Finding High-Value Training Data Subset through Differentiable Convex Programming

Soumi Das✉[1], Arshdeep Singh[1], Saptarshi Chatterjee[1], Suparna Bhattacharya[2], and Sourangshu Bhattacharya[1]

[1] Indian Institute of Technology, Kharagpur, West Bengal
✉soumi_das@iitkgp.ac.in
[2] Hewlett Packard Labs, Hewlett Packard Enterprise, Bangalore

Abstract. Finding valuable training data points for deep neural networks has been a core research challenge with many applications. In recent years, various techniques for calculating the "value" of individual training datapoints have been proposed for explaining trained models. However, the value of a training datapoint also depends on other selected training datapoints - a notion which is not explicitly captured by existing methods. In this paper, we study the problem of selecting high-value subsets of training data. The key idea is to design a learnable framework for online subset selection, which can be learned using mini-batches of training data, thus making our method scalable. This results in a parameterised convex subset selection problem that is amenable to a differentiable convex programming paradigm, thus allowing us to learn the parameters of the selection model in an end-to-end training. Using this framework, we design an online alternating minimization based algorithm for jointly learning the parameters of the selection model and ML model. Extensive evaluation on a synthetic dataset, and three standard datasets, show that our algorithm finds consistently higher value subsets of training data, compared to the recent state of the art methods, sometimes $\sim 20\%$ higher value than existing methods. The subsets are also useful in finding mislabelled training data. Our algorithm takes running time comparable to the existing valuation functions.

Keywords: Data valuation · Subset selection · Convex optimisation · Explainability

1 Introduction

Estimation of "value" of a training datapoint from a Machine Learning model point of view, broadly called *data valuation* [10, 19, 16] , has become an important problem with many applications. One of the early applications included explaining and debugging training of deep neural models, where the influence of the training data points on the test loss is estimated [13, 16]. Another application involves estimating the expected marginal value of a training datapoint w.r.t. general value functions e.g. accuracy on a test set [10], which can be used in buying and selling of data in markets. [19] lists a number of other applications, including detecting domain shift between training and test data, suggesting adaptive data collection strategy, etc.

Most existing formulations for data valuation assume a supervised machine learning model, $y = f(x, \theta)$ with parameters θ. For all the above mentioned applications, the key technical question is to determine the improvement in the value function (usually test set loss) given that a new set of datapoints are added to the training set. The influence function based methods [13, 16] aim to estimate the influence of each individual datapoint, on the test loss of a trained model. While these techniques are computationally efficient, they ignore an important fact: *the value of a training datapoint depends on the other datapoints in the training dataset.* For a given training data point x, another datapoint y may add to the value of x, possibly because it is very different from x. Alternately, y may reduce the value of x if it is very similar to x.

Shapley value based methods [10] also estimate the value of individual datapoints using the expected marginal gain in the value function while adding the particular datapoint. Hence, this method considers the effect of other datapoints, but only in an aggregate sense. Also, this method is computationally expensive. DVRL [19] is the closest to our approach. It learns a parameterised selection function which maximizes the reward of selecting training datapoints using Reinforcement Learning. The reward function is the marginal improvement in value function after selection of a training datapoint over the previous running average of value function. Unfortunately, this reward leads to selection of many training datapoints which are similar to each other, if their inclusion contributes to making the loss better than past average. To the best of our knowledge, none of the existing data valuation techniques explicitly consider similarities between training points, when selecting high value subsets.

In this paper, we consider a set value function for assigning value to a subset of training datapoints. A proxy for the value function is learned though an embedding of datapoints, such that distance between the embedded datapoints represent the inverse value of replacing one of the datapoints with another. In this framework, we propose the problem of *high-value data subset selection*, which computes the optimal subset of a given size maximizing the value of the subset, or minimizing the total distance between selected points and other training datapoints in the embedding space.

In order to efficiently compute the high value data subsets, we address two challenges. Firstly, the embedding function of datapoints needs to be learned by optimizing the value of the selected subset, which itself is the output of an optimization problem. Hence, our formulation should be able to propagate the gradient of value-function back through the optimization problem. Secondly, for practicality, our formulation should be able to select the datapoints in an online manner from minibatches of data. We address both these challenges by designing a novel learned online subset selection formulation for selecting subsets of training datapoints from minibatches. Our formulation is inspired by the online subset selection technique based on facility location objective [6], but uses a learned distance function using a parameterised embedding of the training datapoints. In order to learn the embedding parameters, our formulation is also compatible with the *differentiable convex programming* paradigm [1], which allows us to propagate the gradients of the value function (loss on test set) back to the embedding layer. We propose an online alternating minimization based algorithm to jointly learn the embedding function for selection of optimal training data subset, and learn the optimal model parameters for optimizing the loss on the selected subset. Our algorithm scales linearly in training data size for a fixed number of epochs.

We benchmark our algorithm against state of the art techniques. e.g. DVRL [19], Data Shapley [10], etc. using extensive experiments on multiple standard real world datasets, as well as a synthetic dataset. We show that selection of high value subset of training data points using the proposed method consistently leads to better test set accuracy, for similar sized selected training dataset, with upto 20% difference in test accuracy. Also, removal of high value subset of training datapoints leads to a much more significant drop in test accuracy, compared to baselines. We also show that the proposed method can detect higher fraction of incorrectly labelled data, for all sizes of selected subsets. Moreover, correcting the detected mislabelled points leads to higher increase in test set accuracy compared to baseline. We observe that DVRL [19], which is the closest competitor, selects datapoints in a biased manner, possibly because there, it selects many similar high-performing datapoints at each stage.

