This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2024.3506494

CheckSelect: Online Checkpoint Selection for Flexible, Accurate, Robust, and Efficient Data Valuation

Soumi Das, Manasvi Sagarkar, Suparna Bhattacharya, and Sourangshu Bhattacharya

***Abstract*—In this paper, we argue that data valuation techniques should be *flexible, accurate, robust, and efficient* (FARE). Here, accuracy and efficiency refer to the notion of identification of most important data points in less time compared to a full training. Flexibility refers to the ability of the method to be used with various value functions, while robustness refers to the ability to be used with different data distributions from a related domain. We propose a two-phase approach towards achieving these objectives, where the first phase, checkpoint selection, extracts important model checkpoints while training on a related dataset, and the second data valuation and subset selection phase extracts the high-value subsets. A key challenge in this process is to efficiently determine the most important checkpoints during the training, since the total value function is unknown. We pose this as an online sparse approximation problem and propose a novel online orthogonal matching pursuit algorithm for solving it. Extensive experiments on standard datasets show that CheckSelect provides the best accuracy among the baselines while maintaining efficiency comparable to state of the art. We also demonstrate the flexibility and robustness of CheckSelect on a standard domain adaptation task, where it outperforms existing methods in data selection accuracy without the need to re-train on the full target-domain dataset.**

***Impact Statement*—Recent times are witnessing a proliferation of data from multiple sources leading to redundancies, bias, and noise in the collected data. Training AI models using such a large pool of noisy data leads to wastage of computing resources. Existing data valuation and subset selection (DVSS) techniques are either too expensive or do not offer high-end task performance for smaller-sized selected datasets. The current work describes a novel DVSS technique that is (a) *efficient* in terms of time, (b) *accurate* and *robust* in terms of performance and, (c) *flexible* in terms of its adaptive property leading to its usage in domain adaptation applications. We empirically show that the proposed method outperforms several SOTA baselines by a significant margin of maximum upto $\sim 30\%$.**

***Index Terms*—Online Sparse Approximation, Online checkpoint selection, Data valuation.**

I. Introduction

Soumi Das was a PhD student at the Indian Institute of Technology, Kharagpur, and is now a Post-Doctoral researcher at Max Planck Institute for Software Systems, Saarbruecken, Germany (e-mail: soumidas@mpi-sws.org).

Manasvi Sagarkar was a Masters student at the Indian Institute of Technology, Kharagpur, and is now a Data Scientist at Indeed, Bangalore, India (e-mail: manasvisagarkar@gmail.com).

Suparna Bhattacharya is an HPE Fellow in the AI Research Lab at Hewlett Packard Labs (e-mail: suparna.bhattacharya@hpe.com).

Sourangshu Bhattacharya is an Associate Professor at the Indian Institute of Technology, Kharagpur (e-mail: sourangshu@cse.iitkgp.ac.in). He is thankful to DST Core Research Grant, Govt. of India and Hewlett Packard Enterprise India for their support.

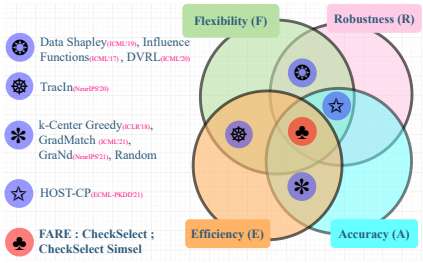

Fig. 1: **DVSS techniques binned as per FARE aspects.**

WITH the widespread use of deep learning models in many computer vision and other applications, re-training of such models on newly collected data has become commonplace. Redundancies and noise in the collected datasets lead to inefficiencies in the re-training, which has led to development and popularity of *data valuation* and *subset selection* (DVSS) techniques[1], [2], [3], [4]. To be effective, DVSS techniques must be (1) **efficient** – not be more computationally expensive than the re-training process, and (2) **accurate** – models trained on relatively few selected data points should perform accurately on test data points. Many early DVSS techniques, e.g. random selection, and facility location - based techniques [5] perform well on these metrics but fail on the crucial metrics of: (3) **flexibility** – ability of the user to incorporate any user defined performance criterion, e.g. accuracy on a specific class, and (4) **robustness** – towards changes in the data distribution of the collected data, e.g. imbalance in classes, change of domain, etc. Figure 1 shows the FARE properties of recently developed DVSS techniques. While some of the recent DVSS techniques e.g. influence functions [6], data shapley [7], and DVRL [3] address the problem of flexibility, their efficiency – accuracy tradeoff makes them impractical to be used even with moderately large sized datasets ($\sim$ 100000 images). On the other hand, recently proposed gradient-norm-based techniques e.g. GraNd [1], CRAIG [8], GradMatch [2], etc. perform well on accuracy – efficiency tradeoff, but perform poorly on the flexibility and robustness aspects. Another DVSS approach using differentiable convex programming [9] performs well on the three aspects of accuracy, robustness, and flexibility, but has poor efficiency.

In this paper, we propose a DVSS technique that performs well on all of the criteria: *Flexibility, Accuracy, Robustness, and Efficiency (FARE)* (Figure 1). The key idea behind our approach is to split the DVSS process into two phases: (1)

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2024.3506494

CheckSelect – which collects informative model checkpoints from a training run on the same or a related dataset. and (2) a data valuation phase which uses the stored model checkpoints to compute values for newly collected data points. Hence our method can be used in full training mode, or re-training mode with available checkpoints from a previous training run. While the full-training mode has a time complexity similar to state-of-the-art (SOTA) DVSS techiniques, the re-training mode is substantially faster than existing DVSS techniques (except random selection). We note that a recent DVSS method, TracIn [4] uses randomly stored checkpoints for data valuation, and can be used in two-modes. However, their accuracy is not on par with the SOTA DVSS techniques.

The most informative checkpoints during the first phase are defined as updates that result in the best approximation to the *value function*, which is defined here as the decrease in the validation-set loss function during training. Since the total value function can be estimated as additive changes with each training update, the problem of estimating most significant updates (and hence checkpoints) can be posed as an online version of sparse approximation problem [10]. A key difference is that in our case the true value function is only available at the end of the training process. Hence, commonly used greedy approaches for sparse approximation e.g. orthogonal matching pursuit (OMP) [11] cannot be applied directly. This leads us to the online sparse approximation setting [12], [13], which is surprisingly not very well studied. One of the main contributions of this paper is to propose *CheckSelect*, an *online sparse approximation* algorithm for checkpoint selection. The key idea is to replace an existing selected checkpoint with the current checkpoint if its projection on the current residual exceeds that of the existing selected checkpoint. Another drawback of data valuation-based subset selection algorithms is that the selected subset often lacks diversity [9]. Our second contribution is an algorithm for the data-valuation phase, called *Simsel*, which selects the most diverse subset of the training data, where similarity between two training points is defined using their contributions to value estimation of each validation data point. We use a submodular subset selection algorithm for computing the subsets [14].

We empirically compare CheckSelect with a wide array of SOTA methods, on the accuracy–efficiency aspects, as well as the aspect of robustness. Results indicate that the proposed method outperforms all the baseline methods in terms of accuracy, without compromising on efficiency. Moreover, CheckSelect is more robust compared to the DVSS techniques with similar accuracy and efficiency, e.g. k-center greedy and GraNd, since it shows lower variance and higher resilience to class imbalance. Finally, we demonstrate both flexibility and robustness of CheckSelect on the standard domain adaptation task of office-home dataset [15], where data selected using checkpoints from the source dataset and valuation on the target dataset outperforms baselines on the domain adaptation task. To summarize, our main contributions are:

- We propose a DVSS method that performs well on all the four criteria (*FARE*) where we split the process into two phases - (1) checkpoint selection and (2) data valuation.
- We propose *CheckSelect* that is an online sparse approximation algorithm for the first phase while the second phase of data valuation is derived from the first.
- We propose an algorithm, *Simsel* that is used for diverse subset selection, lacking in the current DVSS techniques.

