GLISTER: Generalization based Data Subset Selection for Efficient and Robust Learning

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

[1] University of Texas at Dallas, [2] Indian Institute of Technology, Bombay

Krishnateja.Killamsetty@utdallas.edu, durgas@cse.iitb.ac.in, ganesh@cse.iitb.ac.in, rishabh.iyer@utdallas.edu

Abstract

Large scale machine learning and deep models are extremely data-hungry. Unfortunately, obtaining large amounts of labeled data is expensive, and training state-of-the-art models (with hyperparameter tuning) requires significant computing resources and time. Secondly, real-world data is noisy and imbalanced. As a result, several recent papers try to make the training process more efficient and robust. However, most existing work either focuses on robustness or efficiency, but not both. In this work, we introduce GLISTER, a GeneraLIzation based data Subset selecTion for Efficient and Robust learning framework. We formulate GLISTER as a mixed discrete-continuous bi-level optimization problem to select a subset of the training data, which maximizes the log-likelihood on a held-out validation set. We then analyze GLISTER for simple classifiers such as gaussian and multinomial naive-bayes, k-nearest neighbor classifier, and linear regression and show connections to submodularity. Next, we propose an iterative online algorithm GLISTER-ONLINE, which performs data selection iteratively along with the parameter updates, and can be applied to any loss-based learning algorithm. We then show that for a rich class of loss functions including *cross-entropy, hinge-loss, squared-loss*, and *logistic-loss*, the inner discrete data selection is an instance of (weakly) submodular optimization, and we analyze conditions for which GLISTER-ONLINE reduces the validation loss and converges. Finally, we propose GLISTER-ACTIVE, an extension to batch active learning, and we empirically demonstrate the performance of GLISTER on a wide range of tasks including, (a) data selection to reduce training time, (b) robust learning under label noise and imbalance settings, and (c) batch-active learning with a number of deep and shallow models. We show that our framework improves upon the state of the art both in efficiency and accuracy (in cases (a) and (c)) and is more efficient compared to other state-of-the-art robust learning algorithms in case (b). The code for GLISTER is at: https://github.com/dssresearch/GLISTER.

Introduction

With the quest of achieving human-like performance for machine learning and deep learning systems, the cost of training and deploying machine learning models has been significantly increasing. The wasted computational and engineering energy becomes very clear in deep learning algorithms, wherein extensive hyper-parameter tuning and network architecture search needs to be done. This results in staggering compute costs and running times[1].

As a result, efficient and robust machine learning is a very relevant and important research problem. In this paper, we shall focus on three goals: **Goal 1:** Train machine learning and deep learning models on effective subsets of data, thereby significantly reducing training time and compute, while not sacrificing accuracy. **Goal 2:** To (iteratively) select effective subsets of labeled data so as to reduce the labeling cost. **Goal 3:** Select data subsets to remove noisy labels and class imbalance, which is increasingly common in operational machine learning settings.

Background and Related Work

A number of papers have studied data efficient training and robust training of machine learning and deep learning models. However, the area of data efficient training of models that are also robust is relatively under-explored. Below, we summarize papers based on efficiency and robustness.

Reducing Training Time and Compute (Data Selection): A number of recent papers have used submodular functions as *proxy* functions (Wei, Iyer, and Bilmes 2014; Wei et al. 2014b; Kirchhoff and Bilmes 2014; Kaushal et al. 2019). These approaches have been used in several domains including speech recognition (Wei et al. 2014a,b), machine translation (Kirchhoff and Bilmes 2014) and computer vision (Kaushal et al. 2019). Another common approach uses coresets. Coresets are weighted subsets of the data, which approximate certain desirable characteristics of the full data (e.g. the loss function) (Feldman 2020). Coreset algorithms have been used for several problems including k-means clustering (Har-Peled and Mazumdar 2004), SVMs (Clarkson 2010) and Bayesian inference (Campbell and Broderick 2018). Coreset algorithms however require specialized (and often very different algorithms) depending on the model and problem at hand, and have had limited success in deep learning. A very recent coreset algorithm called CRAIG (Mirzasoleiman, Bilmes, and Leskovec 2020), which tries to select representative subsets of the training data that closely approximate the full gradient, has shown promise for several machine learning models. The resulting subset selection problem becomes an instance of facility location problem (which is submodular). Another data selection framework, which is very relevant to this work, poses the data selection problem as that of selecting a subset

[1] https://medium.com/syncedreview/the-staggering-cost-of-training-sota-ai-models-e329e80fa82

of the training data such that the resulting *model* (trained on the subset) perform well on the full dataset (Wei, Iyer, and Bilmes 2015). (Wei, Iyer, and Bilmes 2015) showed that the resulting problem is a submodular optimization problem for Nearest Neighbor (NN) and Naive Bayes (NB) classifiers. The authors empirically showed that these functions worked well for other classifiers such as logistic regression and deep models (Kaushal et al. 2019; Wei, Iyer, and Bilmes 2015).

Reducing Labeling Cost (Active Learning): Traditionally, active learning techniques like uncertainty sampling (US) and query by committee (QBC) have shown great promise in several domains of machine learning (Settles 2009). However, with the emergence of batch active learning (Wei, Iyer, and Bilmes 2015; Sener and Savarese 2018), simple US and QBC approaches do not capture diversity in the batch. Among the approaches to diversified active learning, one of the first approaches was Filtered Active Submodular Selection (FASS) (Wei, Iyer, and Bilmes 2015) that combines the uncertainty sampling method with a submodular data subset selection framework to label a subset of data points to train a classifier. Another related approach (Sener and Savarese 2018) defines active learning as a core-set selection problem and has demonstrated that the model learned over the k-centers of the dataset is competitive with respect to the one trained over the entire data. Very recently, an algorithm called BADGE (Ash et al. 2020) was proposed, that samples groups of points that have diverse and higher magnitude of hypothesized gradients to incorporate both predictive uncertainty and sample diversity into every selected batch.

