GRAD-MATCH: Gradient Matching based Data Subset Selection for Efficient Deep Model Training

Krishnateja Killamsetty [1] **Durga Sivasubramanian** [2] **Ganesh Ramakrishnan** [2] **Abir De** [2] **Rishabh Iyer** [1]

Abstract

The great success of modern machine learning models on large datasets is contingent on extensive computational resources with high financial and environmental costs. One way to address this is by extracting subsets that generalize on par with the full data. In this work, we propose a general framework, GRAD-MATCH, which finds subsets that closely match the gradient of the *training or validation* set. We find such subsets effectively using an orthogonal matching pursuit algorithm. We show rigorous theoretical and convergence guarantees of the proposed algorithm and, through our extensive experiments on real-world datasets, show the effectiveness of our proposed framework. We show that GRAD-MATCH significantly and consistently outperforms several recent data-selection algorithms and achieves the best accuracy-efficiency trade-off. GRAD-MATCH is available as a part of the CORDS toolkit: https://github.com/decile-team/cords.

1. Introduction

Modern machine learning systems, especially deep learning frameworks, have become very computationally expensive and data-hungry. Massive training datasets have significantly increased end-to-end training times, computational and resource costs (Sharir et al., 2020), energy requirements (Strubell et al., 2019), and carbon footprint (Schwartz et al., 2019). Moreover, most machine learning models require extensive hyper-parameter tuning, further increasing the cost and time, especially on massive datasets. In this paper, we study efficient machine learning through the paradigm of subset selection, which seeks to answer the following question: *Can we train a machine learning model on much smaller subsets of a large dataset, with negligible loss in test accuracy?*

Data subset selection enables efficient learning at multiple levels. First, by using a subset of a large dataset, we can enable learning on relatively low resource computational environments without requiring a large number of GPU and CPU servers. Second, since we are training on a subset of the training dataset, we can significantly improve the end-to-end turnaround time, which often requires multiple training runs for hyper-parameter tuning. Finally, this also enables significant reduction in the energy consumption and CO2 emissions of deep learning (Strubell et al., 2019), particularly since a large number of deep learning experiments need to be run in practice. Recently, there have been several efforts to make machine learning models more efficient via data subset selection (Wei et al., 2014a; Kaushal et al., 2019; Coleman et al., 2020; Har-Peled & Mazumdar, 2004; Clarkson, 2010; Mirzasoleiman et al., 2020a; Killamsetty et al., 2021). Existing approaches either use proxy functions to select data points, or are specific to particular machine learning models, or use approximations of quantities such as gradient error or generalization errors. In this work, we propose a data selection framework called GRAD-MATCH, which exactly minimizes a residual error term obtained by analyzing adaptive data subset selection algorithms, therefore admitting theoretical guarantees on convergence.

1.1. Contributions of this work

Analyses of convergence bounds of adaptive data subset selection algorithms. A growing number of recent approaches (Mirzasoleiman et al., 2020a; Killamsetty et al., 2021) can be cast within the framework of *adaptive data subset selection*, where the data subset selection algorithm (which selects the data subset depending on specific criteria) is applied in conjunction with the model training. As the model training proceeds, the subset on which the model is being trained is improved via the current model's snapshots. We analyze the convergence of this general framework and show that the convergence bound *critically* depends on an additive error term depending on how well the subset's weighted gradients match either the full training gradients or the full validation gradients (*c.f.*, Section 2).

Data selection framework with convergence guarantees. Inspired by the result above, we present GRAD-MATCH,

[1]Department of Computer Science, University of Texas at Dallas, Dallas, USA [2]Department of Computer Science and Engineering, Indian Institute of Technology, Bombay, India. Correspondence to: Krishnateja Killamsetty <krishnateja.killamsetty@utdallas.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

a gradient-matching algorithm (*c.f.*, Section 3), which directly tries to minimize the *gradient matching error*. As a result, we are able to show convergence bounds for a large class of convex loss functions. We also argue that the resulting bounds we obtain are tighter than those of recent works (Mirzasoleiman et al., 2020a; Killamsetty et al., 2021), which either use upper bounds or their approximations. We then show that minimizing the gradient error can be cast as a weakly submodular maximization problem and propose an orthogonal matching pursuit based greedy algorithm. We then propose several implementation tricks (*c.f.*, Section 4), which provide significant speedups for the data selection step.

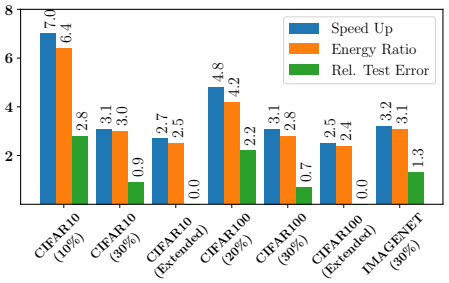

Figure 1. Efficiency of GRAD-MATCH relative to full training on the CIFAR-10, CIFAR-100, and Imagenet datasets.

Best tradeoff between efficiency and accuracy. In our empirical results (*c.f.*, Section 5), *we show that* GRAD-MATCH *achieves much a better accuracy and training-time trade-off than several state-of-the-art approaches such as* CRAIG *(Mirzasoleiman et al., 2020a),* GLISTER *(Killamsetty et al., 2021), random subsets, and even full training with early stopping.* When training with ResNet-18 on Imagenet, CIFAR-10, and CIFAR-100, we observe around 3× efficiency improvement with an accuracy drop of close to 1% for 30% subset. With smaller subsets (*e.g.*, 20% and 10%), we sacrifice a little more in terms of accuracy (*e.g.*, 2% and 2.8%) for a larger speedup in training (4× and 7×). Furthermore, we see that by extending the training beyond the specified number of epochs (300 in this case), we can match the accuracy on the full dataset using just 30% of the data while being overall 2.5× faster. In the case of MNIST, the speedup improvements are even more drastic, ranging from 27× to 12× with 0.35% to 0.05% accuracy degradation.

1.2. Related work

A number of recent papers have used submodular functions as *proxy* functions (Wei et al., 2014a;c; Kirchhoff & Bilmes, 2014; Kaushal et al., 2019) (to actual loss). Let n be the number of data points in the ground set. Then a set function $f : 2^{[n]} \rightarrow \mathbf{R}$ is submodular (Fujishige, 2005) if it satisfies the diminishing returns property: for subsets $S, T \subseteq [n], f(j|S) \triangleq f(S \cup j) - f(S) \geq f(j|T)$. Another commonly used approach is that of coresets. Coresets are weighted subsets of the data, which approximate certain desirable characteristics of the full data (, *e.g.*, the loss function) (Feldman, 2020). Coreset algorithms have been used for several problems including k-means and k-median clustering (Har-Peled & Mazumdar, 2004), SVMs (Clarkson, 2010) and Bayesian inference (Campbell & Broderick, 2018). Coreset algorithms require algorithms that are often specialized and very specific to the model and problem at hand and have had limited success in deep learning.