II. Related Work

Finding influential training datapoints or data valuation/selection has become a recent research challenge. There have been several works in the literature of data valuation encompassing influence functions [6], shapley values [7], differentiable convex programming [9], tracking training trajectories [4] and more. Alongside, a number of coreset/data selection works like [5],[1],[2] have emerged in the recent past which follows different methodologies as described in the last section. A brief review of such methods have been well described in the works of [16]. Clustering-based methods [17], [18] run unsupervised training on the whole dataset, by inducing labels on cluster centers, hence not being selection algorithms. The other line of work surrounding dataset distillation [19] synthesize datapoints during training instead of finding a representative set, thus not turning out to be fair comparisons.

Based on our studies, the techniques for data subset selection follow two broad approaches: (1) ones that learn complex functions as selection criteria and are jointly trained with the end-task, e.g. [20],[21], etc.; and (2) others that are designed to optimize relatively simple objective functions (e.g. facility location [22]). Such techniques use various discrete and continuous optimization approaches e.g. convex optimization [23], submodular optimization [8], and orthogonal matching pursuit (OMP) algorithms [2]. The first class of techniques are typically computationally expensive, while the latter ones can work in practical scenarios. In this paper, we adopt the OMP-based approach, owing to its simplicity and efficiency. The main drawback of existing OMP-based techniques for DVSS [2], is that they require the final predicted outcome to be available at the time of greedy selection of datapoints. On the other hand, TracIn [4] valuates datapoints based on only uniformly selected checkpoints. A few works on online OMP algorithms exist [13] [12], which mainly add instances independent of past queried instances, in an online streaming setting. On the contrary, the online OMP-based technique proposed here tracks the already selected checkpoints to make an informed decision on the incoming ones. To the best of our knowledge, ours is one of the first works to develop an algorithm for the selection of checkpoints alongside the training procedure and use them to evaluate training datapoints making the process efficient.

III. Online Checkpoint Selection for DVSS

In this section, we describe the proposed data valuation method *CheckSelect* – a two-phase DVSS method. It uses selected checkpoints from a training run of stochastic gradient descent (SGD) algorithm to approximate the value of a training datapoint. The first phase (**CheckSelect**) selects checkpoints from the run of an SGD training algorithm, which are used to approximate the overall value function. The checkpoints are

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2024.3506494

selected in an online manner using a variant of the orthogonal matching pursuit algorithm (OMP) for sparse approximation. The second phase (**SimSel**) computes a high-value subset of the training datapoints using the approximated contributions of each training datapoint toward each validation datapoint. The main motivation behind the two-phase approach is to enable the scalability of the method in a real-life training setting and the use of training runs from related training tasks to improve selection efficiency on related tasks. Table I shows the notations used in the paper and their explanations.

A. Background and Problem Definition

Let $\mathcal{D} = \{d_i = (u_i, w_i) | i = 1, 2, .., N\}$ be the training dataset for a supervised learning task with features $u_i \in \mathcal{U}$ and labels $w_i \in \mathcal{W}$. Also, let $\mathcal{D}' = \{d'_j = (u'_j, w'_j) | j = 1, 2, .., M\}$ be the validation dataset which is used for assigning values/scores to a training datapoint $d \in \mathcal{D}$. Data valuation assigns a score $v(d)$ to a training datapoint d such that $\sum_{i=1}^{N} v(d_i) = \mathcal{V}(\mathcal{D})$, which is the total value of the training dataset. A commonly used valuation function is the total loss on a validation dataset $\mathcal{D}'$ [7]. TracIn [4] uses the trajectory of SGD updates to define a surrogate value function – the decrease in loss during the training. Let $\theta_1, ..., \theta_T$ be a sequence of parameters generated by training with a given deep learning model on training dataset $\mathcal{D}$ using a mini-batch SGD-like algorithm for T epochs. Given a validation datapoint $d' \in \mathcal{D}'$, the first order approximation for the per epoch decrease in loss, $\mathcal{V}_t(d')$ is given by [4]: $\mathcal{V}_t(d') = l(\theta_{t+1}, d') - l(\theta_t, d') = \nabla l(\theta_t, d').(\theta_{t+1} - \theta_t)$, where $l(\theta_t, d')$ is the loss on d' using end-of-epoch model parameter θ_t. The influence of any training point d on the validation datapoint d', termed as *TracIn*(d, d'), uses first order approximation of difference in parameters $(\theta_{t+1} - \theta_t)$ for end-of-epoch checkpoints θ_t uniformly sampled from a training run $CheckPoints \subset \{1, 2, ..., \#Epochs\}$ (see [4] for details). The estimate is given by:

$$TracIn(d, d') = \sum_{t} \eta_t \nabla l(\theta_t, d).\nabla l(\theta_t, d') \tag{1}$$

where η_t is the step length of the SGD algorithm.

We observe that the residual error of value function estimates given by this method is very poor. Two main reasons for the poor approximation are: (1) TracIn considers only the first-order approximation, while many training algorithms use second-order gradient information, and (2) TracIn does not consider the intermediate checkpoints while approximating $(\theta_{t+1} - \theta_t)$. Many mini-batch SGD training algorithms trained on large datasets perform important updates in the middle of epochs. Hence, considering the intermediate updates (or checkpoints) can provide better approximation to the value function than using end-of-epoch checkpoints only. We show it empirically in the experiment section in Figure 8. We propose a sparse approximation-based formulation, **CheckSelect**, for selection of checkpoints in the next section.

B. Online Sparse Approximation for Checkpoint Selection

In order to consider the effect of intermediate checkpoints, we redefine the checkpoints in a training trajectory as θ_q^t, where $t = 1, ..., T$ are the epoch indices, and $q = 1, ..., O$ are the minibatch indices. Hence, we can define the per epoch decrease in loss function as:

$$\mathcal{V}_t(d') = \sum_{i=1}^{O} l(\theta_i^t, d') - l(\theta_{i-1}^t, d') \tag{2}$$

Following Taylor series expansion, and approximating the quadratic form of Hessian matrix with the square of the first order approximation, the second order expansion becomes: $\mathcal{V}_t(d') \approx -\eta \sum_{i=1}^{O} \nabla l(\theta_{i-1}^t, \mathcal{D}_i)^T \nabla l(\theta_{i-1}^t, d') + \frac{1}{2} * (\nabla l(\theta_{i-1}^t, \mathcal{D}_i)^T \nabla l(\theta_{i-1}^t, d'))^2$. This approximation is a sum of terms, which can be thought as features calculated from the $(i, t)^{th}$ checkpoint. This leads to our definition of the features for the sparse approximation:

$$\begin{aligned} X_i^t(\mathcal{D}_i, d') = \nabla l(\theta_{i-1}^t, \mathcal{D}_i)^T \nabla l(\theta_{i-1}^t, d') + \\ \frac{1}{2} * (\nabla l(\theta_{i-1}^t, \mathcal{D}_i)^T \nabla l(\theta_{i-1}^t, d'))^2 \end{aligned} \tag{3}$$

Note that here, $l(\theta_{i-1}^t, \mathcal{D}_i)$ is the total loss over all examples $d \in \mathcal{D}$ in minibatch i.