To summarise, our key contributions are:

1. We highlight the problem of high-value data subset selection, which selects a high-value subset, rather than assigning valuation to individual datapoints.
2. We formulate a novel learned online data subset selection technique which can be trained jointly with the ML model.
3. We propose an online alternating minimization based algorithm which trains the joint model using mini-batches of training data, and scales linearly with training data.
4. Results with extensive experimentation show that our method consistently outperforms recent baselines, sometimes with an accuracy improvement of $\sim 20\%$ for the same size subsets.

2 Related Work

The exponential rise in quantity of data and depth of deep learning models have given rise to the question of scalability. Several subset selection methods have been designed using submodular[2] and convex approaches[7], to combat the problem. However, all these techniques only rely on some measure of similarity (pairwise, pointwise) [6][5] and do not really account for the explanation behind its choice in view of the end-task. Hence the other question which arises is explainability. A set of methods for explainability is prototype based approaches which finds the important examples aimed towards the optimisation of a given value function, or in other words, improvement of a given performance metric (e.g test set accuracy). Pioneering studies on influence functions [4] have been used to estimate the change in parameter θ , given an instance x_i is absent and is used to obtain an estimate of $\theta_{-i} - \theta$. This aids in approximating the importance of the instance x_i. Following the works of [13] on influence functions, there have been several other works which try to measure the influence of training points on overall loss by tracking the training procedure [12] while some try to measure the influence of training points on a single test point [16]. A recent work by [18] performs rapid retraining by reusing information cached during training phase.

Shapley value, an idea originated from game theory has been a driving factor in the field of economics. It was used as a feature ranking score for explaining black-box models [14][15]. However, [10] were the first to use shapley value to quantify data points where they introduce several properties of data valuation and use Monte Carlo based methods for approximation. There has also been a recent study in using shapley for quantifying the importance of neurons of a black box model [11]. A follow-up work by [9] aims to speed up the computation of data shapley values. However, all the above methods do not have a distinctive selection mechanism for providing an explainable representative set of points.

Our work is in close proximity to the prototype based approaches. While [9] aims to work with training set distribution, our work is essentially inclined towards selection from training datasets, hence leading to choose [10] as one of our baselines. Among the methods around influence functions, we use the work of [13] , [16] as two of our baselines. A recent work by [19] attempts to adaptively learn data values using reinforcement signals, along with the predictor model. This work is the closest in terms of the problem objective we are intending to solve and hence we use it as one of our baselines. However, they intend to optimize the selection using reinforcement trick which requires a greater number of runs, while we use a learnable convex framework for the purpose of selection. To the best of our knowledge, this is the first of its kind in using a differentiable convex programming based learning framework to optimize the selection mechanism in order to improve on a given performance metric.

3 High-Value Data Subset Selection

In this section, we formally define the problem of high-value data subset selection and propose a learned approximate algorithm for the same.

3.1 Motivation and Problem Formulation

Data valuation for deep learning is an important tool for generating explanations of deep learning models in terms of training data. Let $\mathcal{D} = \{(x_i, y_i) | i = 1, \ldots, n\}$ be the training dataset with features $x_i \in \mathcal{X}$, and labels $y_i \in \mathcal{Y}$, and $\mathcal{D}^t = \{(x_i^t, y_i^t) | i = 1, \ldots, m\}$ be a test dataset, which is used for valuation of the training data in $\mathcal{D}$. Also, let $s \in \mathcal{S}$ denote a subset of of the training data $\mathcal{D}$ ($\mathcal{S}$ is the power set of $\mathcal{D}$). Let $f(\theta)$ denote a parameterized family of learnable functions (typically implemented with neural networks), parameterized by a set of parameters θ. Given an average loss function $\mathcal{L}(f(\theta), s)$, defined as $\mathcal{L}(f(\theta), s) = \frac{1}{|s|}\sum_{(x_i,y_i)\in s} L(y_i, f(x_i;\theta))$, we define the optimal parameters $\theta^*(s)$ for a training data subset s as:

$$\theta^*(s) = \arg\min_{\theta} \mathcal{L}(f(\theta), s) \tag{1}$$

Given an optimal parameter $\theta^*(s)$ and the test dataset $\mathcal{D}^t$, we define the value function $v(s)$, which provides a valuation of the subset s as:

$$v(s) = v(\theta^*(s), \mathcal{D}^t) = -\mathcal{L}(f(\theta^*(s)), \mathcal{D}^t) \tag{2}$$

The *high-value subset selection problem* can be defined as:

$$\max_{s\in\mathcal{S}} v(s) \;\; \text{sub. to } |s| \le \gamma n \tag{3}$$

where γ is the fraction of retained datapoints in the selected subset.

This problem is NP-Hard for a general value function $v(s)$. Next, we describe our broad framework to approximate the above problem with a two step approach:

1. Learn an embedding function $h(x, \phi)$ of the data point x, such that the distance between datapoints x_i and x_j, $d_{ij} = d(x_i, x_j) = D(h(x_i, \phi), h(x_j, \phi))$ represents their inverse-ability to replace each other for a given learning task. Here, D is a fixed distance function.
2. Select the set of most important datapoints s^* by minimizing the sum of distance between each datapoint and its nearest selected neighbor, and update the parameters ϕ of the embedding function such that $v(s*)$ is maximized.

