Adaptive Data Compression for Classification Problems

FARHAD POURKAMALI-ANARAKI1, (Member, IEEE), WALTER BENNETTE2

1Department of Computer Science, University of Massachusetts, Lowell, MA 01854 USA (e-mail: farhad_pourkamali@uml.edu)
2Information Directorate, Air Force Research Lab, Rome, NY 13441 USA (e-mail: walter.bennette.1@us.af.mil)

Corresponding author: Farhad Pourkamali-Anaraki (e-mail: farhad_pourkamali@uml.edu).

I. INTRODUCTION

A fundamental problem in machine learning theory and practice revolves around developing data compression techniques to find compact summaries that allow for training machine learning models comparable to those trained on the original data set [1]. Data summarization is noticeably helpful when confronting data sets that are too large to fit in primary memory [2] or distributed across multiple machines [3]. Another motive is to alleviate the high computational burden required by optimization problems for inference and learning. Therefore, data compression techniques have the potential to substantially reduce computational costs and memory requirements, allowing practitioners to employ off-the-shelf machine learning algorithms in resource-constrained applications with no significant loss in performance.

Data compression techniques take a wide variety of forms to select a subset of informative and representative samples for performing learning tasks on the reduced data. Among existing data subset selection techniques, coreset construction has emerged as an effective tool that involves devising sampling schemes to perform the task at hand on the selected subset, known as coreset [4], [5]. A natural choice is based on choosing samples using uniform probabilities, which we refer to as “random sampling” in this paper. However, this simple and easy-to-implement approach is more likely to ignore critical information regarding the structure of the original training data set as the coreset size becomes smaller, thus negating the potential benefits of data compression. For example, the applicability of random sampling is limited when facing imbalanced classification problems, where at least one of the target classes contains a much smaller number of samples than the other classes [6], [7].

As a result, prior works introduced various coreset construction techniques that aim to provide enhanced trade-offs between performance and efficiency compared to random sampling. These methods cover several supervised and unsupervised learning problems, such as logistic regression [8], support vector machines [9], k-means clustering [10], and Gaussian mixture modeling [11], to name a few. An essential component of these methods is to design “importance sampling” strategies, i.e., nonuniform probabilities that reveal the sensitivity score of each sample with respect to the loss function.
function that we wish to minimize [12]. More recently, a few studies have looked at the design of importance sampling schemes to accelerate the training process of neural network models via data compression (e.g., using the per sample gradient norm [13] and approximating the gradient of the whole training data set [14]).

Despite the availability of various data compression techniques, a significant limitation is that the size of the reduced data set should be provided in advance, causing a practical difficulty for users to identify the optimal exploitation of computational resources based on the unique structure of the original training data set and the learning task. This problem is exacerbated for data sets that exhibit highly skewed class distributions, nonlinear, and multiclass classification problems, requiring an exhaustive search through a set of manually-specified hyperparameter values. Although coreset construction methods are typically equipped with theoretical guarantees regarding the subset size, i.e., the number of samples in the coreset, these results involve constants dependent on unknown and difficult-to-compute parameters, such as the complexity measure defined in [8]. Hence, there is a need for developing rigorous and practical techniques to determine suitable compression ratios with minimal manual efforts (compression ratio is defined as the ratio of the reduced data size to the original training data size, thus smaller is better).

Another limitation of the prior work is that coreset construction methods are typically built around specific learning algorithms [15]. This is mainly because defining sensitivity scores relies on the choice of the loss function and optimizer. For example, practitioners cannot instantly use importance sampling schemes for logistic regression [8] when employing support vector machines or neural network models [16]. A possible workaround is to deploy clustering algorithms, such as k-medoids clustering [17]–[19], where the goal is to minimize the sum of dissimilarities between data samples belonging to a cluster and the closest cluster representative. Therefore, this approach uses all cluster representatives to summarize the input data. However, these types of methods that target the representativeness of the selected subset are model-agnostic, and their main focus is restricted to selecting samples distributed across the entire input space. Consequently, they are impractical in the presence of high-dimensional data with large sample sizes due to excessive computational overhead and the difficulty associated with finding appropriate distance metrics [20].

The discussed shortcomings of existing data compression techniques can be attributed to the absence of rigorous approaches to adapt to the unique structure of the problem at hand. To fill this gap, we propose an adaptive data compression paradigm for classification problems that build on recent advances in interactive teaching [21]. The proposed approach can be viewed as an “iterative importance sampling” scheme, where the goal is to adjust the sampling mechanism and the subset size based on the intrinsic structure of the original data set and the performance of the machine learning model trained on the reduced data. Therefore, the introduced data compression approach provides feedback to decide whether the selected data subset offers an informative summary of the data with no prior knowledge of the optimal compression ratio. Furthermore, under stringent constraints on the size of the selected subset, we introduce new strategies for coupling the proposed approach and an efficient technique to promote the diversity of the selected subset. Another significant advantage of the proposed approach compared to the prior work is its applicability to a broad range of machine learning problems, including ensemble learning, multiclass classification, class-imbalanced learning, and neural networks models.

To be precise, we introduce an interactive protocol for dynamically summarizing a given labeled data set by choosing a few samples at a time and assessing the resulting classifier’s performance. This procedure can be viewed as teaching and giving the learner a quiz to determine whether additional training examples are required. Hence, we identify influential data points for learning a classifier consistent with the entire training data set and we name the selected data subset as the “teaching set” in this paper. We also introduce post-processing techniques to further investigate the quality of the candidate samples in each round of interaction and incorporate a diversity measure before adding them to the existing teaching set. In sharp contrast to previous research, the introduced adaptive data compression approach does not require users to explicitly specify the teaching set size. Instead, the number of selected samples in each round of interaction is determined based on the learner’s feedback and two hyperparameters that are straightforward to tune, as will be shown through comprehensive experimental investigations. The proposed approach is also computationally efficient because probing the learner’s predictions in each round of interaction requires training machine learning models using only a fraction of the entire training data set.

Interactive teaching has been extensively explored in the literature in the context of active learning, specifically designed for reducing labeling efforts and costs [22]–[24]. Active learning aims to iteratively construct a small subset of labeled samples from a large pool of unlabeled data points. In each round, the teacher/user supplies labels of a predefined number of chosen examples based on an acquisition function that measures, for instance, how uncertain the trained classifier is about its predictions. Although any active learning algorithm is applicable for summarizing labeled data sets in our framework, several disadvantages exist. Active learning methods disregard the information regarding ground-truth labels of samples that are not in the teaching set. Instead, they focus on other metrics, such as the model’s uncertainty, that are more challenging to compute in various settings, e.g., high-dimensional data problems and neural network models [25]. Another complexity of active learning arises from the need to provide the precise number of selected samples in advance [26]. In contrast, our proposed adaptive approach lifts this restriction by automatically inferring the number of examples chosen across rounds of interaction (hence, we enable choosing varying numbers of samples instead
of batches of identical sizes required by active learning methods). Therefore, our proposed approach in this paper is optimized for data subset selection problems in supervised data settings, where practical interactive teaching techniques have been much less studied.

A graphical representation of the proposed adaptive data compression approach for a binary classification problem using support vector machines with the radial basis function or Gaussian kernel is shown in Figure 1 (each class forms a half-circle). The first round starts with an initial teaching set comprising five samples selected uniformly at random from each class (red triangles represent teaching examples). The learner uses the initial teaching set to train a classifier (decission boundaries are plotted using solid lines). As expected, the initial teaching set does not represent the intrinsic structure of the input data, leading to a noticeable number of misclassified samples (shown in orange). The teacher uses feedback from the learner to design a randomized sampling method integrated with a diversity measure to augment the existing teaching set. That is, the teacher identifies regions where the classifier misbehaves and thus selects informative and representative samples from these regions. Hence, the teacher provides additional samples represented by purple squares, leading to an improved but still imperfect classifier in the second round. The teacher thus offers more labeled samples without needing to set the exact number of samples in advance, which yields a classifier consistent with the entire data set as displayed on the right panel.