This definition naturally leads to our sparse approximation formulation for estimating $\mathcal{V}_t(d')$ as $\mathcal{V}_t(d') \approx \sum_{i \in s_t} (\beta_i^t X_i^t(\mathcal{D}_i, d'))$, where s_t is the set of checkpoint indices selected in epoch t, and β_i^t are the learned coefficients for each of the selected checkpoints θ_i^t. We also denote $S_t = \cup_{m=1}^{t} s_m$ as the cumulative set of all checkpoint indices selected till epoch t. Denoting $\vec{\mathcal{V}}_t$ as the vector of all $\vec{\mathcal{V}}_t = (\mathcal{V}_t(d'),\ \forall d' \in \mathcal{D}')$, and $\vec{X}_i^t$ as the vector of all $\vec{X}_i^t = (X_i^t(\mathcal{D}_i, d'),\ \forall d' \in \mathcal{D}')$ analogously, we arrive at our initial formulation for sparse estimation as:

$$\min_{s_1, ..., s_T, \beta} ||(\sum_{t=1}^{T} \sum_{i \in s_t} (\beta_i^t \vec{X}_i^t)) - (\sum_{t=1}^{T} \vec{\mathcal{V}}_t|)||_2^2 \quad s.t. |S_T| \leq k \tag{4}$$

where β denotes the set of all β_i^t and k is the sparsity degree. Next, we propose an Orthogonal Matching Pursuit (OMP) based algorithm to solve this problem. OMP [11] is a popular algorithm for solving the above-mentioned sparse approximation formulation of checkpoint selection. Among its key advantages is the fact that it is a greedy method, and can be used in online mode when the datapoints $(\vec{\mathcal{V}}_t, \vec{X}_i^t)$ are revealed one-at-a-time. However, the current problem has an added complication: the features $\vec{X}_i^t$ are made available in an online manner, as the training progresses, but the final value function $(\sum_{t=1}^{T} \vec{\mathcal{V}}_t|)$ is available only at the end of epoch T. Since, it is impractical to store all the checkpoints $\theta_i^t, i = 1, ..., O;\ t = 1, ..., T$, vanilla OMP algorithm cannot be applied. We call this the *online sparse approximation* (OSA) problem, which has not been well studied in the literature.

In order to motivate the proposed algorithm for OSA, we define the vector $\vec{y_t} = (y_t(d');\ \forall d' \in \mathcal{D}')$ denoting the cumulative decrease in loss across epochs as:

$$y_t(d') = \sum_{m=1}^{t} \vec{\mathcal{V}}_m \ \forall t = 1, ..., T \tag{5}$$

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2024.3506494

TABLE I: **Description of notations used**

Topics	Notation	Explanation
Datasets	$\mathcal{D},\ d_i = (u_i, w_i),\ N = \lvert\mathcal{D}\rvert$	Training dataset of size N and training datapoint i, N
	$\mathcal{D}_q \subseteq \mathcal{D}$, $q = 1, ..., O$	q^{th} Minibatch of Training Dataset
	$\mathcal{D}',\ M = \lvert\mathcal{D}'\rvert$	Validation dataset
Training model parameters	T , m	Total Epochs, running index for epochs
	$\theta_t, t \in T$	Model parameter/Checkpoint at epoch t
Loss function	$l_t(\theta_t, d')$	Loss incurred on d' from θ_t
Value function	$\mathcal{V}_t(d'),\ \mathcal{V}_t(\mathcal{D}')$	Total Value function(for all training datapoints) in epoch t for validation datapoint d' / Dataset $\mathcal{D}'$
	$\vec{y}_t$	Cumulative value function vector $\sum_{m\in\{1,...,t\}} \mathcal{V}_m(d')\forall d' \in \mathcal{D}'$
	$X_q^t(\mathcal{D}_i, d')$	Features from checkpoint θ_q^t (batch q, epoch t)
	S_t	Set of all selected checkpoints till epoch t
	β_j	Coefficient of selected checkpoint S_j

In the OSA setting, at the end of epoch t we want to approximate:

$$\vec{y}_t \approx \sum_{(i,m)\in S_t} \beta_i^m \vec{X_i^m}$$

where, S_t is the set of selected checkpoint indices till epoch t, that has values from $\{(i,m)|m = 1, .., t, i = 1, ..., O\}$, and $\vec{X_i^m}$ are the features for i^{th} selected checkpoint at epoch m.

Algorithm 1 describes the proposed *CheckSelect* algorithm for solving the online sparse approximation problem. The algorithm proceeds by solving a sequence of sparse approximation problems for each epoch t as described in step 11 of Algorithm 1, resulting in the set of selected checkpoints S_t and the corresponding weights $\vec{\beta}_t$. The sparse approximation problem minimizes the approximation error between the current estimate vector $\vec{\xi}_t = \sum_{q,m\in S_t} \beta_q^m \vec{X}_q^m$ and the cumulative value function vector $\vec{y}_t$. We note that at each epoch, we select a maximum of k checkpoints, i.e. $|S_t| \leq k, \forall t = 1, ..., T$. Lines 16 - 21 maintain the size of S_t by either replacing an existing checkpoint (using **CheckReplace**) if there are already k selected checkpoints, or adding the current checkpoint.

The **CheckReplace** subroutine described in Algorithm 1.1 eliminates the sub-optimal checkpoints by substituting them if the current checkpoint under consideration is found to result in a lower residual. Lines 4 - 12 in Algorithm 1.1 compare the features of an current checkpoint $\vec{X}_i^t$ with features of all existing selected checkpoints $\vec{X}_q^m \in S_t$ in a loop. For each existing checkpoint feature $\vec{X}_q^m$ we calculate the modified residual $\vec{\gamma}$ which discards the effect of $\vec{X}_q^m$ from the current residual $\vec{\rho}_t$ (line 5, Algo 1.1). Lines 6 and 7 compute the projections of the current checkpoint $\vec{X}_i^t$ and the under-consideration for replacement checkpoint $\vec{X}_q^m$ on the modified residual vector $\vec{\gamma}$, termed as π and π' respectively. The conditions in line 8 evaluate to true if the under-consideration checkpoint $\vec{X}_q^m$ is a potential candidate for replacement, i.e. its projection is smaller than the current checkpoint, but larger than the current candidate for replacement. To summarize, **CheckReplace** replaces the existing checkpoint with the highest marginal residual projection lower than the projection of the current checkpoint, if one exists. The pseudocodes of the algorithm and the replacement subroutine are provided in Algorithms 1 and 1.1, respectively.

Algorithm 1 CheckSelect: Online checkpoint selection

1: **Input**: k: Total number of checkpoints to be selected
2: $\vec{y}_t$: Cumulative value function after epoch t
3: S_t: Set of selected checkpoint indices till epoch t
4: $\vec{\beta}_t \in \mathbb{R}^{|S_t|}$: Weight of selected checkpoints till epoch t
5: $\vec{X}_i^t$: Features calculated at checkpoint (i, t) (Eqn: 3)
6: **Initialize:**
7: $S_0 = \phi$
8: **for** t = 1,...,T **do**
9: **Input:** $\vec{y}_t, \vec{X}_i^t, \forall i \in \{1, ..., O\}, ||\vec{X}_i^t||_2 = 1$
10: **Process**:
11: Update $\vec{\beta}_t = \text{argmin}_\beta \|\vec{y}_t - \sum_{q,m\in|S_t|}(\beta_q^m \vec{X}_q^m)||_2$
12: Update $\vec{\xi}_t = \sum_{q,m\in S_t} \beta_q^m \vec{X}_q^m$
13: **for** each batch $i \in \{1, 2, \ldots, O\}$: **do**
14: /* Add or Replace selected checkpoints */
15: **if** $|S_t| = k$ **then**
16: $S_t \longleftarrow$ **CheckReplace**$(\vec{y}_t, \vec{\xi}_t, S_t, \vec{\beta}_t, \vec{X}_i^t)$
17: **else**
18: $S_t \longleftarrow S_t \cup \{(i, t)\}$
19: **end if**
20: Update $\vec{\beta}_t = \text{argmin}_\beta \|\vec{y}_t - \sum_{q,m\in|S_t|}(\beta_q^m \vec{X}_q^m)||_2$
21: Update $\vec{\xi}_t = \sum_{q,m\in S_t} \beta_q^m \vec{X}_q^m$
22: **end for**
23: **end for**
24: **Output**:Final set of selected checkpoint indices S_T, learned coefficients $\{\beta_q^m|(q, m) \in S_T\}$, and corresponding checkpoints $\{\theta_q^m|(q, m) \in S_T\}$

Algorithm 1.1 CheckReplace $(\vec{y}_t, \vec{\xi}_t, S_t, \vec{\beta}_t, \vec{X}_i^t)$

