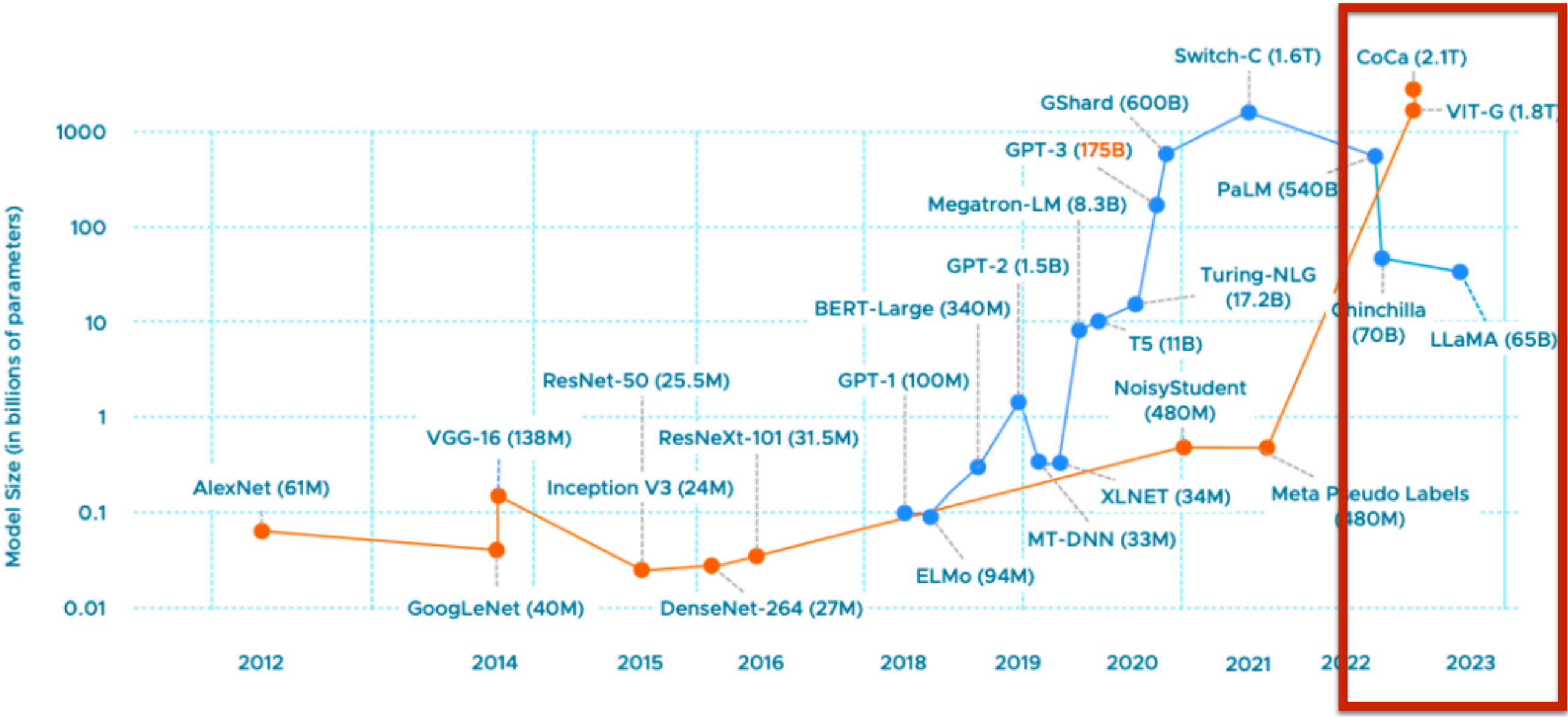
How Big Are We Now?



Increasing model size is a proxy for increasing performance (power-law between model size and performance)

[Kaplan et al 2020]

How Big Are We Now?



Data is as important as scaling model size!
“For 2x model size, data should also be 2x”
[Hoffmann et al 2022]

Datasets are Growing very Rapidly

- Vision and language datasets have historically grown at 0.1 and 0.2 orders of magnitude (OOM) per year, respectively.
- There is a transition around 2014-2015, after which training datasets became much bigger

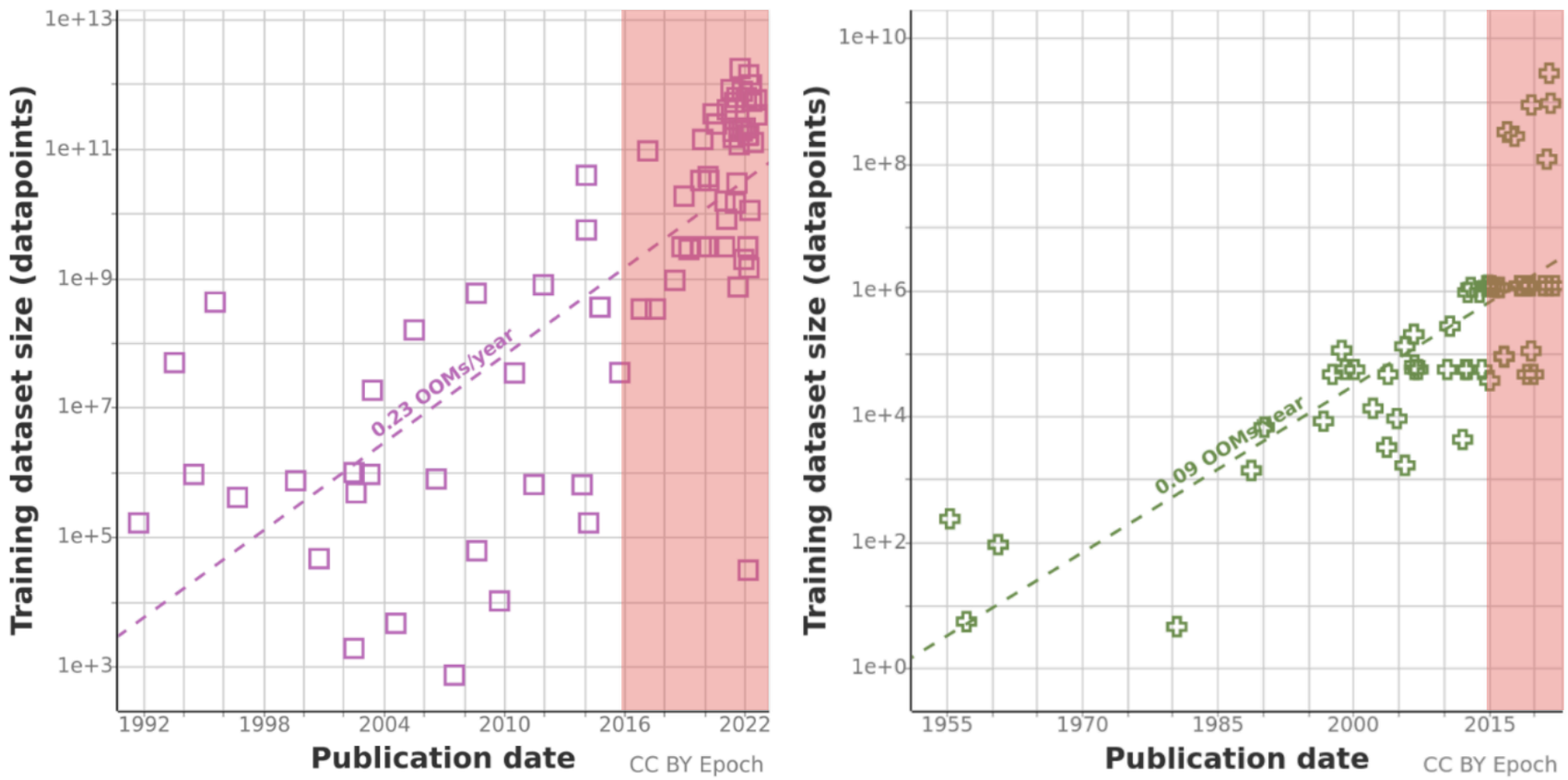


Figure 1: Training datasets for language (left) and vision (right).

https://www.lesswrong.com/posts/asqDCb9XzXnLjSfgL/trends-in-training-dataset-sizes

Data requirement is too big to train models sustainably and deploy them safely!

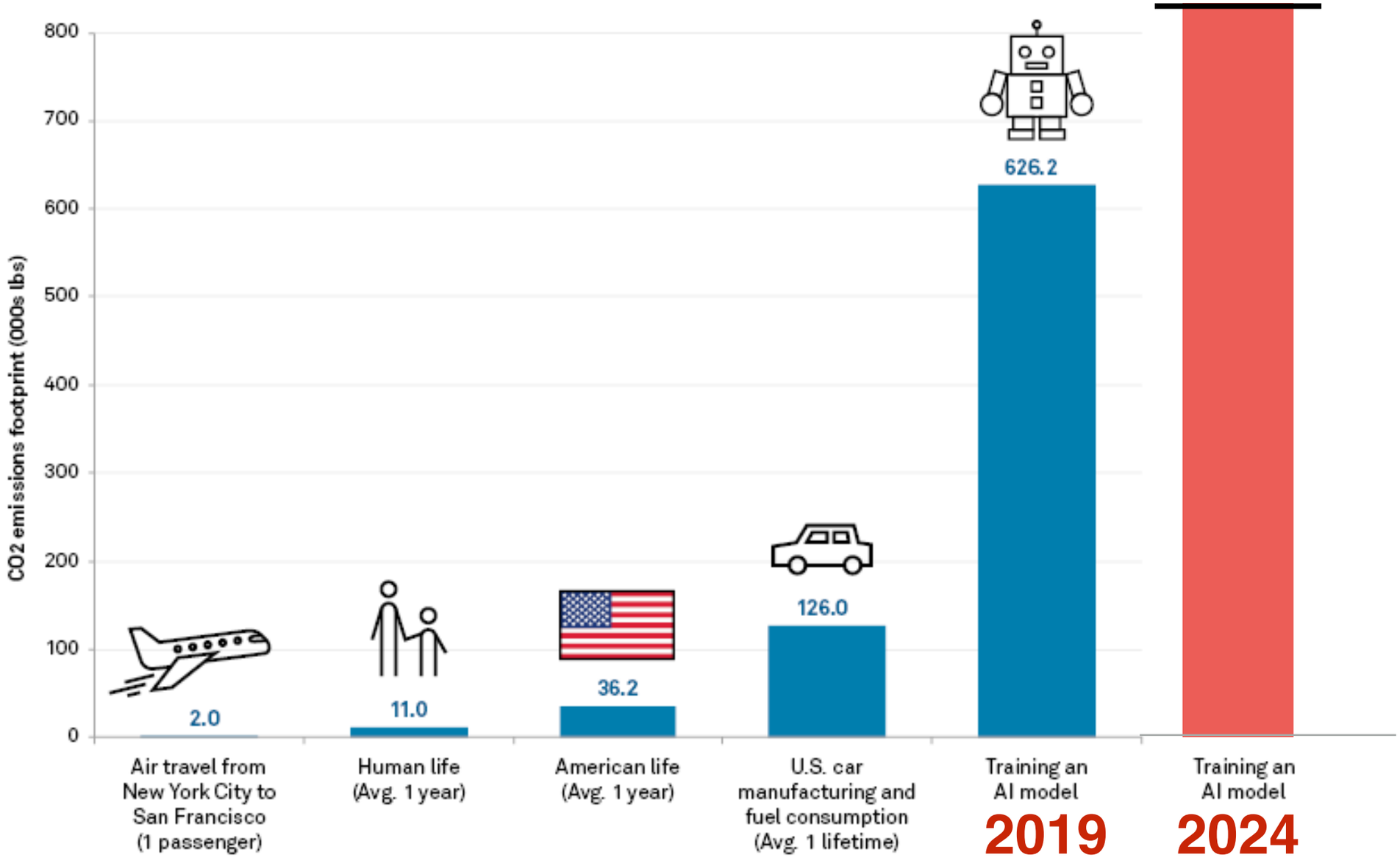
Problem 1: Training on Large Data is Expensive

Source: Spectrum

By 2025, we may have a $10 billion model

And Produces a lot of CO2!

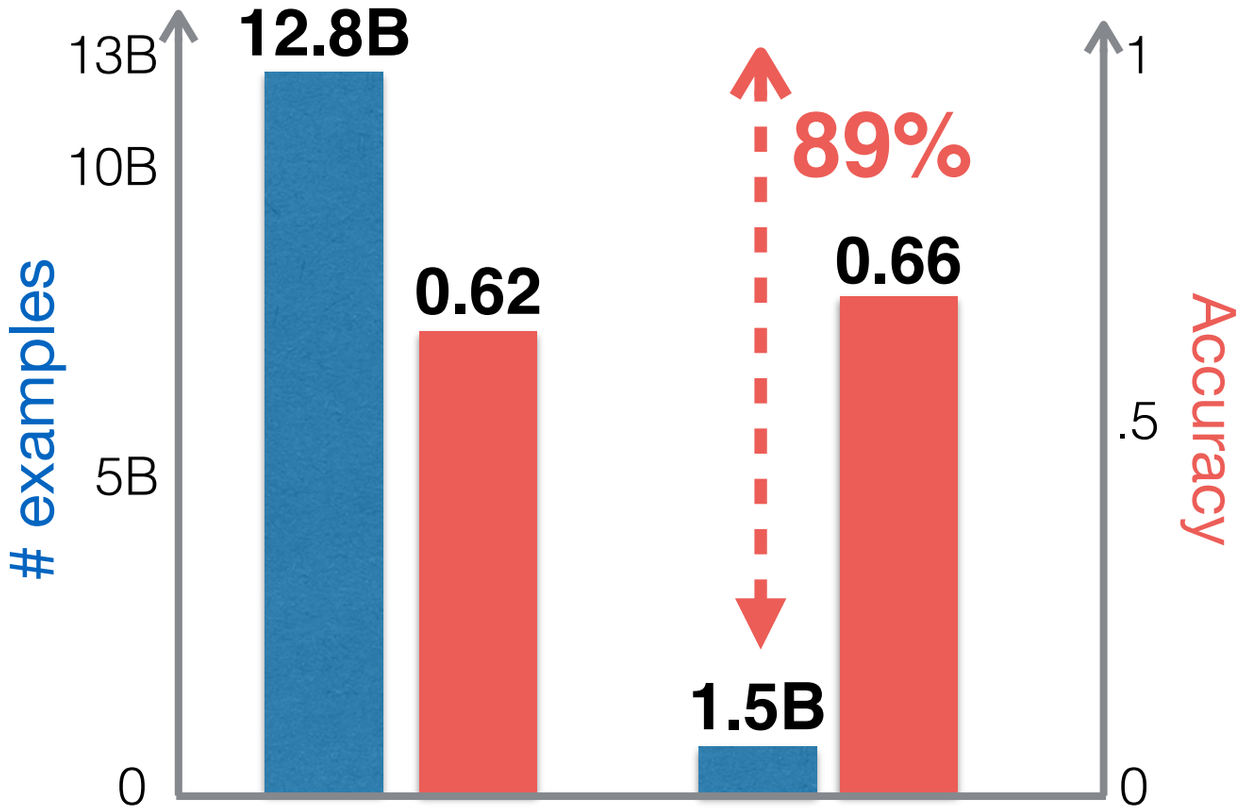
400K-700K tons!



Data compiled Oct. 9, 2019.
An "American life" has a larger carbon footprint than a "Human life" because the U.S. is widely regarded as one of the top carbon dioxide emitters in the world.
Source: College of Information and Computer Sciences at University of Massachusetts Amherst

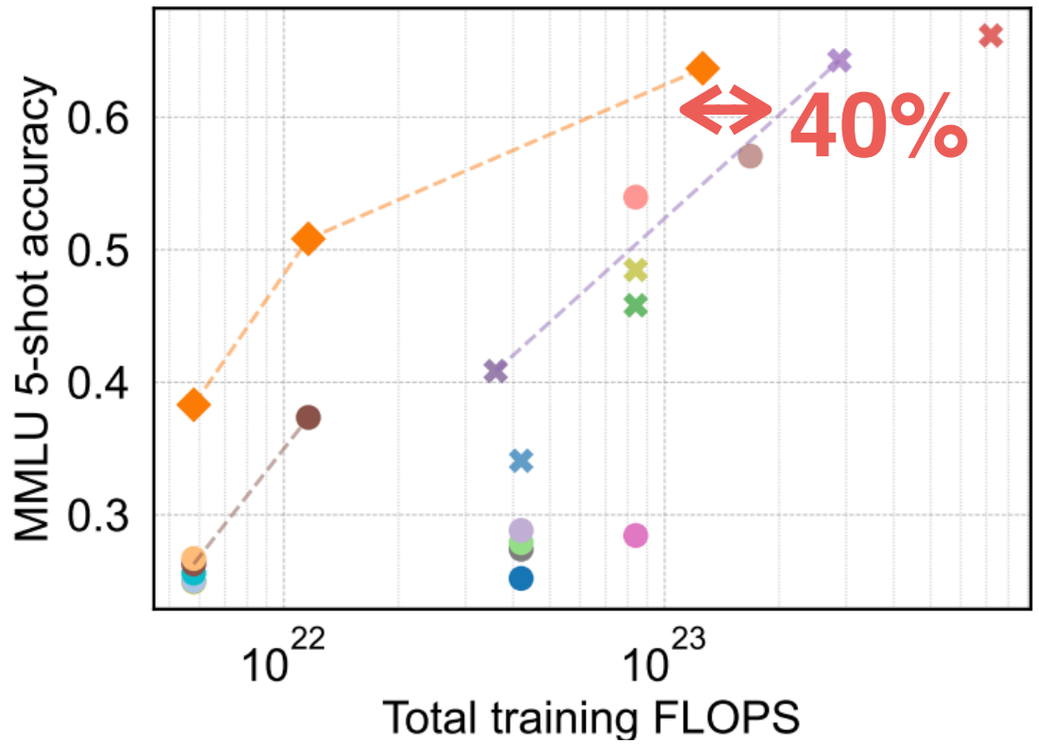
Problem 2: Smaller Higher-quality Data Yield Better Performance (& Efficiency)

- **Data Filtering (filtering noisy, duplicate, or irrelevant data)**
  - Contrastive Language-Image Pretraining (CLIP)
  - Training Large Language Models (LLMs)



CLIP zero-shot performance

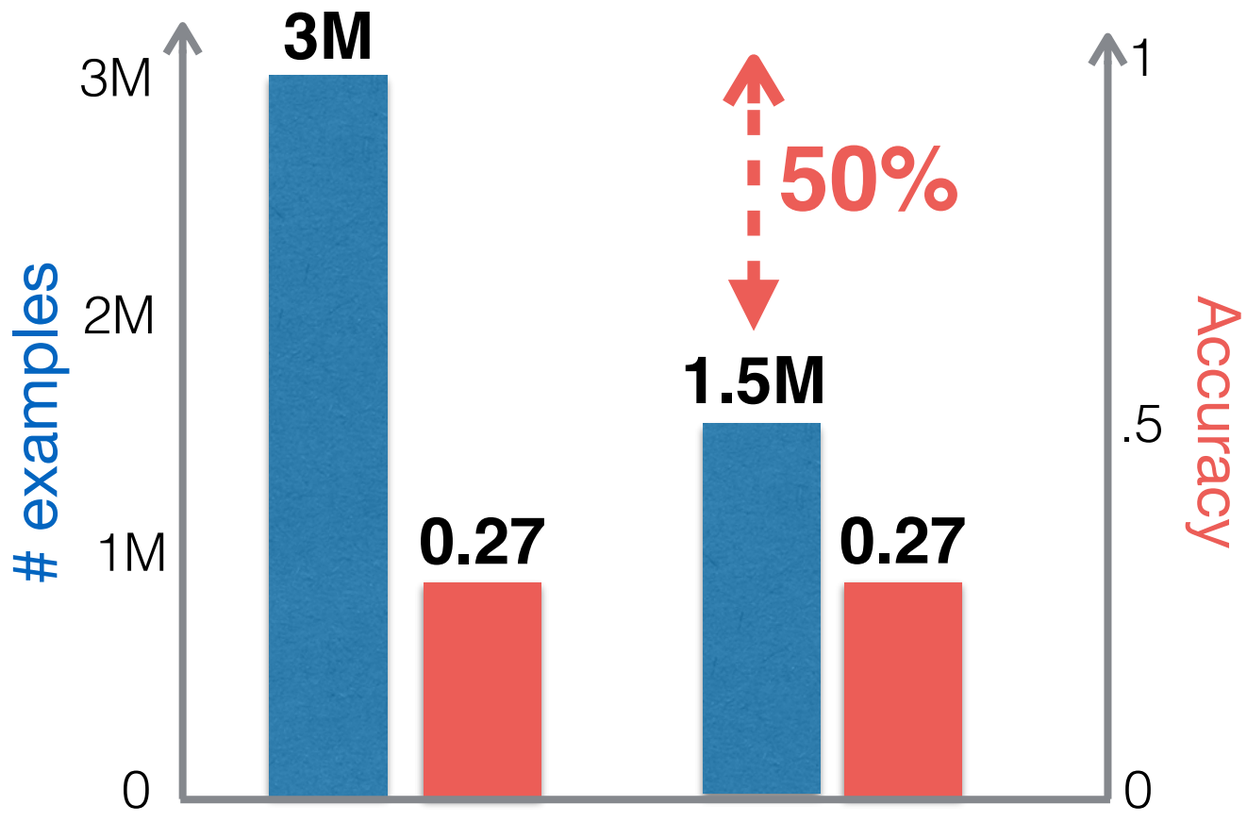
CLIP-ViT, LION [DataComp'23]



7B model [DataComp-LM'24]

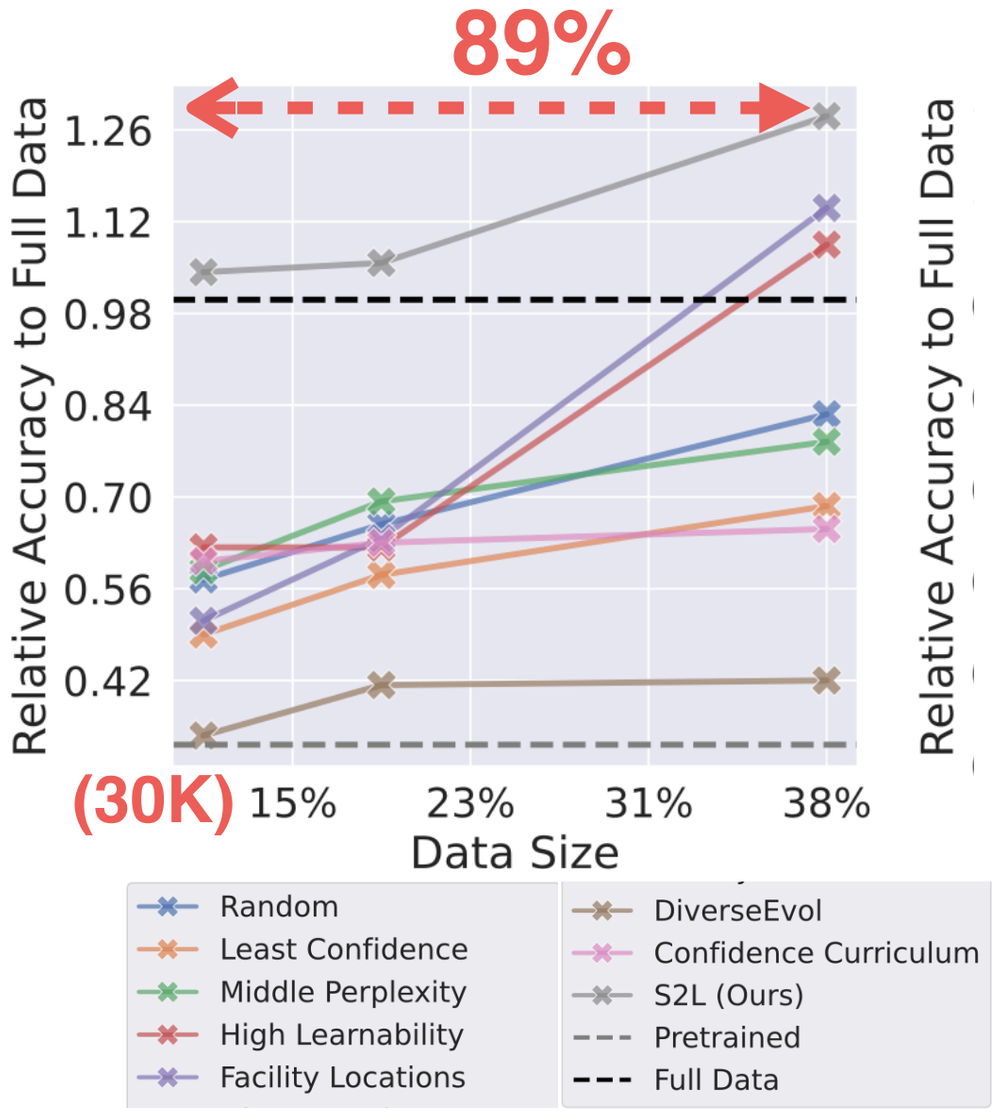
Problem 2: Smaller Higher-quality Data Yield Better Performance (& Efficiency)

- **Data Selection (eliminating redundancy & outliers)**
  - Contrastive Language-Image Pretraining (CLIP)
  - Training Large Language Models (LLMs)



CLIP zero-shot performance

CLIP-R50, CC3M [AISTATS'24]

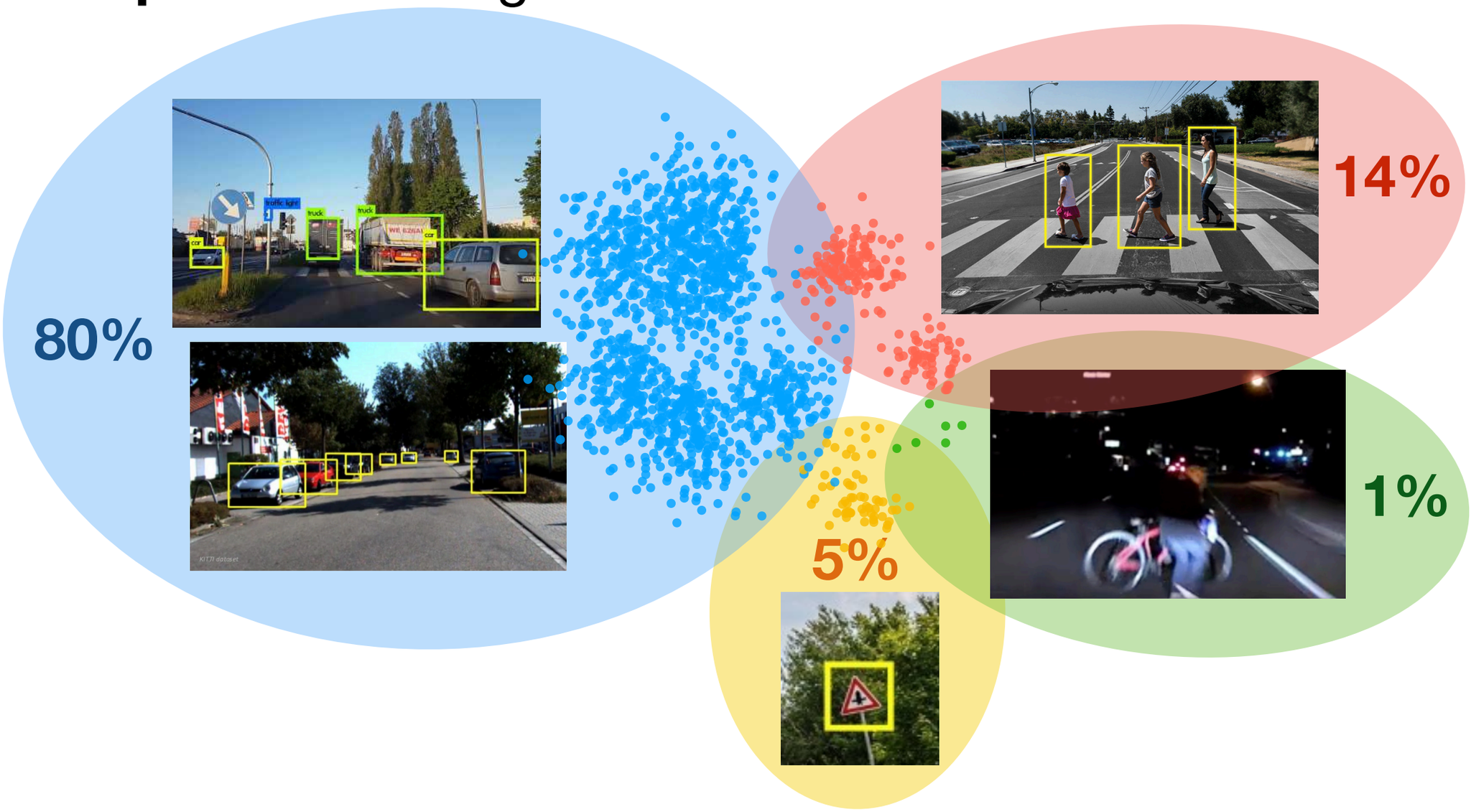


Pythia-410M, MathInstruct [ArXiv'23]

