GRADIENT MATCHING FOR EFFICIENT LEARNING

Krishnateja Killamsetty, Durga Sivasubramanian & Baharan Mirzasoleiman
krishnateja.killamsetty@utdallas.edu,durgas@cse.iitb.ac.in & baharan@cs.ucla.edu

Ganesh Ramakrishnan, Abir De & Rishabh Iyer
ganesh@cse.iitb.ac.in,abir@cse.iitb.ac.in,rishabh.iyer@utdallas.edu

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. Our extensive experiments on real-world datasets show that GRAD-MATCH significantly and consistently outperforms several recent data-selection algorithms and is Pareto-optimal with respect to the accuracy-efficiency trade-off[1].

1 INTRODUCTION

Modern machine learning systems, especially deep learning frameworks, have become increasingly 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 on 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 significantly reducing 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.

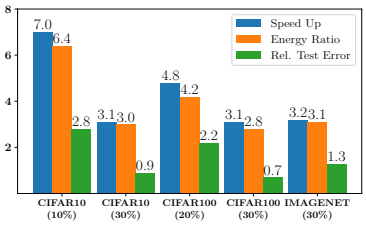

Figure 1: Efficiency of GRAD-MATCH relative to full training on the CIFAR-10, CIFAR-100, and Imagenet datasets.

Contributions of this work: Our work's significant contribution is in introducing the GRAD-MATCH, an end-to-end framework for efficient training of machine learning models. GRAD-MATCH, a gradient-matching algorithm (*c.f.*, Section 2), tries to select a subset of the training set for training by minimizing the *gradient matching error*. In our empirical results (*c.f.*, Section 3), *we show that* GRAD-MATCH *is Pareto-optimal in the accuracy and training-time trade-off when compared to state-of-the-art data selection approaches such as* CRAIG *(Mirzasoleiman et al., 2020),* GLISTER *(Killamsetty et al., 2021), random subsets, and even full training with early stopping.* See Figure 1 for some concrete numbers, wherein the time speedup and energy-saving ratio are nearly the same, and we refer to them jointly as efficiency improvement. For example, with ResNet-18 on Imagenet, we observe around 3x efficiency improvement with an accuracy

[1]The code of GRAD-MATCH is available at https://anonymous.4open.science/r/2eada7f6-5514-4409-8d90-ff4f8d7d5c03/

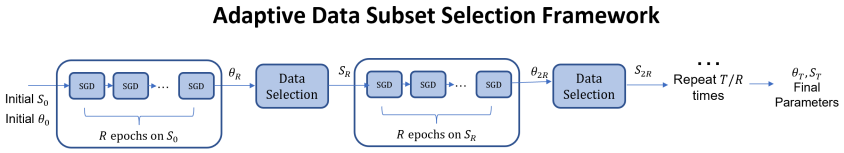

Figure 2: Block Diagram of GRAD-MATCH, where data selection is performed every R epochs of SGD and SGD updates are performed on the subsets obtained by GRAD-MATCH.

drop of 1.3%; for CIFAR-10, we observe 7x and 3x efficiency improvements with accuracy drops of 2.8% and 0.9%, respectively; for CIFAR-100, we achieve 4.8x and 3x efficiency improvements while losing around 2.1% and 0.7% in terms of accuracy, respectively.

Related work Coresets are weighted subsets of the data, which approximate certain desirable characteristics of the full data, *e.g.*, the loss function (Feldman, 2020). However, most coreset algorithms (Har-Peled & Mazumdar, 2004; Clarkson, 2010; Campbell & Broderick, 2018) require algorithms that are often specialized and very specific to the model and problem at hand and have had limited deep learning success. A very recent coreset algorithm called CRAIG (Mirzasoleiman et al., 2020) has shown promise for both deep learning and classical machine learning models such as logistic regression. Unlike other core-set techniques that largely focused on approximating loss functions, CRAIG tried to select representative subsets of the training data that closely approximate the full gradient. Another approach (Killamsetty et al., 2021; Wei et al., 2015) posed data selection as a discrete bi-level optimization problem and tried to select subsets that maximize the model validation accuracy. 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. Finally, our work is 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. This work demonstrates both energy and time savings *solely* based on a more principled subset selection approach.

2 THE GRAD-MATCH ALGORITHM

We now design an adaptive data selection algorithm that minimizes the gradient error term: $\text{Err}(\boldsymbol{w}^t, \mathbb{X}^t, L, L_T, \theta_t) = \|\sum_{i\in\mathbb{X}_t} w_i^t \nabla_\theta L_T^i(\theta_t) - \nabla_\theta L(\theta_t)\|$, where L is the training loss. To goal is to select a subset $\mathbb{X}$ such that the weighted sum over the gradients on the subset equals the full training gradients. The complete algorithm is shown in Algorithm 1. As shown in Figure 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 $\mathbb{X}^t$ and weights $\boldsymbol{w}^t$. *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.* 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 $\mathbb{X}^t$, divide them up into mini-batches of size B, and run mini-batch SGD with instance weights.