A very recent coreset algorithm called CRAIG (Mirzasoleiman et al., 2020a) has shown promise for both deep learning and classical machine learning models such as logistic regression. Unlike other coreset techniques that largely focus on approximating loss functions, CRAIG selects representative subsets of the training data that closely approximate the full gradient. Another approach poses the data selection as a discrete bi-level optimization problem and shows that, for several choices of loss functions and models (Killamsetty et al., 2021; Wei et al., 2015), the resulting optimization problem is submodular. A few recent papers have studied data selection approaches for robust learning. Mirzasoleiman et al. (2020b) extend CRAIG to handle noise in the data, whereas Killamsetty et al. (2021) study class imbalance and noisy data settings by assuming access to a clean validation set. In this paper, we study data selection under class imbalance in a setting similar to (Killamsetty et al., 2021), where we assume access to a clean validation set. Our work is also related to highly distributed deep learning systems (Jia et al., 2018) which make deep learning significantly faster using a cluster of hundreds of GPUs. In this work, we instead focus on *single GPU* training runs, which are more practical for smaller companies and academic labs and potentially complementary to (Jia et al., 2018). Finally, our work is also complementary to that of (Wang et al., 2019), where the authors employ tricks such as selective layer updates, low-precision backpropagation, and random subsampling to achieve significant energy reductions. In this work, we demonstrate both energy and time savings *solely* based on a more principled subset selection approach.

2. GRAD-MATCH through the lens of adaptive data subset selection

In this section, we will study the convergence of general adaptive subset selection algorithms and use the result (Theorem 1) to motivate GRAD-MATCH.

Notation. Denote $\mathcal{U} = \{(x_i, y_i)\}_{i=1}^{N}$, as the set of training examples, and let $\mathcal{V} = \{(x_j, y_j)\}_{j=1}^{M}$ denote the validation set. Let θ be the classifier model parameters. Next, denote by $L_T^i(\theta) = L_T(x_i, y_i, \theta)$, the training loss at the i^{th} epoch of training, and let $L_T(\mathcal{X}, \theta) = \sum_{i \in \mathcal{X}} L_T(x_i, y_i, \theta)$ be the loss on a subset $\mathcal{X} \subseteq \mathcal{U}$ of the training examples. Let the

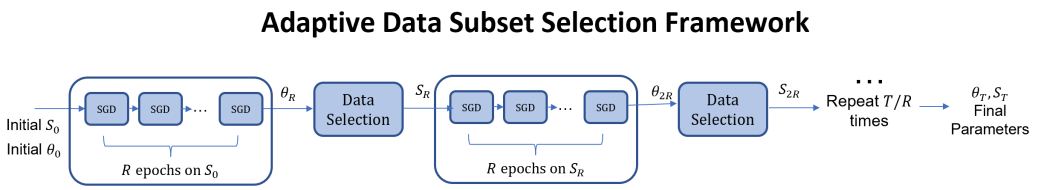

Figure 2. Block Diagram of any adaptive data selection algorithm, where data selection is performed every R epochs of (stochastic) gradient descent, and the gradient descent updates are performed on the subsets obtained by the data selection.

validation loss be denoted by L_V. In Table 1 in Appendix A, we organize and tabulate the notations used throughout this paper.

Adaptive data subset selection: (Stochastic) gradient descent algorithms proceed by computing the gradients of *all* the training data points for T epochs. We study the alternative approach of adaptive data selection, in which training is performed using a weighted sum of gradients of the training data subset instead of the full training data. In the adaptive data selection approach, the data selection is performed *in conjunction* with training such that subsets get incrementally refined as the learning algorithm proceeds. This incremental subset refinement allows the data selection to *adapt* to the learning and produces increasingly effective subsets with progress of the learning algorithm. Assume that an adaptive data selection algorithm produces weights $\mathbf{w}^t$ and subsets $\mathcal{X}^t$ for $t = 1, 2, \cdots, T$ through the course of the algorithm. In other words, at iteration t, the parameters θ_t are updated using the weighted loss of the model on subset $\mathcal{X}^t$ by weighing each example $i \in \mathcal{X}^t$ with its corresponding weight w_i^t. For example, if we use gradient descent, we can write the update step as: $\theta_{t+1} = \theta_t - \alpha \sum_{i \in \mathcal{X}^t} w_i^t \nabla_\theta L_T(x_i, y_i, \theta_t)$. Note that though this framework uses a potentially different set $\mathcal{X}^t$ in each iteration t, we need not perform data selection every epoch. In fact, in practice, we run data selection only every R epochs, in which case, the same subsets and weights will be used between epochs $t = R\tau$ and $t = R(\tau + 1)$. In contrast, the non-adaptive data selection settings (Wei et al., 2014b;c; Kaushal et al., 2019) employ the same $\mathcal{X}^t = \mathcal{X}$ for gradient descent in every iteration. In Figure 2, we present the broad adaptive data selection scheme. In this work, we focus on a setting in which the subset size is fixed, *i.e.*, $|\mathcal{X}^t| = k$ (typically a small fraction of the training dataset).

Convergence analysis: We now study the conditions for the convergence of either the full training loss or the validation loss achieved by any adaptive data selection strategy. Recall that we can characterize any data selection algorithm by a set of weights $\mathbf{w}^t$ and subsets $\mathcal{X}^t$ for $t = 1, \cdots, T$. We provide a convergence result which holds for any adaptive data selection algorithm, and applies to Lipschitz continuous, Lipschitz smooth, and strongly convex loss functions.

Before presenting the result, we define the term:

$$\text{Err}(\mathbf{w}^t, \mathcal{X}^t, L, L_T, \theta_t) = \left\| \sum_{i \in \mathcal{X}^t} w_i^t \nabla_\theta L_T^i(\theta_t) - \nabla_\theta L(\theta_t) \right\|$$

The norm considered in the above equation is the l2 norm. Next, assume that the parameters satisfy $||\theta||^2 \leq D^2$, and let L denote either the training or validation loss. We next state the convergence result:

Theorem 1 *Any adaptive data selection algorithm. run with full gradient descent (GD), defined via weights $\mathbf{w}^t$ and subsets $\mathcal{X}^t$ for $t = 1, \cdots, T$, enjoys the following guarantees:*

(1). If L_T is Lipschitz continuous with parameter σ_T, optimal model parameters are θ^*, and $\alpha = \frac{D}{\sigma_T \sqrt{T}}$, then $\min_{t=1:T} L(\theta_t) - L(\theta^*) \leq \frac{D\sigma_T}{\sqrt{T}} + \frac{D}{T} \sum_{t=1}^{T-1} Err(\mathbf{w}^t, \mathcal{X}^t, L, L_T, \theta_t)$.

(2) If L_T is Lipschitz smooth with parameter $\mathcal{L}_T$, optimal model parameters are θ^*, and L_T^i satisfies $0 \leq L_T^i(\theta) \leq \beta_T$, $\forall i$, then setting $\alpha = 1/\mathcal{L}_T$, we have $\min_{t=1:T} L(\theta_t) - L(\theta^*) \leq \frac{D^2 \mathcal{L}_T + 2\beta_T}{2T} + \frac{D}{T} \sum_{t=1}^{T-1} Err(\mathbf{w}^t, \mathcal{X}^t, L, L_T, \theta_t)$.

(3) If L_T is Lipschitz continuous with parameter σ_T, optimal model parameters are θ^*, and L is strongly convex with parameter μ, then setting a learning rate $\alpha_t = \frac{2}{\mu(1+t)}$, we have $\min_{t=1:T} L(\theta_t) - L(\theta^*) \leq \frac{2{\sigma_T}^2}{\mu(T+1)} + \sum_{t=1}^{t=T} \frac{2Dt}{T(T+1)} Err(\mathbf{w}^t, \mathcal{X}^t, L, L_T, \theta_t)$.

