Fixing Mini-batch Sequences with Hierarchical Robust Partitioning

Shengjie Wang[†], **Wenruo Bai**[*], **Chandrashekhar Lavania**[*], **Jeffrey A. Bilmes**[†*]
University of Washington, Seattle
Depts. of: [*]Electrical and Computer Engineering, [†]Computer Science and Engineering

Abstract

We propose a general and efficient hierarchical robust partitioning framework to generate a deterministic sequence of mini-batches, one that offers assurances of being high quality, unlike a randomly drawn sequence. We compare our deterministically generated mini-batch sequences to randomly generated sequences; we show that, on a variety of deep learning tasks, the deterministic sequences significantly beat the mean and worst case performance of the random sequences, and often outperforms the best of the random sequences. Our theoretical contributions include a new algorithm for the robust submodular partition problem subject to cardinality constraints (which is used to construct mini-batch sequences), and show in general that the algorithm is fast and has good theoretical guarantees; we also show a more efficient hierarchical variant of the algorithm with similar guarantees under mild assumptions.

1 INTRODUCTION

Stochastic mini-batch gradient methods achieve outstanding performance in practice without placing excessive demands on machine memory. Theoretically, such methods (Duchi et al., 2011; Li et al., 2014; Sutskever et al., 2013; Kingma and Ba, 2014) require a mini-batch of unbiased random samples of data for each gradient update step (Bottou, 2010), so that the gradient estimate is unbiased and a good convergence rate can be achieved.

While sampling independent random mini-batches is essential from a theoretical perspective, it intrinsically conflicts with the efficient use of computational learning systems. Training sets are getting larger (thereby driving accuracy higher) and they typically do not fit in cache or memory. The only feasible approach is to repeatedly load data from main memory and/or disk to form mini-batches, but doing so from a convergence rate perspective (i.e., randomly with replacement) is costly because caches do not help when memory access patterns are random and hence unpredictable.

In general, to achieve high computational efficiency, time spent loading an independent minibatch should occur simultaneously with computation on a previous minibatch, as in a pipeline. Prepossessing techniques, such as data augmentation (Cui et al., 2015; Ko et al., 2015; Uhlich et al., 2017; He et al., 2016), which have been developed to improve the accuracy, can mitigate demands on memory bandwidth since augmented data may be created via access only to local caches. While this reduces memory bandwidth requirements, at a cost of less independent minibatches, it is only a stopgap measure, as computational capability is improving faster than available memory bandwidth. For example, GPUs are more than five times faster than a few years ago, e.g., the Nvidia V100 (7.8 TFLOPs) vs. the K40 (1.4 TFLOPs), not to mention issues associated with having multiple hungry GPUs on the same machine running simultaneously. Hence, independent random sampling of mini-batches is becoming ever more impractical.

As sequential access is significantly faster than random access, the only practical strategy is to iterate through a fixed sequence of data samples (written a priori to disk) rather than obtain an unbiased random mini-batch at every gradient update step. Although we could consider randomly re-shuffle the data points after every epoch, this can also be an overwhelmingly costly operation. Therefore, to achieve efficient training systems in practice, we often rely on one fixed sequence of data points to iterate through multiple times to train a model (Yu et al., 2012). For example, when training a deep model on ImageNet (Russakovsky et al., 2015), a commonly used approach is to generate

Proceedings of the 22[nd] International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

a fixed randomly shuffled list of indices of the images, construct a database (usually optimized for sequential access) based on this list, and iterate through the order multiple times (Jia, 2014). Even though applying stochastic gradient methods on a deterministic sequence of data can be theoretically suboptimal, the benefits include improved and predictable data access patterns and better reproducibility.

Ideally, a good deterministic sequence of mini-batches should have the following properties: (1) mini-batch representativeness, where every mini-batch is representative and non-redundant, and thus stochastic gradients calculated using the mini-batches are not too far from the true gradients; and (2) order consistency, where groups of mini-batches in the sequence should be more broadly representative, so the corresponding sequence of gradients does not drive a model in the wrong direction. As mentioned above, however, one widely used approach to generate the fixed sequence is random shuffling, which could result in a suboptimal sequence due to non-representativeness. This can impede the performance of the trained model.a

In this paper, we propose a method to generate a deterministic sequence of mini-batches. Our method consists of hierarchical runs of the max-min robust submodular partition algorithm (Wei et al., 2015b) of the dataset with various cardinality constraints, which generates a partial order over mini-batches of data points within a hierarchical structure, and where both mini-batches and groups of mini-batches in the hierarchy are encouraged to be representative of the entire dataset (see Figure 1, bottom, for a simple illustration). Specifically, we offer the following contributions: (1) we provide new theoretical guarantees on a cardinality constrained max-min robust submodular partition problem; (2) we propose a hierarchical structure for robust partitioning to significantly improve the efficiency of the original partitioning algorithm (especially when the number of blocks in the partition is very large) with theoretical guarantees under mild assumptions; (3) with the new hierarchical partitioning approach, we can deterministically generate a sequence of representative mini-batches, and utilize such a sequence for stochastic gradient methods; (4) on deep learning tasks, we show that our deterministic sequences of mini-batches significantly outperform the worst case and mean of randomly generated sequences (both likely to occur in practice); for most of the cases, our approach outperforms the best of the random sequences.