Lines 3 and 5 are the data subset selection steps, run either with the full training gradients or validation gradients. Recall, that the data selection optimization problem is:

$$\boldsymbol{w}^t, \mathbb{X}^t = \underset{\boldsymbol{w},\mathbb{X}:|\mathbb{X}|\leq k}{\arg\min} \text{Err}(\boldsymbol{w}, \mathbb{X}, L, L_T, \theta_t) = \underset{\boldsymbol{w},\mathbb{X}:|\mathbb{X}|\leq k}{\arg\min} \|\sum_{i\in\mathbb{X}_t} w_i^t \nabla_\theta L_T^i(\theta_t) - \nabla_\theta L(\theta_t)\| \tag{1}$$

During data-selection, we select the weights $\boldsymbol{w}^t$ and subset $\mathbb{X}^t$ by optimizing Eq (1). To this end, we define:

$$E(\mathbb{X}) = \min_{\boldsymbol{w}} \text{Err}(\boldsymbol{w}, \mathbb{X}, L, L_T, \theta_t) \tag{2}$$

Note that the optimization problem in Eq. (1) is equivalent to solving the optimization problem $\min_{X:|X|\leq k} E(X)$. The detailed optimization algorithm is presented in Section 2. We also note that CRAIG (Mirzasoleiman et al., 2020) is closely related to this, and minimizes an upper bound of Eq 2. As a result, we expect GRAD-MATCH to be more efficient compared to CRAIG.

GRAD-MATCH for mini-batch SGD: We now discuss an alternative formulation of GRAD-MATCH, specifically for mini-batch SGD, 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 $E_\lambda^B(\mathbb{X})$, which is defined as:

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

The constraint now is $|\mathbb{X}| \leq b_k$, where $\mathbb{X}$ is a subset of mini-batches instead of being a subset of data points. 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 OMP algorithm (discussed in the next section). Note that, we also use the mini-Batch version for CRAIG Mirzasoleiman et al. (2020), which we refer to as CRAIGPB. Since CRAIGPB operates on a much smaller ground set (of mini-batches), it is much more efficient than the original version of CRAIG.

Algorithm 1 GRAD-MATCH Algorithm

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

1: **for** epochs t in $1, \cdots, T$ **do**
2: **if** (t mod $R == 0$) and (isValid $== 1$) **then**
3: $\mathbb{X}^t, \boldsymbol{w}^t = \text{OMP}(L_T, L_V, \theta_t, k, \epsilon)$
4: **else if** (t mod $R == 0$) and (isValid $== 0$) **then**
5: $\mathbb{X}^t, \boldsymbol{w}^t = \text{OMP}(L_T, L_T, \theta_t, k, \epsilon)$
6: **else**
7: $\mathbb{X}^t = \mathbb{X}^{t-1}$
8: **end if**
9: $\theta_{t+1} = \text{BatchSGD}(\mathbb{X}^t, \boldsymbol{w}^t, \eta, L_T, B, \text{Epochs} = 1)$
10: **end for**
11: Output final model parameters $\theta^{(T)}$

Orthogonal Matching Pursuit (OMP) algorithm: We next study the optimization algorithm for solving equation (1). We first define the regularized version of (1) as:

$$\text{Err}_\lambda(\mathbb{X}, \boldsymbol{w}, L_T, L, \theta_t) = \|\sum_{i \in \mathbb{X}} w_t^i \nabla_\theta L_T^i(\theta_t) - \nabla_\theta L(\theta_t)\|^2 + \lambda ||\boldsymbol{w}||^2$$

Then, we have $E_\lambda(\mathbb{X}) = \min_{\boldsymbol{w}} \text{Err}_\lambda(\mathbb{X}, \boldsymbol{w}, L_T, L, \theta_t)$, and we would like to minimize $E_\lambda(\mathbb{X})$ subject to the constraint $\mathbb{X} : |\mathbb{X}| \leq k$. The detailed OMP algorithm is given in Appendix A.1.

Practical Implementational Tricks and Warm Start: To further improve the efficiency of GRAD-MATCH (and GRAD-MATCHPB), we adopt a set of practical tricks, which we discuss in Appendix A.2.1. We mention one essential trick, which is warm-starting. For each of the algorithms we consider in this paper, we consider a warm start variant, where we initially run a small number of epochs T_f of full training. In our experiments, we set T_f so that 50% of the effective number of epochs with the subset comprises full training. For example, if we are using a 10% subset of data, we run 15 epochs of full training (which is equivalent to 150 epochs of training with the subset) followed by 150 epochs of subset training. See more details in Appendix A.2.1

3 EXPERIMENTS

Our experiments aim to demonstrate the stability and efficiency of GRAD-MATCH where we study the tradeoffs between accuracy and efficiency (time/energy). For the data selection experiments, we use the full loss gradients (*i.e.*, $L = L_T$).

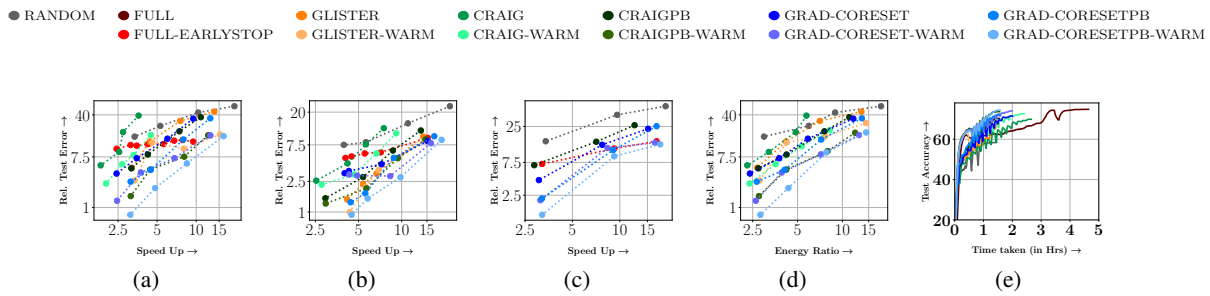

Figure 3: Sub-figures (a-c) show speedup vs relative error in % tradeoff (**both log-scale**) of different algorithms, with (a) CIFAR-100, (b) CIFAR-10, (c) ImageNet. Sub-figure (d) show energy gains vs. relative error for CIFAR-100. In each scatter plot, smaller subsets are on the left and larger are on the right. Sub-figure (e) shows a convergence plot of different strategies at 30% subset of CIFAR-100. In every case, the speedups & energy ratios are computed w.r.t full training. *Variants of* GRAD-MATCH *are Pareto-Optimal (bottom-right in each scatter plot is Pareto-Optimal frontier) in almost all cases.*

Baselines in each setting. We compare the variants of our propose: algorithm (*i.e.,* GRAD-MATCH, GRAD-MATCHPB, GRAD-MATCH-WARM, GRAD-MATCHPB-WARM) with variants of CRAIG Mirzasoleiman et al. (2020) (*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 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), CIFAR10 (60000 instances) Krizhevsky (2009), and ImageNet-2012 (1.4 Million instances) Russakovsky et al. (2015) datasets. More experimetal details are in Appendix A.3.