1: $\vec{\rho}_t = \vec{y}_t - \vec{\xi}_t$ //residual vector
2: $\pi_{max} = -\infty$ // highest projected residual from S
3: $(j, n) = \phi$ //checkpoint index to be removed from S
4: **for** each index q, m in S_t: **do**
5: $\vec{\gamma} = \vec{\rho}_t + \beta_q^m * \vec{X}_q^m$ //modified residual
6: $\pi = \text{abs}(\vec{X}_i^t.\vec{\gamma})$ //projection of incoming feature X_i^t
7: $\pi' = \text{abs}(\vec{X}_q^m.\vec{\gamma})$ //projection of existing feature X_j
8: **if** $\pi > \pi'$ and $\pi' > \pi_{max}$ **then**
9: $\pi_{max} \longleftarrow \pi'$
10: $(j, n) \longleftarrow (q, m)$
11: **end if**
12: **end for**
13: **if** $(j, n) \neq \phi$ **then**
14: $S_t \leftarrow S_t \setminus \{(j, n)\} \cup \{(i, t)\}$ //Replace $(j, n) \in S_t$ with (i, t)
15: **end if**
16: return S_t

C. Data valuation using selected checkpoints

In this section, we describe our algorithm for estimating the value for each training datapoint using the selected checkpoints obtained from the *CheckSelect* algorithm. There are 2 types of training datapoints $d_i \in \mathcal{D}$: (1) Those belonging to some minibatch $\mathcal{D}_q$, $d_i \in \mathcal{D}_q$, such that $(q, m) \in S_T$ for at least one epoch index m, and (2) those which belong to minibatches $\mathcal{D}_{q'}$, for which no updates are selected in S_T. Note that, sparse approximation of the cumulative value function given by CheckSelect only selects a subset of important

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2024.3506494

checkpoints, and not an exhaustive list of subsets. Hence, a training datapoint belonging to the second set of minibatches $\mathcal{D}'_q$ is not necessarily of zero value. Hence we use two different methods to estimate the value for these two types of training datapoints.

Given a training datapoint $d_i \in \mathcal{D}_q$ from the first set descibed above, and the set of selected checkpoint indices S_T, learned coefficients $\{\beta_q^m | (q,m) \in S_T\}$, and corresponding checkpoints $\{\theta_q^m | (q,m) \in S_T\}$ computed using algorithm 1.1; we compute the features corresponding to the individual training datapoint as:

$$X_q^m(d_i, \mathcal{D}') = \nabla l(\theta_q^m, d_i)^T \nabla l(\theta_q^m, \mathcal{D}') + \frac{1}{2} * (\nabla l(\theta_q^m, d_i)^T \nabla l(\theta_q^m, \mathcal{D}'))^2 \quad (6)$$

We note that there may be multiple epoch indices m for which $d_i \in \mathcal{D}_q$ and $\theta_q^m \in S_T$ have been selected. While performing data valuation for the training datapoint d_i, we consider the contribution from all of them for computing the value of datapoint d_i. Hence given the set $\sigma(d_i) = \{m | d_i \in \mathcal{D}_q \text{ and } \theta_q^m \in S_T\}$, we estimate the value of a training datapoint d_i as:

$$Value(d_i \in \mathcal{D}_q) = \sum_{m \in \sigma(d_i)} (\beta_q^m X_q^m(d_i, \mathcal{D}')) \quad (7)$$

For datapoints $e \in \mathcal{D} \setminus \mathcal{B}$, where $\mathcal{B} = \{\mathcal{D}_q | (q,m) \in S_T \text{ for some } m = 1, ..., T\}$ consists of the selected mini-batches; there is no direct information to estimate their value using the checkpoints. Hence for such datapoints, we find the nearest neighboring point $z \in \mathcal{B}$ (using feature similarity) from the same class as datapoint $e \in \mathcal{D} \setminus \mathcal{B}$. We compute the data value for e using Equation 7 and the corresponding learned coefficient value for z. The intuition behind using the nearest neighbor technique is that if the distance between a datapoint e (whose value is unknown) and a datapoint z (among the set of known scored points) is less than ϵ which is sufficiently small, the TracIn value function defined in 1, for a given fixed trajectory, are also close. The following proposition states the formal result.

Proposition 1: Let $e = (u_1, w_1) \in \mathcal{D} \setminus \mathcal{B}$ be a training datapoint not included in the mini-batches corresponding to selected checkpoints, and $z = (u_2, w_1)$ is its nearest neighbor, such that $\|u_1 - u_2\| \leq \epsilon$ and $w_1 = w_2$. Also, let $\theta_1, ..., \theta_T$ be a fixed training trajectory and the loss function, $l(e, \theta)$, of the neural network is L-smooth, i.e.

$$\|\nabla_\theta l(\theta, e) - \nabla_\theta l(\theta, z)\| \leq L \|u_1 - u_2\|$$

Then:

$$|TracIn(e, \mathcal{D}') - TracIn(z, \mathcal{D}')| \leq L \times |\mathcal{D}'| \times T \times \epsilon \quad (8)$$

As we know, $TracIn(d) = \sum_{t=1}^{T} \sum_{d' \in \mathcal{D}'} [\eta_t \nabla l(\theta_t, d) . \nabla l(\theta_t, d')]$ is the estimate of total value function for the training datapoint $d \in \mathcal{D}$, hence the above result follows from the fact that $\theta_1, ..., \theta_T$ are fixed and $\|\nabla_\theta l(\theta, e) - \nabla_\theta l(\theta, z)\| \leq L \|u_1 - u_2\|$.

Simsel- similarity-based subset selection: While the valuation scores derived above (Eqn. 7) or those given by TracIn (Eqn. 1) evaluate the contribution of each training datapoint towards the decrease in loss, they may not lead to selection of the most diverse set of training datapoints catering to needs of all validation datapoints. Hence, selecting data subsets by simply sorting the datapoints according to their value, may not lead to a high accuracy for a given limit on the size of selected training datasets (see e.g. [9]). We also propose **Simsel**, a similarity-based data subset selection technique that attempts to select a diverse subset of data, thus achieving a high test set accuracy for a given subset size K. The key idea behind Simsel is to define a *similarity score* between the training datapoints, $d_i \in \mathcal{D}$ using a vector of value function estimates for each validation datapoint $d' \in \mathcal{D}'$. Hence using the vectorized form of Eqn. 7, we define the *contribution* vector, $\vec{\mathcal{C}}(d_i) \in \mathbb{R}^M$ as:

$$\vec{\mathcal{C}}(d_i) = [\sum_{m \in \sigma(d_i)} (\beta_i^m \vec{X}_i^m(d_i, d')) \ ; \ \forall d' \in \mathcal{D}'] \quad (9)$$

The similarity between two training datapoints $S(d_i, d_{i'})$ is defined as the cosine similarity between the contribution vectors, and the corresponding distance function is defined as:

$$S(i, i') = \cos(\vec{\mathcal{C}}(d_i)^T \vec{\mathcal{C}}(d_{i'})) \ ; \ \delta_{ii'} = 1 - S(i, i') \quad (10)$$

Algorithm 1.2 Simsel: Diverse Subset Selection

Input:
  Training dataset $\mathcal{D}$ sorted based on obtained scores from Eqn. 7, Subset size K
Output:
  Subset of training data $\mathcal{D}_S^O \subseteq \mathcal{D}$
Algorithm:
  $\mathcal{D}_S^0 =$ Top K elements from $\mathcal{D}$ based on scores.

for each $\mathcal{D}_l, l = 1, ..., O$ **do**
  $\mathcal{D}^C = \mathcal{D}_S^{l-1} \cup \mathcal{D}_l$
  $\mathcal{D}_S^l =$ Top K representatives $\in \mathcal{D}^C$ using Equation 11.
end for