2 RELATED WORK

Robust submodular partitioning (i.e., the submodular fair allocation) was introduced in Golovin (2005) and further studied in Khot and Ponnuswami (2007); Asadpour and Saberi (2010); Wei et al. (2015b) with algorithms and guarantees. In the more recent work (Wei et al., 2015b), they used the partitioning algorithm to separate datasets into blocks for training machine learning models in parallel, with the intuition that the partitioned data block on each machine should be consistent with the whole dataset so the convergence of the parallel system can be improved. We extend their methods substantially to solve the problem of deterministically generating mini-batch sequences for SGD training. In particular, we propose a fast and scalable algorithm to approximately solve the robust submodular partitioning problem under a block size (cardinality) constraint (see Section 3). While this problem is a special case of that considered in Cotter et al. (2018), our analysis is for a different and simpler algorithm that still has guarantees. Moreover, in Wei et al. (2015b), the number of partitioned blocks is typically limited in quantity as every block should contain sufficient data for training the model on a single machine, whereas in our setting, the number of mini-batches can be very large, and the priority queues generated based on the lazy greedy trick (Minoux, 1978) impose a considerable memory cost thereby making the algorithm infeasible in practice (see Sec. 3 for details). To overcome this problem, we extend the robust submodular partitioning method to a recursive hierarchical formulation, which also enforces an order consistency property over the generated mini-batch sequence (see Section 4).

A mini-batch diversification method based on determinantal point processes (DPPs) is given in Zhang et al. (2017), which relies on similarity measurements between data points so that mini-batches with redundant data are given low-probability and mini-batches with more diverse data are given high-probability. Our objective is different from Zhang et al. (2017), as we aim to generate a fixed sequence of representative mini-batches for sequential disk access, while their method generates non-deterministic mini-batches. Moreover, their method does not enforce representativeness of groups of mini-batches, while our approach does. We also note that DPPs are related to submodular functions (a DPP is log-submodular), and our framework is very general as it can be used with any submodular function. Finally, DPP methods are computationally expensive, while our hierarchical partitioning method is efficient enough for very large datasets such as ImageNet. Salehi et al. (Salehi et al., 2017) give a multi-arm bandit sampling approach to adaptively sample mini-batches for stochastic optimization with the aim to better control the variance of gradients of the mini-batches and to improve the convergence rate. This work focuses on generating random mini-batches with less variance than uni-

formly random during training (their method requires accessing gradient information), while we strive to find a fixed mini-batch sequence before training starts. Both Zhang et al. (2017) and Salehi et al. (2017) shares similar intuition with us that mini-batches with more representativeness/more diversity can contribute to better performance of the training system. Our approach, however, stays mindful of the computational requirements as well. Shamir (2016) proposes a sampling strategy specifically for the sampling without replacement scenario, which exhibits convergence properties for convex problems but it requires a global (entire dataset) gradient calculation at each minibatch which can be very slow especially for DNNs; it also takes as input a permutation rather than producing as output a good sequence of minibatches. Zhu et al. (2016) modifies the stochastic algorithms to utilize the structural information obtained from raw clustering on the training dataset to get improved running time. Again, it requires a global gradient calculation and is not designed for minibatch stochastic gradient settings. Curriculum learning (Bengio et al., 2009) and self-paced learning with diversity (Jiang et al., 2014) are also related as they focus on generating mini-batches better during the training process. However, they require random data access, which is often infeasible for very large data sets, while our method produces a single reusable sequential high quality data access pattern, a key unique benefit of our method.

3 SUBMODULAR PARTITIONS FOR MINI-BATCHES

A submodular function f is a set function $2^V \rightarrow \mathbb{R}$, with V as the ground set, and satisfying the property that $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$, where $A, B \subseteq V$. Submodular functions have favorable theoretical properties while achieving good results in practice on various real world applications (Zheng et al., 2014; Nagano et al., 2010; Krause et al., 2008a; Jegelka and Bilmes, 2011; Krause et al., 2008b; Wei et al., 2015a). Unless stated otherwise, the submodular functions we discuss in this paper are restricted to monotone non-decreasing normalized submodular functions ($f(B) \geq f(A) \forall A \subseteq B \subseteq V$, $f(\emptyset) = 0$). Submodular functions are expressive models to describe the level of representativeness of a given set with respect to the ground set. In addition, algorithms associated with optimizing various forms of submodular functions often rely on greedy procedures and generate deterministic results often with theoretical guarantees.

Given a submodular function f on a ground set V, the max-min robust partition problem (submodular fair allocation) (Golovin, 2005) is defined as:

$$\max_{\pi \in \Pi(V)} \min_{i=1:m} f(\pi_i(V)), \tag{1}$$

where $\Pi(V)$ indicates all possible partitions of ground set V, $\pi_i(V)$ represents the set of data points of block i in partition π, and $m = |\pi|$ denotes the number of blocks in the partition. Intuitively, the max-min robust partition objective encourages the worst block (according to the submodular function evaluation) of the partition to be representative of the ground set of data points, and therefore, all blocks in the partition are at least that representative as well.