Results: Sub-figures (a-c) of Figure 3 show a scatter plot of relative error *vs.* speedups, both *w.r.t* full training. Sub-figures (d) show a scatter plot of 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 energy consumed by the GPU/CPU cores, we use pyJoules[2]. 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%**. In particular, we see that GRAD-MATCHPB-WARM is almost consistently at the Pareto-optimal (*i.e.*, the bottom right of the plots) in all datasets. 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. 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 A.3. Next, we compare the end-to-end training performance through a convergence plot. We plot test-accuracy versus training time in sub-figure(e) of Figure 3. 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.

4 CONCLUSIONS

We introduce a *Gradient Matching* framework GRAD-MATCH, which optimizes the gradient error between the weighted subset and the full training set. We demonstrate GRAD-MATCH algorithm's efficacy by demonstrating Pareto-optimal trade-offs between accuracy and efficiency on several

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

datasets. We are incredibly excited about the potential of GRAD-MATCH for making training more time and energy efficient and the potential to run model training on low-resource and edge devices, particularly in privacy-preserving and federated learning settings.

REFERENCES

Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *ICLR*, 2020.

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

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

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning, 2020.

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

Sariel Har-Peled and Soham Mazumdar. 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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Xianyan Jia, Shutao Song, Wei He, Yangzihao Wang, Haidong Rong, Feihu Zhou, Liqiang Xie, Zhenyu Guo, Yuanzhou Yang, Liwei Yu, et al. Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes. *arXiv preprint arXiv:1807.11205*, 2018.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. *In AAAI*, 2021.

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

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017.

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models, 2020.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green ai. *arXiv preprint arXiv:1907.10597*, 2019.

Burr Settles. Active learning. *Synthesis lectures on artificial intelligence and machine learning*, 6(1): 1–114, 2012.

Or Sharir, Barak Peleg, and Yoav Shoham. The cost of training nlp models: A concise overview. *arXiv preprint arXiv:2004.08900*, 2020.

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

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. 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.

Yue Wang, Ziyu Jiang, Xiaohan Chen, Pengfei Xu, Yang Zhao, Yingyan Lin, and Zhangyang Wang. E2-train: Training state-of-the-art cnns with over 80% energy savings. *arXiv preprint arXiv:1910.13349*, 2019.

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

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

Gert W Wolf. 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.

Supplementary Material

A Appendix

A.1 Orthogonal Matching Pursuit(OMP) Algorithm

We next study the optimization algorithm for solving equation (1). We first define the regularized version of (1) as:

$$\text{Err}_\lambda(\mathbb{X}, \boldsymbol{w}, L_T, L, \theta_t) = \big\| \sum_{i \in \mathbb{X}} w_t^i \nabla_\theta L_T^i(\theta_t) - \nabla_\theta L(\theta_t) \big\|^2 + \lambda ||\boldsymbol{w}||^2$$

Then, we have $E_\lambda(\mathbb{X}) = \min_{\boldsymbol{w}} \text{Err}_\lambda(\mathbb{X}, \boldsymbol{w}, L_T, L, \theta_t)$, and we would like to minimize $E_\lambda(\mathbb{X})$ subject to the constraint $\mathbb{X} : |\mathbb{X}| \leq k$. We can also convert this into a maximization problem. For that, define: $F_\lambda(\mathbb{X}) = L_{\max} - \min_{\boldsymbol{w}} \text{Err}_\lambda(\mathbb{X}, \boldsymbol{w}, L_T, L, \theta_t)$. Note that minimizing E_λ is equivalent to maximizing F_λ.

Algorithm 2 OMP

Require: Training loss L_T, target loss: L, current parameters: θ, regularization: λ, subset size: k, tolerance: ϵ
$\mathbb{X} \leftarrow \emptyset$
$r \leftarrow \nabla_w \text{Err}_\lambda(\mathbb{X}, \boldsymbol{w}, L_T, L, \theta)|_{\boldsymbol{w}=0}$
while $|\mathbb{X}| \leq k$ and $E_\lambda(\mathbb{X}) \geq \epsilon$ **do**
  $e = \text{argmax}_j |r_j|$
  $\mathbb{X} \leftarrow \mathbb{X} \cup \{e\}$
  $\boldsymbol{w} \leftarrow \text{argmin}_{\boldsymbol{w}} \text{Err}_\lambda(\mathbb{X}, \boldsymbol{w}, L_T, L, \theta)$
  $r \leftarrow \nabla_{\boldsymbol{w}} \text{Err}_\lambda(\mathbb{X}, \boldsymbol{w}, L_T, L, \theta)$
end while
return $\mathbb{X}$, $\boldsymbol{w}$

A.2 More details on Grad-Match

A.2.1 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. (2020); 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. (2020). 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 2, 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.

A.3 MORE EXPERIMENTAL DETAILS AND ADDITIONAL RESULTS

A.3.1 DATASETS DESCRIPTION

We used various standard datasets, namely, MNIST, CIFAR10, SVHN, CIFAR100, ImageNet, to demonstrate the effectiveness and stability of GRAD-MATCH.