The objective function for selection problem in second step can be written as:

$$s^* = \min_{s\in\mathcal{S}} \sum_{(x,y)\in\mathcal{D}} \min_{(x',y')\in s} d(x, x') \tag{4}$$

This objective function corresponds to the facility location problem [6], which can be relaxed to the following convex linear programming problem:

$$\min_{z_{ij}\in[0.1]} \sum_{i,j=1}^{n} z_{ij} d(x_i, x_j) \tag{5}$$

$$\text{sub. to} \sum_{j=1}^{n} z_{ij} = 1, \;\; \forall i = \{1, \ldots, n\}$$

Here $z_{ij} = 1$ indicate that the datapoint j is the nearest selected point or "representative" for data point i. Hence, the objective function sums the total distance of each point from its representative point, and the constraint enforces that every datapoint i should have exactly one representative point. A point j is selected if it is representative to at least one point, i.e. $\sum_{i=1}^{n} z_{ij} \ge 1$. Let u_j be the indicator variable for selection of datapoint j. For robustness, we calculate u_j as:

$$u_j = \frac{1}{\xi} \max\{\sum_{i=1}^{n} z_{ij}, \xi\} \tag{6}$$

where $0 \leq \xi \leq 1$ is a constant.

There are two main challenges in utilizing this formulation for the problem of optimizing the value function. Firstly, the optimal parameters $\theta^*(s)$ depend on the variables z_{ij} through variables u_j, which are output of an optimization problem. Hence, optimizing $v(s^*)$ w.r.t parameters (ϕ) of the embedding function requires differentiating through the optimization problem. Secondly, for most practical applications calculating the subset of training data, s, by optimizing the above optimization problem is too expensive for an iterative optimization algorithm which optimizes ϕ. Moreover, optimizing the parameters (θ) of the neural network model $f(\theta)$, typically involves an online stochastic optimization algorithm (e.g. SGD) which operates with mini-batches of training data points. In the next section, we describe the proposed technique for joint online training of both the learning model parameters θ and embedding model parameters ϕ.

3.2 Joint Online Training of Subset Selection and Learning Model

Our problem of selection of high-value subsets from training data can be described as joint optimization of value function for selection of best subset, and optimization of loss function on the selected subset computing the best model. Given training and test datasets $\mathcal{D}$ and $\mathcal{D}^t$, overall value function parameterised by the embedding model parameters ϕ for selection was defined as (Eq. 2):

$$v(s(\phi)) = v(\theta^*(s(\phi)), \mathcal{D}^t) = \frac{1}{|\mathcal{D}^t|} \sum_{(x,y)\in\mathcal{D}^t} L(y, f(\theta^*(s(\phi)), x)) \tag{7}$$

where $\theta^*(s(\phi)) = \arg\min_\theta \mathcal{L}(f(\theta), s(\phi))$ (from Eq. 1). Representing $s(\phi)$ in terms of the selection variables u_j (Eq. 6), we can write the model loss on selected training set using embedding model parameter ϕ as:

$$\mathcal{L}(\theta; \phi, \mathcal{D}) = \frac{1}{\gamma|\mathcal{D}|} \sum_{(x_j,y_j)\in\mathcal{D}} u_j(\phi) L(y_j, f(x_j; \theta)) \tag{8}$$

We note that the above objective functions $\mathcal{L}(\theta)$ and $v(s(\phi))$ are coupled in the variables ϕ and θ. On one hand, the loss on the selected training set $\mathcal{L}(\theta; \phi, \mathcal{D})$ depends on the current embedding model parameters ϕ. On the other hand, intuitively, the value $v(s(\phi))$ of a selected set of datapoints $s(\phi)$ should also depend on the current model parameter value θ', which maybe be updated using loss from the selected set of datapoints $s(\phi)$. Hence, we define the cumulative value function $\mathcal{V}(\phi; \theta', \mathcal{D}, \mathcal{D}^t)$ as:

$$\mathcal{V}(\phi; \theta', \mathcal{D}, \mathcal{D}^t) = v(\hat{\theta}, \mathcal{D}^t), \text{ where } \hat{\theta} = \theta' - \alpha \nabla_\theta \mathcal{L}(\theta; \phi, \mathcal{D}) \tag{9}$$

Here, $\hat{\theta}$ is the one step updated model parameter using the selected examples from dataset $\mathcal{D}$ using parameter ϕ, and α is the stepsize for the update. We combine the two objectives into a joint objective function, $J(\theta, \phi; \mathcal{D}, \mathcal{D}^t)$. The combined optimization problem becomes:

$$\theta^*, \phi^* = \arg\min_{\theta,\phi} J(\theta, \phi; \mathcal{D}, \mathcal{D}^t) = \arg\min_{\theta,\phi} (\mathcal{V}(\phi; \theta', \mathcal{D}, \mathcal{D}^t) + \mathcal{L}(\theta; \phi, \mathcal{D})) \tag{10}$$

As discussed above, since the training dataset $\mathcal{D}$ is normally much bigger in size, we design our algorithm to optimize the above objective function in an online manner. Hence the training dataset $\mathcal{D}$ is split into k equal-sized minibatches $\mathcal{D} = \{D_1, \ldots, D_k\}$. The above joint objective function can be decomposed as:

$$J(\theta, \phi; \mathcal{D}, \mathcal{D}^t) = \frac{1}{k}\sum_{i=1}^{k}\{\mathcal{L}(\theta; \phi, D_i) + \mathcal{V}(\phi; \theta', D_i, \mathcal{D}^t)\} \tag{11}$$