Problem 3: Real-world Datasets are Biased

- Model performs poorly on minorities (Fairness, Safety)
- Spurious biases (& generally feature imbalance) impede out-of-distribution (OOD) performance

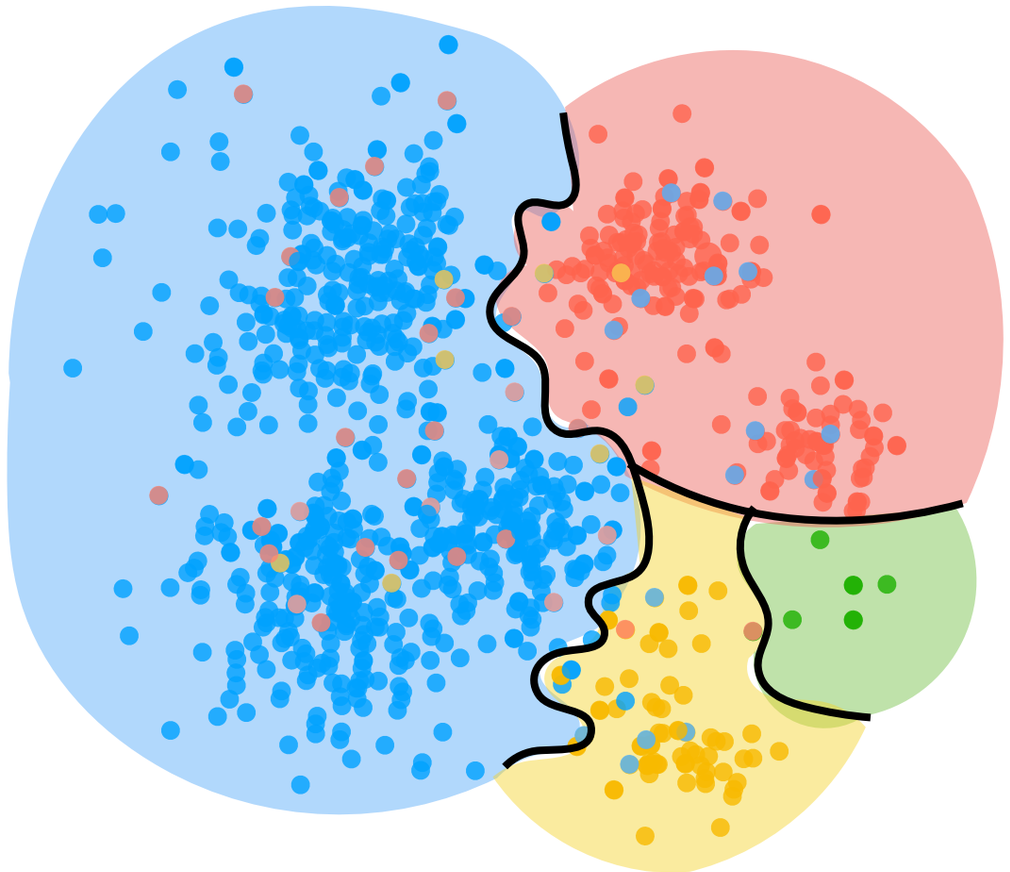
Example: self driving data



Problem 4: Real Data is Unlabeled

- We label the data automatically

  Example: crowed-sourcing, automated labeling, …

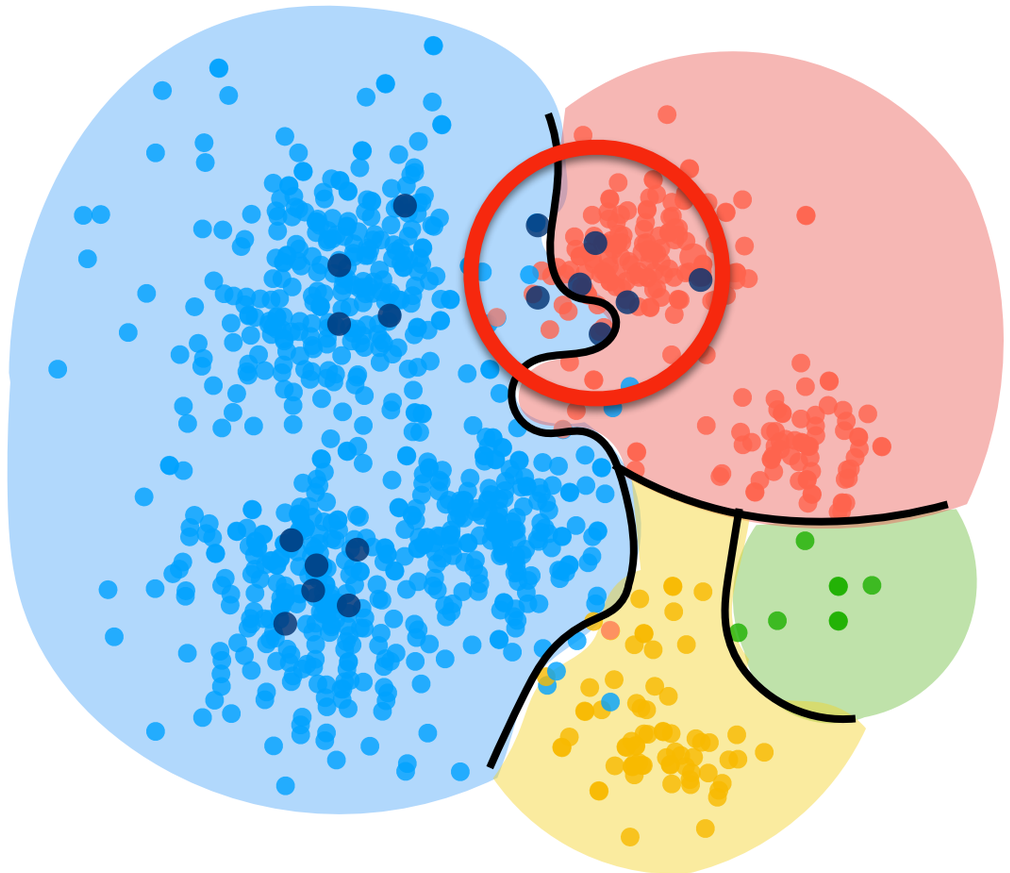


Over-parameterized models overfit and memorize the mislabeled data

Problem 5: Examples May be Corrupted

- Many large datasets are collected from Internet or users

  Example: large Image and NLP datasets, …



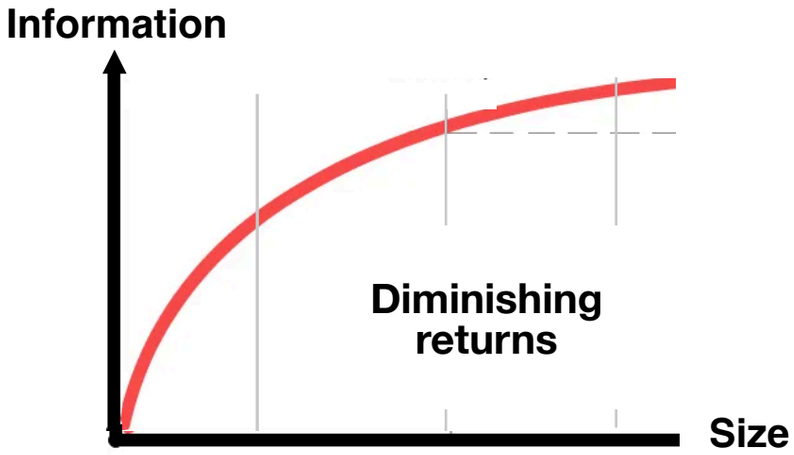
Adversarial attacks change the prediction of a test-time target example and cannot be visually identified

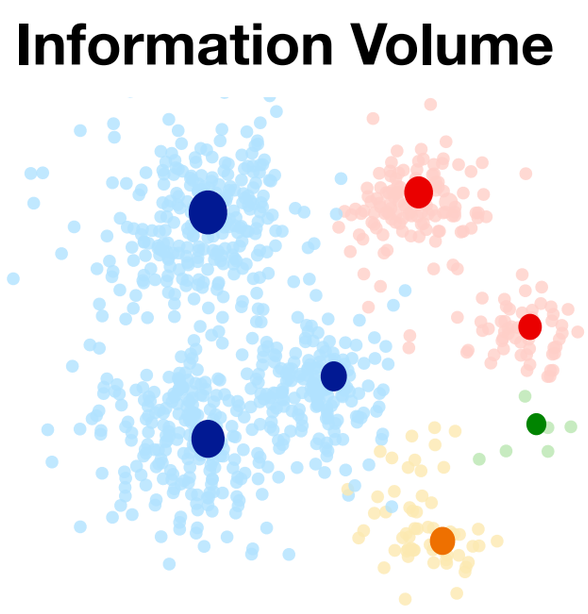
Can we select smaller subsets of large data to improve **efficiency, performance** and **robustness** of learning from large datasets?

Today: **Efficiency & performance** (problem 1&2)!

Why Smaller Data Can Yield Better Performance?

Key insight: the non-redundant information content is asymptotically a diminishing return as data volume increases





By training on the subset, the model can learn the information volume better!

Data Quality vs Quantity!

- How can we find high-quality subsets of large datasets?
  - Data filtering methods
    - Can only filter obvious issues, but cannot eliminate redundancy

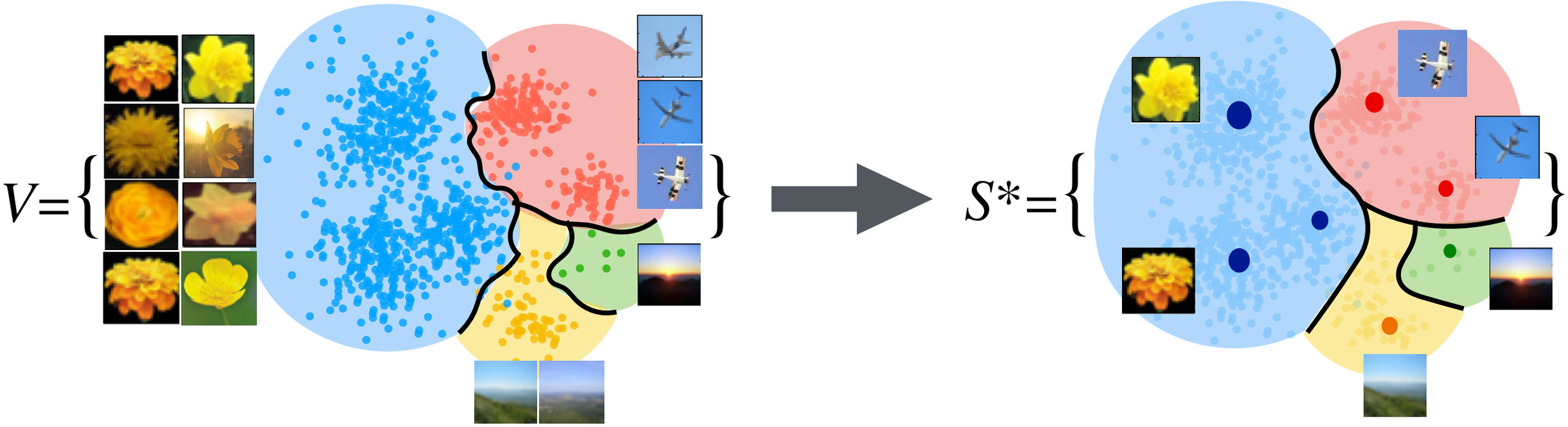
- Finding smallest subsets that generalize on par with full data
- Finding a curriculum that ensures faster and better generalization

Our focus!

How can we find smallest subsets that generalize on par with full data?

How to Find the Most Beneficial Subsets?

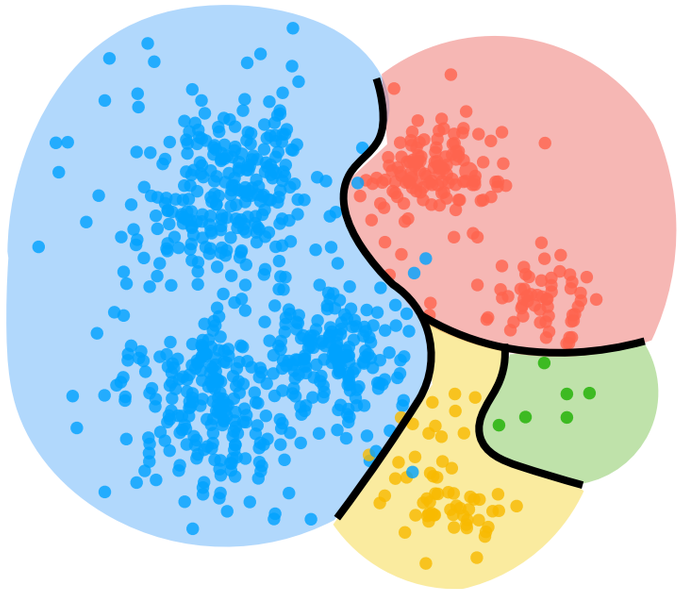
- The most informative subset $S^* = \arg\max_{S \subseteq V} F(S)$, s.t. $|S| \leq k$
- **What is a good choice for $F(S)$?** **Set function**



- Can speed up training by up to $|V|/|S|$.

Which Subsets are Most Beneficial for Training?

1. How to chose an informative subset for training?
   - Points close to decision boundary vs. a diverse subset?



2. Finding S^* must be fast
   - Otherwise we don't get any speedup

3. We need theoretical guarantees
   - For the quality of the trained model
   - For convergence of incremental gradient method on the subset

Outline

- Motivation: why is data-efficiency important?
- Part 1: Data-efficient Supervised Learning
- Part 2: Data-efficient self-supervised Contrastive Pretraining
- Part 3: Foundation Models
  - 3a: Data-efficient Contrastive Language-Image Pretraining
  - 3b: Data-efficient Training of Large Language Models

Setting: Training Machine Learning Models

Often reduces to minimizing a regularized empirical risk function

Training data volume: $\{(x_i, y_i), i \in V\}$

Feature **Label**

$$w_* \in \arg\min_{w \in \mathscr{W}} f(w), \quad f(w) = \sum_{i \in V} f_i(w) + r(w), \quad f_i(w) = l(w, (x_i, y_i))$$

Regularizer

Loss function associated with training example $i \in V$

Examples:

- **Convex** $f(w)$: Linear regression, logistic regression, ridge regression, regularized support vector machines (SVM)
- **Non-convex** $f(w)$: Neural networks

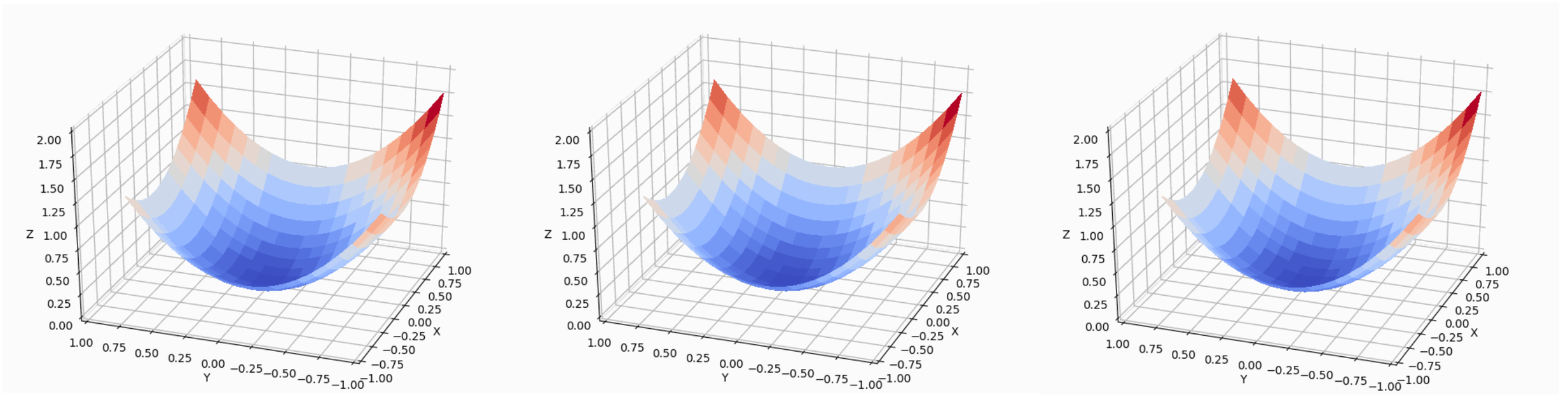
Setting: Training Machine Learning Models

Incremental gradient methods are used to train on large data

- Sequentially step along the gradient of functions f_i

$$w_i^k = w_{i-1}^k - \alpha_k \nabla f_i(w_{i-1})$$

- They are **slow** to converge



Proposed Framework: Learning from Coresets

Idea: select the smallest subset S^* and weights γ that closely estimates the full gradient

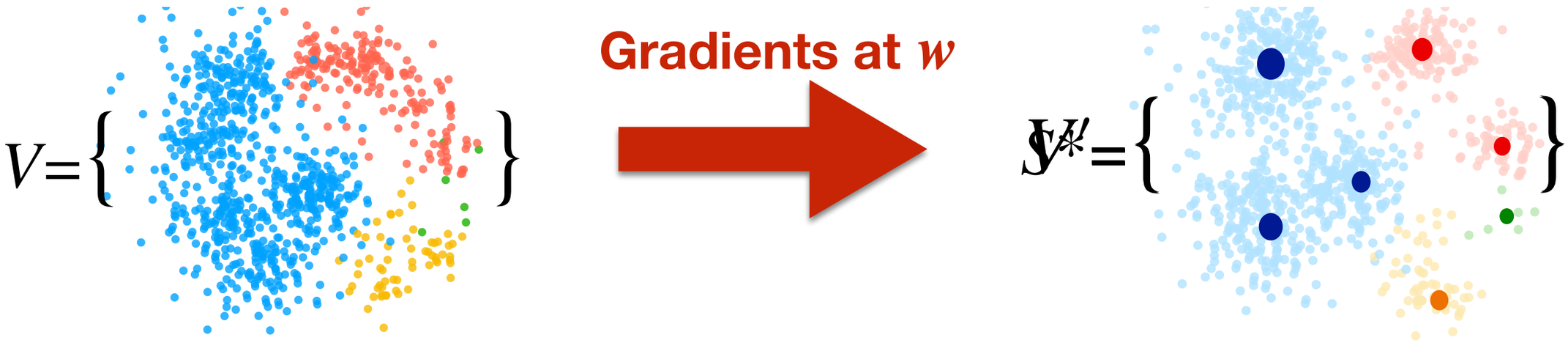
$$S^* = \arg\min_{S \subseteq V, \gamma_j \geq 0 \ \forall j} |S|, \quad \text{s.t.} \quad \max_{w \in \mathscr{W}} \| \sum_{i \in V} \nabla f_i(w) - \sum_{j \in S} \gamma_j \nabla f_j(w) \| \leq \epsilon .$$

Full gradient **Gradient of S**

Solution: for every $w \in \mathscr{W}$, S^* is the set of **medoids** of all the data points in the **gradient space**

Training Data: $\{(x_i, y_i), i \in V\}$

$$V' = \{ \nabla f_i(w), i \in V\}$$



Our Approach: Learning from Coresets

How can we find medoids/exemplars in big datasets?

- **Exemplar clustering is submodular!**

$$F(S^*) = \sum_{i \in V} \min_{j \in S^*} \|\nabla f_i(w) - \nabla f_j(w)\| \leq \epsilon$$

[ICML tutorial'13]

Submodularity is a natural diminishing returns property

$$\forall\, A \subseteq B \text{ and } B \not\ni x: \qquad F(A \cup \{x\}) - F(A) \geq F(B \cup \{x\}) - F(B)$$

A simple greedy algorithm can find exemplars S^* in large datasets

However, S^* depends on w!

- We have to update S^* after every SGD update

Slow! :(

Our approach: Learning from Coresets

Can we find a subset S^* that bounds the estimation error for all $w \in \mathscr{W}$?

$$F(S^*) = \sum_{i \in V} \min_{j \in S^*} \|\nabla f_i(w) - \nabla f_j(w)\| \leq \epsilon$$

Idea: consider worst-case approximation of the estimation error over the entire parameter space $\mathscr{W}$

$$F(S^*) = \sum_{i \in V} \min_{j \in S^*} \|\nabla f_i(w) - \nabla f_j(w)\|$$

d_{ij}: upper-bound on the gradient difference over the entire parameter space $\mathscr{W}$

Our approach: Learning from Coresets

How can we efficiently find upper-bounds d_{ij}?

- **Convex** $f(w)$: Linear/logistic/ridge regression, regularized SVM

$$d_{ij} \leq \text{const. } \|x_i - x_j\|$$

Feature vector

☑ S^* **can be found as a preprocessing step**

- **Non-convex** $f(w)$: Neural networks

$$d_{ij} \leq \text{const. } (\|\nabla_{z_i^{(L)}} f_i(w) - \nabla_{z_j^{(L)}} f_j(w)\|) \qquad \text{[KF'19]}$$

Input to the last layer

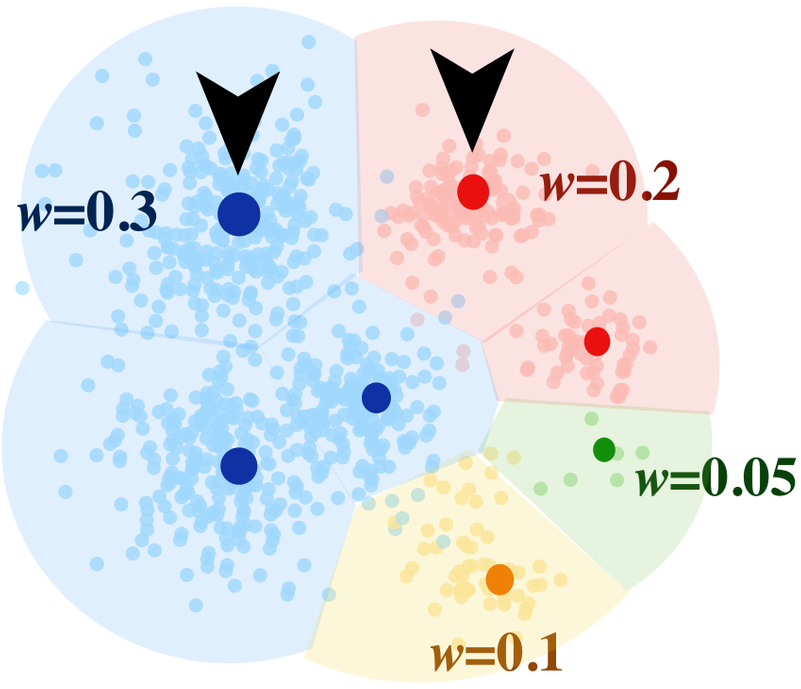
☑ d_{ij} **is cheap to compute, but we have to update** S^*

CRAIG: Learning from Coresets

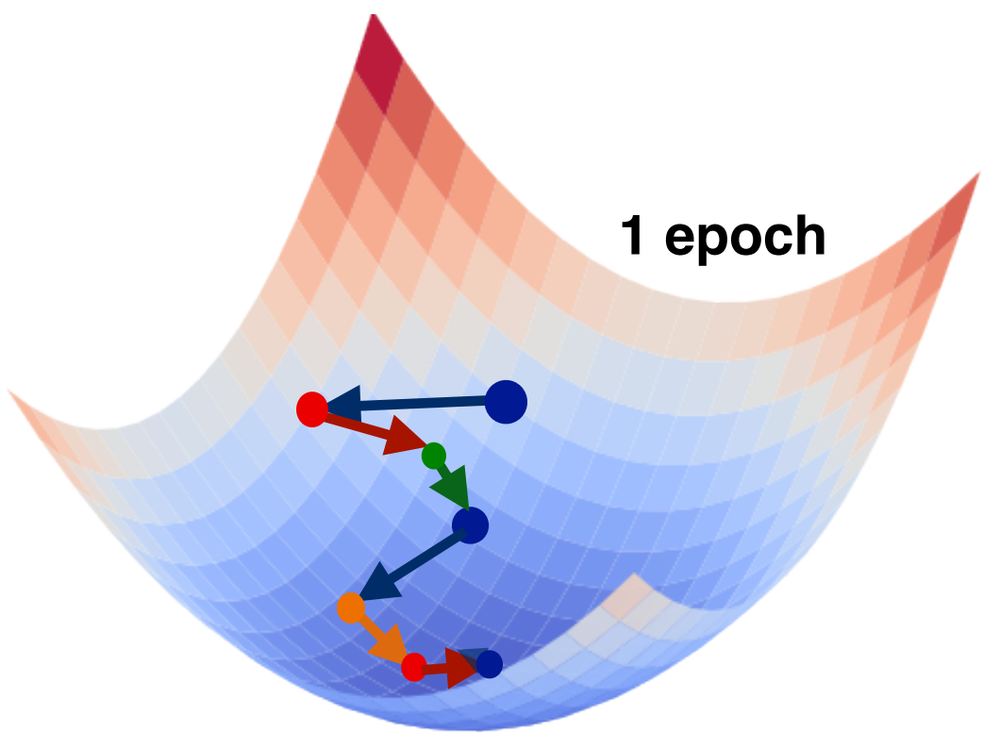
Idea: select a weighted subset that closely estimates the full gradient

Algorithm:

- **(1)** use **greedy** to find the set of exemplars S^* from dataset V
- **(2)** **weight** every elements of S^* by the size of the corresponding cluster
- **(3)** apply weighted incremental gradient descent on S^*



Gradients of data points $i \in V$



Loss function

Coresets for Data-efficient Training of Machine Learning Models, ICML 2020

CRAIG: Learning from Coresets

Idea: select a weighted subset that closely estimates the full gradient

Algorithm:

