Faster Training by Automatically Selecting the Best Training Data for Computer Vision Tasks

Tony Cai
Stanford University
tonycai@stanford.edu

Dennis Duan
Stanford University
djd@stanford.edu

Ananth Agarwal
Stanford University
ananthag@stanford.edu

Abstract

*The appetite for massive datasets has grown as deep learning networks have exploded in size. Yet while such datasets are valued for the robustness they can provide a model, their size often results in time, resource, and financial costs that are challenging for researchers and developers with limited research budget. Moreover, existing strategies to train on subsets of training data only rely on information learned within a single training run. However, as models are typically trained multiple times in practice (*i.e.*, for hyperparameter search), we explored heuristics that use statistics from previous training runs to speed up training and found that adaptive data selection per training run may not be required to obtain significant speedup in training an image classification model. However, when applying similar techniques to the object detection task, we found that simply biasing training towards higher-loss data for object detection did not offer the same benefit, which suggests that alternative heuristics may be needed to improve training efficiency for more complicated computer vision tasks.*

1. Introduction

Training better deep neural networks often revolves around increasing the amount of training data used. But is all this data really needed for effective learning? Training on subsets of the training set without degrading the model performance would reduce the time and cost of training and potentially allow for better training in resource-constrained environments such as mobile devices. However, determining how many and which examples to trim during training remains a difficult task.

Current approaches to this problem generally rely only on information learned within the confines of a single training run to select subsets for training within that run; we define these as "intra-run" approaches. For example, Jiang *et al*. propose SELECTIVE-BACKPROP, which biases training to spend more time on high-loss, harder examples than low-loss, easier ones, and demonstrate that it provides significant speedup in converging to a target error rate for image classification [4]. While intra-run approaches have produced encouraging results in training efficiency, in most practical settings, training is never done once. For instance, hyperparameter and model search are important parts of designing effective models, both of which involve multiple training runs on the same data. A natural question to ask is whether training statistics from prior training runs can help train models more effectively and reduce the cost of training good models. We make two primary contributions to answer this question for image classification: (1) We attempt to reproduce the training speedups reported by the authors of the adaptive data subset selection algorithm GRAD-MATCH [6] by evaluating it against a new baseline, and (2) We define a new algorithm called CACHED-GRAD-MATCH that takes as input data subsets chosen by GRAD-MATCH in previous training runs and uses them to train an image classification model.

While numerous previous works have researched intra-run methods for image classification, we extended this to the more challenging computer vision task of object detection. Object detection classifies multiple object instances and generates bounding boxes. A key optimization difference between object detection and image classification is the object detection objective combines bounding box classification and localization loss. Our contribution is to evaluate if the speedups without performance compromise afforded by SELECTIVE-BACKPROP translate to object detection on the popular MS COCO dataset [10], where we have regression loss in addition to the base classification loss that SELECTIVE-BACKPROP has been previously tested on.

2. Related Work

Adaptive data subset selection is a technique in which periodically during training, a new subset of the data is chosen to train and update the model. The subset selection is "adaptive" because the chosen subset can update as the model parameters are updated. GRAD-MATCH is an

adaptive data subset selection algorithm where subsets are chosen that minimize the difference between the gradient of the subset and that of the full data [6]. The authors demonstrate that GRAD-MATCH achieves the best speedup-accuracy tradeoff (*i.e.*, highest speedup, lowest relative test error) compared to other state-of-the-art subset selection algorithms (CRAIG [12], GLISTER [7]), random subsets, and full training with early stopping for image classification. They do not explore potential information that can be incorporated from previous training runs, but rather they rely solely on gradients computed in previous epochs of a single run. Toneva *et al.* showed that in many common datasets, certain examples were consistently "forgotten" (*i.e.*, correctly classified earlier in training but misclassified later on) by a variety of models, and that this could be used to prune datasets by only including frequently-forgotten examples [19]. The authors empirically demonstrated that it's possible to remove 30% of the CIFAR dataset [9] and still obtain the same model performance. While this does use information from past training, it relies exclusively on "forgetting" events as a metric for identifying unimportant samples. Since models are often trained multiple times, we aim to explore whether information from previous runs can improve efficiency of training later runs.

Segmenting training datasets into more-useful and less-useful data is a common thread of research into more efficient deep learning. Paul *et al.* identified a "Data Diet" that uses individual initial loss gradient norms to identify data that can be pruned early in training without reducing accuracy on CIFAR-10 [14]. Shrivastava *et al.* proposed online hard example mining to improve object detection performance and training efficiency [18]. Jiang *et al.*'s SELECTIVE-BACKPROP achieves a speedup of up to 3.5x over standard Stochastic Gradient Descent to a target error for image classification by using importance sampling to train on high-loss examples [4]. Once enough forward passes have been done such that a batch of appropriate size is created, the loss from these selected examples is backpropagated. One limitation with implementing this approach is that excluding training examples for backwards passes is difficult in PyTorch and TensorFlow, so the paper had to use a work-around. MosaicML proposes a solution that adds an extra forward pass for each batch to compute losses for sampling [20]. After selecting the training examples, this subset is used for standard forward and backward passes, thus completing the iteration for this batch. Using this approach, MosaicML report an improvement in training time and actually, a slight improvement in validation accuracy on CIFAR-10.

