Data2Model: Predicting Models from Training Data

***Abstract*—Understanding how changes in training data affect a trained model is critical to building trust in various stages of a machine learning pipeline: from cleaning poor-quality samples and tracking important ones to be collected during data preparation, to calibrating uncertainty of model prediction, to interpreting why certain behaviors of a model emerge during deployment. In this paper, we present a framework, DATA2MODEL, for predicting the output model of a learning algorithm given the input data points. Specifically, DATA2MODEL learns a parameterized function that takes a dataset S as the input and predicts the model obtained by training on S. Despite the potential complexity of the underlying end-to-end training process being approximated, we show that a neural network-based set function class can successfully predict the trained model from its training data. We introduce novel global and local regularization techniques for preventing overfitting and rigorously characterize the expressive power of neural networks (NN) in approximating the end-to-end training process. We perform extensive empirical investigations and demonstrate that DATA2MODEL gives rise to a wide range of applications that boost interpretability and accountability of machine learning (ML), such as data valuation, data selection, memorization quantification, and model calibration.**

I. INTRODUCTION

What are the training points with the most or least contribution to the model? How to select data that benefit model performance? Is a training data point memorized by the model? How to produce accurate estimates of uncertainty on model predictions? Which point is the most responsible for learning a given parameter in the model?

Answering these questions is central to building trust in machine learning (ML). Prior work shows that addressing these questions requires analyzing the input-output behavior of a learning process and gaining an understanding of how models (and the resulting predictions) might have differed if different training data points are observed [1, 2, 3, 4].

However, due to the complexity of the learning process (e.g., end-to-end training), analyzing how its input—a dataset—affects the output—the trained model—is challenging. Recent work has made significant progress in approximating how a small change (e.g., removing one or a handful of training points) on the full training set would change the trained model [5, 6, 7, 8], but the existing techniques remain limited in estimating the effects of removing a large group of training points [9]. Ilyas et al. [10] propose to learn a model that predicts the classification performance at a given test point directly from the training data. However, this (expensive) learning procedure will need to be redone if the classification performance at a different test point is of interest to the evaluation; therefore, it lacks the flexibility to accommodate different evaluation goals.

In this paper, we propose a framework, DATA2MODEL, to analyze the dependence of the trained models on training data by building an explicit model for the trained models in terms of training data. Compared to [10], DATA2MODEL estimates the trained model parameters instead of the prediction performance at a certain test point, thereby providing flexibility to switch to different test points for evaluation. At the same time, the test performance estimates based on DATA2MODEL are more accurate than those from [10] because DATA2MODEL effectively leverages the knowledge of how test performance is calculated from a trained model.

Our contributions are summarized as follows:

#1: DATA2MODEL—a framework to understand the input-output behavior of a learning algorithm. Consider a training set S and a learning algorithm $\mathcal{A}$ that takes in a set of data points and outputs a trained model parameterized by θ. A typical example of $\mathcal{A}$ is empirical risk minimization (ERM), which returns an optimum of a classification or regression learning objective function. The key idea of DATA2MODEL is to construct a cheap-to-evaluate surrogate function $\widehat{\mathcal{A}}$ whose output roughly matches $\mathcal{A}(\cdot)$ for a variety of training sets S. $\widehat{\mathcal{A}}$ will be referred to as a **DATA2MODEL function** hereinafter. By parameterizing the surrogate function $\widehat{\mathcal{A}}$, we transform the challenge of constructing a surrogate function into a supervised learning problem. In this problem, the feature in each training example is a set S and the corresponding label is given by $\mathcal{A}(S)$.

#2: Instantiating DATA2MODEL. There are several critical design choices of DATA2MODEL: What is the right model class to learn $\widehat{\mathcal{A}}$? How to design a proper objective function to fit $\widehat{\mathcal{A}}$? Ilyas et al. [10] converts an input subset S' to a binary vector that encodes the presence of each training sample of S within S'. While this reduces the input complexity, the contents of an input subset are ignored; therefore, the resulting approach can only estimate the outcome of learning for a subset *within* the training dataset yet fails to estimate the learning result for new *unseen* points. In this paper, we use a neural-network-based *set function* to approximate $\mathcal{A}$, which takes in the actual contents of a subset and predicts the trained model. This design enables the learned $\widehat{\mathcal{A}}$ to generalize to subsets involving unseen points. To mitigate overfitting, we propose two novel regularization techniques: the global regularizer encourages that the models predicted by the fitted $\widehat{\mathcal{A}}$ predict the labels in the held-out set as effectively as those obtained from the original $\mathcal{A}$; and the local one ensures that the gradient of the classification or regression learning objective function with respect to these models is close to zero.

#3: Approximatability of a learning algorithm with neural networks. When instantiating DATA2MODEL, we choose to approximate $\mathcal{A}$ with a neural network. While a neural

network seems a reasonable first choice given its popularity in various vision and language tasks, it remains unclear whether a neural network can approximate complex training processes from end to end. This paper presents the *first* formal study of the approximatability of a learning process with neural networks. In particular, our results suggest that learning processes associated with strongly convex or smooth problems can be efficiently approximated by neural networks with ReLU units.

#4: Applications of DATA2MODEL. We explore a variety of applications of DATA2MODEL towards answering the questions posed at the beginning of the section. Our findings are summarized as follows.

- **DATA2MODEL functions successfully predict trained models from the training data (Table II and III).** The Spearman correlation between the accuracy of the predicted models and the ground-truth is mostly greater than 97%.
- **DATA2MODEL functions enable the discovery of the connection between data and model parameters (Figure 4 and Figure 5).** The level of sensitivity of a ground-truth model's parameter to different points matches the results obtained from a predicted model.
- **DATA2MODEL functions successfully boost the accuracy of data valuation—measuring the contribution of individual points to learning (Table IV).** While accurately measuring the value of data has a great potential to improve model performance and transparency, being more accurate comes with significant computational costs. In particular, existing data value notions require retraining a target model from scratch on many subsets of the training set and evaluating the corresponding model performance scores. We leverage the ability of DATA2MODEL functions to predict models directly from training data and significantly improve the accuracy of data valuation.
- **DATA2MODEL functions lead to improved data selection in the presence of bad data (Figure 7).** We leverage the Shapley value (SV) estimated through DATA2MODEL functions to rank the contributions of all training points and select the ones with the highest contributions. DATA2MODEL functions lead to a significant improvement in data selection effectiveness while maintaining a similar computation cost to conventional Shapley value estimation methods.
- **DATA2MODEL functions lead to more efficient and accurate quantification of model memorization (Figure 8).** To assess whether a given training point is memorized or not, we evaluate DATA2MODEL functions on different subsets with the point included and excluded and then compare the prediction at the point. We find that DATA2MODEL functions enable the successful discovery of memorized training samples.
- **DATA2MODEL functions improve the accuracy of prediction confidence (Table V).** We incorporate the predictions of the model's output by DATA2MODEL functions into an ensemble, and it leads to an improvement in the accuracy of prediction confidence on various datasets.

II. APPROACH

A. Preliminaries

Denote a data point as (x, y), where $x \in \mathbb{R}^d$ is the feature vector and $y \in \mathbb{R}$ is the label. We define $f(x;\theta)$ as the **Base Model**, which is a parametric function that is parameterized by $\theta \in \mathbb{R}^{d_{\text{param}}}$ and maps an input feature to a label (*i.e.*, logistic regression model). The loss function $\ell(\theta;(x,y)) = \ell(f(x;\theta), y)$ is defined to measure the discrepancy between the prediction $\widehat{y} = f(x;\theta)$ and the true label y, which can be a classification or regression learning objective function . Denote the full training dataset containing n training data points as $D = \{(x_i, y_i)\}_{i=1}^n$. We further denote S as a subset with n_S data points sampled from D. A *learning algorithm* (*i.e.*, *solver*) $\mathcal{A}$ is defined as the mapping from an input dataset S to the parameters of the trained model $\widehat{\theta}_S$ *i.e.*, $\widehat{\theta}_S = \mathcal{A}(S)$. A typical example of a learning algorithm is empirical risk minimization (ERM) with ℓ_2 penalty. Specifically, the parameters are optimized to minimize the training loss averaged over the subset S:

$$\widehat{\theta}_S := \underset{\theta}{\operatorname{argmin}}\, L(\theta; S) = \frac{1}{n_S} \sum_{i \in S} \ell(\theta; (x_i, y_i)) + \frac{\lambda}{2} \|\theta\|_2^2 , \tag{1}$$

where λ is a hyperparameter that controls the degree of penalization on the ℓ_2-norm of the estimated parameters to increase the model's generalizability.

B. Algorithm

1) Problem Formulation.

As mentioned in the Section I, our goal is to predict the outcome of the learning algorithm $\mathcal{A}$ given training set S. While $\mathcal{A}(S)$ is a complicated function that typically involves loss function optimization, we leverage a classic technique in machine learning: picking a suitable function class and finding the function in the class that best approximates the target function. To this end, we construct DATA2MODEL as a surrogate function $\widehat{\mathcal{A}}$ to obtain outputs that match $\mathcal{A}(\cdot)$ given an arbitrary training set S.

Definition 1 (DATA2MODEL function)**.** *Given a data domain $\mathcal{Z}$, a learning algorithm $\mathcal{A}$, a distribution $\mathcal{D}$ over the space of data* sets $2^{\mathcal{Z}}$*, for any dataset S drawn from $\mathcal{D}$, let $\widehat{\theta}_S = \mathcal{A}(S)$ be the model trained on S using $\mathcal{A}$. The proposed* DATA2MODEL *function is a parametric function $\widehat{\mathcal{A}}$ optimized to predict the $\widehat{\theta}_S$ from a training set $S \sim \mathcal{D}$,* i.e.,

$$\widehat{\mathcal{A}} : 2^{\mathcal{Z}} \to \mathbb{R}^{d_{\text{param}}}, \tag{2}$$

$$\text{where} \quad \widehat{\mathcal{A}} = \underset{\widetilde{\mathcal{A}}}{\operatorname{argmin}}\, \mathbb{E}_{S \sim \mathcal{D}} \left[L\left(\widetilde{\mathcal{A}}(S), \mathcal{A}(S) \right) \right], \tag{3}$$

where $L(\cdot,\cdot)$ is a loss function.

2) Instantiating

To develop a generic framework that not only can predict a model from the subsets within the training dataset, but also from *unseen* subsets drawn from the same distribution $\mathcal{D}$, the contents of input data are preserved to be used as the input for DATA2MODEL function, which makes $\mathcal{A}(\cdot)$ a set function. Taking this into consideration, we choose deep neural networks (DNNs) as the function class for DATA2MODEL function $\widehat{\mathcal{A}}$ given their strong expressive power. Moreover, since $\mathcal{A}(\cdot)$ is a set function, the architectures of neural networks are restricted to those that are invariant to input permutations. Finding the best neural network that approximates $\mathcal{A}(\cdot)$ becomes a *supervised learning* problem when there are samples of $(S, \mathcal{A}(S))$ available.

3) Design of the Loss Function L

Despite the strong expressiveness, with large size of parameters, the DNN adopted as the surrogate model can be easily overfitted on the training set if we only push the model to minimize the discrepancy between the predicted and actual model parameters. To prevent the DNN from overfitting, we propose two novel regularization techniques, where the utility loss is proposed to maintain the utility of the predicted model as a global regularizer, while the optimality of the predicted parameters is imposed by the Karuch-Kuhn-Tucker (KKT) loss as a local regularizer.

Global Regularizer. From an overall perspective, an accurately predicted model should maintain a utility close to the ground-truth model when evaluated on any test points, intuitively. For the utility loss as a global regularizer, denote $\mathcal{U}(S; \widehat{\theta})$ as the utility function that evaluates the utility of optimized model parameters $\widehat{\theta}$ on a dataset S. For one training sample, $(\widehat{\theta}_S; S)$, the utility loss is specified as:

$$L_U = \left| \mathcal{U}(S; \widehat{\mathcal{A}}(S)) - \mathcal{U}(S; \widehat{\theta}_S) \right|.$$

Local Regularizer. From a local perspective, an accurately predicted model should satisfy the optimality conditions, which constrains the gradient of the predicted parameters with respect to the learning objective function. With this in mind, we proposed to employ KKT conditions in the loss function, which are found to guarantee the optimality of a solution [11] for convex problems. Specifically, for an unconstrained convex problem (1), the optimal parameters $\theta_S^* \in \operatorname{argmin}_{\theta \in \mathbb{R}^{d_{\text{param}}}} L(\theta; S)$ should satisfy the stationary KKT condition as follows:

$$\nabla L(\theta_S^*; S) = 0. \tag{4}$$

If the learning algorithm can find a near-optimal solution, then the stationary KKT condition is almost satisfied by the optimized parameter $\widehat{\theta}_S$. Therefore, to compel the KKT condition, the KKT loss of one training sample is added as: $L_{KKT} = \left\| \nabla L\left(\widehat{\mathcal{A}}(S); S\right) \right\|$. This condition also applies to convex problems with hard constraints where the KKT stationary condition can be enforced on the Lagrangian of the optimization problem. Besides, the stationary condition can be treated as the first-order optimality condition for nonconvex optimization problems. Therefore, by adding KKT regularization, we enhance the generalization of DATA2MODEL for generic optimization problems, including neural networks.

Combining the loss on the discrepancy between prediction and groud-truth with proposed regularizers, the total loss of one training sample for the proposed DNN is:

$$L_{\text{DNN}} = \left\| \widehat{\theta}_S - \widehat{\mathcal{A}}(S) \right\| + L_{KKT} + L_U. \tag{5}$$

4) Algorithm Details

The proposed DATA2MODEL framework proceeds in two phases: *offline training* and *online estimation* (Figure 1.)

Offline Training. The offline training of DATA2MODEL consists of the parameters sampling step and DNN training step. With the set of training samples $\Phi = \{(S_1, \widehat{\theta}_1), (S_2, \widehat{\theta}_2), \dots\}$, the objective of offline training phase is to train an effective parameter model which can predict the optimized parameters accurately given new input data points.

The very first question encountered during the training phase is how to generate training subsets and what distribution should be used.

Denote π as a permutation of all training data points from the full set and $D_{\pi[:i]}$ as a set of points that precedes the i-th point in π. We adopt permutation sampling to sample the trained parameters $\mathcal{A}(D_{\pi[:i]})$ for training DNN-based $\widehat{\mathcal{A}}$. The offline training workflow is summarized in Algorithm 1.

Algorithm 1: DATA2MODEL Offline Training

input : Full dataset $D = \{(x_i, y_i)\}_{i=1}^n$, learning algorithm $\mathcal{A}$, the number of permutations T
output: Trained $\widehat{\mathcal{A}}$

```
Φ ← ∅
for t = 1, ..., T do
    π_t ← GenerateUniformRandomPermutation(D)
    for i = 1, ... n do
        θ̂_i^{π_t} = A(D_{π_t[:i]})
        Φ = Φ ∪ {(D_{π_t[:i]}; θ̂_i^{π_t})}
    end
end
Trained Â with Φ
return Â
```

Compared with other kinds of subset sampling strategies such as sampling uniformly at random, permutation sampling enables the evaluation of the change of model parameters with slightly varied subsets. Therefore, the training samples collected by permutation can better reflect the effect of individual data points, thus, allowing the DNN to better learn their influences in the trained parameters. Thus, permutation sampling is used to generate the training subsets in the offline training of DATA2MODEL. The empirical experiment results show a slight advantage of permutation sampling over uniform sampling when we evaluate the performance of the DNN trained by subsets. Note that if the distribution of

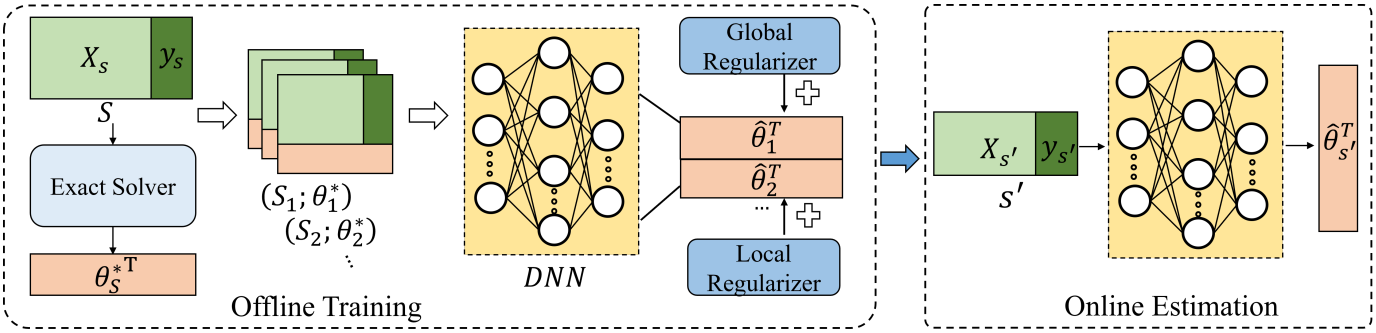

Fig. 1: The offline training phase and online estimate of the proposed DATA2MODEL.

the subset to be estimated in the testing phase (*i.e.*, test distribution) is known a prior, it is suggested to directly sample the training samples from the test distribution. For instance, in data valuation by SV, the definition of SV requires the distribution to assign uniform probability to all data sizes and uniform probability to all subsets of a given size. Then, in the offline phase, the subsets should be sampled from the distribution above.