Name	No. of classes	No. samples for training	No. samples for validation	No. samples for testing	No. of features
CIFAR10	10	50,000	-	10,000	32x32x3
MNIST	10	60,000	-	10,000	28x28
SVHN	10	73,257	-	26,032	32x32x3
CIFAR100	100	50,000	-	10,000	32x32x3
ImageNet	1000	1,281,167	50,000	100,000	224x224x3

Table 1: Description of the datasets

Table 1 gives a brief description about the datasets. Here not all datasets have an explicit validation and test set. For such datasets, 10% and 20% samples from the training set are used as validation and test set, respectively. The feature count given for the ImageNet dataset is after applying the `RandomResizedCrop` transformation function from PyTorch Paszke et al. (2017).

A.3.2 EXPERIMENTAL SETTINGS

We ran experiments using an SGD optimizer with an initial learning rate of 0.01, the 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 ResNet18 model He et al. (2016) and train the model for 300 epochs (except for ImageNet, where we train the model for 350 epochs).

To demonstrate our method's effectiveness in a robust learning setting, we artificially generate class-imbalance for the above datasets by removing almost 90% of the instances from 30% of total classes available. We ran all experiments on a single V100 GPU, except for ImageNet, where we used an RTX 2080 GPU. However, for a given dataset, all experiments were run on the same GPU so that the speedup and energy comparison across techniques is fair.

Data selection setting: Since the goal of our experiments is efficiency, we use smaller subset sizes. 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.

A.3.3 OTHER SPECIFIC SETTINGS

Here we discuss various parameters' required by Algorithm 2, their significance, and the values used in the experiments.

- **k** determines the subset size with which we train the model.
- ϵ determines the extent of gradient approximation we want. We use a value of 1e-10 in our experiments.

Data Selection Results										
		Budget(%)	Top-1 Test accuracy of the Model(%) 5%	10%	20%	30%	Model Training time(in hrs) 5%	10%	20%	30%
Dataset	Model	Selection Strategy								
CIFAR10	ResNet18	FULL (skyline for test accuracy)	95.09	95.09	95.09	95.09	4.34	4.34	4.34	4.34
		RANDOM (skyline for training time)	71.2	80.8	86.98	87.6	0.22	0.46	0.92	1.38
		GLISTER	85.5	91.92	92.78	93.63	0.43	0.91	1.13	1.46
		GLISTER-WARM	86.57	91.56	92.98	94.09	0.42	0.88	1.08	1.40
		CRAIG	82.74	87.49	90.79	92.53	0.81	1.08	1.45	2.399
		CRAIG-WARM	84.48	89.28	92.01	92.82	0.6636	0.91	1.31	2.20
		CRAIGPB	83.56	88.77	92.24	93.58	0.4466	0.70	1.13	2.07
		CRAIGPB-WARM	86.28	90.07	93.06	93.8	0.4143	0.647	1.07	2.06
		GRAD-MATCH	86.7	90.9	91.67	91.89	0.40	0.84	1.42	1.52
		GRAD-MATCH-WARM	87.2	92.15	92.11	92.01	0.38	0.73	1.24	1.41
		GRAD-MATCHPB	85.4	90.01	93.34	93.75	0.36	0.69	1.09	1.38
		GRAD-MATCHPB-WARM	86.37	92.26	93.59	94.17	0.32	0.62	1.05	1.36
CIFAR100	ResNet18	FULL (skyline for test accuracy)	75.37	75.37	75.37	75.37	4.871	4.871	4.871	4.871
		RANDOM (skyline for training time)	19.02	31.56	49.6	58.56	0.2475	0.4699	0.92	1.453
		GLISTER	29.94	44.03	61.56	70.49	0.3536	0.6456	1.11	1.5255
		GLISTER-WARM	57.17	64.95	62.14	72.43	0.3185	0.6059	1.06	1.452
		CRAIG	36.61	55.19	66.24	70.01	1.354	1.785	1.91	2.654
		CRAIG-WARM	57.44	67.3	69.76	72.77	1.09	1.48	1.81	2.4112
		CRAIGPB	38.95	54.59	67.12	70.61	0.4489	0.6564	1.15	1.540
		CRAIGPB-WARM	57.66	67.8	70.84	73.79	0.394	0.6030	1.10	1.5567
		GRAD-MATCH	41.01	59.88	68.25	71.5	0.5143	0.8114	1.40	2.002
		GRAD-MATCH-WARM	57.72	68.23	71.34	74.06	0.3788	0.7165	1.30	1.985
		GRAD-MATCHPB	40.53	60.39	70.88	72.57	0.3797	0.6115	1.09	1.56
		GRAD-MATCHPB-WARM	58.26	69.58	73.2	74.62	0.300	0.5744	1.01	1.5683
SVHN	ResNet18	FULL (skyline for test accuracy)	96.49	96.49	96.49	96.49	6.436	6.436,	6.436	6.436
		RANDOM (skyline for training time)	89.33	93.477	94.7	95.31	0.342	0.6383	1.26	1.90
		GLISTER	94.78	95.37	95.5	95.82	0.5733	0.9141	1.62	2.514
		GLISTER-WARM	94.99	95.50	95.8	95.69	0.5098	0.8522	1.58	2.34
		CRAIG	94.003	94.86	95.83	96.223	1.3886	1.7566	2.39	3.177
		CRAIG-WARM	93.81	95.27	96.0	96.15	1.113	1.4599	2.15	2.6617
		CRAIGPB	94.26	95.367	95.92	96.043	0.5934	1.009	1.65	2.413
		CRAIGPB-WARM	94.339	95.724	96.06	96.385	0.5279	0.93406	1.58	2.332
		GRAD-MATCH	94.01	94.45	95.4	95.73	0.8153	1.1541	1.64	2.981
		GRAD-MATCH-WARM	94.94	95.13	96.03	95.79	0.695	0.9313	1.59	2.417
		GRAD-MATCHPB	94.37	95.36	96.12	96.24	0.5134	0.8438	1.6	2.52
		GRAD-MATCHPB-WARM	94.77	95.64	96.21	96.425	0.4618	0.7889	1.51	2.398

