Towards Sustainable Learning: Coresets for Data-efficient Deep Learning

Yu Yang (UCLA), Hao Kang (Ga. Tech), Baharan Mirzasoleiman (UCLA)

Eeshaan Jain

Given at QuML, Aalto University

Efficient Learning

- Training models on large datasets requires expensive computational resources, and has wasteful consequences on the environment
- One way to solve this is to train on a reliable subset of the dataset, since all instances might not be of equal importance or may not contribute to generalization
- Recent papers have shown that for strongly convex models, a weighted subset of data (*coreset*) – that matches the full gradient – has convergence guarantees on gradient descent
- However, this isn't the case for non-convex models

Non-convex Models

- Unlike strongly convex models, the training dynamics of non-convex models cannot be bounded beforehand
- Since non-convex models are learned with stochastic gradients, they require unbiased estimates of the full gradient
- Finally, iteratively selecting coresets from full data is an expensive task and significantly limits the training-speedup

The paper has the following contributions:

1. Modeling of non-convex loss for coreset selection
2. Proposal of a selection method for (mini-batch) SGD
3. Improving efficiency of coreset selection

Background (1)

- The model parameters are obtained through ERM:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathcal{W}} \mathbb{E}_{(\mathbf{x}_i,y_i)\sim\mathcal{D}}\left[\mathcal{L}(\mathbf{w};(\mathbf{x}_i,y_i))\right]$$

- Since for over-parameterized models, GD becomes extremely slow, stochastic methods are used which select one or more mini-batches of size m ($\mathcal{M}$: i.i.d.), and performs:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\frac{1}{m}\sum_{i\in\mathcal{M}}\mathbf{g}_{t,i}$$

  where $\mathbf{g}_{t,i}$ is the gradient of example i at time t.
- Existing coreset methods solve

$$S^* = \arg\min_{S\subseteq V,\gamma_j\geqslant 0\forall j\in S} |S| \text{ s.t. } \max_{\mathbf{w}_t\in\mathcal{W}} \left\|\sum_{i\in V}\mathbf{g}_{t,i} - \sum_{j\in V}\gamma_j\mathbf{g}_{t,j}\right\| \leqslant \epsilon$$

- The normed gradient distance between data points can be upper-bounded with the feature vector difference turning the problem into the submodular cover problem:

Background (2)

$$S^* = \arg\min |S| \text{ s.t. } C - \sum_{i \in V} \min_{j \in S} \|x_i - x_j\| \geqslant C - \epsilon$$

A set function $f : 2^V \to \mathbb{R}_+$ is said to be submodular if for $S, T \subseteq V$ where $S \subseteq T$ and $v \in V \setminus T$, if $f(S \cup v) - f(S) \geqslant f(T \cup v) - f(T)$

This can be solved using the greedy algorithm in $\mathcal{O}(n \cdot k)$

- For neural networks, finding gradients is slow and doesn't yield high quality subsets. Usually just the last layer gradients are used –

$$S_t^* = \arg\max_{S \subseteq V} C - \sum_{i \in V} \min_{j \in S} \left\| g_{t,i}^L - g_{t,j}^L \right\| \text{ s.t. } |S| \leqslant k \qquad \text{(SCS)}$$

- It isn't clear when the coresets are to be updated for training non-convex models, and finding coresets from full data has convergence guarantees for (Incremental) GD, and not stochastic

(C1) For deep networks, the value of $\mathcal{L}(\cdot)$ changes very rapidly for different datapoints, i.e., $\nabla\mathcal{L}_i(\mathbf{w}_t)$ might be drastically different than $\nabla\mathcal{L}_i(\mathbf{w}_t + \delta)$ for $\mathbf{w}_t + \delta \in \text{Neighborhood}(\mathbf{w}_t)$