We solve the above problem using an online alternating minimization method similar to the one described in [3]. Our algorithm updates θ and ϕ as:

$$\begin{aligned}
&\text{Initialize } \theta^1 \text{ and } \phi^1 \text{ randomly}\\
&\text{for } t = 1, \ldots, T\ :\\
1:\quad &\theta^{t+1} = \theta^t - \alpha\frac{1}{k}\nabla_\theta \mathcal{L}(\theta; \phi^t, D_{i(t)})\\
2:\quad &\phi^{t+1} = \phi^t - \beta\frac{1}{k}\nabla_\phi \mathcal{V}(\phi; \theta^t, D_{i(t)}, \mathcal{D}^t)
\end{aligned}$$

Here $i(t)$ is the index of the minibatch chosen at update number t. Step 1 is the standard SGD update minimizing the loss over a subset of the minibatch $D_{i(t)}$ chosen using embedding parameters ϕ^t. Hence, it can be implemented using standard deep learning platforms. Implementation of step 2 poses two challenges: (1) computation of $\nabla_\phi \mathcal{V}(\phi; \theta^t, D_{i(t)}, \mathcal{D}^t)$ requires differentiation through an optimization problem, since $u_j(\phi)$ is the result of an optimization. (2) the datapoints in $D_{i(t)}$ must also be compared with datapoints in a previous minibatch for selecting the best representative points. Hence we need an online version of the optimization problem defined in Eq. 5. We describe solutions to these challenges in the next section.

3.3 Trainable Convex Online Subset Selection Layer

In order to implement the algorithm outlined in the previous section, we have to compute $\nabla_\phi \mathcal{V}(\phi; \theta^t, D_i, \mathcal{D}^t)$, which can be done by combining Eq. 7,8 & 9 as:

$$\begin{aligned}
\nabla_\phi \mathcal{V}(\phi; \theta^t, D_i, \mathcal{D}^t) = -\frac{1}{|\mathcal{D}^t|}&\sum_{(x_j, y_j)\in\mathcal{D}^t} \nabla_{\hat{\theta}} l(y_j, f(\hat{\theta}, x_j))\cdot \\
&\frac{1}{\gamma|D_i|}\sum_{(x_j, y_j)\in D_i}(\nabla_\phi u_j(\phi)\nabla_\theta L(y_j, f(x_j, \theta))
\end{aligned} \tag{12}$$

The key challenge here is computation of $\nabla_\phi u_j(\phi)$, since $u_j(\phi)$ is the output of an optimization problem. Also, the optimization problem in Eq. 5 selects from all datapoints in $\mathcal{D}$, whereas efficient loss minimization demands that we only select from the current minibatch $D_{i(t)}$. We use the online subset selection formulation, proposed in [6]. Let $\mathcal{O}(t)$ denote the set of old datapoints at time t which have already been selected, and let $D_{i(t)}$ denote the new set from which points will be selected. We use two sets of indicator variables: $z^o_{ij} = 1$ indicating that old datapoint j is a representative of new datapoint i, and $z^n_{ij} = 1$ indicating that new datapoint j is a representative of new datapoint i. Under this definition the optimization problem can written as:

$$\begin{aligned} \min_{z^o_{ij}, z^n_{ij} \in [0,1]} & \sum_{x_i \in D_{i(t)}, x_j \in \mathcal{O}(t)} z^o_{ij} d(x_i, x_j) + \sum_{x_i \in D_{i(t)}, x_j \in D_{i(t)}} z^n_{ij} d(x_i, x_j) \qquad (13) \\ \text{sub. to} & \sum_{x_j \in \mathcal{O}(t)} z^o_{ij} + \sum_{x_j \in D_{i(t)}} z^n_{ij} = 1, \ \forall i = \{1, \dots, n\} \\ & \sum_{x_j \in D_{i(t)}} u_j \leq \gamma |D_{i(t)}| \end{aligned}$$

where $u_j = \frac{1}{\xi} \max\{\sum_{x_i \in D_{i(t)}} z^n_{ij}, \xi\}$. The first constraint enforces that all new points must have a representative, and the second constraint enforces that the total number of representatives from the new set is less than $\gamma|D_{i(t)}|$. The objective function minimizes the total distance of between all points and their representatives. As previously, $d(x_i, x_j) = D(h(x_i, \phi), h(x_j, \phi))$, where $h(x, \phi)$ is the embedding of the point x using parameters ϕ.

In order to compute $\nabla_\phi u_j(\phi)$ we use the differentiable convex programming paradigm [1]. The DCP framework allows us to embed solutions to optimization problems as layers in neural networks, and also calculate gradients of output of the layers w.r.t. parameters in the optimization problem, so long as the problem formulation satisfies all the disciplined parameterized programming (DPP) properties. The above convex optimization problem generally satisfies all the properties of disciplined parameterized programming (DPP) [1], with parameter ϕ, except for the objective function. Specifically, the constraints are all convex and are not functions of parameters. The objective function is also linear in variables z^o_{ij}, but should be an affine function of the parameters ϕ in order to satisfy the DPP constraints. Hence, we use the affine distance function $D(a, b) = |a - b|$. Hence, the DCP framework allows us to calculate $\nabla_\phi u_j(\phi) = \nabla_{z^n_{ij}} u_j \nabla_\phi z^n_{ij}(\phi)$. Algorithm 1 summarizes the proposed framework for jointly learning a deep learning model (parameterized by θ) and a selection function for selecting a fraction γ of the training examples, parameterized by ϕ.