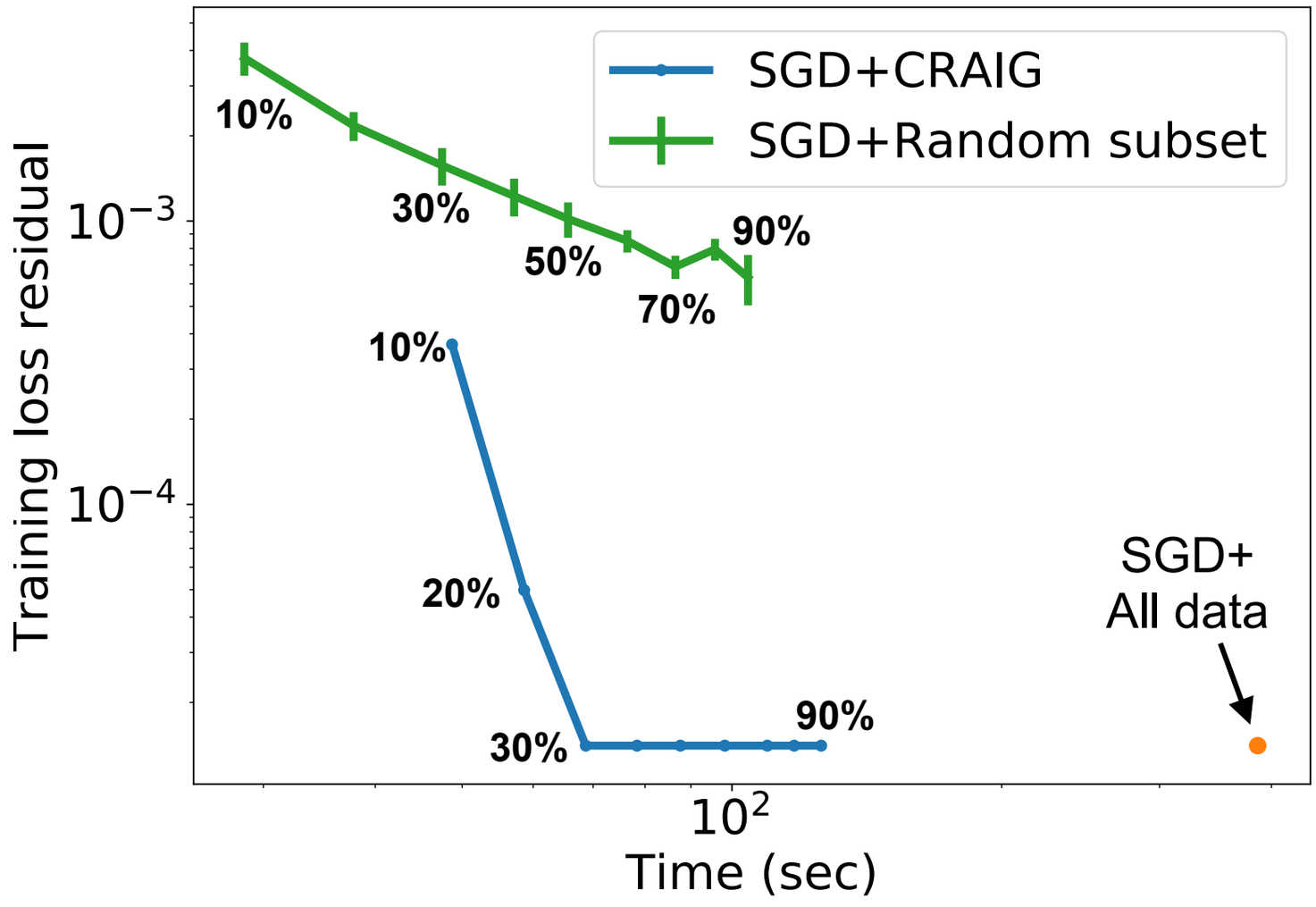
- **(1)** use **greedy** to find the set of exemplars S^* from dataset V
- **(2)** **weight** every elements of S^* by the size of the corresponding cluster
- **(3)** apply weighted incremental gradient descent on S^*

Theorem: For a μ-strongly convex loss function, CRAIG with decaying step-size $\Theta(1/k^\tau), \tau < 1$ converges to a $2\epsilon/\mu$ neighborhood of the optimal solution, with a rate of $\mathcal{O}(1/k^\tau)$

Application of CRAIG to Logistic Regression

Training on subsets of various size of Ijcnn1 with **50K points**

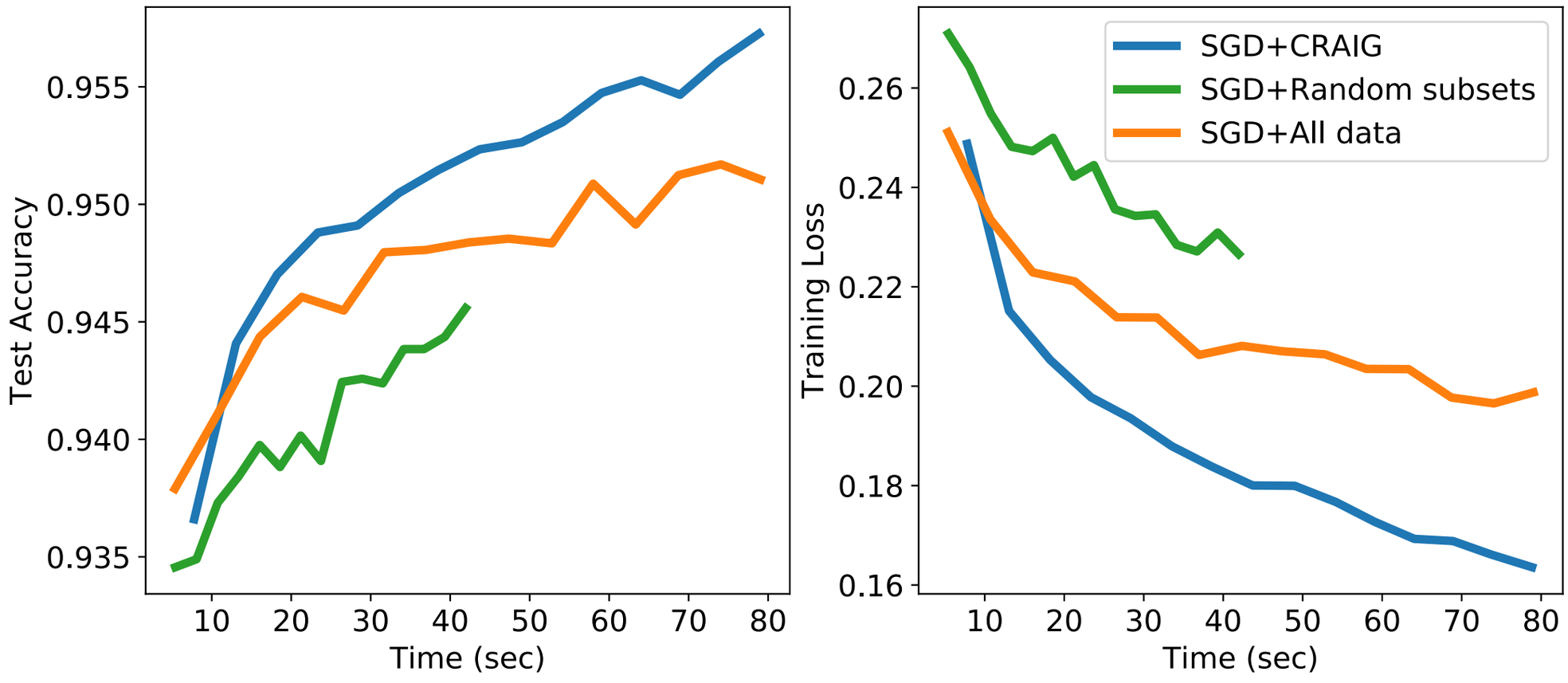
(Imbalanced)



Up to 7x faster than training on the full data, with the same accuracy

Application of CRAIG to Neural Networks

Training on MNIST with a 2-layer neural network with 50K points

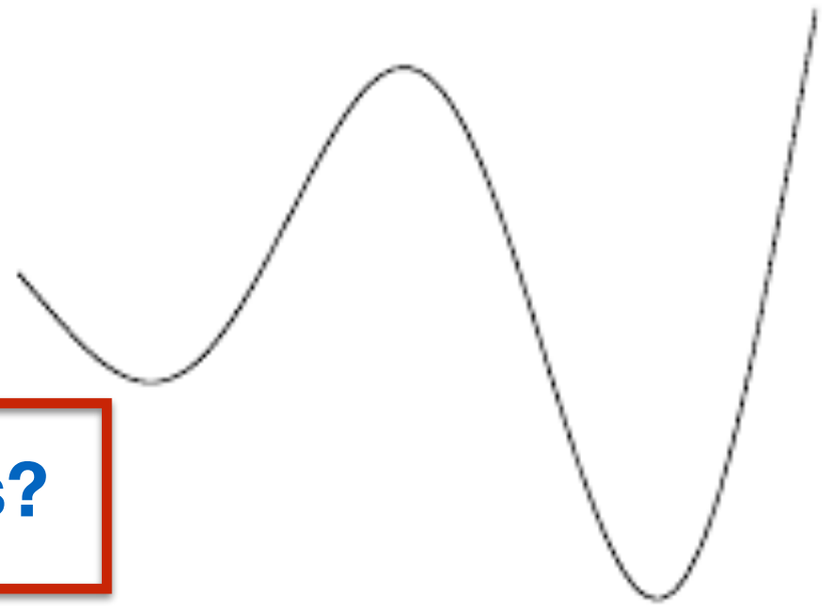


2x-3x faster than training on the full data, with **better generalization**

Can we find coresets for training **deep networks**?

Coresets for Data-efficient Deep Learning

- Can we find coresets for minimizing a non-convex loss?
- **Challenges:**
  - (1) **Loss changes** very rapidly
    - **When should we update the coresets?**



  - (2) Deep networks are trained with (mini-batch) SGD
    - (Mini-batch) SGD requires **unbiased** gradient estimates with **small variance**

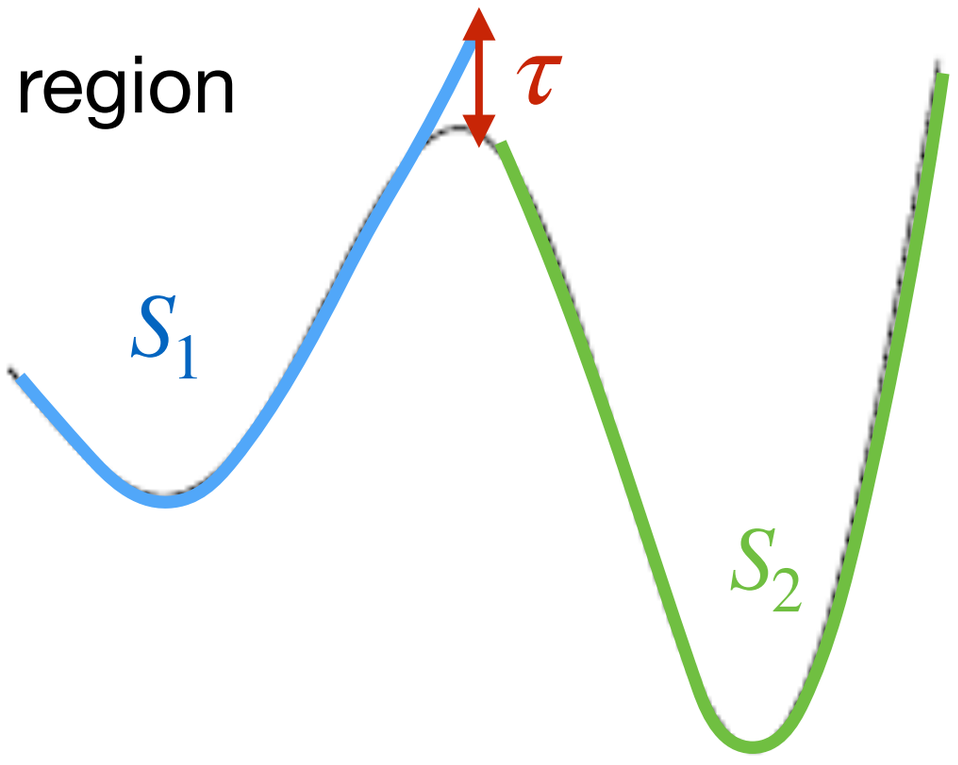
CREST: Coresets for Data-efficient Deep Learning

- **Change 1:** loss changes very rapidly
  - **When should we update the coresets?**
- **Solution:**
  - 1- Modeling the non-convex loss as piece-wise convex

$$\mathscr{F}^l(\delta) = \frac{1}{2}\delta^T H_S \delta + g_S \delta + \mathscr{L}(w)$$

  - 2- Selecting one subset per convex region
  - 3- Train on it as long as the convex approximation is valid

$$\frac{|\,\mathscr{F}^l(\delta) - \mathscr{L}(w+\delta)\,|}{\mathscr{L}(w+\delta)} \leq \tau$$



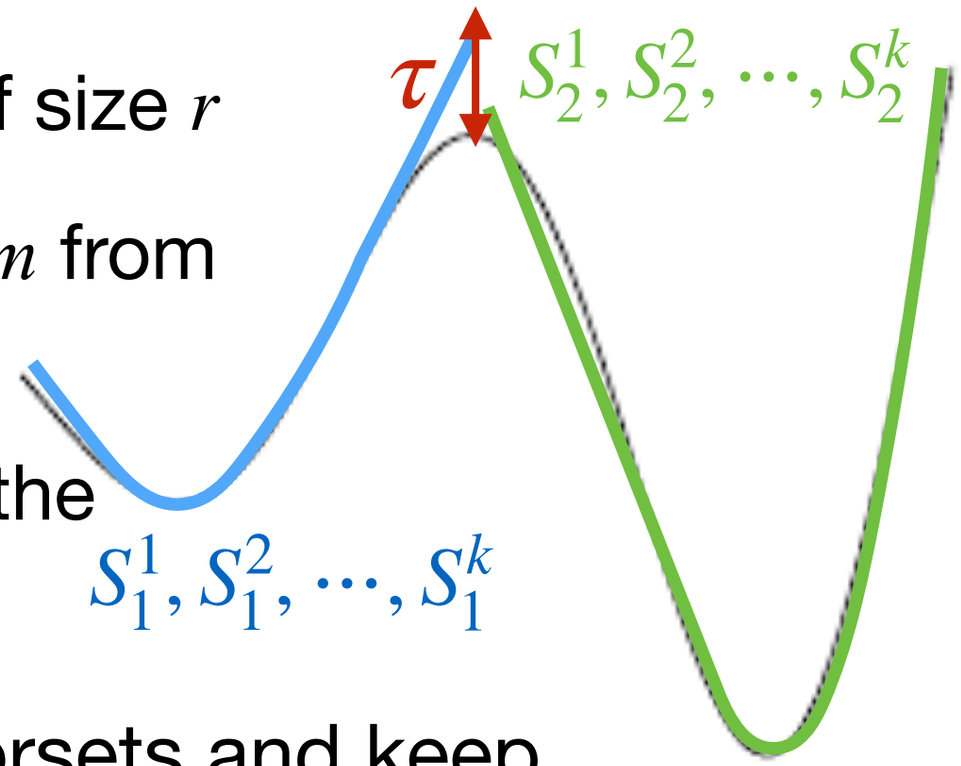
CREST: Coresets for Data-efficient Deep Learning

- Can we find coresets for minimizing a non-convex loss?
- **Challenges:**
  - (1) **Loss changes** very rapidly
    - **When should we update the coresets?**
  - (2) Deep networks are trained with (mini-batch) SGD
    - (Mini-batch) SGD requires unbiased gradient estimates with small variance
    - Mini-batches selected from coresets that capture the full gradient has a **large bias and variance**

Can we find mini-batch coresets with small bias & variance?

CREST: Coresets for Data-efficient Deep Learning

- **Change 2:** SGD requires unbiased gradient estimates
  - **Can we find mini-batch coresets with small bias & variance?**
- **Solution:**
  - 1- Select multiple random subsets of size r
  - 2- Find a coreset of mini-batch size m from each random subset
  - 3- Make a convex approximation to the union of the mini-batch coresets
  - 4- Randomly sample mini-batch corsets and keep training on them as long as the approximation is valid



CREST: Coresets for Data-efficient Deep Learning

- **Crest algorithm:**
  - 1- Modeling the non-convex loss as piece-wise convex
  - 2- Find mini-batch coresets of size m from larger random subsets of size r (have nearly-unbiased gradients)
  - 3- Randomly select and train on mini-batch corsets as long as the convex approximation is valid

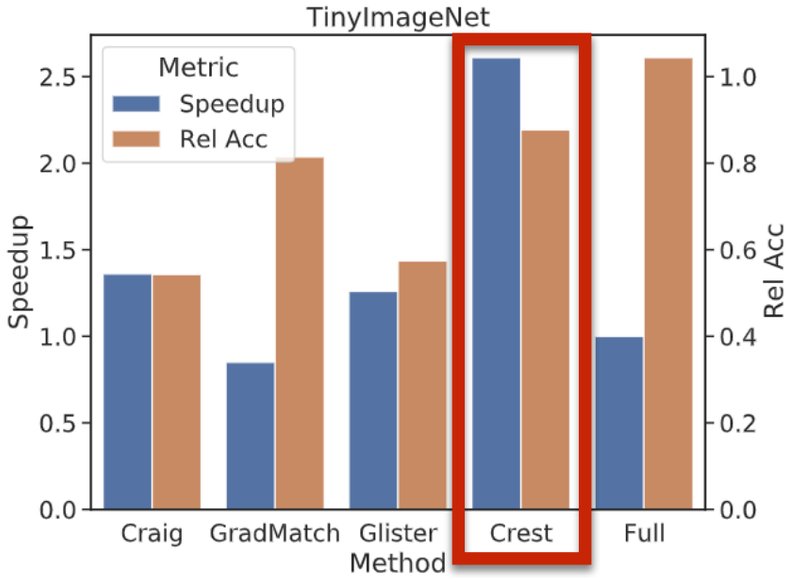
Theorem (informal): Training with SGD on Crest mini-batch corsets guarantees convergence to an ϵ-stationary point of a non-convex loss, r/m **times faster** than random mini-batches of size m:

$$\tilde{\mathcal{O}}\left(\frac{L(\mathscr{L}(w_0)-\mathscr{L}^*)}{\epsilon^2}(1+\frac{\sigma^2}{r\epsilon^2})\right)$$

CREST: Coresets for Data-efficient Deep Learning

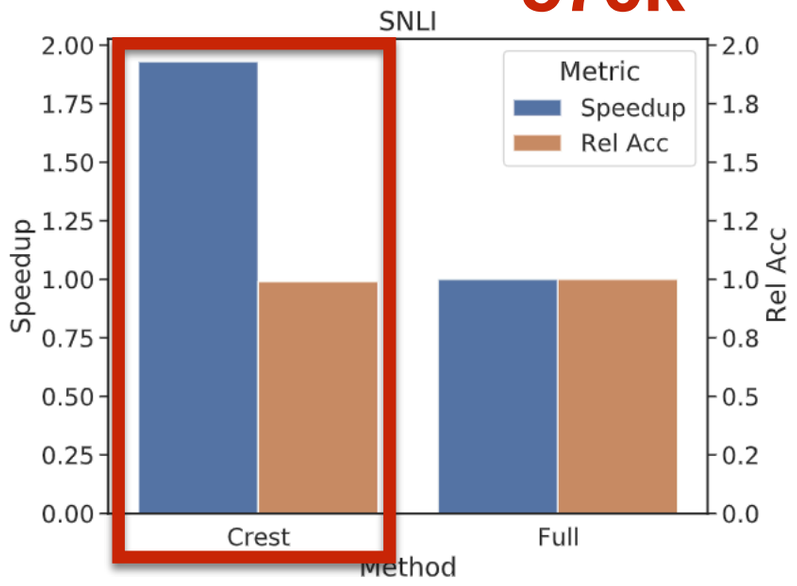
Dataset	Backprop	Random	Craig	Grad-match*	Glister*	**Crest**(Ours)
CIFAR-10	10%	5.25	10.90	6.15	**2.07**	3.36
CIFAR-100	10%	11.27	20.72	26.65	33.02	**7.35**
TinyImageNet	10%	16.31	49.86	21.71	44.89	**13.22**
SNLI (Finetune)	10%	1.5	-	-	-	**0.54**

Achieves much higher test acc!



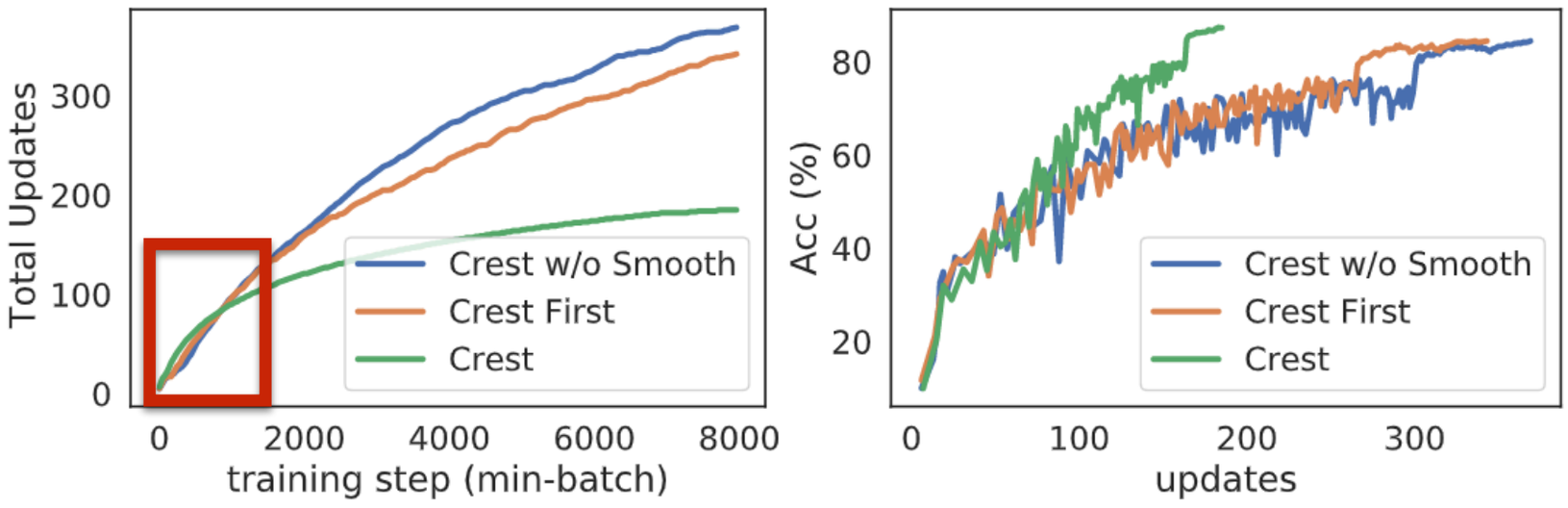
2-3x speedup!

570k

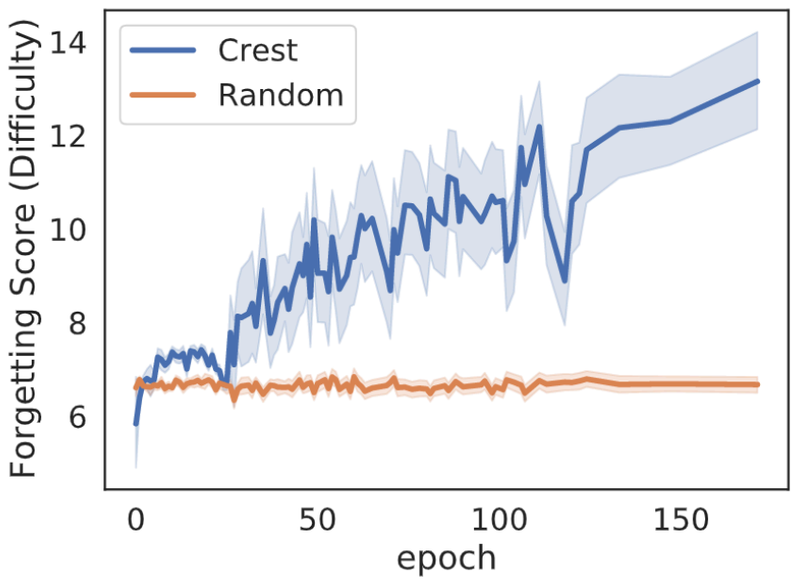


Scales to very large data!

CREST: Coresets for Data-efficient Deep Learning



Crest updates more in the beginning and less later in training



Trains on difficult-to-learn examples!

Similar framework can be used for learning robustly against **noisy labels** and **data poisoning attacks**

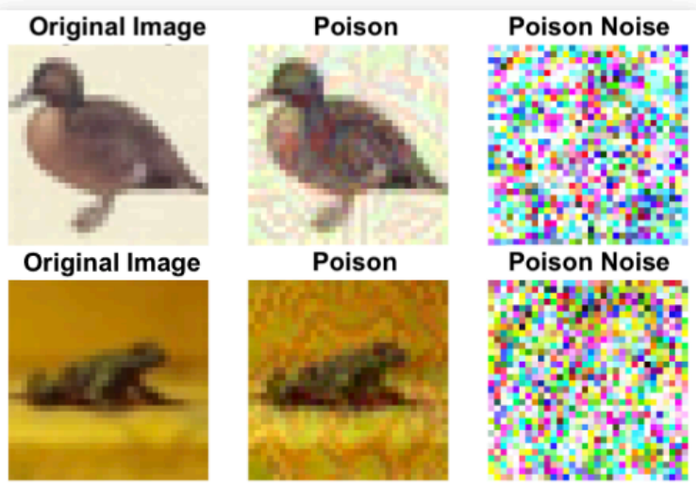
Smaller Higher-quality Data Improve Robustness

robustness

label noise

Robust traing against label nosie

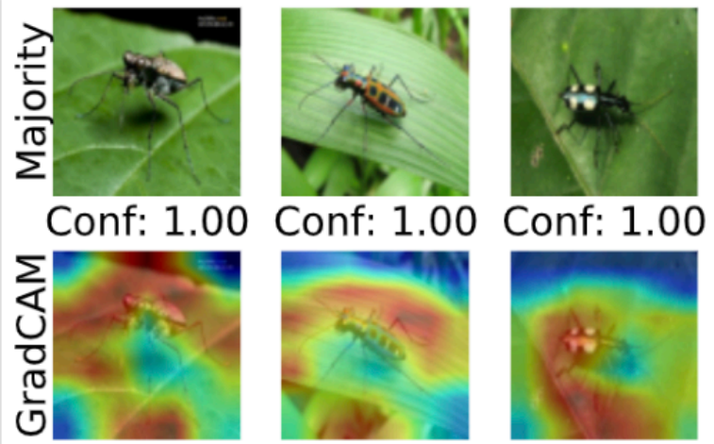
The subsets are not noisy



data poisoning

Robust traing against data poisoning

Poisons are clusters of size 1



spurious correlations

Robust traing against spurious correlations

Balance gradient clusters early in training

- Coresets for robust training of deep neural networks against noisy labels [ICML'20]
- Not all poisons are created equal: Robust training against data poisoning [ICML'22]
- Better Safe than Sorry: Pre-training CLIP against Targeted Data Poisoning and Backdoor Attacks [ICML'23]

There are also several heuristics!

Intuition: Find difficult-to-learn examples…

Forgettability score

- Forgetting event: when a data point is misclassified after begin correctly classified

- Unforgettables: data points that have no forgetting event

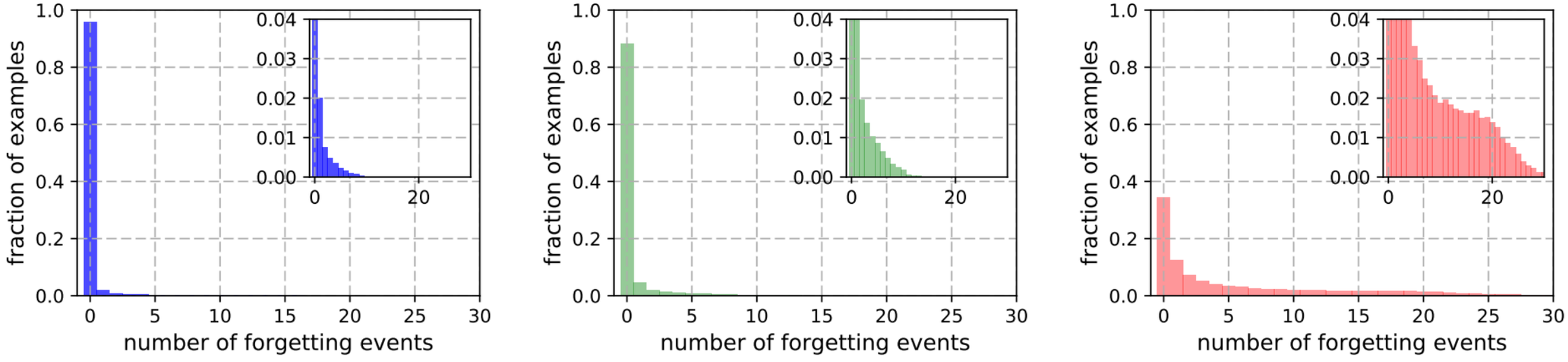


Figure 1: Histograms of forgetting events on (from left to right) *MNIST*, *permutedMNIST* and *CIFAR-10*. Insets show the zoomed-in y-axis.