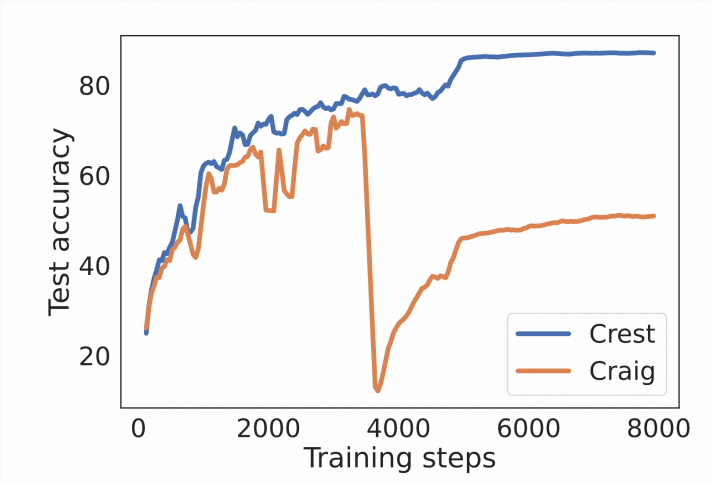

Figure 1: CRAIG Performance

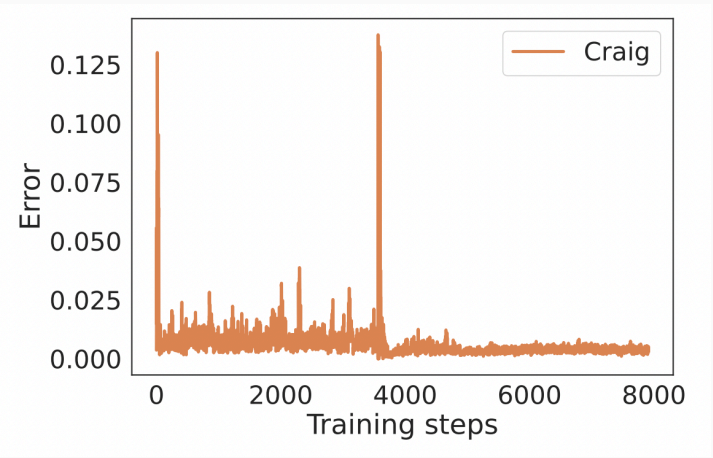

Figure 2: CRAIG Gradient Error

Crest: Challenge 2 & 3

(**C2**) There are no convergence guarantees when using stochastic gradient methods. Moreover, some examples may get high weight which makes the mini-batch gradient variance larger than random