The rest of the paper is outlined as follows. We present the problem formulation and a review of the related work in Section II. The proposed adaptive data compression approach and post-processing techniques are introduced in Section III. In Section IV, we conduct a comprehensive set of numerical experiments using traditional classifiers and neural network models to demonstrate the effectiveness of our proposed approach on several real data sets, including imbalanced and multiclass data. Section V presents our concluding remarks.

II. BACKGROUND

In this section, we discuss the notation and preliminaries regarding data subset selection techniques for classification problems, which is the main focus of this paper. We then review previous works in the literature that are the most relevant to this paper.

A. PROBLEM FORMULATION

Consider a data set that consists of \( n \) labeled samples \( \{(x_i, y_i)\}_{i=1}^n \), where \( x_i \in \mathbb{R}^d \), \( y_i \in \{1, \ldots, r\} \), and \( r \) denotes the number of classes. Therefore, \( \mathcal{X} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d \) represents the finite instance space and \( \mathcal{Y} = \{y_1, \ldots, y_n\} \) contains the corresponding set of labels. In this paper, bold lower case letters denote vectors and calligraphic upper case letters represent sets. We also consider a set of concepts or hypothesis space \( \mathcal{H} \) so that each concept \( h \in \mathcal{H} \) takes the form \( h : \mathcal{X} \to \{1, \ldots, r\} \). For example, let us consider the linear support vector machine classifier for binary classification problems with \( r = 2 \) [27]. In this case, the hypothesis space is all hyperplanes in the form of \( w^T x + b \) that aim to separate the two classes, where the optimal values of \( w \in \mathbb{R}^d \) and \( b \in \mathbb{R} \) can be obtained by solving a constrained quadratic optimization problem [28]. The computation time and memory usage scale as a quadratic function of the data size [29], hence inferring the optimal values of \( \mathcal{W} = \{w, b\} \), or the target concept, is computationally prohibitive for data sets containing tens of thousands of samples under limited resources. We face similar computational challenges when training classifiers on large-scale data sets, motivating efforts to improve scalability.

The data subset selection problem can be viewed as searching for a subset \( \mathcal{S} \subset \mathcal{X} \), containing \( m := |\mathcal{S}| < n \) samples,
to build a classifier that is comparable to the one trained on the entire data set. Let us define the compression ratio
\[ \gamma := \frac{m}{n}, \]
where smaller values lead to higher savings in terms of computational and memory costs. Most existing data compression techniques need the subset size \( m \), or equivalently the compression ratio \( \gamma \), in advance, causing a practical difficulty for users to tune this critical hyperparameter, which is an obstacle to determining the optimal usage of computational resources. This provision is problematic because recent results in statistical learning theory reveal that the sample complexity, i.e., the minimal number of samples for finding the target concept, depends on several important factors, e.g., the data distribution and the complexity of the hypothesis space [30], which are difficult to estimate in practice. Take the linear support vector machine classifier as an example, the optimal teaching set consists of the two samples nearest to the decision boundary on either side of it when the original training data set is linearly separable. However, when encountering high-dimensional data sets in new applications, confirming the linear separability assumption and finding the optimal decision boundary are nontrivial tasks. In the following, we explain several existing data compression techniques in the literature to put our work into context.

### B. RANDOM SAMPLING

The simplest way to reduce the size of the original data set is to choose \( m \) samples uniformly at random, which we refer to as random sampling in this work. That is, we assign uniform probabilities, i.e., \( p(x) = 1/n \), to all \( n \) samples in the data set \( \mathcal{X} \) [31]. However, this general-purpose approach is oblivious to the data distribution, the availability of ground-truth labels, and the performance of the learning task at hand. Hence, for small values of the compression ratio \( \gamma \), data subset selection using random sampling leads to the unsatisfactory performance of machine learning models. This problem is exacerbated when facing complex data sets, such as class-imbalanced data, because random sampling overlooks underrepresented classes. On the other hand, increasing the subset size will adversely affect savings in computational and memory resources. As a result, random sampling typically fails to provide reasonable trade-offs between accuracy and efficiency.

### C. CORESET CONSTRUCTION

A popular coreset construction technique for improving accuracy-efficiency trade-offs centers around designing importance sampling schemes, i.e., nonuniform probabilities that reveal the sensitivity of each sample with respect to the learning task at hand. Machine learning problems often involve minimizing a loss or cost function in the form of
\[ \text{cost}(\mathcal{X}) := \sum_{x \in \mathcal{X}} f(x), \]
which is aggregating the per-sample costs and \( \mathcal{W} \) denotes the parameter set to be optimized. The reduced data set \( \mathcal{S} \) is an \( \varepsilon \)-coreset, for \( \varepsilon \in (0, 1/3) \), if for any feasible parameter values satisfies the coreset property with high probability [3]:
\[ (1 - \varepsilon)\text{cost}(\mathcal{X}) \leq \text{cost}(\mathcal{S}) \leq (1 + \varepsilon)\text{cost}(\mathcal{X}). \]  
(1)

As the above definition depends on the cost function, coreset construction algorithms are tailored for specific machine learning models. For example, two recent works studied coreset construction for logistic regression [8] and support vector machine [9] classifiers. Each importance sampling scheme is only applicable to a particular machine learning model, posing a significant challenge when we don’t have prior knowledge of what machine learning model will perform well on the given training data set. Another disadvantage of this line of work is that coreset construction methods have targeted only a small number of machine learning models. For example, we are not aware of any coreset construction algorithm for ensemble learning methods, such as random forests. The main difficulty arises from establishing the lower and upper bounds in Eq. (2) that should hold for any set of plausible parameter values.

Furthermore, the applicability of existing methods is limited because of unspecified constants and difficult-to-compute parameters in their theoretical analyses regarding the optimal coreset size. For example, the coreset construction method for logistic regression requires finding a complexity measure for the original data set known to be computationally prohibitive [8]. The authors thus designed a polynomial-time approximation method, which will make it challenging for users to identify an appropriate coreset size. Hence, theoretical results regarding the optimal coreset size do not immediately translate into practice. Moreover, worst-case bounds are typically pessimistic [32], thus inevitably ignoring the unique structure of a classification problem.

In [10], the authors presented a coreset construction algorithm for various clustering problems, such as k-means clustering [33]–[35]. Hence, the introduced importance sampling scheme can be viewed as selecting a subset of representative or diverse samples without being restricted to a specific machine learning model. Given the mean of the data set, i.e., \( c := 1/n \sum_{i=1}^{n} x_i \), the provided importance sampling score for each \( x_i \in \mathcal{X} \) consists of two parts:
\[ p(x_i) = \frac{1}{2n} + \frac{1}{2} \frac{\text{dist}(x_i, c)^2}{\sum_{x \in \mathcal{X}} \text{dist}(x, c)^2}, \]  
(2)

where \( \text{dist}(\cdot, \cdot) \) is a user-specified distance metric (Euclidean distance in the ambient space \( \mathbb{R}^d \) is a popular choice for simplicity). The first term in the above equation represents the uniform distribution for selecting all samples with a nonzero probability while the second terms places higher weights on samples that are farther from the mean. Like other coreset construction techniques, the presented theoretical bound for the coreset size \( m \) involves unspecified constants, thus tuning this critical hyperparameter remains challenging.