Robust Learning: A number of approaches have been proposed to address robust learning in the context of noise, distribution shift and class imbalance. Several methods rely on reweighting training examples either by knowledge distillation from auxilliary models (Han et al. 2018; Jiang et al. 2018; Malach and Shalev-Shwartz 2017) or by using a clean held out validation set (Ren et al. 2018; Zhang and Sabuncu 2018). In particular, our approach bears similarity to the learning to reweight framework (Ren et al. 2018) wherein the authors try to reweight the training examples using a validation set, and solve the problem using a online meta-learning based approach.

Submodular Functions: Since several of the data selection techniques use the notion of submodularity, we briefly introduce submodular functions and optimization. Let $V = \{1, 2, \cdots, n\}$ denote a ground set of items (for example, in our case, the set of training data points). Set functions are functions $f : 2^V \rightarrow \mathbf{R}$ that operate on subsets of V. A set function f is called a submodular function (Fujishige 2005) if it satisfies the diminishing returns property: for subsets $S, T \subseteq V, f(j|S) \triangleq f(S \cup j) - f(S) \geq f(j|T)$. Several natural combinatorial functions such as facility location, set cover, concave over modular, *etc.*, are submodular functions. Submodularity is also very appealing because a simple greedy algorithm achieves a $1 - 1/e$ constant factor approximation guarantee (Nemhauser, Wolsey, and Fisher 1978) for the problem of maximizing a submodular function subject to a cardinality constraint (which most data selection approaches involve). Moreover, several variants of the greedy algorithm have been proposed which further scale up submodular maximization to almost linear time complexity (Minoux 1978; Mirzasoleiman et al. 2014, 2013).

Our Contribution

Most prior work discussed above, either study robustness or efficiency, but not both. For example, the data selection approaches such as (Wei, Iyer, and Bilmes 2015; Mirzasoleiman, Bilmes, and Leskovec 2020; Shinohara 2014) and others focus on approximating either gradients or performance on the *training sets*, and hence would not be suitable for scenarios such as label noise and imbalance. On the other hand, the approaches like (Ren et al. 2018; Jiang et al. 2018) and others, focus on robustness but are not necessarily efficient. For example, the approach of (Ren et al. 2018) requires 3x the standard (deep) training cost, to obtain a robust model. GLISTER is the first framework, to the best of our knowledge, which focuses on both efficiency and robustness. Our work is closely related to the approaches of (Wei, Iyer, and Bilmes 2015) and (Ren et al. 2018). We build upon the work of (Wei, Iyer, and Bilmes 2015), by first generalizing their framework beyond simple classifiers (like nearest neighbor and naive bayes), but with general loss functions. We do this by proposing an iterative algorithm GLISTER-ONLINE which does data selection via a meta-learning based approach along with parameter updates. Furthermore, we pose the problem as optimizing the *validation* set performance as opposed to training set performance, thereby encouraging generalization. Next, our approach also bears similarity to (Ren et al. 2018), except that we need to solve a *discrete* optimization problem instead of a meta-gradient update. Moreover, we do not run our data selection every iteration, thereby ensuring that we are significantly faster than a single training run. Finally, we extend our algorithm to the active learning scenario. We demonstrate that our framework is more efficient and accurate compared to existing data selection and active learning algorithms, and secondly, also generalizes well under noisy data, and class imbalance scenarios. In particular, we show that GLISTER achieves a **3x - 6x speedup on a wide range of models and datasets, with very small loss in accuracy**.

Problem Formulation

Notation: Denote $\mathcal{U}$ to be the full training set with instances $\{(x^i, y^i\}_{i \in \mathcal{U}}$ and $\mathcal{V}$ to be a held-out validation set $\{(x^i, y^i\}_{i \in \mathcal{V}}$. Define $L(\theta, S) = \sum_{i \in S} L(\theta, x^i, y^i)$ as the loss on a set S of instances. Denote L_T as the training loss, and hence $L_T(\theta, \mathcal{U})$ is the full training loss. Similarly, denote L_V as the validation loss (i.e. $L_V(\theta, \mathcal{V})$ as the loss on the validation set $\mathcal{V}$. In this paper, we study the following problem:

$$\underset{S \subseteq \mathcal{U}, |S| \leq k}{\operatorname{argmin}} \; L_V(\operatorname{argmin}_\theta L_T(\theta, S), \mathcal{V}) \tag{1}$$

Equation (1) tries to select a subset S of the training set $\mathcal{U}$, such that the loss on the set $\mathcal{V}$ is minimized. We can replace the loss functions L_V and L_T with log-likelihood functions LL_V and LL_T in which case, the argmin becomes argmax:

$$\underset{S \subseteq \mathcal{U}, |S| \leq k}{\operatorname{argmax}} \; LL_V(\operatorname{argmax}_\theta LL_T(\theta, S), \mathcal{V}) \tag{2}$$

Finally, we point out that we can replace L_V (or LL_V) with the training loss L_T (or log-likelihood LL_T), in which case we get the problem studied in (Wei, Iyer, and Bilmes 2015). The authors in (Wei, Iyer, and Bilmes 2015) only consider simple models such as nearest neighbor and naive bayes classifiers.

