Submodular Batch Selection for Training Deep Neural Networks

K J Joseph [1] Vamshi Teja R [1] Krishnakant Singh [1] Vineeth N Balasubramanian [1]

Abstract

Mini-batch gradient descent based methods are the de facto algorithms for training neural network architectures today. We introduce a mini-batch selection strategy based on submodular function maximization. Our novel submodular formulation captures the informativeness of each sample and diversity of the whole subset. We design an efficient, greedy algorithm which can give high-quality solutions to this NP-hard combinatorial optimization problem. Our extensive experiments on standard datasets show that the deep models trained using the proposed batch selection strategy provide better generalization than Stochastic Gradient Descent as well as a popular baseline sampling strategy across different learning rates, batch sizes, and distance metrics.

1. Introduction

Stochastic gradient descent (SGD) remains the standard method for training deep neural networks. The mini-batch SGD is an optimization algorithm where mini-batches of data $D_t = \{d_1, d_2, \cdots, d_m\}$ containing m examples are sampled uniformly from the dataset D, at time t. A loss function value w.r.t. the current model parameters w_t is computed as $\mathcal{L}(w_t) = \sum_{i=1}^{m} l(d_i|w_t)$ (where $l(.)$ is any differentiable loss function for the neural network), and the weights are updated to minimize $\mathcal{L}(w_t)$, according to the following equation:

$$w_{t+1} = w_t - \mu_t \frac{\partial \mathcal{L}(w_t)}{\partial w_t} \tag{1}$$

where μ_t is the learning rate at the t^{th} step.

In this work, we hypothesize and validate that not only is the update of w_t given $\frac{\partial \mathcal{L}(w_t)}{\partial w_t}$ crucial, but also the selection of the mini-batch D_t used to compute the gradient. We formulate batch selection as solving a submodular optimization problem, which contributes to significant improvement in the generalization performance of the model. *To the best of our knowledge, this is the first submodular batch selection methodology for SGD.* Each mini-batch selection is posed as a cardinality-constrained monotone submodular function maximization problem. This helps us leverage a greedy algorithm to solve this NP-hard combinatorial optimization problem, which guarantees a solution for a submodular objective function which is at least (in the worst case) $(1 - \frac{1}{e})$ (approximately 0.63) of the optimal solution (Nemhauser et al., 1978).

The key contribution of our work is a new submodular sampling strategy for mini-batch SGD, which helps train deep neural networks to have better generalization capability. To achieve this, we *formulate a submodular objective function*, which takes into account the informativeness that each sample can add to the subset and at the same time ensure that the subset as a whole, is diverse. Further, we propose *an efficient algorithm to scale to high sampling rates*, as required for SGD while training neural networks. We conduct extensive experimental studies of the proposed submodular mini-batch selection methodology and show improved generalization capability over SGD as well as related previous efforts such as Loss based sampling (Loshchilov & Hutter, 2015). We also show that the improved performance of the proposed methodology is consistent across different learning rates, mini-batch sizes and distance metrics. We use the terms batch selection and mini-batch selection interchangeably in this work.

2. Related Work

Mini-batch selection strategies have been explored in convex settings in the past. (Zhao & Zhang, 2014) proposed a sampling scheme based on partitioning data into balanced strata leading to faster convergence, while (Zhao & Zhang, 2015) proved that the optimal sampling distribution is directly related to the absolute values of the gradient of the samples for convex objectives. However, the prohibitive cost of evaluating the gradient impedes their usage in practice. In non-convex settings, such as in deep neural networks, there have been fewer efforts for mini-batch selection, especially in the context of SGD. Extensions of (Zhao & Zhang, 2015) to neural networks do not scale, due to the large number of trainable parameters in the deep models. Recently, (Loshchilov & Hutter, 2015; Alain et al., 2015)

[1] Indian Institute of Technology Hyderabad, India. Correspondence to: K J Joseph <cs17m18p100001@iith.ac.in>.

Proceedings of the Workshop on Negative Dependence in Machine Learning at the 36^{th} *International Conference on Machine Learning*, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

have tried to alleviate the cost of computing gradients by using loss-based sampling as an approximation to the gradients. Unfortunately, these methods are very sensitive to hyperparameters and perform inadequately in many cases (Katharopoulos & Fleuret, 2017). The work was, in fact, validated only on MNIST data, which was acknowledged as a limitation of the work in (Loshchilov & Hutter, 2015). The only other efforts to our knowledge consider more efficient approximations to the batch gradients i.e. variance of the samples (Chang et al., 2017) or an upper bound on the gradient norm (Katharopoulos & Fleuret, 2018), but the generalization performance is only comparable to SGD in most cases. On the other hand, there have been efforts to speed up mini-batch SGD in general such as (Allen-Zhu, 2017; Johnson & Zhang, 2013). However, these efforts do not focus on batch selection or importance sampling, and show lower performance than existing importance sampling methods as noted in (Katharopoulos & Fleuret, 2018).

Submodular optimization has been successfully applied to varied tasks like document summarization (Lin & Bilmes, 2011), sensor placement (Shamaiah et al., 2010), speech recognition systems (Wei et al., 2014), to name a few. However, there has been no work so far on using submodularity for batch selection. (Das & Kempe, 2008) proved that variance between a predictor variable using full batch and a mini-batch is submodular. We were initially motivated by this observation to propose a submodular batch selection strategy for SGD. The existing efforts that are closest to ours include Determinantal Point Process (DPP) (Zhang et al., 2017), Repulsive Point Processes (RPP) (Zhang et al., 2018) and (Singh & Balasubramanian, 2018). Both DPP and RPP can be considered a special case of probabilistic submodular functions (PSF), although not explicitly called so in their work. However, these methods are computationally inefficient, with even faster versions prohibitively costly to be applied in deep neural networks (Li et al., 2016). (Singh & Balasubramanian, 2018) attempt a similar objective, but the objective considered is truly not submodular, and their results are largely inconclusive. Another body of work that can be considered close to our efforts are those of self-paced learning (Thangarasa & Taylor, 2018) and curriculum learning (Zhou & Bilmes, 2018). However, their objectives are different, and one can consider using our batch selection strategy along with any such method too.