Table 2: Data Selection Results for CIFAR10, CIFAR100 and SVHN datasets

MNIST Data Selection Results										
			Top-1 Test accuracy of the Model(%)				Model Training time(in hrs)			
		Budget(%)	1%	3%	5%	10%	1%	3%	5%	10%
Dataset	Model	Selection Strategy								
MNIST	LeNet	FULL (skyline for test accuracy)	99.35	99.35	99.35	99.35	0.82	0.82	0.82	0.82
		RANDOM (skyline for training time)	94.55	97.14	97.7	98.38	0.0084	0.03	0.04	0.084
		GLISTER	93.11	98.062	99.02	99.134	0.045	0.0625	0.082	0.132
		GLISTER-WARM	97.63	98.9	99.1	99.15	0.04	0.058	0.078	0.127
		CRAIG	96.18	96.93	97.81	98.7	0.3758	0.4173	0.434	0.497
		CRAIG-WARM	98.48	98.96	99.12	99.14	0.2239	0.258	0.2582	0.3416
		CRAIGPB	97.72	98.47	98.79	99.05	0.08352	0.106	0.1175	0.185
		CRAIGPB-WARM	98.47	99.08	99.01	99.16	0.055	0.077	0.0902	0.1523
		GRAD-MATCH	98.954	99.174	99.214	99.24	0.05	0.0607	0.097	0.138
		GRAD-MATCH-WARM	98.86	99.22	99.28	99.29	0.046	0.057	0.089	0.132
		GRAD-MATCHPB	98.7	99.1	99.25	99.27	0.04	0.051	0.07	0.11
		GRAD-MATCHPB-WARM	99.0	99.23	99.3	99.31	0.038	0.05	0.065	0.10

Table 3: Data Selection Results for MNIST dataset

ImageNet Data Selection Results								
			Top-1 Test accuracy(%)			Model Training time(in hrs)		
		Budget(%)	5%	10%	30%	5%	10%	30%
Dataset	Model	Selection Strategy						
ImageNet	ResNet18	FULL (skyline for test accuracy)	70.36	70.36	70.36	276.28	276.28	276.28
		RANDOM (skyline for training time)	21.124	33.512	55.12	14.12	28.712	81.7
		CRAIGPB	44.28	55.36	63.52	22.24	38.9512	96.624
		GRAD-MATCH	47.24	56.81	66.21	18.24	35.7042	90.25
		GRAD-MATCH-WARM	55.86	58.21	68.241	16.48	33.024	88.248
		GRAD-MATCHPB	45.15	59.04	68.12	16.12	30.472	86.32
		GRAD-MATCHPB-WARM	56.61	61.16	69.06	15.28	29.964	86.05

Table 4: Data Selection Results for ImageNet dataset

- λ determines how much regularization we want. We set $\lambda = 0.5$.

A.3.4 DATA SELECTION RESULTS:

This section shows the results of training neural networks on subsets selected by different data selection strategies for various datasets. Table 2 shows the test accuracy and the training time of the ResNet18 model on CIFAR10, CIFAR100, and SVHN datasets for 300 epochs. Table 3 shows the test accuracy and the training time of the LeNet model on the MNIST dataset for 200 epochs. Table 4 shows the test accuracy and the training time of the ResNet18 model on the ImageNet dataset for 350 epochs. From the results, it is evident that GRAD-MATCHPB-WARM not only outperforms other baselines in terms of accuracy but is also more efficient in model training times. Furthermore, GLISTER and CRAIG could not be 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. Overall, we observe that GRAD-MATCH and its variants consistently outperform all baselines by achieving higher test accuracy and lower training times.

Energy Consumption Results						
			Energy consumption for training the Model(in KWH)			
		Budget(%)	5%	10%	20%	30%
Dataset	Model	Selection Strategy				
CIFAR10	ResNet18	FULL	0.5032	0.5032	0.5032	0.5032
		RANDOM	0.0592	0.0911	0.1281	0.18
		GLISTER	0.0693	0.1012	0.1392	0.1982
		GLISTER-WARM	0.0672	0.0990	0.1360	0.1932
		CRAIG	0.0832	0.1195	0.1499	0.2063
		CRAIG-WARM	0.0770	0.1118	0.1438	0.2043
		CRAIGPB	0.0709	0.1031	0.1384	0.2005
		CRAIGPB-WARM	0.0682	0.1023	0.1355	0.2016
		GRAD-MATCH	0.0734	0.1173	0.1501	0.2026
		GRAD-MATCH-WARM	0.0703	0.1083	0.1429	0.2004
		GRAD-MATCHPB	0.0670	0.1006	0.1378	0.1927
		GRAD-MATCHPB-WARM	0.0649	0.0978	0.1354	0.1912
CIFAR100	ResNet18	FULL	0.5051	0.5051	0.5051	0.5051
		RANDOM (Skyline for Energy Consumption)	0.0582	0.0851	0.1116	0.1910
		GLISTER	0.0674	0.0991	0.1454	0.2084
		GLISTER-WARM	0.0650	0.0940	0.1444	0.2018
		CRAIG	0.1146	0.1294	0.1795	0.2378
		CRAIG-WARM	0.0895	0.1209	0.1651	0.2306
		CRAIGPB	0.0747	0.0946	0.1443	0.2053
		CRAIGPB-WARM	0.0710	0.0916	0.1447	0.2039
		GRAD-MATCH	0.0721	0.1129	0.1577	0.2297
		GRAD-MATCH-WARM	0.0688	0.0980	0.1531	0.2125
		GRAD-MATCHPB	0.0672	0.0978	0.1477	0.2100
		GRAD-MATCHPB-WARM	0.0649	0.0928	0.1411	0.2001