Algorithm 1.2 describes a scalable online subset selection technique for the selection of subset of training datapoints D_S^O. In every iteration l, we construct a combined dataset $\mathcal{D}^C = \mathcal{D}_S^{l-1} \cup \mathcal{D}_l$. We then use convex optimization-based subset section to select the top k datapoints from $\mathcal{D}^C$ to construct $\mathcal{D}_S^l$. The objective function for the optimization problem for selecting the top-k subset is defined in Eqn 11. It uses a facility location-like objective aimed at minimizing the dissimilarity between pairs of instances (δ_{ij}), and returns k diverse representatives.

$$p_{ij}^* = \arg\min_{p_{ij}} \sum_{i,j=1}^{|\mathcal{D}^C|} p_{ij} \delta_{ij} \ \textbf{sub.to}: \ p_{i,j} \in [0,1] \ \forall i, j \ ; \ \sum_{j=1}^{|\mathcal{D}^C|} p_{i,j} = 1 \ \ \forall i \ , \ \sum_{j=1}^{|\mathcal{D}^C|} \|[p_{1,j} \dots p_{|\mathcal{D}^C|,j}]\|_2 \leq K \quad (11)$$

Here, p_{ij}s are relaxed binary variables with $p_{ij} = 1$ denoting that datapoint d_j is a representative for datapoint d_i. The objective function minimizes the total distance of every point from their representatives. The constraints ensure that every datapoint get exactly one representative, and the total number

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2024.3506494

of representatives is less than K. The output of the program are the K representatives, $\mathcal{D}_S = \{j | p^*_{ij} = 1\}$ which are returned as the selected points. The online subset selection objective is similar to techniques presented in [9], [23].

Computational Complexity: The checkpoint selection algorithm can be run as a part of a training algorithm. In addition to the gradient-based updates, **CheckSelect** takes $\mathcal{O}(kM)$ time for each update. Hence the total time complexity is $\mathcal{O}(kMOT)$. **Simsel** solves the optimization problem in Equation 11 O-times, each with complexity $\mathcal{O}(K^2|\mathcal{D}_l|^2|)$. Hence, the overall complexity of selection is $\mathcal{O}(OK^2|\mathcal{D}_l|^2)$.

IV. Experimental Results

TABLE II: **Efficiency & Accuracy Comparison for DVSS methods on CIFAR10 dataset.**

Methods	Time (mins)				Accuracy		
	Pretrain phase	Data selection / Valuation time			Subset fraction		
		5%	20%	35 %	5%	20%	35%
Random	-	-	-	-	65.5	83.9	87.0
kCenterGreedy	-	2.45	2.53	2.88	65.1	87.6	91.8
GraNd	133	0.95	0.96	0.97	57.9	85.1	91.5
GradMatch	133	1.83	4.13	8.52	51.4	71.1	88.1
CheckSelect	151	5.38	5.38	5.38	**69.7**	**89.4**	**93.6**
CheckSelect Simsel	151	8.4	9.98	11.6	**68.2**	**89.9**	**93.8**
TracIn	133	60	60	60	32.23	62.76	80.6
TracIn Simsel	133	63	64.6	66.2	57.5	78.6	91.5
Influence Functions	133	120	120	120	35.2	83.5	87.6
DVRL	179	125	125	125	38.9	82.1	86.5
Data Shapley	180	129.8	129.8	129.8	39.2	61.13	69.1

We empirically evaluate the effectiveness of the proposed method **CheckSelect** [1] (Algorithms 1 and 1.1) and **Simsel** (Algorithm 1.2) on the aspects of *efficiency, accuracy, robustness and flexibility*. We demonstrate our results primarily using computer vision tasks of large-scale image classification using three standard datasets CIFAR10 (10 classes, 50000 examples), CIFAR100 (100 classes, 50000 examples) [24], and TinyImagenet (200 classes, 100000 examples) [25]. Section IV-A evaluates the **Accuracy** and **Efficiency** of our method using the text classification task of 20 newsgroups dataset [26], regression task in California Housing Price [27] dataset and the time series based multivariate gait data [28] that classifies 3 different type of joints. Section IV-B demonstrates the **Flexibility and Robustness** of the proposed algorithms in the class imbalance and domain adaptation setting using the standard *Office-Home* [15] dataset. In section IV-C we analyze the working of the proposed algorithm in an adversarial setting where datapoints are mislabelled along with showing various properties of selected data subsets to demonstrate their usefulness.

Implementation Details: We use the following architectures for the respective tasks: (a) Image classification : ResNet18 [29] model; (b) Text classification : 3 layer neural network using TF-IDF vectors of the documents; (c) Regression : 3 layer neural network using the dataset features; (d) Time series: 2 layer recurrent neural network. We use SGD as the optimizer with a learning rate of 0.1 and batch size as 100, for all the tasks.

[1]The codes are available at https://github.com/SoumiDas/CheckSelect/.

A. Efficiency and Accuracy Comparison

We assess the suitability of the baselines and proposed methods using the *time taken* for data valuation and subset selection (DVSS) and the *accuracy* (Root Mean Square Error i.e. RMSE in case of regressions task) of the model obtained from training using the selected subset. For the comparison, we experiment with an extensive set of the baseline methods taken from the deepcore library [16], e.g. **Random**, **k-center Greedy** [5], **GraNd** [1], **GradMatch**, and [2]. Additionally, we also compare with the standard data valuation methods e.g. **TracIn** [4], **Influence Functions** [6], **DVRL** [3], and **Data Shapley** [7]. Following [16], we use the data selected in the final epoch for GradMatch [2]. For the proposed techniques, we report results with: (1) **CheckSelect** - uses CheckSelect for checkpoint selection, followed by subsets selected using the total estimated data valuation scores, (2) **CheckSelect-Simsel**- uses CheckSelect for checkpoint selection followed by Simselfor data selection using the per-validation datapoint scores. We also report results for **TracIn Simsel**, which uses the Simsel algorithm for subset selection using the scores obtained from TracIn. We use a set of 10 checkpoints for all our experiments. We have performed our experiments on a 64-bit Intel Xeon Machine with a single Quadro-P5000 GPU, and reported running times for the same.

We draw a comparison of the time consumed by the different DVSS methods along with their performance to return a set of selected subset from CIFAR10 in Table II. We divide the total running time into two parts : (1) **Pretrain phase**: this is the time needed to obtain the inputs for executing the data selection/valuation algorithm (e.g. training using SGD on the dataset); and (2) **Data Selection/Valuation**: this is the time needed to obtain the selected subset of a given size expressed as a fraction of the total dataset size. All the methods except Random and kcenterGreedy require pre-training on the entire dataset to obtain a trained model or a set of trajectory points (like TracIn and CheckSelect) in the Pretraining phase. We note that the proposed method is competitive in terms of total time taken with all methods except random selection and k-center greedy. Moreover, in the re-train mode where the pre-training phase is not needed, the proposed method takes time competitive with all other methods. Finally, we observe that **CheckSelect** and **CheckSelect Simsel** are much faster than other data valuation methods – TracIn [4] , Influence functions [6], Data Shapley [7] and DVRL [3]. It is also interesting to observe that the data valuation time for **CheckSelect** does not change with subset size and is much smaller than TracIn, which needs the entire training dataset for calculating the data values, while **CheckSelect** only needs the data points in the mini-batches corresponding to the selected checkpoints. In Table II, we can observe that the proposed method surpasses all the baselines in terms of accuracy. One interesting