More recently, research efforts have been devoted to the study of data subset selection for training neural network models using stochastic gradient descent. For example, a coreset construction method was proposed to select a
weighted subset of the original data set $\mathcal{X}$ for approximating the full gradient [14]. Specifically, the main optimization problem for finding the smallest subset $\mathcal{S}$ is expressed in the following form:

$$\arg \min_{\mathcal{S} \subseteq \mathcal{X}, \alpha_j \geq 0} |\mathcal{S}|$$

s.t. $\max \left\{ \sum_{x_i \in \mathcal{X}} \nabla f_W(x_i) - \sum_{x_j \in \mathcal{S}} \alpha_j \nabla f_W(x_j) \right\} \leq \epsilon$, \hspace{1cm} (3)

where $\alpha_j$ represents the per-sample step size when using stochastic gradient descent. The constraint in this optimization problem ensures that the full gradient is approximated by an error at most $\epsilon > 0$ for all feasible values of the optimization parameters. However, this problem is computationally intractable because of computing the full gradient. To address this problem, the authors presented a greedy algorithm for applications with limited resources, which requires fixing the coreset size in advance to solve a submodular maximization problem [36] (i.e., using the constraint $|\mathcal{S}| \leq m$). Hence, like other coreset construction methods, a practical difficulty is to identify suitable values of $m$ ahead of time with no option to adapt to the unique structure of the problem at hand.

### D. ACTIVE LEARNING

Active learning methods aim to alleviate labeling costs and efforts by iteratively selecting a subset of labeled samples from a large pool of unlabeled data [22]. Hence, active learning methods do not directly fit into the list of data compression techniques in fully supervised data settings, which is this paper’s primary focus. However, we decided to compare our method against active learning strategies because of their adaptive approach to finding informative and representative samples for labeling. To be precise, given an initial labeled data set $\mathcal{L}$ and access to a large set of unlabeled data $\mathcal{U}$, active learning methods design an iterative process to choose the best sample from $\mathcal{U}$ to be labeled and added to $\mathcal{L}$. Hence, active learning techniques typically rely on the predictive uncertainty information for the elements of $\mathcal{U}$ since their corresponding ground-truth labels are assumed to be unknown. Given a trained classifier and an acquisition function $\text{acq}(x)$, active learning methods decide the next label for labeling by solving the following problem (see [25] for a comprehensive list of acquisition functions; here we use the predictive entropy [37]):

$$\arg \max_{x \in \mathcal{U}} \text{acq}(x),$$

$$\text{acq}(x) := -\sum_{l=1}^{r} p(y = l|x, \mathcal{L}) \log p(y = l|x, \mathcal{L}).$$ \hspace{1cm} (4)

A challenging aspect of applying active learning is that most existing methods select one sample in each iteration, which can be rather expensive due to the need to retrain the model (e.g., neural network models often need thousands of labeled samples). Batch-mode active learning, i.e., selecting multiple samples in each iteration, is proposed to alleviate this problem [26], [38]. However, significant challenges remain because users should still specify the acquisition function and the number of samples in advance. In contrast, our proposed work discussed in the next section allows for adjusting the size of the chosen samples in each iteration, thus removing the restriction on having batches of identical sizes.

### III. THE PROPOSED APPROACH

This section outlines an adaptive approach for finding concise summaries of labeled data sets, building on the notion of interactive teaching. The underlying idea is that the teacher or user supplies labeled samples in each round of interaction and probes the learner’s performance. We initially assign uniform probabilities to individual data points $x \in \mathcal{X}$, and we then increase them based on the difficulty level of being correctly classified in the hypothesis space $\mathcal{H}$. Therefore, our data compression approach does not require tuning vital hyperparameters such as the teaching set size (recall that we refer to the reduced data set as the teaching set in this work). Instead, our data compression approach relies on a parameter that determines how often a sample should be misclassified for being included in the teaching set. We also introduce a modified version that incorporates a diversity criterion to remove redundant instances, thus providing a principled way to guide the data summarization procedure based on the learning task at hand and the intrinsic structure of the original data. Furthermore, the proposed approach is more flexible than the prior work because the user can choose various machine learning models (e.g., support vector machines and neural networks) to perform model selection instead of being restricted to a particular model.

#### A. INTERACTIVE TEACHING FOR DATA COMPRESSION

We initialize the teaching set $\mathcal{S}$ with $n_0 \ll n$ samples chosen uniformly at random from each class for training a classifier in the form of $h : \mathcal{X} \rightarrow \{1, \ldots, r\}$. However, any arbitrary teaching set can be used for initialization as long as we have at least one sample per class. The main ingredient is to dynamically augment the teaching set by comparing predictions produced by the learner $h(x_1), \ldots, h(x_n)$ with the corresponding ground-truth labels $y_1, \ldots, y_n$. That is, given $h \in \mathcal{H}$ at each iteration, we examine whether $h(x_i)$ is consistent with the actual label $y_i$. When we have a correct prediction, we notice that adding $x_i$ to the teaching set $\mathcal{S}$ is less likely to improve the predictive power of the classifier that will be trained on the new teaching set in the subsequent round of interaction. We thus focus on instances that are more difficult concerning the current concept $h$. This procedure will be repeated until we obtain a classifier that performs well on the entire training data set or a stopping criterion is met. This paper uses a fixed number of iterations to compare our approach with the existing methods discussed in the previous section.

The detailed adaptive data compression approach is presented in Algorithm 1, which we refer to as “$\lambda$-sampling.” The importance scores $p(x_1), \ldots, p(x_n)$ are initialized using uniform probabilities and we draw $t(x_1), \ldots, t(x_n)$ from the
Algorithm 1 λ-sampling

Input: \( \mathcal{X} = \{x_1, \ldots, x_n\}, \mathcal{Y} = \{y_1, \ldots, y_n\} \), number of initial teaching samples per class \( n_0 \), threshold parameter \( \lambda > 0 \)

1: Initialize the teaching set \( S \) by selecting \( n_0 \) samples from each class
2: Set \( p(x) = 1/n \) for each \( x \in \mathcal{X} \) and sample \( t(x) \) from the exponential distribution with the scale parameter \( \lambda/n \)
3: loop
4: Learner provides \( h: \mathcal{X} \to \{1, \ldots, r\} \) using the most recent teaching set \( S \)
5: Teacher examines the learner’s performance \( \Delta(h) = \{x \in \mathcal{X}: h(x) \neq y\} \)
6: If \( \Delta(h) = \emptyset \), accept the teaching set \( S \) and the classifier \( h \) (exit the loop)
7: Double each \( p(x) \) for \( x \in \Delta(h) \)
8: If this doubling causes \( p(x) \) to exceed \( t(x) \) for the first time, add the sample \( x \) and its label \( y \) to the teaching set \( S \)
9: end loop

exponential distribution with the scale parameter \( \beta := \lambda/n \), where \( \lambda > 0 \) is a tuning parameter. Next, the teacher finds data samples that are not consistent with the learned model, i.e., \( \Delta(h) := \{x \in \mathcal{X}: h(x) \neq y\} \). Afterward, the importance scores of those samples that belong to this set will be multiplied by two and compared with \( t(x) \). If this doubling causes \( p(x) \) to exceed \( t(x) \) for the first time, we add \( x \in \Delta(h) \) to the teaching set. As a result, this step can be viewed as a quiz to identify all data samples in the disagreement region to increase their assigned importance scores. This analogy also exhibits the influence of the threshold parameter \( \lambda \) on the data summarization procedure. Since the scale parameter gives the mean of an exponentially distributed random variable, an interactive protocol with a small value of \( \lambda \) selects misclassified points more frequently than those found by setting a higher value.