Online Estimation. After the offline training phase, the trained DNN $\widehat{\mathcal{A}}$ can be used for efficient model parameters prediction given a new training subset or a batch of subsets through an evaluation of $\widehat{\mathcal{A}}$. Note the subsets are not limited to those containing observed data points.

III. Characterization of Efficiently Approximatable Learning Algorithms

In this paper, we try to use a neural network to fit the function $\widehat{\theta} = \mathcal{A}(D)$. When the context is clear, we omit the learning algorithm and simply write the function as $\widehat{\theta}(D)$. We denote the function associated with kth parameter as $\widehat{\theta}_k(D)$, and thus $\widehat{\theta}(D) = (\widehat{\theta}_1(D), \ldots, \widehat{\theta}_{d_{\text{param}}}(D))$, where d_{param} is the number of model parameters. Although D is a set of data points, it can be equivalently viewed as a vector concatenation of all (x_i, y_i). We assume we always normalize input features to $[0,1]$. Therefore, we have $D \in [0,1]^{n(d+1)}$. In order to fit $\widehat{\theta}(D)$ with a neural network, the very first question is *whether $\widehat{\theta}(D)$ could be efficiently expressed or approximated by neural networks of certain structures.* Here, the efficiency is measured by the total number of computational units in the neural networks.

Despite the strong expressive power of neural networks, not every function could be efficiently approximated by them. Particularly, while the famous universal approximation theorem (UAT) includes many functions, there are requirements regarding the continuity of the functions and compactness of the domain. A recent study [12] shows that *functions $g : \mathbb{R}^d \to \mathbb{R}$ that has a smaller upper bound of gradient norm $\|\nabla g\|$ could be more efficiently approximated by neural networks with ReLU activation*. As long as $\widehat{\theta}_k(D)$ could be efficiently approximated by certain neural network architectures, we could put d_{param} such networks in parallel to approximate $\widehat{\theta}(D)$. The parallel networks share the same input but no internal connections and computational units. Therefore, we can *reduce the problem of understanding the efficient approximatability of a function to that one of bounding its gradient norm*. Particularly, this section aims to understand *sufficient* conditions for which $\left\|\frac{\partial \widehat{\theta}_k}{\partial D}\right\|$ could be upper bounded.

Our first result is under the setting that the learning algorithm $\mathcal{A}$ is able to find an *optimal parameter* $\theta^* \in \operatorname{argmin}_\theta L(\theta; D)$. There are three major assumptions in our result: (1) the loss function $\ell(\theta)$ is α-strongly convex, (2) $\left\|\frac{\partial}{\partial\theta\partial z_i}\ell(\theta, z)\right\|$ is upper bounded by some constant B_1, and (3) $\|\theta^*\| \le B_2$.

Definition 2 (α-strongly convex)**.** *A differentiable function f is α-strongly convex if*

$$f(y) \ge f(x) + \nabla f(x)^T (y - x) + \frac{\alpha}{2}\|y - x\|^2, \quad (6)$$

for some $\alpha > 0$ and for all x, y in the domain.

Since $\ell(\theta; (x, y))$ is α-strongly convex, the training loss $L(\theta; D) = \frac{1}{n}\sum_{i=1}^n \ell(\theta; (x_i, y_i)) + \frac{\lambda}{2}\|\theta\|_2^2$ is $(\alpha+\lambda)$-strongly convex. Together with the condition that $\|\theta^*\|$ is finite, it implies L has a unique global minimum $\theta^* = \operatorname{argmin}_\theta L(\theta; D)$. Our main result for strongly convex functions is below:

Theorem 1. *If for all $z \in [0,1]^d$, the loss function $\ell(\theta; z)$ is α-strongly convex in θ, and $\left\|\frac{\partial}{\partial\theta\partial z_i}\ell(\theta, z)\right\| \le B_1$ for $i \in [d]$, then for $\theta_k(D) = [\operatorname{argmin}_\theta L(\theta; D)]_k$, where $L(\theta; D) = \frac{1}{n}\sum_{i=1}^n \ell(\theta; (x_i, y_i)) + \frac{\lambda}{2}\|\theta\|_2^2$ and $[\cdot]_k$ means the kth element in the vector, then we have*

$$\left\|\frac{\partial \theta_k}{\partial D}\right\| \le B_1 \frac{\sqrt{d_{\text{param}}(d+1)}}{\sqrt{n}(\alpha+\lambda)}. \quad (7)$$

We then analyze the case where the learning algorithm $\mathcal{A}$ is gradient descent. Gradient descent is a commonly used optimization algorithm in machine learning. Formally, suppose the parameter θ is initialized by 0 and denoted by $\theta^{(0)}$. For the ith iteration, we update $\theta^{(t)} = \theta^{(t-1)} - \eta\nabla L(\theta^{(t-1)}; D)$ with a learning rate η. Given the maximum step number T, we have $\widehat{\theta}(D) = \theta^{(T)}$. The only different assumption for bounding $\left\|\frac{\partial \widehat{\theta}_k}{\partial D}\right\|$ in this case is that the loss function $\ell(\theta)$ is β-smooth instead of α-strongly convex.

Definition 3 (β-smoothness)**.** *A differentiable function f is*

β-smooth if

$$\|\nabla f(x) - \nabla f(y)\| \leq \beta \|x - y\| \tag{8}$$

for some $\beta > 0$, and for all x, y in the domain.

Theorem 2. *If for all $z \in [0,1]^d$, the loss function $\ell(\theta; z)$ is β-smooth in θ, and $\left\| \frac{\partial}{\partial \theta \partial z_i} \ell(\theta, z) \right\| \leq B_1$ for $i \in [d]$, then for $\theta_k^{(t)}(D)$ defined by the kth entry of $\theta^{(t)}$, which is iteratively computed by $\theta^{(t)} = \theta^{(t-1)} - \eta \nabla L(\theta^{(t-1)}; D)$ and $\theta^{(0)} = 0$ with a learning rate $\eta > 0$, and $L(\theta; D) = \frac{1}{n} \sum_{i=1}^{n} \ell(\theta; (x_i, y_i)) + \frac{\lambda}{2} \|\theta\|_2^2$, we have*

$$\left\| \frac{\partial \theta_k^{(t)}}{\partial D} \right\| \leq \left(1 - (1 - \eta\lambda - \eta d_{\text{param}} \beta)^t\right) \frac{B_1 \sqrt{(d+1)}}{\sqrt{n}(\lambda + d_{\text{param}} \beta)}.$$

Theorem 1 and 2 demonstrate that $\left\| \frac{\partial \widehat{\theta}_k}{\partial D} \right\|$ could be upper bounded the dimension of data domain d and model parameter dimension d_{param}, and the dependency is only $O(\sqrt{d_{\text{param}} d / n})$ for the case of $\mathcal{A}$ being able to find the optimal parameter, and $O(\sqrt{d} / d_{\text{param}} \sqrt{n})$ for the case of gradient descent.

IV. Experiments

A. Experimental setup

1) Computational Considerations

The retraining of a large base model such as a neural network on a large dataset is computationally intensive. To apply the proposed method efficiently to large base models, we propose to employ the technique of transfer learning, where the weights in all layers except the last one are fixed as a feature extractor [13]. To retrain a neural network on large size subsets, only the last layer in the neural network will be modified and adjusted to the training dataset, This technique has been widely used in natural language processing and computer vision domains [14]. This provides suitable scenarios for the proposed DATA2MODEL with a moderate number of model parameters to be learned and efficient retraining of a wealth of models on different subsets. Besides, recent advances in *fast neural network training* (e.g., FFCV[15]) allow one to train a wealth of models on different subsets very efficiently by removing the data loading bottleneck. Our paper does not incorporate such advanced techniques, and we expect that doing so can further improve efficiency.

2) Base Model

To demonstrate that our proposed DATA2MODEL is a model-agnostic approach, three kinds of base models (*i.e.*, Logistic Regression (LR), Support Vector Machine (SVM)), and a neural network (Resnet-18)) are chosen, and DATA2MODEL will learn to predict the parameters of these models. For LR and SVM, a regularization term with ℓ_2-norm coefficient $\lambda = 1$ is added to the loss function. We adopt L-BFGS-B [16] as the learning algorithm (*i.e.*, solver) for LR and LIBLINEAR [17] as the solver for SVM with a squared hinge loss. Stochastic Gradient Descent (*i.e.*, SGD with learning rate as 0.001, and momentum as 0.9) is adopted as the solver for the training of NN.

3) Dataset

We evaluate the proposed DATA2MODEL on five datasets:

Iris([18]). We use the binary version of Iris dataset with the first two classes of data. The binary version contains 100 samples in total with feature dimension $d = 4$. 67 data points are randomly selected to generate training subsets. The remaining 33 data points are used for SV calculation, and 17 of them are used for dataset addition.

SPAM([19]). SPAM dataset is a collection of spam and non-spam e-mails with feature dimension $d = 215$. 300 data points are randomly selected to generate training subsets. 500 data points are used as the testing set for SV calculation, and 150 of them are used for dataset addition.

HIGGS([20]). HIGGS dataset is produced using Monte Carlo simulations with feature dimension $d = 30$. After preprocessing, we keep 25 features. 300 data points are randomly selected to generate training subsets. 500 data points are used as the testing set for SV calculation, and 150 of them are used for dataset addition.

MNIST([21]). MNIST dataset is a collection of grayscale handwritten digits with size 28×28 and 10 classes. We use a CNN in the preprocessing stage to extract the features and reduce the dimension to 128. 500 data points are randomly selected to generate training subsets. 500 data points are used as the testing set for SV calculation, and 250 of them are used for dataset addition.

CIFAR-10([22]) and Hymenoptera([23]). CIFAR-10 is a collection of of 60,000 3-channel images in 10 classes. Hymenoptera is a small 3-channel dataset used to classify ants and bees, which consists of 245 training images and 153 testing images. These two datasets are used for applying DATA2MODEL to large NNs by transfer learning. In the experiment, we fine-tune the weights of the last layer of a pre-trained Resnet-18 model on CIFAR-10 dataset for the binary classification of hymenoptera images and use DATA2MODEL to estimate the parameters of the new model. In particular, a Resnet-18 model is pre-trained on CIFAR-10. The dimension of the weight in the last layer is 512 (*i.e.*, $d = 512$). 208 images are randomly selected to generate training subsets and the remaining 190 are treated as the testing set, where 100 of them are used for dataset addition.

All the input data x are normalized to rescale the norm $\|x\|$ to be within the range of $[0, 1]$

4) Sampling Distribution

To train the DATA2MODEL model, 15000 subsets are sampled from each dataset following permutation sampling procedures in Algorithm 1 to construct the training sample set Φ.

5) Proposed Network Structure

We adopt a canonical model architecture DeepSets [24] for proposed DNN. DeepSets is designed to be permutation invariant of the input samples, which is fit for set function learning (*i.e.*, mapping a set of samples to a target output).

Dataset	Total Training Data Points	Total Testing Data Points	Dataset Deletion			Dataset Addition		
			Size of Starting Subset	Size of Deleted Subsets	Size of Ending Subset	Size of Starting Subset	Size of Added Subsets	Size of Ending Subset
Iris	67	33	63	[5, 10, 15, ..., 30]	33	67	[1, 2, 3, ..., 17]	84
SPAM, HIGGS	300	500	300	[5, 10, 15, ..., 150]	150	150	[5, 10, 15, ..., 150]	300
MNIST	500	500	500	[5, 10, 15, ..., 250]	250	250	[5, 10, 15, ..., 250]	500
Hymenoptera	208	190	208	[5, 10, 15, ..., 100]	108	108	[5, 10, 15, ..., 100]	208

TABLE I: Setting for Dataset Deletion and Dataset Addition.

6) Evaluation Metrics and Baseline

To evaluate the performance of the proposed DATA2MODEL, baseline comparisons are conducted in two main scenarios: *dataset deletion* and *dataset addition*, where DATA2MODEL is used to predict the model on deleted or added datasets. During the deletion, the size of training subsets is decreased from 100% of the full size to 50% (i.e, for SPAM, from 300 to 150). During the addition, the size of training subsets is increased from 50% of the full size to 100% (i.e, for SPAM, from 150 to 300). Notably, the added samples are unseen during the offline training phase. In both scenarios, 10 subsets are generated randomly with each size. To illustrate the generalizability of the proposed DATA2MODEL to new data points from the same distribution, previously unseen data points are incorporated to form a larger subset in the dataset addition scenario. Table I details the size of the added or deleted subsets for each dataset for dataset deletion and dataset addition.

In these two scenarios, **Influence Function** is selected as one baseline. With the ability to approximate the effect of single data point deletion or addition on model parameters, Influence Function can also be extended to evaluate the subset change [9]. Besides, DeltaGrad [7] can also serve as a baseline for batch deletion or addition. However, it requires the application of SGD to optimize the base model, which cannot provide the optimal solution in a timely manner and cannot solve the primal problem of SVM accurately. Therefore, we do not include DeltaGrad in the experiments. Additionally, we do the ablation study through creating **PARALEARN** as another baseline. PARALEARN has the same neural network structure as DATA2MODEL and also predicts the optimized parameter for a convex learning model. The difference is that it does not include the local and global regularizers.

Three metrics are adopted to evaluate the effectiveness of the proposed method: 1) **the Euclidean distance** between the estimated model parameters and the exact parameters θ^* provided by the solver, *i.e.*, $\left\|\widehat{\theta}-\theta^*\right\|$; 2) **the Normalized-Root-Mean-Squared Error (NRMSE)** of the estimated utility calculated by the estimated model parameters; 3) **the Spearman rank-order correlation** ([25]) between the ground-truth utility and the estimated utility on the testing set. The first metric evaluates the accuracy of parameter estimation. Here, we use the parameters optimized by retraining the model with the solver as the ground-truth. We adopt the Spearman correlation in addition to the NRMSE because most of the applications of subsets utility [26] desire that the utility is ranked in the correct order.

7) Machine configuration

We run experiments with one Intel(R) Xeon(R) Gold 5218 CPU and use one GeForce RTX 2080 Ti for DNN training and inference.

B. Experimental results

1) DATA2MODEL can predict training outcomes

Table II and III summarize results of proposed DATA2MODEL and the baseline in scenarios of dataset deletion and addition, respectively. LR is used as the base model for the first four datasets while NN is used for the last Hymenoptera dataset. First we focus on the results with LR as the base model. In general, DATA2MODEL demonstrates a significant advantage in accurate prediction of parameters if we compare $\left\|\widehat{\theta}-\theta^*\right\|$. **Comparing the results of different datasets, the Euclidean distance grows slightly with the dimension of the input feature dimension, but the estimated utility is unaffected.** It can also be shown the predicted parameters well maintain the utility by checking the NRMSE of utility and the Spearman correlation (*i.e.*, over 95% for most datasets) between the estimated utility and ground-truth. However, it is seen that both the parameter prediction accuracy and the utility estimation accuracy are relatively low on the HIGGS dataset. This might be caused by the low performance of the base model (*i.e.*, LR) on this dataset, which generates similar optimal parameters as training samples in the training phase.

By investigating the performance of two baselines, although PARALEARN outperforms the Influence Function on Iris dataset, the Spearman correlation for other datasets cannot be calculated. This is because it provides constant prediction regardless of the input subset change, resulting in constant utility for all subsets. **This result indicates the proposed KKT and utility loss effectively prevent overfitting.**

We visualize the result on MNIST in Figure 2. The x axis of Figure 2 a) and b) is the ground-truth utility (*i.e.*, testing loss) and the y axis shows the estimated utility. The red line represents the ground-truth utility and each circle represents the estimated utility of one subset, where the circle size is proportional to the subset size. It clearly shows that the error of loss predicted by Influence Function increases with the scale of the change on dataset (either deletion or addition) since the utility estimated by Influence Function rapidly deviates from the red line, whereas DATA2MODEL consistently provides accurate utility prediction.