3. Methods

We discuss two existing intra-run data selection algorithms, namely GRAD-MATCH and SELECTIVE-BACKPROP, and propose CACHED-GRAD-MATCH, a new variant of GRAD-MATCH that utilizes data subset selection history from previous training runs to inform new training runs.

3.1. Data Selection

GRAD-MATCH. GRAD-MATCH operates as follows: at epoch t, given a budget of k examples, an adaptive subset selection algorithm outputs a pair of length-k vectors, $(\mathcal{X}_t, w_t)$, where $\mathcal{X}_t$ contains the indices of the examples to train with, and w_t contains weights corresponding to the examples in $\mathcal{X}_t$. Then, at epoch t, the model is trained only on the examples in $\mathcal{X}_t$, and model updates are applied using the weighted losses of those examples, with the loss weights specified by w_t. Note that while this formulation allows subset selection to occur in every epoch, GRAD-MATCH performs this selection every R epochs and uses the chosen subset and weight for the subsequent R epochs [6].

In our work, we leverage GRAD-MATCH as a baseline subset selection algorithm. Intuitively, GRAD-MATCH attempts to choose a subset of examples with at most k elements that most closely approximate the loss gradient of the full training set by minimizing the loss gradient error, defined below:

$$Err = \Big|\Big| \sum_{i \in \mathcal{X}_t} w_t(i) \nabla_\theta L_i(\theta_t) - \nabla_\theta L(\theta_t) \Big|\Big| \qquad (1)$$

where θ_t is the model's updatable parameters at epoch t, L_i is the loss from example i, L is the total loss over all examples, and the norm is the L^2 norm. To accomplish this, GRAD-MATCH uses an orthogonal matching pursuit based algorithm [6]. While GRAD-MATCH provides significant training speedups when training with much smaller datasets, each subset selection run requires non-trivial computation time.

To further improve training time, the GRAD-MATCH authors also considered a warm-start variant, which trains on the full dataset for T_f epochs before doing data subset selection. This may also be applied to the CACHED-GRAD-MATCH method described below. We also experimented with the warm-start variant where $T_f = 20$.

CACHED-GRAD-MATCH. To explore the idea of informing subset selection by incorporating information from previous runs, we propose a simple algorithm, CACHED-GRAD-MATCH, in which $(\mathcal{X}_t, w_t)$ pairs are saved from an initial training run using GRAD-MATCH and replayed in subsequent training runs. Specifically, if the pair $(\mathcal{X}_i, w_i)$ was used at epochs $[t, t+R)$ in the initial training run, subsequent training runs will use $(\mathcal{X}_i, w_i)$ at epochs $[t, t+R)$ as well. While this is limited to training models on the same

dataset and the same R as the original GRAD-MATCH training run, this algorithm drastically reduces the computation time needed for subset selection, further improving training time over GRAD-MATCH. In Sec. 5, we evaluate its performance when training models with different hyperparameters and architectures.

RANDOMPB. To validate previous results from GRAD-MATCH, we also run baseline training runs with full data and with random subset selection (in which examples are randomly chosen without replacement to fill the subset budget) as baselines, as is done in the GRAD-MATCH paper [6]. However, we also evaluate a *per-batch* variant of random selection, RANDOMPB, in which the dataset is divided into fixed batches, and batches (rather than examples) are chosen at random to fill the budget. A per-batch variant of GRAD-MATCH, GRAD-MATCHPB, showed improved results over GRAD-MATCH [6], but a per-batch random subset selection algorithm was not provided as a baseline.

SELECTIVE-BACKPROP Similar to Stochastic Gradient Descent, SELECTIVE-BACKPROP traverses the training batches once per epoch, but it only uses a subset of examples per batch in the gradient update stage. It uses non-uniform sampling to ensure that gradients from high-loss examples are prioritized in model updates. Specifically, the subset selection algorithm $P_{select}(\text{losses})$ samples s examples without replacement per training batch of size b_{size}, where the i^{th} example is selected with probability $\frac{L_i}{L_{batch}}$.

For our implementation, we use MosaicML's approach of using two forward passes (see Sec. 2). Additionally, they define a `interrupt` hyperparameter that inserts a standard training batch every `interrupt` SELECTIVE-BACKPROP training batches. SELECTIVE-BACKPROP is enabled from epoch `sb_start` to epoch `sb_end`, and the model is trained on the full training dataset in the remaining epochs. This has been found to improve the speed-accuracy tradeoff [20]. The full algorithm we utilize is summarized in Algorithm 1.