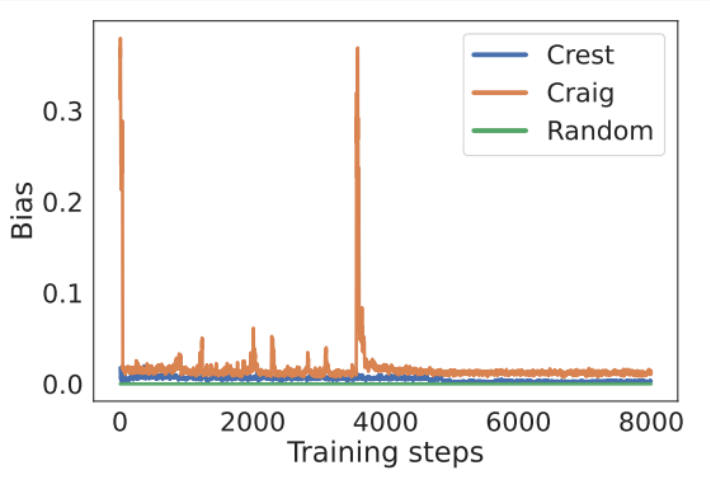

Figure 3: Bias of mini-batch gradients

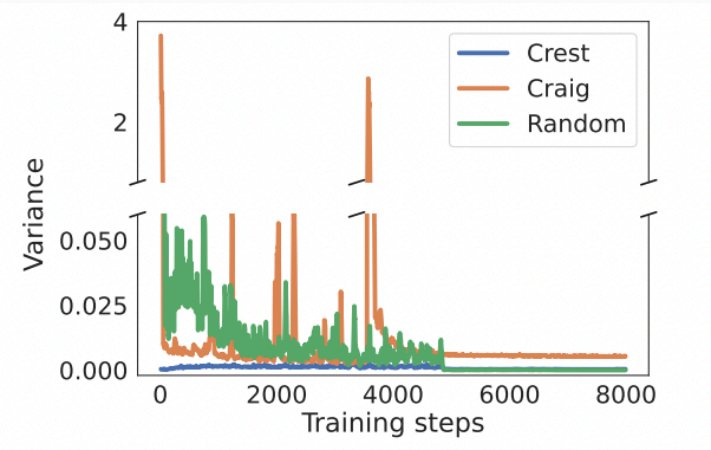

Figure 4: Var. of mini-batch gradients

(**C3**) The $\mathcal{O}(n \cdot k)$ complexity during coreset selection prevents speedup for large datasets when selecting iteratively

Overcoming C1

- Overview:
  - Model the non-convex loss as a piece-wise quadratic function, thus the problem is broken down to finding coresets for a series of quadratic problems
  - A coreset S_ℓ is found at every selection step ℓ, capturing $\mathbf{w}_{t_\ell}$, which is followed by a quadratic loss approximation $\mathcal{F}^\ell$ based on the gradient and curvature
  - The coreset S_ℓ is in the δ-neighborhood, i.e., $\mathcal{L}(\mathbf{w}_{t_\ell} + \delta) = \mathcal{F}^\ell(\delta)$ to ensure convergence and small gradient error within $\mathcal{N}_\ell$
  - To find S_ℓ, (SCS) is solved
- Methodology:
  - A 2nd-order Taylor series expansion of $\mathcal{L}(\mathbf{w}_{t_\ell})$ around $\mathcal{N}_\ell$ gives

$$\mathcal{F}^\ell(\delta) = \frac{1}{2}\delta^\top \mathbf{H}_{t_\ell, S_\ell}\delta + \mathbf{g}_{t_\ell, S_\ell} + \mathcal{L}(\mathbf{w}_{t_\ell})\delta$$

    Here $\mathbf{H}_{t_\ell, S_\ell}$ and $\mathbf{g}_{t_\ell, S_\ell}$ are the weighted mean of the Hessian and gradient of instances in S_ℓ.

Overcoming C1

- Hutchinsons trace estimator method is used to obtain an efficient and stochastic estimate of the coreset Hessian diagonal with $\mathbf{z} \sim$ Rademacher:

$$\mathbf{H}_{t_\ell, S_\ell} \mathbf{z} = \frac{\partial \mathbf{g}_{t_\ell, S_\ell} \mathbf{z}}{\partial \mathbf{w}_{t_\ell}}$$

$$\text{diag}(\mathbf{H}_{t_\ell, S_\ell}) = \mathbb{E}[\mathbf{z} \odot (\mathbf{H}_{t_\ell, S_\ell} \mathbf{z})]$$

- The gradient and diagonal-Hessian are further smoothened using exponential averaging since they can be noisy
- To enhance efficiency, every T_1 iterations, the relative error relative to in $\mathcal{F}_\ell(\delta)$, ρ_{t_ℓ} is computed (after getting an unbiased estimate of $\mathcal{L}(\cdot)$ using random samples). If the error $< \tau$, then the coreset isn't changed else a new coreset is selected

Overcoming C2

- Overview:
  - Sample multiple subsets $\{V_1, \ldots, V_P\}$ uniformly at random and directly select a smaller coreset S_ℓ^p of size m from each V_p
- Methodology:
  - P smaller submodular problems are solved (one for each V_p) and $S_\ell = \bigcup_{p \in [P]} S_\ell^p$
  - Small error of mini-batch gradients cancel each other out
  - For a fixed mini-batch size, selecting mini-batch coresets from larger random subsets results in a smaller variance but may introduce a larger bias