For our mini-batch partitioning task, we focus on generating a sequence of blocks where every block in the partition is a mini-batch. Suppose every resulting mini-batch is representative due to the max-min robust objective, then the ordering of the mini-batches used to train becomes less crucial, and an arbitrary ordering of the mini-batches will suffice. However, for most problems, the mini-batches are too small to represent the ground set V (meaning $\max_i f(B_i) \ll f(V)$ where i is an index over mini-batches and B_i is the i^{th} mini-batch, a property that may be true even if Eq. (1) is solved optimally), in which case the ordering of the mini-batches is critical. The reason is that, with a poor order, a sub-sequence of minibatches can be redundant and non-representative even though every mini-batch is as representative and non-redundant as possible for its size.

For the remainder of this section, however, we assume the simple case where the mini-batch size is large enough to represent the ground set. This allows us to introduce our approach. We discuss the more general case with small mini-batches later in Section 4.

Since every partition block is a mini-batch, we need to enforce a block size/cardinality constraint such that every block should have a fixed size k. We define $\Pi(V, k)$ to be the set of all possible partitions with size k (for simplicity, assume $|V| \bmod k = 0$), then we have:

$$\max_{\pi \in \Pi(V,k)} \min_{i=1:m} f(\pi_i(V)). \tag{2}$$

We also presents a binary programming version of the objective in Section E in the supplement (Wang et al., 2018). Assuming large k, Eq. (2) naturally describes what we desire for a deterministic mini-batch sequence of training data. Firstly, the max-min objective ensures that all mini-batches are as representative of the entire training dataset as possible. Secondly, the class of submodular functions we can use for the objective is sufficiently expressive that by solving such a problem, we have a general framework that could potentially work for various forms of data and different notions of representativeness (in Section 5, we use a variant of the facility location function as the submodular function to model the representativeness). Finally, we claim that variations of the greedy algorithm can

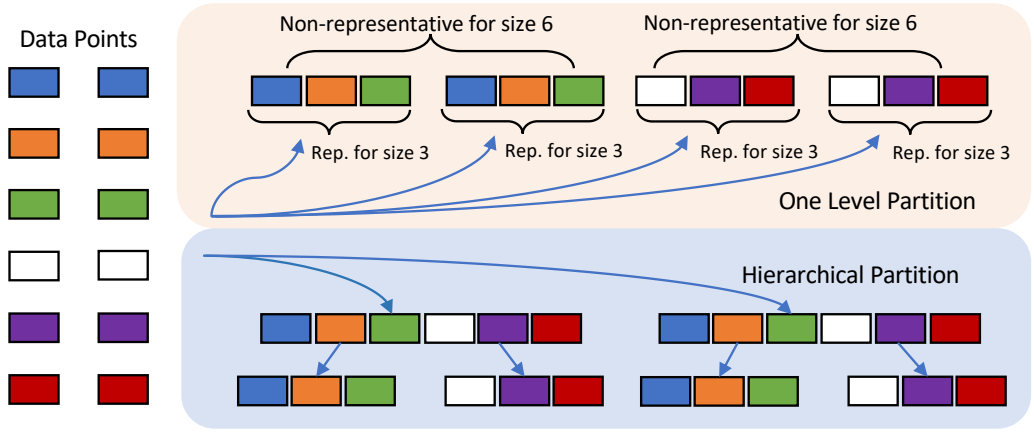

Figure 1: Simple example about how hierarchical partitioning can enforce better order consistency. Suppose we partition a dataset of 12 samples into mini-batches of size 3, and representativeness can be defined as the number of samples with distinct colors. In the upper part, though every mini-batch is representative for its size, the combination of either the first two or the last two mini-batches becomes non-representative for a combined block of size 6 (3 different colors for a set of size 6). On the other hand, the lower part shows the case where we first partition the data into blocks of size 6, and then further partition into mini-batches of size 3, which in this example enforces the representativeness of the combination of blocks.

be applied to solve Eq. (2) efficiently with theoretical guarantees, and that are applicable to large datasets such as ImageNet. We propose Algorithm 1 to address Eq. (2) and show it has guarantees.

Algorithm 1: Cardinality Constrained Submodular Robust Partition (RobustPartitionK(f, V, k))

1 p
input : f, V, k
2 $m := |V|/k$ $R := V$ Let $A_1 = A_2 = ... = A_m = \emptyset$
while $R \neq \emptyset$ **do**
3 $j^* \in \text{argmin}_{j,|A_j|<k} f(A_j)$; // least valued block.
4 $v^* \in \text{argmax}_{v \in R} f(v|A_{j^*})$; // best for block.
5 $A_{j^*} := A_{j^*} \cup \{v^*\}$; // add to block.
6 $R = R \setminus \{v^*\}$
7 **end**
8 Sort A_j's by $f(A_j)$ so that $f(A_{j_1}) \geq f(A_{j_2}) \geq ... \geq f(A_{j_m})$
9 **return** $(A_{j_1}, A_{j_2}, ..., A_{j_m})$

Theorem 1. *For submodular function f on ground set V and block size (mini-batch size) constraint k, suppose $m = |V|/k$, Algorithm 1 gives an approximation ratio of $\frac{e-1}{(e-1)m+1}$.*

The proof is in the supplement (Wang et al., 2018) Sec. C. The bound almost matches (within a factor of $m/(m+1)$) the best known bound for the unconstrained case (Eq. 1) in Wei et al. (2015b). The more general approach of Cotter et al. (2018) can achieve a bi-criterion bound that is hard to compare to the current bound. In Algorithm 1, every iteration finds the worst expandable (i.e., size less than the constraint k) block A_{j^*} in the current partition, and greedily adds element v^* to the block A_{j^*} that increases the gain $f(v^*|A_{j^*})$ the most. In addition, we sort the partitioned blocks by their submodular evaluations in descending order (line 9 of Algorithm 1). Due to the robust max-min objective, all the blocks are encouraged to have similar evaluations, and we choose to put the slightly more representative blocks first, in accordance with the intuition behind curriculum learning (Bengio et al., 2009), which selects easy samples first. Thus, we get a deterministic sequence of representative mini-batches.