Algorithm 1 SELECTIVE-BACKPROP for Object Detection

```
for epoch in range(epochs) do
    for i, X_{b_size} in enumerate(data_loader) do
        if epoch in [sb_start, sb_end) then
            losses = forward(X_{b_size})
            if i % interrupt == 0 then
                backward(losses)
            else
                X_s = P_select(losses)
                backward(forward(X_s))
            end if
        else
            backward(forward(X_{b_size}))
        end if
    end for
end for
```

3.2. Image Classification

Since both Jiang *et al*. [4] and MosaicML [20] have reported SELECTIVE-BACKPROP speedups for image classification, we instead focus on applying the GRAD-MATCH, CACHED-GRAD-MATCH, and RANDOMPB algorithms on various image classification models and configurations. We used ResNet [3] and MobileNetV2 [17] as the baseline image classification models.

3.3. Object Detection

The Single Shot MultiBox Detector (SSD) object detection model proposed by Liu *et al*. defines 8,732 default boxes of different aspect ratios and scales across different resolution feature maps output by convolutional layers stacked on top of a backbone CNN [11]. To train the model, $N \geq 1$ default boxes are matched to each ground truth box g containing an instance of class c^p. The loss function for an example image x with class categories c with predicted box locations l for the N default boxes is as follows:

$$L_{obj}(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \quad (2)$$

where L_{conf} is the classification softmax loss for c_p against the other classes in c, and L_{loc} is the Smooth L1 [2] regression loss measuring the difference between the default boxes predicted locations and the ground truth, and α is a hyperparameter that weighs the two losses. Liu *et al*. highlight that SSD is a faster detector than YOLO [15] and two-stage detectors like Faster R-CNN [16], and the SSD training procedure utilizes hard negative mining [11]. We use this SSD model as a baseline for our object detection speedup experiments.

To apply SELECTIVE-BACKPROP to object detection, we use $L_{total,i} = \sum_g L_{obj,i}$ (the total loss across all ground truth boxes g for each image i) to weigh each training example and sample across them within a batch of size b_{size}: $P_{select}\{L_{total,i} \ldots L_{total,b_{size}}\}$. Our implementation of MosaicML's recommended approach for SELECTIVE-BACKPROP is illustrated in Algorithm 1. Training SSD with this approach will be evaluated against a baseline without SELECTIVE-BACKPROP to evaluate if SELECTIVE-BACKPROP offers a speedup.

Nvidia provides a PyTorch implementation of Liu *et al*. 's original model, SSD300, which replaces the original backbone with ResNet [3] and uses input size of 300x300 [13]. For inference, the Nvidia library integrates with COCO's python library that calculates validation metrics given predicted labels and locations. The primary

accuracy measurement is mean average precision, mAP. For each of the labels in COCO, across all validation images the area under the Precision-Recall curve is calculated and then averaged across all labels, yielding a final mAP. Positive/negative detections are contingent on defining a intersection-over-union (IoU) threshold. COCO's primary evaluation metric is the average of mAP for each of 0.50 to 0.95 with step size 0.05, over all object sizes. We report three additional metrics for mAP [0.50:0.95] for small (most difficult), medium, and large objects.

Applying GRAD-MATCH and CACHED-GRAD-MATCH to object detection is more complicated, since SSD doesn't have a single linear embedding layer (as opposed to ResNet for image classification) for prediction. Rather, SSD predicts class label and bounding box correction for objects of different sizes, using multiple classifiers corresponding to feature maps of different resolutions. Therefore, it may be more effective to concatenate gradients of losses from every SSD classifier, and run GRAD-MATCH's orthogonal matching pursuit algorithm to obtain optimal data subsets. Unfortunately, due to time constraints, we were unable to integrate SSD with GRAD-MATCH because we ran into technical issues integrating the gradients from SSD with GRAD-MATCH.

4. Dataset and Features

For image classification, we used the CIFAR-10 dataset [9]. To be consistent with the GRAD-MATCH baseline [6], we normalized the image RGB values across each RGB channel. While CIFAR-10 is a simple problem, prior work on data subset selection [19] has demonstrated that up to 30% of data can be removed without affecting the model's generalizability, so it make sense as a starting point to evaluate data subset selection algorithms.

Object detection was evaluated on MS COCO [10]. Due to compute constraints, we sampled half of the training and validation data, for a total of 58.6k training images and 2.5k validation images. We implemented this sampling on top of MosaicML's PyTorch `DataLoader` implementation, which also provides the performance-improving data augmentation approaches from the original SSD paper: for each image, either use the original image, a random patch, or a patch with a minimum IoU overlap with ground truth objects [11]. Patches are resized and may be horizontally flipped.

5. Experiments

To ensure consistency, all experiments were conducted on an AWS EC2 `g4dn.xlarge` instance with an Nvidia Tesla T4 GPU.

5.1. Image Classification

For image classification, we only evaluated GRAD-MATCH and its variants, since MosaicML already evaluated the effectiveness of SELECTIVE-BACKPROP on image classification (they observed that SelectiveBackprop reduced training time by approximately 10%). We first evaluated CACHED-GRAD-MATCH on different hyperparameter setups by collecting one set of initial $(\mathcal{X}_t, w_t)$ pairs from training ResNet18 on CIFAR10 using GRAD-MATCH as a backbone, and using that set on training runs where we varied training hyperparameters. This allowed us to explore the efficacy of CACHED-GRAD-MATCH in an environment mimicking hyperparameter search, where it would be infeasible to collect multiple training histories. We also evaluated the efficacy of CACHED-GRAD-MATCH when transferring across model architectures by collecting initial $(\mathcal{X}_t, w_t)$ pairs from training MobileNetV2 [17] on CIFAR10 and applying those pairs to train different models on the same dataset.