Table 5: Energy consumptions results for training a ResNet18 model on CIFAR10, CIFAR100 datasets for 300 epochs

Energy Consumption Results: Table 5 shows the energy consumption (in KWH) for different subset sizes of CIFAR10, CIFAR100 datasets. The results show that GRAD-MATCHPB-WARM strategy is the most efficient in energy consumption out of all other selection strategies. Similarly, we could also observe that the PerBatch variants, i.e., CRAIGPB, GRAD-MATCHPB have better energy efficiency compared to CRAIG, GRAD-MATCH respectively.

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 half. We present results on MNIST and CIFAR in sub-figures (f, g) in Figure 3. The results show that GRAD-MATCH and its variants outperform other baselines in all cases with the exception of 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.

Ablation study for R, per-batch, warm-start and κ: Next, we study the effect of varying R values on the performance of GRAD-MATCH and its variants. We study the result on CIFAR100 dataset at 5% subset for varying values of R (5,10,20) in the sub-figure (a) of Figure 4. The first takeaway is

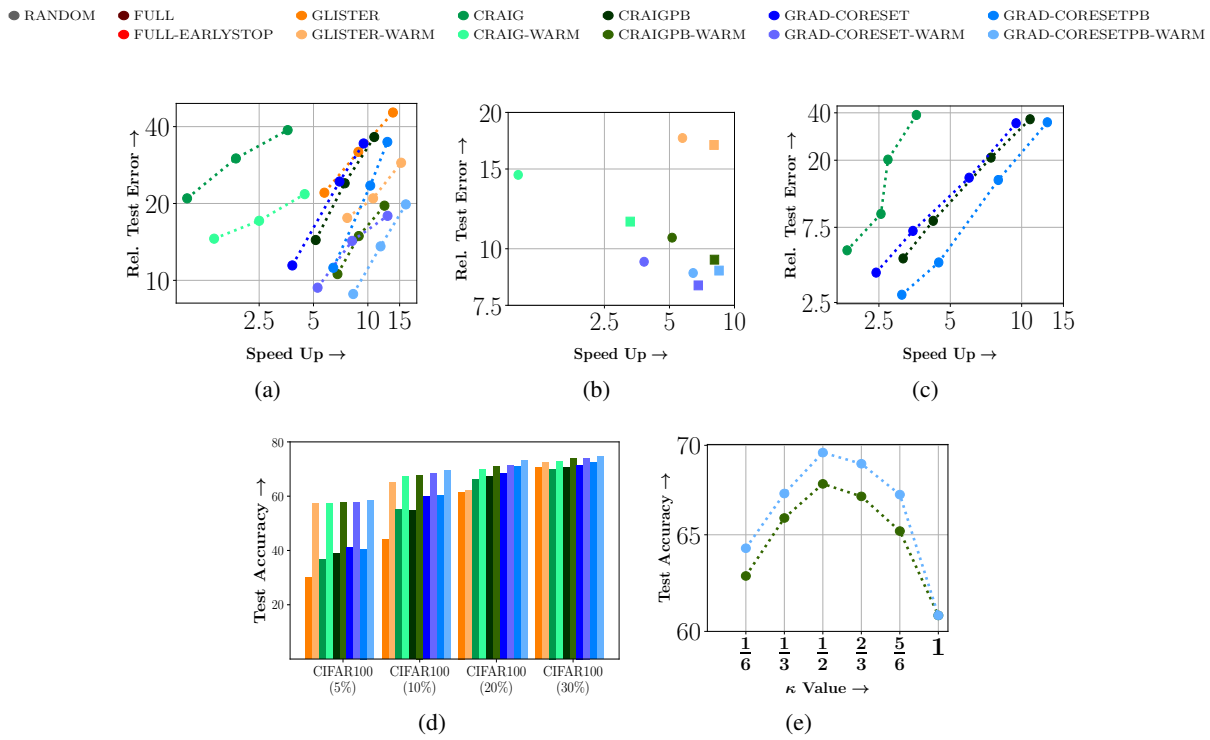

Figure 4: Sub-figure (a) compares the effect of varying R (5, 10 and 20) for different strategies (5% CIFAR-100). Sub-figure (b) compares $R = 5$ with a 5% subset (circles) and $R = 20$ with 10% subset (squares) showing that the latter is more efficient and accurate. Sub-figure (c) compares the per-batch (PB) versions with non-PB versions showing that the former has 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, and sub-figure (e) shows the effect of the warm-start parameter κ for CIFAR-100.

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-offs with different values of R. From sub-figure (b) of Figure 4, 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 the lesser number of subset selection runs. We then compare the PB variants of CRAIG and GRAD-MATCH with their non-PB variants (sub-figure (c) of Figure 4). We see that the PB versions are more efficient and lie consistently to the bottom right (*i.e.*, lesser relative test accuracy and higher speedups) than non-PB counter-parts. 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, with similar relative errors. Next, we study the effect of warm-start along with data selection. As shown in sub-figure (d) of Figure 4, 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. Sub-figure (e) of Figure 4 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.