An Empirical Study of Example Forgetting during Deep Neural Network Learning [ICLR'19]

Selecting forgettable examples

- Let's drop examples that are unforgettable (over multiple seeds)!

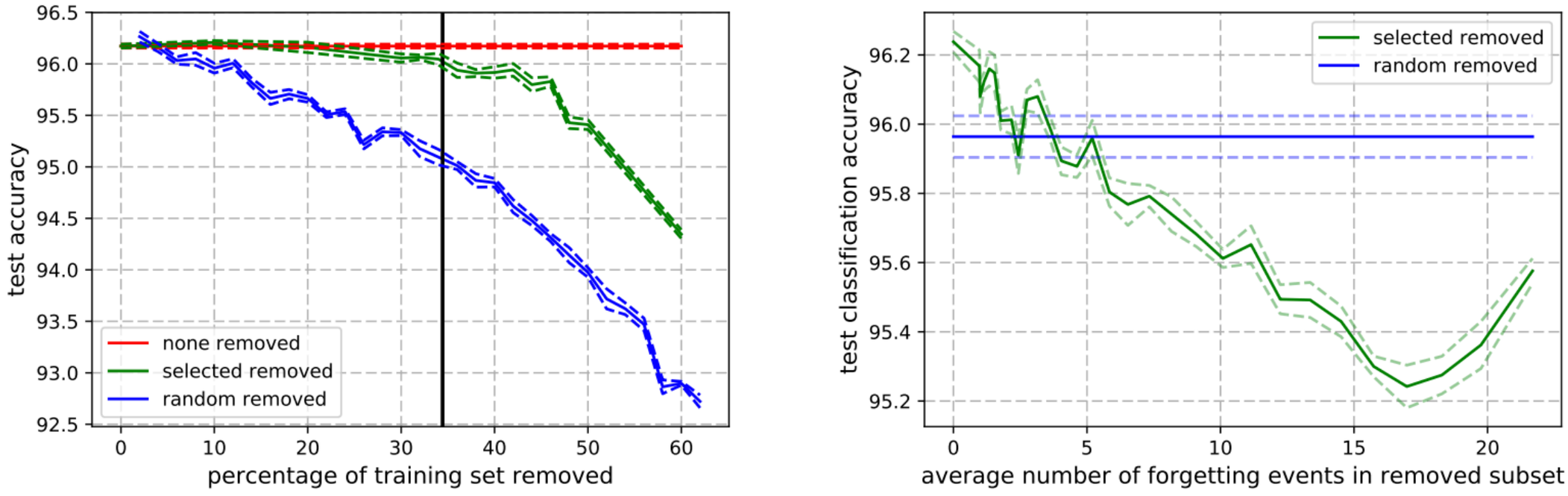
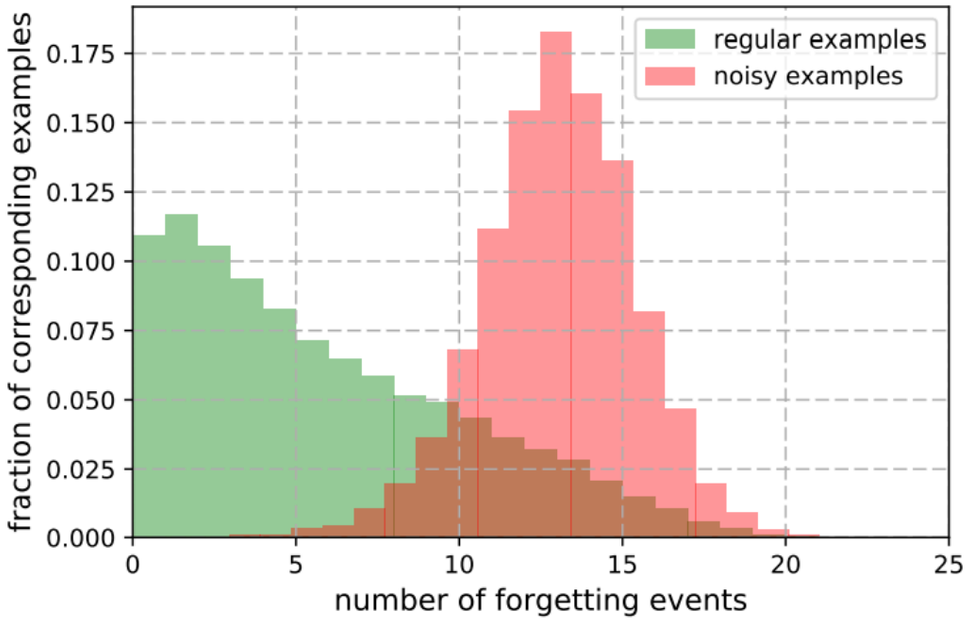


Figure 5: *Left* Generalization performance on *CIFAR-10* of ResNet18 where increasingly larger subsets of the training set are removed (mean +/- std error of 5 seeds). When the removed examples are selected at random, performance drops very fast. Selecting the examples according to our ordering can reduce the training set significantly without affecting generalization. The vertical line indicates the point at which all unforgettable examples are removed from the training set. *Right* Difference in generalization performance when contiguous chunks of 5000 increasingly forgotten examples are removed from the training set. Most important examples tend to be those that are forgotten the most.

How about noisy-labeled examples?

- Noisy-labeled examples are more forgettable



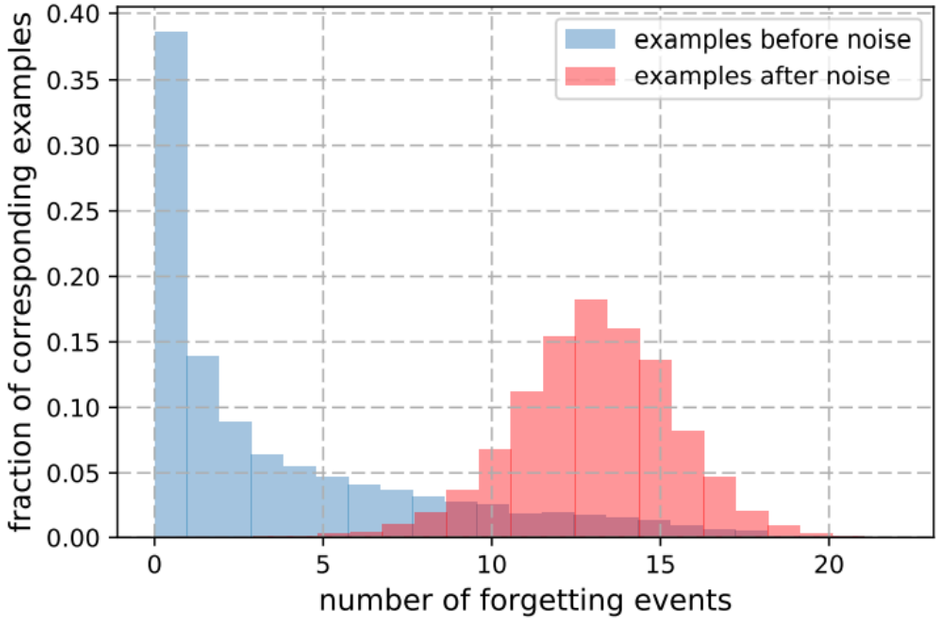


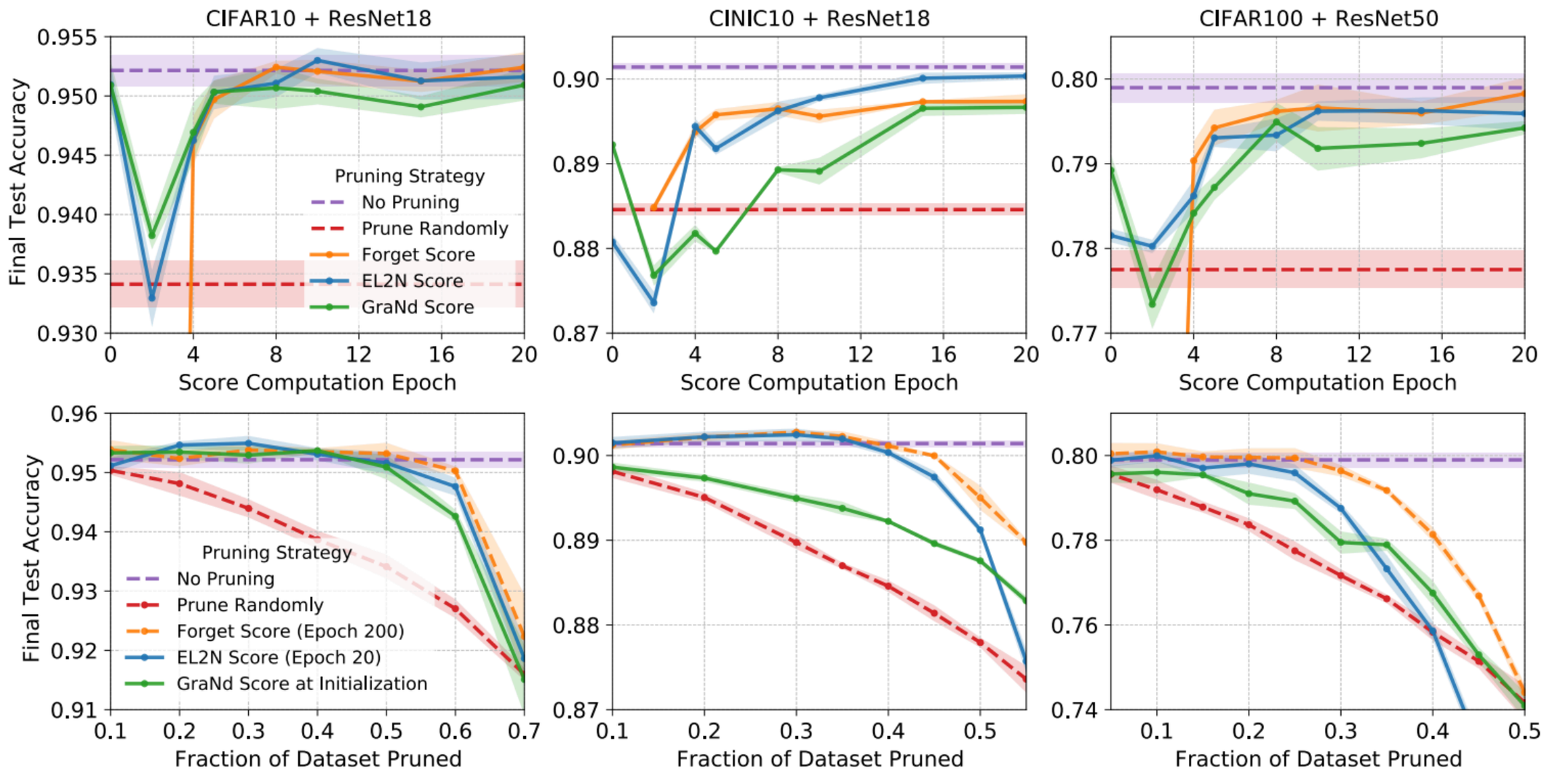
Figure 3: Distributions of forgetting events across training examples in *CIFAR-10* when 20% of labels are randomly changed. *Left.* Comparison of forgetting events between examples with noisy and original labels. The most forgotten examples are those with noisy labels. No noisy examples are unforgettable. *Right.* Comparison of forgetting events between examples with noisy labels and the same examples with original labels. Examples exhibit more forgetting when their labels are changed.

GraND and EL2N score

- GraNd score of a training example (x, y) at time t: $\mathbb{E}_{w_t}\|g_t(x,y)\|_2$

Gradient of loss w.r.t input to the last layer

- EL2N score of a training example (x, y) at time t: $\mathbb{E}\|p(w_t,x)-y\|_2$



Deep Learning on a Data Diet: Finding Important Examples Early in Training [NeurIPS'21]

How about noisy-labeled examples?

- Noisy-labeled examples have higher EL2N score

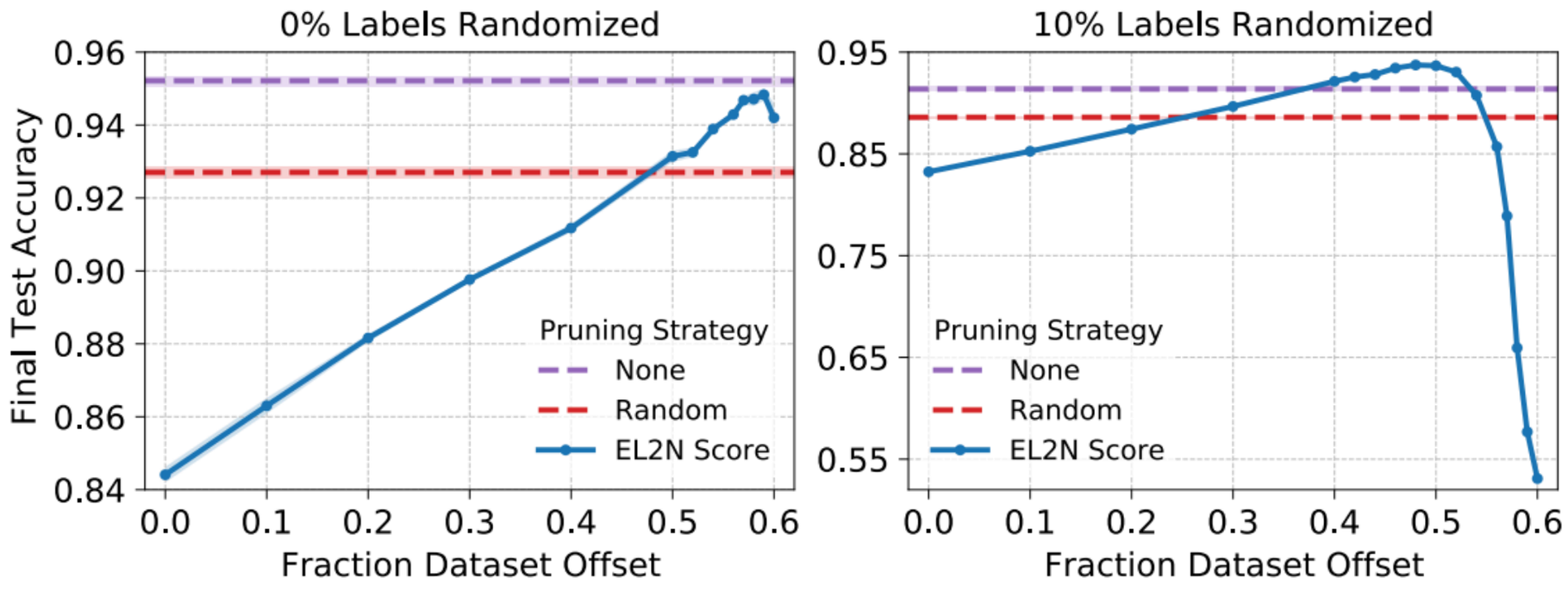


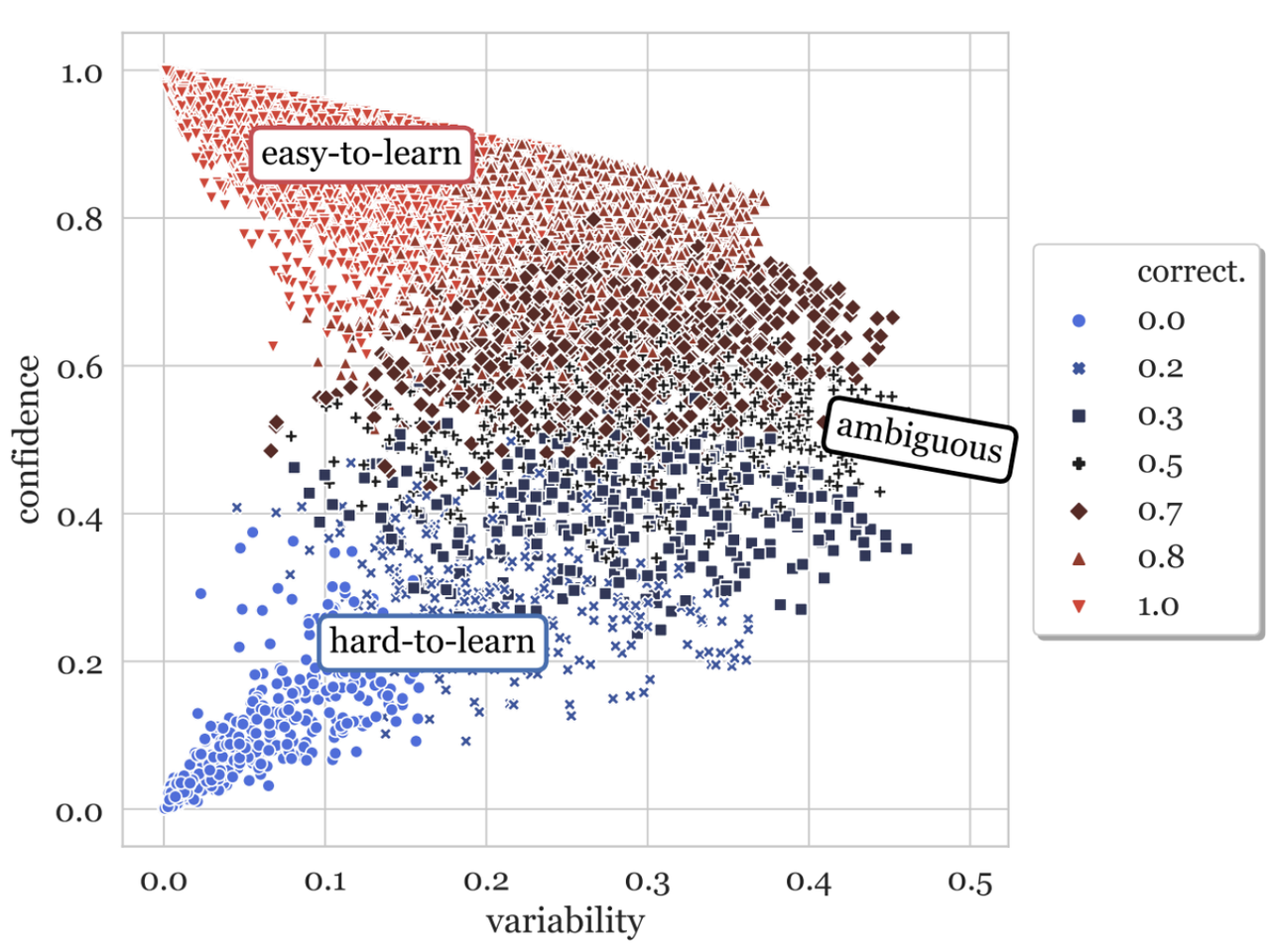
Figure 2: ResNet18 trained on a 40% subset of CIFAR-10 with clean *(left)* and 10% randomized labels *(right)*. The training subset contains the *lowest* scoring examples *after* examples with scores below the offset are discarded. Scores computed at epoch 10.

Dataset Cartography

- **High variability (ambiguous)**: true class probabilities fluctuate frequently during training

$$\hat{\sigma}_i = \sqrt{\frac{\sum_{e=1}^{E} \left(p_{\boldsymbol{\theta}^{(e)}}(y_i^* \mid \boldsymbol{x}_i) - \hat{\mu}_i\right)^2}{E}} \qquad \hat{\mu}_i = \frac{1}{E}\sum_{e=1}^{E} p_{\boldsymbol{\theta}^{(e)}}(y_i^* \mid \boldsymbol{x}_i)$$

- **High confidence, low variability**: model predicts them correctly and consistently
- **low confidence, low variability**: many of them are mislabeled



Dataset Cartography: Mapping and Diagnosing Datasets with Training Dynamics [ACL'20]

Does Not Work for Selecting Small Subsets

All fail here badly!

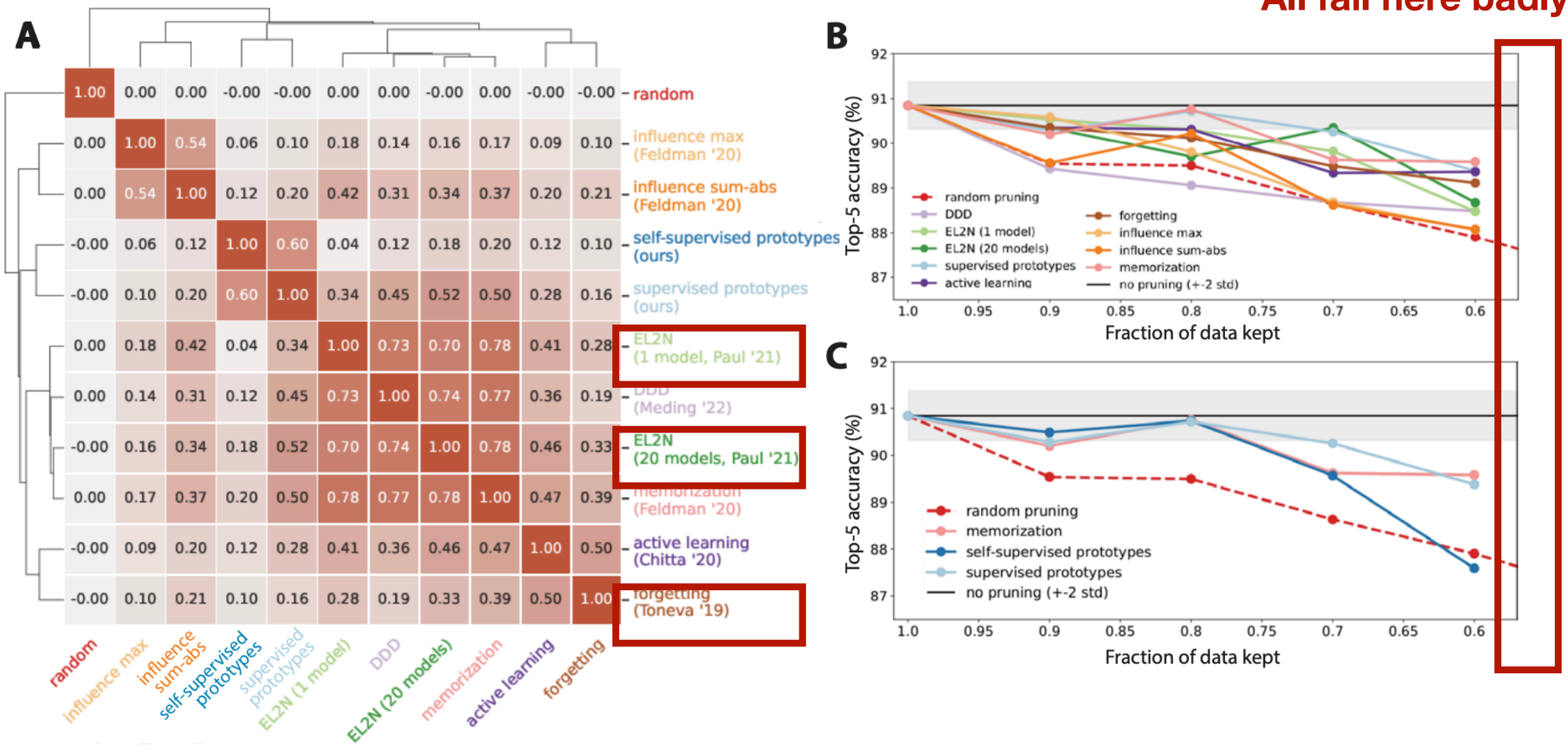
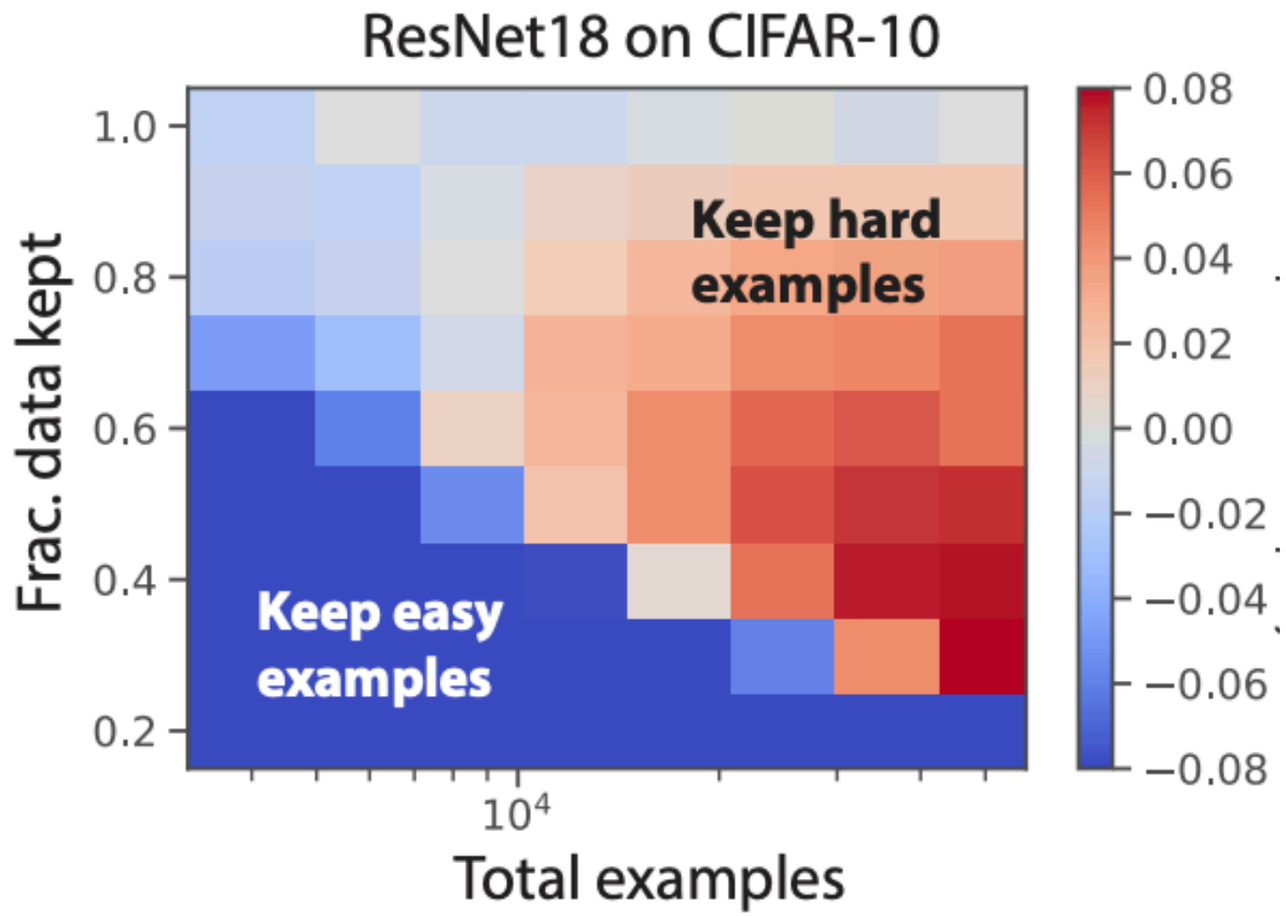


Figure 5: Dataset pruning at ImageNet scale. **A:** Spearman's rank correlation between all pairs of ImageNet metric scores, along with hierarchical clustering (as provided by `seaborn.clustermap`). **B:** Benchmarking supervised metrics on ImageNet (top-5 validation accuracy). **C:** Comparing top-5 performance on ImageNet when pruning according to the best existing supervised metric (memorization) and our supervised and self-supervised prototype metrics. In all 3 cases, training on 80% of ImageNet matches training on 100%. See App. B for pruning and training details.