3. Methodology

3.1. Submodular Batch Selection

An appropriate batch selection strategy for mini-batch SGD would need to consider multiple criteria to choose the most relevant samples. A primary criterion we consider is that each selected sample must be as informative as possible. We use the model uncertainty as the measure of informativeness.

Uncertainty Score [$U(x_i)$]: The uncertainty of each data point is computed as the entropy of the current model w^t at training iteration t. C is the set of all classes. This allows the model to select the samples that confuse it the most in a mini-batch:

$$U(x_i) = -\sum_{y \in C} P(y|x_i, w^t) \log P(y|x_i, w^t) \qquad (2)$$

A subset that maximizes only the Uncertainty Score, would potentially lead to the inclusion of similar data points with high entropy in the mini-batch. This redundancy should be avoided to make the mini-batch diverse. The following score helps to contain the inclusion of redundant data points:

Redundancy Score [$R(x_i)$]: Two data points x_i and x_j may separately furnish valuable information, but including both may make the subset less maximally informative. We use Redundancy Score to take this into account. Given $\phi(.)$ to be any distance metric between the two data points, a greater value of the minimum distance between points in the subset would imply more diversity among the data points in the subset. (Needless to say, this score is dependent on the choice of distance metric, and we study this in our experiments.)

$$R(x_i) = \min_{x_j \in S : i \neq j} \phi(x_i, x_j) \qquad (3)$$

Going further, one can notice that outlier samples may maximize the above scores. In order to counter such a selection of batches, we introduce the following score.

Mean Closeness Score [$MC(x_i)$]: This term encourages the selection of data points that are closer to the mean of all the examples ($\mu = \frac{1}{|V|}\sum_{k=1}^{|V|} x_k$) to be picked. This avoids the selection of outlier samples to the extent possible.

$$MC(x_i) = \phi(x_i, \mu) \qquad (4)$$

Finally, there has been recent work to show that closeness in the feature space of a deep neural network may be a better indicator of how similar two samples are (as shown by (Wei et al., 2014) in the speech domain). We hence also include a term to explicitly enforce diversity in the feature space of the given data.

Feature Match Score [$FM(x_i)$]: This score selects samples that are diverse across each dimension in the feature space. $g(.)$ is a non-negative monotone non-decreasing concave function, U is a set of fixed features and $m_u(x_i)$ is a non-negative score, measuring the degree to which data point x_i, possesses the feature u.

$$FM(x_i) = \sum_{u \in U} g(m_u(x_i)) \qquad (5)$$

Our implementation of this score (as well as others) is described in Appendix C.

We combine the above-mentioned scores to form our objective function for batch selection, $\mathcal{F}(S)$, as below and then show the submodularity of the proposed $\mathcal{F}(S)$.

$$\mathcal{F}(S) = \sum_{x_i \in S} \lambda_1 U(x_i) + \lambda_2 R(x_i) + \lambda_3 MC(x_i) + \lambda_4 FM(x_i) \tag{6}$$

Given a dataset with N training data points, a mini-batch of size k is selected by solving the following cardinality-constrained submodular optimization problem:

$$\max_{S \subseteq V, |S| \leq k} \mathcal{F}(S) \tag{7}$$

We now show that the score function $\mathcal{F}$ is indeed submodular and is monotonically non-decreasing. This would allow us to solve the problem in (7) using a greedy approach (Nemhauser et al., 1978).

Lemma 1. *The score function $\mathcal{F}(.)$, defined in Eqn 6 is submodular.*

Proof. Consider two subsets of training examples from a dataset $V = \{x_1, x_2, \cdots, x_n\}$; S_1 and S_2, such that $S_1 \subseteq S_2 \subseteq V$. Let a be an element not selected so far: $a \in V \setminus S_2$. The marginal gain of adding a to S_1 is given by:

$$\begin{aligned}\mathcal{F}(a|S_1) = &\ \mathcal{F}(\{a\} \cup S_1) - \mathcal{F}(S_1)\\ = &\ \lambda_1 U(a) + \lambda_2 \min_{a_j \in S_1} \phi(a, a_j) + \lambda_3 MD(a) + \lambda_4 FM(a)\end{aligned}$$

Similarly, the marginal gain of adding a to S_2 is given by:

$$\begin{aligned}\mathcal{F}(a|S_2) = &\ \mathcal{F}(\{a\} \cup S_2) - \mathcal{F}(S_2)\\ = &\ \lambda_1 U(a) + \lambda_2 \min_{a_j \in S_2} \phi(a, a_j) + \lambda_3 MD(a) + \lambda_4 FM(a)\end{aligned}$$

Since $S_1 \subseteq S_2$, the minimum distance of the new point a, from S_1 would be greater than any element from S_2, as there may exist a point in S_2 that is much closer to a, than any element from its subset S_1. Hence,

$$\min_{a_j \in S_1} \phi(a, a_j) \geq \min_{a_j \in S_2} \phi(a, a_j)$$

Thus, we can claim $\mathcal{F}(a|S_1) \geq \mathcal{F}(a|S_2)$. Hence, the score function $\mathcal{F}(.)$ is submodular. □

Lemma 2. *The score function $\mathcal{F}(.)$ in Eqn 6 is a monotonically non-decreasing function.*