Figure 2 c) and d) demonstrate the Euclidean distance $\left\|\widehat{\theta}-\theta^*\right\|$ with error bar where the x axis in c) is the size

Dataset	Algorithm	Parameter		Utility			
		$\left\Vert\widehat{\theta}-\theta^*\right\Vert$	Std	NRMSE	Std	Spearman Corr	Std
Iris	**DATA2MODEL**	**9.67E-02**	**1.51E-02**	**4.23%**	**1.62%**	**0.9786**	**0.0327**
	Influence function	8.97E-01	1.39E-02	74.62%	16.98%	0.4071	0.4262
	ParaLearn	1.95E-01	2.49E-02	30.53%	10.74%	0.6821	0.2538
SPAM	**DATA2MODEL**	**3.72E-01**	**1.27E-02**	**6.48%**	**1.19%**	**0.9856**	**0.0040**
	Influence function	1.07E+00	3.40E-03	50.60%	3.70%	0.1841	0.1797
	ParaLearn	2.36E+00	2.15E-02	125.55%	10.62%	N/A	N/A
HIGGS	**DATA2MODEL**	**2.41E-01**	**1.77E-02**	**12.58%**	**3.86%**	**0.7980**	**0.0662**
	Influence function	3.32E-01	5.11E-03	22.68%	6.11%	0.7504	0.1275
	ParaLearn	5.11E-01	2.89E-02	65.30%	17.03%	N/A	N/A
MNIST	**DATA2MODEL**	**9.92E-01**	**9.60E-03**	**2.28%**	**0.42%**	**0.9978**	**0.0010**
	Influence function	1.04E+01	6.65E-01	48.85%	2.81%	0.0278	0.2375
	ParaLearn	4.58E+00	4.96E-03	113.94%	7.11%	N/A	N/A
Hymenoptera	**DATA2MODEL**	**1.18E+00**	1.14E-02	**6.54%**	**0.53%**	**0.99195**	**0.00252**
	Influence function	1.24E+00	**6.24E-03**	56.43%	2.71%	0.03623	0.13195
	ParaLearn	2.04E+00	1.16E-02	73.97%	4.05%	N/A	N/A

TABLE II: A summary of DATA2MODEL results and baseline comparison results in the scenario of dataset deletion with LR (*i.e.*, for Iris, SPAM, HIGGS, and MNIST) and NN (*i.e.*, for Hymenoptera) as the base model. Mean and standard deviation reported over 10 experimental trials. The best result is highlighted in **bold**.

Dataset	Algorithm	Parameter		Utility			
		$\left\Vert\widehat{\theta}-\theta^*\right\Vert$	Std	NRMSE	Std	Spearman Corr	Std
Iris	**DATA2MODEL**	**1.34E-01**	**8.81E-03**	**5.10%**	**0.52%**	**0.9882**	**0.0067**
	Influence function	3.56E-01	6.13E-04	63.07%	3.15%	0.0779	0.1521
	ParaLearn	2.85E-01	4.22E-03	43.91%	2.83%	0.9730	0.0161
SPAM	**DATA2MODEL**	**5.96E-01**	**4.56E-03**	**10.22%**	**0.82%**	**0.9919**	**0.0026**
	Influence function	1.11E+00	2.34E-03	65.44%	2.26%	0.1886	0.1683
	ParaLearn	2.19E+00	2.10E-02	145.73%	4.36%	N/A	N/A
HIGGS	**DATA2MODEL**	**3.26E-01**	**1.56E-02**	**14.11%**	**2.71%**	**0.8054**	**0.0684**
	Influence function	4.58E-01	3.23E-03	20.40%	4.27%	0.9008	0.0381
	ParaLearn	1.92E+00	1.86E-02	838.62%	128.59%	N/A	N/A
MNIST	**DATA2MODEL**	**1.81E+00**	**1.22E-02**	**5.46%**	**0.25%**	**0.9969**	**0.0009**
	Influence function	1.31E+01	4.75E-01	125.26%	4.22%	-0.8431	0.0367
	ParaLearn	4.76E+00	6.01E-03	131.73%	2.17%	N/A	N/A
Hymenoptera	**DATA2MODEL**	**1.94E+0**	1.57E-02	**17.45%**	0.58%	**0.70000**	**0.09458**
	Influence function	7.11E+00	2.81E-02	558.29%	8.77%	0.35000	0.14044
	ParaLearn	2.25E+00	**1.37E-02**	27.13%	**0.24%**	N/A	N/A

TABLE III: A summary of DATA2MODEL results and baseline comparison results in the scenario of dataset addition with LR as the base model. Mean and standard deviation were reported over 10 experimental trials. The best results are highlighted in **bold**.

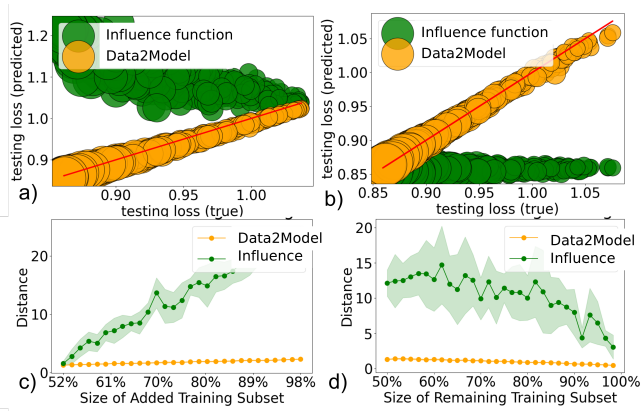

Fig. 2: Results on MNIST dataset with LR as the base model. a) and b): the estimated loss by predicted parameters of each subset in the scenario of dataset deletion and addition; c) and d): the Euclidean distance between predicted parameters and optimal parameters in the same scenario.

of the training subset after addition and the x axis in d) is the size of the remaining training subset after deletion, both of which are formatted as the percentage of the size of full training set (*i.e.*, $n = 300$ for MNIST). From Figure 2 c) and d), it is shown that Influence Function can make an accurate approximation of the parameter change with a small scale of change (*i.e.*, at the beginning of data addition and deletion). However, the distance grows with the scale of change rapidly due to the nature of linear approximation of the Influence Function.

For better readability, we defer the results of two scenarios with SVM as the base model to the Appendix, which demonstrate the similar performance of DATA2MODEL in terms of efficiency and effectiveness.

By checking the performance of applying DATA2MODEL on transfer learning in NN. similar conclusions can be drawn compared to those of LR. Firstly, DATA2MODEL can accurately predict the high-dimensional parameters for the transfer learning of NN. Secondly, the utility of the refitted NN can be accurately estimated by the proposed method, which indicates the effectiveness and generalizability brought

by the proposed regularizers. Thirdly, the parameter prediction error is larger than the results of convex base models (*i.e.*, Table II and III), which might be due to the stochastic training process of NN.

Similar to Figure 2, Figure 3 visualizes the estimated loss and the Euclidean distance between predicted parameters and optimal parameters by Influence Function and DATA2MODEL with NN as the base model. It shows the estimation error of parameters by Influence Function keeps increasing with the change of the subset while the error of the proposed method remains at a lower level after the intersection point. This demonstrates the advantage of the proposed method over linear approximation by Influence Function, and it also quantifies the size of the subset after addition or deletion where the proposed method dominates the performance. In summary, DATA2MODEL demonstrates its superior performance in predicting the the parameters for the training of ML models with subsets of different sizes.

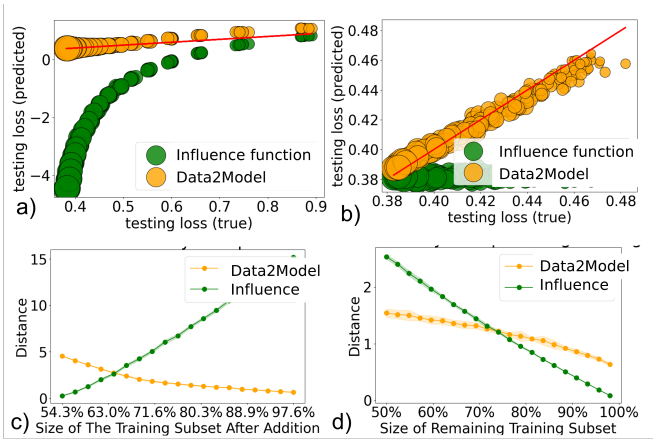

Fig. 3: Results on Hymenoptera images with NN as the base model. a) and b): the estimated loss by predicted parameters of each subset in the scenario of dataset deletion and addition; c) and d): the Euclidean distance between predicted parameters and optimal parameters in the same scenario.

C. Applications

We leverage DATA2MODEL to a variety of ML applications to answer questions mentioned in Section I.

Which point is the most responsible for learning a given parameter in the model?

DATA2MODEL Learns Learning Patterns To demonstrate the effectiveness of neural networks in learning the mappings between training data and the final parameters, we plot the saliency map of model parameter changes when a specific data point is excluded ("turned off") from the training set. In detail, LR is selected as the base model, 300 data points from SPAM and 500 points from MNIST are used as the full training dataset. A LR model with L1 penalty [27] is firstly trained on the full training set to identify the significant parameters. Then, a subset of parameters is randomly selected for visualization. We flip the label of the first 20 samples in each training set to investigate the effect of noisy data on the fitted model, as well as to examine the generalizability of the proposed method.

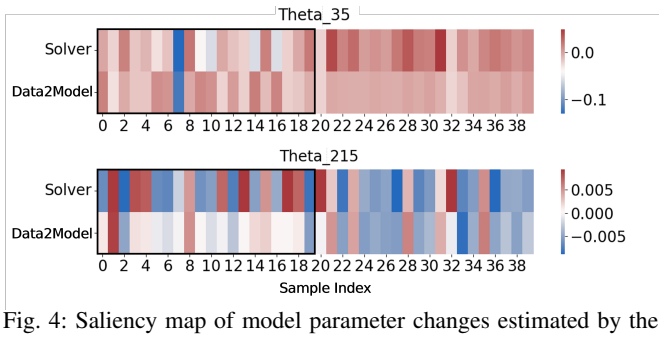

Fig. 4: Saliency map of model parameter changes estimated by the solver and DATA2MODEL with LR as the base model on SPAM dataset. Flipped sample indexes are boxed.

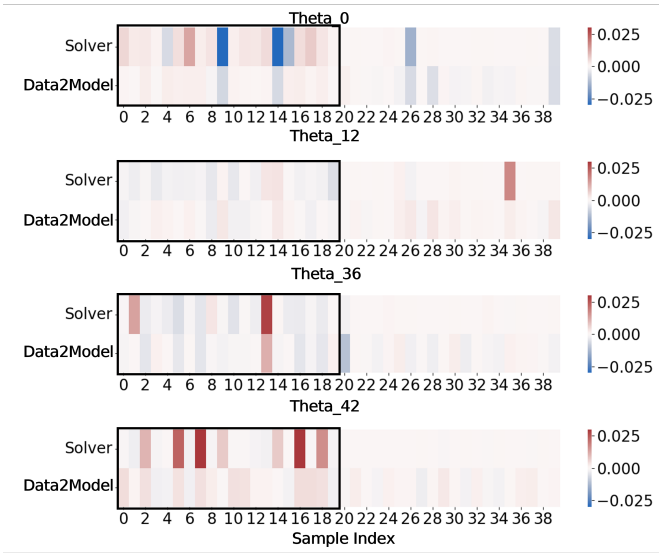

Fig. 5: Saliency map of model parameter changes estimated by the solver and DATA2MODEL with LR as the base model on MNIST dataset. Flipped sample indexes are boxed.

By comparing the color of the first row in Figure 4 and 5, the parameter changes caused by noisy data (*i.e.*, first 20 samples) are more significant than others (last 20 samples). As we can see, DATA2MODEL does learn similar patterns in parameter changes compared with the changes estimated by the solver caused by the exclusion of both noisy and good samples. For example, DATA2MODEL accurately identifies the impact of removing the 9th and 14th sample on θ_0 in MNIST dataset (*i.e.*, Figure 5). This demonstrates the capability of DATA2MODEL in accurately capturing the mapping from the training data to the learned model.

What are the training points with most or least contribution to the model?

Data Valuation. Quantifying the value of each training data point to a learning task is a fundamental problem in ML.

SV is a widely used data value notion nowadays to identify the contribution of each training points [2, 28, 1, 29, 30]. However, the exact SV calculation for a training dataset with n points involves computing the marginal utility of every point in all subsets, which requires 2^n times of utility evaluation by retraining the model. Since DATA2MODEL is capable of efficient utility evaluation by predicting the model parameters on a new training subset, DATA2MODEL can speed up the SV calculation.

To validate the performance of DATA2MODEL on calculating SV, we use permutation sampling [31] to generate the subsets evaluated from the full training set D in sequence. We compare the SV results calculated by DATA2MODEL, and UTLLEARN (*i.e.*, predicted SV) with the SV calculated by the solver as ground truth, and we also record the total time consumed by each method for subset utility evaluation. Here, we create UTLLEARN as a baseline which has the same neural network structure as DATA2MODEL but directly predicts the utility of a subset, and thus, not including KKT loss. We set the number of permutations $T \in \{10, 50, 100, 500, 1000\}$ for Iris, SPAM, and HIGGS. For MNIST, we set $T \in \{10, 50, 100, 200\}$ due to long computation time of the solver caused by high feature dimension. Similarly, we use NRMSE and Spearman correlation as performance metrics to evaluate the predicted SV.

Table IV summarizes the results for Iris, SPAM and HIGGS. It can be observed that DATA2MODEL outperforms UTLLEARN in terms of SV prediction under all scenarios. Compared to UTLLEARN, DATA2MODEL has more network parameters to train due to the higher output dimension d_{param}. However, DATA2MODEL gains its advantage from the KKT regularization which enforces the optimality of predicted parameters, thus, achieving accurate parameter and utility prediction.

Additionally, comparing the results of a varied number of permutations, it is shown that the Spearman correlation tends to increase with number of permutations T. When T is small, the ground-truth SV calculated by permutation sampling is very sensitive to the utility of small subsets. Therefore, the relatively inaccurate prediction on small subsets results in the high NRMSE and low Spearman correlation. With the increase of T, the SV predicted by DATA2MODEL can better represent the true SV with the increasing correlation.

Figure 6 presents the trends of Spearman Correlation of predicted SV by DATA2MODEL, and compares the total computation time of SV calculation with the solver on MNIST and HIGGS. It demonstrates the advantage of DATA2MODEL over the solver in computation time, where time consumed by DATA2MODEL has a slow growth with the increasing number of permutations.

How to select data that benefit model performance?

Data Selection. Despite the rapid growth of big data, the performance of all learning algorithms is upper bounded by the quality of training data [32]. Low-quality training data could be attributed to the contamination of various harmful examples (*i.e.*, noisy samples, mislabeled samples

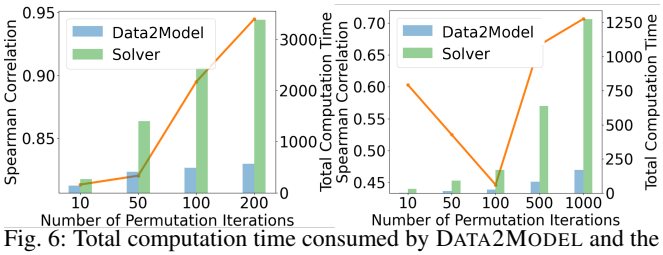

Fig. 6: Total computation time consumed by DATA2MODEL and the solver on training a different number of subsets in SV calculation for MNIST (left) and HIGGS (right).

) or the lack of representativeness of the data distribution. Furthermore, a small but high-quality training dataset can significantly reduce the computation workload of an ML model as well as maintain comparable utility.

In this application, we aim to efficiently identify the quality of training data by estimating the Shapley value of each data point. We estimate the SV of training samples in SPAM and MNIST by DATA2MODEL with LR as the base model (*i.e.*, DATA2MODEL-SV). The label of 10% training samples is flipped in the experiment. Then, the samples are removed from the training set according to their SV from the smallest to the largest. The base model is retrained on the shrunk training set to obtain classification accuracy on the testing set. Shapley value estimated by Permutation Sampling with L-BFGS-B solver (*i.e.*, Perm-SV), and randomly selecting the sample to be removed (*i.e.*, Random) are compared with DATA2MODEL as baselines. To maintain similar computation time by DATA2MODEL-SV and Perm-SV, 1000 and 50 permutations are performed to calculate SV, respectively. Figure 7 shows that after removing the sample number of samples, DATA2MODEL maintains a higher testing accuracy than the baselines while Perm-SV outperforms Random in most cases. This indicates that SV can effectively identify the quality of a data point, and DATA2MODEL-SV can estimate the SV more accurately with a larger number of permutations. Figure 7 also shows that there is an increasing trend of the testing accuracy on both SPAM and MNIST in the initial phase by DATA2MODEL-SV, which indicates that the mislabeled low-quality data are removed effectively to improve the model's learning performance.