For these experiments, we extended code [5] from the authors of GRAD-MATCH to include CACHED-GRAD-MATCH and RANDOMPB. Models were trained for 300 epochs each, and for training runs involving data selection, the subset budget was set to 10% of the training dataset. Unless otherwise specified, training was done using an SGD optimizer, 0.01 learning rate with 0.9 momentum, and cosine annealing, and other PyTorch default hyperparameters. In each training run for CACHED-GRAD-MATCH, we also performed training runs using the same model, hyperparameters, and dataset using GRAD-MATCH, RANDOMPB, and full training to serve as baseline comparisons.

5.2. Object Detection

We evaluated three configurations for training SSD with a ResNet50 backbone: a baseline without SELECTIVE-BACKPROP, SELECTIVE-BACKPROP with `interrupt` $=$ 2, and SELECTIVE-BACKPROP with `interrupt` $= 3$ (see Algorithm 1). All other hyperparameters were fixed for the three configurations; due to compute constraints, we were not able to run hyperparameter sweeps. We followed MosaicML's recommendations of starting SELECTIVE-BACKPROP halfway through training (`sb_start`), ending it once 90% has completed (`sb_end`), and keeping half the examples $s = \frac{b_{size}}{2}$ in SELECTIVE-BACKPROP iterations [20]. We evaluate the three configurations on mAP across different IoU thresholds and total training time. For overall training, we used the default hyperparameters published by Nvidia in their SSD repository, namely SGD with momentum, and weight decay on a subset of parameters [13]. The base learning rate is 2.6e^{-3}, and is warmed up within the first epoch, and decayed $\frac{2}{3}$ and $\frac{5}{6}$ of the way through training. To fit within our resource limits, we use batch size b_{size} of 32 and train for 30 epochs.

6. Results

6.1. Image Classification

For each experiment, we provide the following results:

1. **Speedup-accuracy tradeoffs (*e.g.*, Fig. 1)**: For each experiment setup, we consider the relative speedup and relative test errors achieved from training a model using CACHED-GRAD-MATCH as compared to using the full dataset. We provide the speedup and test error at various points in training, each relative to the final training time and final test error collected from our corresponding full training run using the same hyperparameters. Each curve corresponds to results collected from one training run, and points along the curve represent data collected every 20 epochs between epochs 200 and 300.

2. **Fixed-accuracy speedups over full training (*e.g.*, Tab. 1)**: For each experiment setup, we also consider the relative speedup of CACHED-GRAD-MATCH when compared to the time at which the corresponding full training baseline surpassed the final CACHED-GRAD-MATCH test accuracy.

Learning rate variations. We collected $(\mathcal{X}_t, w_t)$ pairs from GRAD-MATCH training with learning rate 0.01 and used these to perform CACHED-GRAD-MATCH training runs with learning rates of 0.001, 0.003, and 0.03, keeping all else fixed. For each learning rate, we also generated baselines with GRAD-MATCH, RANDOMPB, and full training. Results are shown in Fig. 1 and Tab. 1.

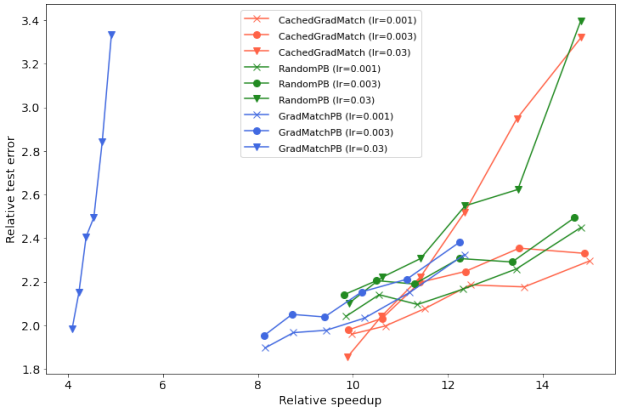

Figure 1. Speedup-accuracy tradeoffs of subset selection algorithms using different learning rates.

Optimizer variations. We collected $(\mathcal{X}_t, w_t)$ pairs from GRAD-MATCH training with an SGD optimizer and used these to perform CACHED-GRAD-MATCH training runs using Adam and RMSProp optimizers, keeping all else fixed. For each optimizer, we also generated baselines with GRAD-MATCH, RANDOMPB, and full training. Results are shown in Fig. 2 and Tab. 2.

	Speedup		
Learning rate	0.001	0.003	0.03
GRAD-MATCH	0.81	0.82	2.72
CACHED-GRAD-MATCH	**1.00**	**0.99**	**7.92**

Table 1. Fixed-accuracy speedups over full training using different learning rates.

	Speedup		
	Adam	RMSProp	SGD
GRAD-MATCH	**0.83**	0.81	3.07
CACHED-GRAD-MATCH	0.67	**0.97**	**4.68**

Table 2. Fixed-accuracy speedups over full training using different optimizers.