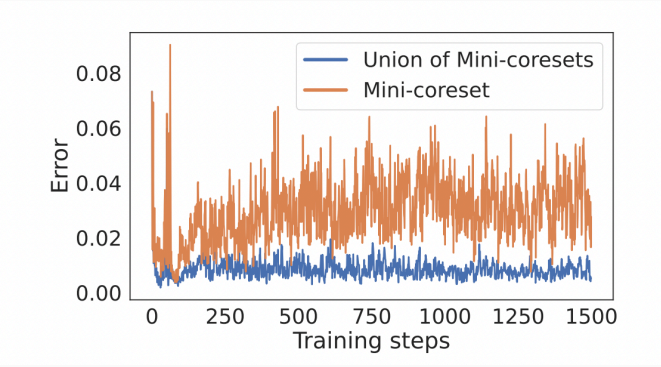

Figure 5: Gradient error on union

Overcoming C3

- Overview:
  - When examples are learned, their gradient and loss almost become zero, and they don't contribute to training
  - Every T_2 iterations, the loss of examples are checked, and those consistently with a value less than α are dropped
  - Dropping s examples increases full average gradient by $\frac{n}{n-s}$, which is equivalent to increase learning rate
  - If mini-batch coresets closely capture gradient of random subsets V_p, CREST with a small enough τ , converges to a ν-stationary point of the nonconvex loss, but r/m times faster than mini-batch SGD with mini-batch size m on full data

Algorithm

Algorithm 1 CoREsets for STochastic GD (CREST)

Require: Model parameter $\boldsymbol{w}_0$, mini-batch size m, random partition size r, learning rate η, total training iterations N, checking interval T_2, multipliers b, h, thresholds α, τ.

$t \leftarrow 0, T_1 \leftarrow 1$, update $\leftarrow 1$

while $t < N$ **do**

  if update $== 1$ **then**

    for $p = 1$ to P **do**

      Select a random subset $V_p \subseteq V$ s.t. $|V_p| = r$

      $S_l^p \in \arg\max_{\substack{S \subseteq V_p \\ |S| \leq m}} C - \sum_{i \in V_p} \min_{j \in S} \|\mathbf{g}_{t_l,i}^L - \mathbf{g}_{t_l,j}^L\|$

    $S_l = \bigcup_{p \in [P]} S_l^p$

    Calculate $\mathcal{F}^l$ with $\overline{\mathbf{H}}_{t,S_l}, \overline{\mathbf{g}}_{t,S_l}$

  for $j = 1$ to T_1 **do**

    $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta \mathbf{g}_{S_l,t}$

    $t \leftarrow t + 1$

    if $t \mod T_2 == 0$ **then**

      $V = \{j \in V | \mathcal{L}_j(\boldsymbol{w}_i) > \alpha, \forall i \in [t - T_2, t]\}$.

  $\boldsymbol{\delta} \leftarrow \boldsymbol{w}_t - \boldsymbol{w}_{t-T_1}$

  Calculate ρ_t from Equation (10).

  if $\rho_t > \tau$ **then**

    update $\leftarrow 1$,

    $T_1 \leftarrow h \times \|\overline{\mathbf{H}}_0\| / \|\overline{\mathbf{H}}_t\|$, $P \leftarrow b \times T_1$

  else

    update $\leftarrow 0$

Theorem

For any $\delta, \lambda > 0$, assume that $\mathcal{L}$ is L-gradient Lipschitz and the stochastic gradients $\mathbf{g}_{t,i}$ have a bounded variance: $\mathbb{E}_{i \in V}[\|\mathbf{g}_{t,i} - \nabla \mathcal{L}(\mathbf{w}_t)\|] \leqslant \sigma^2$.