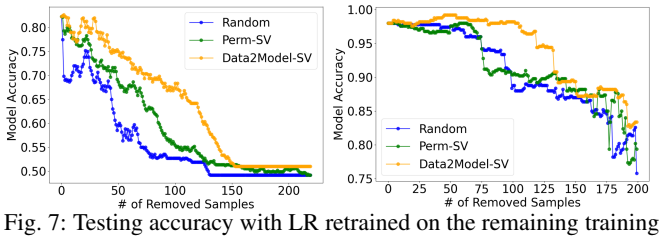

Fig. 7: Testing accuracy with LR retrained on the remaining training samples for SPAM (left) and MNIST dataset (right).

Is a training data point memorized by the model?

Memorization and Influence Score Estimation. One of the most important features of ML algorithms is their capability for generalization. Label memorization score has been proposed to identify and understand the utility of typical, atypical samples and outliers for a model in order

Permutation		50			100			500			1000		
		Shapley value		Time (sec)	Shapley value		Time (sec)	Shapley value		Time (sec)	Shapley value		Time (sec)
Dataset	Algorithm	NRMSE	Spearman		NRMSE	Spearman		NRMSE	Spearman		NRMSE	Spearman	
Iris	**DATA2MODEL**	**8.52%**	**0.9265**	**1.05E+01**	**7.69%**	**0.9463**	**2.22E+01**	**6.82%**	**0.9747**	7.84E+01	**6.22%**	**0.9585**	2.27E+02
	DeepUtility	25.97%	-0.2526	1.08E+01	28.46%	-0.1118	2.21E+01	36.14%	0.0388	**7.87E+01**	42.79%	-0.0480	**2.22E+02**
	solver	–	–	4.05E+01	–	–	8.64E+01	–	–	2.70E+02	–	–	8.57E+02
SPAM	**DATA2MODEL**	**13.46%**	**0.3774**	**1.00E+02**	**18.16%**	**0.3686**	1.23E+02	**15.57%**	**0.7069**	1.43E+02	**18.51%**	**0.7772**	2.86E+02
	DeepUtility	15.21%	-0.2081	1.14E+02	20.76%	-0.2091	**1.26E+02**	18.72%	-0.1187	**1.42E+02**	22.96%	-0.0600	**2.75E+02**
	solver	–	–	3.07E+02	–	–	6.50E+02	–	–	8.96E+02	–	–	4.73E+03
HIGGS	**DATA2MODEL**	**12.15%**	**0.5249**	**1.37E+01**	**15.60%**	**0.4463**	**2.42E+01**	**17.54%**	**0.6655**	**8.30E+01**	**17.45%**	**0.7066**	**1.69E+02**
	DeepUtility	43.12%	-0.2494	1.27E+01	61.82%	-0.3029	2.15E+01	86.85%	-0.5729	1.02E+02	90.52%	-0.6742	1.75E+02
	solver	–	–	9.17E+01	–	–	1.70E+02	–	–	6.39E+02	–	–	1.27E+03

TABLE IV: A summary of DATA2MODEL results and baseline comparison results in SV prediction with LR on IRIS, SPAM and HIGGS with permutation number $T \in \{50, 100, 500, 1000\}$. Mean and standard deviation were reported over 10 experimental trials. The best result is highlighted in **bold**.

to understand the generalization [33]. For a model $f(x;\theta)$ trained on a dataset D by a learning algorithm $\mathcal{A}$, the label memorization score on sample $(x_i, y_i) \in D$ is defined as:

$$\mathrm{mem}(\mathcal{A}, D, i) := \Pr_{f \leftarrow \mathcal{A}(D)}\left[f\left(x_i\right) = y_i\right] - \Pr_{f \leftarrow \mathcal{A}(D \backslash i)}\left[f\left(x_i\right) = y_i\right], \tag{9}$$

where $D \backslash i$ denotes the subset D with (x_i, y_i) removed.

Following [33], we adopt the *subsampling-based estimation algorithm* to estimate the memorization score. Specifically, we sample m subsets of equal size $|S_i| = 0.7|D|$ and the memorization score can be estimated as the following:

$$\widehat{\mathrm{mem}}(\mathcal{A}, D, i) = \frac{1}{|\{j : x_i \in S_j\}|} \sum_{j : x_i \in S_j} I[f\left(\mathcal{A}\left(S_j\right), x_i\right) = y_i] - \frac{1}{|\{j : x_i \notin S_j\}|} \sum_{j : x_i \notin S_j} I[f\left(\mathcal{A}\left(S_j\right), x_i\right) = y_i].$$

However, with a large dataset D, the number of subsets m is substantially large to accurately estimate the label memorization of each sample [33]. This computation constraint can be partially overcome by the DATA2MODEL since it can efficiently estimate the trained model on a subset, thus capable of accelerating the memorization score estimation.

In the experiment, two base models (NN and LR) with two datasets (Hymenoptera and MNIST) are employed for memorization estimation. As mentioned in Section IV-B, the NN is pre-trained on CIFAR-10 and the last layer is trained to adjust to Hymenoptera. In order to maintain comparable computational time for the comparison between DATA2MODEL and the SGD solver, 1000 subsets are randomly generated for each sample with size of 280 for DATA2MODEL, and 50 subsets are generated for the SGD solver to estimate the memorization of 208 training samples in Hymenoptera. In order to demonstrate and compare the estimation accuracy of memorization scores, we depict the effect of removing memorized samples on the test accuracy. Specifically, in the left panel of Figure 8, we show the model test accuracy when data points whose memorization scores are higher than a certain threshold are excluded from the training set. As we can see, memorization scores estimated by DATA2MODEL result in higher test accuracy compared with the baseline estimator, which means that DATA2MODEL can improve memorization score estimation and thus better identify outliers.

When adopting LR as the base model and MNIST as the dataset, memorization scores of the first 200 samples are estimated by L-BFGS-B (*i.e.*, the solver for LR) and DATA2MODEL on 500 testing samples. Since the learning algorithm, in this case, is deterministic, we modify the definition of memorization score as the following:

$$\mathrm{mem}(\mathcal{A}, D, i) \tag{10}$$

$$:= \mathrm{conf}(f_{\mathrm{include}}(x_i), y_i) - \mathrm{conf}(f_{\mathrm{exclude}}(x_i), y_i), \tag{11}$$

where $f_{\mathrm{include}} = \mathcal{A}(D)$ and $f_{\mathrm{exclude}} = \mathcal{A}(D \setminus i)$, and $\mathrm{conf}(f, y)$ denote the confidence score of f on class y.

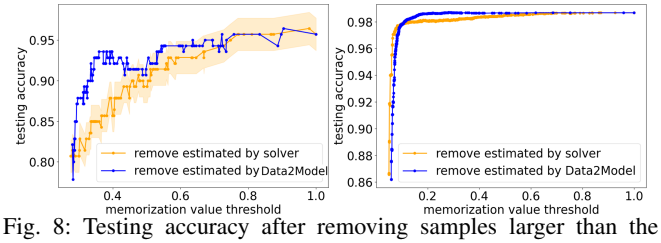

Fig. 8: Testing accuracy after removing samples larger than the memorization value threshold (*i.e.*, ground-truth and estimated by DATA2MODEL) on Hymenoptera with pre-trained NN (left) and on MNIST with LR (right). Mean and standard deviation were reported over 10 experimental trials.

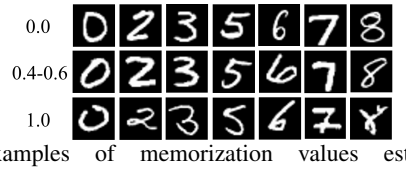

Fig. 9: Examples of memorization values estimated by DATA2MODEL from the first 200 samples of MNIST class 0, 2, 3, 5, 6, 7, 8.

The right panel of Figure 8 demonstrates the effect of removing samples with the memorization values estimated by DATA2MODEL and the solver, which can be treated as the ground-truth memorization value, on the test accuracy of MNIST dataset. As shown by Figure 8, the trend of the removal effect estimated by DATA2MODEL is very close to the ground-truth trend. Compared to the solver, it takes 1.84% of the time to estimate the value by DATA2MODEL, which greatly reduces the computation workload.

Figure 9 visualizes samples with memorization values estimated by DATA2MODEL around 0, 0.5, and 1, respectively, which shows the effectiveness of DATA2MODEL in identifying memorized examples. Intuitively, samples with an estimated memorization value of 0 are relatively normal, whereas those with a value of 1 are atypical (e.g., highly unclear or incorrectly categorized). Therefore, during the training of the base model, samples with high memorization values should be carefully validated to be included in the training dataset.

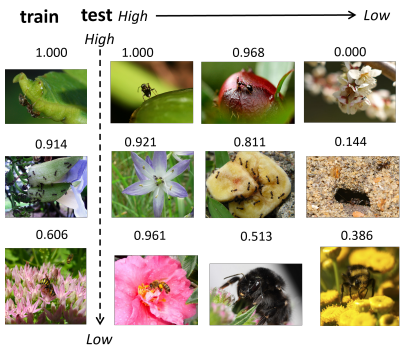

Fig. 10: Examples of selected influence pairs in Hymenoptera. Left column shows the examples with memorization from high to low in the training set. For each training example, examples with high and low estimates are presented in the testing set.

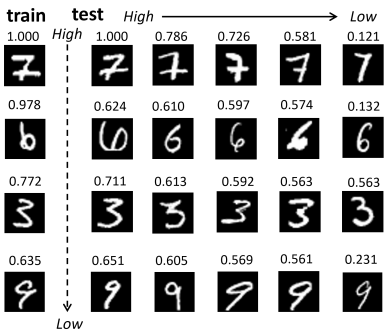

Fig. 11: Examples of selected influence pairs in MNIST. The left column shows examples of memorization score from high to low in the training set. For each training example, examples with high and low estimates are presented in the testing set.

Furthermore, we present pairs of examples with influence estimated by DATA2MODEL from high to low in Hymenoptera and MNIST as shown in Figure 10 and 11. The influence of a training sample (x_i, y_i) on a testing sample $z = (x, y)$ is measured as:

$$\text{infl}(\mathcal{A}, D, i, z) := \Pr_{f \leftarrow \mathcal{A}(D)}[f(x) = y] - \Pr_{f \leftarrow \mathcal{A}(D \setminus i)}[f(x) = y]. \quad (12)$$

As shown in Figure 10 and 11, high-influence pairs corresponds to similar (or even near-duplicated) images.These samples in the test set benefit mostly from the high-memorized training samples due to the long-tail distribution. In contrast, examples in pairs with lower influences are more regular. This validates the effectiveness of the influence memorization value estimated by the proposed DATA2MODEL, which helps to better interpret the influence of samples and the generalization of ML algorithms.

How to produce accurate estimates of uncertainty on model predictions?

Model Calibration. Classification models' confidence calibration performance is crucial in mission-critical tasks [34], and a well-calibrated model should have a confidence (*i.e.*, probability associated with the predicted class label) matching with its ground truth accuracy. Expected Calibration Error (ECE) [35] is the primary metric of model calibration performance, which is calculated by partitioning predictions (with range [0, 1]) into M equally-spaced bins and calculating the weighted average of differences between bins' accuracy and confidence:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left|\text{acc}\left(B_m\right) - \text{conf}\left(B_m\right)\right|,$$

where n is the number of training samples, B_m is the set of indices of samples whose prediction confidence falls into the interval $I_m = (\frac{m-1}{M}, \frac{m}{M})$ for m $\in$ {1,..., M}. Accuracy and confidence are calculated below:

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{1}[\widehat{y}_i = y_i], \quad (13)$$

$$\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \widehat{p}_i, \quad (14)$$

where $\widehat{y}_i$ and y_i are the predicted and true class labels of sample i, respectively, and $\widehat{p}_i$ is the confidence for sample i on label y_i. A model with better calibration performance should have a lower ECE.

Dataset	Regular	DATA2MODEL
Iris	0.0562	**0.0508 ± 0.0002**
SPAM	0.2693	**0.1661 ± 0.0054**
MNIST	0.5492	**0.5227 ± 0.0014**

TABLE V: Expected Calibration Error (ECE). 'Regular' refers to an ensemble of LR models obtained during the training of $\widehat{\mathcal{A}}$; 'DATA2MODEL' refers to an ensemble of a combination of these models and 5000 models generated by the trained $\widehat{\mathcal{A}}$.

One way to improve a model's calibration performance is bagging [36], which generates an ensemble of models on subsampled datasets and aggregates over each prediction to obtain final prediction results. However, training a large number of models can be time-consuming. We show here that DATA2MODEL provides an efficient way to perform bagging, which effectively reduces the ECE. Specifically, we use LR base models as a baseline of the ensemble. Further, we combine these models with models predicted by the trained DNN, $\widehat{\mathcal{A}}$, to get a larger ensemble. To generate models using

$\widehat{\mathcal{A}}$, each time we randomly sample a subset of instances with a ratio 0.6 from the training set with replacement and obtain estimated models predicted by $\widehat{\mathcal{A}}$. As shown in Table V, combining more models generated by $\widehat{\mathcal{A}}$ effectively lower the ECE on Iris, SPAM, and MNIST dataset. We exclude the HIGGS dataset in this application due to the low performance of the base model, LR, on this dataset, which results in a large variance in the prediction.

V. Related Work

Rapid Model Parameter Approximation. Machine unlearning and incremental model maintenance provide post-hoc techniques to estimate model parameters without retraining from scratch. Ginart et al. [37], Nguyen et al. [38], Brophy and Lowd [39] have investigated unlearning strategies on specific types of learning algorithms, whereas Bourtoule et al. [40] provides generally applicable strategies. For simple linear models, Cauwenberghs and Poggio [41], Schelter [42], Wu et al. [43] introduce incremental model maintenance techniques for efficient updating a model for both data deletion and addition. In DNNs, Influence Function [5] is proposed to measure the effect of a manipulated data point by using Taylor expansion to approximate model parameters. DeltaGrad [7] saves optimizer's update steps in order to more accurately approximate the removal of multiple sample points. Golatkar et al. [44] used the approximation of the Fisher Information Matrix for the remaining data sample to measure the update step to modify the model parameters. Although all techniques are capable of efficient model parameter approximation for a change with a small number of samples [45, 46], they are not scalable for situations when a large number of training points are altered.

Learning to Optimize. L2O focuses on learning the optimization algorithm (*i.e.*, optimizer). An early approach was provided in Andrychowicz et al. [47] to use Recurrent Neural Networks (RNN), especially LSTM models, to learn the optimizer. For larger model training, it requires the LSTM model to iterate through more time steps. Unfortunately, that will create a vanishing gradient, an exploding gradient, or memory insufficiency of the LSTM optimization, which in turn will render unstable training of L2O. Due to these barriers, more works [48, 49, 50, 51, 52, 53, 54] focus on overcoming those problems by improving the LSTM structure. Even though, improved LSTMs are able to skip the "bottleneck", those works focus mainly on specific optimizers' family, such as SGD, Adam, or RMSProp. Another branch of L2O technique was developed in parallel. Li and Malik [55] proposed reinforcement learning (RL) technique, in which the RL policy is the update step of the optimizer and the reward is the loss for the optimizer. However, RL methods share similar problems as LSTM models; they are not scalable. Therefore, Almeida et al. [56] proposed to update the optimizer's hyperparameters instead of learning to update parameters, which helps to improve the generalization ability of L2O models. Despite the advantages in fast optimization and potential generalizability, L2O learns to learn and optimize the optimizer, which is still computationally intensive to be applied to repeated model training on a large number of subsets.

Learning Optimization by DNNs. A line of research studies using deep learning to solve convex optimization problems by encoding constraints and dependencies into network structure to find the optimal end-to-end mapping. Agrawal et al. [57] embedded disciplined convex optimization problems as differentiable layers within DNN architectures as a new solver. Amos and Kolter [58] introduced a network architecture to solve differentiable optimization by end-to-end deep learning training. Similar to L2o, repeated training is still computationally demanding with these approaches. Moreover, they also suffer from poor generalization performances, as they are limited to linear programs (LP), and quadratic programs, and are difficult to be applied to other settings. Chen et al. [54] proposed to use DNN to predict the optimal set of active constraints to improve the generalizability, but only for the form of the linear program.

VI. Conclusion

We propose Data2Model, as a framework to analyze the dependence of the trained models on training data via supervised learning. We introduce two novel regularization techniques to prevent overfitting the context of predicting models from training data. We show that the expressiveness of DNNs makes them suitable for approximating the input-output behavior of a learning algorithm. We showcase the applications of Data2Model to build trust in ML.