Proof. Consider a subset S and an element $a \in V \setminus S$. When a is added to S, the function value of $\mathcal{F}(\{a\} \cup S)$ changes by $\lambda_1 U(a) + \lambda_2 \min_{a_j \in S} \phi(a, a_j) + \lambda_3 MD(a) + \lambda_4 FM(a)$. All these are non-negative quantities. Thus, $\mathcal{F}(\{a\} \cup S) \geq \mathcal{F}(S)$, and the score function $\mathcal{F}(.)$ is hence monotonically non-decreasing. □

Theorem 1. *Let S^* denote the optimal solution of the problem in (7) and S denote the solution obtained for the same problem using a greedy approach. Then:*

$$\mathcal{F}(S) \geq (1 - \frac{1}{e})\mathcal{F}(S^*)$$

Proof. Having proved that score function $\mathcal{F}(.)$ is submodular in Lemma 1 and that it is monotonically non-decreasing in Lemma 2, the proof follows directly from Theorem 4.3 in (Nemhauser et al., 1978). □

3.2. Scaling to High Sampling Rates

It is interesting to note that application settings where submodularity has worked well hitherto (Wei et al., 2014; 2015; Chakraborty et al., 2015; Lin & Bilmes, 2011) do not require a high sampling rate, as much as demanded by mini-batch selection in SGD. Consider a mini-batch training algorithm that consists of p epochs and q iterations in each epoch, the submodular batch selection needs to be carried out $p \times q$ times. Concretely, for training on the CIFAR-100 dataset having 50,000 examples and a batch size of 50, we need 100,000 batch selections for 100 epochs. Even a greedy algorithm (Nemhauser et al., 1978) that uses scores such as pairwise distance metrics has a complexity of $O(n^2)$, where n is the dataset size. This would be too slow for use with SGD. We hence seek more efficient mechanisms to implement the proposed batch selection strategy.

Recent efforts have attempted to make submodular sampling faster in a more general context (not in SGD), such as Lazy Greedy (Minoux, 1978), Lazier than Lazy Greedy (LtLG) (Mirzasoleiman et al., 2015), and Distributed Submodular Maximization (Mirzasoleiman et al., 2013). In this work, we present a new methodology for efficient submodular sampling inspired by Distributed Submodular Maximization algorithm and Lazier than Lazy Greedy algorithm (LtLG), as described in Algorithm 1. The algorithm partitions the training set V into m partitions in Line 2 and runs Lazier than Lazy Greedy algorithm (LtLG) (Mirzasoleiman et al., 2015) (described later in this section) on each partition (Lines 5 through 8) to obtain m subsets, each of size b. These subsets ($S_i s$) are then merged in Line 9. The final subset S is selected from this merged set by running LtLG again on it. This divide-and-conquer strategy is motivated by the Distributed Submodular Maximization algorithm (Mirzasoleiman et al., 2013).

The Lazier than Lazy Greedy (LtLG) (Mirzasoleiman et al., 2015) algorithm starts with an empty set and adds an element from set R, which maximizes the marginal gain $\mathcal{F}(a|S) = \mathcal{F}(a \cup S) - \mathcal{F}(S)$. This is repeated until the cardinality constraint ($|S| \leq b$) is met. The set R is created by randomly sampling $s = \frac{|V|}{b} log \frac{1}{\epsilon}$ items from the superset V, where ϵ is a user-defined tolerance level. We refer the readers to (Mirzasoleiman et al., 2015) for further information. The model at the k^{th} training iteration, w_k, is used while computing $\mathcal{F}(.)$ as defined in Equation 6. The solution produced by Algorithm 1, S, has the following approximation guarantee with the optimal solution S^*:

$$\mathcal{F}(S) \geqslant \frac{(1 - e^{-1})^2}{min(m, b)} (1 - e^{-1} - \epsilon) \mathcal{F}(S^*) \tag{8}$$

Here, m refers to the number of partitions, b refers to the mini-batch size and e is the base of natural logarithm. Our empirical results (Section 4.1) shows that the approximation is much better than this lower bound in practice. The end-to-end training algorithm is described in Appendix B.

Algorithm 1 Algorithm GETMINIBATCH

```
Input: Training set $V$, Model at $k^{th}$ iteration $w_k$, Batch size $b$, Number of partitions $m$, $\mathcal{F} : 2^V \to \mathbb{R}$ (Eqn 6).
Output: Mini-batch $S \subseteq V$ satisfying $|S| \leq b$.
 1: $S \leftarrow \phi$
 2: Partition $V$ into $m$ sets $V_1, V_2, V_3, \cdots, V_m$.
 3: for i = 1 to m do
 4:     $S_i \leftarrow \phi$
 5:     for j = 1 to b do                    ▷ Do LtLG for each partition.
 6:         $R \leftarrow$ a subset of size $s$ obtained by sampling randomly from $V_i \setminus S_i$.
 7:         $a_j \leftarrow \arg\max_{a \in R} \mathcal{F}(a|S_i)$
 8:         $S_i \leftarrow S_i \cup \{a_j\}$
 9: $S_{merged} \leftarrow \bigcup_{i=1}^{m} S_i$    ▷ Merge result of each partition.
10: for j = 1 to b do                        ▷ Do LtLG on the merged set.
11:     $R \leftarrow$ a subset of size $s$ obtained by sampling randomly from $S_{merged} \setminus S$.
12:     $a_j \leftarrow \arg\max_{a \in R} \mathcal{F}(a|S)$
13:     $S \leftarrow S \cup \{a_j\}$
14: Return $S$
```

4. Experiments and Results

We conduct extensive experimental evaluations to study the effectiveness of submodular mini-batches in training deep neural networks over Stochastic Gradient Descent (SGD) and Loss-based sampling (Loshchilov & Hutter, 2015), as in earlier efforts such as (Katharopoulos & Fleuret, 2018). For brevity, we refer to our proposed method of selecting submodular mini-batches for training as **Sub****M**odular **D**ata **L**oader (**SMDL**). We study the performance on the standard image classification task (as used in related earlier efforts) with SVHN (Netzer et al., 2011), CIFAR-10 and CIFAR-100 (Krizhevsky & Hinton, 2009) datasets. ResNet 20 (He et al., 2016) is used as the network architecture for SVHN and CIFAR-10, while ResNet 32 is used with CIFAR-100. Our implementation details are described in Appendix C.