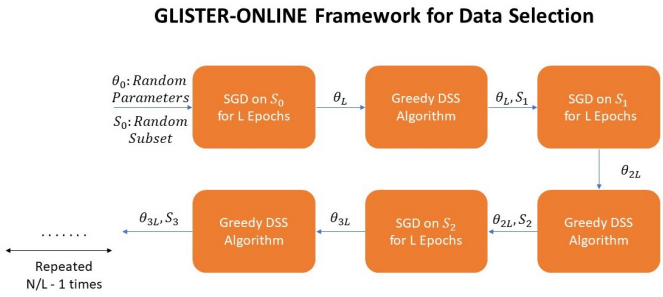


Figure 1: Main flowchart of the GLISTER-ONLINE framework for Data Selection.

Special Cases: We start with the naive bayes and nearest neighbor cases, which have already been studied in (Wei, Iyer, and Bilmes 2015). We provide a simple extension here to consider a validation set instead of the training set. First consider the naive bayes model. Let $m_{x_j,y}(S) = \sum_{i \in S} 1[x_j^i = x_j \wedge y_j = y]$ and $m_y(S) = \sum_{i \in S} 1[y_j = y]$. Also, denote $\mathcal{V}^y \subseteq \mathcal{V}$ as a set of the validation instances with label y. Furthermore, define $w(i,j) = d - ||x^i - x^j||_2^2$, where $d = \max_{i,j} ||x^i - x^j||_2^2$. We now define two submodular functions. The first is the naive-bayes submodular function $f_{NB}^v(S) = \sum_{j=1:d} \sum_{x_j \in \mathcal{X}} \sum_{y \in \mathcal{Y}} m_{x_j,y}(\mathcal{V}) \log m_{x_j,y}(S)$, and the second is the nearest-neighbor submodular function: $f_V^{NN}(S) = \sum_{y \in \mathcal{Y}} \sum_{i \in \mathcal{V}^y} \max_{s \in S \cap \mathcal{V}^{y^i}} w(i,s)$.

The following Lemma analyzes Problem (2) in the case of naive bayes and nearest neighbor classifiers.

Lemma 1. *Maximizing equation* (2) *in the context of naive bayes and nearest neighbor classifiers is equivalent to optimizing* $f_V^{NB}(S)$ *and* $f_V^{NN}(S)$ *under the constraint that* $|S \cap \mathcal{V}^y| = k\frac{|\mathcal{V}^y|}{|\mathcal{V}|}$ *with* $|S| = k$*, which is essentially a partition matroid constraint.*

This result is proved in the Appendix, and follows a very similar proof technique from (Wei, Iyer, and Bilmes 2015). However, in order to achieve this formulation, we make a natural assumption that the distribution over class labels in S is same as that of U. Furthermore, the Lemma above implied that solving equation (2) is basically a form of submodular maximization, for the NB and NN classifiers. In the interest of space, we defer the analysis of Linear Regression (LR) and Gaussian Naive Bayes (GNB) to the appendix. Both these models enable closed form for the inner problem, and the resulting problems are closely related to submodularity.

GLISTER-ONLINE Framework

In this section, we present GLISTER-ONLINE, which performs data selection jointly with the parameter learning. This allows us to handle arbitrary loss functions L_V and L_T. A careful inspection of equation (2) reveals that it is a nested bi-level optimization problem:

$$\overbrace{\underset{S \subseteq \mathcal{U}, |S| \leq k}{\text{argmax}} \; LL_V(\underbrace{\underset{\theta}{\text{argmax}} \, LL_T(\theta, S)}_{inner-level}, \mathcal{V})}^{outer-level} \tag{3}$$

The outer layer, tries to select a subset from the training set $\mathcal{U}$, such that the model trained on the subset will have the best log-likelihood LL_V on the validation set $\mathcal{V}$. Whereas in the inner layer, we optimize the model parameters by maximizing training log-likelihood LL_T on the subset selected S. Due to the fact that equation (3) is a bi-level optimization, it is expensive and impractical to solve for general loss functions. This is because in most cases, the inner optimization problem cannot be solved in closed form. Hence we need to make approximations to solve the optimization problem efficiently.

Algorithm 1 GLISTER-ONLINE Algorithm

Require: Training data $\mathcal{U}$, Validation data $\mathcal{V}$, Initial subset $S^{(0)}$ of size=k, $\theta^{(0)}$ model parameters initialization
Require: η: learning rate. T = total epochs, L = epoch interval for selection, , r = No of Taylor approximations, λ = Regularization coefficient
for all epoch t in T **do**
  if t mod $L == 0$ **then**
    $S^{(t)} = \text{GreedyDSS}(\mathcal{U}, \mathcal{V}, \theta^{(t-1)}, \eta, k, r, \lambda)$
  else
    $S^{(t)} = S^{(t-1)}$
  end if
  Perform one epoch of batch SGD to update θ_t on LL_T and using the data subset $S^{(t)}$.
end for
Output final subset $S^{(T)}$ and parameters $\theta^{(T)}$

Online Meta Approximation Algorithm Our first approximation is that instead of solving the inner optimization problem entirely, we optimize it by iteratively doing a meta-approximation, which takes a single step towards the training subset log-likelihood LL_T ascent direction. This algorithm is iterative, in that it proceeds by simultaneously updating the model parameters and selecting subsets. Figure 1 gives a flowchart of GLISTER-ONLINE. Note that instead of performing data selection every epoch, we perform data selection every L epochs, for computational reasons. We will study the tradeoffs associated with L in our experiments.