Beyond neural scaling laws: beating power law scaling via data pruning, [NeurIPS'22]

How Much Can We Prune?

- If one does not have much data to start with, it is better to keep the easiest examples to avoid overfitting
- With abundant (scarce) initial data, one should retain only hard (easy) examples.



Any problem?

None of the previous metrics consider similarity between data points!!

Submodular optimization does!

Selecting Coresets with Submodular Maximization Yields a Curriculum

- Early in training, the most effective subsets for learning deep models are easy-to-learn examples.
- As training proceeds, the model learns the most from examples with increasing levels of learning difficulty.
- Interestingly, the model never requires training on easiest-to-learn examples

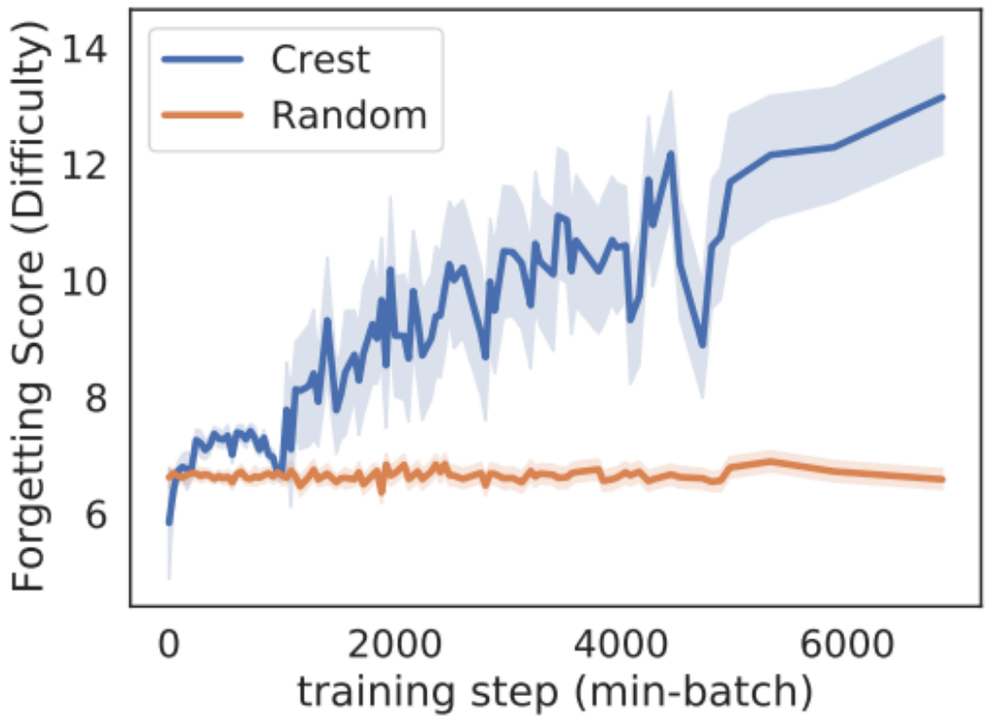


Figure 5: Average forgettability score of CREST coresets

Outline

- Motivation: why is data-efficiency important?
- Part 1: Data-efficient Supervised Learning
- Part 2: Data-efficient self-supervised Contrastive Pretraining
- Part 3: Foundation Models
  - 3a: Data-efficient Contrastive Language-Image Pretraining
  - 3b: Data-efficient Training of Large Language Models

Self-Supervised Learning on Large Datasets

- Supervised Learning requires high-quality labels
  - Getting high-quality labels can be very expensive!
  - Mislabeled examples drastically affect the performance

- How can we learn from massive datasets without labels?
  - **Pre-training *with Self-Supervision:*** Learning from large datasets without explicit labels!

SSL Pre-training on Large Datasets

- **Benefits**:
  1. **No labels** needed
  2. **Fast adaptation** to new tasks
  3. Superior robustness to **distribution shift**

- SSL Pre-Training:
  - **Contrastive Learning (CL)** self-supervised by augmentations of a large pool (e.g. 1M) of images

Self-supervised Contrastive Learning

- Contrastive Learning learns an **encoder** by:
  - **Pulling** augmentations from the **same example** closer together
  - **Pushing** augmentations from **different examples** further away

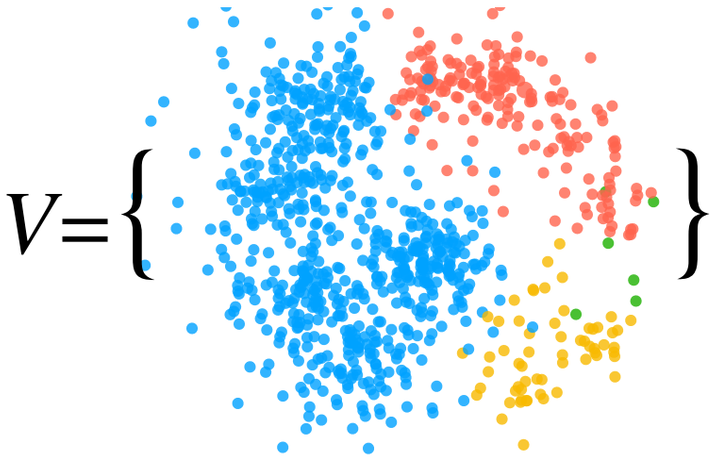
$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$$

Evaluation: Linear Probe

- 1- **Pretraining**: Learn representations for training data with CL

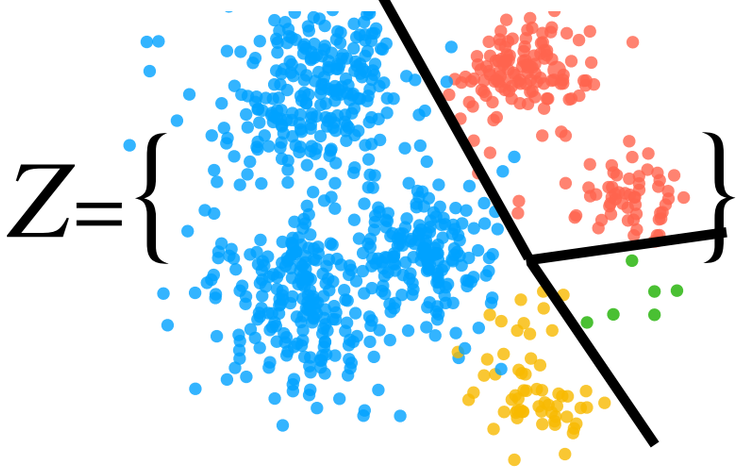
- 2- **Linear probe**: A linear model (linear probe) is trained on representations of the pre-trained encoder

Training Data: $\{x_i, i \in V\}$



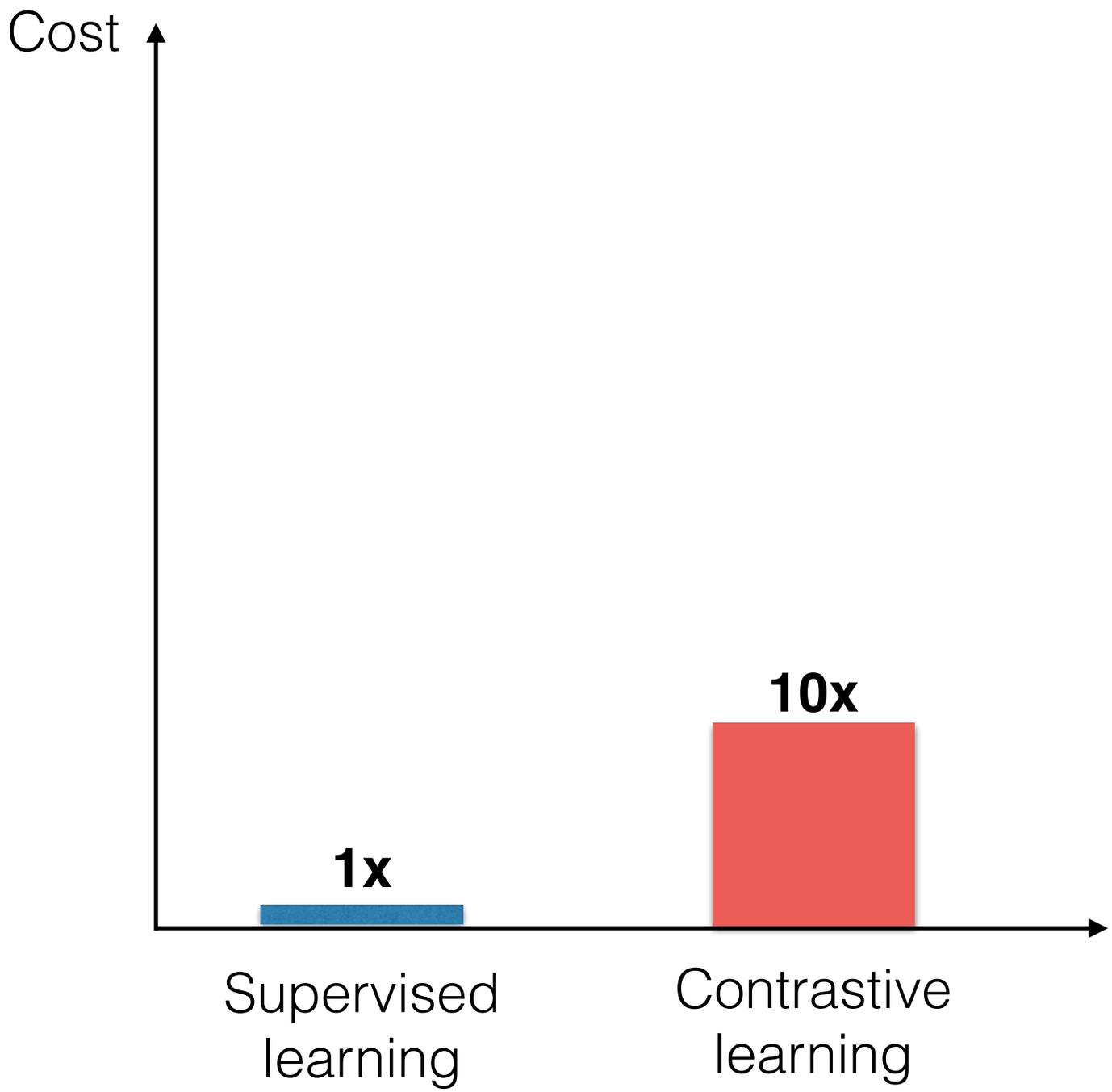
Contrastive Learning

$Z = \{f(x_i), i \in V\}$



Train a linear model on the representations

However, CL is Very Expensive!



Can we Improve the **Data-efficiency** of **SSL Pre-training** without Losing Performance?

Self-supervised Contrastive Learning

- **Minimizing the contrastive loss**:

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$$

- **Unfortunately**
  - Loss and gradient of every example depends on all the other examples!
  - Even other metrics (that are popular for supervised learning) cannot be used to identify importance of examples for CL

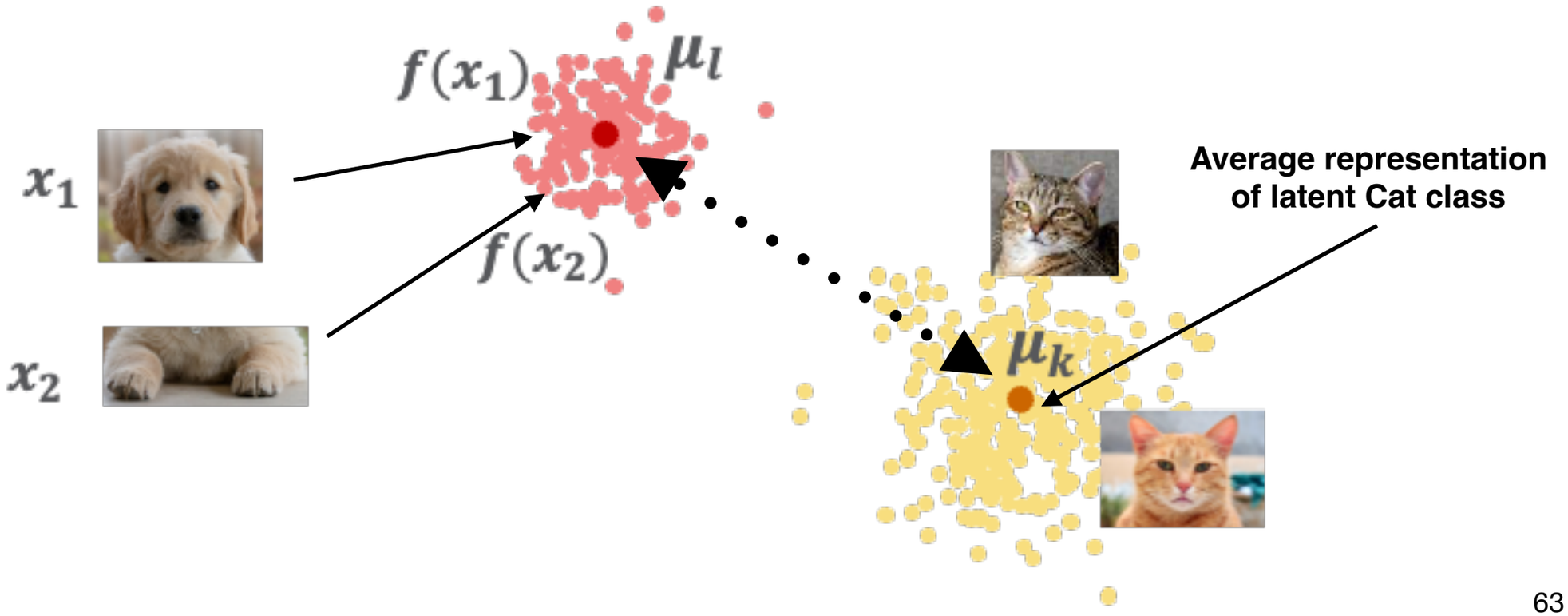
How does Self-supervised Contrastive Learning work?

- **Minimizing the contrastive loss:**

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$$

- **(1) Alignment:** Aligns augmentations of the same example

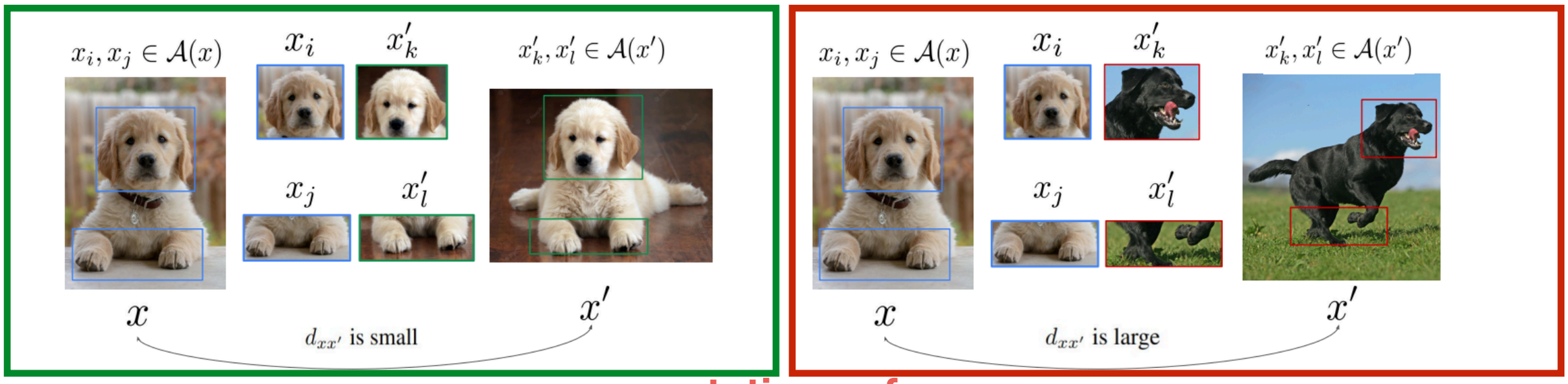
- **(2) Divergence**: Pushes centers of latent class representations apart



Can we find a coreset that preserves **alignment** and **divergence** of full data?

Which Examples Contribute the Most to CL?

- **Definition**: Expected augmentation distance (a useful notion of similarity)



augmentations of x

$$d_{i,j} = E_{x \in A(x_i), x' \in A(x_j)} \|x - x'\|$$

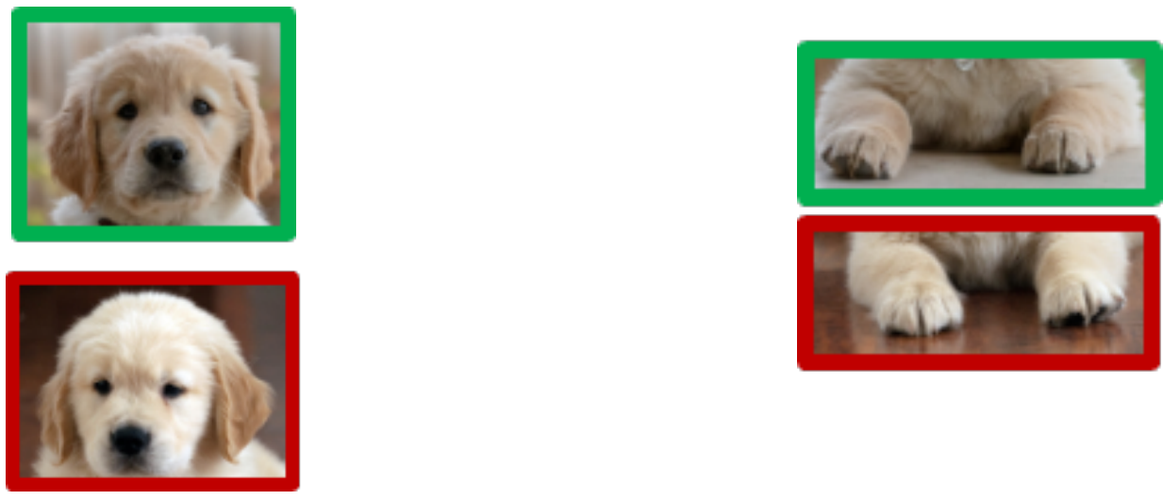
Data-Efficient Contrastive Self-supervised Learning: Most Beneficial Examples for Supervised Learning Contribute the Least [ICML'23]

Ensuring Alignment of Augmented Views

- (1) How can we preserve **alignment** of augmentations?
  - **Idea:** For every example that we discard, ensure there is an example in the subset with similar augmentations!

$$\min_{j \in S_k} d_{i,j} \leq \delta, \forall i \in V_k \backslash S_k$$

Derivation in Paper.



Intuition: select a diverse set of example in every latent class!

Data-Efficient Contrastive Self-supervised Learning: Most Beneficial Examples for Supervised Learning Contribute the Least [ICML'23]

Ensuring Divergence of Class Centers

- (2) How can we preserve **divergence** of class centers?
  - **Idea:** Find a subset that preserves centers of latent classes of full data!
  - Distance between centers of subset and full data can be bounded by expected augmentation distance

$$\nu_\mu = \|\mu_k - \mu_k^S\| \leq c \cdot \mathbb{E}_{i \in V, j \in S} d_{i,j}$$

Derivation in Paper.

Intuition: select the most central example in every latent class!

Data-Efficient Contrastive Self-supervised Learning: Most Beneficial Examples for Supervised Learning Contribute the Least [ICML'23]

SAS: Which Examples Contribute the Most to CL?

- Examples that contribute the most:
  - Preserving **alignment**: Select a **diverse** subset from every latent class
  - Preserving **divergence***:* Select **central** examples in every latent class

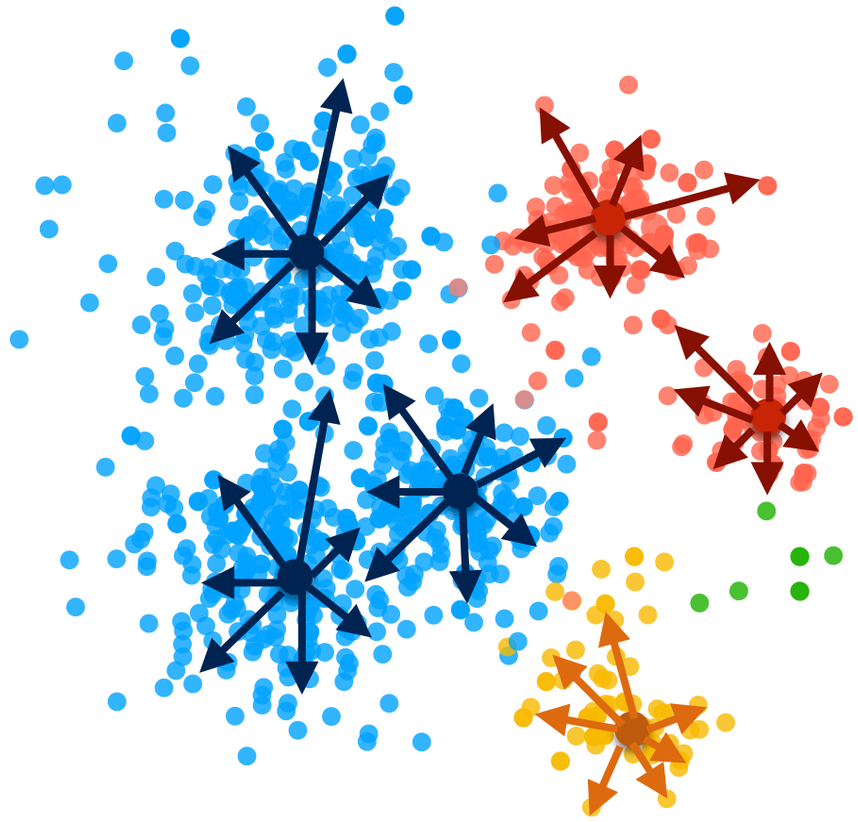
$$S_k = \arg\min_{S \subseteq V, |S| \leq r_k} \sum_{i \in V_k \setminus S_k} \sum_{j \in S_k} d_{i,j}$$

Submodular!!

augmentations of x

$$d_{i,j} = E_{x \in A(x_i), x' \in A(x_j)} \|x - x'\|$$

Expected augmentation distance



SAS: Which Examples Contribute the Most to CL?

- Two practical considerations:

1. How do we compute “Expected Augmentation Distance”

$$d_{i,j} = E_{\overbrace{x \in A(x_i), x' \in A(x_j)}^{\text{augmentations of } x}} \|x - x'\|$$

Expected augmentation distance

Can use a cheap proxy model (i.e. much smaller or partially trained model)!

2. How do we approximate latent classes? We don’t have labels!

Cheap Proxy Model + K-Means or Linear Probe with 1% labels

or

Guess latent classes using foundational models like CLIP

SAS: Which Examples Contribute the Most to CL?

- **Intuition:** Are representative of the all subclasses in a latent class
- **Theory:** (1) Align all the examples in the latent class

  (2) Preserves the centers of latent classes

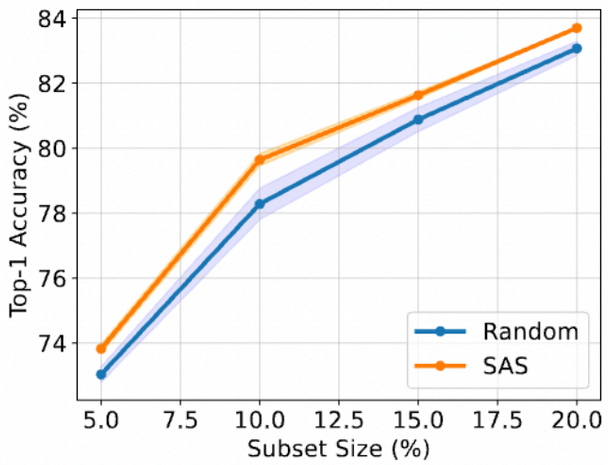
  (3) Guarantee generalization on full data

Theorem (informal): Contrastive learning on the subset grantees downstream generalization on the full data:

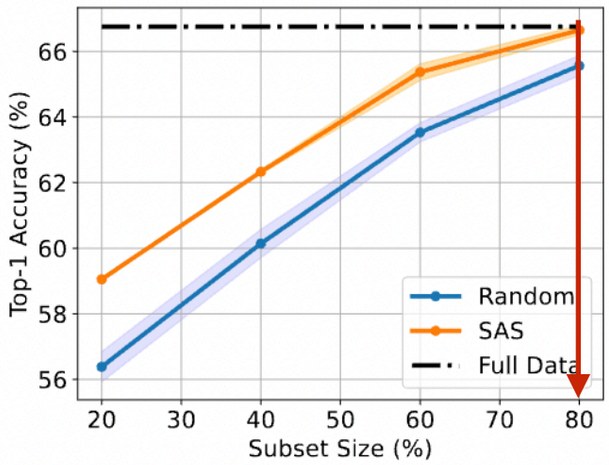
$$\xi(g_f^S(V)) \leq (1-\sigma) + R_\epsilon + \nu_R$$

Data-Efficient Contrastive Self-supervised Learning: Most Beneficial Examples for Supervised Learning Contribute the Least [ICML'23]

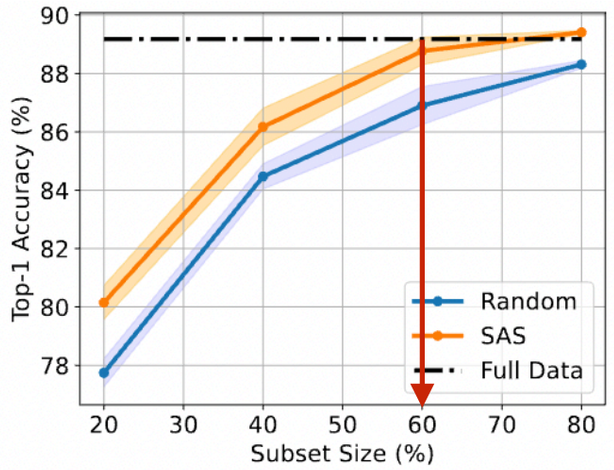
Easy Examples Contribute the Most to CL



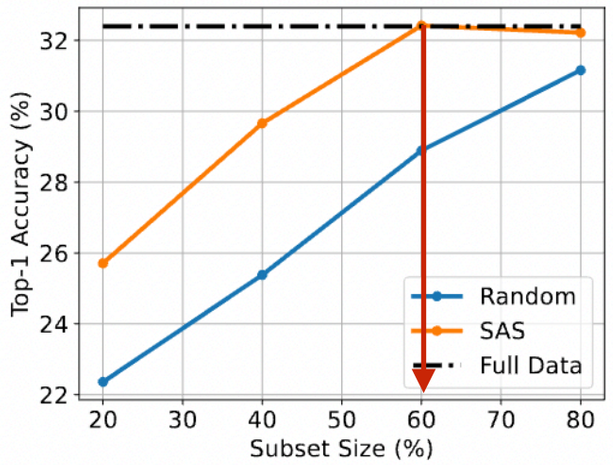
(a) CIFAR10



(b) CIFAR100



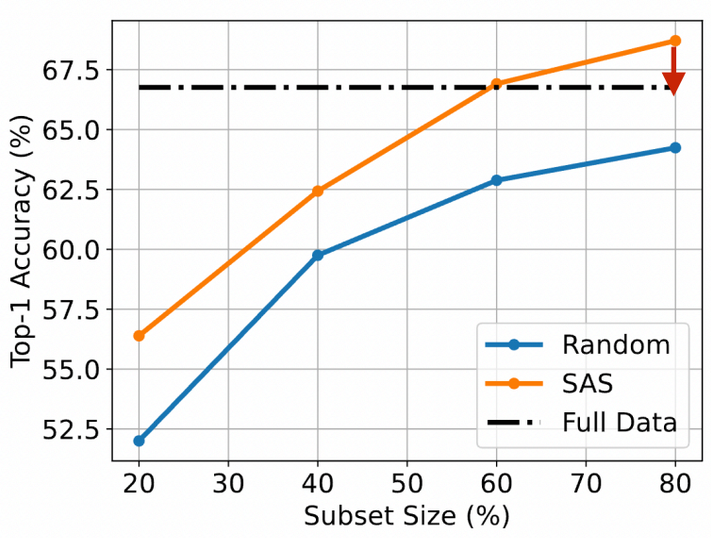
(c) STL10



(d) TinyImageNet (ResNet-18)

Figure 3. Downstream Classification Accuracy of *SAS* Subsets vs. Random Subsets (reporting mean and std over 3 runs).