4.1. Results

We present the major result of our proposed submodular mini-batch selection method, SMDL in Figure 1 and Table 1. We train the two network architectures on three datasets. The generalization performance of these classification models, as measured by their test accuracy and test loss, is used as the evaluation metric. We see from Figure 1 and Table 1 that SMDL is able to achieve lower error and loss, *consistently* across epochs, *on all the three* datasets. It is worth noting that Loss-based sampling fails significantly on SVHN. Such deterioration of generalization performance is also noted in (Loshchilov & Hutter, 2015). SMDL maintains its performance consistently across all the datasets. The values reported on SGD are its mean after conducting five trials. To maintain consistency across experiments, all models are trained only for 100 epochs. Use of batch normalization (Ioffe & Szegedy, 2015) and adaptive learning

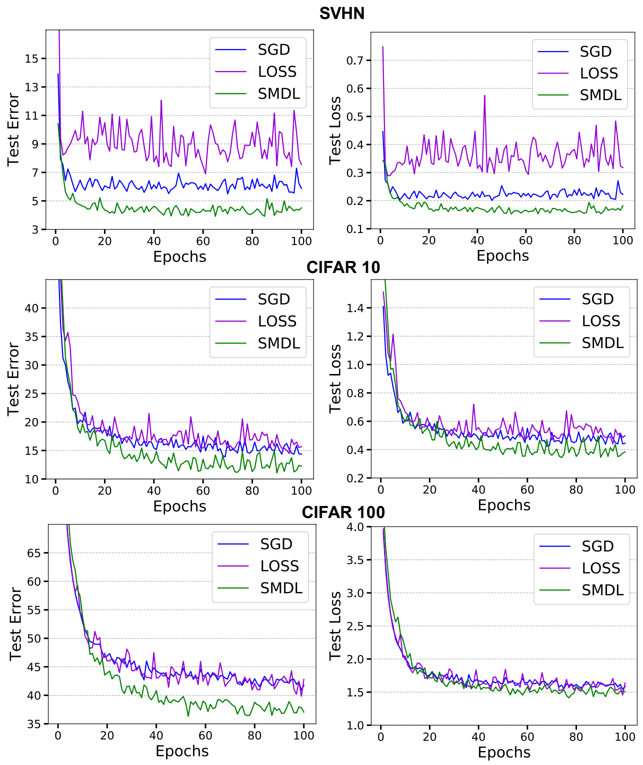


Figure 1: Test error and loss plots of the proposed method SMDL, SGD and loss based sampling scheme on multiple datasets. SMDL *consistently outperforms* the baselines, both in terms of loss and generalization performance.

Dataset		SVHN			CIFAR-10			CIFAR-100		
Method		Loss Based	SGD	SMDL	Loss Based	SGD	SMDL	Loss Based	SGD	SMDL
Accuracy(%)	Mean	90.87	93.60	**95.46**	81.06	82.54	**84.58**	53.77	53.57	**57.23**
	Final	92.44	94.34	**95.49**	84.32	85.63	**87.76**	57.22	58.27	**62.95**
Loss	Mean	0.363	0.230	**0.175**	0.590	0.535	**0.487**	1.755	1.764	**1.717**
	Final	0.318	0.215	**0.182**	0.497	0.445	**0.384**	1.639	1.586	**1.504**

Table 1: Quantitative comparison of SMDL with SGD and Loss based sampling scheme. *Mean* refers to the mean accuracy(%) across epochs and *Final* refers to the final accuracy of model.

methods like Adam (Kingma & Ba, 2014) will complement the reported results of SMDL and other baseline methods.

These results support our claim that selecting a mini-batch which respects diversity and informativeness of the samples helps in more generalizable deep learning models. We note that the models that are trained with submodular mini-batches are robust to changes in learning rate and mini-batch sizes (results in Appendix D.3). Further, we study the effect of the trade-off parameters and different distance metrics in Appendix D.1 and D.2 respectively.

5. Conclusion

In this work, we cast the selection of diverse and informative mini-batches for training a deep learning model as a submodular optimization problem. We design a novel submodular objective and propose a scalable algorithm to do submodular selection. Extensive experimental valuation on three datasets reveals significant improvement in convergence and generalization performance of the model trained with submodular mini-batches over SGD and Loss based sampling (Loshchilov & Hutter, 2015).

References

Alain, G., Lamb, A., Sankar, C., Courville, A., and Bengio, Y. Variance reduction in sgd by distributed importance sampling. *arXiv preprint arXiv:1511.06481*, 2015.

Allen-Zhu, Z. Katyusha: The first direct acceleration of stochastic gradient methods. *The Journal of Machine Learning Research*, 18(1):8194–8244, 2017.

Brahma, P. P. and Othon, A. Subset replay based continual learning for scalable improvement of autonomous systems. In *CVPR Workshops*, pp. 1179–11798. IEEE, 2018.

Chakraborty, S., Balasubramanian, V., and Panchanathan, S. Adaptive batch mode active learning. *IEEE TNNLS*, 26 (8):1747–1760, 2015.

Chang, H.-S., Learned-Miller, E., and McCallum, A. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *Advances in Neural Information Processing Systems*, pp. 1002–1012, 2017.

Das, A. and Kempe, D. Algorithms for subset selection in linear regression. In *ACM STOC*, pp. 45–54. ACM, 2008.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*, pp. 315–323, 2013.