***Case 1* (CREST: *Nearly-unbiased*).** *Let step size be* $\eta = \min\{\frac{1}{L}, \frac{\tilde{D}\sqrt{r}}{\sigma\sqrt{N}}\}$, *for some* $\tilde{D} > 0$ *and* N *be the number of training iterations. If the gradient bias of mini-batch coresets* $\mathbb{E}[\|\boldsymbol{\xi}_{t_l}\|] \leq \epsilon\|\nabla \mathcal{L}(\boldsymbol{w}_{t_l})\|$ *and* $\tau \leq \min_l(\|\nabla \mathcal{L}(\boldsymbol{w}_{t_l} + \boldsymbol{\delta}_l)\| - 3\epsilon\|\nabla \mathcal{L}(\boldsymbol{w}_{t_l})\|)\|\boldsymbol{\delta}_l\|/2\mathcal{L}(\boldsymbol{w}_{t_l} + \boldsymbol{\delta}_l)$, *for* $0 \leq \epsilon \leq \min\{1, \|\nabla \mathcal{L}(\boldsymbol{w}_{t_l} + \boldsymbol{\delta}_l)\|/3\|\nabla \mathcal{L}(\boldsymbol{w}_{t_l})\|\}$, *then with probability at least* $1 - \lambda$, CREST *will visit a* ν*-stationary point at least once in the following number of iterations:*

$$\tilde{\mathcal{O}}\left(\frac{L(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*)}{\nu^2}(1 + \frac{\sigma^2}{r\nu^2})\right). \tag{12}$$

Case 2 (Biased). *If the bias of mini-batches* $\mathbb{E}[\|\boldsymbol{\xi}_t\|] \leq \epsilon$, *but* ϵ *is larger than the full gradient norm anytime during the training, then the number of iterations is:*

$$\tilde{\mathcal{O}}\left(\frac{L(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}^*)}{\nu^2 - \epsilon}(1 + \frac{\sigma^2 + r\epsilon^2}{r(\nu^2 - \epsilon)})\right). \tag{13}$$

In particular, if $\epsilon \geq \nu^2$, *convergence is not guaranteed.*

Experiments (1)

Dataset - Model	Backprop	SGD†	Random	Craig	Grad-match‡	Glister*	**Crest** (Ours)
CIFAR-10 - ResNet-20	10%	$21.3_{\pm 8.0}$	$7.2_{\pm 1.4}$	$13.0_{\pm 5.1}$	$6.0_{\pm 0.1}$	$7.0_{\pm 0.1}$	$\mathbf{5.5}_{\pm \mathbf{0.2}}$
CIFAR-100 - ResNet-18	10%	$36.5_{\pm 2.9}$	$11.7_{\pm 0.4}$	$17.2_{\pm 4.5}$	$12.7_{\pm 0.9}$	$27.6_{\pm 4.0}$	$\mathbf{9.4}_{\pm \mathbf{0.3}}$
TinyImageNet - ResNet-50	10%	$32.8_{\pm 2.1}$	$16.0_{\pm 0.5}$	$28.5_{\pm 0.6}$	$27.7_{\pm 0.2}$	$32.8_{\pm 2.1}$	$\mathbf{15.4}_{\pm \mathbf{0.6}}$
SNLI - RoBERTa (Finetune)	10%	$1.2_{\pm 0.3}$	$1.2_{\pm 0.3}$	-	-	-	$\mathbf{0.8}_{\pm \mathbf{0.2}}$

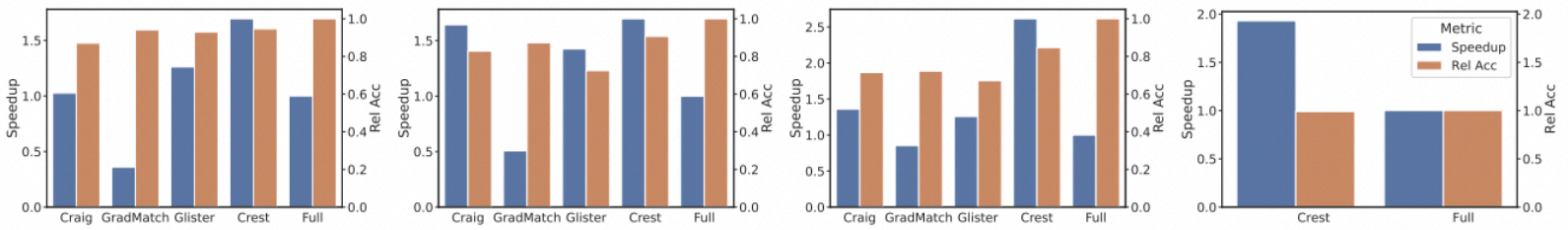

Figure 6: Normalized run-time and test accuracy

Experiments (2)

Table 2: Average time for different components of CREST for training ResNet-18 on CIFAR-100 with batch size 128.