GLISTER-ONLINE proceeds as follows. We update the model parameters θ_t on a subset obtained in the last subset selection round. We perform subset selection only every L epochs, which we do as follows. At training time step t, if we take one gradient step on a subset S, we achieve: $\theta^{t+1}(S) = \theta^t + \eta \nabla_\theta LL_T(\theta^t, S)$. We can then plug this *approximation* into equation (3) and obtain the following discrete optimization problem. Define $G_\theta(S) = LL_V(\theta + \eta \nabla_\theta LL_T(\theta, S), \mathcal{V})$ below:

$$S^{t+1} = \underset{S \subseteq \mathcal{U}, |S| \leq k}{\text{argmax}} \; G_{\theta^t}(S) \tag{4}$$

Next, we show that the optimization problem (equation (4)) is NP hard.

Lemma 2. *There exists log-likelihood functions* LL_V *and* LL_T *and training and validation datasets, such that equation* (4) *is NP hard.*

The proof of this result is in the supplementary material. While, the problem is NP hard in general, we show that for many important log-likelihood functions such as the negative

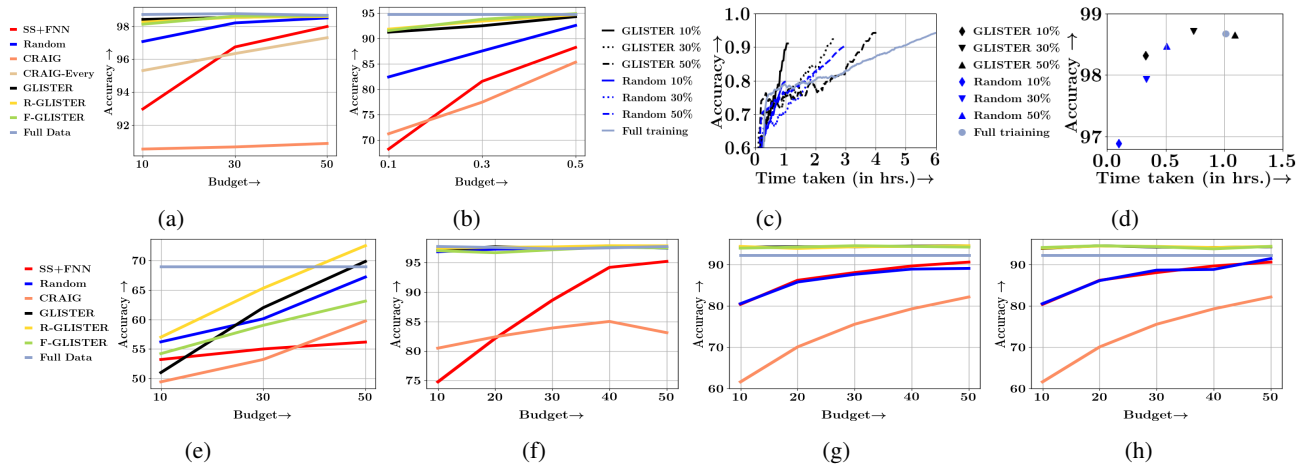


Figure 2: **Top Row**: Data Selection for Efficient Learning. (a) MNIST Accuracy vs Budgets, (b) CIFAR-10 Accuracy vs Budget , (c) CIFAR 10 Convergence plot, and (d) MNIST Accuracy vs Total time taken. **Bottom Row**: Data Selection in Class Imbalance and Noise: Accuracy vs Budget (e) CIFAR-10 (Class Imb), (f) MNIST (Class Imb), (g) DNA (Class Imb), and (d) DNA (Noise).

cross entropy, negative logistic loss, and others, $G_{\theta^t}(S)$ is submodular in S for a given θ^t.

Theorem 1. *If the validation set log-likelihood function LL_V is either the negative logistic loss, the negative squared loss, negative hinge loss, or the negative perceptron loss, the optimization problem in equation* (4) *is an instance of cardinality constrained submodular maximization. When LL_V is the negative cross-entropy loss, the optimization problem in equation* (4) *is an instance of cardinality constrained weakly submodular maximization.*

The proof of this result, along with the exact forms of the (weakly) submodular functions are in the supplementary material. Except for negative squared loss, the submodular functions for all other losses (including cross-entropy) are monotone, and hence the lazy greedy (Minoux 1978) or stochastic greedy (Mirzasoleiman et al. 2014) give $1-1/e$ approximation guarantees. For the case of the negative squared loss, the randomized greedy algorithm achieves a $1/e$ approximation guarantee (Buchbinder et al. 2014). The lazy greedy algorithm in practice is amortized linear-time complexity, but in the worst case, can have a complexity $O(nk)$. On the other hand, the stochastic greedy algorithm is linear time, i.e. it obtains a $1-1/e-\epsilon$ approximation in $O(n\log 1/\epsilon)$ iterations.

Before moving forward, we analyze the computational complexity of GLISTER-ONLINE. Denote $m=|\mathcal{V}$ (validation set size), and $n=|\mathcal{U}|$ (training set size). Furthermore, let F be the complexity of a forward pass and B the complexity of backward pass. Denote T as the number of epochs. The complexity of full training with SGD is $O(nTB)$. Using the naive (or lazy) greedy algorithm, the worst case complexity is $O(nkmFT/L+kTB)$. With stochastic greedy, we get a slightly improved complexity of $O(nmFT/L+kTB)$. Since $B\approx 2F$, and m is typically a fraction of n (like 5-10% of n), and L is a constant (e.g. $L=20$), the complexity of subset selection (i.e. the first part) can be significantly more than the complexity of training (for large values of n), thereby defeating the purpose of data selection. Below, we study a number of approximations which will make the subset selection more efficient, while not sacrificing on accuracy.