To explain the impact of \( \lambda \) on the teaching set size in more detail, recall that the formula for the cumulative distribution function of an exponentially distributed random variable \( T \) is \( F(\alpha) := \text{Prob}(T \leq \alpha) = 1 - \exp(-\alpha/\beta) \) for \( \alpha \geq 0 \), where \( \beta > 0 \) is the scale parameter. Hence, in our framework, the probability of including \( x \in \mathcal{X} \) in the teaching set \( S \) can be computed as follows:

\[
\text{Prob}\left(\text{value of } p(x) \text{ exceeds } t(x)\right) = F(p(x)) = 1 - \exp\left(-\frac{1}{\beta}p(x)\right) \leq \frac{1}{\beta}p(x). \tag{5}\]

The upper bound in this equation follows from the inequality \( 1 - \exp(-\alpha) \leq \alpha \). To simply understand why this inequality holds, let us define \( g(\alpha) := \exp(-\alpha) - (1 - \alpha) \). The first derivative of \( g(\alpha) \) takes the form \( g'(\alpha) = -\exp(-\alpha) + 1 \), hence we have one critical point \( \alpha = 0 \) for which the derivative equals zero. We also have the second derivative \( g''(0) > 0 \), thus we have shown that \( g(\alpha) \geq g(0) = 0 \) and this completes the proof of the inequality. Since we defined the scale parameter \( \beta := \lambda/n \) in our framework, the upper bound in Eq. (5) can be expressed as \( (n/\lambda)p(x) \). Specifically, if the initial importance scores remain unchanged, i.e., the final value of \( p(x) = 1/n \) for each \( x \in \mathcal{X} \), the upper bound reduces to \( 1/\lambda \). Therefore, the probability of including \( x \) in the teaching set is inversely proportional to the value of the tuning parameter \( \lambda \). Therefore, the probability of including \( x \) in the teaching set is inversely proportional to the value of the tuning parameter \( \lambda \). On the other hand, given a fixed threshold parameter, the introduced algorithm progressively updates importance scores to increase the probability of storing samples that are difficult to classify.

It is worth pointing out that another recent work in interactive machine learning presented theoretical results to show that the size of the teaching set using a slightly different version of \( \lambda \)-sampling is upper bounded in expectation [39]. To be precise, the main distinction is that the prior work uses a stopping criterion based on the sum of the importance scores of samples that belong to the disagreement set \( \Delta(h) \). However, in this work, we bring the focus on practical aspects of interactive teaching for data summarization to better understand the evolution of the teaching set as a function of the number of interaction rounds. Most importantly, we notice that the \( \lambda \)-sampling algorithm does not impose any diversity measure since its primary purpose is to identify a subset of informative samples based on assessing the classifier’s performance, thus increasing the chance of having redundant samples in the teaching set. Therefore, when the objective is to summarize a data set using an informative teaching set containing as few samples as possible, the performance of \( \lambda \)-sampling is sensitive to the choice of the hyperparameter \( \lambda \). With this rationale in place, we introduce a modified version of \( \lambda \)-sampling in the next section, where we consider the original data set’s structure while monitoring the classifier’s performance as we iteratively update the teaching set \( S \).

B. INTEGRATION OF INFORMATIVENESS AND DIVERSITY MEASURES

In this section, we couple Algorithm 1 and a novel post-processing technique for two main reasons: (a) imposing a diversity criterion to eliminate redundant samples from the teaching set and (b) monitoring the interactive teaching method to determine whether adding new teaching examples is beneficial in terms of reducing the total number of erroneously classified samples. We refer to this version as “\( \lambda \)-sampling + post-processing,” which is summarized in Algorithm 2. The key idea is to find a diverse cover for ensuring that the minimum pairwise distance between the selected samples exceeds a threshold value \( \delta > 0 \).

To further explain this step, let \( T \) denote the set of candidate teaching samples in each round of interaction. We apply the \( \delta \)-pruning function to this set. That is, we randomly pick one sample and remove all other data points that fall in its \( \delta \)-radius. We continue this process until all remaining...
Algorithm 2 \(\lambda\)-sampling + post-processing

**Input:** \(\mathcal{X} = \{x_1, \ldots, x_n\}, \mathcal{Y} = \{y_1, \ldots, y_n\}\), number of initial teaching samples per class \(n_0\), threshold parameter \(\lambda > 0\), radius parameter \(\delta > 0\)

1. Initialize the teaching set \(S\) by selecting \(n_0\) samples from each class
2. Set \(p(x) = 1/n\) for each \(x \in \mathcal{X}\) and sample \(t(x)\) from the exponential distribution with the scale parameter \(\lambda/n\)
3. **loop**
   4. Learner provides \(h : \mathcal{X} \rightarrow \{1, \ldots, r\}\) using the most recent teaching set \(S\)
   5. Teacher examines the learner’s performance \(\Delta(h) = \{x \in \mathcal{X} : h(x) \neq y\}\)
   6. If \(\Delta(h) = \emptyset\), accept the teaching set \(S\) and the classifier \(h\) (exit the loop)
   7. Double each \(p(x)\) for \(x \in \Delta(h)\) and set \(T = \emptyset\)
   8. If this doubling causes \(p(x)\) to exceed \(t(x)\) for the first time, add the data sample \(x\) and its label to the set \(T\)
   9. \(T' \leftarrow \delta\)-pruning(\(T\))
10. Remove elements of \(T'\) in the \(\delta\)-radius around the members of \(S\)
11. Learner provides \(h' : \mathcal{X} \rightarrow \{1, \ldots, r\}\) based on the training data \(S \cup T'\)
12. Teacher examines the learner’s performance \(\Delta(h') = \{x \in \mathcal{X} : h'(x) \neq y\}\)
13. If \(|\Delta(h')| < |\Delta(h)|\), accept the temporary teaching set, i.e., \(S \leftarrow S \cup T'\)
14. Otherwise, select \(n_0\) samples uniformly at random from each class to update \(S\) and reduce the radius parameter by multiplying \(\delta\) by 0.9
15. **end loop**

samples are at least \(\delta\) away from each other. We call this iterative procedure to cover the candidate teaching set “\(\delta\)-pruning.” Hence, we set \(T' \leftarrow \delta\)-pruning(\(T\)). We emphasize that the introduced post-processing technique differs from prior clustering-based data compression techniques, finding representatives of the disagreement set instead of the entire input data set. As shown in our experimental results, the combination of \(\lambda\)-sampling and \(\delta\)-pruning steps lessen the need for extensive tuning of \(\lambda\). Also, from a computational perspective, our approach requires finding a cover just for a small fraction of the original data set in each round of interaction.

Furthermore, our proposed post-processing technique monitors the quality of the candidate teaching set after the pruning step by retraining the classifier using \(S \cup T'\) (note that we also ensure that the elements of \(T'\) are distant from the members of the most current teaching set \(S\)). When the size of the disagreement set decreases, we update the teaching set for the subsequent iteration. However, if the candidate teaching set gives rise to higher misclassification rates, we reject this set to select a small number of samples uniformly at random from each class. For simplicity and reducing the number of hyperparameters, we propose to select \(n_0\) samples per class in our post-processing technique. We also reduce the parameter \(\delta\) by multiplication by a scaling factor less than one (here, we choose 0.9). The rationale is that the user may select the value of the hyperparameter \(\delta\) to be too large, hence applying the \(\delta\)-pruning step reduces the effectiveness of the base \(\lambda\)-sampling algorithm. To this end, we propose to adaptively reduce the value of \(\delta\) based on the classifier’s overall performance. However, this step is optional when precise knowledge of pairwise distances is available.