We present the proof in Appendix B.1. We can also extend this result to the case in which SGD is used instead of full GD to update the model. The main difference in the result is that the inequality holds with expectations over both sides. We defer the convergence result and proof to Appendix B.2. Theorem 1 suggests that an effective data selection algorithm should try to obtain subsets which have very small error $\text{Err}(\mathbf{w}^t, \mathcal{X}^t, L, L_T, \theta_t)$ for $t = 1, \cdots, T$. If the goal of data selection is to select a subset $\mathcal{X}^t$ at every epoch which *approximates* the full training set, the data selection procedure should try to minimize $\text{Err}(\mathbf{w}^t, \mathcal{X}^t, L_T, L_T, \theta_t)$. On the other hand, to select subset $\mathcal{X}^t$ at every epoch which *approximates* the validation set, the data selection procedure should try to minimize $\text{Err}(\mathbf{w}^t, \mathcal{X}^t, L_V, L_T, \theta_t)$. In Appendix B.3, we provide conditions under which an adaptive data selection approach reduces the loss function L (which can either be the training loss L_T or the validation loss L_V). In particular, we show that any data

selection approach which attempts to minimize the error term $\text{Err}(\mathbf{w}^t, \mathcal{X}^t, L, L_T, \theta_t)$ is also likely to reduce the loss value at every iteration. In the next section, we present GRAD-MATCH, that directly minimizes this error function $\text{Err}(\mathbf{w}^t, \mathcal{X}^t, L, L_T, \theta_t)$.

3. The GRAD-MATCH Algorithm

Following the result of Theorem 1, we now design an adaptive data selection algorithm that minimizes the gradient error term: $\text{Err}(\mathbf{w}^t, \mathcal{X}^t, L, L_T, \theta_t)$ (where L is either the training loss or the validation loss) as the training proceeds. The complete algorithm is shown in Algorithm 1. In Algorithm 1, isValid is a boolean Validation flag indicator that indicates whether to match the subset loss gradient with validation set loss gradient like in the case of class imbalance (isValid=True) or training set loss gradient (isValid=False). As discussed in Section 2, we do the subset selection only every R epochs, and during the rest of the epochs, we update the model parameters (using Batch SGD) on the previously chosen set $\mathcal{X}^t$ with associated weights $\mathbf{w}^t$. *This ensures that the subset selection time in itself is negligible compared to the training time, thereby ensuring that the adaptive selection runs as fast as simple random selection.* The full algorithm is shown in Algorithm 1. Line 9 of Algorithm 1 is the mini-batch SGD and takes as inputs the weights, subset of instances, learning rate, training loss, batch size, and the number of epochs. We randomly shuffle elements in the subset $\mathcal{X}^t$, divide them up into mini-batches of size B, and run mini-batch SGD with instance weights.

Lines 3 and 5 in Algorithm 1 are the data subset selection steps, run either with the full training gradients or validation gradients. We basically minimize the error term $\text{Err}(\mathbf{w}^t, \mathcal{X}^t, L, L_T, \theta_t)$ with minor difference. Define the regularized version of $\text{Err}(\mathbf{w}^t, \mathcal{X}^t, L, L_T, \theta_t)$ as:

$$\text{Err}_\lambda(\mathbf{w}, \mathcal{X}, L, L_T, \theta_t) = \text{Err}(\mathbf{w}, \mathcal{X}, L, L_T, \theta_t) + \lambda\|\mathbf{w}\|^2 \tag{1}$$

The data selection optimization problem then is:

$$\mathbf{w}^t, \mathcal{X}^t = \underset{\mathbf{w}, \mathcal{X}:|\mathcal{X}|\leq k}{\text{argmin}} \text{Err}_\lambda(\mathbf{w}, \mathcal{X}, L, L_T, \theta_t) \tag{2}$$

In $\text{Err}_\lambda(\mathbf{w}, \mathcal{X}, L, L_T, \theta_t)$, the first term is the additional error term that adaptive subset algorithms have from the convergence analysis discussed in Section 2, and the second term is a squared l2 loss regularizer over the weight vector $\mathbf{w}$ with a regularization coefficient λ to prevent overfitting by discouraging the assignment of large weights to individual data instances or mini-batches. During data-selection, we select the weights $\mathbf{w}^t$ and subset $\mathcal{X}^t$ by optimizing equation (2). To this end, we define:

$$E_\lambda(\mathcal{X}) = \min_{\mathbf{w}} \text{Err}_\lambda(\mathbf{w}, \mathcal{X}, L, L_T, \theta_t) \tag{3}$$

Note that the optimization problem in Eq. (2) is equivalent to solving the optimization problem $\min_{\mathcal{X}:|\mathcal{X}|\leq k} E_\lambda(\mathcal{X})$. The detailed optimization algorithm is presented in Section 3.1.

GRAD-MATCH for mini-batch SGD: We now discuss an alternative formulation of GRAD-MATCH, specifically for mini-batch SGD. Recall from Theorem 1 and optimization problem (2), that in the case of full gradient descent or SGD, we select a subset of data points for the data selection. However, mini-batch SGD is a combination of SGD and full gradient descent, where we randomly select a mini-batch and compute the full gradient on that mini-batch. To handle this, we consider a variant of GRAD-MATCH, which we refer to as GRAD-MATCHPB. Here we select a subset of mini-batches by matching the weighted sum of mini-batch training gradients to the full training loss (or validation loss) gradients. Once we select a subset of mini-batches, we train the neural network on the mini-batches, weighing each mini-batch by its corresponding weight. Let B be the batch size, $b_n = n/B$ as the total number of mini-batches, and $b_k = k/B$ as the number of batches to be selected. Let $\nabla_\theta L_T^{B_1}(\theta_t), \cdots, \nabla_\theta L_T^{B_{b_n}}(\theta_t)$ denote the mini-batch gradients. The optimization problem is then to minimize $E_\lambda^B(\mathcal{X})$, which is defined as:

$$E_\lambda^B(\mathcal{X}) = \min_{\mathbf{w}} \|\sum_{i\in\mathcal{X}} w_t^i \nabla_\theta L_T^{B_i}(\theta_t) - \nabla_\theta L(\theta_t)\| + \lambda\|\mathbf{w}\|^2$$

The constraint now is $|\mathcal{X}| \leq b_k$, where $\mathcal{X}$ is a subset of mini-batches instead of being a subset of data points. The use of mini-batches considerably reduces the number of selection rounds during the OMP algorithm by a factor of B, resulting in $B\times$ speed up. In our experiments, we compare the performance of GRAD-MATCH and GRAD-MATCHPB and show that GRAD-MATCHPB is considerably more efficient while being comparable in performance. GRAD-MATCHPB is a simple modification to lines 3 and 5 of Algorithm 1, where we send the mini-batch gradients instead of individual gradients to the orthogonal matching pursuit (OMP) algorithm (discussed in the next section). Further, we use the subset of mini-batches selected directly without any additional shuffling or sampling in our current experiments. We will consider augmenting the selected mini-batch subsets with additional shuffling or including new mini-batches with augmented images in our future work.