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2024.3506494

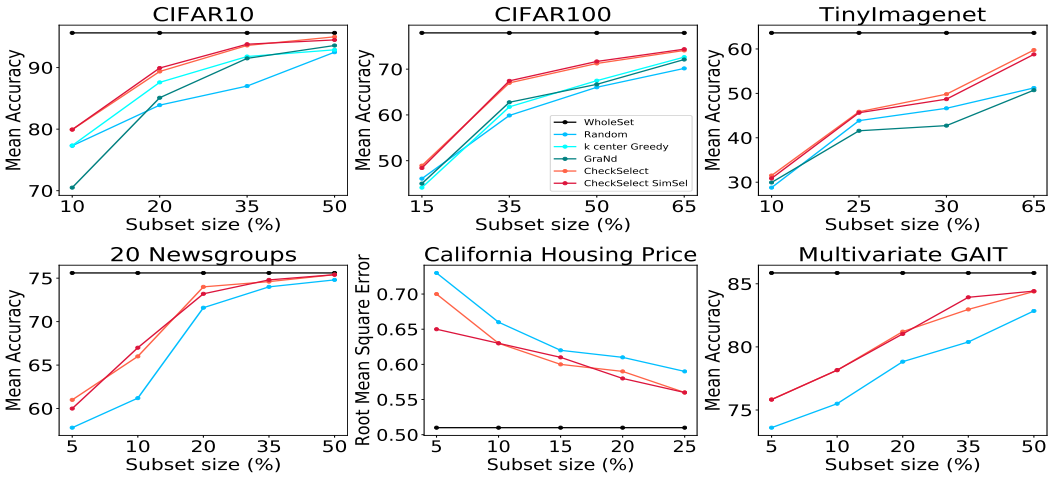

Fig. 2: **Performance of selected subsets for CIFAR10, CIFAR100, TinyImageNet *(top-row)* , 20 Newsgroups, California Housing Price and GAIT *(bottom-row)* for different fractions.**

TABLE III: **Accuracy and Robustness: Variance in accuracy for varying fractions for CIFAR10 and CIFAR100.**

Method	**CIFAR10**				**CIFAR100**				
Subset size →	**5%**	**10%**	**20%**	**35%**	**10%**	**15%**	**35%**	**50%**	**65%**
Random	66.56±2.30	78.13± 1.93	86.73± 2.09	89.70±1.98	33.48 ± 0.60	46.90 ± 0.88	58.50 ± 1.13	64.60 ± 1.08	69.20 ±0.80
kCenterGreedy	64.50± 1.40	76.90± 2.30	86.70±1.24	91.50±0.42	34.13 ±1.51	43.53 ±0.67	61.60 ±0.46	67.90 ± 0.40	72.21 ± 0.50
GraNd	57.20±1.71	70.86±1.53	85.46±0.74	91.46±0.45	33.40 ± 1.30	44.48 ± 0.37	62.45 ±0.35	68.35 ± 1.18	72.14 ± 0.30
CheckSelect	68.70±0.78	80.11±0.43	88.99±0.49	93.00±0.49	35.98 ± 0.61	48.10 ± 0.73	67.06 ± 0.40	71.80 ± 0.39	74.27 ± 0.21
CheckSelect Simsel	**70.50±0.52**	**81.10±0.64**	**90.10±0.45**	**93.80±0.42**	**36.32 ± 0.52**	**50.20 ± 0.45**	**67.25 ± 0.45**	**72.30 ± 0.32**	**75.30 ± 0.41**

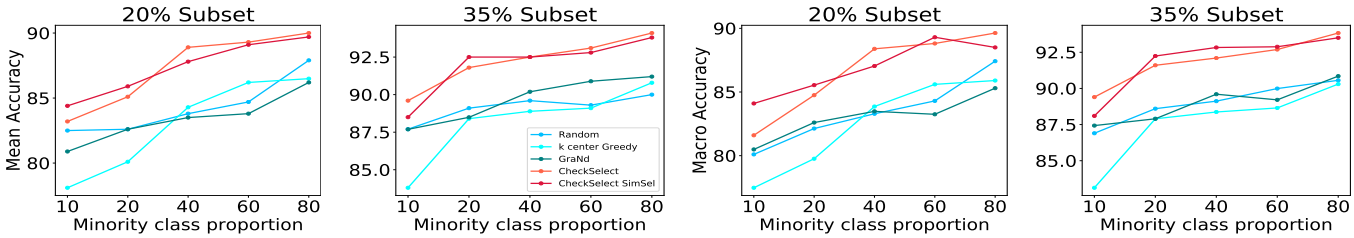

Fig. 3: **Robustness: Evaluation of subsets for CIFAR10 across minority class proportions over two metrics.**

observation is that TracIn Simsel performs significantly better than TracIn, thus indicating the usefulness of *Simsel* algorithm. However, the impact of Simsel is less pronounced in the case of CheckSelect Simsel because CheckSelect yields a high-value optimal subset while approximating the value function, which leaves little scope for improvement using Simsel.

From the above study, we find Random, k-Center Greedy, and GraND to be the strongest competitors from the **accuracy – efficiency** consideration, while other methods are either slow (DVRL, Data Shapley, TracIn) or inaccurate (GradMatch, TracIn, Influence Function, DVRL, Data Shapley). In Figure 2, we compare the performance of proposed methods **Check-Select** and **CheckSelect Simsel** on six different datasets – CIFAR10 , CIFAR100, TinyImagenet, 20Newsgroups, California Housing Price and multivariate Gait data. We observe that the proposed methods outperform the baselines on all the tasks by a significant margin, across the various fractions of selected data. Moreover, for all the tasks, the gap in performance from the nearest baseline remains consistent as the performance of the selected subset approaches that of the whole dataset (black line). Since for a data pruning application, the user usually tolerate a limited reduction in accuracy from the wholeset, we conclude that the proposed methods provide a better alternative to the existing SOTA methods for DVSS.

B. Flexibility and Robustness Comparison

In this section, we evaluate the proposed methods with the best baselines (Random, kCenterGreedy, GraNd) for their flexibility and robustness. Figure 3 shows the mean and macro-accuracies of the DVSS methods on an imbalanced classification task, with reduced data from 2 randomly chosen classes, and full data from the rest of the 8 classes in the CIFAR10 dataset. We observe that the gap between the proposed methods and the baselines is substantially larger for the imbalanced classification, compared to the balanced classification problem, thus denoting **robustness**. Table III also reports the variation in the performance of models trained on the selected sets calculated from 3 random runs of the training algorithm, which leads to different checkpoints. We can observe in Table III that the proposed method has a higher accuracy as well as a lower standard deviation. This basically denotes that the proposed algorithm is *robust as well as accurate*, across runs with different seeds.

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2024.3506494

Next, we look at the setting of selection of instances in *other but related domains* where our framework is also capable of using different value functions while finding out the subsets, thus demonstrating both ***flexibility and robustness***.

Recall that the checkpoint selection algorithm, CheckSelect, requires a valuation dataset which is used to calculate the *value function*. In many practical applications, one is interested in data valuation in a related *Target* domain dataset (D_t), even though the original checkpoints were selected using a value function from a *Source* domain dataset (D_s). We call this the *Domain Adaptation* setting for data valuation and data subset selection. The main advantage of this setting is that it avoids the relatively expensive training and checkpoint selection for the target domain.

For experimentation, we use the standard Office-Home dataset [15] containing 4 domains - Art(**A**), Clipart(**C**), Product(**P**), and Real-World(**R**), each with 65 categories. We report results for all 16 *source – target pairs* taken from the 4 domains. The experiments are conducted using a pretrained ResNet-50 model for initialisation, following standard procedure [30]. For every source – target pair, both source dataset D_s and target dataset D_t are divided into training and validation counterparts (D_{st} , D_{sv} and D_{tt} , D_{tv}) respectively. In Table IV, we report results for the following methods:

1) **Random**: Instances are randomly sampled from D_{tt}.
2) **k-center Greedy**[5]: Instances are selected from D_{tt} using k-center algorithm.
3) **GraNd**[1]: Instances are selected from D_{tt} based on their scores, using model trained on D_{st}.
4) **TracIn**[4]: Checkpoints are sampled while training on D_{st}, data-valuation is performed on D_{tt} using TracIn score with valuation dataset D_{tv}.
5) **CheckSelect**: Checkpoints are selected using CheckSelect, while training on D_{st} using a value function given by D_{sv}. Data valuation is performed on D_{tt} using value function on the valuation dataset D_{tv}. This represents the setting where target domain data is *not* available during checkpoint selection, and no-retraining is needed for data valuation on target domain.