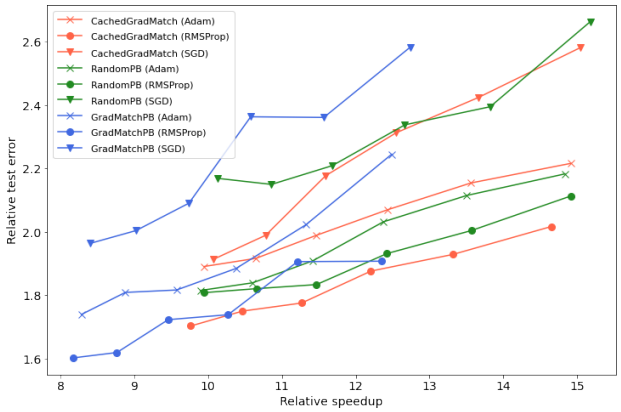

Figure 2. Speedup-accuracy tradeoffs of subset selection algorithms using different optimizers.

Architecture variations. We collected $(\mathcal{X}_t, w_t)$ pairs from training MobileNetV2 using GRAD-MATCH and used these to train ResNet18 and Vision Transformer model [1]. While we saw similar speedup in training the transformer model, we didn't have enough data (due to long training time) so it is excluded from the report.

	Speedup on ResNet
GRAD-MATCH	3.86
GRAD-MATCH-warm	**4.69**
CACHED-GRAD-MATCH	2.24
CACHED-GRAD-MATCH-warm	4.51

Table 3. Fixed-accuracy speedups over full training for different model architectures.

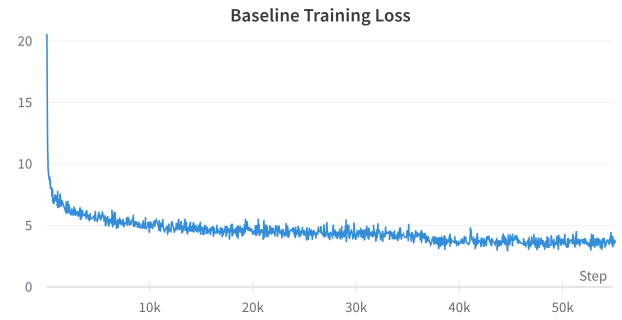

Figure 4. SSD Baseline Training Loss, 30 epochs

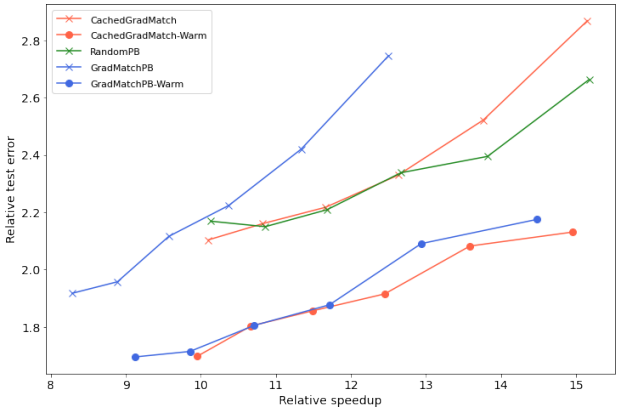

Figure 3. Speedup-accuracy tradeoffs of subset selection algorithms for different model architectures.

6.2. Object Detection

Fig. 4 and Fig. 5 contain plots of SSD training loss for the baseline and with SELECTIVE-BACKPROP. In Fig. 5, `sb_start` occurs at step 27.5k. While the baseline training loss in Fig. 4 is steady until the learning rate is annealed two-thirds of the way through training, at which point loss decreases and settles around 3.5, the interval that SELECTIVE-BACKPROP runs, `sb_start` - `sb_end`, is prominently marked by a noisy, small increase in loss. This shows the effect of Algorithm 1; since we are intentionally biasing towards selecting higher loss training examples within a batch, it is expected that the loss would go up. It is

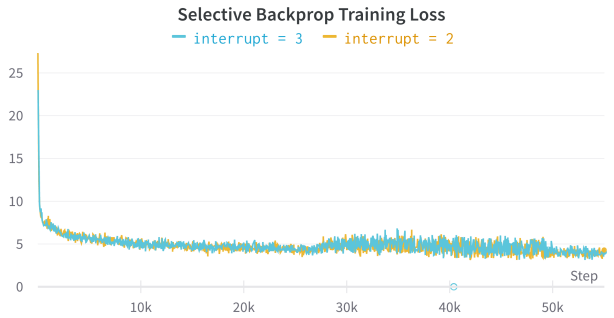

Figure 5. SSD SELECTIVE-BACKPROP Training Loss, 30 epochs

	mAP IoU 0.50:0.95				Time	
	All	Small	Med.	Large	Hours	t_{diff} (s)
Base	**9.68**	**2.65**	**10.85**	**15.48**	7.61	N/A
SB_2	9.09	2.47	9.71	14.8	**7.44**	41.07
SB_3	8.79	2.22	9.53	14.1	N/A[1]	**51.14**

Table 4. SSD Evaluation Metrics

also notable that since we are limited to original batch size of 32 and selecting half, the SELECTIVE-BACKPROP iterations have batch size 16. Smaller batch sizes typically result in more zigzag training loss behavior, even if we weren't trying to select high loss training data. Within Fig. 5, there is no noticeable difference in the behavior between the two `interrupt` values, and once SELECTIVE-BACKPROP is ended, the loss settles at a higher value than the baseline.