3.1. Orthogonal Matching Pursuit (OMP) algorithm

We next study the optimization algorithm for solving equation (2). Our objective is to minimize $E_\lambda(\mathcal{X})$ subject to the constraint $\mathcal{X} : |\mathcal{X}| \leq k$. We can also convert this into a maximization problem. For that, define: $F_\lambda(\mathcal{X}) = L_{\max} - \min_{\mathbf{w}} \text{Err}_\lambda(\mathcal{X}, \mathbf{w}, L, L_T, \theta_t)$. Note that we minimize $E_\lambda(\mathcal{X})$ subject to the constraint $\mathcal{X} : |\mathcal{X}| \leq k$ until $E_\lambda(\mathcal{X}) \leq \epsilon$, where ϵ is the tolerance level. Note that minimizing E_λ is equivalent to maximizing F_λ. The following result shows that F_λ is weakly submodular.

Theorem 2 *If $|\mathcal{X}| \leq k$ and $\max_i ||\nabla_\theta L_T^i(\theta_t)||_2 < \nabla_{\max}$, then $F_\lambda(\mathcal{X})$ is γ-weakly submodular, with $\gamma \geq \frac{\lambda}{\lambda + k\nabla_{\max}^2}$*

We present the proof in Appendix B.4. Recall that a set

Algorithm 1 GRAD-MATCH Algorithm

Require: Train set: $\mathcal{U}$; validation set: $\mathcal{V}$; initial subset: $\mathcal{X}^{(0)}$; subset size: k; TOL: ϵ; initial params: θ_0; learning rate: α; total epochs: T, selection interval: R, Validation Flag: isValid, Batchsize: B

for epochs t in $1, \cdots, T$ **do**
  if (t mod $R == 0$) and (isValid $== 1$) **then**
    $\mathcal{X}^t, \mathbf{w}^t = \text{OMP}(L_T, L_V, \theta_t, k, \epsilon)$
  else if (t mod $R == 0$) and (isValid $== 0$) **then**
    $\mathcal{X}^t, \mathbf{w}^t = \text{OMP}(L_T, L_T, \theta_t, k, \epsilon)$
  else
    $\mathcal{X}^t = \mathcal{X}^{t-1}$
  end if
  $\theta_{t+1} = \text{BatchSGD}(\mathcal{X}^t, \mathbf{w}^t, \alpha, L_T, B, \text{Epochs} = 1)$
end for
Output final model parameters θ_T

Algorithm 2 OMP

Require: Training loss L_T, target loss: L, current parameters: θ, regularization coefficient: λ, subset size: k, tolerance: ϵ

$\mathcal{X} \leftarrow \emptyset$
$r \leftarrow \nabla_w \text{Err}_\lambda(\mathcal{X}, \mathbf{w}, L, L_T, \theta)|_{\mathbf{w}=0}$
while $|\mathcal{X}| \leq k$ and $E_\lambda(\mathcal{X}) \geq \epsilon$ **do**
  $e = \text{argmax}_j |r_j|$
  $\mathcal{X} \leftarrow \mathcal{X} \cup \{e\}$
  $\mathbf{w} \leftarrow \text{argmin}_{\mathbf{w}} \text{Err}_\lambda(\mathcal{X}, \mathbf{w}, L, L_T, \theta)$
  $r \leftarrow \nabla_{\mathbf{w}} \text{Err}_\lambda(\mathcal{X}, \mathbf{w}, L, L_T, \theta)$
end while
return $\mathcal{X}$, $\mathbf{w}$

function $F : 2^{[n]} \rightarrow \mathbb{R}$ is γ-weakly submodular (Gatmiry & Gomez-Rodriguez, 2018; Das & Kempe, 2011) if $F(j|S) \geq \gamma F(j|T), S \subseteq T \subseteq [n]$. Since $F_\lambda(\mathcal{X})$ is approximately submodular (Das & Kempe, 2011), a greedy algorithm (Nemhauser et al., 1978; Das & Kempe, 2011; Elenberg et al., 2018) admits a $(1 - \exp(-\gamma))$ approximation guarantee. While the greedy algorithm is very appealing, it needs to compute the gain $F_\lambda(j|\mathcal{X})$, $O(nk)$ number of times. Since computation of each gain involves solving a least squares problem, this step will be computationally expensive, thereby defeating the purpose of data selection. To address this issue, we consider a slightly different algorithm, called the orthogonal matching pursuit (OMP) algorithm, studied in (Elenberg et al., 2018). We present OMP in Algorithm 2. In the corollary below, we provide the approximation guarantee for Algorithm 2.

Corollary 1 *Algorithm 2, when run with a cardinality constraint $|\mathcal{X}| \leq k$ returns a $1 - \exp\left(\frac{-\lambda}{\lambda + k\nabla^2_{\max}}\right)$ approximation for maximizing $F_\lambda(\mathcal{X})$ with $\mathcal{X} : |\mathcal{X}| \leq k$.*

We can also find the minimum sized subset such that the resulting error $E_\lambda(\mathcal{X}) \leq \epsilon$. We note that this problem is essentially: $\min_{\mathcal{X}} |\mathcal{X}|$ such that $F_\lambda(\mathcal{X}) \geq L_{\max} - \epsilon$. This is a weakly submodular set cover problem, and the following theorem shows that a greedy algorithm (Wolsey, 1982) as well as OMP (Algorithm 2) with a stopping criterion $E_\lambda(\mathcal{X}) \leq \epsilon$ achieve the following approximation bound:

Theorem 3 *If the function $F_\lambda(\mathcal{X})$ is γ-weakly submodular, $\mathcal{X}^*$ is the optimal subset and $\max_i ||\nabla_\theta L^i_T(\theta_t)||_2 < \nabla_{\max}$, (both) the greedy algorithm and OMP (Algorithm 2), run with stopping criteria $E_\lambda(\mathcal{X}) \leq \epsilon$ result in sets $\mathcal{X}$ such that $|\mathcal{X}| \leq \frac{|\mathcal{X}^*|}{\gamma} \log\left(\frac{L_{\max}}{\epsilon}\right)$ where $L_{\max}$ is an upper bound of F_λ .*

The proof of this theorem is in Appendix B.5. Finally, we derive the convergence result for GRAD-MATCH as a corollary of Theorem 1. In particular, assume that by running OMP, we can achieve sets $\mathcal{X}^t$ such that $E_\lambda(\mathcal{X}^t) \leq \epsilon$, for all $t = 1, \cdots, T$. If L is Lipschitz continuous, we obtain a convergence bound of $\min_t L(\theta_t) - L(\theta^*) \leq \frac{D\sigma_T}{\sqrt{T}} + D\epsilon$. In the case of smooth or strongly convex functions, the result can be improved to $O(1/T)$. For example, with smooth functions, we have: $\min_t L(\theta_t) - L(\theta^*) \leq \frac{D^2\mathcal{L}_T + 2\beta_T}{2T} + D\epsilon$. More details can be found in Appendix B.6.