STEP	TIME (SECONDS)
SELECTION (CREST)	0.006
SELECTION (CRAIG)	0.089
LOSS APPROXIMATION	0.115
CHECKING THRESHOLD	0.796

Figure 7: CREST running time

Experiments (3)

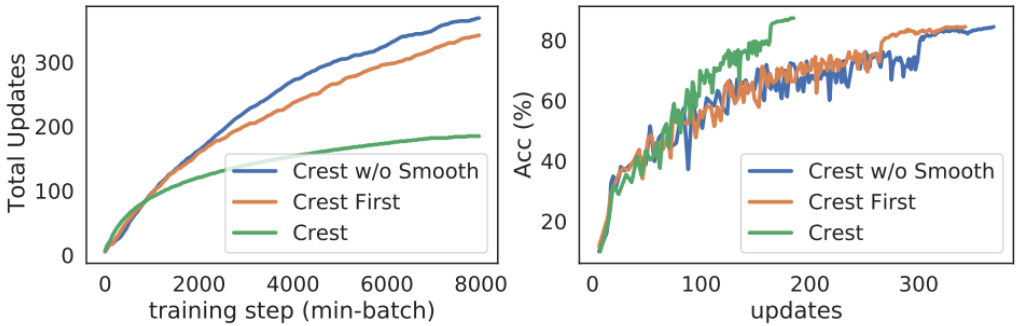

Figure 4: Training ResNet-20 on CIFAR-10 with CREST under 10% training budget. (Left) Number of coreset updates vs. training iterations. (Right) test accuracy vs. the total number of coreset updates.

Figure 8: Test error and update comparison

Experiments (4)

Forgetting score counts the number of times examples are misclassified after being correctly classified during the training, and quantifies the difficulty of learning an example

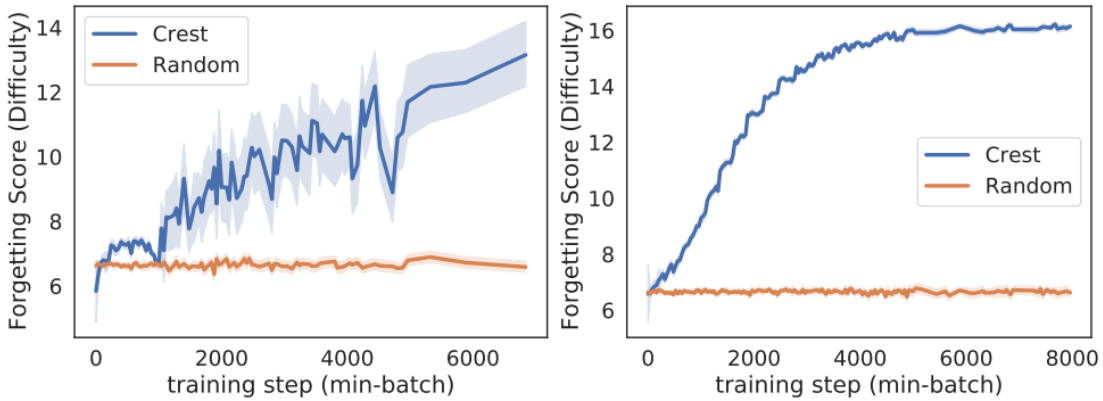

Figure 5: Average forgettability score of CREST coresets during training, when learned examples are not discarded (Left), and are discarded (Right). Learning difficulty of examples selected by CREST increases during the training.

Figure 9: Forgetting score analysis