Tab. 4 shows mAP (as defined in Sec. 3.3) and training time performance. SELECTIVE-BACKPROP with `interrupt` $= 2$ is abbreviated SB_2, and likewise for SB_3. In addition to total training time, since over half the epochs in the SELECTIVE-BACKPROP runs were standard full-dataset training epochs, we evaluated the difference between the absolute runtime of epochs with and without SELECTIVE-BACKPROP enabled. Letting t_i be the runtime of an epoch, we define t_{diff} as the difference between the average standard epoch time and average SELECTIVE-BACKPROP-enabled epoch time:

$$t_{diff} = \underset{\substack{i \in [1,\texttt{epochs}] \\ i \notin [\texttt{sb\_start},\texttt{sb\_end})}}{\bar{t}_i} - \underset{\substack{j \in [1,\texttt{epochs}] \\ j \in [\texttt{sb\_start},\texttt{sb\_end})}}{\bar{t}_j} \quad (3)$$

Across all object area sizes, SELECTIVE-BACKPROP scores lower mAP than the baseline. Moreover, we can see that having more SELECTIVE-BACKPROP iterations within

[1]The AWS EC2 instance was stopped and restarted between running SB_3 and the baseline and SB_2, so the SB_3 absolute training time is not directly comparable to the other two due to potential GPU environment differences.

an epoch where it is enabled, which is the case when $\texttt{interrupt} = 3$ vs. 2, also reduces mAP. This indicates that the $\frac{b_{size}}{2}$ examples discarded on these iterations, especially when done in successive iterations, costs generalizability. We had hypothesized that since small object areas are harder than larger areas, training examples with small objects would be more likely to be selected by SELECTIVE-BACKPROP, and so it may even outperform the baseline for small object mAP, but this is not the case in our experiments. However, there is a clear speed-accuracy tradeoff. Despite the extra forward pass on $\frac{b_{size}}{2}$ examples required for each training iteration (line 9 of Algorithm 1), SELECTIVE-BACKPROP still trains faster. The t_{diff} values show that as the ratio of SELECTIVE-BACKPROP to standard training runs increases within a training run with SELECTIVE-BACKPROP enabled, the speedup increases.

7. Discussion

7.1. Image Classification

GRAD-MATCH. In general, GRAD-MATCH provided significant training speedups over the full training run, with modest drops in training accuracy. However, GRAD-MATCH seemed to perform worse in terms of fixed-accuracy speedups compared to full training when run with suboptimal hyperparameter choices. Specifically, our training runs with Adam and RMSProp optimizers showed worse fixed-accuracy speedups (Tab. 2) compared to full training, and in these runs we used a learning rate of 0.01, while the recommended is 0.001 [8]. One possible explanation is that GRAD-MATCH may be be more susceptible to noisier gradients due to its smaller effective dataset size, and larger learning rates would amplify these noisy gradients.

CACHED-GRAD-MATCH. CACHED-GRAD-MATCH also provided significant training speedups, and in many situations outperformed GRAD-MATCH. This is likely due to the fact that at selectdion time, CACHED-GRAD-MATCH only needs to read the previous $(\mathcal{X}_t, w_t)$ from memory while GRAD-MATCH relies on a more complex computation. The speedup-accuracy tradeoffs often favored CACHED-GRAD-MATCH over GRAD-MATCH (Figs. 1 and 2), where the frontiers for CACHED-GRAD-MATCH were more efficient (below and two the right, in the case of these graphs) than GRAD-MATCH. This is also validated by the fixed-accuracy speedup results, where CACHED-GRAD-MATCH mostly outperformed GRAD-MATCH in different hyperparameter scenarios (Tabs. 1 and 2). The architecture variations resulted in comparatively worse performance from CACHED-GRAD-MATCH, which makes sense as subset selections tailored to one model architecture's training may not transfer as well to training other models; however, we still see reasonable speedups comparable to GRAD-MATCH in these scenarios (Fig. 3).

RANDOMPB. Surprisingly, RANDOMPB seemed to perform comparably to GRAD-MATCH and nearly as well as CACHED-GRAD-MATCH in our experiments. This also helps provide intuition behind the strong performance of CACHED-GRAD-MATCH; if even random subset selection can provide reasonable speedups, then having some adaptively chosen subsets might also be expected to perform well, even if those subsets were chosen for a different training scenario.

Warm variants. In general, the warm variants of GRAD-MATCH and CACHED-GRAD-MATCH performed better than the base versions, which validates results from previous work [6], as seen in Fig. 3. This may result from the fact that a partially tuned model is more likely to have smaller gradients than a randomly initialized model, leading to better convergence when adopting smaller training subsets. However, more experimentation would be required to better understand this phenomenon.