3.2. Connections to existing work

Next, we discuss connections of GRAD-MATCH to existing approaches such as CRAIG and GLISTER, and contrast the resulting theoretical bounds. Let $\mathcal{X}$ be a subset of k data points from the training or validation set. Consider the expression for loss $L(\theta) = \sum_{i \in \mathcal{W}} L(x_i, y_i, \theta)$ so that $L = L_T$ when $\mathcal{W} = \mathcal{U}$ and $L = L_V$ when $\mathcal{W} = \mathcal{V}$. Define $\hat{E}(\mathcal{X})$ to be:

$$\hat{E}(\mathcal{X}) = \sum_{i \in \mathcal{W}} \min_{j \in \mathcal{X}} \|\nabla_\theta L^i(\theta_t) - \nabla_\theta L^j_T(\theta_t)\| \quad (4)$$

Note that $\hat{E}(\mathcal{X})$ is an upper bound for $E(\mathcal{X})$ (Mirzasoleiman et al., 2020a):

$$E(\mathcal{X}) = \min_{\mathbf{w}} \text{Err}(\mathbf{w}, \mathcal{X}, L, L_T, \theta_t) \leq \hat{E}(\mathcal{X}) \quad (5)$$

Given the set $\mathcal{X}^t$ obtained by optimizing $\hat{E}$, the weight $\mathbf{w}^t_j$ associated with the j^{th} point in the subset $\mathcal{X}^t$ will be: $\mathbf{w}^t_j = \sum_{i \in \mathcal{W}} \mathbb{I}[j = \arg\min_{s \in \mathcal{X}^t} \|\nabla_\theta L^i_T(\theta_t) - \nabla_\theta L^s(\theta_t)\|]$. Note that we can minimize both sides with respect to a cardinality constraint $\mathcal{X} : |\mathcal{X}| \leq k$; the right hand side of eqn. (5) is minimized when $\mathcal{X}$ is the set of k *medoids* (Kaufman et al., 1987) for all the components in the gradient space. In Appendix B.7, we prove the inequality (5), and also discuss the maximization version of this problem and how it relates to facility location. Similar to GRAD-MATCH, we also use the mini-Batch version for CRAIG (Mirzasoleiman et al., 2020a), which we refer to as CRAIGPB. Since CRAIGPB operates on a much smaller groundset (of mini-batches), it is much more efficient than the original version of CRAIG.

Next, we point out that a convergence bound, very similar to GRAD-MATCH can be shown for CRAIG as well. This result is *new and different* from the one shown in (Mirzasoleiman et al., 2020a) since it is for full GD and SGD, and not for incremental gradient descent algorithms discussed in (Mirzasoleiman et al., 2020a). If the subsets $\mathcal{X}^t$ obtained by running CRAIG satisfy $\hat{E}(\mathcal{X}^t) \leq \epsilon, \forall t = 1, \cdots, T$, we

can show $O(1/\sqrt{T})$ and $O(1/T)$ bounds (depending on the nature of the loss) with an additional ϵ term. However, the bounds obtained for CRAIG will be weaker than the one for GRAD-MATCH since $\hat{E}$ is an upper bound of E. Hence, *to achieve the same error, a potentially larger subset could be required for* CRAIG *in comparison to* GRAD-MATCH.

Finally, we also connect GRAD-MATCH to GLISTER (Killamsetty et al., 2021). While the setup of GLISTER is different from GRAD-MATCH since it directly tries to optimize the generalization error via a bi-level optimization, the authors use a Taylor-series approximation to make GLISTER efficient. The Taylor-series approximation can be viewed as being similar to maximizing the dot product between $\sum_{i \in \mathcal{X}} \nabla_\theta L_T^i(\theta_t)$ and $\nabla_\theta L(\theta_t)$. Furthermore, GLISTER does not consider a weighted sum the way we do, and is therefore slightly sub-optimal. In our experiments, we show that GRAD-MATCH outperforms CRAIG and GLISTER across a number of deep learning datasets.

4. Speeding up GRAD-MATCH

In this section, we propose several implementational and practical tricks to make GRAD-MATCH scalable and efficient (in addition to those discussed above). In particular, we will discuss various approximations to GRAD-MATCH such as running OMP per class, using the last layer of the gradients, and warm-start to the data selection.

Last-layer gradients. The number of parameters in modern deep models is very large, leading to very high dimensional gradients. The high dimensionality of gradients slows down OMP, thereby decreasing the efficiency of subset selection. To tackle this problem, we adopt a last-layer gradient approximation similar to (Ash et al., 2020; Mirzasoleiman et al., 2020a; Killamsetty et al., 2021) by only considering the last layer gradients for neural networks in GRAD-MATCH. This simple trick significantly improves the speed of GRAD-MATCH and other baselines.

Per-class and per-gradient approximations of GRAD-MATCH: To solve the GRAD-MATCH optimization problem, we need to store the gradients of all instances in memory, leading to high memory requirements for large datasets. In order to tackle this problem, we consider per-class and per-gradient approximation. We solve multiple gradient matching problems using per-class approximation - one for each class by only considering the data instances belonging to that class. The per-class approximation was also adopted in (Mirzasoleiman et al., 2020a). To further reduce the memory requirements, we additionally adopt the per-gradient approximation by considering only the corresponding last linear layer's gradients for each class. The per-gradient and per-class approximations not only reduce the memory usage, but also significantly speed up (reduce running time of) the data selection itself. By default, we use the per-class and per-gradient approximation, with the last layer gradients and we will call this algorithm GRAD-MATCH. We do not need these approximations for GRAD-MATCHPB since it is on a much smaller ground-set (mini-batches instead of individual items).

Warm-starting data selection: For each of the algorithms we consider in this paper (*i.e.*, GRAD-MATCH, GRAD-MATCHPB, CRAIG, CRAIGPB, and GLISTER), we also consider a warm-start variant, where we run T_f epochs of full training. We set T_f in a way such that the number of epochs T_s with the subset of data is a fraction κ of the total number of epochs, *i.e.*, $T_s = \kappa T$ and $T_f = \frac{T_s k}{n}$, where k is the subset size. We observe that doing full training for the first few epochs helps obtain good *warm-start* models, resulting in much better convergence. Setting T_f to a large value yields results similar to the full training with early stopping (which we use as one of our baselines) since there is not enough data-selection.

Other speedups: We end this section by reiterating two implementation tricks already discussed in Section 3, namely, doing data selection every R epochs (in our experiments, we set $R = 20$, but also study the effect of the choice of R), and the per-batch (PB) versions of CRAIG and GRAD-MATCH.

5. Experiments

Our experiments aim to demonstrate the stability and efficiency of GRAD-MATCH. While in most of our experiments, we study the tradeoffs between accuracy and efficiency (time/energy), we also study the robustness of data-selection under class imbalance. For most data selection experiments, we use the full loss gradients (*i.e.*, $L = L_T$). As an exception, in the case of class imbalance, following (Killamsetty et al., 2021), we use $L = L_V$ (*i.e.*, we assume access to a clean validation set).

Baselines in each setting. We compare the variants of our proposed algorithm (*i.e.,* GRAD-MATCH, GRAD-MATCHPB, GRAD-MATCH-WARM, GRAD-MATCHPB-WARM) with variants of CRAIG (Mirzasoleiman et al., 2020a) (*i.e.,* CRAIG, CRAIGPB, CRAIG-WARM, CRAIGPB-WARM), and variants of GLISTER (Killamsetty et al., 2021) (*i.e.,* GLISTER, GLISTER-WARM). Additionally, we compare against RANDOM (*i.e.,* randomly select points equal to the budget), and FULL-EARLYSTOP, where we do an early stop to full training to match the time taken (or energy used) by the subset selection.