Up to 40% of examples can be excluded!

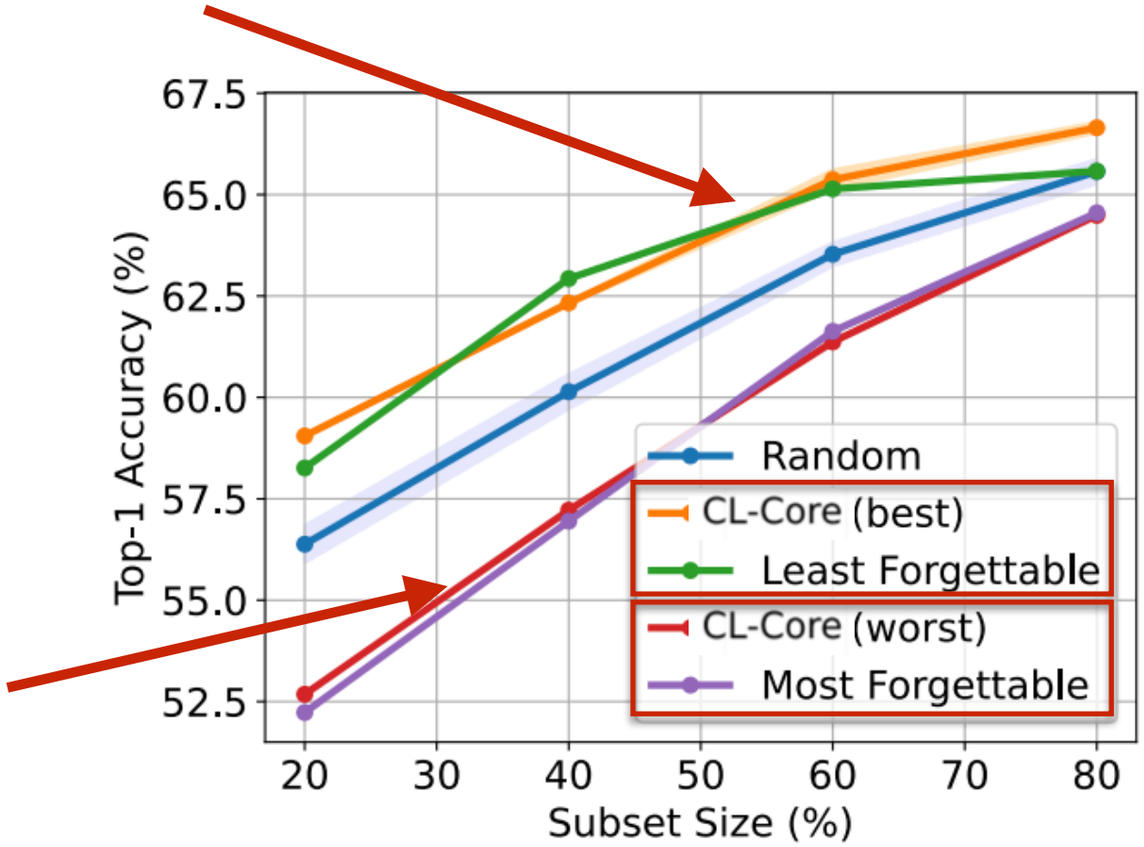


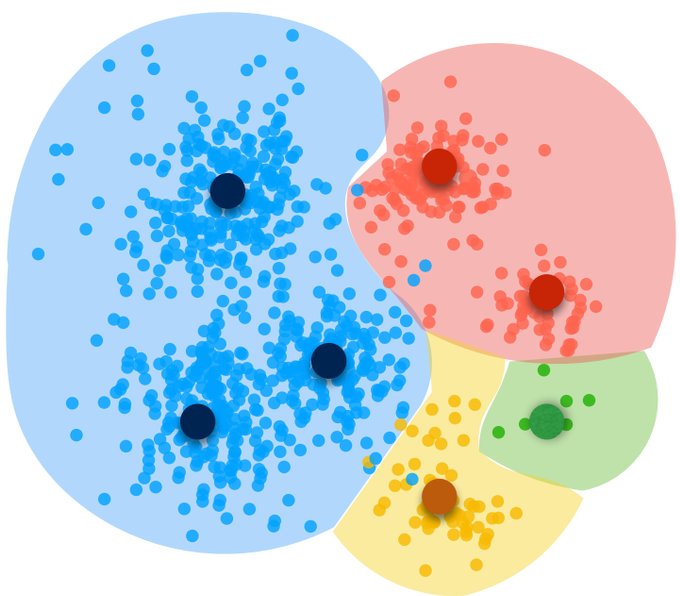
(a) BYOL (STL10)

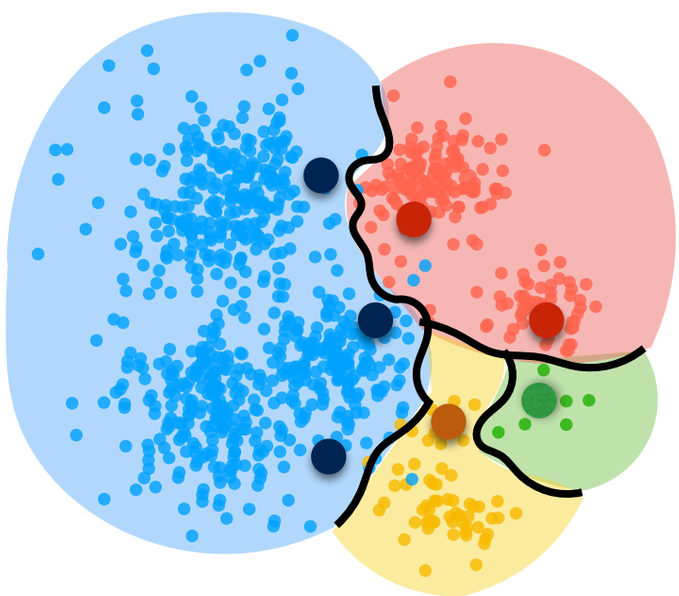
Better than full data!

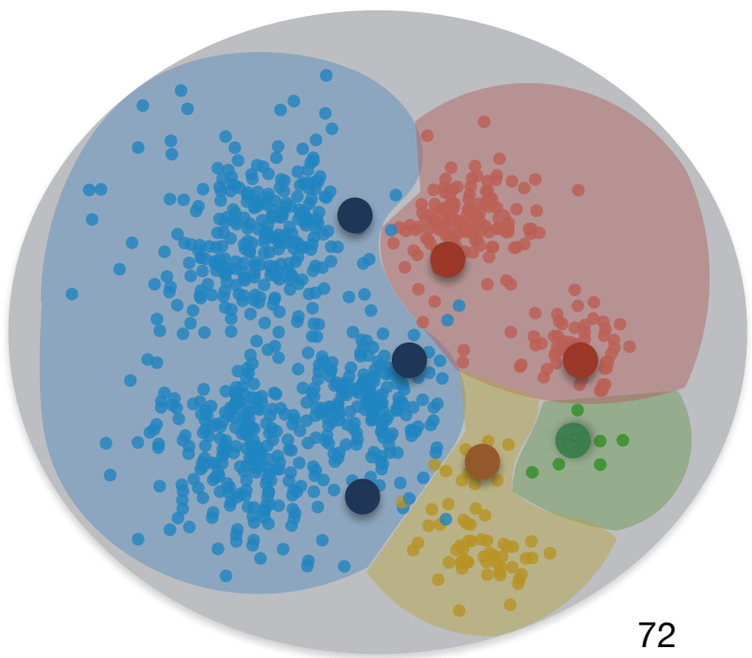
Which Examples Contribute the Most to CL?

- **Easy-to-learn examples** (for supervised learning) contribute the most to CL!
  - **In contrast**: easy-to-learn examples can be excluded without harming supervised learning



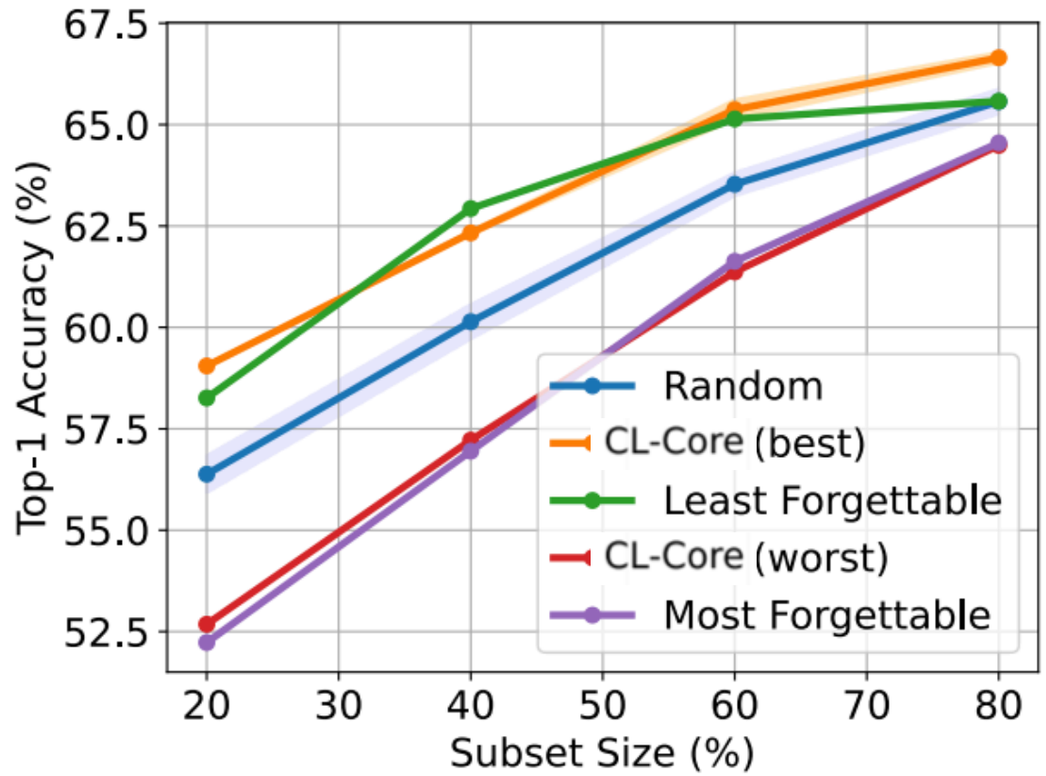


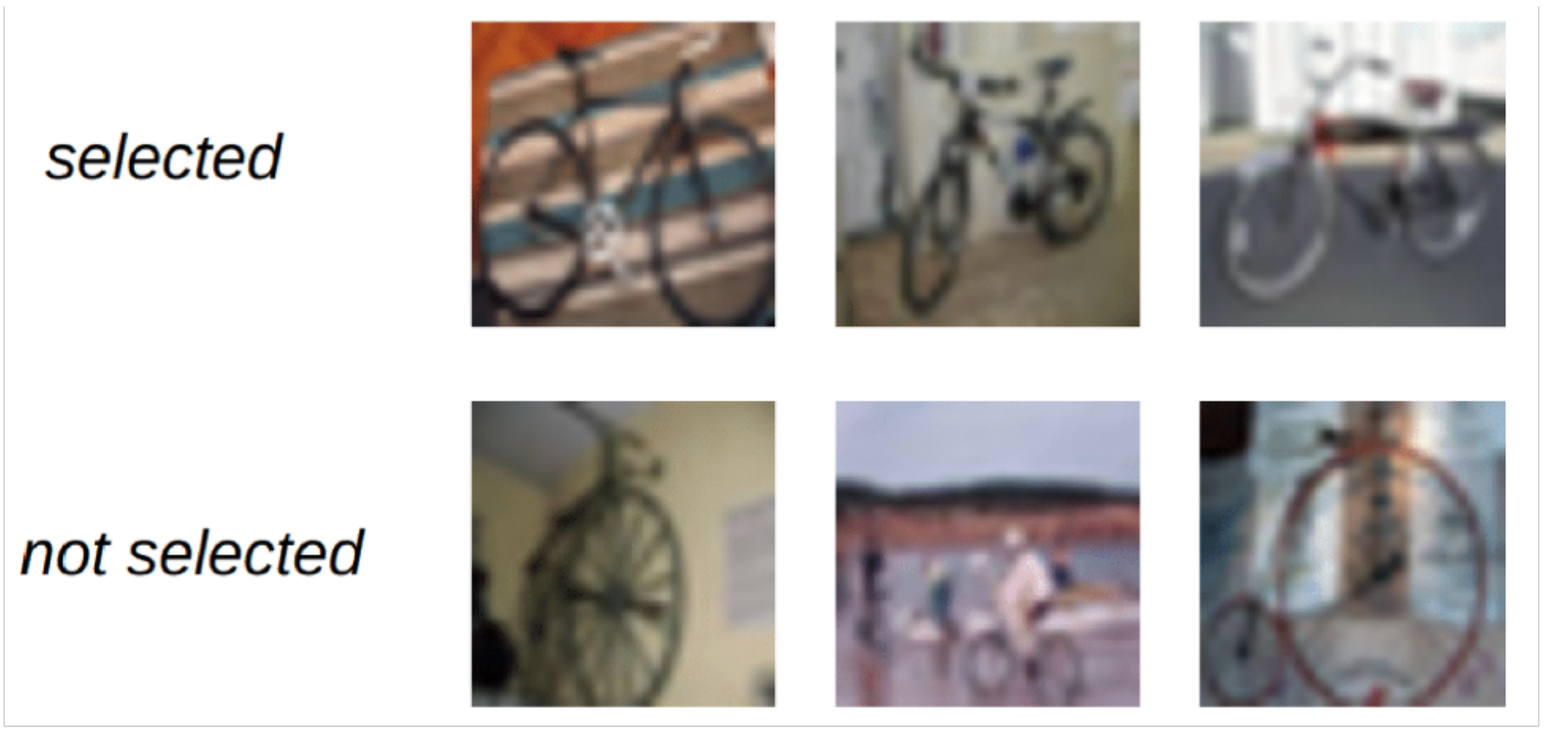




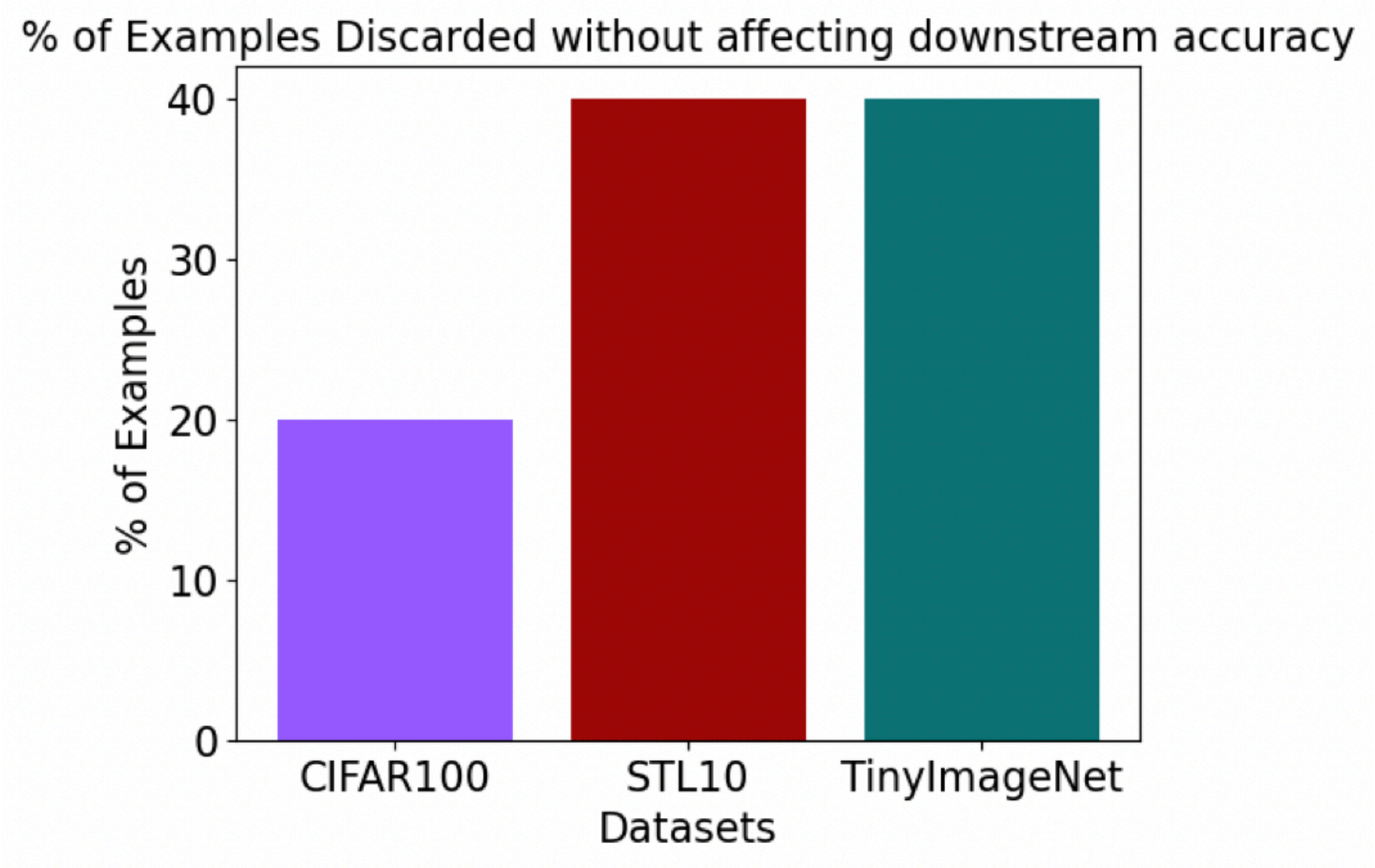
Which Examples Contribute the Most to CL?

- **Easy-to-learn examples** (for supervised learning) contribute the most to CL!
  - **In contrast**: easy-to-learn examples can be excluded without harming supervised learning





Data-efficient Self-supervised Contrastive Learning



Speeds up Training by 40%

Outline

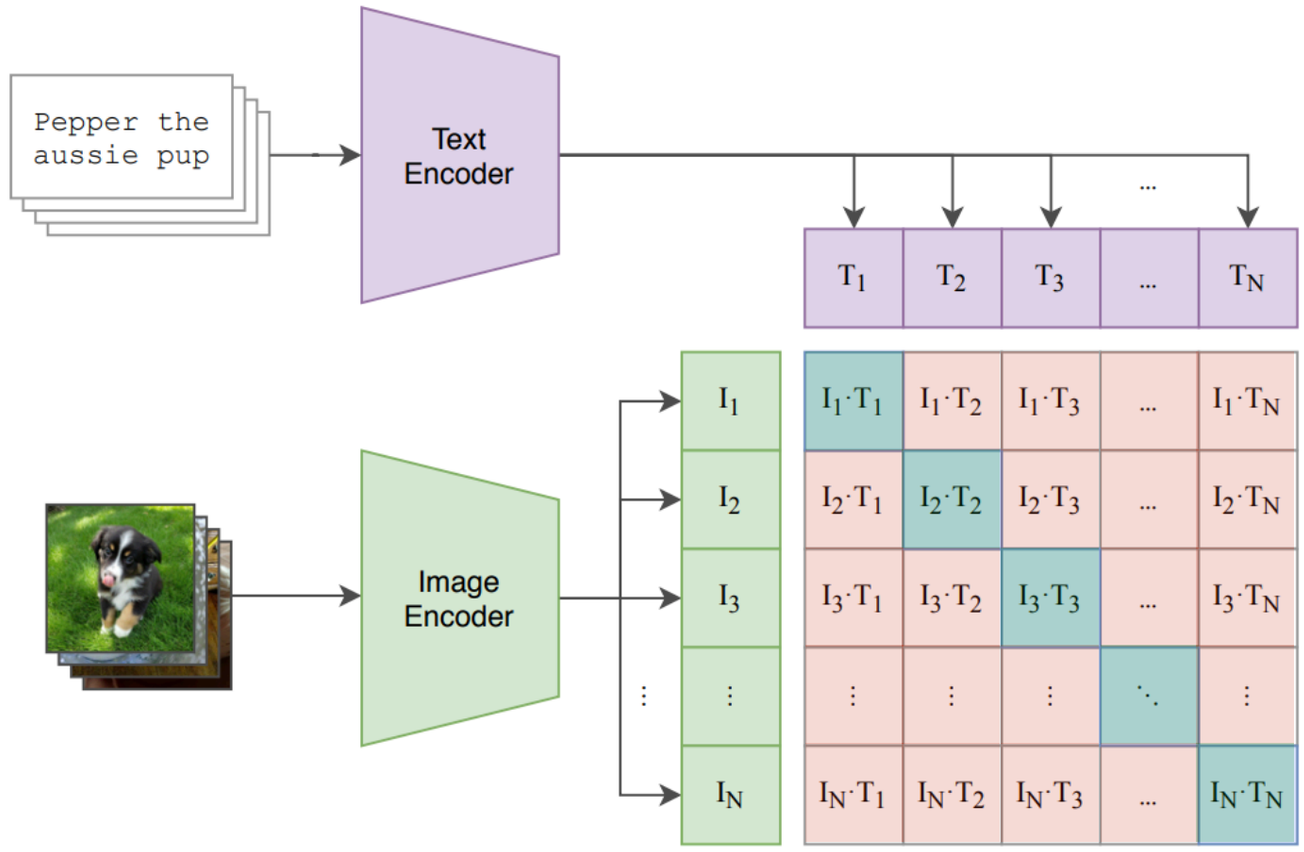
- Motivation: why is data-efficiency important?
- Part 1: Data-efficient Supervised Learning
- Part 2: Data-efficient self-supervised Contrastive Pretraining
- Part 3: Foundation Models
  - 3a: Data-efficient Contrastive Language-Image Pretraining
  - 3b: Data-efficient Training of Large Language Models

Outline

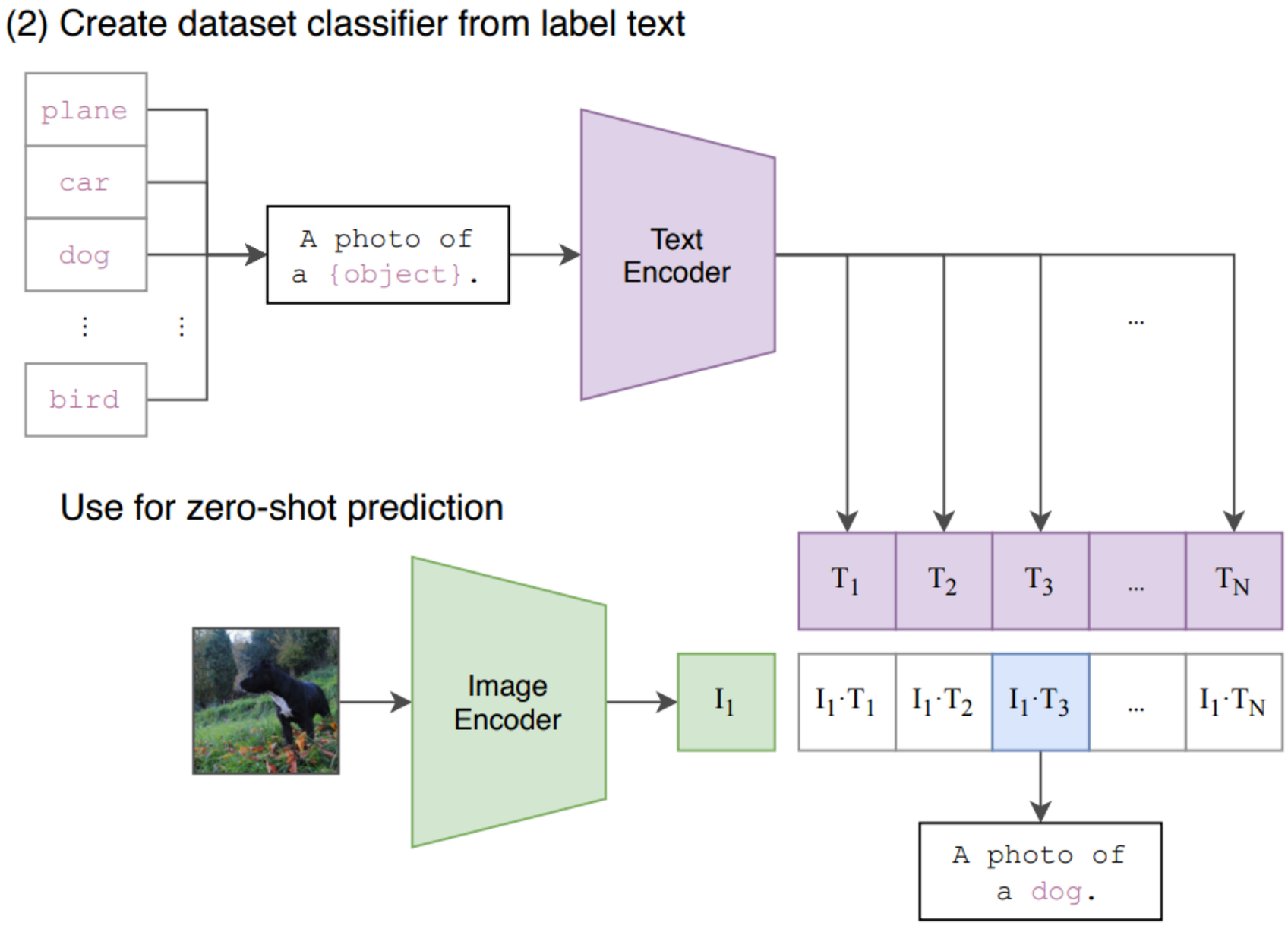
- Motivation: why is data-efficiency important?
- Part 1: Data-efficient Supervised Learning
- Part 2: Data-efficient self-supervised Contrastive Pretraining
- Part 3: Foundation Models
  - 3a: Data-efficient Contrastive Language-Image Pretraining
  - 3b: Data-efficient Training of Large Language Models

Contrastive Language-Image Pre-training (CLIP)

- CLIP learns **image and text encoders** by:
  - **Pulling** representations of **paired** image-captions **closer** together
  - **Pushing** representations of **unpaired** image-captions **further** away



Contrastive Language-Image Pre-training (CLIP)



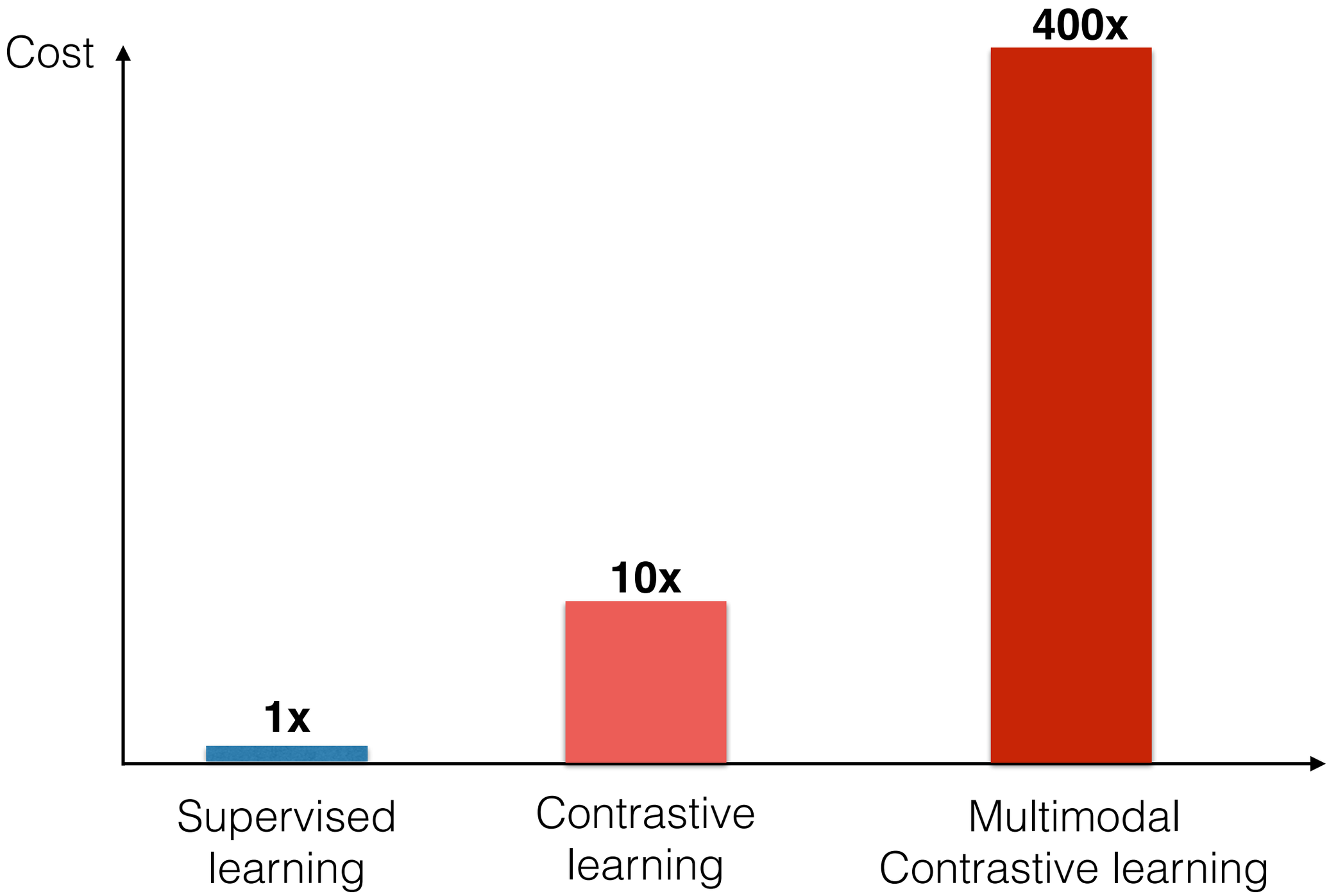
Enables zero-shot transfer of the model to downstream tasks

CLIP Pre-training on Large Datasets

- **Benefits**:
  1. **Zero-shot** classification
  2. Even better robustness to **distribution shift**

However, CLIP requires a LOT more data!