Convergence. To match previous experiments [6], we ran all training runs for 300 epochs. However, in the cases of GRAD-MATCH, CACHED-GRAD-MATCH, and RANDOMPB, this was likely not long enough to allow the model to converge, as training loss was still decreasing when training stopped. This may be one cause for the relatively poor fixed-accuracy speedup results. Future work may involve longer training runs to allow all training runs to converge in order to provide a more detailed analysis.

Analysis of example choices. To better understand why CACHED-GRAD-MATCH seemed to transfer well to the other training scenarios we tested, we qualitatively examined the which images were given more weight in GRAD-MATCH training. To do this, we defined the w-score of an image i as the total normalized w for example i summed across all epochs. We observed that images with large w-scores seem noticeably different from each other, while images with low w-scores seem more similar to each other (Figs. 7 and 8 in Appendix A.1). This matches our intuition, since training the model on more varied examples should improve the model's test accuracy more than training on very similar examples, and this should generally hold regardless of the specific architecture and hyperparameter choices.

To further illustrate this, Fig. 6 shows the histogram of w-scores across training examples, which shows that high w-scores are concentrated on a small number of examples. We also examine the cosine similarity between weights selected by GRAD-MATCH when run on different model ar-

chitectures in Tab. 5. We can see that there is a noticeable overlap between the selected subsets despite noticeable changes in hyperparameters or model architectures, suggesting that some examples may be important regardless of the training scenario. This is consistent with previous findings [19, 21], which conjecture that neural network boundaries are defined by a small subset of examples.

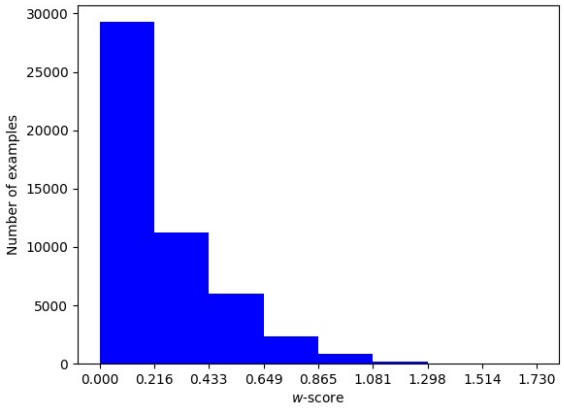

Figure 6. w-score histogram for GradMatch training on ResNet18

	Adam, ResNet	RMSProp, ResNet	RMSProp, MobileNet
Adam, ResNet	–	0.745	0.695
RMSProp, ResNet	–	–	0.687
RMSProp, MobileNet	–	–	–

Table 5. Cosine similarities of w-scores for different models

7.2. Object Detection

While we do observe a speedup from applying SELECTIVE-BACKPROP to object detection, since our experiment results show a non-trivial degradation in accuracy, we don't achieve the same benefits as Jiang *et al*. and MosaicML [20] do for image classification. In Appendix A.2, we highlight the stark effect that SELECTIVE-BACKPROP had on the weight distribution of a layer in the ResNet50 backbone of SSD. From this, we can see that Algorithm 1 has the intended effect of forcing re-weighting of parameters in accordance with trying to learn high-loss training data, but we do not observe the faster convergence to an optimal error that Jiang *et al*. do. With additional compute resources, we would run training for at least double the epochs, and with bigger batch sizes to potentially reduce

Figure 7. Images with low w-scores

the noise in the loss curves. We would also build a more complete picture of the speed-accuracy trade-off curve by running hyperparameter sweeps over `sb_start`, `sb_end`, and `interrupt`. For the high-loss examples, investigating the distribution of the loss between box classification and box regression could help identify if we perhaps need to have separate α (Eq. (2)) values when running each of the forward passes in Algorithm 1. Another direction we would pursue as future research is to try to adapt GRAD-MATCH to run on the SSD model, which would require us to carefully design how to integrate gradients across layers, since SSD has classification and regression heads.

8. Conclusion

We thoroughly investigated the effectiveness of GRAD-MATCH as an adaptive subset selection framework. While GRAD-MATCH may significantly speed up training, it also has the potential drawback where the model accuracy and convergence rate is more dependent on the model optimizer. We also proposed and evaluated a novel data selection approach which reuses prior training, and showed that significant performance gains may be obtained without incurring the extra cost of adaptive data selection. Additionally, we explored whether existing data selection strategy can effectively handle more challenging problems like object detection. We see that the benefit of data selection is noticeably less pronounced for object detection.

A. Appendix

A.1. Images selected by GRAD-MATCH

For our analysis we qualitatively compared 10 training examples, though we only show a small subset in this report (shown in Fig. 7 and Fig. 8).