Datasets, model architecture and experimental setup: To demonstrate the effectiveness of GRAD-MATCH and its variants on real-world datasets, we performed experiments on CIFAR100 (60000 instances) (Krizhevsky, 2009), MNIST (70000 instances) (LeCun et al., 2010), CIFAR10 (60000 instances) (Krizhevsky, 2009), SVHN (99,289 instances) (Netzer et al., 2011), and ImageNet-2012 (1.4 Million instances) (Russakovsky et al., 2015) datasets. Wherever the datasets do not have a pre-specified validation set,

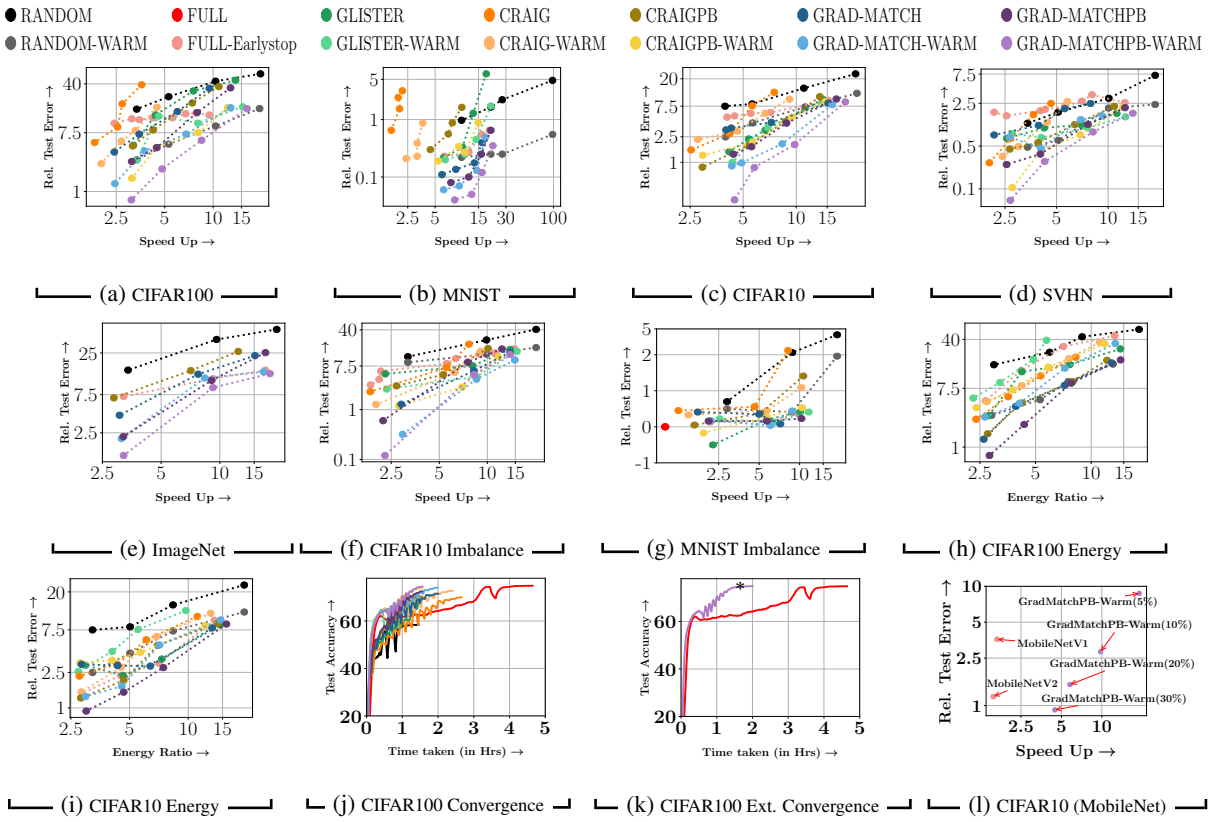

Figure 3. Sub-figures (a-g) show speedup vs relative error in % tradeoff (**both log-scale**) of different algorithms. In each scatter plot, smaller subsets are on the left, and larger ones are on the right. Results are shown for (a) CIFAR-100, (b) MNIST, (c) CIFAR-10, (d) SVHN, (e) ImageNet, (f) CIFAR-10 imbalance, and (g) MNIST imbalance. Sub-figures (h, i) show energy gains vs. relative error for CIFAR-100 and CIFAR-10. Sub-figure (j) shows a convergence plot of different strategies at 30% subset of CIFAR-100. Sub-figure (k) shows an extended convergence plot of GRAD-MATCHPB-WARM at 30% subset of CIFAR-100 by running it for more epochs. Sub-figure (l) shows results of GRAD-MATCHPB-WARM using ResNet18 and Full training using MobileNet-V1 and MobileNet-V2 models on the CIFAR-10 dataset. In every case, the speedups & energy ratios are computed w.r.t full training. *Variants of* GRAD-MATCH *achieve best speedup-accuracy tradeoff (bottom-right in each scatter plot represents best speedup-accuracy tradeoff region) in almost all cases.*

we split the original training set into a new train (90%) and validation sets (10%). We ran experiments using an SGD optimizer with an initial learning rate of 0.01, a momentum of 0.9, and a weight decay of 5e-4. We decay the learning rate using cosine annealing (Loshchilov & Hutter, 2017) for each epoch. For MNIST, we use the LeNet model (LeCun et al., 1989) and train the model for 200 epochs. For all other datasets, we use the ResNet18 model (He et al., 2016) and train the model for 300 epochs (except for ImageNet, where we train the model for 350 epochs). In most of our experiments, we train the data selection algorithms (and full training) using the same number of epochs; the only difference is that each epoch is much smaller with smaller subsets, thereby enabling speedups/energy savings. We consider one additional experiment where we run GRAD-MATCHPB-WARM for 50 more epochs to see how quickly it achieves comparable accuracy to full training. All experiments were run on V100 GPUs. Furthermore, the accuracies reported in the results are mean accuracies after five runs, and the standard deviations are given in Appendix C.5. More details are in Appendix C.

Data selection setting: Since the goal of our experiments is efficiency, we use smaller subset sizes. For MNIST, we use sizes of [1%, 3%, 5%, 10%], for ImageNet-2012 we use [5%, 10%, 30%], while for the others, we use [5%, 10%, 20%, 30%]. For the warm versions, we set $\kappa = 1/2$ (i.e. 50% warm-start and 50% data selection). Also, we set $R = 20$ in all experiments. In our ablation study experiments, we study the effect of varying R and κ.

Speedups and energy gains compared to full training: In Figures 3a,3b,3c,3d,3e, we present scatter plots of relative error *vs.* speedups, both *w.r.t* full training. Figures 3h,3i show scatter plots of relative error *vs.* energy efficiency, again *w.r.t* full training. In each case, we also include the cost of subset selection and subset training while computing the wall-clock time or energy consumed. For calculating the