The worst case running time of Algorithm 1 is $\mathcal{O}(|V|^2)$ evaluations of f plus $\mathcal{O}(|V|^2)$ basic operations. We care mostly about the number of function evaluations as each f evaluation can be significantly more expensive than a basic operation. Though not wholly intractable, $\mathcal{O}(|V|^2)$ evaluations can become overwhelming when the dataset size gets large. To accelerate the algorithm, we can apply the lazy evaluation trick (Minoux, 1978) at line 5 of Algorithm 1. Essentially, we can create a priority queue for each of the m blocks, and initialize every priority queue with all the singleton values $f(v), v \in V$. When we pop the top of the priority queue, get element v and evaluate $f(v|A_{j^*})$, and if such value is larger than the next value in the priority queue, then by submodularity, v is guaranteed to be in $argmax_{v \in R} f(v|A_{j^*})$, or otherwise we push $f(v|A_{j^*})$ back to the priority and keep popping. In the worst case, the running time with the priority queue would still be $\mathcal{O}(|V|^2)$ evaluations of

f, however, it is widely recognized that, in practice, the running time will be much less than $\mathcal{O}(|V|^2)$ and close to $\mathcal{O}(|V|\log|V|)$ depending on the specific f.

While the acceleration that the lazy evaluation trick brings to the algorithm is good, it has a significant memory cost. The peak memory taken by the collection of priority queues is proportional to $m|V| = |V|^2/k$ (assuming the priority queue implementation has linear memory cost), which could be extreme for a large number of blocks. We hence propose a method to construct the priority queues online rather than initialize every priority queue with all the singleton values. The details of this method can be found in the supplementary document (Wang et al., 2018) Section B, but in the worst case, the memory cost is still quadratic in $|V|$.

We also note that lazier-than-lazy greedy (LTLG) (Mirzasoleiman et al., 2015) can also be applied to line 5 of Algorithm 1 to speed up the greedy step. Depending on the sample size of LTLG method, we can trade-off the performance of the algorithm with the memory and computational cost. If efficiency is the priority, we can apply LTLG on top of our method to achieve an improved speed-up at the cost of some performance.

To further mitigate the computation and memory efficiency issue of Algorithm 1, and also generate mini-batches with better order consistency (which is critical if the mini-batch size k is small relative to $|V|$ and every mini-batch cannot represent the entire dataset), we propose a hierarchical robust partitioning framework, which runs Algorithm 1 with various block size constraint k's on different hierarchical levels.

4 HIERARCHICAL ROBUST SUBMODULAR PARTITION

In Algorithm 2, we describe our hierarchical robust partitioning framework. Instead of having one block size constraint k and partitioning only once, we have a hierarchy of constraints $k_1 > k_2 > \ldots > k_r$, and ideally $k_i \bmod k_{i+1} = 0$. We start by partitioning the ground set into blocks of size k_1, and for every block we get from running Algorithm 1, we further partition each block into smaller blocks with a block size constraint k_2 and so on. In the end, k_r is the mini-batch size, so we get representative mini-batches.

By using Algorithm 2, we significantly reduce the memory cost of applying the lazy evaluation greedy trick. For iteration i of Algorithm 2, the peak memory cost is proportional to $m_i k_{i-1}$ (note $k_0 = |V|$), assuming the memory cost of a priority queues increases linearly. The overall peak memory cost is $\max_{i=1:r} m_i k_{i-1}$. It is easy to see that

Algorithm 2: Hierarchical Submodular Robust Partitioning

input : $f, V, k_1, ..., k_r$

1 $k_0 := |V|$; $Q_1 := (V)$; // Q_i's store sequence of sets to further partition
2 **for** $i := 1;\ i \le r;\ i := i+1$ **do**
3 $m_i := k_{i-1}/k_i$; // m_i: number of blocks for the next partition
4 $Q_{i+1} = ()$; // Q_{i+1} initialized with an empty sequence
5 **for** $j := 1;\ j \le |Q_i|;\ j := j+1$ **do**
6 $A_1, ..., A_{m_i} = \text{RobustPartitionK}(f, Q_i[j], k_i)$; // $Q_i[j]$: jth set in the sequence
7 Append $A_1, ..., A_{m_i}$ to Q_{i+1} ; // Add partitioned blocks of $Q_i[j]$ to solution
8 **end**
9 **end**
10 **return** Q_{r+1}

$\forall i = 1 : r, m_i k_{i-1} \le m|V|$, which is the peak memory cost of Algorithm 1 (note $m|V| = m_1 m_2 ... m_r k_0$). In fact, if we have $r = 2$ and $k_1 = |V|/2$, the memory cost is halved, and the more layers we have in the hierarchy, the less the overall memory cost becomes.