Katharopoulos, A. and Fleuret, F. Biased importance sampling for deep neural network training. *arXiv preprint arXiv:1706.00043*, 2017.

Katharopoulos, A. and Fleuret, F. Not all samples are created equal: Deep learning with importance sampling. *arXiv:1803.00942*, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Li, C., Jegelka, S., and Sra, S. Fast dpp sampling for nystr\" om with application to kernel methods. *arXiv preprint arXiv:1603.06052*, 2016.

Li, M., Zhang, T., Chen, Y., and Smola, A. J. Efficient minibatch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 661–670. ACM, 2014.

Lin, H. and Bilmes, J. A class of submodular functions for document summarization. In *ACL*, pp. 510–520. Association for Computational Linguistics, 2011.

Loshchilov, I. and Hutter, F. Online batch selection for faster training of neural networks. *ICLR Workshops*, 2015.

Minoux, M. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization techniques*, pp. 234–243. Springer, 1978.

Mirzasoleiman, B., Karbasi, A., Sarkar, R., and Krause, A. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems*, pp. 2049–2057, 2013.

Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., Vondrák, J., and Krause, A. Lazier than lazy greedy. In *AAAI*, 2015.

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

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop*, pp. 5, 2011.

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

Shamaiah, M., Banerjee, S., and Vikalo, H. Greedy sensor selection: Leveraging submodularity. In *49th IEEE conference on decision and control (CDC)*, pp. 2572–2577. IEEE, 2010.

Singh, K. K. and Balasubramanian, V. N. *Submodular Importance Sampling for Neural Network Training*. PhD thesis, Indian Institute of Technology Hyderabad, 2018.

Thangarasa, V. and Taylor, G. W. Self-paced learning with adaptive deep visual embeddings. *arXiv preprint arXiv:1807.09200*, 2018.

Wei, K., Liu, Y., Kirchhoff, K., Bartels, C., and Bilmes, J. Submodular subset selection for large-scale speech training data. In *ICASSP*, pp. 3311–3315. IEEE, 2014.

Wei, K., Iyer, R., and Bilmes, J. Submodularity in data subset selection and active learning. In *ICML*, pp. 1954–1963, 2015.

Zhang, C., Kjellstrom, H., and Mandt, S. Determinantal point processes for mini-batch diversification. *arXiv preprint arXiv:1705.00607*, 2017.

Zhang, C., Öztireli, C., Mandt, S., and Salvi, G. Active mini-batch sampling using repulsive point processes. *arXiv:1804.02772*, 2018.

Zhao, P. and Zhang, T. Accelerating minibatch stochastic gradient descent using stratified sampling. *arXiv preprint arXiv:1405.3080*, 2014.

Zhao, P. and Zhang, T. Stochastic optimization with importance sampling for regularized loss minimization. In *ICML*, pp. 1–9, 2015.

Zhou, T. and Bilmes, J. Minimax curriculum learning: Machine teaching with desirable difficulties and scheduled diversity. In *ICLR*, 2018.

Appendix Section

A. Primer on Submodularity

Given a finite set $V = \{1, 2, \cdots, n\}$, a discrete set function $\mathcal{F} : 2^V \to \mathbb{R}$, that returns a real value for any subset $S \subseteq V$ is *submodular* if

$$\mathcal{F}(A) + \mathcal{F}(B) \geq \mathcal{F}(A \cup B) + \mathcal{F}(A \cap B) \quad \forall A, B \subseteq V \tag{9}$$

A more intuitive way of defining a submodular function is in terms of the marginal gain of adding a new element to a subset. Let $\mathcal{F}(e|S) = \mathcal{F}(e \cup S) - \mathcal{F}(S)$ denote the marginal gain of adding an element a to S. $\mathcal{F}$ is *submodular* if

$$\mathcal{F}(a|S) \geq \mathcal{F}(a|T) \qquad \forall S \subseteq T \subseteq V \setminus a \tag{10}$$

This is also called the *diminishing returns property*, where the incremental gain of adding a new element to a set decreases as the set grows from S to T. Hence, a subset that maximizes a submodular objective function $\mathcal{F}(.)$ would have least redundant elements over other subsets of the same cardinality because any redundant element will reduce the value of the submodular objective function.

A function is *monotone non-decreasing* if $\forall A \subseteq B, \mathcal{F}(A) \leq \mathcal{F}(B)$. $\mathcal{F}(.)$ is said to be *normalized* if $\mathcal{F}(\emptyset) = 0$. A greedy algorithm (Nemhauser et al., 1978) can be used to maximise a normalized monotone submodular function with cardinality constraints, with a worst-case approximation factor of $1 - \frac{1}{e}$. An instance of such a greedy algorithm can be as follows. In the i^{th} iteration, the algorithm selects an item s_i that maximizes the conditional gain, i.e. $s_i = \arg\max_{a \in V \setminus S_{i-1}} \mathcal{F}(a|S_{i-1})$. The subset S_i, initially empty, is updated as: $S_i \leftarrow \{s_i\} \cup S_{i-1}$. The algorithm terminates when S_i meets the cardinality constraint $|S_i| \leq k$.

B. Gradient Descent with Submodular Batches

The proposed batch selection strategy can work with any mini-batch gradient descent based optimization algorithms. Algorithm 2 summarizes the end-to-end training procedure.

Within each iteration in each epoch, a mini-batch S is selected using Algorithm 1 (Line 5). Lines 6 and 7 update w with a gradient descent optimizer $\pi(., \eta)$, consuming the current set S. η is the learning rate. Any differentiable loss function can be used as $L(.)$. Momentum-based and adaptive learning rate-based gradient descent methods could be used to further improve the learning based on submodular batches.