A.3.5 STANDARD DEVIATION AND STATISTICAL SIGNIFICANCE RESULTS:

Table 7 shows the standard deviation results over five training runs on CIFAR10, CIFAR100, and MNIST datasets. The results show that the GRAD-MATCHPB-WARM has the least standard deviation compared to other subset selection strategies. Note that the standard deviation of subset selection strategies is large for smaller subsets across different selection strategies. Furthermore, GLISTER

	Random	Glister	Glister-Warm	Craig	Craig-Warm	CraigPB	CraigPB-Warm	Grad-Match	Grad-Match-Warm	Grad-MatchPB	Grad-MatchPB-Warm
Random											
Glister	0.0006										
Glister-Warm	0.0002	0.0017									
Craig	0.0003	0.048	0.00866								
Craig-Warm	0.0002	0.0375	0.0492	0.0004							
CraigPB	0.0002	0.0334	0.0403	0.0010	0.0139						
CraigPB-Warm	0.0002	0.003	0.017	0.0002	0.0005	0.0002					
Grad-Match	0.0002	0.028	0.031918	0.0030	0.0107	0.0254	0.015				
Grad-Match-Warm	0.0002	0.0008	0.0018	0.0008	0.0057	0.0065	0.0117	0.0005			
Grad-MatchPB	0.0002	0.0048	0.0067	0.0002	0.0305	0.0002	0.0075	0.0248	0.0305		
Grad-MatchPB-Warm	0.0002	0.00007	0.0028	0.0002	0.0002	0.0002	0.0011	0.0005	0.0091	0.0002	

Table 6: Pairwise significance p-values using Wilcoxon signed rank test

Standard Deviation Results						
			Standard deviation of the Model(for 5 runs)			
		Budget(%)	5%	10%	20%	30%
Dataset	Model	Selection Strategy				
CIFAR10	ResNet18	Full	0.032	0.032	0.032	0.032
		Random	0.483	0.518	0.524	0.538
		Glister	0.453	0.107	0.046	0.345
		Glister-Warm	0.325	0.086	0.135	0.129
		Craig	0.289	0.2657	0.1894	0.1647
		Craig-Warm	0.123	0.1185	0.1058	0.1051
		CraigPB	0.152	0.1021	0.086	0.064
		CraigPB-Warm	0.0681	0.061	0.0623	0.0676
		Grad-Match	0.192	0.123	0.112	0.1023
		Grad-Match-Warm	0.1013	0.1032	0.091	0.1034
		Grad-MatchPB	0.0581	0.0571	0.0542	0.0584
		Grad-MatchPB-Warm	0.0542	0.0512	0.0671	0.0581
CIFAR100	ResNet18	Full	0.051	0.051	0.051	0.051
		Random	0.659	0.584	0.671	0.635
		Glister	0.463	0.15	0.061	0.541
		Glister-Warm	0.375	0.083	0.121	0.294
		Craig	0.3214	0.214	0.195	0.187
		Craig-Warm	0.18	0.132	0.125	0.115
		CraigPB	0.12	0.134	0.123	0.115
		CraigPB-Warm	0.1176	0.1152	0.1128	0.111
		Grad-Match	0.285	0.176	0.165	0.156
		Grad-Match-Warm	0.140	0.134	0.142	0.156
		Grad-MatchPB	0.104	0.111	0.105	0.097
		Grad-MatchPB-Warm	0.093	0.101	0.100	0.098
			Standard deviation of the Model(for 5 runs)			
		Budget(%)	1%	3%	5%	10%
MNIST	LeNet	Full	0.012	0.012	0.012	0.012
		Random	0.215	0.265	0.224	0.213
		Glister	0.256	0.218	0.145	0.128
		Glister-Warm	0.128	0.134	0.119	0.124
		Craig	0.186	0.178	0.162	0.125
		Craig-Warm	0.0213	0.0223	0.0196	0.0198
		CraigPB	0.021	0.0209	0.0216	0.0204
		CraigPB-Warm	0.023	0.0192	0.0212	0.0184
		Grad-Match	0.156	0.128	0.135	0.12
		Grad-Match-Warm	0.087	0.084	0.0896	0.0815
		Grad-MatchPB	0.0181	0.0163	0.0147	0.0129
		Grad-MatchPB-Warm	0.0098	0.012	0.0096	0.0092

Table 7: Standard deviation results for CIFAR10, CIFAR100 and MNIST datasets for 5 runs

has higher standard deviation values than random for smaller subsets, which partly explains the fact that it does not work as well for very small subsets (e.g. 1% - 5%). We could also observe that the warm start variants of subset selection strategies have lower variance than non-warm-start ones from the standard deviation numbers, partly because of the better initialization they offer. Finally,

the PerBatch variants GRAD-MATCHPB and CRAIGPB have lower standard deviation compared to GRAD-MATCH and CRAIG which proves the effectiveness of Per-Batch approximation.

In Table 6, we show the p-values of one-tailed Wilcoxon signed-rank test Wilcoxon (1992) performed on every single possible pair of data selection strategies to determine whether there is a significant statistical difference between the strategies in each pair, across all datasets. Our null hypothesis is that there is no difference between the data selection strategies pair. From the results, it is evident that GRAD-MATCHPB-WARM variant significantly outperforms other baselines at $p < 0.01$.

A.3.6 OTHER RESULTS:

Gradient Errors: Table 8 shows the average gradient error obtained by various subset selection algorithms for the MNIST dataset. We observe that the gradient error of GRAD-MATCHPB is the smallest, followed closely by CRAIGPB. From the results, it is also evident that the PerBatch variants i.e., GRAD-MATCHPB and CRAIGPB achieves lower gradient error compared to GRAD-MATCH and CRAIG. Also note that GRAD-MATCH has a lower gradient error compared to CRAIG and GRAD-MATCHPB has a lower gradient error compared to CRAIG-PB. This is expected since GRAD-MATCH directly optimizes the gradient error while CRAIG minimizes an upper bound. Also note that GLISTER has a signifcantly larger gradient error at 1% subset which partially explains the reason for bad performance of GLISTER for very small percentages.