Approximations: We start with a simple approximation of just using the **Last layer approximation** of a deep model, while computing equation (4). Note that this can be done in closed form for many of the loss functions described in Theorem 1. Denote f as the complexity of computing the function on the last layer (which is much lesser than F), the complexity of stochastic greedy is reduced to $O(nmfT/L\log 1/\epsilon+kTB)$. The reason for this is that GLISTER-ONLINE tries to maximize $LL_V(\theta+\eta\sum_{j\in S}\nabla LL_T(\theta,j),\mathcal{V})$, and the complexity of evaluating LL_V is $O(mf)$. Multiplying this by the complexity of the stochastic or naive greedy, we get the final complexity. While this is better than earlier, it can still be slow since there is a factor nm being multiplied in the subset selection time. The second approximation we make is the **Taylor series approximation**, which computes an approximation $\hat{G}_\theta(S\cup e)$ based on the Taylor-series approximation. In particular, given a subset S, define $\theta^S=\theta+\eta\sum_{j\in S}\nabla LL_T(\theta,j)$. The Taylor-series approximation $\hat{G}_\theta(S\cup e)=LL_V(\theta^S)+\eta\nabla_\theta LL_T(\theta,e)^T LL_V(\theta^S,\mathcal{V})$. Note that $LL_T(\theta,e)$ can be precomputed before the (stochastic) greedy algorithm is run, and similarly $LL_V(\theta^S,\mathcal{V})$ just needs to be computed once before picking the best $e\notin S$ to add. With the Taylor series approximation, the complexity reduces to $O(k[m+n]fT/L+kTB)$ with the naive-greedy and $O([km+n\log 1/\epsilon]fT/L+kTB)$ with the stochastic greedy. Comparing to without using the Taylor series approximation, we get a speedup of $O(m)$ for naive-greedy and $O(n\log 1/\epsilon/k)$ for the stochastic greedy, which can be significant when k is much smaller than n (e.g., 10% of n). With the Taylor series approximation, we find that the time for subset selection (for deep models) is often comparable in complexity to one epoch of full training, but since we are doing subset selection only every L epochs, we can still get a speedup roughly equal to $n/k+1/L$. However, for shallow networks or 2 layer neural networks, the subset selection time can still be orders of magnitude slower than full training. In this case, we do one final approximation, which we call

the r**-Greedy Taylor Approximation**. In this case, we recompute the validation log-likelihood only r times (instead of k). In other words, we use the *stale* likelihoods for k/r greedy steps. Since we are using the same likelihood function, this becomes a simple modular optimization problem where we need to pick the top k/r values. While in principle, this approximation can be used in conjunction with stochastic greedy, we find that the accuracy degradation is considerable, mainly because stochastic greedy picks the best item greedily from $O(n/k \log 1/\epsilon)$ random data instances, which can yield poor sets if a large number of items are selected every round of greedy (which is what happens in r-greedy). Rather, we use this in conjunction with naive-greedy. The complexity of the r-taylor approximation with naive-greedy algorithm is $O(r[m+n]fT/L + kTB)$. For smaller models (like one or two layer neural network models), we set $r = 0.03k$ (we perform ablation study on r in our experiments), thereby achieving 30x speedup to just the Taylor-series approximation. For deep models, since the complexity of the gradient descent increases (i.e. B is high), we can use larger values of r, and typically set $r \approx k$. As we demonstrate in our experiments, even after the approximations presented above, we significantly outperform other baselines (including CRAIG and Random), and is comparable to full training while being much faster, on a wide variety of datasets and models.

Regularization with Other Functions Note that since the optimization problem in equation (4) is an instance of submodular optimization, we can also *regularize* this with another data-selection approach. This can be particularly useful, if say, the validation dataset is small and we do not want to overfit to the validation loss. The regularized objective function is (λ is a tradeoff parameter):

$$S^{t+1} = \text{argmax}_{S \subseteq \mathcal{U}, |S| \leq k} G_{\theta^t}(S) + \lambda R(S). \qquad (5)$$

We consider two specific kinds of regularization functions $R(S)$. The first is the supervised facility location, which is basically NN-submodular function on the training set features (Wei, Iyer, and Bilmes 2015), and the second is a random function. The random function can be thought of as a small perturbation to the data selection objective.

Implementation Aspects The detailed pseudo-code of the final algorithm is in Algorithm 1. *GreedyDSS* in Algorithm 1 refers to the greedy algorithms and approximations discussed above to solve equation (5). The parameters of GreedyDSS are: a) training set $\mathcal{U}$, validation set: $\mathcal{V}$, current parameters $\theta^{(t)}$, learning rate η, budget k, the parameter r governing the number of times we do taylor-series approximation, and finally the regularization coefficient λ. In the interest of space, we defer the detailed algorithm to the supplementary material. We use the PyTorch (Paszke et al. 2017) framework to implement our algorithms. To summarize the implementation aspects, the main hyper-parameters which govern the tradeoff between accuracy and efficiency are r, L, the regularization function R, coefficient λ and the choice of the greedy algorithm. For all our experiments, we set $L = 20$. For our deep models experiments (i.e. more than 2-3 layers), we use just the Taylor-approximation with $r = k$ and stochastic greedy, while for shallow models, we use $r \approx 0.03k$ with the naive greedy. We perform ablation studies in Section to understand the effect of these parameters in experiments, and in particular, the choices of r and L. We do not significantly tune λ in the regularized versions and just set in a way so both components (i.e. the GLISTER loss and regularizer) have roughly equal contributions. See the supplementary material for more details of the hyper-parameters used.

Convergence Analysis In this section, we study conditions under which GLISTER-ONLINE reduces the objective value, and the conditions for convergence. To do this, we first define certain properties of the validation loss. A function $f(x) : \mathcal{R}^d \rightarrow \mathcal{R}$ is said to be Lipschitz-smooth with constant $\mathcal{L}$ if $\|\nabla f(x) - \nabla f(y)\| \leq \mathcal{L}\|x - y\|, \forall x, y \in \mathcal{R}^d$. Next, we say that a function $f : \mathcal{R}^d \rightarrow \mathcal{R}$ has σ-bounded gradients if $\|\nabla f(x)\| \leq \sigma$ for all $x \in \mathcal{R}^d$

The following result studies conditions under which the validation loss reduces with every training epoch l.