We perform this experiment with selection of 20 checkpoints (for TracIn and CheckSelect). Table IV reports the validation accuracy on target domain (D_{tv}) using models trained on subsets of target domain training data along with the difference (Δ) with the best performing baseline. We can observe that **CheckSelect** performs substantially better than all the baselines on the target domain, especially on the Art domain that has the least number (2427) of instances, thus confirming the robustness and flexibility of CheckSelect in the simple domain adaptation setting.

C. Analysis of CheckSelect

In this section, we demonstrate the usefulness of the selected data through label flipping analogy and representation of selected subsets using data maps.

Identification of mislabels: Following [4], we randomly mislabel 20% of data with the most misclassified label. Next, we compute the self influence score for each training data point using the selected checkpoints from CheckSelect. Figure 4 shows the variation in number of detected mislabelled points with fraction of inspected data that is sorted on the basis of their self-influence scores.. We can observe that CheckSelect can detect a large number of mislabelled examples faster (at $\sim 20\%$ inspected data) than the baselines, thus demonstrating its robustness.

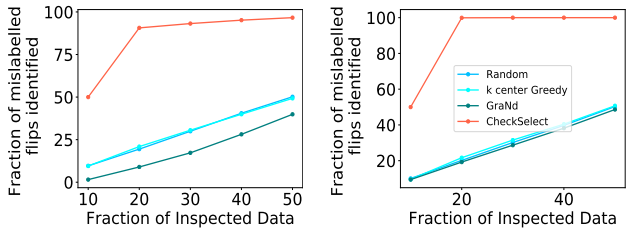

Fig. 4: **Fraction of detected flipped labels for CIFAR10(left) and CIFAR100(right) with fraction of inspected data.**

Data Maps: Inspired by [31], we construct data maps to visualise the selected dataset with respect to an underlying model. We measure the confidence of a datapoint x_i across T epochs as:

$$c_{x_i} = \frac{1}{T}\sum_{t=1}^{T} p_{\theta_t}(y_i|x_i) \tag{12}$$

where $p_{\theta_t}(y_i|x_i)$ is the probability of x_i to belong to the actual label y_i at epoch t. In case of loss, we measure the usual cross-entropy loss (l) for the point x_i.

Owing to the fact that both TracIn and CheckSelect use checkpoints for valuation, we try to analyse the obtained subsets from these two methods using data maps. Following [31], we construct data maps to visualise the selected dataset with respect to an underlying model. The intuition is that lower the confidence (c) or higher the loss (l) of a data point is, harder it is to be classified. Selecting such harder examples (*low confidence-high loss*) as a part of subset results in a better trained model.

We show the t-SNE plot from three dimensions - class, confidence and loss over high-scoring 1000 datapoints. We measure the confidence and loss in 3 levels - High, Medium and Low. We show in Figure 5 Row-1 (*left 2*), the distribution of the points across classes, and it can be seen that CheckSelect selects diverse set of instances from different classes, while TracIn selects the most from *bird* followed by *airplane*. We also observe in the histogram visualization in Row-1 (*right 2*), that CheckSelect has a majority of instances from low-confidence and high-loss values. We can thus relate this back to their better performance. We visualise the tSNE plots in Figure 5 (Row-2) that TracIn (*left 2*) selects instances mostly from medium confidence (MC) and medium loss (ML) while CheckSelect (*right 2*) has a majority of low confidence (LC) and high loss (HL) points, thus affirming the selection of valuable subsets. An interesting observation for CheckSelect based t-SNE plots are that instances from each of the classes are having a diverse level (*high, medium, low*) of confidence and loss values.

Variation with number of selected checkpoints: We varied the number of selected checkpoints for CheckSelect and

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2024.3506494

TABLE IV: **Flexibility and Robustness: Validation accuracy in target domain for various DVSS settings (20% selection).**

Methods	Source ->Target											
	A->C	**A->P**	**A->R**	**C->A**	**C->P**	**C->R**	**P->A**	**P->C**	**P->R**	**R->A**	**R->C**	**R->P**
Random	45.03	57.74	56.5	48.9	57.74	56.5	48.9	45.03	56.5	48.9	45.03	57.74
k-center [5]	46.3	67.41	51.4	49.48	61.5	59.45	52.47	40.54	57.56	48.81	40.54	60.0
GraNd[1]	45.86	62.2	58.74	51.5	64.4	64.3	53.62	49.52	61.2	50.53	45.5	54.42
TracIn[4]	31.4	33.95	34.63	36.34	39.76	34.87	34.62	30.85	36.28	33.76	29.9	33.25
CheckSelect	**47.99**	**70.11**	**67.49**	**51.61**	**69.18**	**65.48**	**54.4**	**57.32**	**69.03**	**50.75**	**48.64**	**64.41**
Δ	**1.69**	**2.7**	**8.75**	**0.11**	**4.78**	**1.18**	**0.78**	**7.8**	**7.83**	**0.22**	**3.14**	**4.41**

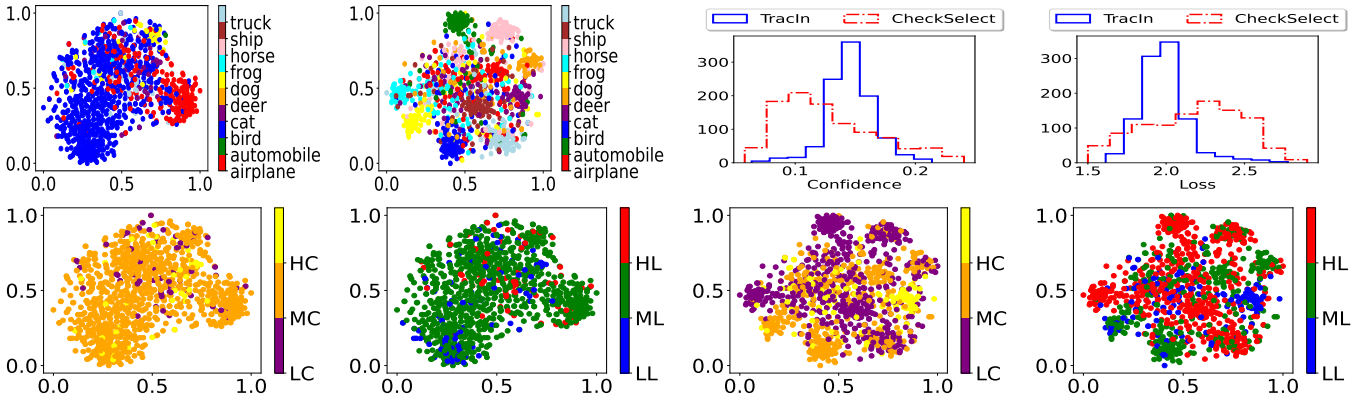

Fig. 5: **Data Maps for CIFAR10: (Row-1) t-SNE of selected top 1000 datapoints from TracIn and CheckSelect respectively followed by distribution of their confidence and loss values. ; (Row-2) t-SNE plot of selected top 1000 datapoints from TracIn (2 from left) and CheckSelect (2 from right) on the basis of Confidence and Loss.**

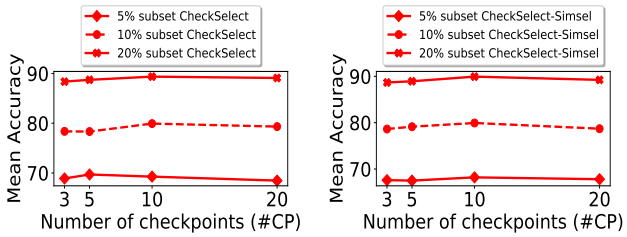

Fig. 6: **Performance of different fractions of subset with varying number of checkpoints in CheckSelect.**

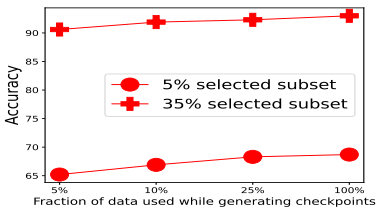

Fig. 7: **Performance of different fractions of subset with varying number of training datapoints while selecting checkpoints using CheckSelect.**

evaluated subsets of different fractions of CIFAR10. We can observe in Figure 6 that the performance is mostly robust across different number of checkpoints with 10 being a fairly reasonable parameter from the aspect of performance. This parameter vary depending on the nature of the tasks under consideration.