For future work, it is intriguing to rigorously study the learnability of the mapping from data to the trained model using neural networks. Moreover, while providing promising results, our current design simply examines the ℓ_2 distance between the ground-truth and the predicted model parameters. It is interesting to explore more sophisticated ways to measure parameter distance that accounts for the difference in predictive behaviors of the two models.

References

[1] R. Jia, D. Dao, B. Wang, F. A. Hubis, N. Hynes, N. M. Gürel, B. Li, C. Zhang, D. Song, and C. J. Spanos, "Towards efficient data valuation based on the shapley value," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1167–1176.

[2] A. Ghorbani and J. Zou, "Data shapley: Equitable valuation of data for machine learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2242–2251.

[3] C. Zhang, D. Ippolito, K. Lee, M. Jagielski, F. Tramèr, and N. Carlini, "Counterfactual memorization in neural language models," *arXiv preprint arXiv:2112.12938*, 2021.

[4] B. Efron, "Bootstrap methods: another look at the jackknife," in *Breakthroughs in statistics*. Springer, 1992, pp. 569–593.

[5] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1885–1894.

[6] A. Schioppa, P. Zablotskaia, D. Vilar, and A. Sokolov, "Scaling up influence functions," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8179–8186.

[7] Y. Wu, E. Dobriban, and S. Davidson, "Deltagrad: Rapid retraining of machine learning models," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 355–10 366.

[8] G. Pruthi, F. Liu, S. Kale, and M. Sundararajan, "Estimating training data influence by tracing gradient descent," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19 920–19 930, 2020.

[9] P. W. W. Koh, K.-S. Ang, H. Teo, and P. S. Liang, "On the accuracy of influence functions for measuring group effects," *Advances in neural information processing systems*, vol. 32, 2019.

[10] A. Ilyas, S. M. Park, L. Engstrom, G. Leclerc, and A. Madry, "Datamodels: Predicting predictions from training data," *arXiv preprint arXiv:2202.00622*, 2022.

[11] D. P. Bertsekas, "Nonlinear programming," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.

[12] D. Yarotsky, "Error bounds for approximations with deep relu networks," *Neural Networks*, vol. 94, pp. 103–114, 2017.

[13] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 403–412.

[14] L. Yuan, D. Chen, Y.-L. Chen, N. Codella, X. Dai, J. Gao, H. Hu, X. Huang, B. Li, C. Li *et al.*, "Florence: A new foundation model for computer vision," *arXiv preprint arXiv:2111.11432*, 2021.

[15] G. Leclerc, A. Ilyas, L. Engstrom, S. M. Park, H. Salman, and A. Madry, "ffcv," https://github.com/libffcv/ffcv/, 2022.

[16] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on mathematical software (TOMS)*, vol. 23, no. 4, pp. 550–560, 1997.

[17] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *the Journal of machine Learning research*, vol. 9, pp. 1871–1874, 2008.

[18] R. Fisher and T. Creator, "Iris," UCI Machine Learning Repository, 1988.

[19] M. Hopkins, E. Reeber, G. Forman, and J. Suermondt, "Spambase," UCI Machine Learning Repository, 1999.

[20] P. Baldi, P. Sadowski, and D. Whiteson, "Searching for exotic particles in high-energy physics with deep learning," *Nature communications*, vol. 5, no. 1, pp. 1–9, 2014.

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

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

[23] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[24] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola, "Deep sets," *arXiv preprint arXiv:1703.06114*, 2017.

[25] L. Myers and M. J. Sirois, *Spearman Correlation Coefficients, Differences between*. American Cancer Society, 2006.

[26] A. Akhbardeh, H. Sagreiya, A. El Kaffas, J. K. Willmann, and D. L. Rubin, "A multi-model framework to estimate perfusion parameters using contrast-enhanced ultrasound imaging," *Medical physics*, vol. 46, no. 2, pp. 590–600, 2019.

[27] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[28] R. Jia, D. Dao, B. Wang, F. A. Hubis, N. M. Gurel, B. Li, C. Zhang, C. J. Spanos, and D. Song, "Efficient task-specific data valuation for nearest neighbor algorithms," *arXiv preprint arXiv:1908.08619*, 2019.

[29] T. Wang, J. Rausch, C. Zhang, R. Jia, and D. Song, "A principled approach to data valuation for federated learning," in *Federated Learning*. Springer, 2020, pp. 153–167.

[30] R. Jia, F. Wu, X. Sun, J. Xu, D. Dao, B. Kailkhura, C. Zhang, B. Li, and D. Song, "Scalability vs. utility: Do we have to sacrifice one for the other in data importance quantification?" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8239–8247.

[31] S. Maleki, "Addressing the computational issues of the shapley value with applications in the smart grid," Ph.D. dissertation, University of Southampton, 2015.

[32] A. Jain, H. Patel, L. Nagalapatti, N. Gupta, S. Mehta, S. Guttula, S. Mujumdar, S. Afzal, R. Sharma Mittal, and V. Munigala, "Overview and importance of data quality for machine learning tasks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 3561–3562.

[33] V. Feldman and C. Zhang, "What neural networks memorize and why: Discovering the long tail via influence estimation," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2881–2891, 2020.

[34] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On

calibration of modern neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1321–1330.

[35] M. P. Naeini, G. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[36] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[37] A. Ginart, M. Y. Guan, G. Valiant, and J. Zou, "Making ai forget you: Data deletion in machine learning," *arXiv preprint arXiv:1907.05012*, 2019.

[38] Q. P. Nguyen, B. K. H. Low, and P. Jaillet, "Variational bayesian unlearning," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[39] J. Brophy and D. Lowd, "Machine unlearning for random forests," in *International Conference on Machine Learning*. PMLR, 2021, pp. 1092–1104.

[40] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot, "Machine unlearning," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 141–159.

[41] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," *Advances in neural information processing systems*, pp. 409–415, 2001.

[42] S. Schelter, "amnesia–towards machine learning models that can forget user data very fast," in *1st International Workshop on Applied AI for Database Systems and Applications (AIDB19)*, 2019.

[43] Y. Wu, V. Tannen, and S. B. Davidson, "Priu: A provenance-based approach for incrementally updating regression models," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 447–462.

[44] A. Golatkar, A. Achille, and S. Soatto, "Eternal sunshine of the spotless net: Selective forgetting in deep networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9304–9312.

[45] S. Basu, P. Pope, and S. Feizi, "Influence functions in deep learning are fragile," *arXiv preprint arXiv:2006.14651*, 2020.

[46] A. Mahadevan and M. Mathioudakis, "Certifiable machine unlearning for linear models," *arXiv preprint arXiv:2106.15093*, 2021.

[47] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," in *Advances in neural information processing systems*, 2016, pp. 3981–3989.

[48] Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, M. Botvinick, and N. Freitas, "Learning to learn without gradient descent by gradient descent," in *International Conference on Machine Learning*. PMLR, 2017, pp. 748–756.

[49] K. Lv, S. Jiang, and J. Li, "Learning gradient descent: Better generalization and longer horizons," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2247–2255.

[50] O. Wichrowska, N. Maheswaranathan, M. W. Hoffman, S. G. Colmenarejo, M. Denil, N. Freitas, and J. Sohl-Dickstein, "Learned optimizers that scale and generalize," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3751–3760.

[51] L. Metz, N. Maheswaranathan, J. Nixon, D. Freeman, and J. Sohl-Dickstein, "Understanding and correcting pathologies in the training of learned optimizers," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4556–4565.

[52] Y. Cao, T. Chen, Z. Wang, and Y. Shen, "Learning to optimize in swarms," *Advances in neural information processing systems*, vol. 32, p. 15044, 2019.

[53] X. Chen, Y. Li, R. Umarov, X. Gao, and L. Song, "Rna secondary structure prediction by learning unrolled algorithms," *arXiv preprint arXiv:2002.05810*, 2020.

[54] X. Chen, H. Dai, Y. Li, X. Gao, and L. Song, "Learning to stop while learning to predict," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1520–1530.

[55] K. Li and J. Malik, "Learning to optimize," *arXiv preprint arXiv:1606.01885*, 2016.

[56] D. Almeida, C. Winter, J. Tang, and W. Zaremba, "A generalizable approach to learning optimizers," *arXiv preprint arXiv:2106.00958*, 2021.

[57] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter, "Differentiable convex optimization layers," *arXiv preprint arXiv:1910.12430*, 2019.

[58] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 136–145.

APPENDIX

A. Proof of Theorem 1

Theorem 1 (Restated). *If for all $z \in [0,1]^d$, the loss function $\ell(\theta; z)$ is α-strongly convex in θ, and $\left\| \frac{\partial}{\partial\theta\partial z_i} \ell(\theta, z) \right\| \leq B_1$ for $i \in [d]$, then for $\theta_k(D) = [\operatorname{argmin}_\theta L(\theta; D)]_k$, where $L(\theta; D) = \frac{1}{n}\sum_{i=1}^n \ell(\theta; (x_i, y_i)) + \frac{\lambda}{2}\|\theta\|_2^2$ and $[\cdot]_k$ means the kth element in the vector, then we have*

$$\left\| \frac{\partial \theta_k}{\partial D} \right\| \leq B_1 \frac{\sqrt{d_{\text{param}}(d+1)}}{\sqrt{n}(\alpha+\lambda)}. \tag{15}$$

Proof. Given a loss function $L(\theta; D) = \frac{1}{n}\sum_{i=1}^n \ell(\theta; (x_i, y_i)) + \frac{\lambda}{2}\|\theta\|_2^2$, the unique local minimum $\widehat{\theta}(D)$ is characterized by the implicit function

$$\nabla L(\widehat{\theta}, D) = \frac{1}{n}\sum_{i=1}^n \nabla_\theta \ell(\widehat{\theta}, (x_i, y_i)) + \lambda\widehat{\theta} = 0. \tag{16}$$

Denote $\sigma(j) = \lceil j/(d+1) \rceil$ to be the index of the data point (and hence the loss function ℓ) jth dimension of D corresponds to. By the implicit function theorem and the chain rule, we have

$$\left(\sum_{i=1}^n \frac{\partial \ell_i}{\partial \theta^T \partial \theta} + n\lambda I \right) \frac{\partial \theta}{\partial D_j} + \frac{\partial \ell_{\sigma(j)}}{\partial\theta\partial D_j} = 0, \tag{17}$$

where $\ell_i = \ell(\cdot; (x_i, y_i))$ and, therefore, we obtain

$$\frac{\partial \theta}{\partial D_j} = -[H^{-1}] \frac{\partial \ell_{\sigma(j)}}{\partial\theta\partial D_j}, \tag{18}$$

where $H = \sum_{i=1}^n \frac{\partial \ell_i}{\partial\theta^T\partial\theta} + n\lambda I$ is the Hessian matrix. We use $[H^{-1}]_k$ to denote the kth row of the Hessian inverse, and, hence,

$$\frac{\partial \theta}{\partial D_j} = \left[\frac{\partial \theta_1}{\partial D_j}, \ldots, \frac{\partial \theta_{d_{\text{param}}}}{\partial D_j} \right] \tag{19}$$

$$= \left[-[H^{-1}]_1 \frac{\partial \ell_{\sigma(j)}}{\partial\theta\partial D_j}, \ldots, -[H^{-1}]_{d_{\text{param}}} \frac{\partial \ell_{\sigma(j)}}{\partial\theta\partial D_j} \right]. \tag{20}$$

Therefore, we have

$$\left\| \frac{\partial \theta}{\partial D_j} \right\|_2^2 = \sum_{k=1}^{d_{\text{param}}} \left([H^{-1}]_k \frac{\partial \ell_{\sigma(j)}}{\partial\theta\partial D_j} \right)^2 \tag{21}$$

$$\leq \sum_{k=1}^{d_{\text{param}}} \left\| [H^{-1}]_k \right\|^2 \left\| \frac{\partial \ell_{\sigma(j)}}{\partial\theta\partial D_j} \right\|^2 \tag{22}$$

$$= \left\| \frac{\partial \ell_{\sigma(j)}}{\partial\theta\partial D_j} \right\|^2 \left\| H^{-1} \right\|_F^2 \tag{23}$$

$$\leq B_1^2 \left\| H^{-1} \right\|_F^2, \tag{24}$$

where the inequality is due to the Cauchy–Schwarz inequality.

The remaining work is to bound $\left\| H^{-1} \right\|_F$. Denote $\lambda_1, \ldots, \lambda_{d_{\text{param}}}$ as the eigenvalues of H. Observe that

$$\left\| H^{-1} \right\|_F = \sqrt{tr((H^{-1})^T H^{-1})} \tag{25}$$

$$= \sqrt{tr(H^{-1}H^{-1})} \tag{26}$$

$$= \sqrt{\sum_{i=1}^{d_{\text{param}}} \frac{1}{\lambda_i^2}}, \tag{27}$$

where the second equality holds due to H being symmetric, hence H^{-1} is also symmetric. Further, the last equality holds, because the eigenvalues for H^{-1} are $\frac{1}{\lambda_1}, \ldots, \frac{1}{\lambda_{d_{\text{param}}}}$. Since each ℓ is α-strongly convex, we know that the minimum eigenvalue of the Hessian matrix H is at least $n\alpha + n\lambda$. Therefore, we obtain

$$\left\| H^{-1} \right\|_F \leq \frac{\sqrt{d_{\text{param}}}}{n\alpha + n\lambda}.$$

Then, it follows that

$$\left\| \frac{\partial \theta}{\partial D_j} \right\|_2 \leq B_1 \frac{\sqrt{d_{\text{param}}}}{n(\alpha+\lambda)}, \tag{28}$$

which implies

$$\left| \frac{\partial \theta_k}{\partial D_j} \right| \leq B_1 \frac{\sqrt{d_{\text{param}}}}{n(\alpha+\lambda)}, \tag{29}$$

and, finally, we have

$$\left| \frac{\partial \theta_k}{\partial D} \right| \leq B_1 \frac{\sqrt{d_{\text{param}}(d+1)}}{\sqrt{n}(\alpha+\lambda)}. \tag{30}$$

□

B. Proof of Theorem 2

Theorem 2. *If for all $z \in [0,1]^d$, the loss function $\ell(\theta; z)$ is β-smooth in θ, and $\left\| \frac{\partial}{\partial\theta\partial z_i} \ell(\theta, z) \right\| \leq B_1$ for $i \in [d]$, then for $\theta_k^{(t)}(D)$ defined by the kth entry of $\theta^{(t)}$, which is iteratively computed by $\theta^{(t)} = \theta^{(t-1)} - \eta\nabla L(\theta^{(t-1)}; D)$ and $\theta^{(0)} = 0$ with a learning rate $\eta > 0$, and $L(\theta; D) = \frac{1}{n}\sum_{i=1}^n \ell(\theta; (x_i, y_i)) + \frac{\lambda}{2}\|\theta\|_2^2$, we have*

$$\left\| \frac{\partial \theta_k^{(t)}}{\partial D} \right\| \leq \left(1 - (1 - \eta\lambda - \eta d_{\text{param}}\beta)^t\right) \frac{B_1\sqrt{(d+1)}}{\sqrt{n}(\lambda + d_{\text{param}}\beta)}.$$

Proof. Consider an iteration $\theta_k^{(t)} = \theta_k^{(t-1)} - \eta\nabla L(\theta^{(t-1)}; D)$.

Taking a derivative with respect to D_j for both sides, we have

$$\frac{\partial \theta_k^{(t)}}{\partial D_j} = \frac{\partial \theta_k^{(t-1)}}{\partial D_j} \tag{31}$$

$$- \eta \left[\lambda \frac{\partial \theta_k^{(t-1)}}{\partial D_j} + \frac{1}{n} \sum_{i=1}^{n} \frac{\partial^2 \ell_i}{\partial \theta_k^2} \frac{\partial \theta_k^{(t-1)}}{\partial D_j} + \frac{1}{n} \frac{\partial \ell_{\sigma(j)}}{\partial \theta_k \partial D_j} \right] \tag{32}$$

$$= \left(1 - \eta\lambda - \frac{\eta}{n} \sum_{i=1}^{n} \frac{\partial^2 \ell_i}{\partial \theta_k^2} \right) \frac{\partial \theta_k^{(t-1)}}{\partial D_j} - \frac{\eta}{n} \frac{\partial \ell_{\sigma(j)}}{\partial \theta_k \partial D_j} \tag{33}$$

$$\leq (1 - \eta\lambda - \eta d_{\text{param}} \beta) \frac{\partial \theta_k^{(t-1)}}{\partial D_j} - \frac{\eta}{n} B_1, \tag{34}$$

where the inequality follows due to $\left| \frac{\partial \ell_{\sigma(j)}}{\partial \theta_k \partial D_j} \right| \leq B_1$ and $\frac{\partial^2 \ell_i}{\partial \theta_k^2} \leq d_{\text{param}} \beta$.