Theorem 2. *Suppose the validation loss function L_V is Lipschitz smooth with constant $\mathcal{L}$, and the gradients of training and validation losses are σ_T and σ_V bounded respectively. Then the validation loss always monotonically decreases with every training epoch l, i.e. $L_V(\theta_{l+1}) \leq L_V(\theta_l)$ if it satisfies the condition that $\nabla_\theta L_V(\theta_l, \mathcal{V})^T \nabla_\theta L_T(\theta_l, S) \geq 0$ for $0 \leq l \leq T$ and the learning rate $\alpha \leq \min_l \frac{2\|\nabla_\theta L_V(\theta_l, \mathcal{V})\| \cos(\Theta_l)}{\mathcal{L}\sigma_T}$ where Θ_l is the angle between $\nabla_\theta L_V(\theta_l, \mathcal{V})$ and $\nabla_\theta L_T(\theta_l, S)$.*

The condition basically requires that for the subset selected S_i, the gradient on the training subset loss $L_T(\theta_l, S_i)$ is in the same direction as the gradient on the validation loss $L_V(\theta_l, \mathcal{V})$ at every epoch l. Note that in our Taylor-series approximation, we anyways select subsets S_i such that the dot product between gradients of subset training loss and validation loss is maximized. So, as long as our training data have some instances that are similar to the validation dataset, our model selected subset S_i should intuitively satisfy the condition mentioned in Theorem 2. We end this section by providing a convergence result.

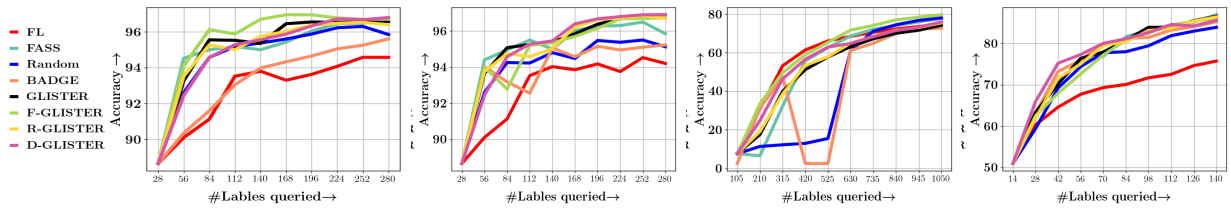
The following theorem shows that under certain conditions, GLISTER-ONLINE converges to the optimizer of the validation loss in $\mathcal{O}(1/\epsilon^2)$ epochs.

Theorem 3. *Assume that the validation and subset training losses satisfy the conditions that $\nabla_\theta L_V(\theta_l, \mathcal{V})^T \nabla_\theta L_T(\theta_l, S) \geq 0$ for $0 \leq l \leq T$, and for all the subsets encountered during the training. Also, assume that $\delta_{\min} = \min_l \frac{\nabla L_T(\theta_t)}{\sigma_G}$. Then, the following convergence result holds:*

$$\min_l L_V(\theta_l) - L_V(\theta^*) \leq \frac{R\sigma_T}{\delta_{\min}\sqrt{T}} + \frac{R\sigma_T \sum_{l=0}^{T} \sqrt{1 - \cos\Theta_l}}{T\delta_{\min}}$$

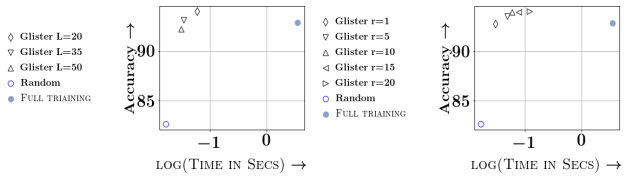
$$\textit{where} \quad \cos\Theta_l = \frac{\nabla_\theta L_T(\theta_l)^T \cdot \nabla_\theta L_V(\theta_l)}{\|\nabla_\theta L_T(\theta_l)\| \, \|\nabla_\theta L_V(\theta_l)\|}$$

Since our Taylor approximation chooses a subset that maximizes the dot-product between the gradients of the training loss and the validation loss, we expect that the angle between the subset training gradient and validation loss gradients be close to 0 i.e., $\cos\theta_l \approx 1$. Finally, note that $\delta_{\min}$ needs to be greater than zero, which is also reasonable, since having close to zero gradients on the subset gradients would imply overfitting to the training loss. In GLISTER-ONLINE we



(a) (b) (c) (d)

Figure 3: Active Learning Results: (a) SVM-Guide, (b) SVM Guide with Class Imbalance, (c) Letter, and (d) DNA Datasets.



(a) (b)

Figure 4: Ablation study comparing the effect of L and r on DNA dataset. In our experiments, we choose $L = 20$, and for our smaller datasets, we set $r = 0.03k$. The x-axis is log scale to the base e.

only train on the subset for L epochs, ensuring the training gradients on the subset do not go to zero.