### C. EXTENSION TO NEURAL NETWORK MODELS

From an algorithmic perspective, the \(\lambda\)-sampling algorithm solely requires predicting class labels for each \(x \in \mathcal{X}\) to find the disagreement set in each round of interaction. Hence, it can be seamlessly integrated with any classifier, including ensemble learning algorithms [40] and neural network models [41], [42]. However, applying the modified version presented in Algorithm 2 when using neural networks requires some attention. Note that a large class of neural networks, including feedforward and convolutional neural networks, are composed of a series of feature extraction layers followed by a fully connected layer that acts as a classifier (known as the softmax layer). Hence, the main difficulty associated with Algorithm 2 arises from the fact that obtaining a meaningful distance metric \(\text{dist}(\cdot, \cdot)\) in the original ambient space \(\mathbb{R}^d\) is not straightforward. To address this problem, we propose to use the Euclidean distance between the features extracted from the layer immediately preceding the softmax layer to impose the diversity measure [43]. That is, we use a data-driven approach for identifying an appropriate distance metric to apply the \(\delta\)-pruning step instead of relying on users to adjust this important hyperparameter. We illustrate this point in Figure 2 using a convolutional neural network consisting of a convolutional layer, pooling layer, and fully connected layer for feature extraction. In this case, softmax is used as the activation for the last layer of the classifier to produce a probability distribution for making predictions.

Depending on the choice of hyperparameters, such as small values of the initial teaching samples \(n_0\) and large values of the threshold parameter \(\lambda\), it is difficult to extract valuable features when training neural networks using minimal amounts of labeled data. In these situations, enforcing the diversity measure using \(\delta\)-pruning too early compromises the quality of the teaching set, thus adversely affecting the subsequent iterations of adaptive data compression. To tackle this challenge, we propose using the base algorithm, i.e., \(\lambda\)-sampling, for a few iterations when utilizing neural network models to enhance the embedding quality of the input data provided by the layer preceding the softmax layer. We then switch to the modified version presented in Algorithm 2 to impose the diversity criterion. Hence, the hybrid approach provides more control over the evolution of the teaching set, making possible the delicate balance between the model performance and computational savings.
IV. EXPERIMENTAL RESULTS

This section presents a comprehensive empirical evaluation of the two adaptive data compression algorithms (Algorithm 1 and 2) using a wide range of classifiers and several real data sets. We report crucial metrics, including the teaching set size, model performance, and elapsed time as a function of the number of interaction rounds. We choose 7 rounds of interaction across all experiments to demonstrate the effectiveness of our adaptive approach compared to the related work. We also consider two different values of the threshold parameter \( \lambda \in \{10, 20\} \) for calculating importance scores and fix \( n_0 = 10 \) for initializing the teaching set. Thus, we analyze the performance of our adaptive data compression approach when the size of the initial teaching set is relatively small, especially when compared to the number of samples in the original training data set.

To adjust the radius parameter \( \delta \) in Algorithm 2, we randomly select 2,000 samples from the original training data set and compute the averaged pairwise distances between them. The reasons for this selection are twofold. First, reducing the computational cost associated with finding all pairwise distances (thus, the cost will be independent of the input data size for improving scalability). Second, our goal is to show that the proposed approach is not overly sensitive to this hyperparameter, hence corroborating the practicality and flexibility of our approach in resource-constrained applications.

We divide each studied data set into training (70%) and testing/validation data (30%), where the goal is to summarize the training data set while the test data set is used for performance evaluation in each round of interaction. Note that our adaptive data compression approach does not utilize any feedback regarding the classifier’s performance on the test data set during the data summarization procedure. Since we consider challenging scenarios, such as subset selection for class-imbalanced data, we report the balanced accuracy score for all experiments, defined as the arithmetic mean of sensitivity or recall (true positive rate) and specificity (true negative rate) for binary classification problems, i.e., \( r = 2 \):

\[
\text{balanced accuracy} = \frac{1}{2} \left( \frac{\text{TP}}{\text{TP} + \text{FN}} + \frac{\text{TN}}{\text{TN} + \text{FP}} \right). \tag{6}
\]

Balanced accuracy is always between 0 and 1, and higher values of this metric indicate better prediction models. For multiclass classification problems with \( r > 2 \), balanced accuracy is the average of recall obtained on each class.

In Section IV-A, we use two traditional classifiers that directly work with the input features or attributes in \( \mathbb{R}^d \): support vector machine and random forest classifiers. We use scikit-learn implementations of these two classifiers in Python [44]. To be more specific, we use support vector machines with the radial basis function or Gaussian kernel and the regularization parameter is set to 10. This classifier aims to find (nonlinear) decision boundaries for separating the \( r \) given classes in the training data set. An interesting property of support vector machines is their ability to find a subset of training points, called support vectors, to produce the decision function. However, the computational cost and memory requirements of support vector machines scale at least quadratically with the size of the training data set. For this reason, an important task is to summarize the input data set before training the support vector machine classifier to overcome the difficulties of computational cost and memory burden. On the other hand, random forest classifiers belong to ensemble learning techniques, fitting a number of decision tree classifiers on various subsets of the training data set. We set the number of trees to 100, which is the default value in scikit-learn and used in the original work [45]. Since the studied machine learning algorithms take different approaches to classify input data sets, our experiments reveal trade-offs between accuracy and resource consumption in various settings.

In Section IV-B, we consider a convolutional neural network similar to the one depicted in Figure 2 for classifying image data consisting of convolutional and pooling layers as well as the softmax activation function in the output layer. To be precise, this network starts with a single convolutional layer with a filter size \((3, 3)\) and the number of filters is set to 32. We also use a \((2, 2)\) max pooling layer followed by a flatten layer. We then use a fully connected or dense layer comprising 100 neurons to provide features to the classifier part of the neural network model. The output layer consists of \( r \) neurons with the softmax activation function representing a probability distribution over \( r \) class labels. Thus, in this example, we adjust the parameter \( \delta \) based...
on the 100-dimensional features extracted from the dense layer before the output layer. The neural network model is implemented using the Keras library for Python. Therefore, this experiment allows us to reveal accuracy-efficiency trade-offs and illustrate the ease and applicability of the proposed adaptive compression approach for classification problems using neural networks.

A. TRADITIONAL CLASSIFIERS

In this section, we consider three real data sets: webpage, protein, and svhn. The first two data sets are used for imbalanced binary classification problems, imported from the imbalanced-learn which is an open-source Python toolbox [46]. The webpage data set contains 34,780 samples in \( \mathbb{R}^{300} \), hence the whole training data set consists of 24,346 samples with an imbalance ratio of 34.03. The imported protein data set has a total of 145,751 samples in \( \mathbb{R}^{74} \), thus the training data set contains 102,025 samples with an imbalance ratio of 111.61. Hence, these two data sets represent challenging data subset selection problems in the presence of high levels of imbalance.

The Street View House Numbers or svhn data set is a digit classification benchmark data set with \( r = 10 \) classes that contains 73,257 images of \( 32 \times 32 \) pixels, thus the training data set consists of 51,279 samples and the ambient space is \( \mathbb{R}^{1,024} \). As the primary focus of this section is to demonstrate the performance of our approach for imbalanced and multiclass classification problems, we train a convolutional neural network on the entire data set to find a low-dimensional embedding of the image data. As mentioned earlier, we evaluate the performance of our approach without this extra dimension reduction step in Section IV-B. For the svhn data set, we use two convolutional layers with a filter size \((3, 3)\) and the number of filters is 64. We also use two \((2, 2)\) max pooling layers after each convolutional layer. After flattening the filtered data, we use two dense layers with 128 and 64 neurons, respectively. Therefore, we reduce the dimension of the input data from 1,024 to 64 before starting the data summarization procedure to demonstrate the performance of our approach when facing multiclass classification problems given a reasonable feature space representation.