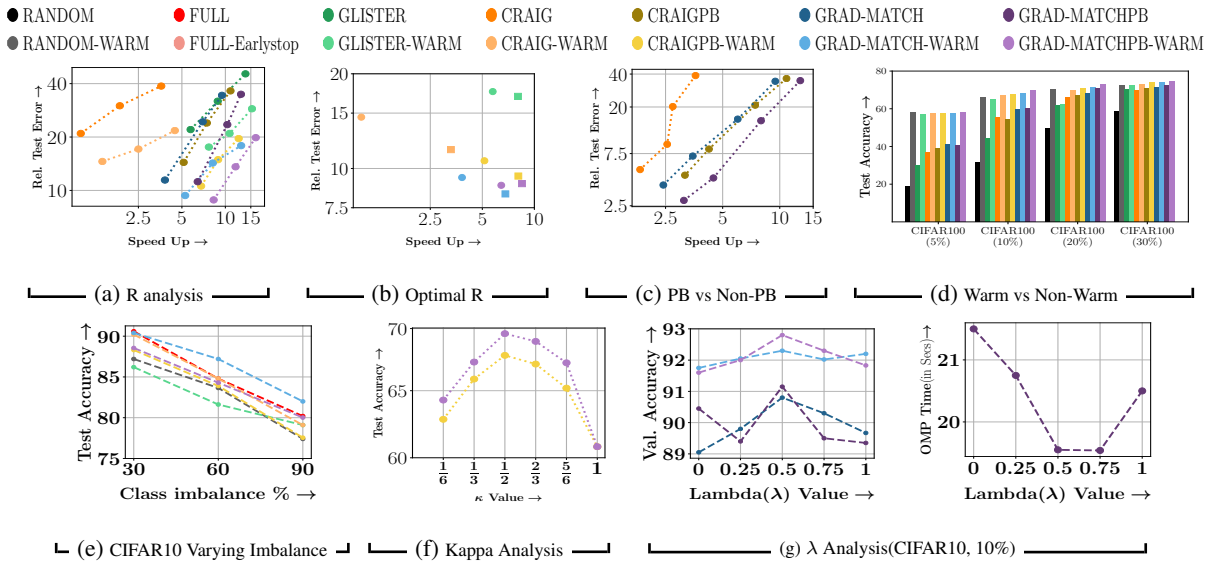

Figure 4. Sub-figure (a) compares the effect of varying R (5, 10 and 20) for different strategies (5% CIFAR-100). Sub-figure (b) compares $R = 5$ with a 5% subset (circles) and $R = 20$ with 10% subset (squares) showing that the latter is more efficient and accurate. Sub-figure (c) compares the per-batch (PB) versions with non-PB versions showing that the former has a better accuracy-efficiency trade-off. Sub-figure (d) compares the warm-start variants with the variants without warm-start for different subset sizes of CIFAR-100. Sub-figure (e) shows the performance of different selection strategies for 30% CIFAR-10 with varying percentages of imbalanced classes: 30%, 60%, and 90%. GRAD-MATCH-WARM outperforms all baselines, including full training (which under-performs due to a high imbalance). Sub-figure (f) shows the effect of the warm-start parameter κ for CIFAR-100. Sub-figure (g) shows the effect of the regularization parameter λ on GRAD-MATCH and its variants for 10% CIFAR-10.

energy consumed by the GPU/CPU cores, we use pyJoules[1]. As a first takeaway, we note that GRAD-MATCH and its variants, achieve significant speedup (single GPU) and energy savings when compared to full training. In particular (*c.f.*, Figure 1) for CIFAR-10, GRAD-MATCHPB-WARM achieves a 7x, 4.2x and 3x speedup and energy gains (with 10%, 20%, 30% subsets) with an accuracy drop of only 2.8%, 1.5% and 0.9% respectively. For CIFAR-100, GRAD-MATCHPB-WARM achieves a 4.8x and 3x speedup with an accuracy loss of 2.1% and 0.7% respectively, while for ImageNet, (30% subset), GRAD-MATCHPB-WARM achieves a 3x speedup with an accuracy loss of 1.3%. The gains are even more significant for MNIST.

Comparison to other baselines: GRAD-MATCHPB-WARM not only outperforms random selection and FULL-EARLYSTOP consistently, but also outperforms variants of CRAIG, CRAIGPB, and GLISTER. Furthermore, GLISTER and CRAIG could not run on ImageNet due to large memory requirements and running time. GRAD-MATCH, GRAD-MATCHPB, and CRAIGPB were the only variants which could scale to ImageNet. Furthermore, GLISTER and CRAIG also perform poorly on CIFAR-100. We see that GRAD-MATCHPB-WARM almost consistently achieves best speedup-accuracy tradeoff (*i.e.*, the bottom right of the plots) on all datasets. We note that the performance gain provided by the variants of GRAD-MATCH compared to other baselines like GLISTER, CRAIG and FULL-EARLYSTOP is statistically significant (Wilcoxon signed-rank test (Wilcoxon, 1992) with a p value $= 0.01$). More details on the comparison (along with a detailed table of numbers) are in Appendix C.

Convergence and running time: Next, we compare the end-to-end training performance through a convergence plot. We plot test-accuracy versus training time in Figure 3j. The plot shows that GRAD-MATCH and specifically GRAD-MATCHPB-WARM is more efficient compared to other algorithms (including variants of GLISTER and CRAIG), and also converges faster than full training. Figure 3k shows the extended convergence of GRAD-MATCHPB-WARM on 30% CIFAR-100 subset, where the GRAD-MATCH is allowed to train for as few more epochs to achieve comparable accuracy with Full training at the cost of losing some efficiency. The results show that GRAD-MATCHPB-WARM achieves similar performance to full training while being 2.5x faster, after running for just 30 to 50 additional epochs. Note that the points marked by * in Figure 3k denotes the standard training endpoint (i.e., 300 epochs) used for all experiments using CIFAR-100.

Comparison to smaller models: We compare the speedups achieved by GRAD-MATCH to the speedups achieved using smaller models for training to understand the

[1] https://pypi.org/project/pyJoules/.

importance of data subset selection. We perform additional experiments (*c.f.*, Figure. 3l) on the CIFAR-10 dataset using MobileNet-V1 and MobileNet-V2 models as two proxies for small models. The results show that GRAD-MATCHPB-WARM outperforms the smaller models on both test accuracy and speedup (e.g., MobileNetV2 achieves less than 2x speedup with 1% accuracy drop).

Data selection with class imbalance: We check the robustness of GRAD-MATCH and its variants for generalization by comparing the test accuracies achieved on a clean test dataset when class imbalance is present in the training dataset. Following (Killamsetty et al., 2021), we form a dataset by making 30% of the classes imbalanced by reducing the number of data points by 90%. We present results on CIFAR10 and MNIST in Figures 3f,3g respectively. We use the (clean) validation loss for gradient matching in the class imbalance scenario since the training data is biased. The results show that GRAD-MATCH and its variants outperform other baselines in all cases except for the 30% MNIST case (where GLISTER, which also uses a clean validation set, performs better). Furthermore, in the case of MNIST with imbalance, GRAD-MATCH-WARM even outperforms training on the entire dataset. Figure 4e shows the performance on 10% CIFAR-10 with varying percentages of imbalanced classes: 30% (as shown in Figure 3f), 60% and 90%. Grad-Match-Warm outperforms all baselines, including full training (which under-performs due to a high imbalance). The trend is similar when we vary the degree of imbalance as well.