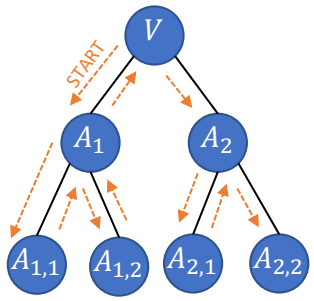

Figure 2: An example partition hierarchy. Leaves visited by any depth-first-search traversal order is consistent with the partial ordering defined by the hierarchy.

In addition to the memory efficiency with the lazy evaluation trick, Algorithm 2 also enforces groups of mini-batches in the hierarchy structure to be representative. Essentially, as we have a hierarchy of block size constraints $k_1, ..., k_r$, not only the final (lowest) level mini-batches are representative for their size, but the combination of multiple mini-batches is also representative for their combined size based on the choice of k's (see Figure 1 for a simple illustration). The original partitioning algorithm is a special case of the hierarchical partitioning with one level in the structure of hierarchy. As mentioned above, Algorithm 1 only applies to the case where the mini-batch is large enough to represent the dataset, and Algorithm 2 applies to general mini-batch sizes, while also making the original algorithm more efficient. Also, note that

the generated hierarchy structure defines a partial ordering of the data samples, and any depth-first-search traversal of the hierarchy structure is consistent with the partial ordering (see Figure 2). We can also continue to use the decreasing submodular evaluation ordering as well (Algorithm 1 line 9).

Though we can get a significant reduction in memory cost, as well as generating representative groups of mini-batches by utilizing the hierarchical robust partitioning algorithm, we lose the original theoretical guarantee relative to the objective defined in Eq. (2). In fact, we show that the approximation ratio can be arbitrarily bad by a simple example in the supplementary material (Wang et al., 2018) Section D. However, under mild assumptions about the data points and the function f, i.e., if we assume that in the process of calling Algorithm 1 from Algorithm 2 blocks are not filled in an extremely imbalanced way (which we find not typically occur in practice), then we have the following bound for Algorithm 2.

Definition 1. *We run Algorithm 1 with ground set V', block size constraint k' and $m' = |V'|/k'$, the greedy step (line 5) gets executed $T = |V'|$ times. and we get a sequence of sets $Q = (A_1^T, A_2^T, \ldots, A_{m'}^T)$ as the output, with $A_j'^T$ having the minimal evaluation, i.e. $j' \in \operatorname{argmin}_{i=1:m'} f(A_i^T)$. There exists an earliest greedy step t ($1 \leq t \leq T$), such that $|A_{j'}^t| = k'$, and $j' \in \operatorname{argmin}_{1:m'} f(A_i^t)$ (A_i^t is the ith block at greedy step t), we define $\tau := \min_{i=1:m'} |A_i^t|$.*

Theorem 2. *If we have $\tau \geq 2$ as defined in Def. 1 for every call to Algorithm 1 from Algorithm 2, then we achieve an approximation ratio of $(\frac{\tau-1}{2\tau-1})^r \frac{k_r}{|V|}$.*

The proof is in the supplement (Wang et al., 2018) Sec. D. τ as defined in Def. 1 indicates whether the blocks are filled in a balanced manner in Algorithm 1. $\tau \geq 2$ means that when the worst block (i.e., the block with minimal evaluation) has size k', and all other blocks have higher evaluations, the smallest block (in terms of size) has at least two elements, or in other words, the greedy steps do not generate drastically imbalanced blocks. In practice, for the real datasets in our experiments, τ always has large values, so the extra factor we get from Theorem 2 compared to Theorem 1 is close to 2^{-r}, and r is the number of layers in the hierarchy structure, which is at most $\log |V|$ and typically quite small in practice (3 to 5 in our experiments). This means Algorithm 2 will perform well. More details are found in the supplement (Wang et al., 2018) Section F.

The $\{k_i\}_{i=1}^r$ values are hyper-parameters for our method. Suppose the desired mini-batch size is large enough to represent the entire dataset, then the choices of $\{k_i\}_{i=1}^r$ essentially should follow the memory capacity of the hardware. However, for real-life problems, the data may be very complex, and the mini-batch size is restricted by the GPU's memory size, and thus quite small compared to the dataset size. For example, in ImageNet, there are over one million training data points in 1000 classes, while the most widely adopted batch sizes on a single GPU card range from 128 to 512. The batch size of 128 can hardly be fully representative, as there are 1000 distinct classes. Therefore, Algorithm 2 is indeed required for partitioning such datasets as a poor order could happen similar to the case shown in Figure 1. Thus, one guideline for setting $\{k_i\}_{i=1}^r$ in addition to the efficiency restriction is to have $\{k_i\}_{i=1}^r$ start with multiple times of the number of classes, and gradually drill down to the desired mini-batch size. Algorithm 2 generates mini-batches of better quality than the ordinary robust partitioning case when we encounter datasets with more number of classes and larger data point size.