1) Teaching set size vs. accuracy

Figure 3 and 4 show the mean and standard deviation of the teaching set size and balanced accuracy on the webpage data set using support vector machine and random forest classifiers, respectively. In addition to the \( \lambda \)-sampling algorithm and its modification, we also consider another form of adaptive data compression, where we directly apply the \( \delta \)-pruning step to the disagreement set \( \Delta(h) \) without using the randomized importance sampling step, i.e., we remove lines 7 and 8 of of Algorithm 2, and we set \( T \leftarrow \Delta(h) \) in each round of interaction. As a result, this variant, which we call \( \delta \)-pruning in these figures, does not depend on the choice of \( \lambda \).

As expected from the discussion in Section III, we empirically observe that increasing the value of \( \lambda \) results in the reduction of the teaching set size for each algorithm (except for \( \delta \)-pruning because the randomized importance sampling part is removed). We also notice that the post-processing technique further reduces the teaching set size in all cases. To be more specific, the teaching set produced by Algorithm 2 is typically an order of magnitude smaller than its counterpart attained by Algorithm 1. Comparing balanced accuracy scores in Figure 3, we notice that the classifier’s performance trained on the resulting teaching set is on par with the baseline accuracy of using the whole training data set, which we show by a horizontal dashed line. Therefore, the two proposed adaptive compression algorithms provide interesting trade-offs between the accuracy and the size of the teaching set. When the main objective is to summarize
a data set using as few samples as possible, Algorithm 2 outperforms Algorithm 1 with a modest decrease in performance. However, Algorithm 1 is a better option when the classification accuracy is prioritized over the teaching set size. Moreover, Figure 3 reveals that just covering the disagreement region in each round of interaction, i.e., δ-pruning, provides less control over the teaching set size than Algorithm 2, highlighting the significance of integrating informativeness and diversity measures to build the teaching set.

Next, we compare the balanced accuracy scores when employing random forest classifiers in Figure 4. In this case, Algorithm 1 reaches the baseline performance of training a classifier on the entire training data set after about 4 rounds of interaction. However, the accuracy loss for Algorithm 2 is more pronounced when using random forest classifiers compared to support vector machines. This is mainly because support vector machines are more appropriate models for data compression since the underlying optimization problem automatically looks for support vectors or exemplars. However, even using the post-processing technique, the balanced accuracy score gets close to 0.80 on imbalanced data while substantially reducing the size of the teaching set, which is favored when we seek smaller teaching sets under limited resources. Another interesting observation is that the performance of Algorithm 2 with respect to the size of the teaching set is less sensitive to the choice of λ in Figure 3 and 4, which means that the user saves computation time and resources by avoiding extensive optimization of the hyperparameter λ.

We observe similar trends for training support vector machine and random forest classifiers on the protein data set in Figure 5 and 6. Although the size of the original training data set is about 4 times the number of samples in the webpage data set, we notice that the teaching set size obtained by Algorithm 2 remains almost unchanged. Specifically, for both classifiers, we see that the resulting teaching sets contain about 200 samples on average, i.e., the compression ratio γ is close to 0.002, without incurring any significant loss in performance (note that the performance of Algorithm 2 is on par with the baseline accuracy when using random forest classifiers in this case). Hence, this experiment further highlights the practicality of the proposed post-processing technique for reducing the size of the teaching set when facing large-scale classification problems.

Moreover, we analyze the effectiveness of our adaptive data compression approach for balancing class-imbalanced data sets. Although data subset selection is different from the line of work on resampling methods for tackling the class imbalance problem, such as oversampling the minority class and undersampling the majority class [47], we report the averaged imbalance ratio in each round of interaction in Figure 7 for λ = 10. Interestingly, we discern that Algorithm 2 substantially reduces the imbalance ratio in the resulting teaching sets even after one round of interaction. For example, the imbalance ratio across the two data sets and classifiers is less than 4 after 4 iterations, whereas the original imbalance ratios for webpage and protein are 34.03 and 111.61, respectively. Hence, the interactive teaching method effectively balances highly imbalanced data sets, underscoring its practicality for finding informative and representative samples when facing complex data sets.

In the next experiment, we evaluate the performance of adaptive data compression for a multiclass classification problem on the svhn data set with r = 10 categories. Based on Figure 8 and 9, we observe that the proposed post-processing technique reduces the teaching set size by almost an order of magnitude. Particularly, the size of the final teaching set is less than 1,000 across both classifiers and the two values of λ (hence, the compression ratio γ is consistently below 0.02). In terms of classification accuracy on the test data set, we see that the balanced accuracy score of Algorithm 2 exceeds 0.92 in all four experiments. On the
other hand, when the primary focus is the performance of the classifier trained on the teaching set, Algorithm 1 is a better alternative while attaining the compression ratio of about 0.2. Thus, similar to the previous experiments for binary classification problems, the proposed adaptive data compression algorithms have been found to succeed in consistently finding desirable teaching sets without specifying the precise size of the teaching set ahead of time.

2) Time complexity results
So far, we have investigated the trade-offs between the classifier’s performance and the size of the teaching set. In this section, we focus on the computational complexity of the proposed adaptive compression algorithms. To this end, we present the elapsed time as the function of the number of interaction rounds for webpage, protein, and svhn data sets in Figure 10, 11, and 12, respectively. In the following, we discuss the main findings.

- Support vector machines are computationally more expensive than random forest classifiers given a fixed training data set. The running time for training each classifier on the original data set (without any data subset selection) is shown by a horizontal dashed line.
- In Figure 10, we see that applying the \( \delta \)-pruning step to the disagreement region, i.e., removing the randomized sampling component, results in excessive computational complexity, reflecting the significance of \( \lambda \)-sampling for cost reduction. For example, the \( \delta \)-pruning variant is about an order of magnitude slower than other counterparts, even on the webpage data set with the smallest sample size among the three data sets that we study in this section. This observation also explains why we omitted this variant for larger data sets. Although extensive tuning of \( \delta \) may alleviate the computational burden of covering the disagreement region via \( \delta \)-pruning, the computational cost of hyperparameter optimization should be included for a fair comparison.
- When employing support vector machines, Algorithm 2, i.e., \( \lambda \)-sampling with the proposed post-processing technique, exhibits a reduced computational cost compared to \( \lambda \)-sampling. Specifically, on the protein data set containing more than 100,000 samples, we see that
Algorithm 1, i.e., λ-sampling, is typically more computationally efficient than the modified version with the post-processing technique. The main reason for the distinction is that training a random forest classifier is faster than support vector machines on two data sets of identical sizes. Thus, the extra cost of δ-covering leads to a moderate increase in the overall computation time. However, as we can see on the two larger data set, i.e., Figure 11 and 12, employing both algorithms for constructing teaching sets takes less time than training a single random forest classifier on the whole training set.

Finally, increasing the value of λ leads to a reduction of the overall computational cost across all experiments. As explained in Section III, the upper bound for the probability of including each sample in the teaching set is inversely proportional to λ. Hence, increasing λ accelerates the computation of the disagreement region and the δ-pruning step in each round of interaction.