Algorithm 2 Algorithm SUBMODULAR SGD

```
Input: Training Set $V$, Optimizer $\pi(., \eta)$, # of epochs p,
    Batch size $b$, # of partitions $m$.
Output: Trained model $w_p^{\frac{|V|}{b}}$.
1: $\tau \leftarrow 1$
2: Initialize the model $w_\tau^1$.
3: for i = 1 to p do
4:     for j = 1 to $\frac{|V|}{b}$ do
5:         $S \leftarrow \text{GetMiniBatch}(V, w_\tau^j, b, m)$
6:         $\nabla J(w_\tau^j) \leftarrow \frac{\partial}{\partial w} \sum_{k \in S} L(y_k, f(x_k, w_\tau^j))$
7:         $w_\tau^{j+1} \leftarrow w_\tau^j + \pi(\{w_\tau^{1:j}\}, \{\nabla J_\tau^{1:j}\}, \eta)$
8:     $w_{\tau+1}^1 \leftarrow w_\tau^{j+1}; \tau \leftarrow \tau + 1$
9: return $w_p^{\frac{|V|}{b}}$
```

C. Implementation Details

Algorithm 2 gives the generalized training procedure for submodular mini-batch selection. In each iteration, the current model w_τ^j is used to evaluate the submodular objective score (Equation 6). The feature representation for the images is obtained from the penultimate fully connected layer of this model. The probability values ($P(y|x_i, w)$) that are used in the computation of *Uncertainty Score* is the softmax output from the model. Euclidean distance between the image features is used for *Redundancy Score* computation in Equation 3. We do an ablation study on the effect of using other distance metrics in Section D.2. *Mean Closeness Score* is computed as the cosine similarity between each data-point and the mean of all the training examples.

We follow the method used in (Brahma & Othon, 2018) and (Zhou & Bilmes, 2018) to compute the fixed feature set U used for evaluating *Feature Match Score* (Equation 5). We train a corresponding neural network, say M, on a random subset of training data, for an epoch. The features from the penultimate fully connected layer of M is used as U. Square root function is used as $g(.)$. $m_u(x_i)$ is the feature at u^{th} index of the representation of x_i from the model M.

After a grid search and an empirical study, we use the following values for the co-efficients of the terms in the objective function: $\lambda_1 = 0.2, \lambda_2 = 0.1, \lambda_3 = 0.5, \lambda_4 = 0.2$. Ablation studies of the effect of the λ parameters are presented in Section D.1. Each of the scores is individually normalized across the selected pool of samples, before being combined, to ensure fair contribution. All the score computation (which depends on the softmax output and the feature representation from fully-connected layers) is a function of the model at each iteration (Line 5, Algorithm 2). As we are using gradient descent for updating the parameters, we know that the model does not change drastically between iterations. Computational efficiency can be improved if we share the same model between successive iterations. We use a *refresh rate* of 5 for all the experiments. A study of how our method behaves with different refresh rates is shown in Figure 2(d). We note that increasing the refresh rate decreases the performance of the model. This is because the model changes over multiple iterations, which in turn affects the quality of the mini-batch selected.

We develop a modularized and configuration-driven tool in PyTorch (Paszke et al., 2017), which implements submodular selection and the other two baseline methods: SGD and Loss based sampling (Loshchilov & Hutter, 2015). All the experiments are run for 100 epochs with a batch size of 50, a momentum parameter of 0.9 and weight decay of 0.0001. For SGD, all the reported results are the average of five runs. The partition size (m in Algorithm 1 and 2) is set to 10. Code is open-sourced: https://josephkj.in/projects/SMDL

D. Ablation Studies

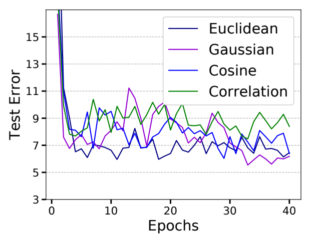
D.1. Effect of trade-off parameters.

We study the effect of trade-off parameters that control the contribution of each of the four score functions in the submodular objective function (Equation 6). For this, we train ResNet 20 on a small subset of SVHN dataset. Except for the Redundancy Score (controlled via λ_2), all the other terms are modular. Hence λ_2 should be non-zero to make the objective function submodular. We vary each of the other $\lambda_i (i \in \{1, 3, 4\})$ with values from $\{0, 0.2, 0.5, 0.8, 1.0\}$, by fixing λ_2=0.5 and rest to zero. These results are reported in row 1, 3 and 4 of Table 2. Then, we fix the best values obtained for λ_1, λ_3 and λ_4 and vary λ_2. The result is populated in the second row of Table 2. It is evident from the table that the following trade-off parameters (after normalization) achieves best performance on the subset: $\lambda_1 = 0.25, \lambda_2 = 0.25, \lambda_3 = 0.4, \lambda_4 = 0.1$. Empirically, we find that $\lambda_1 = 0.2, \lambda_2 = 0.1, \lambda_3 = 0.5, \lambda_4 = 0.2$, achieves best performance on the whole dataset. These set of trade-off parameters also generalises well to CIFAR-10 and CIFAR-100 datasets.

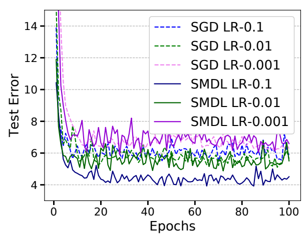
From Table 2 we can observe that each of the score function has a profound effect on the quality of the model being trained. Setting each of them to zero hurts the performance the most (Column 2 and 3). The second row reveals that the submodular term has the maximal impact if we set it zero. Each of the scores is independently competent while combining them gives the best performance.