5 CHOICE OF SUBMODULAR f

Our objective defined in Eq. (2) is a general framework for generating deterministic mini-batch sequences as the class of submodular function f is flexible and expressive for describing the representativeness of a set of data points. In this paper, for supervised classification tasks, we use a nearest-neighbor submodular function (a special case of a facility location function) as our choice of f:

$$f_{NN}(S) = \sum_{v \in V} \max_{v' \in S} \operatorname{sim}(v, v'), \tag{3}$$

where $\operatorname{sim}(\cdot)$ is defined over a pair of data points, and outputs the similarity of the two points. f_{NN} is a special case of the facility location function since similarities between data points with different class labels are zero (i.e., a sparse similarity graph), i.e. $y(v) \neq y(v') \rightarrow \operatorname{sim}(v, v') = 0$, where $y(v)$ denotes the label of data point v. Variations of the facility location function have been successfully applied to various data selection/summarization tasks (Wei et al., 2014, 2013; Tsang et al., 2005).

f_{NN} naturally captures the maximum likelihood estimates over the given data set for a nearest-neighbor classifier. Under mild assumptions, maximizing f_{NN} is equivalent to find the optimal subset of data points to form a nearest-neighbor classifier (Wei et al., 2015a). Ideally, given a specific model (e.g., a nearest-neighbor classifier or a deep neural network), we should design f based on the objective of the model, so the exact quantization of representativeness defined by f is consistent with the given model. We choose to use f_{NN} because the nearest-neighbor classifier is quite generic, and f_{NN} is efficient to compute even for large datasets. Moreover, f_{NN} is also very flexible as it can be applied with various

ways of calculating the similarity graph, We calculate the similarities between pairs of data points using:

$$\text{sim}(v_1, v_2) = e^{\frac{-||x(v_1)-x(v_2)||_2}{\sigma}}. \quad (4)$$

where $x(v)$ represents the vector representation or features of data point v, and σ is a chosen parameter, which works as a normalization factor. In practice, $x(v)$ can be the original data representation (e.g. raw pixel values for images), the final layer output of a deep neural network classifier, or the bottleneck layer of an autoencoder. While σ can be tuned, in principle we set it to $mean_{i,j}||x(v_i) - x(v_j)||_2$ for all the i, j pairs in the f_{NN} similarity graph.

6 EXPERIMENTS

We evaluate our hierarchical robust partitioning algorithm for generating mini-batch sequences for the CIFAR-100 (Krizhevsky and Hinton, 2009) and ImageNet(ILSVRC12) (Russakovsky et al., 2015) datasets. We show that our deterministically generated sequences consistently outperform randomly generated sequences, a widely-adopted approach for training deep neural networks. The initialization of the two datasets are fixed by random seed. For CIFAR-100, we use the PyTorch toolkit (Paszke et al., 2017), and for ImageNet, we use MXNET (Chen et al., 2015).

6.1 CIFAR-100 Dataset

CIFAR-100 dataset is an image classification dataset with 50,000 training images and 10,000 testing images, with each image of dimension $(3, 32, 32) = 3072$. The dataset has 100 classes, and all the classes have the same number of training/testing data points. We randomly select 100 samples from each class in the training set as the validation set. We train a convolutional autoencoder network to extract more compact representations of the raw pixel data (see the supplement (Wang et al., 2018) Sec. G for details of the autoencoder structure). The autoencoder network in use has 48 layers with a 128-dimensional bottleneck layer and was trained on the large TinyImage data set (Torralba et al., 2008). The network employs residual blocks (He et al., 2015). Furthermore, the training was performed using ADAM (Kingma and Ba, 2014) with batch normalization and rectified linear units. The output of the bottleneck layer act as the 128-dimensional feature representation of the data, which is then used to compute the similarities between data points for f_{NN}.

The deterministic data sequence is generated through a multi-step mechanism. First, we generate a sparse similarity graph using the similarity metric discussed in Sec 5 (containing $400^2 \times 100 = 16M$ entries) that can be used to instantiate f_{NN}. Next, we apply Algorithm 2 with f_{NN} as the submodular function, and block size constraints $k_1 = 1024$, $k_2 = 512$ and $k_3 = 128$, where k_3 is equal to the mini-batch size used for training the deep neural network. We also note that for the data points that are not selected by Algorithm 2 (40000 mod 1024 = 64), we form a final batch (padded depending on the implementation of the toolkit) and append to the end of our mini-batch sequence.

For a given sequence of mini-batches, we train a Wide Res-net (Zagoruyko and Komodakis, 2016) having structure WRN-28-8 (using the same terminology as Zagoruyko and Komodakis (2016)). The network was trained using the NAG (Sutskever et al., 2013) optimization method, an initial learning rate of 0.1 and a linearly decaying learning rate schedule. The data was augmented using random flipping of images. We note that the training hyper-parameters are determined by manually tuning on the validation set accuracy while training on a randomly generated sequence (same for ImageNet dataset described below). We compare the validation set accuracy, and test set accuracy of our deterministically generated sequence to 30 randomly generated sequences. Figures 3a and 3b demonstrate that our method outperforms the random generated sequences, and can also do better than the best over 30 random runs. The p-value significance test for the test accuracy is 0.0009, which means our improvement is quite significant.