3) Comparison with related work
In this section, we compare the performance of adaptive data compression with the prior work on data subset selection that we discussed in Section II. Between the two proposed algorithms, we primarily focus on Algorithm 2 with the choice of λ = 20 because of being more suitable for resource-constrained applications. We compute the mean of the teaching set size in each round of interaction to search for data subsets of identical sizes using random sampling, coreset construction, and active learning methods. Therefore, a significant advantage of our approach is removing the necessity to provide the teaching set size in each round of interaction. Since we work with two different classification algorithms, we consider the general-purpose coreset construction algorithm in [10]. As discussed in Eq. (2), this coreset construction technique does not take into account any
information regarding the cost function, so it is applicable for the data subset selection problem when using any classifier, including ensemble learning methods for which, to the best of our knowledge, no coreset construction technique exists. The active learning algorithm that we use for comparison centers on selecting one sample at a time and we use the provided implementation in [48].

Figure 13 compares efficiency and accuracy of the proposed adaptive approach with the related work on the webpage data set. As expected, the performance of random sampling is inadequate despite its high computational efficiency. In this figure, we note that the coreset construction algorithm leads to higher quality teaching sets with a modest increase in computation time. However, the performance of coreset construction is noticeably inferior to the baseline accuracy of training a classifier on the whole training data set. We also observe that active learning leads to high accuracy levels close to our proposed method while being an order of magnitude slower. Using batch-mode active learning may be an appealing alternative but the user should determine the desired number of samples in each batch. Our approach, however, infers the teaching set size in each interaction without relying on any external information.

![Figure 13](image)

FIGURE 13. Comparing the proposed adaptive data compression method with related works on the webpage data set.

Our findings in Figure 14 regarding the second imbalanced data, i.e., the protein data set, are consistent with our prior discussion. The performance of random sampling is unsatisfactory while coreset construction improving trade-offs between accuracy and efficiency. On the other hand, the accuracy of active learning is on par with our adaptive approach. Still, it is less efficient because of selecting a fixed number of samples and the need to measure predictive uncertainties. Furthermore, based on our results in Figure 5 and 6, we emphasize that when the main objective is the classifier’s performance on the teaching set, Algorithm 1 outperforms all other methods under comparison along while removing the need to adjust the teaching set size in advance.

![Figure 14](image)

FIGURE 14. Comparing the proposed adaptive data compression method with related works on the protein data set.

In the last experiment of this section, we compare the performance of data subset selection techniques on the svhn data set in Figure 15. Since the number of samples per class is balanced, random sampling performs better than before and its accuracy score is in the range of 0.88 to 0.92 for both classifiers. However, we see that the coreset construction algorithm does not provide any noticeable performance enhancement compared to random sampling. This is consistent with prior findings indicating that the advantages of nonuniform sampling techniques are minimal when facing well-balanced data sets [31], [49].

![Figure 15](image)

FIGURE 15. Comparing the proposed adaptive data compression method with related works on the svhn data set.

On the other hand, it is shown that our adaptive data com-
pression approach outperforms both random sampling and coreset construction and the overall computation time is comparable to running each classifier on the entire training data set. We also see similar trends regarding the relation between Algorithm 2 and active learning. The main disadvantage of active learning is the need to specify the number of samples in each round as opposed to inferring this quantity based on the unique structure of the problem at hand. Moreover, in Figure 8 and 9, we saw that the performance of our $\lambda$-sampling algorithm reaches 0.94 for both classifiers, thus the proposed adaptive data compression approach provides more flexibility when working with supervised data regimes, including imbalanced and multiclass classification problems.

**B. NEURAL NETWORK MODELS**

In this section, we present accuracy-efficiency trade-offs regarding the two proposed adaptive data compression algorithms when employing neural networks for classifying the input data. We consider the mnist data set, imported from Keras, containing 60,000 training samples of size $28 \times 28$ pixels with $r = 10$. As mentioned earlier, we use a neural network model that consists of a single convolutional layer with a filter size $(3, 3)$ and the number of filters is set to 32. We also use a $(2, 2)$ max pooling layer followed by a flatten layer. The layer preceding the softmax layer contains 100 neurons. A schematic representation of the convolutional neural network is provided in Figure 2. We set the number of epochs to 20 and the batch size is equal to 128.

Similar to the previous experiments, we consider two different values of $\lambda$, i.e., $\lambda \in \{10, 20\}$. Utilizing the $\lambda$-sampling algorithm is straightforward here because we only need to compare the predictions made by the neural network with the ground-truth labels. However, as we discussed in Section III-C, a significant challenge for Algorithm 2 is that the initial teaching set contains $r \times n_0 = 100$ samples, thus enforcing the diversity measure using the learned features with such a small number of samples is unrealistic. Hence, we propsoed using a hybrid approach: in the first two rounds of interaction, we skip the post-processing technique and impose the diversity measure starting the third round of interaction to extract meaningful features. Since the dense layer preceding the softmax layer contains 100 neurons, we will use the averaged pairwise Euclidean distance in $\mathbb{R}^{100}$ among 2,000 randomly selected samples to set the parameter $\delta$. Recall that we are working with a small subset of the original data to tune this parameter because of reducing computational costs and demonstrating that the performance of our adaptive approach is not sensitive to the value of this parameter.

Figure 16 reports the teaching set size as a function of the number of interaction rounds, classification accuracy, and time complexity results using the convolutional neural network on the mnist data set. We observe that increasing the value of $\lambda$ leads to smaller teaching sets for both algorithms. On the other hand, $\lambda$-sampling with post-processing is highly effective for reducing the size of the resulting teaching set in resource-constrained applications with limited memory/storage space. Specifically, employing the post-processing technique slows down the rate at which new samples are added to the teaching set. Furthermore, the classifier’s performance trained using the teaching set reaches 0.96 in the second round of interaction for both values of $\lambda$. Although Algorithm 1 performs slightly better than Algorithm 2 in terms of classification accuracy, they both get very close to the baseline accuracy of training the convolutional neural network on the entire teaching set consisting of 60,000 samples (shown by dashed lines).

In terms of the elapsed time, our adaptive data compression algorithm $\lambda$-sampling after 7 rounds of interaction takes about the same time as training the convolutional neural network on the original training data set. It is worth pointing out that in this paper, we fixed the total number of interaction rounds. However, as we can see in Figure 16, running both algorithms for 3 rounds of interaction is enough to achieve high accuracy levels. In this case, Algorithm 1 and 2 both require less time than the baseline approach to form the teaching set for training the classifier. Hence, we conclude that the $\lambda$-sampling with post-processing is a more appealing option for summarizing data sets under stringent constraints regarding the time and space requirements.

Next, we compare the performance of adaptive data compression with existing works: random sampling, coreset con-
struction, and active learning. Here, we use the coreset construction algorithm specifically designed for neural networks [14]. Based on Equation 3, this coreset construction algorithm aims to approximate the gradient of the entire training data set. Like other methods under comparison, the user should specify the teaching set size ahead of time, which is a limiting factor in real-world applications. Based on Figure 17, we see that random sampling outperforms the recent coreset construction algorithm for neural networks while being orders of magnitude faster. Furthermore, Algorithm 2 with $\lambda = 20$ provides the best trade-offs between accuracy and efficiency while inferring the teaching set size using both informativeness and diversity measures.