GLISTER-ACTIVE Framework

In this section we extend GLISTER to the active learning setting. We propose GLISTER-ACTIVE for the mini-batch adaptive active learning where we select a batch of *B* samples to be labeled for *T* rounds. This method is adaptive because samples selected in the current round are affected by the previously selected points, as the model gets updated. The goal here is to select a subset of size *B* from the pool of unlabeled instances such that the subset selected has as much information as possible to help the model come up with appropriate decision boundary. GLISTER-ACTIVE is very similar to GLISTER-ONLINE except for three critical differences. First, the data-selection in Line 6 of Algorithm 1 is only on the unlabeled instances. Second, we use the hypothesized labels instead of the true labels in the greedy Taylor optimization (since we do not have the true labels). This is very similar to existing active learning approaches like BADGE and FASS, which also use hypothesized labels (*i.e.*, predictions from the current model). Thirdly, instead of selecting k examples every time and running only on that subset, GLISTER-ACTIVE selects a batch of B instances over the unlabeled examples and adds it to the current set of labeled examples (after obtaining the labels). Similar to GLISTER-ONLINE, we consider both the unregularized and regularized data selection objectives. In the interest of space, we defer the algorithm, along with other details to the supplementary material.

Experimental Results

Our experimental section aims to showcase the stability and efficiency of GLISTER-ONLINE and GLISTER-ACTIVE on a number of real world datasets and experimental settings. We try to address the following questions through our experiments: 1) How does GLISTER-ONLINE tradeoff accuracy and efficiency, compared to the model trained on the full dataset and other data selection techniques? 2) How does GLISTER-ONLINE work in the presence of class imbalance and noisy labels? 3) How well does GLISTER-ONLINE scale to large deep learning settings? and 4) How does GLISTER-ACTIVE compare to other active learning algorithms?

Baselines in each setting: We compare the following baselines to our GLISTER framework. We start with data selection for efficient training. **1. Random:** Just randomly select k (budget size) training examples. **2. CRAIG:** We compare against CRAIG (Mirzasoleiman, Bilmes, and Leskovec 2020) which tries to approximate the gradients of the full training sets. **3. SS + FNN:** This is the KNN submodular function from (Wei, Iyer, and Bilmes 2015), but using the training set. We do not compare to the NB submodular function, because as demonstrated in (Wei, Iyer, and Bilmes 2015) the KNN submodular function mostly outperformed NB submodular for non Naive-Bayes models. For the case of class imbalance and noisy settings, we assume that the training set is imbalanced (or noisy), while the validation set is balanced (or clean). For data selection experiments in this setting, we consider the baselines 1-3 above, i.e. CRAIG, Random and SS + FNN, but with some difference. First, we use a stronger version of the random baseline in the case of class imbalance, which ensures that the selected random set is balanced (and hence has the same class distibution as the validation set). For SS + FNN baseline, we select a subset from the training data using KNN submodular function, but using the validation set instead of the training set. In the case of noisy data, we consider 1, 2 and 4 as baselines for data selection. Finally, we consider the following baselines for active learning. **1. FASS:** FASS algorithm (Wei, Iyer, and Bilmes 2015) selects a subset using KNN submodular function by filtering out the data samples with low uncertainty about predictions. **2. BADGE:** BADGE algorithm (Ash et al. 2020) selects a subset based on the diverse gradient embedding obtained using hypothesized samples. **3. Random:** In this baseline, we randomly select a subset at every iteration for the data points to be added in the labeled examples. For data selection, we consider two variants of GLISTER, one with the Facility Location as a regularized (F-GLISTER), and the second with random as a regularizer (R-GLISTER). In the case of active learning, we add one more which is diversity regularized (D-GLISTER), where the diversity is the pairwise sum of distances.

Datasets, Model Architecture and Experimental Setup: To demonstrate effectiveness of GLISTER-ONLINE on

real-world datasets, we performed experiments on DNA, SVMGuide, Digits, Letter, USPS (from the UCI machine learning repository), MNIST, and CIFAR-10. We ran experiments with shallow models and deep models. For shallow models, we used a two-layer fully connected neural network having 100 hidden nodes. We use simple SGD optimizer for training the model. The shallow model experiments were run on the first five datasets, while on MNIST and CIFAR-10 we used a deep model. For MNIST, we use LeNet model (LeCun et al. 1989), while for CIFAR-10, we use ResNet-18 (He et al. 2016). Wherever the datasets do not a validation set, we split the training set into a train (90%) and validation set (10%).

Data Selection for Efficient Learning: We begin by studying the effect of data selection for efficiency (i.e. to reduce the amount of training time and compute). For this purpose, we compare for different subset sizes of 10%, 30%, 50% in the shallow learning setting. We demonstrate the results on MNIST and CIFAR-10 (for deep learning). In the supplementary material, we show results on the other datasets (i.e. the UCI datasets). The results are shown in Figure 2 (top row). The first two plots (i.e. a and b) show that GLISTER and its variants, significantly outperform all other baselines (including CRAIG and Random). To make CRAIG comparable to GLISTER, we run the data selection with $L = 20$ (i.e. every 20 epochs). This is mainly due to computational reasons since when run every epoch, our implementation of CRAIG was much slower than full training, effectively defeating the purpose of data selection. We also observe that with just 50% of the data, GLISTER achieves comparable accuracy to full training. We also note that the regularization offered by facility location and random selection helps, particularly for larger subsets by avoiding the over-fitting to the validation set. What is also encouraging that GLISTER-ONLINE performs well even at **very small subset sizes**, which is important for data selection (e.g. for doing hyper-parameter turnings several times on very small subsets). Perhaps surprisingly, CRAIG performs very poorly at small data-sizes. Plots c and d, show the timing results on CIFAR-10 and MNIST. We see that on CIFAR-10, GLISTER **achieves a 6x speedup at 10%, 2.5x speedup at 30%, and 1.5x speedup at 50%, while loosing 3%, 1.2% and 0.2% respectively in accuracy**. Similarly, for MNIST, **we see a 3x speedup at 10% subset, with a loss of only 0.2% in accuracy**. This timing also includes the subset selection time, thereby making the comparison fair.