Ablation study results. Next, we study the effect of R, per-batch gradients, warm-start, λ, κ and other hyperparameters. We start with the effect of R on the performance of GRAD-MATCH and its variants. We study the result on CIFAR-100 dataset at 5% subset for varying values of R (5,10,20) in the Figure 4a (with leftmost point corresponding to R=5 and the rightmost to R=20).The first takeaway is that, as expected, GRAD-MATCH and its variants outperform other baselines for different R values. Secondly, this also helps us understand the accuracy-efficiency trade-off with different values of R. From Figure 4b, we see that a 10% subset with $R = 20$ yields accuracy similar to a 5% subset with $R = 5$. However, across the different algorithms, we observe that $R = 20$ is more efficient (from a time perspective) because of fewer subset selection runs. We then compare the PB variants of CRAIG and GRAD-MATCH with their non-PB variants (*c.f.*, Figure 4c). We see that the PB versions are efficient and lie consistently to the bottom right (, *i.e.*, lesser relative test accuracy and higher speedups) than non-PB counterparts. One of the main reasons for this is that the subset selection time for the PB variants is almost half that of the non-PB variant (*c.f.*, Appendix C.4), with similar relative errors. Next, we study the effect of warm-start along with data selection. As shown in Figure 4d, warm-start, in the beginning, helps the model come to a reasonable starting point for data selection, something which just random sets do not offer. We also observe that the effect of warm-start is more significant for smaller subset sizes (compared to larger sizes) in achieving large accuracy gains compared to the non-warm start versions. Figure 4f shows the effect of varying κ (*i.e.*, the warm-start fraction) for 10% of CIFAR-100. We observe that setting $\kappa = \frac{1}{2}$ generally performs the best. Setting a small value of κ leads to sub-optimal performance because of not having a good starting point, while with larger values of κ we do not have enough data selection and get results closer to the early stopping. The regularization parameter λ prevents OMP from over-fitting (e.g., not assigning large weights to individual samples or mini-batches) since the subset selection is performed only every 20 epochs. Hence variants of GRAD-MATCH performs poorly for small lambda values (e.g., λ=0) as shown in Figure. 4g. Similarly, GRAD-MATCH and its variants perform poorly for large λ values as the OMP algorithm performs sub-optimally due to stronger restrictions on the possible sample weights. In our experiments, we found that $\lambda = 0.5$ achieves the accuracy and efficiency (*c.f.*, Figure. 4g), and this holds consistently across subset sizes and datasets. Furthermore, we observed that ϵ does not significantly affect the performance of GRAD-MATCH and its variants as long as ϵ is small (e.g., $\epsilon \leq 0.01$).

6. Conclusions

We introduce a *Gradient Matching* framework GRAD-MATCH, which is inspired by the convergence analysis of adaptive data selection strategies. GRAD-MATCH optimizes an error term, which measures how well the weighted subset matches either the full gradient or the validation set gradients. We study the algorithm's theoretical properties (convergence rates and approximation bounds, connections to weak-submodularity) and finally demonstrate our algorithm's efficacy by demonstrating that it achieves the best speedup-accuracy trade-off and is more energy-efficient through experiments on several datasets.

Acknowledgements

We thank anonymous reviewers, Baharan Mirzasoleiman and Nathan Beck for providing constructive feedback. Durga Sivasubramanian is supported by the Prime Minister's Research Fellowship. Ganesh and Abir are also grateful to IBM Research, India (specifically the IBM AI Horizon Networks - IIT Bombay initiative) for their support and sponsorship. Abir also acknowledges the DST Inspire Award and IITB Seed Grant. Rishabh acknowledges UT Dallas startup funding grant.

References

Ash, J. T., Zhang, C., Krishnamurthy, A., Langford, J., and Agarwal, A. Deep batch active learning by diverse, uncertain gradient lower bounds. In *ICLR*, 2020.

Campbell, T. and Broderick, T. Bayesian coreset construction via greedy iterative geodesic ascent. In *International Conference on Machine Learning*, pp. 698–706, 2018.

Clarkson, K. L. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Transactions on Algorithms (TALG)*, 6(4):1–30, 2010.

Coleman, C., Yeh, C., Mussmann, S., Mirzasoleiman, B., Bailis, P., Liang, P., Leskovec, J., and Zaharia, M. Selection via proxy: Efficient data selection for deep learning, 2020.

Das, A. and Kempe, D. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. *arXiv preprint arXiv:1102.3975*, 2011.

Elenberg, E. R., Khanna, R., Dimakis, A. G., Negahban, S., et al. Restricted strong convexity implies weak submodularity. *The Annals of Statistics*, 46(6B):3539–3568, 2018.

Feldman, D. Core-sets: Updated survey. In *Sampling Techniques for Supervised or Unsupervised Tasks*, pp. 23–44. Springer, 2020.

Fujishige, S. *Submodular functions and optimization*. Elsevier, 2005.

Gatmiry, K. and Gomez-Rodriguez, M. Non-submodular function maximization subject to a matroid constraint, with applications. *arXiv preprint arXiv:1811.07863*, 2018.

Har-Peled, S. and Mazumdar, S. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pp. 291–300, 2004.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Jia, X., Song, S., He, W., Wang, Y., Rong, H., Zhou, F., Xie, L., Guo, Z., Yang, Y., Yu, L., et al. Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes. *arXiv preprint arXiv:1807.11205*, 2018.

Kaufman, L., Rousseeuw, P., and Dodge, Y. Clustering by means of medoids in statistical data analysis based on the, 1987.

Kaushal, V., Iyer, R., Kothawade, S., Mahadev, R., Doctor, K., and Ramakrishnan, G. Learning from less data: A unified data subset selection and active learning framework for computer vision. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1289–1299. IEEE, 2019.

Killamsetty, K., Sivasubramanian, D., Ramakrishnan, G., and Iyer, R. Glister: Generalization based data subset selection for efficient and robust learning. *In AAAI*, 2021.

Kirchhoff, K. and Bilmes, J. Submodularity for data selection in machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 131–141, 2014.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, 2009.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

LeCun, Y., Cortes, C., and Burges, C. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts, 2017.

Mirzasoleiman, B., Karbasi, A., Badanidiyuru, A., and Krause, A. Distributed submodular cover: Succinctly summarizing massive data. In *Advances in Neural Information Processing Systems*, pp. 2881–2889, 2015.

Mirzasoleiman, B., Bilmes, J., and Leskovec, J. Coresets for data-efficient training of machine learning models, 2020a.

Mirzasoleiman, B., Cao, K., and Leskovec, J. Coresets for robust training of neural networks against noisy labels. *arXiv preprint arXiv:2011.07451*, 2020b.

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Schwartz, R., Dodge, J., Smith, N. A., and Etzioni, O. Green ai. *arXiv preprint arXiv:1907.10597*, 2019.

Settles, B. 2012. doi: 10.2200/S00429ED1V01Y201207AIM018.

Sharir, O., Peleg, B., and Shoham, Y. The cost of training nlp models: A concise overview. *arXiv preprint arXiv:2004.08900*, 2020.

Strubell, E., Ganesh, A., and McCallum, A. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

Toneva, M., Sordoni, A., des Combes, R. T., Trischler, A., Bengio, Y., and Gordon, G. J. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJlxm30cKm.

Wang, Y., Jiang, Z., Chen, X., Xu, P., Zhao, Y., Lin, Y., and Wang, Z. E2-train: Training state-of-the-art cnns with over 80% energy savings. *arXiv preprint arXiv:1910.13349*, 2019.

Wei, K., Iyer, R., and Bilmes, J. Fast multi-stage submodular maximization. In *International conference on machine learning*, pp. 1494–1502. PMLR, 2014a.

Wei, K., Liu, Y., Kirchhoff, K., Bartels, C., and Bilmes, J. Submodular subset selection for large-scale speech training data. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3311–3315. IEEE, 2014b.

Wei, K., Liu, Y., Kirchhoff, K., and Bilmes, J. Unsupervised submodular subset selection for speech data. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4107–4111. IEEE, 2014c.

Wei, K., Iyer, R., and Bilmes, J. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning*, pp. 1954–1963, 2015.

Wilcoxon, F. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pp. 196–202. Springer, 1992.

Wolf, G. W. Facility location: concepts, models, algorithms and case studies. series: Contributions to management science: edited by zanjirani farahani, reza and hekmatfar, masoud, heidelberg, germany, physica-verlag, 2009, 2011.

Wolsey, L. A. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4): 385–393, 1982.