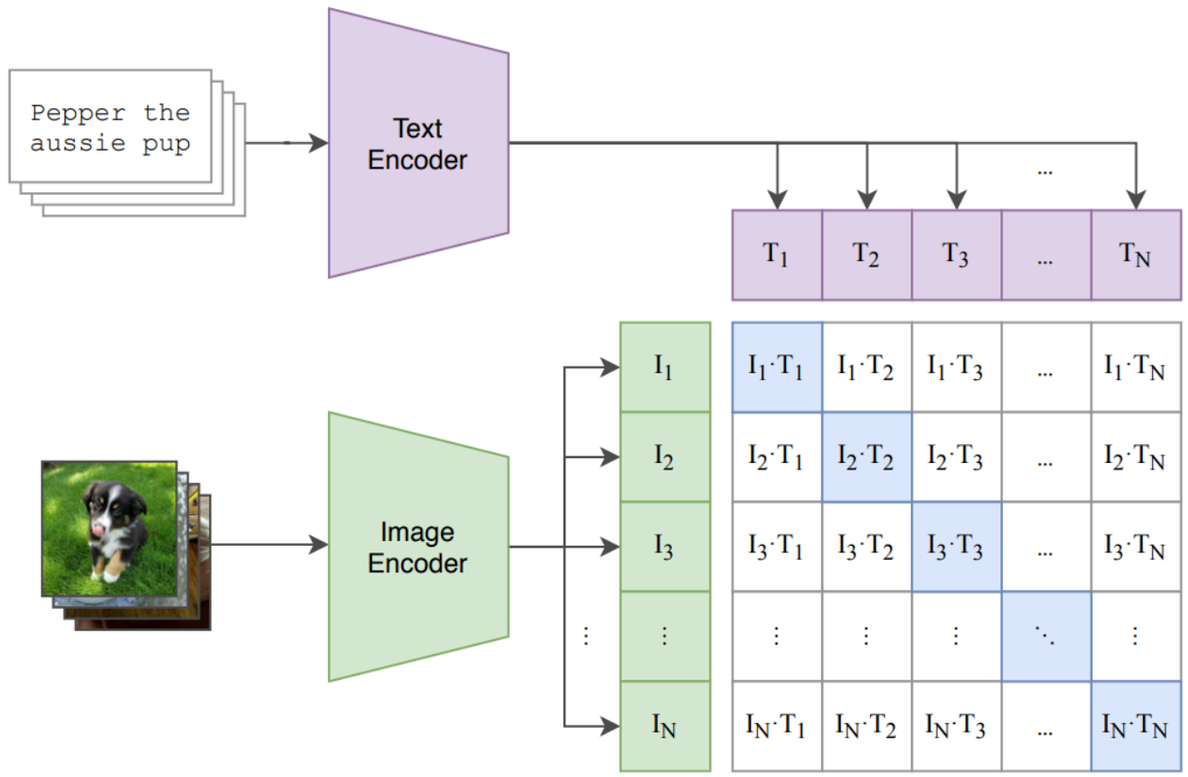
However, CLIP is Very Expensive!



Can we find a **coreset** that **guarantees** similar **zero-shot performance** to training on full data?

Contrastive Language-Image Pre-training (CLIP)

$$\mathcal{L}_{\text{CLIP}} = -\frac{1}{2N}\sum_{j=1}^{N}\log\left[\frac{\exp\left(\left\langle \boldsymbol{z}_j^I, \boldsymbol{z}_j^T \right\rangle / \tau\right)}{\sum_{k=1}^{N}\exp\left(\left\langle \boldsymbol{z}_j^I, \boldsymbol{z}_k^T \right\rangle / \tau\right)}\right] - \frac{1}{2N}\sum_{k=1}^{N}\log\left[\frac{\exp\left(\left\langle \boldsymbol{z}_k^I, \boldsymbol{z}_k^T \right\rangle / \tau\right)}{\sum_{j=1}^{N}\exp\left(\left\langle \boldsymbol{z}_j^I, \boldsymbol{z}_k^T \right\rangle / \tau\right)}\right]$$



- Loss and gradient of every example depends on all the other examples!

Cannot rely on loss or gradient again!

How does CLIP work?

- The **cross-covariance** of data determines the learning dynamics of CLIP [NGDJZZ '23]

$$C_D^V = \frac{1}{|V|} \sum_{i \in V} (x_V^i - \mu_V)(x_L^i - \mu_L)^T \qquad \mu_V = \mathbb{E}x_V, \mu_L = \mathbb{E}x_L,$$

A **beagle** on a **couch**

Intuition: cross-covariance captures how keywords in the caption correlate with objects in the image

Can we find a subset that preserves **cross-covariance** of full data?

Objective: Preserving cross-covariance

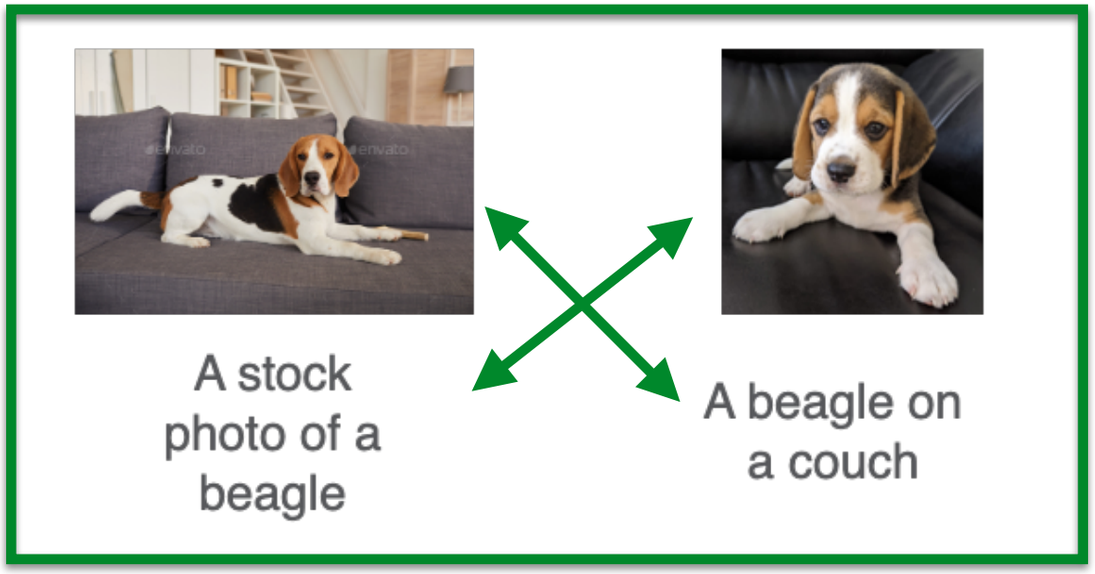
$$C_D^V = \frac{1}{|V|}\sum_{i\in V}(x_V^i - \mu_V)(x_L^i - \mu_L)^T \qquad \mu_V = \mathbb{E}x_V, \mu_L = \mathbb{E}x_L,$$

- To preserve **cross-covariance** of data:

1. Capture cross-covariance in each latent class to ensure **Alignment of Modalities** for full data **[Diversity]**

2. Preserve **Latent Class Centers** in both modalities simultaneously **[Centrality]**

Cross Modal Similarity



- A vision encoder and language encoder trained with the CLIP ensure a **high cosine similarity** for representations of **correlated captions** and **images**

- Hence, we use the r**epresentations of a proxy model** f_V, f_L trained with the CLIP loss to **approximate cross-modal similarity**

Definition: $csim(i,j) = f_V(x_V^i) \cdot f_L(x_L^j) + f_V(x_V^j) \cdot f_L(x_L^i)$

Objective: Preserving Cross-covariance

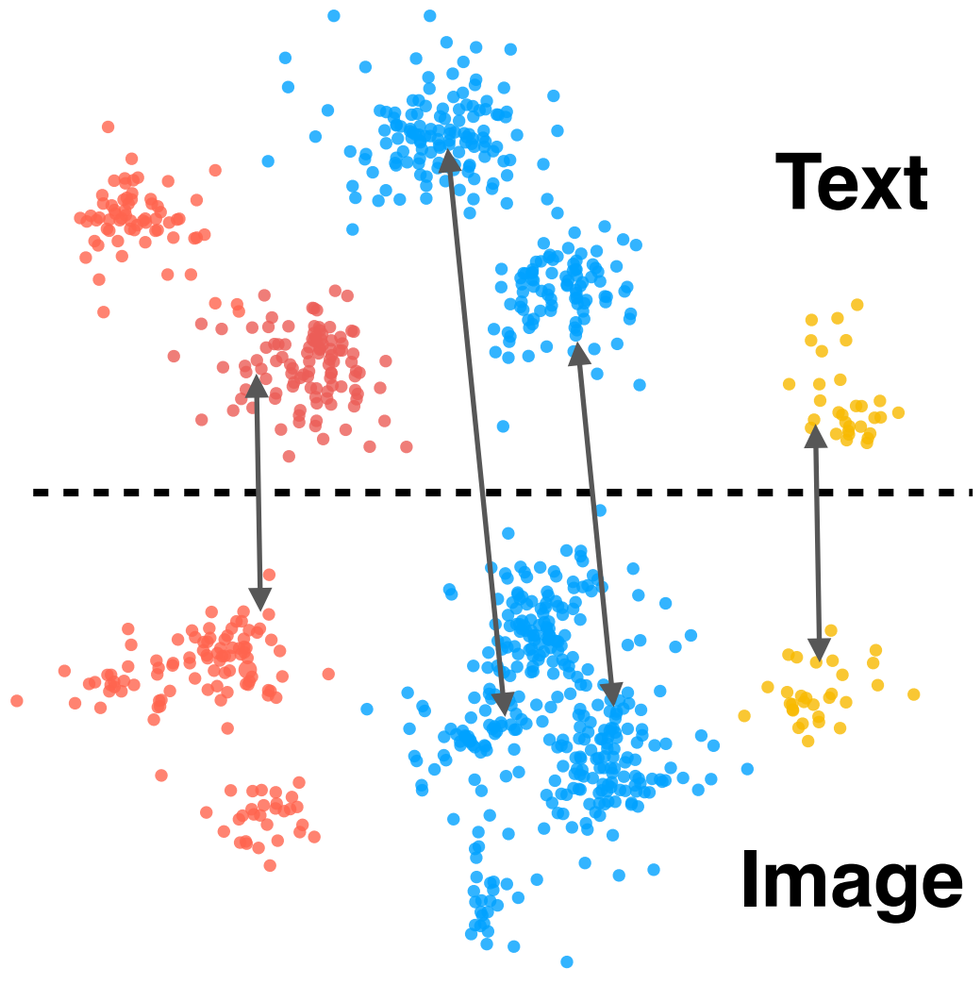
- To preserve **cross-covariance** of data:

  1. Capture cross-covariance in each latent class to ensure **Alignment of Modalities** for full data

(a.k.a. CLIP Score)

$$S_k^* \in \arg\max_{S_k \subseteq V_k} \sum_{i \in S_k} csim(i, i)$$

Intuition: A good proxy results in high cosine similarity for central examples in highly correlated areas

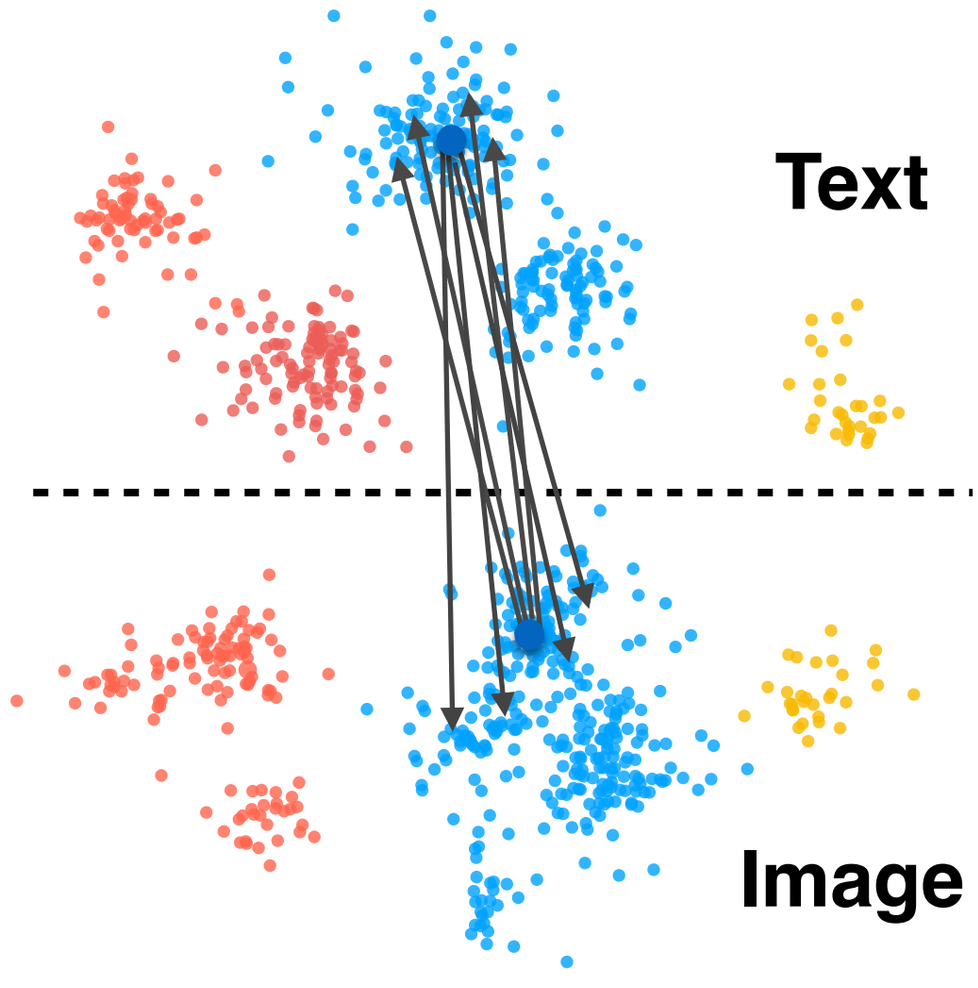


Objective: Preserving Cross-covariance

- To preserve **cross-covariance** of data:

  2. Preserve **Latent Class Centers** in both modalities simultaneously

$$S_k^* \in \arg\max_{S_k \subseteq V_k} \sum_{i \in V_k} \sum_{j \in S_k} csim(i, j)$$



Intuition: Selects image-caption pairs s.t. the image is similar to many captions and caption is similar to many images

ClipCov: Preserving cross-covariance

$$C_D^V = \frac{1}{|V|}\sum_{i\in V}(x_V^i - \mu_V)(x_L^i - \mu_L)^T \qquad \mu_V = \mathbb{E}x_V, \mu_L = \mathbb{E}x_L,$$

- To preserve **cross-covariance** of data:

1. Capture cross-covariance in each latent class to ensure **Alignment of Modalities** for full data **[Diversity]**

2. Preserve **Latent Class Centers** in both modalities simultaneously **[Centrality]**

$$S_k^* \in \arg\max_{S_k \subseteq V_k} \sum_{i\in V_k, j\in S_k} csim(i,j) + \sum_{i\in S_k} csim(i,i)$$

Non-monotone submodular function, subset can be found **efficiently** from large datasets!

ClipCov: Preserving cross-covariance

$$C_D^V = \frac{1}{|V|} \sum_{i \in V} (x_V^i - \mu_V)(x_L^i - \mu_L)^T \qquad \mu_V = \mathbb{E}x_V, \mu_L = \mathbb{E}x_L,$$

- To preserve **cross-covariance** of data:

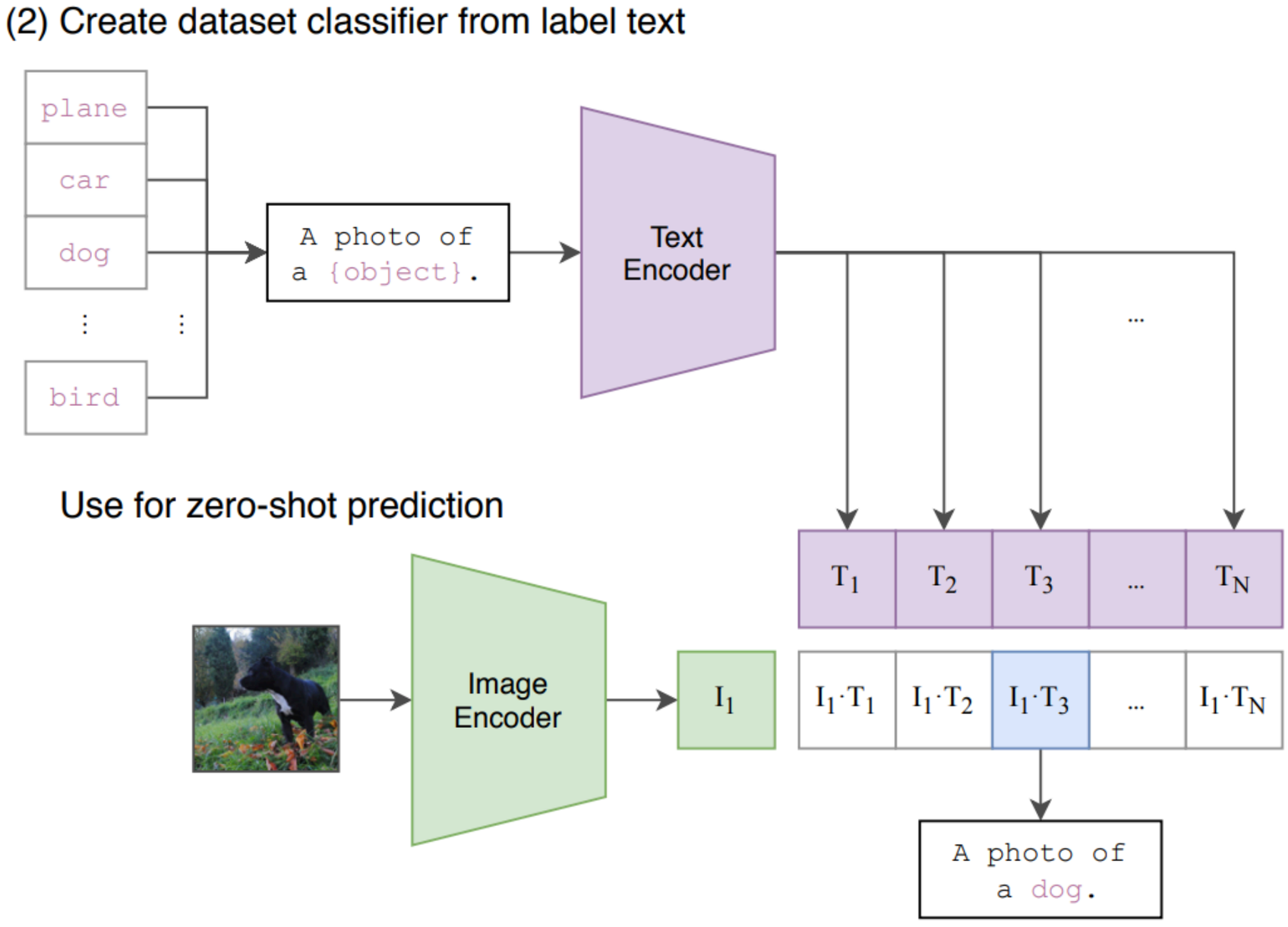
  1. Capture cross-covariance in each latent class to ensure **Alignment of Modalities** for full data **[Diversity]**

  2. Preserve **Latent Class Centers** in both modalities simultaneously **[Centrality]**

CLIP-Cov subsets ensure similar cross covariance and hence similar dynamics to training on full data

Data-Efficient Contrastive Language-Image Pretraining: Prioritizing Data Quality over Quantity [AISTATS'24]

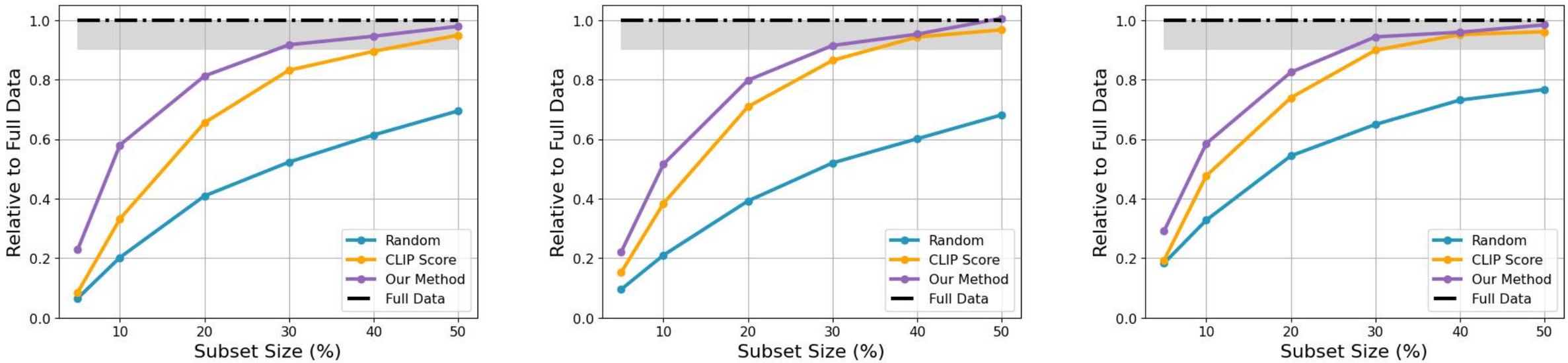
ClipCov: Getting Latent Classes



We can use any fine-grained set of labels (e.g. ImageNet) and our proxy model to get approximate latent classes

Coresets of Different Size from CC3M

Can discard ~50% data!



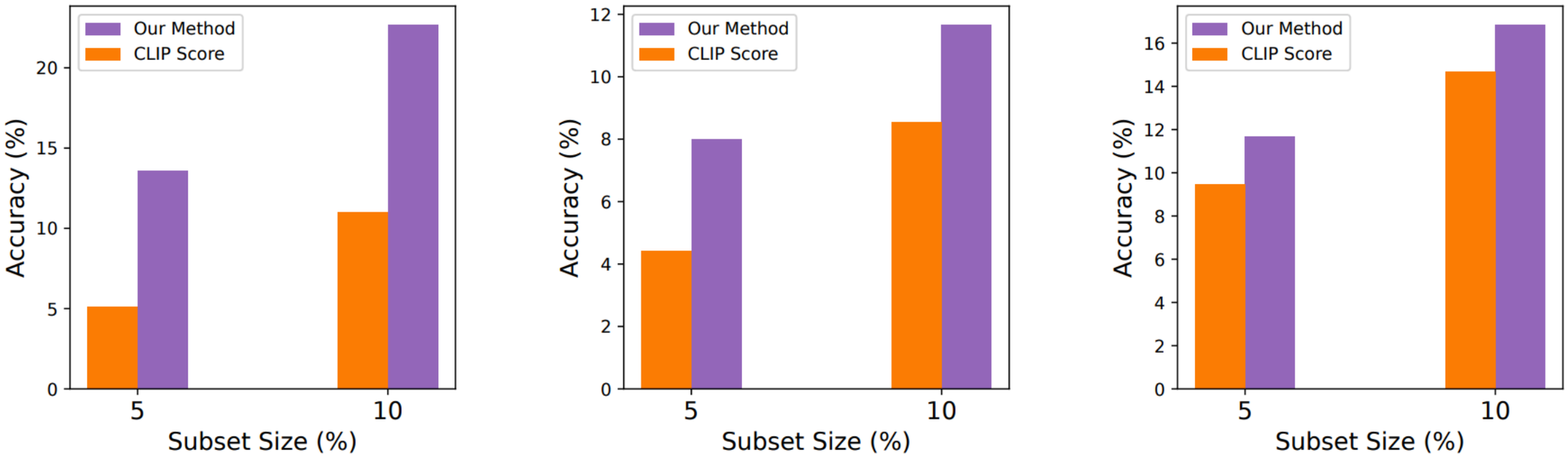
(a) ImageNet (b) ImageNet Dist. Shift (c) Avg. over 11 Datasets

Performance across subset of different sizes selected from Conceptual Captions (CC) 3M

Data-Efficient Contrastive Language-Image Pretraining: Prioritizing Data Quality over Quantity [AISTATS'24]

Small Coresets from CC12M

Over 2x Performance on CC12M!



(a) ImageNet (b) ImageNet Dist. Shift (c) Avg. over 11 Datasets

Performance across subset of sizes 5% and 10% from CC12M

Data-Efficient Contrastive Language-Image Pretraining: Prioritizing Data Quality over Quantity [AISTATS'24]

There are also several heuristics!