Observe that

$$\frac{\partial \theta_k^{(t)}}{\partial D_j} + \frac{\eta B_1}{n\alpha} \leq (1 - \alpha) \left(\frac{\partial \theta_k^{(t-1)}}{\partial D_j} + \frac{\eta B_1}{n\alpha} \right) \tag{35}$$

$$\leq (1 - \alpha)^t \frac{\eta B_1}{n\alpha}. \tag{36}$$

Therefore, we obtain

$$\left| \frac{\partial \theta_k^{(t)}}{\partial D_j} \right| \leq \left(1 - (1 - \eta\lambda - \eta d_{\text{param}} \beta)^t \right) \frac{B_1}{n(\lambda + d_{\text{param}} \beta)}, \tag{37}$$

which results in

$$\left\| \frac{\partial \theta_k^{(t)}}{\partial D} \right\| \leq \left(1 - (1 - \eta\lambda - \eta d_{\text{param}} \beta)^t \right) \frac{B_1 \sqrt{(d+1)}}{\sqrt{n}(\lambda + d_{\text{param}} \beta)}. \tag{38}$$

□

[12] shows that continuous differentiable functions with a smaller upper bound of the gradient norm can be more efficiently approximated by deep ReLU networks. For completeness, we present the result here.

Theorem 3. *For any Lipschitz continuous function $f : [0,1]^d \to \mathbb{R}$ with range B and $\left\| \frac{\partial f}{\partial x} \right\| \leq L$ for all $x \in [0,1]^d$, there is a ReLU network that is capable of expressing any such function within an arbitrary error ε and has no more than $c \ln(2^{d+1} Bd(d+1)/\varepsilon) + 1$ layers and $d \left(2^{d+1} dL/\varepsilon + 1 \right)^d (c \ln(2^{d+1} Bd(d+1)/\varepsilon) + 1)$ computational units.*

As we can see, if L is smaller, the upper bound of the network size is smaller. The proof of this theorem is done by simply plugging in constants for Theorem 3.1 in [12].

C. Additional Results

1) SVM As the Base Model

We summarize the results for dataset deletion and dataset addition experiments with Support Vector Machine (SVM) as base models in VII.

The results in Table VI and VII illustrate that the proposed DATA2MODEL achieves similar performance with different base models (*i.e.*, LR and SVM) in terms of the accuracy of parameter prediction and strong correlation between the predicted utility and the actual utility of subsets.

2) Abaltion Study

In the phase of offline training, we approximate the learning algorithm by fine-tuning the last layer of a pretrained model. We further perform ablation study to validate the effectiveness of employing transfer learning in the large base model retraining. A Resnet-18 model is pre-trained on CIFAR-10 dataset and the weight of the last layer is fine-tuned to a small subset (*i.e.*, with 200 samples) randomly selected from CIFAR-10. To compare with fix embedding of feature extractors, we randomly initialize the weight of a Resnet-18 model and only adjust the weight of the last layer. These two approaches are used to retrain the subsets of 200 samples in the permutation sampling to estimated the SV of these samples. We select the samples with lowest and highest SV estimated by these two approaches in Figure 12 and 13.

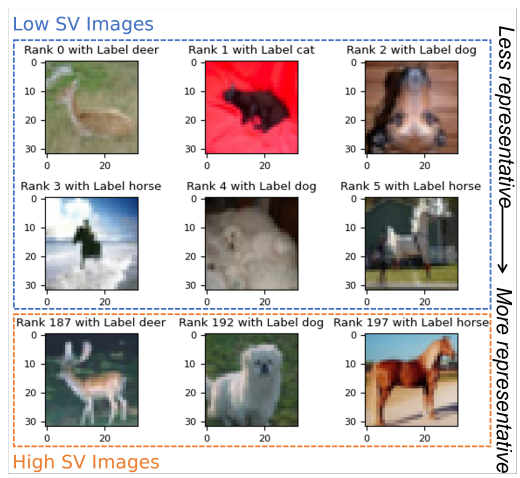

Fig. 12: Images with SV estimated by NN trained with transfer learning on a small subset of CIFAR-10.

As shown in Figure 12, compared to samples in the same class (*i.e.*, deer, dog, and horse), those with lower SV (*i.e.*, in the blue box) are more vague and less representative. By comparing Figure 12 and 13, we find that the same image (the third row and thrid column of Figure 12) is estimated to obtain a high SV by transfer learning, but a low SV by random initialization. However, this image is representative for the class of horse. Moreover, there is no significant difference in representativeness between the images with high and low SV estimated by NN with random initialization as shown in 13, which might be caused by the low learning performance of NN during the subset retraining. This indicates that retrain the large NN with transfer learning can effectively maintain the utility of training samples, which allow us to significantly reduce the model parameters to learn. Therefore, DATA2MODEL can be further utilized to learn parameters of the large DNN models by transfer learning.

Dataset	Algorithm	Parameter		Utility			
		$\left\lVert\widehat{\theta}-\theta^*\right\rVert$	Std	NRMSE	Std	Spearman Corr	Std
Iris	**DATA2MODEL**	**1.16E-01**	**1.53E-02**	**17.25%**	**7.70%**	**0.9214**	**0.0763**
	Influence function	4.63E-01	8.31E-03	54.10%	8.99%	0.0143	0.4427
SPAM	**DATA2MODEL**	**8.74E-01**	**1.58E-02**	**6.85%**	**0.55%**	**0.9826**	**0.0059**
	Influence Function	1.47E+00	7.78E-03	48.96%	3.23%	0.1395	0.2271
HIGGS	**DATA2MODEL**	**5.38E-01**	**2.72E-02**	**7.52%**	**1.16%**	**0.9048**	**0.0442**
	Influence Function	6.12E-01	6.61E-03	16.97%	2.78%	0.7274	0.1377

TABLE VI: A summary of DATA2MODEL results and baseline comparison results in the scenario of dataset deletion with SVM as the base model. The best results are highlighted in **bold**.

Dataset	Algorithm	Parameter		Utility			
		$\left\lVert\widehat{\theta}-\theta^*\right\rVert$	Std	NRMSE	Std	Spearman Corr	Std
Iris	**DATA2MODEL**	1.68E-01	1.15E-02	**40.82%**	**3.01%**	**0.9292**	**0.0379**
	Influence Function	**1.50E-01**	**6.72E-04**	63.29%	3.88%	0.1694	0.2300
SPAM	**DATA2MODEL**	**1.19E+00**	**1.77E-02**	**10.44%**	**1.04%**	**0.9848**	**0.0046**
	Influence Function	1.62E+00	3.27E-03	55.03%	2.01%	0.8080	0.0638
HIGGS	**DATA2MODEL**	**7.23E-01**	**2.07E-02**	20.66%	2.55%	**0.8467**	**0.0378**
	Influence Function	7.30E-01	7.37E-03	**17.34%**	**2.41%**	0.7095	0.0828

TABLE VII: A summary of DATA2MODEL results and baseline comparison results in the scenario of dataset addition with SVM as the base model. The best results are highlighted in **bold**.

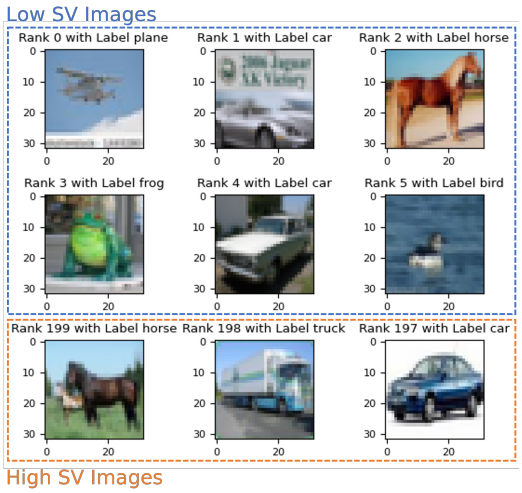

Fig. 13: Images with SV estimated by NN trained without transfer learning on a small subset of CIFAR-10.

D. Past Reviews

Reviewer 1 Questions

1) Summary and Contributions: Briefly summarize the paper and its contributions
   - This paper proposes a deep neural network based solution to the computationally expensive model retraining problem on changing datasets.
2) Strengths: Please describe the strengths of the work according (but not limited) to the following criteria: soundness of the claims (theoretical grounding, empirical evaluation), significance and novelty of the contribution, and relevance to the AISTATS community.
   - The work and the presented ideas are potentially interesting for the AISTATS community.
3) Weaknesses: Please describe the limitations of this work according (but not limited) to the following criteria: soundness of the claims (theoretical grounding, empirical evaluation), significance and novelty of the contribution, and relevance to the AISTATS community.
   - From a practical point of view, when the dimension of θ is large (i.e. overparameterization), I imagine that training the DNN will be very difficult. I'm curious to know how this can be addressed and/or what authors think about this issue of scalability with respect to the dimension of θ. It is not very clear to me what kind of problems and datasets we can expect OPTLEARN to scale to.
     There is a line of work for solving optimization problems using deep neural networks. For instance, the paper "Learning to Solve Network Flow Problems via Neural Decoding" attempts to solve linear problems using neural networks. I think it would be interesting to make a connection between the line of work on solving optimization problems using DNNs and OPTLEARN.
     I find the amount of details on the DNN to be somewhat insufficient, which seems like a critical part of the proposed method. For instance, is it mentioned somewhere in the text or appendix how many layers and neurons the DNN has?
4) Correctness: Are the method and claims correct? Is the empirical methodology correct?
   - The claims seem reasonable to me; but I haven't carefully checked all the claims.
5) Clarity: Is the paper well written? Does it clearly state its contributions, notation and
   - results? I think that the paper is well-written.
6) Relation to prior work: Is it clearly discussed how this work differs from or relates to prior work in the literature?
   - In my opinion the prior work section might be improved (see weaknesses above).
7) Reproducibility: Are there enough details to reproduce the major results of this work?
   - No, the results are not reproducible in the current form of the paper
8) Assumptions and limitations: Does the paper explicitly and clearly state the main assumptions and limitations of the work?
   - Only some assumptions and limitations are discussed, or they are not clear enough
9) Societal impact: Does the paper discuss the societal impact of the work, including the impact that may arise from the misuse of the paper's contribution?
   - No, and it does not seem necessary to include
10) Code release: Do the authors promise to release code for this submission?
    - Yes
11) Score: Please provide an overall score for the submission
    - 5 - Marginally below the acceptance threshold (I lean towards rejection but would not be upset if accepted)
12) Confidence score: Please provide a confidence score for your assessment of this submission
    - 2 - You are willing to defend your assessment, but it is quite likely that you did not understand central parts of the submission or that you are unfamiliar with some pieces of related work. Math/other details were not carefully checked.
13) Ethical Concerns: Does this submission raise potential ethical concerns? These include methods, applications or data that create or reinforce unfair biases and/or that have a primary purpose of harm or injury.
    - No
14) I have read the authors' rebuttal.
    - Agreement accepted
15) How did the authors' rebuttal change your assessment of this submission?
    - I thank the authors for their feedback. Having read the response letter and the other reviews, I'm still on the fence about this paper and would like to keep my score unchanged.

Reviewer 2 Questions

1) Summary and Contributions: Briefly summarize the paper and its contributions
   - The paper discusses how one can improve by re-fitting to ensure KKT conditions when there's more new data added online.
2) Strengths: Please describe the strengths of the work according (but not limited) to the following criteria: soundness of the claims (theoretical grounding, empirical evaluation), significance and novelty of the contribution, and relevance to the AISTATS community.

- The motivation of updating with new data is good

3) Weaknesses: Please describe the limitations of this work according (but not limited) to the following criteria: soundness of the claims (theoretical grounding, empirical evaluation), significance and novelty of the contribution, and relevance to the AISTATS community.
   - The authors do not discuss how their methods are different from classical and simple online learning algorithms, and there is not a solid case to be made to use the proposed method instead of existing online learning algorithms
4) Correctness: Are the method and claims correct? Is the empirical methodology correct?
   - yes
5) Clarity: Is the paper well written? Does it clearly state its contributions, notation and results?
   - yes
6) Relation to prior work: Is it clearly discussed how this work differs from or relates to prior work in the literature?
   - somewhat
7) Reproducibility: Are there enough details to reproduce the major results of this work?
   - Not applicable: Paper of theoretical nature
8) Assumptions and limitations: Does the paper explicitly and clearly state the main assumptions and limitations of the work?
   - Yes, the assumptions and limitations of the work are explicitly and clearly stated
9) Societal impact: Does the paper discuss the societal impact of the work, including the impact that may arise from the misuse of the paper's contribution?
   - No, and it does not seem necessary to include discussion on potential societal impact
10) Code release: Do the authors promise to release code for this submission?
    - Yes
11) Score: Please provide an overall score for the submission
    - 3 - Clear reject (I vote and argue for rejection)
12) Confidence score: Please provide a confidence score for your assessment of this submission
    - 4 - You are confident in your assessment but not absolutely certain.
13) Ethical Concerns: Does this submission raise potential ethical concerns? These include methods, applications or data that create or reinforce unfair biases and/or that have a primary purpose of harm or injury.
    - No

Reviewer 3 Questions

1) Summary and Contributions: Briefly summarize the paper and its contributions
   - The paper propose a method for predicting the argmin of a convex optimization problem in a framework where the input data changes multiples. In this case, retraining the model might be expensive. The algorithm is pretty simple: 1) collect some dataset consisting of a permutation of the input data and the corresponding argmin, 2) predict solution by minimizing the sum of error between the trained solution and the prediction, a term penalizing the violation of KKT condition, and a discrepancy of utility at the trained solution vs the predicted one.
     The authors also propose an error bound on the computational complexity for efficiently learning convex optimizer. The numerical experiments demonstrate some convincing speed up wrt previous method (eg based on estimating influence function) and a naive retraining from scratch.
2) Strengths: Please describe the strengths of the work according (but not limited) to the following criteria: soundness of the claims (theoretical grounding, empirical evaluation), significance and novelty of the contribution, and relevance to the AISTATS community.
   - The numerical experiments show some nice advantage of the proposed method.
     The topic seems relevant to the AISTATS community. However, the authors might showcase a practical example where this multiple data change occurs and that requires a refitting.
3) Weaknesses: Please describe the limitations of this work according (but not limited) to the following criteria: soundness of the claims (theoretical grounding, empirical evaluation), significance and novelty of the contribution, and relevance to the AISTATS community.
   - The computational unit complexity is exponential. To efficiently approximate the solution with deep ReLU, the norm of the gradient must be extremely small even for small dimension d. The bound is more or less vacuous. That's probably why the authors carefully hide the result in the appendix. Theorem 1 and Theorem 2 must be discussed wrt to Theorem 3 (appendix) is the main paper.
     The dimension of the features does not exceed d=300. The number of training sample is quite small too. In these regime, the direct choice of a DNN without discussing simpler "shallow" learning model is questionable. Specially with such unpractical complexity bound + an additional non-convexity in the online phase.
4) Correctness: Are the method and claims correct? Is the empirical methodology correct?
   - The maths in the paper seems legit.
5) Clarity: Is the paper well written? Does it clearly state its contributions, notation and results?

- Fair enough. However, the authors might want to consolidate de notation.
  The authors might want to specifies explicitly some utility functions U for better clarity.
  Why the map $\widehat{\theta}_k$ is differentiable in the first place? It seems that the assumption that the double derivative of l at every point (theta, z) is bounded by B1 implies smoothness of the loss. The authors might also want to specifies the constant B1 and B2 for the mentioned example. For example the logistic loss is not strongly convex unless restricted in a compact domain.