Computational complexity: Running time of the proposed algorithm scales linearly with the training dataset size. The algorithm also works at a time with a minibatch of training data, and hence is memory efficient. These two properties make the current algorithm work on very large training datasets, and complex deep learning models, provided there is enough computational power for inference of the model. A significant overhead of the current algorithm comes from solving an optimization problem for each parameter update. However, this cost can be controlled by keeping the minibatch sizes $|D_i|$ small, typically $|D_i| \leq 100$, which results in a linear optimization in 10000 variables. Another practical trick is to use a fixed number of datapoints for the old set $\mathcal{O}(t)$, where the datapoints can be chosen to be most similar to points in the minibatch D_i using a locality sensitive hashing based nearest neighbor search algorithm. Experimental results show that our algorithm takes a comparable amount of time as our closest competitor DVRL [19], while being much more accurate.

Algorithm 1 ***HOST-CP***: High-value Online Subset selection of Training samples through differentiable Convex Programming

```
1: Input:
2:   Training dataset $\mathcal{D}$ with minibatches $D_1, \ldots, D_k$, Test dataset $\mathcal{D}^t$, fraction of selected instances $\gamma$
3: Output:
4:   Model Parameter $\theta$, Embedding model for selection parameter $\phi$
5: Algorithm:
6: Initialize old set $\mathcal{O}(0) \rightarrow \Phi$ (Null set)
7: Initialize parameters $\theta^0$ and $\phi^0$ randomly.
8: for each timestep $t = 1, \ldots, T$ do
9:     Let $D_{i(t)}$ be the current minibatch.
10:    for each datapoint $x_i \in D_{i(t)}$ and $x_j \in \mathcal{O}(t)$, calculate embeddings $h(x_i, \phi)$ and $h(x_j, \phi)$
11:    for each pair $(i, j)$ , $x_i \in \mathcal{O}$ and $x_j \in D_{i(t)}$ calculate $d^o(x_i, x_j)$
12:    for each pair $(i, j)$ , $x_i \in D_{i(t)}$ and $x_j \in D_{i(t)}$ calculate $d^n(x_i, x_j)$
13:    Calculate $z^n_{ij}$ and $z^o_{ij}$ by solving optimization problem in Equation 13
14:    Calculate $u_j = \frac{1}{\xi} \max\{\sum_{x_i \in D_{i(t)}} z^n_{ij}, \xi\}$ for all $x_j \in D_{i(t)}$
15:    Include the selected points in the old set $\mathcal{O}(t+1)$ for which $u_j = 1$
16:    Calculate $\mathcal{L}(\theta^t; \phi^t, D_{i(t)})$ on the selected set by forward propagation.
17:    Calculate $\theta^{t+1} = \theta^t - \alpha \frac{1}{k} \nabla_\theta \mathcal{L}(\theta; \phi^t, D_{i(t)})$ by backpropagation.
18:    Calculate $\hat{\theta} = \theta^t - \alpha \nabla_\theta \mathcal{L}(\theta; \phi^t, D_{i(t)})$ followed by $\mathcal{V}(\phi; \theta', \mathcal{D}, \mathcal{D}^t)$ using Equation 9, where $\nabla_\phi u_j(\phi)$ is calculated using DCP as described in Section 3.3
19:    Update the embedding model parameter as: $\phi^{t+1} = \phi^t - \beta \frac{1}{k} \nabla_\phi \mathcal{V}(\phi; \theta^t, D_{i(t)}, \mathcal{D}^t)$
20: end for
```

4 Experiment

In this section, we provide our experimental setup by describing the datasets and the different data valuation baselines. In Section 4.2, we show the effectiveness of our proposed method - High-value Online Subset selection of Training samples through diferentiable Convex Programming(***HOST-CP***) over the baselines in terms of performance metric with addition and removal of high value data points. Next in Section 4.3, we show the use case of diagnosing mislabelled examples which, when fixed can improve the performance. Following that in Section 4.4, we provide an analysis of the quality of subsets obtained by one of the baselines, DVRL [19] and the proposed method. We also show a ranking evaluation of selected points and their correspondence with ground truth relevance in Section 4.5. Lastly, Section 4.6 describes the performance of the proposed method in terms of its computational complexities.

4.1 Experimental Setup

Dataset: We have considered datasets of different modalities for experiments. We use four types of datasets: CIFAR10 - a multi-class dataset for image classification, protein dataset [3] for multi-class protein classification, 20newsgroups [4] for text classification and a synthetic dataset for binary classification. Following [10], we generate 10 synthetic datasets, using third-order polynomial function.

For CIFAR10, we obtain the image features from a pre-trained ResNet-18 model and train a 2 layer neural network on the learned features. In case of synthetic data and protein data, we pass the datapoints through a 3 layer neural network, whereas in case of 20newsgroups, we pass the TF-IDF features of the documents through a 3 layer network for classification.

Baselines: We consider four state-of-the-art techniques as our baselines. Two of the baselines - Influence Function - IF [13] and TracIn-CP [16] use the influence function method for data valuation, one uses data shapley (DS) value [10] for selecting high value data points while the fourth one, DVRL [19] uses reinforcement technique while sampling, to rank the data points.