D.2. Variations across different distance metrics

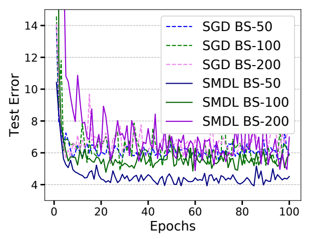
We perform an ablation study on the impact of different distance metrics while computing the value of the submodular objective function. We choose four distance metrics and evaluate its impact on training an image classifier on SVHN dataset. Assuming u and v as the two vectors, the distance metrics considered are: Euclidean ($\|u - v\|_2$), Cosine ($1 - \frac{u \cdot v}{\|u\|_2 \|v\|_2}$), Correlation ($1 - \frac{(u-\bar{u})\cdot(v-\bar{v})}{\|(u-\bar{u})\|_2 \|(v-\bar{v})\|_2}$) where $\bar{u}$ is the mean elements of vector u and Gaussian ($\exp -\frac{(\|u-v\|^2)}{2\sigma^2}$).



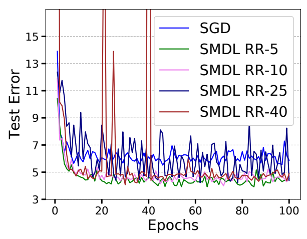
(a) Comparison with different distance metrics.



(b) Test Error plot with different learning rates (LR).



(c) Test Error plot with different mini-batch sizes (BS).



(d) Test Error plot with different Refresh Rates (RR).

Figure 2: The figure shows the ablation results on SVHN dataset. 1)We find that the Euclidean distance measure performs best among other distance metrics. 2)The proposed method SMDL consistently outperforms SGD even with different batch sizes and different learning rates. We note that a batch size of 50 and a learning rate of 0.1 gives the least error for SMDL. 3) We find that a refresh rate of 5 gives the best performance. (refer Appendix C).

	0		0.2		0.5		0.8		1.0	
	Mean	Final	Mean	Final	Mean	Final	Mean	Final	Mean	Final
λ_1	64.41	87.19	58.60	91.13	**66.99**	**92.39**	65.68	92.27	65.80	92.12
λ_2	46.17	83.62	64.55	91.34	**68.28**	**92.33**	67.70	90.77	61.34	85.06
λ_3	64.41	87.19	62.90	91.01	60.91	89.50	**63.58**	**91.02**	59.06	89.77
λ_4	64.41	87.19	62.25	**92.14**	63.03	86.90	**65.50**	91.03	60.30	86.61

Table 2: Ablation study of λ parameters on subset of SVHN dataset. *Mean* across epochs and accuracy(%) of the *Final* model is reported.

Figure 2(a) shows the result of the experiment, where the test error is plotted against epochs. These results suggest that the Euclidean distance metric gives lower test error than the others.

D.3. Effect of learning rate and batch size

Batch size and learning rate are the most important hyper-parameters that impact the learning dynamics of the model. In order to study the robustness of the model trained with submodular mini-batches, we vary batch size and learning rate, keeping all other parameters the same. We compare with SGD trained with the same set of hyper-parameters for a fair comparison.

Figure 2(b) reports the results when mini-batch sizes are set to 50, 100 and 200. Our results agree with the common consensus (Li et al., 2014) that increasing the batch-size decreases the rate of convergence, still SMDL beats SGD by a consistent margin (solid lines in the graph) for all the batch sizes. We set the learning rate to 0.1, 0.01 and 0.001 and observe the same similar pattern in Figure 2(c). These results show that the performance of SMDL is robust to learning rate and batch size changes.

E. Analysis of the computational complexity

The asymptotic complexity of selecting each minibatch using Algorithm 1 is $m \times r^2 \times d$, where m is the number of partition to which the training data is split into (Line 2 of Algorithm 1), r is the sample size of each of the Lazier than Lazy selection (Mirzasoleiman et al., 2015) and d is the dimension of the feature vector.

The divide and conquer strategy, along with the parallelism that can be achieved makes the proposed approach practically viable. The dataset V, is partitioned into m random samples in line 2 of Algorithm 1. Lazier than Lazy selection is run in parallel on multiple cores for each partition V_i to pick b samples (lines 5-8 of Algorithm 1). These $b \times m$ samples are then combined and b items are selected as mini-batch items (lines 9-14 of Algorithm 1).

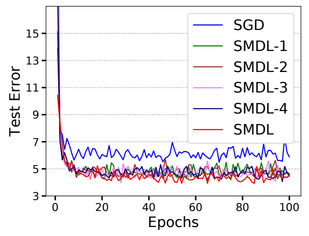
Datasets	Methods		
	SMDL	Loss	SGD
SVHN	937.2305s	7007.7064s	373.1943s
CIFAR 10	948.7460s	5196.3846s	142.5396s
CIFAR 100	764.6001s	8221.2820s	160.0396s

Table 3: Comparison of the average time for completing one epoch (in seconds) taken by SMDL against Loss based sampling and SGD.

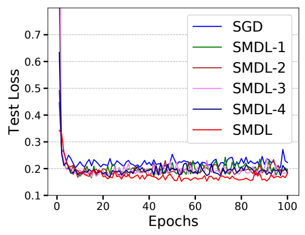
Table 3 compares average time taken for completing one epoch by SMDL, Loss based sampling (Loshchilov & Hutter, 2015) and SGD. It is evident that SMDL takes much more time than SGD but achieves better generalization capability and is much faster than other methods that accomplish the same task like Loss based sampling.

F. Additional ablation results on trade-off parameters

The submodular formulation (Equation 6), is a linear combination of four terms, controlled by trade-off parameters. We study the effect of the values for these parameters in Section D.1. We further do one more ablation to find out whether each of the term is really important for the superior performance of the proposed mini-batch selector. To study this, we train a ResNet 20 on SVHN dataset four times, with only one of the four terms set to one and others to zero. These models are labeled SMDL-{1,2,3,4} in the Figure 3. It is compared against the standard SMDL ($\lambda_1 = 0.2, \lambda_2 = 0.1, \lambda_3 = 0.5, \lambda_4 = 0.2$) and SGD.