6) Relation to prior work: Is it clearly discussed how this work differs from or relates to prior work in the literature?
   - A set of relevant related work is discussed.
7) Additional Comments: Add your additional comments, feedback and suggestions for improvement, as well as any further questions for the authors.
   - The objective function in Equation 1) is invariant wrt any permutation of the data. Hence the argmin constant regardless of the permutation. The authors might want to clarify that point, specially there is multiple notation, subscript for the various dataset change and this is very confusing.
     The pseudo code can include the learning procedure for online estimation and explicity state the optimization problem being solve and the parameter that are optimized. For example L_{DNN} is a function of what?
     The notation sigma(j) in the appendix is very confusing too. For example, one might thing that the second term of equation 3 is zero (the derivative is wrt D_j and the loss is at sigma(j)).
8) Reproducibility: Are there enough details to reproduce the major results of this work?
   - Yes, the results are reproducible Assumptions and limitations: Does the paper explicitly and clearly state the main assumptions and limitations of the work? Only some assumptions and limitations are discussed, or they are not clear enough
9) Societal impact: Does the paper discuss the societal impact of the work, including the impact that may arise from the misuse of the paper's contribution?
   - No, and it does not seem necessary to include discussion on potential societal impact
10) Code release: Do the authors promise to release code for this submission?
    - Yes
11) Score: Please provide an overall score for the submission
    - 6 - Marginally above the acceptance threshold (I lean towards acceptance but would not be upset if rejected)
12) Confidence score: Please provide a confidence score for your assessment of this submission
    - 2 - You are willing to defend your assessment, but it is quite likely that you did not understand central parts of the submission or that you are unfamiliar with some pieces of related work. Math/other details were not carefully checked.
13) Ethical Concerns: Does this submission raise potential ethical concerns? These include methods, applications or data that create or reinforce unfair biases and/or that have a primary purpose of harm or injury.
    - No
14) I have read the authors' rebuttal.
    - Agreement accepted
15) How did the authors' rebuttal change your assessment of this submission?
    - The authors' rebuttal did not change my assessment.

Reviewer 4 Questions

1) Summary and Contributions: Briefly summarize the paper and its contributions
   - The paper proposes a framework called OptLearn to learn to estimate the model parameters for convex learning problems by a deep neural network, which aims to solve the efficiency issues when retraining models on different subsets of the same dataset. OptLearn introduces two regularization terms to guarantee optimality and utility. Experimental results show the superiority of OptLearn in accuracy and efficiency.
2) Strengths: Please describe the strengths of the work according (but not limited) to the following criteria: soundness of the claims (theoretical grounding, empirical evaluation), significance and novelty of the contribution, and relevance to the AISTATS community.
   - The paper targets efficient model updating for convex learning problems, and the proposed OptLearn can directly predict the optimized model parameters for any new subset of dataset. The problem studied in this paper is valuable to the community.
   - The authors introduced KKT and utility regularization terms to obey optimality conditions and maintain utility.
   - The authors showed some case studies of OptLearn, including speeding up the Shapley value calculation and the bagging algorithm, which demonstrate the advantages of the proposed method.
3) Weaknesses: Please describe the limitations of this work according (but not limited) to the following criteria: soundness of the claims (theoretical grounding, empirical evaluation), significance and novelty of the contribution, and relevance to the AISTATS community.

- The paper tries to solve the problems of expensive model retraining for different subsets with large changes. However, the experiments are mainly conducted for the base model LR and SVM on some relatively small datasets. Therefore, the questions are whether OptLearn can scale to larger datasets, and whether OptLearn can generalize to more complicated models, such as some DL models, because the more severe problems of efficiency may appear under these cases.
- The presentation of the methodology of the paper can be improved. As the main contribution of the paper, I would suggest the authors to better describe the proposed Algorithm 1. For example, it would be helpful if the authors can provide some explanations about the term permutation and the permutation sampling method. Besides, what is the deep learning model used in the proposed OptLearn framework?
- For the theoretical analysis of the proposed method, could the authors provide the prove of the theorems?

4) Correctness: Are the method and claims correct? Is the empirical methodology correct?
   - Appear to be. It would be better to see more results on larger datasets.
5) Clarity: Is the paper well written? Does it clearly state its contributions, notation and results?
   - Presentation can be improved, cf. Weak Point (2)
6) Relation to prior work: Is it clearly discussed how this work differs from or relates to prior work in the literature?
   - Yes.
7) Reproducibility: Are there enough details to reproduce the major results of this work?
   - Yes, the results are reproducible
8) Assumptions and limitations: Does the paper explicitly and clearly state the main assumptions and limitations of the work?
   - Only some assumptions and limitations are discussed, or they are not clear enough
9) Societal impact: Does the paper discuss the societal impact of the work, including the impact that may arise from the misuse of the paper's contribution?
   - Yes, the paper properly discusses the potential societal impact of the work
10) Code release: Do the authors promise to release code for this submission?
    - No
11) Score: Please provide an overall score for the submission
    - 6 - Marginally above the acceptance threshold (I lean towards acceptance but would not be upset if rejected)
    - Confidence score: Please provide a confidence score for your assessment of this submission
    - 3 - You are fairly confident in your assessment.
12) Ethical Concerns: Does this submission raise potential ethical concerns? These include methods, applications or data that create or reinforce unfair biases and/or that have a primary purpose of harm or injury.
    - No
13) I have read the authors' rebuttal.
    - Agreement accepted
14) How did the authors' rebuttal change your assessment of this submission?
    - I have read the authors' rebuttal and other reviews, and kept my score unchanged.

Reviewer 5 Questions

1) Summary and Contributions: Briefly summarize the paper and its contributions
   - This paper proposes to learn a parametrized model's parameters via a deep neural network, motivated by the problem of refitting on subsets with large changes.
     The main contributions are as follows:
     a) A model-agnostic method, OptLearn, to takes as input a dataset and estimates optimized model parameters for that dataset. OptLearn leverages two regularization terms, a KKT regularization and a utility regularization
     b) A theoretical analysis that the derivative of the optimal parameters with respect to the dataset is bounded under certain regularity conditions, which in turn implies that DNN's can approximate a function mapping the data to the model parameters
     c) Empirical results highlighting the performance of OptLearn on real-world datasets
2) Strengths: Please describe the strengths of the work according (but not limited) to the following criteria: soundness of the claims (theoretical grounding, empirical evaluation), significance and novelty of the contribution, and relevance to the AISTATS community.
   - The paper is very clearly written, and Figure 1 gives a very good high-level overview of the paper. All notation is clearly defined, all assumptions are clearly stated, and the organization and flow of the paper is very easy to follow. I very much enjoyed reading this work!
   - As models get larger and larger, the idea of approximating model parameters by DNN seems to be a novel and strong idea, and the potential of future developments in this direction is exciting
   - Theorems 1 and 2 highlight the bounded gradient property of the proposed algorithm in order to show the approximatability of the model parameter learning function in two different scenarios.

Theorem 2 also highlights a practical application, gradient descent.
- T he loss function is carefully constructed, theoretically grounded, and seems very reasonable (KKT loss + utility loss)
- Experiments are well designed, thorough and show the effectiveness of OptLearn

3) Weaknesses: Please describe the limitations of this work according (but not limited) to the following criteria: soundness of the claims (theoretical grounding, empirical evaluation), significance and novelty of the contribution, and relevance to the AISTATS community.
   a) Theoretical
      - While there is a theoretical guarantee on bounded gradient norms, this guarantee only suggests approximatability of the learned function with a sufficiently large ReLU network. There are no learnability (finite-sample) guarantees or guarantees on the optimality of the model-predicted weights.
      - The theoretical parts of this work do not directly address the OptLearn algorithm itself, but rather the nature of the proposed framework.
      - I'm a little skeptical of the realistic nature of the theoretical results, though it's possible I'm missing something here. In the theoretical section (Section 4), the parameter vector theta is being separated into components θ_1, θ_2, and so on, which share the same input but no computational units. Does that mean you're approximating each parameter independently with a neural network? If so, this seems like it wouldn't scale well when the number of parameters becomes large.
   b) Experimental
      - The experimental results are mostly on datasets of relatively small size, and it's unclear whether the proposed method scales well to larger models and larger datasets.
      - One of the main novelties of the paper seems to be the construction of the loss function. I'd like to see more experiments (eg, ablation studies) justifying the choice of loss function as the sum of direct training loss, KKT loss, and utility loss. In addition, I'm wondering why $L(\theta, D_S)$ is not a part of the loss function?
      - In the experiments, why is the direct loss (L) from using the solver's parameters and the optimal parameters not being compared here, but rather the utility? Ultimately, what matters is that you're estimating parameters that achieve a good loss (small changes in parameters can lead to large changes in model behavior).
      - While the focus of the paper is on convex problems, given the success of convex optimization methods on nonconvex problems, I would also be interested in seeing the performance of OptLearn on nonconvex problems (eg, estimating neural network parameters).
   c) Minor comments
      - 3.1: d_θ might be clearer than d_{param} notation-wise
      - I'd appreciate seeing Figure 1 earlier in the text (or at least on the same page it's mentioned)
      - Theorems 1 and 2: it would be nice if in the main text, you clarified what the L functions are (as you did in the appendix)
      - Figure 2, c) and d) would better be visualized as a log plot
4) Correctness: Are the method and claims correct? Is the empirical methodology correct?
   - Yes
5) Clarity: Is the paper well written? Does it clearly state its contributions, notation and results?
   - Yes
6) Relation to prior work: Is it clearly discussed how this work differs from or relates to prior work in the literature?
   - Yes. To my understanding, this work focuses on the setting where a large number of training points are altered, whereas previous work is focused on a smaller number of samples. It is unclear to me whether this is the first work using a DNN to learn model parameters (in the L2O section, LSTM and RL methods to learn model parameters are discussed).
7) Reproducibility: Are there enough details to reproduce the major results of this work?
   - Yes, the results are reproducible
8) Assumptions and limitations: Does the paper explicitly and clearly state the main assumptions and limitations of the work?
   - Yes, the assumptions and limitations of the work are explicitly and clearly stated
9) Societal impact: Does the paper discuss the societal impact of the work, including the impact that may arise from the misuse of the paper's contribution?
   - No, and it does not seem necessary to include discussion on potential societal impact
10) Code release: Do the authors promise to release code for this submission?
    - Yes
11) Score: Please provide an overall score for the submission
    - 5 - Marginally below the acceptance threshold (I lean towards rejection but would not be upset if accepted)

12) Confidence score: Please provide a confidence score for your assessment of this submission

- 3 - You are fairly confident in your assessment.

13) Ethical Concerns: Does this submission raise potential ethical concerns? These include methods, applications or data that create or reinforce unfair biases and/or that have a primary purpose of harm or injury.

- No

14) I have read the authors' rebuttal.

- Agreement accepted

15) How did the authors' rebuttal change your assessment of this submission?

- The author's rebuttal provides ideas that can be used to address my concerns, but does not address my concerns directly. For example, they say that finite-sample guarantees can be achieved via Bartlett, et al, but don't include this result in their work. To me the theoretical results in this work don't seem super strong to me, and the loss function is still a bit unmotivated, so I still stand by my original score.

Below is the description of how we respond to the raised questions/concerns.

[R1, R2, R4, R5–General concern about scalability]: It's unclear about the scalability of OptLearn with respect to the dimension of θ (i.e., larger model), to larger datasets, or to DL models. We have addressed comments regarding scalability by transfer learning for large base models (*i.e.*, including NNs). The proposed technique is generalizable to large base models given the prevalence of using pre-trained models in both NLP and CV domain [1][2]. Oftentimes, pre-trained models are fine-tuned using small datasets and only a small part of parameters are fine-tuned. OptLearn is well-suited for these scenarios.

[R1, R4–General question about DNN details]: We apologize for the lack of details. We have describe the model architecture in the main paper and explain the choice in detail. We adopted a canonical model architecture, DeepSet [Zaheer, et.al, 2017], as the DNN model in OptLearn. A DeepSet model is a *set* function $f(S) = \rho(\sum_{x \in S} \phi(x))$ where both ρ and ϕ are neural networks. In the experiment, both ρ and ϕ netwroks have 3 fully-connected layers with 128 neurons in each layer.

[R1-Question]: The connection between the line of work on solving optimization problems using DNNs and OptLearn. We have added the following content into the related work in the main paper. *A line of research studies using deep learning to solve convex optimization problems by encoding constraints and dependencies into network structure to find the optimal end-to-end mapping [3][4]. However, they also suffer from poor generalization performances. [5] proposed to use DNN to predict the optimal set of active constraints to improve the generalizability, but only for the form of linear program (LP).*

[R2–Main Concern]: The authors did not discuss the difference between OptLearn and classical online learning algorithms and there is a solid use case. While we acknowledge that online learning can be applied to the model refitting of altered training data sets, we respectfully disagree with this comment. We'd like to clarify that OptLearn focuses on learning the optimal estimated parameter but online learning algorithms focus on updating the model parameters instead of pursuing the optimality. Indeed, the idea of online learning has been exploited in past work G-Shapley [6]. However, since online learning provides a rough estimation of the change of optimal loss when one data is added or removed, the resulting data value estimation error is very high and the estimated values have poor utility in identifying data quality [7]. Besides, online learning focuses on data points addition or deletion, which leads to larger compounded error when applied to altered (simultaneous addition and deletion) data sets.

[R3–Question]: Why the map $\widehat{\theta}_k$ is differentiable? We made the assumption that $\widehat{\theta}_k$ is differentiable. For convex learning problems that are solvable by gradient-based algorithms, $\widehat{\theta}_k$ is also usually differentiable with respect to D through implicit function theorem. This assumption is also

required by influence function technique [8].

[R5–Concern about theory]: There are no learnability (finite-sample) guarantees or guarantees on the optimality of the model-predicted weights. We adopted DeepSet as the DNN model in OPTLEARN, which belongs to a canonical DNN model architecture. Therefore, the comprehensive study on the learnability on DNNs can be extended and applied to OPTLEARN such as [9].

[R5–Question]: Does that mean you're approximating each parameter independently with a neural network? Yes, we followed the proof technique in [10] where each parameter is assumed to be approximated independently by a neural network. For the output with dimension d_{param}, the network complexity to provide the same error bound will be guaranteed by $O(d_{param}n)$ where $O(n)$ is the size for the single output theoretically. However, in practice, techniques like weight sharing can be applied to reduce the number of parameters to be trained in the DNN.

[R5–Concern about experiments]: What's the performance of OPTLEARN on nonconvex problems (eg, estimating NN parameters). More studies justifying the choice of loss function. We did the ablation study with results summarized in Table 2 and 3 in the supplemental material where we created PARALEARN without the KKT and utility egularization. The results validate the generalization capability enhanced by the proposed regularizers. We have also conducted experiments on nonconvex problems (*i.e.*, Resnet-18) with the proposed method by transfer learning in the main paper.

[R5–Question]: Why why $L(\theta, D_S)$ is not a part of the loss function? Why is the direct loss (L) from using the solver's parameters and the optimal parameters not being compared here, but rather the utility? In the experiment, we use the loss as the utility function (i.e., $\mathcal{U}(D_S; \widehat{\theta}) = L(\widehat{\theta}, D_S)$). Other utility such as the classification accuracy on the data set can also be adopted as the utility function. We have modified the related part in the main paper.

Learning to Refit for Convex Learning Problems

Anonymous Author
Anonymous Institution

Abstract

Machine learning (ML) models need to be frequently retrained on changing datasets in a wide variety of application scenarios, including data valuation and uncertainty quantification. To efficiently retrain the model, linear approximation methods such as influence function have been proposed to estimate the impact of data changes on model parameters. However, these methods become inaccurate for large dataset changes. In this work, we focus on convex learning problems and propose a general framework to learn to estimate optimized model parameters for different training sets using neural networks. We propose to enforce the predicted model parameters to obey optimality conditions and maintain utility through regularization techniques, which significantly improve generalization. Moreover, we rigorously characterize the expressive power of neural networks to approximate the optimizer of convex problems. Empirical results demonstrate the advantage of the proposed method in accurate and efficient model parameter estimation compared with the state-of-the-art.

1 Introduction

In many essential applications, machine learning (ML) models need to be trained repeatedly on different subsets of the same dataset. For example, Shapley Value (SV) [Jia et al., 2019a,b,c, Wang et al., 2020, Ghorbani and Zou, 2019] has been widely employed to fairly value the contribution of each training data point. The number of models to be retrained on subsets grows exponentially in the number of points for exact SV calculation, which motivates the extensive study on estimating SV efficiently. However, consider the important application of SV such as guiding the allocation of monetary rewards to data providers in a collaborative learning process, a small estimation error of SV may cause huge unfairness, mapping to monetary value with a large scale. Another example is tracking responsibility of data providers for model performance by SV, which is also very sensitive to the SV estimation error [Ghorbani and Zou, 2019]. Given the target upper bound of estimation error ϵ in L_2-norm, existing estimation methods require training on $O(N \log N/\varepsilon^2)$ subsets, which easily becomes impractical when ϵ is small [Jia et al., 2019a]. Therefore, the efficient model retraining on a large number of subsets is essential for accurate SV estimation.

Besides SV, retraining models on subsets of data is also used to diagnose the variation of learning results [Giordano et al., 2019a,b] and estimate predictive uncertainty [Lakshminarayanan et al., 2016]. In summary, these applications pose a computational challenge when the number of training data points is large and when the computation complexity of training a model from scratch is high.