4.2 Valuable training data

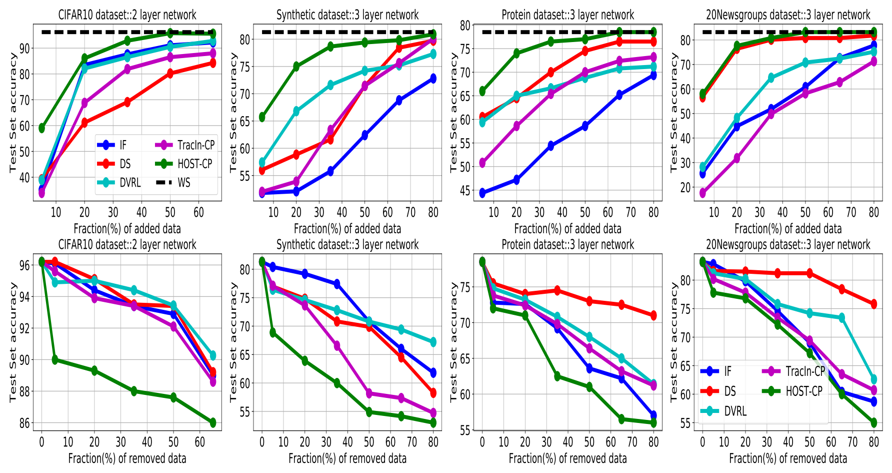

Fig. 1: **Addition (top) and removal (bottom) of valuable training datapoints:** We compare HOST-CP with the aforementioned baselines (IF[13], DS[10], TracIN-CP[16], DVRL[19]) in terms of accuracy on test set. For each method, we select (for addition) or discard (for removal) top k% important training points and assess their importance. HOST-CP performs better(*highest for addition and lowest for removal*) across all fractions and all datasets.

[3] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

[4] `https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html`

We compare our method[5]with state-of-the-art methods in terms of finding the most valuable training data points. We show the effectiveness of the proposed method by addition and removal of an incremental fraction of training instances to and from the dataset. We examine the test set performance for each fraction of selected or removed training points. The baseline methods are used to assign values to each data point, following which the top k% fraction of data are selected/removed from the whole dataset to train the classification network models. Our proposed method has the flexibility to optimally select the significant k% of data from the entire given dataset. We use the subsets of varying k or $(100 - k)$ fractions for training the networks and report the test set accuracies.

In Figure 1, we report the test set accuracies after addition and removal of high-value training data points, that get selected using the respective methods. We can observe in Figure 1 *(top)* that our method surpasses the baselines for all fractions and it approximately performs equivalent to the whole set (WS) within a fraction of 50 - 65%. We also note a similar trend in removal of valuable data points. We can observe in Figure 1 *(bottom)* that our method shows a significant drop in accuracy with removal of high-value data points. All the baselines show varying performances on different datasets. However, the proposed method consistently outperforms across all of them. This clearly suggests that HOST-CP is efficient enough to provide the optimal subset of training data points necessary for explaining the predictions on a given test set.

4.3 Fixing mislabelled data

Increase in data has led to rise in need of crowd sourcing for obtaining labels. Besides the chances of obtained labels being prone to errors [8], several data poisoning attacks are also being developed for mislabelling the data [17]. In this experiment, we impute 10% training data with incorrect labels, using which we compare our method with that of the baselines. The baseline methods consider that the mislabelled data shall lie towards the low ranked data points. We adopt the same idea and use the data points not selected by our method (reverse-selection) for analysis.

In this experiment, we inspect the bottom k% fraction (for baselines) and unselected k% fraction of the data (for the proposed method) for presence of wrong labels. We run this experiment on CIFAR10 and synthetic dataset and report the fraction of incorrect labels fixed with varying k. While imputing 10% of training data, we flip the labels of synthetic dataset since it consists of binary labels, while in case of CIFAR10, we change the original label to one out of 9 classes. We can observe in Table 1 that the proposed method is able to find comparable mislabelled fraction of incorrect labels in comparison to the baselines. We also run another experiment where, we fix the incorrect labels in the considered k% fraction of data and train the entire data. Figure 2 shows a rise in test set accuracy with a significant margin using the proposed method. This

[5] `https://github.com/SoumiDas/HOST-CP`

denotes the effectiveness of HOST-CP, in diagnosing mislabelled examples which turns out to be an important use-case in data valuation techniques.

Table 1: **Diagnosing mislabelled data:** We inspect the training points starting from the least significant data and report the fraction of labels fixed from the inspected data. Overall, HOST-CP detects a higher rate of mislabelled data.

Fraction of data checked (%)	Fraction of incorrect labels fixed									
	CIFAR10					Synthetic Data				
	IF	DS	DVRL	TracIn-CP	HOST-CP	IF	DS	DVRL	TracIn-CP	HOST-CP
20	0.21	0.19	0.195	0.20	**0.23**	0.22	0.22	0.19	0.23	0.23
35	0.35	0.34	0.336	0.35	**0.36**	0.34	0.37	0.33	0.35	**0.39**
50	0.50	0.50	0.49	0.50	**0.51**	0.43	0.54	0.46	0.51	**0.55**
65	0.61	0.61	0.65	0.63	**0.67**	0.56	0.67	0.58	0.68	**0.68**
80	0.74	0.78	0.80	0.79	**0.81**	0.66	0.81	0.79	0.81	**0.83**

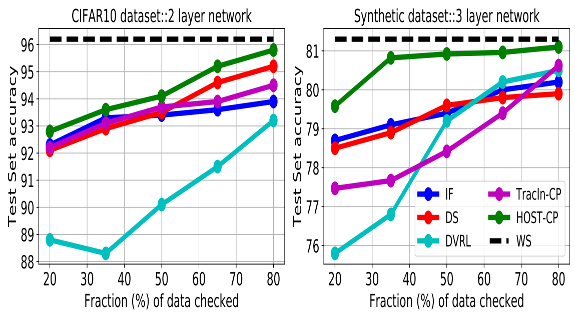

Fig. 2: **Accuracy after fixing mislabelled data:** For every k% fraction of inspected data, we fix the mislabelled instances and re-train. HOST-CP shows improved performance with each fixing of labels.