![FIGURE 17](image)

**FIGURE 17.** Comparing the proposed adaptive data compression approach with related works on the mnist data set.

## V. CONCLUSION

This paper introduced a practical data compression approach for classification problems to identify a subset of influential samples without a priori knowledge of the optimal subset size required by the prior work. The key ingredients of our approach are designing an iterative importance sampling scheme based on the classifier’s performance and incorporating diversity measures in a computationally efficient fashion. As shown through comprehensive experiments, the total time complexity of our approach is less than or comparable to the time required to train the classification algorithm only once on the original data set. On the other hand, the reduced data set or teaching set allows for training accurate classifiers with no significant loss in performance. Hence, our approach provides more control over accuracy-efficiency trade-offs than the prior work, and its performance is not overly sensitive to the choice of hyperparameters. In addition, our approach’s flexibility was demonstrated by using various classifiers from traditional to more recent neural networks in challenging data regimes, such as imbalanced and multiclass data. Future research directions include further evaluation of the proposed approach when employing more complex neural network models and classification problems with imperfect training labels.

## ACKNOWLEDGEMENT

Effort sponsored by the Air Force under PIA FA8750-19-3-1000 and MOU FA8750-19-3-1000. The U.S. Government is authorized to reproduce and distribute copies for Governmental purposes notwithstanding any copyright or other restrictive legends. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force or the U.S. Government.

## REFERENCES

[1] J. Bien and R. Tibshirani, “Prototype selection for interpretable classification,” *The Annals of Applied Statistics*, vol. 5, no. 4, pp. 2403–2424, 2011.

[2] J. Alafate and Y. Freund, “Faster boosting with smaller memory,” in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 11 371–11 380.

[3] H. Lu, M. Li, T. He, S. Wang, V. Narayanan, and K. Chan, “Robust coreset construction for distributed machine learning,” *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2400–2417, 2020.

[4] D. Feldman, “Core-sets: Updated survey,” in *Sampling Techniques for Supervised or Unsupervised Tasks*. Springer, 2020, pp. 23–44.

[5] P. Turner, J. Liu, and P. Rigollet, “A statistical perspective on coreset density estimation,” in *International Conference on Artificial Intelligence and Statistics*, 2021, pp. 2512–2520.

[6] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanxue, and G. Bing, “Learning from class-imbalanced data: Review of methods and applications,” *Expert Systems with Applications*, vol. 73, pp. 220–239, 2017.

[7] Z. Liu, P. Wei, J. Jiang, W. Cao, J. Bian, and Y. Chang, “MESA: Boost ensemble imbalanced learning with MEta-Sampler,” in *Advances in Neural Information Processing Systems*, 2020, pp. 14 463–14 474.

[8] A. Munteanu, C. Schwiegelshohn, C. Sohler, and D. Woodruff, “On coresets for logistic regression,” in *Neural Information Processing Systems*, 2018, pp. 6561–6570.

[9] M. Tukan, C. Baykal, D. Feldman, and D. Rus, “On coresets for support vector machines,” in *International Conference on Theory and Applications of Models of Computation*, 2020, pp. 287–299.

[10] O. Bachem, M. Lucic, and A. Krause, “Scalable K-means clustering via lightweight coresets,” in *International Conference on Knowledge Discovery and Data Mining*, 2018, pp. 1119–1127.

[11] M. Lucic, M. Faulkner, A. Krause, and D. Feldman, “Training Gaussian mixture models at scale via coresets,” *Journal of Machine Learning Research*, vol. 18, no. 1, pp. 1–25, 2018.

[12] A. Raj, C. Musco, and L. Mackey, “Importance sampling via local sensitivity,” in *International Conference on Artificial Intelligence and Statistics*, 2020, pp. 3099–3109.

[13] A. Katharopoulos and F. Fleuret, “Not all samples are created equal: Deep learning with importance sampling,” in *International conference on machine learning*, 2018, pp. 2525–2534.

[14] B. Mirzasoleiman, J. Bilmes, and J. Leskovec, “Coresets for data-efficient training of machine learning models,” in *International Conference on Machine Learning*, 2020, pp. 6950–6960.

[15] D. Feldman, “Introduction to core-sets: an updated survey,” *arXiv preprint arXiv:2011.09384*, 2020.

[16] A. Krizhevsky, I. Sutskever, and G. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[17] J. Neuling and F. Fleuret, “A sub-quadratic exact medoid algorithm,” in *Artificial Intelligence and Statistics*, 2017, pp. 185–193.

[18] P. Hajibabaei, F. Pourkamali-Anaraki, and M. Hariri-Ardabili, “Kernel matrix approximation on class-imbalanced data with an application to scientific simulation,” *IEEE Access*, vol. 9, pp. 83 579–83 591, 2021.

[19] A. Ushakov and I. Vasilyev, “Near-optimal large-scale k-medoids clustering,” *Information Sciences*, vol. 545, pp. 344–362, 2021.

[20] F. Pourkamali-Anaraki, S. Becker, and M. Wakin, “Randomized clustered Nyström for large-scale kernel machines,” in *AAAI Conference on Artificial Intelligence*, 2018, pp. 3960–3967.

[21] F. Cicalese, S. Filho, E. Labor, and M. Molinaro, “Teaching with limited information on the learner’s behaviour,” in *International Conference on Machine Learning*, 2020, pp. 2016–2026.

[22] B. Settles, *Active Learning*. Morgan & Claypool Publishers, 2012.

[23] K. Wang, D. Zhang, Y. Li, R. Zhang, and L. Lin, “Cost-effective active learning for deep image classification,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 12, pp. 2591–2600, 2016.

[24] Y. Yang and M. Loog, “A benchmark and comparison of active learning for logistic regression,” *Pattern Recognition*, vol. 83, pp. 401–415, 2018.
[25] Y. Gal, R. Islam, and Z. Ghahramani, “Deep bayesian active learning with image data,” in *International Conference on Machine Learning*, 2017, pp. 1183–1192.

[26] A. Kirsch, J. van Amersfoort, and Y. Gal, “BatchBALD: Efficient and diverse batch acquisition for deep bayesian active learning,” in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 7026–7037.

[27] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[28] B. Schölkopf and A. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[29] M. Nandan, P. Khargonekar, and S. Talathi, “Fast SVM training using approximate extreme points,” *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 59–98, 2014.

[30] A. Kumar, H. Zhang, A. Singla, and Y. Chen, “The teaching dimension of kernel perceptron,” in *International Conference on Artificial Intelligence and Statistics*, 2021, pp. 2071–2079.

[31] S. Kumar, M. Mohri, and A. Talwalkar, “Sampling methods for the Nyström method,” *Journal of Machine Learning Research*, vol. 13, pp. 981–1006, 2012.

[32] M. Lopes, B. Ericsson, and M. Mahoney, “Error estimation for sketched svd via the bootstrap,” in *International Conference on Machine Learning*, 2020, pp. 6382–6392.

[33] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, no. 14, 1967, pp. 281–297.

[34] F. Castro, M. Marin-Jimenez, N. Guil, and N. de la Blanca, “Multimodal feature fusion for cnn-based gait recognition: an empirical comparison,” *Neural Computing and Applications*, vol. 32, no. 17, pp. 14 173–14 193, 2020.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, and V. Dubourg, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[36] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[37] G. Lemaitre, F. Nogueira, and C. Aridas, “Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning,” *Journal of Machine Learning Research*, vol. 18, no. 1, pp. 559–563, 2017.

[38] A. Fernández, S. Garcia, F. Herrera, and N. Chawla, “SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary,” *Journal of Artificial Intelligence Research*, vol. 61, pp. 863–905, 2018.

[39] T. Danka and P. Horvath, “modAL: A modular active learning framework for Python,” available on arXiv at https://arxiv.org/abs/1805.00979, 2018, accessed: 2021-08-17.

[40] T. Popordanoska, M. Kumar, and S. Teso, “Machine guides, human supervises: Interactive learning with global explanations,” arXiv preprint arXiv:2009.09723, 2020.