Figure 8. Images with high w-scores

A.2. SELECTIVE-BACKPROP Parameter Visualization

Using the gradient and parameter logging provided by Weights & Biases, we were able to take a deeper look into how biasing towards high-loss training examples and discarding the others within a batch impacts learning. Fig. 9 and Fig. 10 present a stark difference between the baseline and SELECTIVE-BACKPROP (`interrupt` is 2) for one of the layers in the ResNet50 backbone in the SSD model. SELECTIVE-BACKPROP is enabled at step 27.5k and disabled at step 49.5k. The weights in the baseline in Fig. 9 generally become more concentrated around 0.22 as training progresses, evidenced by the dark blue near the end of training. In Fig. 10 though, the parameters gradually coalescing around 0.22 is wholly interrupted when SELECTIVE-BACKPROP is enabled. The weights end up much more distributed across the range [0.1, 0.4]. The new distribution of weights could be interpreted positively as an effort towards learning the harder examples in the dataset, or negatively as indicative that the larger magnitude gradients generated from backpropagating larger losses disturbed the model from convergence. Although Tab. 4 shows that the end result of SELECTIVE-BACKPROP is a small degradation in performance, it has a clear effect in the training process that can serve as a launching point for future experiments.

B. Contributions & Acknowledgements

Ananth worked primarily on design and implementation of the object detection experiments. He also participated in early research into GRAD-MATCH and related works on training efficiency. Tony worked on implementing CACHED-GRAD-MATCH, running experiments on GRAD-MATCH and CACHED-GRAD-MATCH in different scenarios, as well as empirical analysis of GRAD-MATCH and CACHED-GRAD-MATCH. Dennis worked on the design and evaluation of CACHED-GRAD-MATCH and RANDOMPB, as well as analysis of the image classification results. All other work was shared equally.

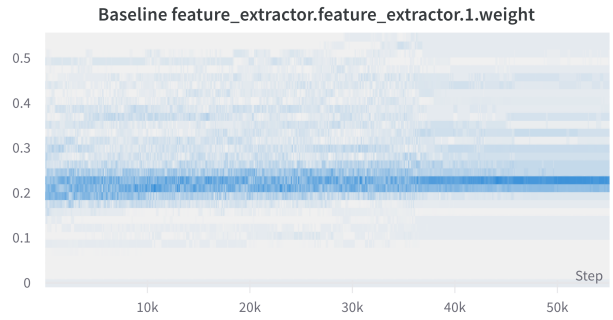

Figure 9. SSD Baseline: Parameter weights for a layer in the ResNet50 backbone

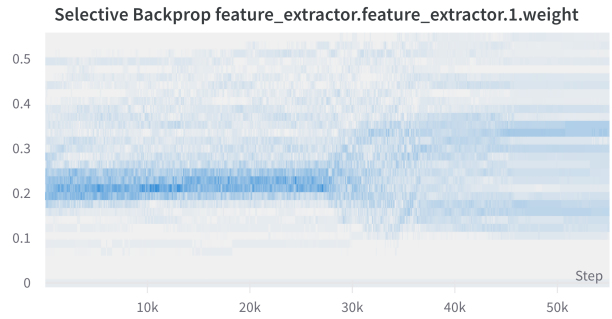

Figure 10. SSD SELECTIVE-BACKPROP: Parameter weights for a layer in the ResNet50 backbone

We thank Jonathan Frankle, Chief Scientist at MosaicML, for mentoring us throughout this project. He provided us with papers and code to begin our project and invaluable guidance on how to design our experiments.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 5

[2] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. 3

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 3

[4] Angela H Jiang, Daniel L-K Wong, Giulio Zhou, David G Andersen, Jeffrey Dean, Gregory R Ganger, Gauri Joshi, Michael Kaminksy, Michael Kozuch, Zachary C Lipton,

et al. Accelerating deep learning by focusing on the biggest losers. *arXiv preprint arXiv:1910.00762*, 2019. 1, 2, 3

[5] Krishnateja Killamsetty, Dheeraj Bhat, Ganesh Ramakrishnan, and Rishabh Iyer. CORDS: COResets and Data Subset selection for Efficient Learning, 3 2022. 4

[6] Krishnateja Killamsetty, S Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*, pages 5464–5474. PMLR, 2021. 1, 2, 3, 4, 7

[7] KrishnaTeja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh K. Iyer. GLISTER: generalization based data subset selection for efficient and robust learning. *CoRR*, abs/2012.10630, 2020. 2

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[9] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 2, 4

[10] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. 1, 4

[11] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015. 3, 4

[12] Baharan Mirzasoleiman, Jeff A. Bilmes, and Jure Leskovec. Data sketching for faster training of machine learning models. *CoRR*, abs/1906.01827, 2019. 2

[13] Nvidia. SSD300 v1.1 For PyTorch, 3 2019. 3, 4

[14] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 34, 2021. 2

[15] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. 3

[16] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. 3

[17] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 3, 4

[18] Abhinav Shrivastava, Abhinav Gupta, and Ross B. Girshick. Training region-based object detectors with online hard example mining. *CoRR*, abs/1604.03540, 2016. 2

[19] 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. *arXiv preprint arXiv:1812.05159*, 2018. 2, 4, 8

[20] Abhinav Venigalla. MosaicML: Selective Backprop, 2021. 2, 3, 4, 8

[21] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021. 8