Redundant Points: Table 9 shows the redundant points, i.e., data points that were never used for training for various subset selection algorithms on the MNIST dataset. The results give us an idea of information redundancy in the MNIST dataset while simultaneously showing that we can achieve similar performances to full training using a much smaller informative subset of the MNIST dataset.

MNIST Gradient Error Results							
			Avg.Gradient error norm				
		Budget(%)	1%	3%	5%	10%	30%
Dataset	Model	Selection Strategy					
MNIST	LeNet	RANDOM	410.1258	18.135	10.515	9.5214	6.415
		CRAIG	68.3288	19.2665	10.9991	6.5159	0.3793
		CRAIGPB	17.6352	2.9641	1.3916	0.4417	0.0825
		GLISTER	545.2769	7.9193	1.8786	2.8121	0.3249
		GRAD-MATCH	66.2003	17.6965	9.8202	2.1122	0.3797
		GRAD-MATCHPB	15.5273	2.202	1.1684	0.3793	0.0587

Table 8: Gradient approximation relative to full training gradient for various data selection strategies for different subset sizes of MNIST dataset

MNIST Redundant Points Results							
			Percentage of Redundant Points in MNIST training data				
		Budget(%)	1%	3%	5%	10%	30%
Dataset	Model	Selection Strategy					
MNIST	LeNet	CRAIG	90.381481	74.057407	60.492593	36.788889	14.425926
		CRAIGPB	90.405556	73.653704	60.327778	35.301852	2.875926
		GLISTER	90.712963	77.540741	67.544444	45.940741	7.774074
		GRAD-MATCH	91.124074	76.4	62.109259	36.114815	2.942593
		GRAD-MATCHPB	90.187037	73.468519	59.757407	36.164815	6.751852

Table 9: Redunant points(i.e., points never used for training) for various data selection strategies for different subset sizes of MNIST dataset

Comparison between variants of GRAD-MATCH: Table 10 shows the test accuracy and the training time for PerClass, PerClassPerGradient and PerBatch variants of GRAD-MATCH using ResNet18 model on different subsets of CIFAR10 and CIFAR100 datasets. First, note that even though the PerClass variant achieves higher accuracy than the PerClassPerGradient variant, it is significantly slower, having a larger training time than full data training for the 30% subset of CIFAR10 and CIFAR100. Since the PerClass variant of GRAD-MATCH is not scalable, we use the PerClassPerGradient variant, which achieves comparable accuracies while being much faster. Finally, note that the PerBatch variants performed better than the other variants in test accuracy and training efficiency.

Comparison with additional subset selection methods: In addition to the baselines we considered so far, we compare GRAD-MATCH with additional existing subset selection strategies like

Comparison between variants of GRAD-MATCH										
			Top-1 Test accuracy(%)				Model Training time(in hrs)			
		Budget(%)	5%	10%	20%	30%	5%	10%	20%	30%
Dataset	Model	GRAD-MATCH Variant								
CIFAR100	ResNet18	PerClassPerGradient	41.01	59.88	68.25	71.5	0.5143	0.8114	1.40	2.002
		PerClass	41.57	59.95	70.87	72.45	0.5357	1.225	1.907	3.796
		PerBatch	40.53	60.39	70.88	72.57	0.3797	0.6115	1.09	1.56
CIFAR10	ResNet18	PerClassPerGradient	86.7	90.9	91.67	91.89	0.40	0.84	1.42	1.52
		PerClass	85.12	91.04	92.12	93.69	0.4225	1.042	1.92	3.48
		PerBatch	85.4	90.01	93.34	93.75	0.36	0.69	1.09	1.38

Table 10: Top-1 test accuracy(%) and training times for variants of GRAD-MATCH for different subset sizes of CIFAR10, CIFAR100 datasets

Additional Data Selection Results			
			Top-1 Test accuracy(%)
		Budget(%)	30%
Dataset	Model	Selection strategy	
CIFAR100	ResNet164	Facility Location	91.1
		Forgetting Events	92.3
		Entropy	90.4
	ResNet18	GRAD-MATCHPB-WARM	94.17
CIFAR10	ResNet164	Facility Location	64.8
		Forgetting Events	63.4
		Entropy	60.4
	ResNet18	GRAD-MATCHPB-WARM	74.62

Table 11: Top-1 test accuracy(%) and training times for additional data selection strategies on 30% CIFAR10 and CIFAR100 subset

Facility Location Wolf (2011), Entropy Settles (2012) and Forgetting Events Toneva et al. (2019) on CIFAR10 and CIFAR100 datasets. The results are in Table 11. Note that we used the numbers reported in paper Coleman et al. (2020) for comparison. The authors in Coleman et al. (2020) used a ResNet-164 Model which is a higher complexity model compared to ResNet-18 which we use in our experiments. Even after using a lower complexity model (ResNet-18), we outperform these other baselines on both CIFAR-10 and CIFAR-100. Furthermore, we achieve this while being much faster (since we observed that the ResNet-164 model is roughly 4x slower compared to ResNet-18). Even though a much smaller model (ResNet-20) is used for data selection, the training is still done with the ResNet-164 model. Finally, note that the selection via proxy method is orthogonal to GRAD-MATCH and can also be applied to GRAD-MATCH to achieve further speedups. We expect that the accuracy of these baselines (Forgetting Events, Facility Location, and Entropy) to be even lower if they are used with a ResNet-18 model. The accuracy reported for these baselines (Table 11) are the best among the different proxy models used in Coleman et al. (2020).