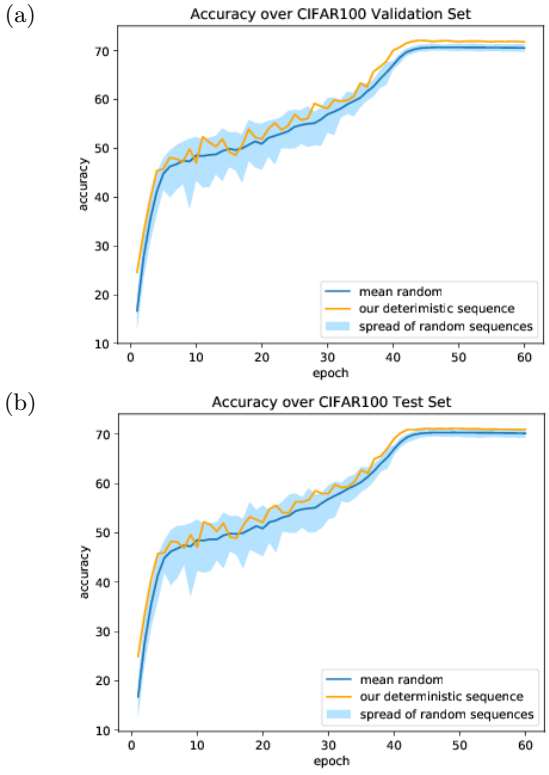

Figure 3: Accuracy over the {validation set (a), test set (b)} for our sequence along with the performance spread for the random sequences.

6.2 ImageNet Dataset

The ImageNet classification dataset contains roughly 1.2 million training images and 1000 classes. The number of data points for each class is almost balanced, and each class has around 1k data points. The validation set consists of around 50k samples, and there is no standard test set. We train a deep neural net classifier and retrieve the final layer output (before softmax) as the feature representations for data points. The DNN classifier is a Residual Network (He et al., 2016) with 18 layers (Resnet-18), and the feature representation has 1000 dimensions. Similar to the CIFAR-100 dataset case, we use the extracted 1000 dimensional features to generate the sparse similarity graph. Then we utilize Algorithm 2 with $k_1 = 32768$, $k_2 = 16384$, $k_3 = 8192$, $k_4 = 4096$, $k_5 = 128$ to generate the deterministic sequence of mini-batches. For the non-selected data points (3215 data samples, since the total number of samples does not divide k_1), we form a new ground set and run Algorithm 1 to partition them into mini-batches of size 128 and append them to the end of the sequence.

We compare the performance of different sequences using the same model used to generate features for data points, i.e., the Resnet-18 network structure. For both cases, we use again the NAG (Sutskever et al., 2013) optimization method with an initial learning rate 0.1, exponentially decaying learning rate schedule, and data augmentation of random flipping and random cropping. We note that for training the feature extraction model, we use a deterministic sequence generated using the L2 distance of the raw data as the distance measurement for constructing the similarity graph.

As shown in Figures 4a and 4b, for most cases, our deterministic sequence yields better results than even the best among the 15 randomly generated sequences, significantly beating the mean and the worst random, which are both likely to happen when training deep models. The p-value significance test result is 0.0151 for top-1 accuracy, and 0.0209 for top-5 accuracy, which means our improvement is quite significant.

In the supplement (Wang et al., 2018) Sec. A, We show the running time results of Algorithm 1 with the lazy evaluation trick, which due to memory demands is only feasible in the hierarchical partitioning framework.

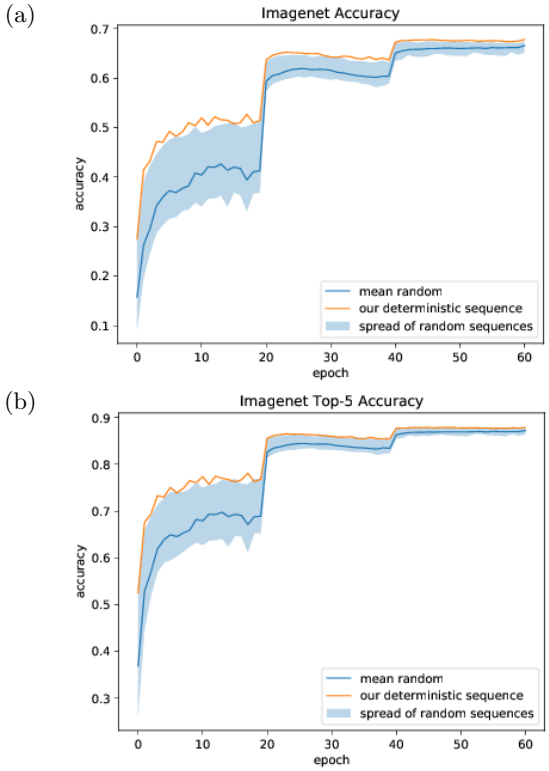

Figure 4: {Top-1 Accuracy (a), Top-5 Accuracy (b)} over the validation set for our sequence along with the performance spread for the random sequences.

7 CONCLUSION

We proposed a robust submodular partitioning based framework to generate a fixed sequence of mini-batches for training machine learning systems with stochastic mini-batch gradient methods. We show such deterministic sequences outperform the randomly generated sequences on the CIFAR-100 and ImageNet datasets. We proposed a hierarchical robust partitioning algorithm, which dramatically improves the efficiency of the ordinary partitioning method, while enhances the quality of the generated sequence of blocks. We also note that the deterministic sequences apply to multi-epoch training systems that require hyperparameter tuning, and therefore the computational cost of our method becomes negligible in the long run.