Checkpoint Selection using fraction of training data: We

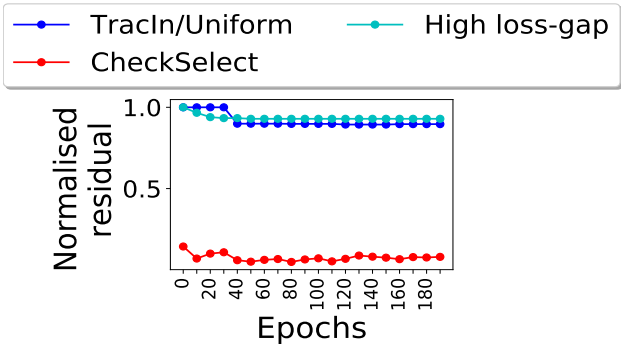

Fig. 8: **Normalised residual values for Uniform, High loss-gap and CheckSelect selected checkpoints.**

varied the number of randomly sampled training datapoints used for selecting checkpoints using CheckSelect, and evaluated subsets of different fractions for CIFAR10. We can observe in Figure 7 that the performance improves with number of datapoints, with 25% training datapoints being sufficient for selecting checkpoints that are further used for 5% and 35% subset selection. Thus, CheckSelect can select high-valued data and can achieve a generalizable performance for subset selection, by training with less number of datapoints. This can help in reducing the training time by a large scale.

D. Approximation Error

We show in Figure 8 the variation of normalised residual error (cumulative value function (y_t) - estimate (ξ_t)) with epochs for 3 different checkpoint selection methods - uniform

checkpoint selection (TracIn), checkpoints with higher loss gap, and CheckSelect based selection. The normalised residual value for CheckSelect is measured by

$$\frac{|\vec{y}_t - \sum_{(i,m)\in S^t}(\beta_i^m \vec{X}_i^m)|_2}{|\vec{y}_t|_2} \tag{13}$$

(notations follow from Methods section) while for uniform and high loss gap selection, it is measured by

$$\frac{|\vec{y}_t - \eta \sum_{t'=1}^{t} \frac{1}{N} \sum_{i\in N}(\nabla l(\theta_{t'}, d_i)\nabla l(\theta_{t'}, d'))|_2}{|\vec{y}_t|_2} \tag{14}$$

where t refers to the epoch under consideration. We observe that the residual values for CheckSelect are significantly lower than the rest, thus denoting how well its approximation is, which eventually gets manifested in the performance.

V. Conclusion

In this paper, we propose a two-phase framework for flexible, accurate, robust, and efficient (*FARE*) data valuation framework through checkpoint selection. We develop *CheckSelect*, an online OMP algorithm for selecting checkpoints, using which subsets of training data are found that can best approximate the value function. These subsets when trained from scratch outperform the SOTA baselines, sometimes by $\sim 30\%$. We also examine our algorithm on class imbalance and domain adaptation settings where training on only 20% subset of target domain data helps in outperforming the baselines across all domains. Additionally, we observe data maps through which we see that CheckSelect selects *harder* datapoints and eventually surpasses all other methods in the subsets of *FARE* domain. As a part of future work, we would like to study our method in the transfer learning and continual learning setup, where besides the difference in data distributions, the feature spaces are different as well.

References

[1] M. Paul, S. Ganguli, and G. K. Dziugaite, "Deep learning on a data diet: Finding important examples early in training," *Advances in Neural Information Processing Systems*, vol. 34, pp. 20 596–20 607, 2021.

[2] K. Killamsetty, D. Sivasubramanian, B. Mirzasoleiman, G. Ramakrishnan, A. De, and R. Iyer, "Grad-match: A gradient matching based data subset selection for efficient learning," *arXiv:2103.00123*, 2021.

[3] J. Yoon, S. Arik, and T. Pfister, "Data valuation using reinforcement learning," in *International Conference on Machine Learning*, 2020.

[4] G. Pruthi, F. Liu, S. Kale, and M. Sundararajan, "Estimating training data influence by tracing gradient descent," in *NeurIPS*, 2020.

[5] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," *arXiv preprint arXiv:1708.00489*, 2017.

[6] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *ICML*, 2017.

[7] A. Ghorbani and J. Zou, "Data shapley: Equitable valuation of data for machine learning," in *ICML*, 2019.

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

[9] S. Das, A. Singh, S. Chatterjee, S. Bhattacharya, and S. Bhattacharya, "Finding high-value training data subset through differentiable convex programming," in *European Conference, ECML PKDD 2021, 2021*. Springer, 2021, pp. 666–681.

[10] J. Tropp, "Greed is good: algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, 2004.

[11] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4655–4666, 2007.

[12] A. J. Weinstein and M. B. Wakin, "Online search orthogonal matching pursuit," in *2012 IEEE Statistical Signal Processing Workshop (SSP)*, 2012, pp. 584–587.

[13] E. M. Saad, G. Blanchard, and S. Arlot, "Online orthogonal matching pursuit," 2021.

[14] J. M. Schreiber, J. A. Bilmes, and W. S. Noble, "apricot: Submodular selection for data summarization in python." *J. Mach. Learn. Res.*, vol. 21, pp. 161–1, 2020.

[15] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation," in *CVPR*, 2017.

[16] C. Guo, B. Zhao, and Y. Bai, "Deepcore: A comprehensive library for coreset selection in deep learning," *arXiv preprint arXiv:2204.08499*, 2022.

[17] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, M. Proesmans, and L. Van Gool, "Scan: Learning to classify images without labels," in *ECCV*. Springer, 2020, pp. 268–285.

[18] A. A. Deshmukh, J. R. Regatti, E. Manavoglu, and U. Dogan, "Representation learning for clustering via building consensus," *arXiv preprint arXiv:2105.01289*, 2021.

[19] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros, "Dataset distillation," *arXiv preprint arXiv:1811.10959*, 2018.

[20] S. Lan, R. Panda, Q. Zhu, and A. K. Roy-Chowdhury, "Ffnet: Video fast-forwarding via reinforcement learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6771–6780.

[21] Z. Wu, C. Xiong, C.-Y. Ma, R. Socher, and L. S. Davis, "Adaframe: Adaptive frame selection for fast video recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1278–1287.

[22] E. Elhamifar and M. Clara De Paolis Kaluza, "Online summarization via submodular and convex optimization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1783–1791.

[23] S. Das, H. Patibandla, S. Bhattacharya, K. Bera, N. Ganguly, and S. Bhattacharya, "Tmcoss: Thresholded multi-criteria online subset selection for data-efficient autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6341–6350.

[24] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[25] Y. Le and X. S. Yang, "Tiny imagenet visual recognition challenge," 2015.

[26] K. Lang, "Newsweeder: Learning to filter netnews," in *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 331–339.

[27] R. K. Pace and R. Barry, "Sparse spatial autoregressions," *Statistics & Probability Letters*, vol. 33, no. 3, pp. 291–297, 1997.

[28] N. Helwig and E. Hsiao-Wecksler, "Multivariate Gait Data," UCI Machine Learning Repository, 2022, DOI: https://doi.org/10.24432/C5861T.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[30] R. Xu, G. Li, J. Yang, and L. Lin, "Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[31] S. Swayamdipta, R. Schwartz, N. Lourie, Y. Wang, H. Hajishirzi, N. A. Smith, and Y. Choi, "Dataset cartography: Mapping and diagnosing datasets with training dynamics," in *Proceedings of EMNLP*, 2020. [Online]. Available: https://arxiv.org/abs/2009.10795