4.4 Qualitative analogy of data valuation

As we had mentioned earlier, methodologically, DVRL [19] is the closest method to our approach. Hence, in order to qualitatively explain the difference in performance between that of DVRL and the proposed method, we try to analyse the quality of subsets obtained by both of them on CIFAR10 dataset. Figure 3 shows the fraction of selected images across classes by both the methods. We compare the top 5% selected subset by DVRL and the proposed method in terms of class distribution. Unlike HOST-CP which compares the past selected data points with the incoming points to compute the new subset, DVRL keeps track of past average test losses and values the instances based on the marginal difference with the current loss. This leads to a certain bias in case of DVRL, towards some particular class of images or similar high-performing datapoints which are more contributory towards reduction in test set losses. Following this, we observe that HOST-CP selects a diverse set of samples, thus justifying its better performance.

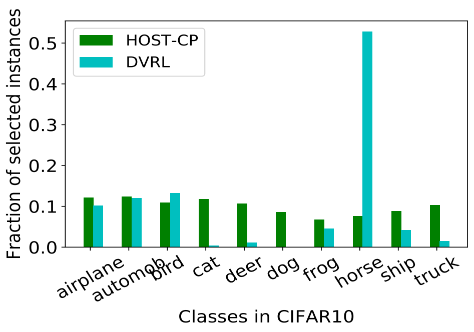

Fig. 3: **Quality of subset:** We observe that unlike DVRL, HOST-CP selects diverse images across the classes leading to better performance.

4.5 Ranking function for value points

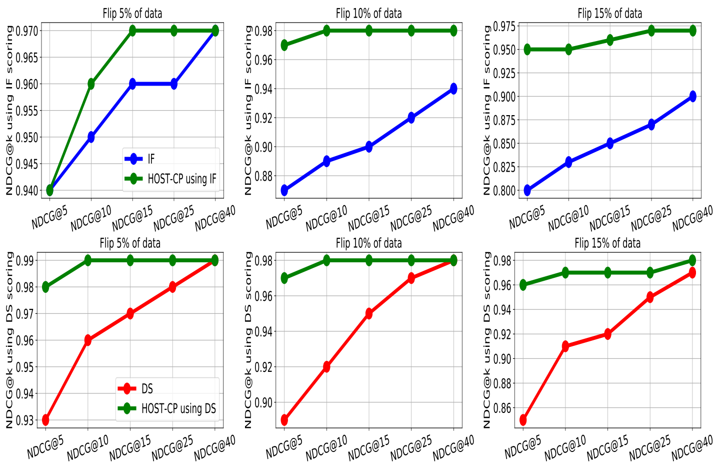

Fig. 4: **Ranking of data points:** We select k% fraction of data for each method and provide scores for each datum. While IF and DS subsets inherently come along with scores, the proposed method, HOST-CP returns a subset which is scored using IF and DS. Following the acquisition of scores for each data point, we use NDCG@k to calculate the ranking values. The proposed method is found to have a higher NDCG score.

Unlike the baseline methods which provide a score to each data point, HOST-CP has no such analogous scoring function. It is designed to return the explainable subset necessary for optimum performance on a given test set. We design this experiment to assess the significance of the selected data points in terms of ranking function values. We use NDCG score for reporting the ranking value.

We use the synthetic dataset and create a series of ground-truth values by flipping n% {=5,10,15} of the data. We assign ground-truth to the flipped and unflipped points with 0 and 1 respectively. We examine the bottom(or low-value) k% of the data for computing the ranking values since the flipped points are more inclined to lie towards the bottom. As we have observed in Section 4.3, we use the reverse-selection procedure to obtain the bottom k% of data for the proposed method, while for the baselines, the low-valued k% points occupy the bottom rung. In order to compute NDCG scores, we adopt the two baselines [13],[10] to provide scores to the subset of points obtained using the proposed method. In case of baselines, we use the k-lowest scoring points for computing rank values.

We flip 5%, 10% and 15% of the data and compute NDCG@k for k $\in$ {5,10,15,25,40}. Here k refers to the bottom k% subset from the methods. In Figure 4, we compare the respective ranking values(NDCG) of subsets obtained using influence function (IF) or data shapley (DS) with the subset obtained using the proposed method followed by usage of the corresponding scoring method (IF or DS) on this acquired subset. We can observe that NDCG values obtained by the proposed method's subset followed by scoring using IF or DS always stays ahead than using the subsets obtained by IF or DS, starting from an early fraction. This shows that the subset obtained using HOST-CP is efficient enough in keeping the proportions of relevant(unflipped) and non-relevant(flipped) points in their respective positions leading to a higher-NDCG score.

4.6 Computational complexity

We analysed the computational complexity of the proposed method on a 64-bit machine with one Quadro-P5000 GPU. For this experiment, we generated synthetic datasets with varying number of training datapoints (500,2000,4000,6000,10000). Keeping the network architecture fixed at a 3 layer network, we varied the size of the training data, on which the subset is meant to be computed. We also varied the size of the test datapoints for which explanations are sought to be found from the training datapoints. We report the time taken to perform the joint subset-training over an epoch, using the proposed method, which aims at finding the best set of explanations from the training set for a given set of test data. Figure 5 shows a linear pattern in time (in minutes) consumed by the proposed method across the varying size of datapoints.