We would like to explore more forms of the submodular function f. As discussed in Section 5, for different training models, we can use different f having customized notions of representativeness (e.g., for different classifiers (Wei et al., 2015a)). Moreover, f need not be submodular, e.g., difference of submodular functions may be useful. Furthermore, our robust objective is imperfect in describing the optimal mini-batch sequence objective. The current objective is a partitioning problem, but the actual goal is to produce an ordering of sets. Though we use a hierarchical decomposition to partially generate a good block ordering, it is still only indirect. We hope to focus on directly optimizing on the optimal sequence of sets in the future.

This work was supported in part by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

References

Arash Asadpour and Amin Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. In *SICOMP*, 2010.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, abs/1512.01274, 2015.

Andrew Cotter, Mahdi Milani Fard, Seungil You, Maya Gupta, and Jeff Bilmes. Constrained interacting submodular groupings. In *International Conference on Machine Learning (ICML)*, Stockholm, Sweden, July 2018.

Xiaodong Cui, Vaibhava Goel, and Brian Kingsbury. Data augmentation for deep neural network acoustic modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(9):1469–1477, 2015.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12 (Jul):2121–2159, 2011.

Daniel Golovin. Max-min fair allocation of indivisible goods. *Technical Report CMU-CS-05-144*, 2005.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

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*, pages 770–778, 2016.

S. Jegelka and J. Bilmes. Submodularity beyond submodular energies: coupling edges in graph cuts. In *CVPR*, 2011.

Yangqing Jia. Tutorial on training imagenet with caffe. http://caffe.berkeleyvision.org/gathered/examples/imagenet.html, 2014.

Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. Self-paced learning with diversity. In *Advances in Neural Information Processing Systems*, pages 2078–2086, 2014.

Subhash Khot and Ashok Ponnuswami. Approximation algorithms for the max-min allocation problem. In *APPROX*, 2007.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. Audio augmentation for speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. In *JMLR*, 2008a.

Andreas Krause, Brendan McMahan, Carlos Guestrin, and Anupam Gupta. Robust submodular observation selection. In *JMLR*, 2008b.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670. ACM, 2014.

M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, 1978.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *AAAI*, pages 1812–1818, 2015.

Kiyohito Nagano, Yoshinobu Kawahara, and Satoru Iwata. Minimum average cost clustering. In *Advances in Neural Information Processing Systems*, pages 1759–1767, 2010.

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

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.

Farnood Salehi, Elisa Celis, and Patrick Thiran. Stochastic optimization with bandit sampling. *arXiv preprint arXiv:1708.02544*, 2017.

Ohad Shamir. Without-replacement sampling for stochastic gradient methods. In *Advances in Neural Information Processing Systems*, pages 46–54, 2016.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 30(11):1958–1970, 2008.

Ivor W Tsang, James T Kwok, and Pak-Ming Cheung. Core vector machines: Fast svm training on very large data sets. In *JMLR*, 2005.

Stefan Uhlich, Marcello Porcu, Franck Giron, Michael Enenkl, Thomas Kemp, Naoya Takahashi, and Yuki Mitsufuji. Improving music source separation based on deep neural networks through data augmentation and network blending. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 261–265. IEEE, 2017.

Shengjie Wang, Wenruo Bai, Chandrashekhar Lavania, and Jeff Bilmes. Supplementary material: Fixing mini-batch sequences with hierarchical robust partitioning. *AISTATS 2019*, 2018.

Kai Wei, Yuzong Liu, Katrin Kirchhoff, and Jeff Bilmes. Using document summarization techniques for speech data subset selection. In *North American Chapter of the Association for Computational Linguistics/Human Language Technology Conference (NAACL/HLT-2013)*, Atlanta, GA, June 2013.

Kai Wei, Yuzong Liu, Katrin Kirchhoff, Chris Bartels, and Jeff Bilmes. Submodular subset selection for large-scale speech training data. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, Florence, Italy, 2014.

Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *ICML*, 2015a.

Kai Wei, Rishabh K Iyer, Shengjie Wang, Wenruo Bai, and Jeff A Bilmes. Mixed robust/average submodular partitioning: Fast algorithms, guarantees, and applications. In *Advances in Neural Information Processing Systems*, pages 2233–2241, 2015b.

Hsiang-Fu Yu, Cho-Jui Hsieh, Kai-Wei Chang, and Chih-Jen Lin. Large linear classification when data cannot fit in memory. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(4):23, 2012.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Cheng Zhang, Hedvig Kjellström, and Stephan Mandt. Determinantal point processes for mini-batch diversification. In *33rd Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, 11 August 2017 through 15 August 2017*. AUAI Press Corvallis, 2017.

Jingjing Zheng, Zhuolin Jiang, Rama Chellappa, and Jonathon P Phillips. Submodular attribute selection for action recognition in video. In *NIPS*, 2014.

Zeyuan Allen Zhu, Yang Yuan, and Karthik Sridharan. Exploiting the structure: Stochastic gradient methods using raw clusters. In *NIPS*, 2016.