(a) Test Error plotted across epochs.



(b) Test Loss plotted across epochs.

Figure 3: The graph brings out the importance of each of the term in the submodular formulation (Equation 6). SMDL-{1,2,3,4} are four models which has only one of the trade-off parameter turned on. It is compared against SGD and standard SMDL.

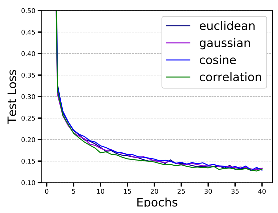
Figure 3 shows the result of the experiment. It is evident that the red line, which represents SMDL with contributions from all the terms achieves better generalization performance across epochs.

G. Test loss plots for various ablation results

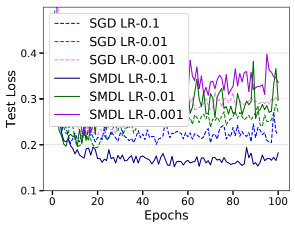
Figure 2 in Section D.3 plots the test error across epochs. Here we plot the test loss for the exact same experiments in Figure 4. We note that Euclidean distance metric works best. SMDL consistently outperforms SGD even with different batch sizes and different learning rates. A batch size of 50, learning rate of 0.1 and a refresh rate of 5 gives best performance.

H. Training error and loss comparison

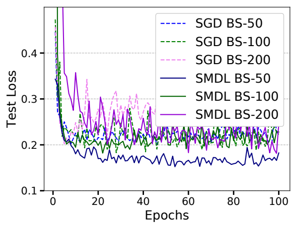
We plot the error and loss of the model on the training data across epochs in Figure 5. It is very interesting to see that SGD and Loss based sampling methods have lower training error. When we read these graphs along with the results in Figure 1, where error and loss on the test set is plotted, we can see that models trained with SMDL batch selection strategy, has higher error on the training set and lower error on the test set. This indicates that the models trained with SMDL *over-fit less* to the training data and has *better generalization* capabilities.



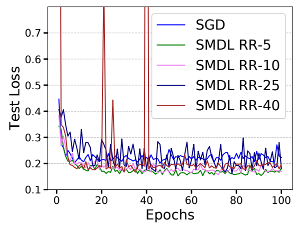
(a) Comparison with different distance metrics.



(b) Test Loss plot with different learning rates (LR).

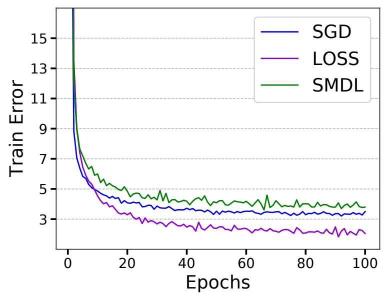


(c) Test Loss plot with different mini-batch sizes (BS).

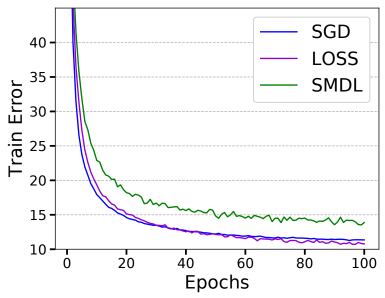


(d) Test Loss plot with different Refresh Rates (RR).

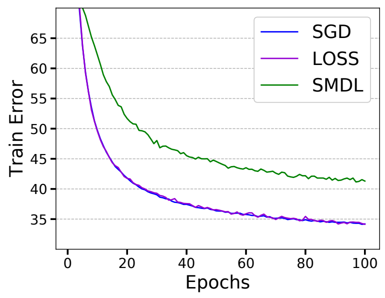
Figure 4: The figure shows the ablation results on SVHN dataset. 1)We find that the Euclidean distance measure performs best among other distance metrics. 2)The proposed method SMDL consistently outperforms SGD even with different batch sizes and different learning rates. We note that a batch size of 50 and a learning rate of 0.1 gives the least loss for SMDL. 3) We find that a refresh rate of 5 gives the best performance. (refer Section C).



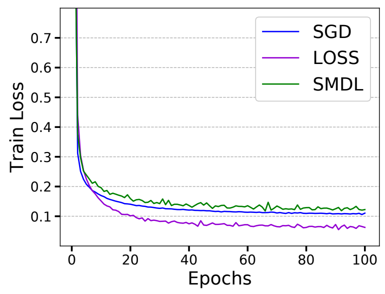
(a) Train Error on SVHN



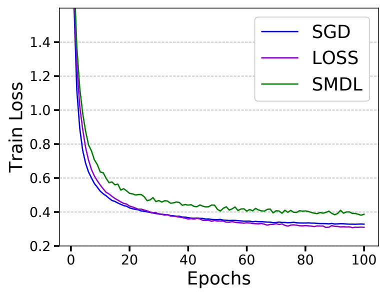
(b) Train Error on CIFAR-10



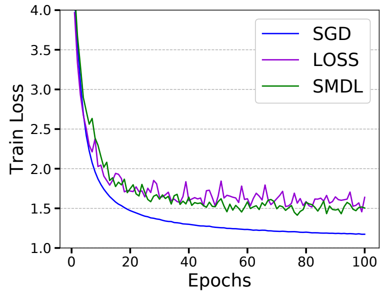
(c) Train Error on CIFAR-100



(d) Train Loss on SVHN



(e) Train Loss on CIFAR-10



(f) Train Loss on CIFAR-100

Figure 5: The figure shows how the training error and loss varies over epochs on different datasets. It is very interesting to note that on all the datasets, the training error is much higher for SMDL while the error on the test set is much lower for SMDL (Figure 1). This means that SMDL is not over-fitting and has better generalization capability than SGD and Loss based sampling (Loshchilov & Hutter, 2015).