We have earlier observed that the proposed method is consistently performing better than all the baselines. Since DVRL is the closest to our approach, we also record the time taken by DVRL and the proposed method on the synthetic dataset. We increase the number of iterations for DVRL to trace if the accuracy obtained by the proposed method is achievable by DVRL under a certain subset cardinality (5%, 20%). We can observe that DVRL saturates at a certain

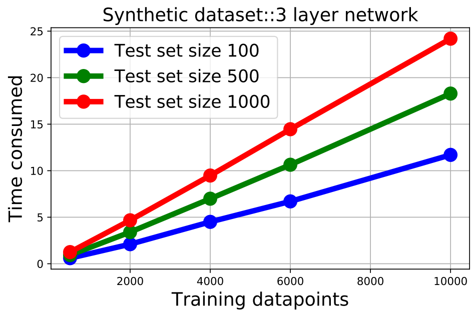

Fig. 5: **Scaling with data:** We show that the change in time consumed by HOST-CP with variation in training data points and test data points using a fixed network is linear with increasing data.

Table 2: **Time comparison**: We compare the times consumed by DVRL and HOST-CP. We observe that HOST-CP has a lower running time than DVRL.

Accuracy (5%)		**Accuracy (20%)**		**Epochs (HOST-CP)/ Iterations (DVRL)**		**Time (mins)**	
HOST -CP	DVRL	HOST- -CP	DVRL	HOST- -CP	DVRL	HOST- -CP	DVRL
62.46	57.8	71.32	66.4	1	25	1.1	3.40
63.34	57.2	73.82	66.8	2	50	2.01	5.31
65.14	57.4	74.89	66.5	3	100	3.05	6.50
65.71	57.6	75.0	66.6	4	150	4.08	10.57
65.72	57.4	75.0	66.4	5	200	5.12	14.26

accuracy value even after increasing the number of iterations to a much higher value. Thus, our method, HOST-CP besides attaining a higher accuracy, also takes considerably lower running time than that of DVRL.

5 Conclusion

In this paper, we propose a technique for finding high-value subsets of training datapoints essential for explaining the test set instances. We design a learning convex framework for subset selection, which is then used to optimise the subset with respect to a differentiable value function. The value function is essentially the performance metric which helps to evaluate the trained models. We compare our method against the state-of-the-art baselines , influence functions [13], data shapley [10], TracIn-CP [16] and DVRL [19] across different datasets in a range of applications like addition or removal of data, detecting mislabelled examples. We also analyse the quality of obtained subsets from DVRL and the proposed method. Lastly, we show that the proposed method scales linearly with the dataset size and takes a lower running time than DVRL which is the closest method to our approach methodologically. Thus, we are able to show that HOST-CP outperforms the baselines consistently in all the applications used for the experiments, thus proving its efficiency in terms of providing high-value subsets.

Acknowledgements

This project is funded by Hewlett Packard Labs, Hewlett Packard Enterprise.

References

1. Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., Kolter, J.Z.: Differentiable convex optimization layers. In: NeurIPS (2019)
2. Buchbinder, N., Feldman, M., Naor, J., Schwartz, R.: Submodular maximization with cardinality constraints. In: ACM-SIAM SODA (2014)
3. Choromanska, A., Cowen, B., Kumaravel, S., Luss, R., Rigotti, M., Rish, I., Diachille, P., Gurev, V., Kingsbury, B., Tejwani, R., et al.: Beyond backprop: Online alternating minimization with auxiliary variables. In: ICML. PMLR (2019)
4. Cook, R.D., Weisberg, S.: Residuals and influence in regression. New York: Chapman and Hall (1982)
5. Das, S., Mandal, S., Bhoyar, A., Bharde, M., Ganguly, N., Bhattacharya, S., Bhattacharya, S.: Multi-criteria online frame-subset selection for autonomous vehicle videos. Pattern Recognition Letters (2020)
6. Elhamifar, E., Kaluza, M.C.D.P.: Online summarization via submodular and convex optimization. In: CVPR (2017)
7. Elhamifar, E., Sapiro, G., Sastry, S.S.: Dissimilarity-based sparse subset selection. IEEE TPAMI (2015)
8. Frénay, B., Verleysen, M.: Classification in the presence of label noise: a survey. IEEE transactions on neural networks and learning systems (2013)
9. Ghorbani, A., Kim, M., Zou, J.: A distributional framework for data valuation. In: ICML. PMLR (2020)
10. Ghorbani, A., Zou, J.: Data shapley: Equitable valuation of data for machine learning. In: ICML. PMLR (2019)
11. Ghorbani, A., Zou, J.Y.: Neuron shapley: Discovering the responsible neurons. In: NeurIPS (2020)
12. Hara, S., Nitanda, A., Maehara, T.: Data cleansing for models trained with sgd. In: NeurIPS (2019)
13. Koh, P.W., Liang, P.: Understanding black-box predictions via influence functions. In: ICML. PMLR (2017)
14. Lundberg, S., Lee, S.I.: A unified approach to interpreting model predictions. arXiv preprint arXiv:1705.07874 (2017)
15. Lundberg, S.M., Erion, G.G., Lee, S.I.: Consistent individualized feature attribution for tree ensembles. arXiv preprint arXiv:1802.03888 (2018)
16. Pruthi, G., Liu, F., Kale, S., Sundararajan, M.: Estimating training data influence by tracing gradient descent. NeurIPS (2020)
17. Steinhardt, J., Koh, P.W., Liang, P.: Certified defenses for data poisoning attacks. In: NIPS. NIPS'17 (2017)
18. Wu, Y., Dobriban, E., Davidson, S.: Deltagrad: Rapid retraining of machine learning models. In: ICML. PMLR (2020)
19. Yoon, J., Arik, S., Pfister, T.: Data valuation using reinforcement learning. In: ICML. PMLR (2020)