CLIP Score

- **Idea:** Select image-caption pairs with very similar image and caption representations or *high CLIP Score*

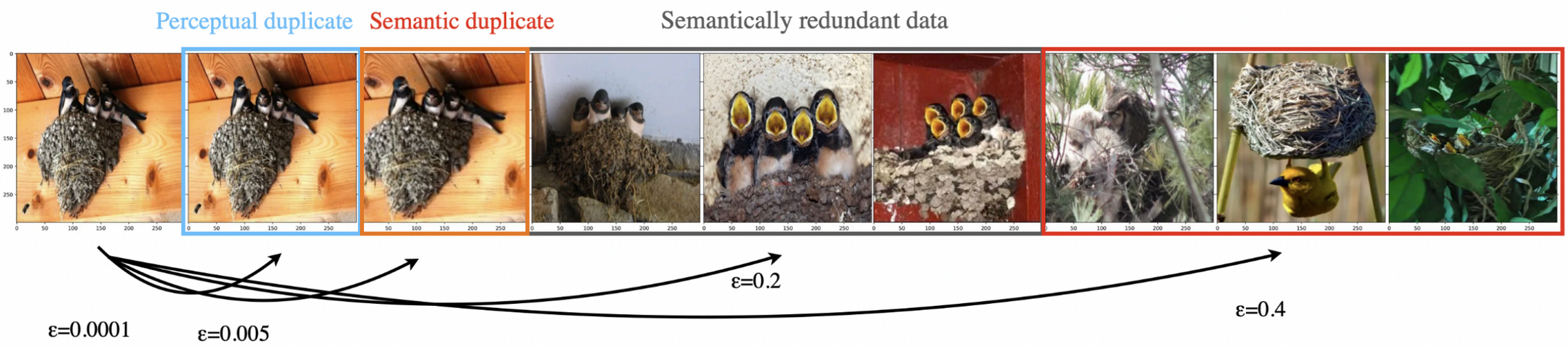
$$S_k^* \in \arg\max_{S_k \subseteq V_k} \sum_{i \in S_k} csim(i, i)$$

- **Note:** We explained why this works (ensures alignment/ diversity)!

SemDeDup

- **Idea:** Image-Caption datasets often have many duplicate images -> can we de-duplicate them for efficiency?

- **Method:** Cluster images using CLIP image embeddings and remove examples in same cluster with cosine similarity greater than 1 - ϵ



C-RHO

- **Idea:** Select examples that are “learnable” but “not easily learnt”

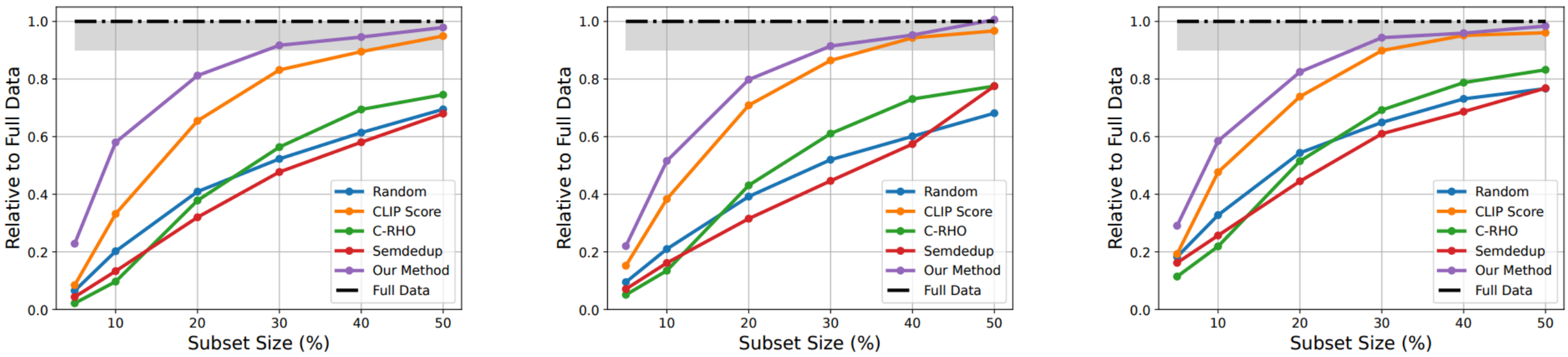
- **Method:**

  - Training Model: Partially trained model

  - Validation Model: Fully (or longer) trained model

  - Score for each data point:
    $CLIP\ Score_{val}(i, i) - CLIP\ Score_{train}(i, i)$

Coresets of Different Size from CC3M



(a) ImageNet (b) ImageNet Dist. Shift (c) Avg. over 11 Datasets

Performance across subset of different sizes selected from Conceptual Captions (CC) 3M

Data-Efficient Contrastive Language-Image Pretraining: Prioritizing Data Quality over Quantity [AISTATS'24]

Outline

- Motivation: why is data-efficiency important?
- Part 1: Data-efficient Supervised Learning
- Part 2: Data-efficient self-supervised Contrastive Pretraining
- Part 3: Foundation Models
  - 3a: Data-efficient Contrastive Language-Image Pretraining
  - 3b: Data-efficient Training of Large Language Models

Can we corsets for training Large Language Models (LLMs)?

Data-efficient Fine-tuning of LLMs

- Assume we fine-tune a pretrained model with SGD/Adam on the following dataset:

- $\mathscr{D} = \{(\mathbf{x}, \mathbf{y})\}_{i=1}^{n}, \quad \mathbf{x} = (x_1, \cdots, x_M), \quad \mathbf{y} = (y_1, \cdots, y_L)$

prompts **responses**

- $\mathscr{L}(\theta) = -\frac{1}{n} \sum_{(x, y_i) \in \mathscr{D}} [\log p_\theta(\mathbf{y} \,|\, \mathbf{x})], \quad p_\theta(y \,|\, x) = \Pi_{l=1}^{L} p_\theta(y_l \,|\, \mathbf{y}_{1:l-1}, \mathbf{x})$

Loss

- Can we find a small subset that closely captures the dynamics of training on full data?

Data-efficient Fine-tuning of LLMs

- Can we find a subset S^* that closely estimates the full gradient?

$$S^* = \arg\min_{S \subseteq V} |S|, \quad \text{s.t.} \quad \max_{w \in \mathcal{W}} \left\| \frac{1}{|V|} \sum_{i \in V} \nabla f_i(w) - \frac{1}{|S|} \sum_{j \in S} \nabla f_j(w) \right\| \leq \epsilon .$$

Full gradient

Gradient of S

- **Problem:**
  - (Even last layer) Gradients are too high-dimensional!
  - For example, dimensionality of the last V projection of Phi-2 has
    - 6.5M dimensions when training the full parameters
    - 327K dimensions when training with LoRA

Data-efficient Fine-tuning: Preliminary Results (S2L)

- **Observation:**
  - Fine-tuning changes the model to a small extent
  - Curvature is small during fine-tuning

Lemma (informal): Assuming a small curvature, loss functions with a similar trajectory (i.e. similar values at multiple points during fine-tuning) have similar gradients

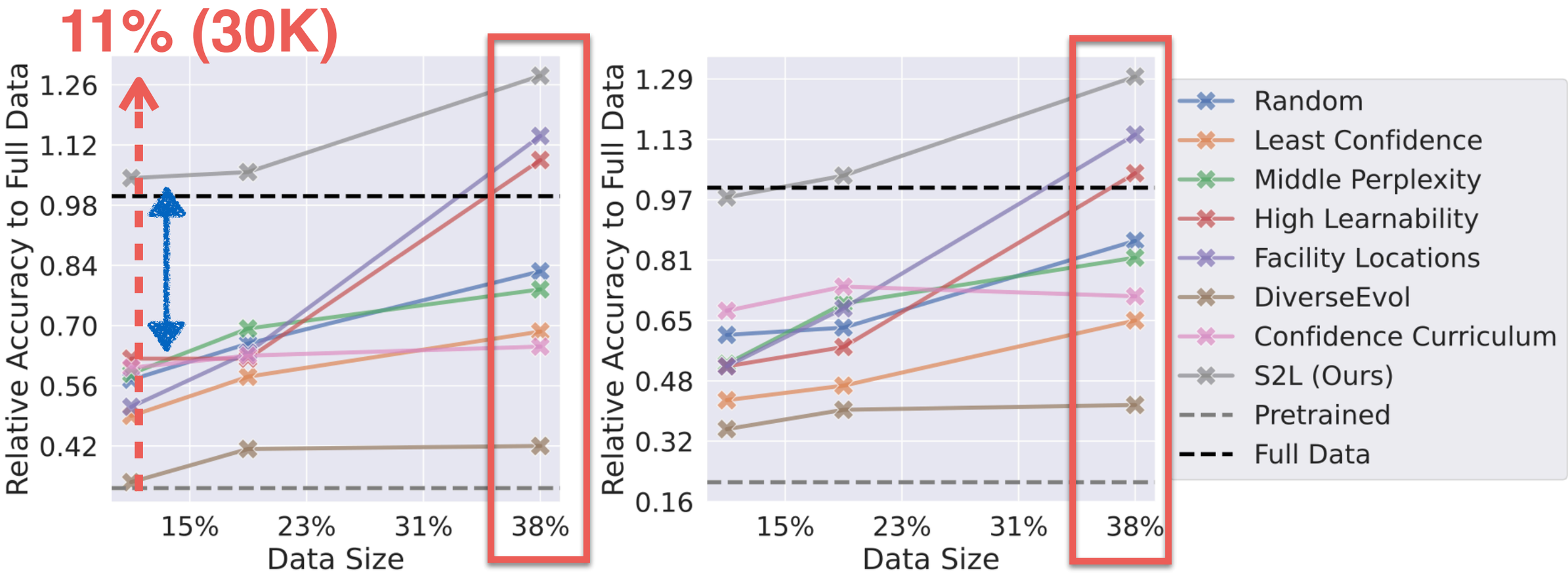
S2L: Data-efficient Fine-tuning of LLMs

- Finding a subset that captures the gradients
  - Fine-tune a small proxy model on full data and save the loss trajectories (e.g. at the beginning of every epoch)
  - Cluster the loss trajectories
  - Sample examples from every loss trajectory cluster

Corollary (informal): As long as the proxy and the target models are similar enough, fine-tuning on the subset with IG converges to a similar solution to that of training on the full data.

S2L: Fine-tuning Pythia-410M on MathInstruct

- Pythia-70M → Pythia-410M



(g) In-domain Avg

GSM8K, MATH, NUMGLUE

(h) Avg

ID: GSM8K, MATH, NUMGLUE
OOD: SVAMP, MATHEMATICS, SIMULEQ

S2L: Fine-tuning Phi-2, Phi-3 on MathInstruct

- Pythia-410M → Phi-2 (2.7B), Phi-3-Mini (3.8B)

1. Same number of iterations (mini-batches): 50K subset matches full data

Target Model	Fine-tuning Data	In-domain GSM8K	In-domain MATH	In-domain NumGLUE	In-domain Avg	Out-domain SVAMP	Out-domain Mathematics	Out-domain SimulEq	Out-domain Avg
Phi-2 (2.7B)	(Pretrained)	53.4	16.1	34.9	34.8	67.9	31.1	27.4	38.5
	Random	67.9	30.1	60.7	52.9	77.1	51.2	37.5	54.1
	High Learnability	59.4	25.2	62.1	48.9	76.6	41.8	27.2	48.7
	Middle Perplexity	66.4	29.5	54.1	50.0	74.8	50.4	39.8	52.5
	Least confidence	61.7	24.7	67.0	51.1	76.5	43.3	52.5	54.3
	Facility Locations	66.2	31.3	62.4	53.3	74.4	58.4	34.6	54.5
	S2L(Ours)	**69.1**	**32.6**	**65.7**	**55.8**	**79.6**	56.4	40.1	57.3
	Full-262K	68.3	32.6	64.3	55.1	78.4	58.4	44.2	**57.7**

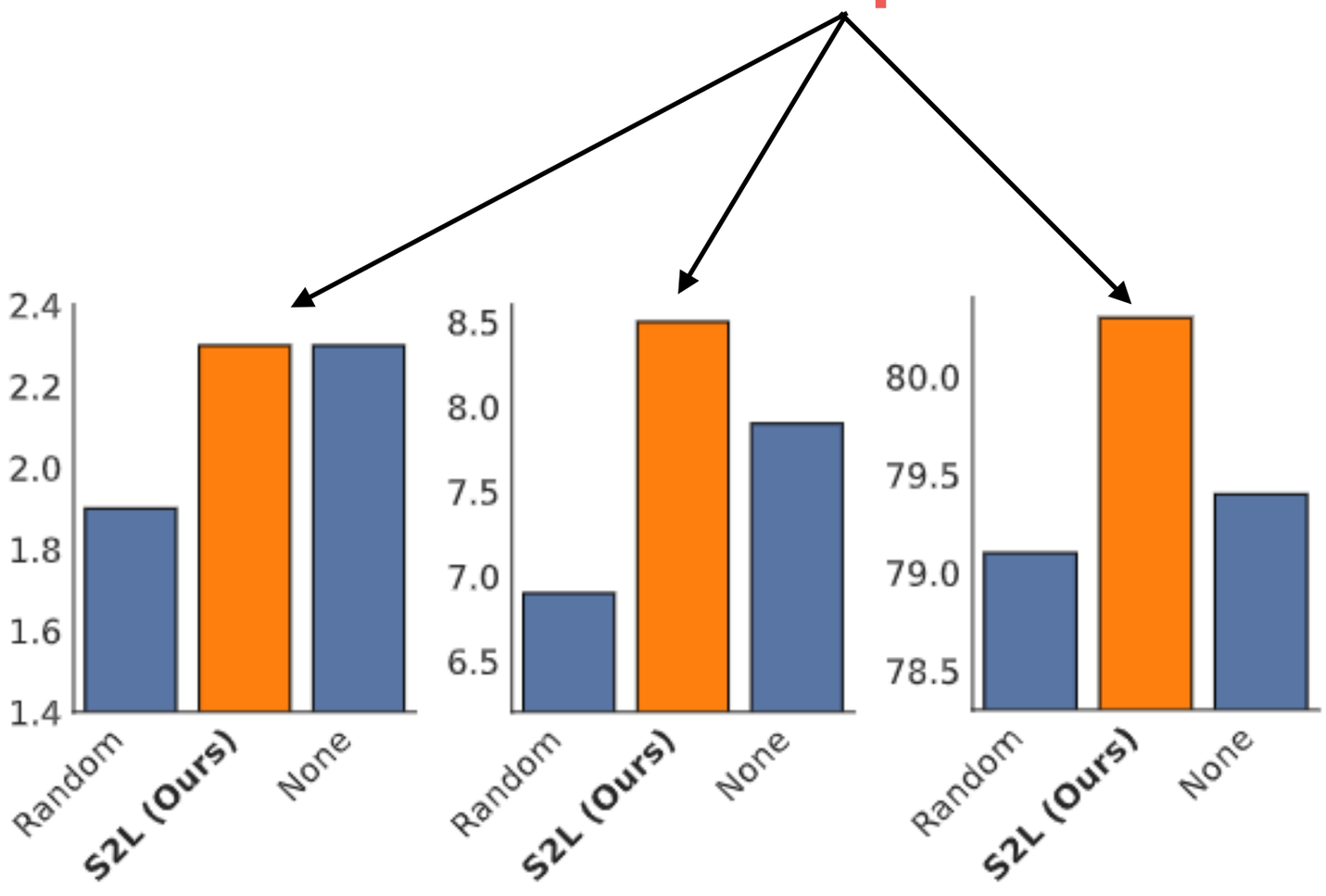
2. Smaller number of epochs: 50% subset matches full data

Target Model	Fine-tuning Data	In-domain GSM8K	In-domain MATH	In-domain NumGLUE	In-domain Avg	Out-domain SVAMP	Out-domain Mathematics	Out-domain SimulEq	Out-domain Avg
Phi-3-mini (3.8B)	(Pretrained)	74.5	26.5	52.1	51.1	83.7	44.3	34.8	52.7
	S2L-50%(Ours)	76.3	42.5	76.4	65.1	83.8	62.1	51.6	65.4
	Full	76.4	42.9	75.3	64.9	84.6	60.2	51.9	65.2

S2L: Fine-tuning Phi-2 on MIMIC-III

- Clinical text summarization on MIMIC-III
- Pythia-410M → Pythia-1B

50% subset outperforms full data!



(a) BLEU (b) ROUGE-L (c) BERTScore

Can we find small subsets that guarantee the performance when training with mini-batch SGD?

Data-efficient Training of LLMs

- Can we (iteratively) find subsets that closely estimates the gradient of a larger random subset?

$$S^* = \arg\min_{S \subseteq V_r} |S|, \quad \text{s.t.} \quad \max_{w \in \mathcal{W}} \left\| \frac{1}{|V_r|} \sum_{i \in V_r} \nabla f_i(w) - \frac{1}{|S|} \sum_{j \in S} \nabla f_j(w) \right\| \leq \epsilon .$$

Gradient of the random subset — **Gradient of S**

- **Problem:**
  - (Even last layer) Gradients are too high-dimensional!

Data-efficient Training: Preliminary results (SSM)

- Finding lower-dimensional gradient estimates
- (1) Use the V-Projection matrix at the last layer
  - Stacks with LoRA and other memory-efficient methods
- Can be calculated with only one forward pass, using zeroth-order gradient calculation

$$\boldsymbol{g_z}(\boldsymbol{\theta}) = \frac{\mathcal{L}(\boldsymbol{\theta} + \epsilon \boldsymbol{z}) - \mathcal{L}(\boldsymbol{\theta} - \epsilon \boldsymbol{z})}{2\epsilon} \boldsymbol{z} \approx \boldsymbol{z}\boldsymbol{z}^\top g(\boldsymbol{\theta}). \qquad \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\mathcal{I}_d})$$

$$\boldsymbol{g_{\hat{z}}}(\boldsymbol{\theta}) = \frac{\mathcal{L}(\boldsymbol{\theta} + \epsilon \hat{\boldsymbol{z}}; \mathcal{B}) - \mathcal{L}(\boldsymbol{\theta} - \epsilon \hat{\boldsymbol{z}}; \mathcal{B})}{2\epsilon} = \frac{\mathcal{L}(\boldsymbol{\theta} + \epsilon \boldsymbol{m} \odot \boldsymbol{z}; \mathcal{B}) - \mathcal{L}(\boldsymbol{\theta} - \epsilon \boldsymbol{m} \odot \boldsymbol{z}; \mathcal{B})}{2\epsilon}, \quad \hat{\boldsymbol{z}} = \boldsymbol{m} \odot \boldsymbol{z}. \quad \boldsymbol{m} \in \{0,1\}^d$$

- (2) Sparsify the above gradient estimate
  - Consider the dimensions with largest gradient magnitude

Preliminary results (SSM): Finding a Curriculum

- Select Small Mini-batches (SSM)

  1. Finding lower-dimensional gradient estimates

     - Use the V-Projection matrix at the last layer

     - Sparsify the above gradient estimate

  2. Find mini-batch subsets from larger random subsets

Step	Method	In-domain				Out-domain				Avg
		GSM8K	MATH	NumGLUE	Avg	SVAMP	Math.	SimulEq	Avg	
Pretrained		52.9	16.4	35.0	34.8	67.9	31.9	28.8	42.9	
1K	bs=64	$66.5_{\pm 0.8}$	$28.4_{\pm 0.3}$	$50.2_{\pm 0.9}$	$48.3_{\pm 0.2}$	$79.2_{\pm 0.4}$	$52.4_{\pm 0.8}$	$24.1_{\pm 1.5}$	$51.9_{\pm 0.2}$	$50.1_{\pm 0.2}$
	bs=128	$67.4_{\pm 0.5}$	$28.8_{\pm 0.3}$	$53.2_{\pm 1.2}$	$49.8_{\pm 0.5}$	$80.4_{\pm 1.3}$	$55.6_{\pm 0.4}$	$29.9_{\pm 2.4}$	$55.3_{\pm 1.0}$	$52.6_{\pm 0.6}$
	SSM	$67.8_{\pm 0.5}$	$30.0_{\pm 0.4}$	$53.5_{\pm 0.8}$	$50.4_{\pm 0.2}$	$80.2_{\pm 0.4}$	$58.0_{\pm 1.8}$	$35.9_{\pm 2.6}$	$58.0_{\pm 1.6}$	$54.2_{\pm 0.8}$
2K	bs=64	$67.9_{\pm 0.4}$	$29.3_{\pm 0.7}$	$57.6_{\pm 0.6}$	$51.6_{\pm 0.2}$	$80.1_{\pm 0.4}$	$57.9_{\pm 0.9}$	$42.2_{\pm 1.4}$	$60.1_{\pm 0.5}$	$55.9_{\pm 0.2}$
	bs=128	$67.7_{\pm 0.8}$	$30.3_{\pm 0.4}$	$58.4_{\pm 0.8}$	$52.1_{\pm 0.3}$	$79.5_{\pm 0.4}$	$57.9_{\pm 0.5}$	$45.5_{\pm 0.7}$	$60.9_{\pm 0.4}$	$56.5_{\pm 0.3}$
	SSM	$68.1_{\pm .2}$	$30.8_{\pm 0.2}$	$58.2_{\pm 1.1}$	$52.4_{\pm 0.2}$	$80.1_{\pm 0.7}$	$58.7_{\pm 0.8}$	$46.4_{\pm 0.7}$	$61.8_{\pm 0.3}$	$57.1_{\pm 0.2}$

[Memory efficient Fine-tuning of LLMs, To be on ArXiv soon]

There are Several Heuristics…

- **Pretraining**
  - Perplexity, Error L2-Norm (EL2N), and memorization ranking => middle perplexity works best [Marion et al, NeurIPS'23]
  - Deduplication [Tirumala et al, NeurIPS'23]
- **Fine-tuning**
  - Manual curation [Zhou et al, NeurIPS'23], Select data via other LLMs like GPT or Chat GPT [Eldan & Li, 2023;Li et al., 2023a; Chen et al., 2024]
  - Selecting centroid of hidden states [Bhatt et al, 2024]
- Assuming access to **target tasks** (not comparable to above methods):
  - Using influence functions [Xia et al, ICML'2024], datamodels [Engstrom et al, ICML'2024], Finding similar examples to target [Xie et al., 2023, Brown et al, 2020]

Further Directions

1. **Modifying the training data distribution** can improve the in-distribution performance

   - **Idea**: reducing the simplicity bias early in training allows finding a more generalizable solution

   - Check out this **preliminary result**: arXiv preprint arXiv:2404.17768

2. **Directly generating high-quality data** (not to confuse with dataset distillation)

   - Supervised-learning, self-supervised learning, Multimodal models, Generative models

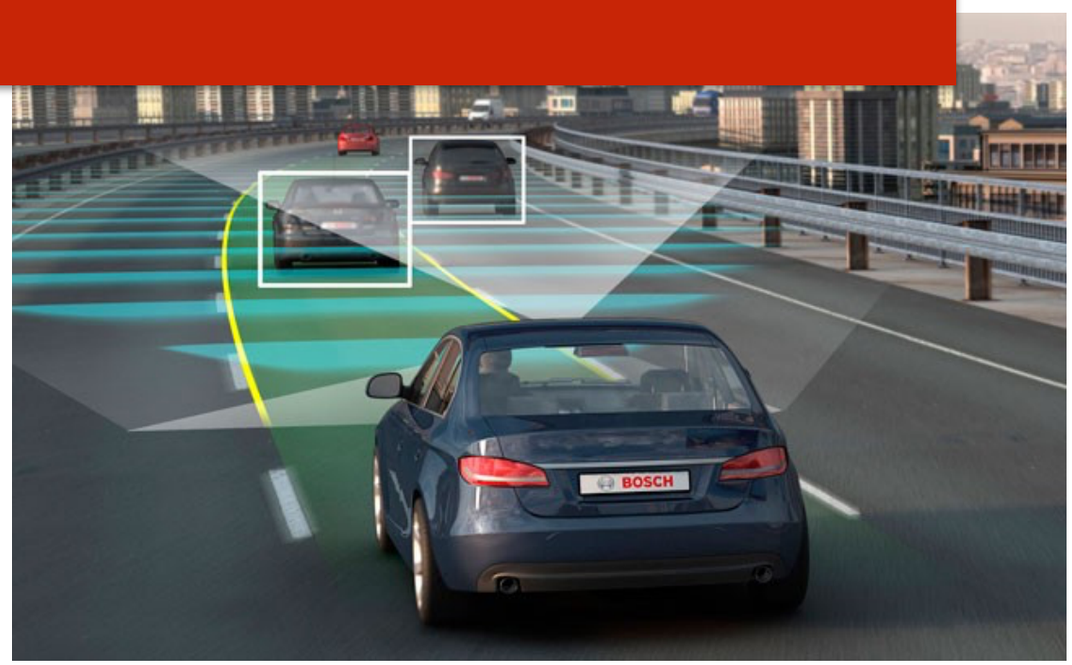
3. Finding subsets for **other optimizers like Adam**

4. **Improve existing solutions**

Personalized medicine

Robotics

Less Data Can be More!



Thank You!

References

- Part 1: Data-efficient Supervised Learning
  - Coresets for Data-efficient Training of Machine Learning Models, Mirzasoleiman et al., ICML 2020
  - Towards Sustainable Learning: Coresets for Data-efficient Deep Learning, Yang et al., ICML 2023
  - Adaptive second order coresets for data-efficient machine learning, Pooladzandi et al, ICML 2022
  - An Empirical Study of Example Forgetting during Deep Neural Network Learning, Toneva et al., ICLR 2019
  - Deep Learning on a Data Diet: Finding Important Examples Early in Training, Paul et al, NeurIPS 2021
  - Dataset Cartography: Mapping and Diagnosing Datasets with Training Dynamics, Swayamdipta et al, ACL 2020

References

- Part 2: Data-efficient self-supervised Contrastive Pretraining
  - Data-Efficient Contrastive Self-supervised Learning: Most Beneficial Examples for Supervised Learning Contribute the Least, Joshi and Mirzasoleiman, ICML 2023.
  - Towards the Generalization of Contrastive Self-Supervised Learning, Huang et al., ICLR 2023.
  - A Simple Framework for Contrastive Learning of Visual Representations, Chen et al., ICML 2020

References

- Part 3a: Data-efficient Contrastive Language-Image Pre-training
  - Data-Efficient Contrastive Language-Image Pre-training: Prioritizing Data Quality over Quantity, Joshi et al, AISTATS 2024.
  - Learning Transferable Visual Models From Natural Language Supervision, Radford et al., ICML 2021.
  - SemDeDup: Data-efficient learning at web-scale through semantic deduplication, Abbas et al., preprint 2023.
  - T-MARS: Improving Visual Representations by Circumventing Text Feature Learning, Maini et al, ICLR 2024.
  - Understanding multimodal contrastive learning and incorporating unpaired data, Nakada et al., AISTATS 2023.

References

- Part 3b: Data-efficient Training of Large Language Models
  - SmallToLarge (S2L): Scalable Data Selection for Fine-tuning Large Language Models by Summarizing Training Trajectories of Small Models, Yang et al., preprint 2024.
  - Memory Efficient Fine-tuning of Large Language Models, preprint 2024.
  - Marion et al, When less is more: Investigating data pruning for pretraining llms at scale. preprint 2023.
  - D4: Improving LLM pretraining via document de-duplication and diversification, Tirumala et al, NeurIPS 2023.
  - Solving quantitative reasoning problems with language models, Lewkowycz et al, NeurIPS 2022.
  - Biderman et al, Emergent and predictable memorization in large language models. *preprint* 2023.
  - Zhou et al, LIMA: Less is more for alignment. NeurIPS, 2023.
  - Bhatt et al, An experimental design framework for label-efficient supervised finetuning of large language model, preprint 2024.
  - Chen et al. Alpagasus: Training a better alpaca with fewer data, ICLR 2024.