Robust Data Selection: To check our model's generalization performance when adversaries are present in training data, we run experiments in class imbalance and Noisy label settings for the datasets mentioned above. We artificially generate class-imbalance for the above datasets by removing 90% of the instances from 30% of total classes available. Whereas for noisy data sets, we flip the labels for a randomly chosen subset of the data where the noise ratio determines the subset's size. In our experimental setting, we use a 30% noise ratio for the noisy experiments. The results for the class imbalance setting is shown in Figure 2 e,f and g for CIFAR-10, MNIST and DNA. The results demonstrate that GLISTER-ONLINE again, significantly outperforms the other baselines. We note that two of the baselines (random with prior and KNN-submodular with validation set information) actually have knowledge about the imbalance. It is also worth noting that for CIFAR-10, R-GLISTER achieves a significant improvement over the other baselines (by around 7%) for subset sizes 30% and 50%. Figure 2 h, shows the results on noisy setting in the DNA dataset. On the smaller dataset DNA, GLISTER and its variants outperform even full training, which is again not surprising because the full data has noise, whereas our approach essentially filters out the noise through its data selection. GLISTER and its variants outperform other baselines by more than 10% which is very significant. In the supplementary material, we provide additional results for both the class imbalance and noisy settings, and show similar gains on the other 5 smaller datasets.

Active Learning: Next, we compare GLISTER-ACTIVE to state-of-the-art active learning algorithms including FASS and BADGE. The results are shown in Figure 3 (right two plots) on SVM-Guide, Letter, and DNA datasets. Again, we see that GLISTER and its variants (particularly, with diversity) outperform the existing active learning baselines, including BADGE, which is currently the state-of-the-art batch active learning algorithm. Since BADGE and FASS have been shown to outpeform techniques like uncertainty Sampling (Settles 2009) and Coreset based approaches (Sener and Savarese 2018), we do not compare them in this work. In our supplementary material we compare our algorithms on many more datasets and also consider active learning in the class imbalance scenario.

Ablation study on the approximations and Speedups on Smaller Datasets: We conclude the experimental section of this paper, by studying the different approximations. We compare the different versions of Taylor approximations in Figure 4, and study the effect of L and r on DNA dataset. In the supplementary material, we provide the ablation study for a few more datasets. For the ablation study with L, we study $L = 20, 35$ and 50. For smaller valueds of L (e.g. below 20, we see that the gains in accuracy are not worth the reduction in effciency), while for larger values of L, the accuracy gets affected. Similarly, we vary r from 1 to 20 (which is 5% of k on DNA dataset). We observe that $L = 20$ and $r = 3\%$ of k generally provides the best tradeoff w.r.t time vs accuracy. Thanks to this choice of r and L, we see significant speedups through our data selection even with smaller neural networks. For example, we see a **6.75x speedup on DNA, 4.9x speedup on SVMGuide, 4.6x on SatImage, and around 1.5x on USPS and Letter**. Detailed ablation studies on SVMGuide, DNA, SatImage, USPS and Letter are in the supplementary.

Conclusion

We present GLISTER, a novel framework that solves a mixed discrete-continuous bi-level optimization problem for efficient training by subset selection on the train data while pivoting on a held-out validation set log-likelihood for model robustness. We study the submodularity of GLISTER for simpler classifiers and extend the analysis to our proposed iterative and online data selection algorithm GLISTER-ONLINE, that can be applied to any loss-based learning algorithm. We also extend the model to batch active learning (GLISTER-ACTIVE). For these variants of GLISTER, on a wide range of tasks, we empirically demonstrate improvements (and associated trade-offs) over state-of-the-art in terms of efficiency, accuracy and robustness.

References

Ash, J. T.; Zhang, C.; Krishnamurthy, A.; Langford, J.; and Agarwal, A. 2020. Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds. In *ICLR*.

Buchbinder, N.; Feldman, M.; Naor, J.; and Schwartz, R. 2014. Submodular maximization with cardinality constraints. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, 1433–1452. SIAM.

Campbell, T.; and Broderick, T. 2018. Bayesian Coreset Construction via Greedy Iterative Geodesic Ascent. In *International Conference on Machine Learning*, 698–706.

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

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

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

Han, B.; Yao, Q.; Yu, X.; Niu, G.; Xu, M.; Hu, W.; Tsang, I.; and Sugiyama, M. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in neural information processing systems*, 8527–8537.

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

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

Jiang, L.; Zhou, Z.; Leung, T.; Li, L.-J.; and Fei-Fei, L. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, 2304–2313.

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

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

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

Malach, E.; and Shalev-Shwartz, S. 2017. Decoupling" when to update" from" how to update". In *Advances in Neural Information Processing Systems*, 960–970.

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

Mirzasoleiman, B.; Badanidiyuru, A.; Karbasi, A.; Vondrák, J.; and Krause, A. 2014. Lazier than lazy greedy. *arXiv preprint arXiv:1409.7938* .

Mirzasoleiman, B.; Bilmes, J.; and Leskovec, J. 2020. Coresets for Data-efficient Training of Machine Learning Models. *In Proc. ICML* .

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

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

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

Ren, M.; Zeng, W.; Yang, B.; and Urtasun, R. 2018. Learning to Reweight Examples for Robust Deep Learning. In *International Conference on Machine Learning*, 4334–4343.

Sener, O.; and Savarese, S. 2018. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In *International Conference on Learning Representations*.

Settles, B. 2009. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Shinohara, Y. 2014. A submodular optimization approach to sentence set selection. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4112–4115. IEEE.

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

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

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

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

Zhang, Z.; and Sabuncu, M. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in neural information processing systems*, 8778–8788.