To achieve efficient model updating, post-hoc approaches have been proposed to estimate the change of model parameters in response to a dataset change. For instance, influence functions [Koh and Liang, 2017] approximate the model change using a first-order Taylor's approximation of the loss function, and hence can be implemented efficiently to understand the effect of small changes on training data, e.g., removing a single point. However, these approaches become inaccurate when estimating the impact of a large change on the training data. Different from the post-hoc approximation methods, learning to optimize (L2O) attempts to learn or even generate a new optimization method for an optimization problem in a data-driven manner. Despite promising empirical results, which show learned optimization algorithms are often much faster than manually designed algorithms [Andrychowicz et al., 2016, Li and Malik, 2016, Chen et al., 2017, Lv et al., 2017, Wichrowska et al., 2017, Metz et al.,

Preliminary work. Under review by AISTATS 2021. Do not distribute.

2019, Cao et al., 2019, Chen et al., 2020a,b, repeatedly applying these algorithms for training models on different datasets is still computationally expensive.

Therefore, we propose to learn to estimate the model parameters by a deep neural network (DNN) for repeated training of subsets with large changes. In this work, we focus on convex learning problems, and develop a model-agnostic method OptLearn to directly estimate the optimized model parameters for any given input data set. We introduce two regularization terms, namely, KKT regularizer and utility regularizer, to impose the learned models to satisfy optimality conditions and maintain utility, respectively.

We further provide the theoretical analysis to support the effectiveness of the proposed method. Specifically, we explore the sufficient conditions such that the mapping between dataset and the corresponding learned model parameters could be efficiently approximated by DNNs. We conduct extensive experiments, and the results demonstrate the efficiency and effectiveness of the proposed method compared to the state-of-the-art.

2 Related Work

Rapid Model Parameter Approximation. Machine unlearning and incremental model maintenance provide post-hoc techniques to estimate model parameters without retraining from scratch. Ginart et al. [2019], Nguyen et al. [2020], Brophy and Lowd [2021] have investigated unlearning strategies on specific types of learning algorithms, whereas Bourtoule et al. [2021] provides generally applicable strategies. For simple linear models, Cauwenberghs and Poggio [2001], Schelter [2019], Wu et al. [2020a] introduce incremental model maintenance techniques for efficient updating a model for both data deletion and addition. In deep neural networks, influence function [Koh and Liang, 2017] is proposed to measure the effect of a manipulated data point by using Taylor expansion to approximate model parameters. DeltaGrad [Wu et al., 2020b] saves optimizer's update steps in order to more accurately approximate the removal of multiple sample points. Golatkar et al. [2020] use the approximation of the Fisher Information Matrix for the remaining data sample to measure the update step to modify the model parameters. Although all techniques are capable of efficient model parameter approximation for a change with small number of samples [Basu et al., 2020, Mahadevan and Mathioudakis, 2021], they are not scalable for situations when a large number of training points are added, deleted, or altered.

Learning to Optimize. L2O focuses on learning the optimization algorithm (i.e., optimizer). An early approach was provided in Andrychowicz et al. [2016] to use Recurrent Neural Networks (RNN), especially LSTM models, to learn the optimizer. For larger model training, it requires the LSTM model to iterate through more time steps. Unfortunately, that will create a vanishing gradient, an exploding gradient, or memory insuficiency of the LSTM optimization, which in turn will render unstable training of L2O. Due to these barriers, more works [Chen et al., 2017, Lv et al., 2017, Wichrowska et al., 2017, Metz et al., 2019, Cao et al., 2019, Chen et al., 2020a,b] focus on overcoming those problems by improving the LSTM structure. Even though, improved LSTMs are able to skip the "bottleneck", those works focus mainly on specific optimizers' family, such as SGD, Adam, or RMSProp. Another branch of L2O technique was developed in parallel. Li and Malik [2016] propose reinforcement learning (RL) technique, in which the RL policy is the update step of the optimizer and the reward is the loss for the optimizer. However, RL methods share similar problems as LSTM models; they are not scalable. Therefore, Almeida et al. [2021] propose to update the optimizer's hyperparameters instead of learning to update parameters, which helps to improve the generalization ability of L2O models. Despite the advantages in fast optimization and potential generalizability, L2O learns to learn and optimize the optimizer, which is still computationally intensive to be applied to repeated model training on large number of subsets.

3 Approach

3.1 Preliminaries

Given a data point (x, y), where $x \in \mathbb{R}^d$ is the feature vector and $y \in \mathbb{R}$ is the label, a model $f(x;\theta)$ parameterized by $\theta \in \mathbb{R}^{d_{param}}$ maps the input feature x to the model's prediction $\hat{y}$. A loss function $\ell(\theta;(x,y)) = \ell(f(x;\theta), y)$ measures the discrepancy between the prediction $\hat{y} = f(x;\theta)$ and the true value y. Denote $D = \{(x_i, y_i)\}_{i=1}^n$ as the full training data set containing all n training data points and D_S as a subset with n_S data points sampled from D. For an empirical risk minimization (ERM) problem, the parameters are optimized to minimize the training loss of the subset D_S as:

$$\begin{aligned} &\min_{\theta_S \in \mathbb{R}^{d_{param}}} L(\theta_S; D_S) \\ &= \frac{1}{n_S} \sum_{i=1}^{n_S} \ell(f(x_i;\theta_S)) + \frac{\lambda}{2} \|\theta_S\|_2^2 , \end{aligned} \tag{1}$$

where λ is a hyperpameter that controls the degree of penalization on the L_2-norm of the estimated parameters to increase the model's generalizability. It is a common case in ML that the training loss L is convex in θ, and thus (1) becomes a convex minimization

problem. For common convex problems, a solver of learning algorithm is designed to optimize the problem (1) and obtain the optimized parameter $\hat{\theta}_S$ for the training set D_S. In general, we define *parameter function* $\mathcal{A}$ as the mapping from the input training set D_S to the optimized parameter $\hat{\theta}_S$ in terms of certain learning algorithms, i.e., $\hat{\theta}_S = \mathcal{A}(D_S)$.

It is usually computationally expensive to run the solver over thousands of different input datasets. To reduce the computational workload of repeated training, we propose to learn and approximate the parameter function $\mathcal{A}$ with deep learning models. That is, the optimized parameters are predicted through an evaluation of neural networks.

3.2 Algorithm

The proposed OptLearn framework proceeds in two phases: *offline training* and *online estimation*, which are summarized in Figure 1.

Offline Training. The offline training of OptLearn consists of the optimal parameters sampling step and the DNN training step. Denote π as a permutation of all the training data points from the full set and D_i^π is a set of points which precedes the i-th point in π. We adopt permutation sampling to sample the optimal parameters for DNN training. Denote the DNN model as $\mathcal{A}_{DNN}$. The offline training workflow is summarized in Algorithm 1.

Algorithm 1: OptLearn Offline Training

input : Full dataset $D = \{(x_i, y_i)\}_{i=1}^n$, learning algorithm $\mathcal{A}$, the number of permutations T
output: Trained $\mathcal{A}_{DNN}$

```
Φ ← ∅
for t = 1, ..., T do
    π_t ← GenerateUniformRandomPermutation(D)
    for i = 1, ... n do
        θ̂_i^{π_t} = A(D_i^{π_t})
        Φ = Φ ∪ {(D_i^{π_t}; θ̂_i^{π_t})}
    end
end
Trained A_DNN with Φ
return A_DNN
```

Compared with random sampling of subsets and optimal parameters, permutation sampling enables the evaluation of the change of model parameters with slightly varied subsets. Therefore, the training samples collected by permutation can better reflect the effect of individual data points, thus, allowing the DNN to better learn their contributions. The empirical experiment results also showed a slight advantage of permutation sampling than random sampling when we evaluated the performance of the DNN trained by subsets. Note that if the distribution of the subset to be estimated in the testing phase (i.e., test distribution) is known a priori, it is suggested to directly sample the training samples from the test distribution. For instance, in data valuation by SV, the definition of SV requires the distribution to assign uniform probability to all data sizes and uniform probability to all subsets of a given size. Then, in the offline phase, the subsets should be sampled from the distribution above.

With the set of training samples $\Phi = \{(D_1, \hat{\theta}_1), (D_2, \hat{\theta}_2), \dots\}$, the objective of offline training phase is to train an effective parameter model which can predict the optimized parameters accurately for new subsets. For convex problems, Karuch-Kuhn-Tucker (KKT) conditions guarantee the optimality of a solution [Bertsekas, 1997]. Therefore, besides the direct training loss on predicted parameters: $L_{DNN} = \left\| \mathcal{A}_{DNN}(D_S) - \hat{\theta}_S \right\|$, we consider the KKT loss to enforce the optimality of the predicted parameters. The utility loss is also added to maintain the similarity between the utility on the training subset D_S calculated by predicted parameters and the ground-truth utility calculated by $\hat{\theta}_S$.

For an unconstrained convex problem (1), the optimal parameters $\theta_S^* \in \operatorname{argmin}_{\theta \in \mathbb{R}^{d_{param}}} L(\theta; D_S)$ should satisfy the stationary KKT condition as follows:

$$\nabla L(\theta_S^*; D_S) = 0. \tag{2}$$

If the solver can find a near-optimal solution, then the stationary KKT condition is almost satisfied by the optimized parameter $\hat{\theta}_S$. Therefore, to compel the KKT condition, the KKT loss of one training sample is added as: $L_{KKT} = \|\nabla L\left(\mathcal{A}_{DNN}(D_S); D_S\right)\|$. It should be mentioned that for convex problems with hard constraints, we can still enforce the KKT stationary condition on the Lagrangian of the optimization problem.

For the utility loss, denote $\mathcal{U}(D_S; \hat{\theta})$ as the utility function that evaluates the utility of optimized model parameters $\hat{\theta}$ on a dataset D_S. For one training sample, $(\hat{\theta}_S; D_S)$, the utility loss is specified as: $L_U = \left| \mathcal{U}(D_S; \mathcal{A}_{DNN}(D_S)) - \mathcal{U}(D_S; \hat{\theta}_S) \right|$. Thus, the total loss of one training sample for the proposed DNN is:

$$L_{DNN} = \left\| \hat{\theta}_S - \mathcal{A}_{DNN}(D_S) \right\| + L_{KKT} + L_U \tag{3}$$

Online Estimation. After the offline training phase, the trained DNN $\mathcal{A}_{DNN}$ is used for efficient parameters prediction for a new training subset or a batch of subsets through an evaluation of $\mathcal{A}_{DNN}$.

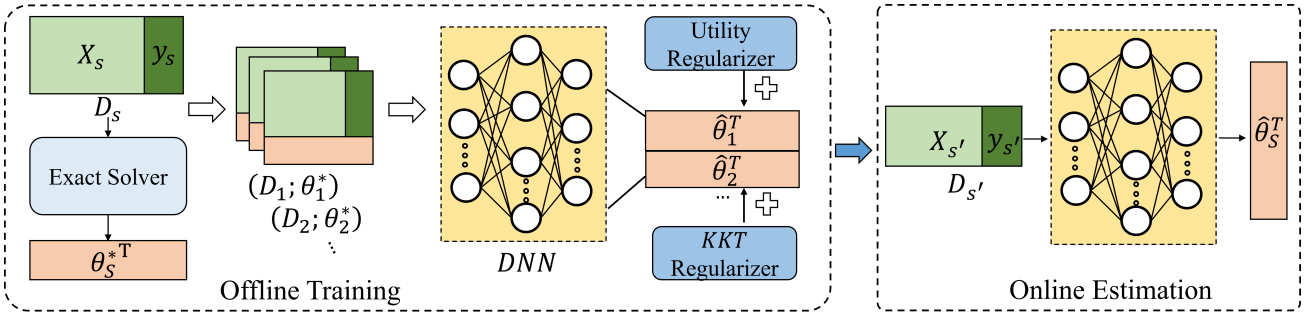

Figure 1: The Offline training phase and online estimate of the proposed OptLearn.

4 Characterization of Efficiently Approximatable Convex Optimizer

In this paper, we try to use a neural network to fit the function $\hat{\theta} = \mathcal{A}(D)$. When the context is clear, we omit the learning algorithm and simply write the function as $\hat{\theta}(D)$. We denote the function associated with kth parameter as $\hat{\theta}_k(D)$, and thus $\hat{\theta}(D) = (\hat{\theta}_1(D), \ldots, \hat{\theta}_{d_{param}}(D))$. Although D is a set of data points, it can be equivalently viewed as a vector concatenation of all (x_i, y_i). We assume that we always normalize data features to $[0, 1]$. Therefore, we have $D \in [0, 1]^{n(d+1)}$. In order to fit $\hat{\theta}(D)$ with a deep neural network, the very first question is *whether $\hat{\theta}(D)$ could be efficiently expressed or approximated by DNNs of certain structures.* Here, the efficiency is measured by the total number of computational units in the DNNs. Despite the strong expressive power of DNNs, not every function could be efficiently approximated by them. Particularly, while the famous universal approximation theorem (UAT) includes many functions, there are requirements about continuity of the functions and compactness of the domain. Recent study [Yarotsky, 2017] shows that functions $f : \mathbb{R}^d \to \mathbb{R}$ that has a smaller upper bound of gradient norm $\|\nabla f\|$ could be more efficiently approximated by DNNs with ReLU activation. As long as $\hat{\theta}_k(D)$ could be efficiently approximated by certain neural network architectures, we could put d_{param} such networks in parallel to approximate $\hat{\theta}(D)$. The parallel networks share the same input but no internal connections and computational units. Therefore, in this section, we aim for understanding *sufficient* conditions for which $\left\|\frac{\partial \hat{\theta}_k}{\partial D}\right\|$ could be upper bounded.

Our first result is under the setting that the learning algorithm $\mathcal{A}$ is able to find an optimal parameter $\theta^* \in \operatorname{argmin}_\theta L(\theta; D)$. There are three major assumptions in our result: (1) the loss function $\ell(\theta)$ is α-strongly convex, (2) $\left\|\frac{\partial}{\partial\theta\partial z_i}\ell(\theta, z)\right\|$ is upper bounded by some constant B_1, and (3) $\|\theta^*\| \le B_2$.

Definition 1 (α-strongly convex)**.** *A differentiable function f is α-strongly convex if*

$$f(y) \ge f(x) + \nabla f(x)^T (y - x) + \frac{\alpha}{2} \|y - x\|^2 , \quad (4)$$

for some $\alpha > 0$ and for all x, y in the domain.

Since $\ell(\theta; (x, y))$ is α-strongly convex, the training loss $L(D) = \frac{1}{n}\sum_{i=1}^n \ell(\theta; (x_i, y_i)) + \frac{\lambda}{2}\|\theta\|_2^2$ is $(\alpha + n\lambda)$-strongly convex. Together with the condition that $\|\theta^*\|$ is finite, it implies L has a unique global minimum $\theta^* = \operatorname{argmin}_\theta L(\theta; D)$. Our main result for strongly convex functions is the following:

Theorem 1. *If for all $z \in [0, 1]^d$, the loss function $\ell(\theta; z)$ is α-strongly convex in θ, and $\left\|\frac{\partial}{\partial\theta\partial z_i}\ell(\theta, z)\right\| \le B_1$, then we have*

$$\left\|\frac{\partial \hat{\theta}_k}{\partial D}\right\| \le B_1 \frac{\sqrt{d_{param}(d+1)n}}{\alpha + n\lambda}. \quad (5)$$

We then analyze the case where the learning algorithm $\mathcal{A}$ is gradient descent. Gradient descent is one of the most commonly used optimization algorithms for convex learning problems. Formally, suppose the parameter θ is initialized by 0 and denoted by $\theta^{(0)}$. For the ith iteration, we update $\theta^{(t)} = \theta^{(t-1)} - \eta\nabla L(\theta^{(t-1)}; D)$ with a learning rate η. Given the maximum step number T, we have $\hat{\theta}(D) = \theta^{(T)}$. The only different assumption for bounding $\left\|\frac{\partial \hat{\theta}_k}{\partial D}\right\|$ in this case is that the loss function $\ell(\theta)$ is β-smooth instead of α-strongly convex.

Definition 2 (β-smoothness)**.** *A differentiable function f is β-smooth if*

$$\|\nabla f(x) - \nabla f(y)\| \le \beta \|x - y\| \quad (6)$$

for some $\beta > 0$, and for all x, y in the domain.

We obtain the following result:

Theorem 2. *If the loss function $\ell(\theta; z)$ is β-smooth and has $\left\|\frac{\partial}{\partial\theta\partial z_i}\ell(\theta, z)\right\| \leq B_1$, then for $\lambda > 0$ we have*

$$\left\|\frac{\partial\hat{\theta}_k}{\partial D}\right\| \leq \frac{(\beta + B_1/n)(1-(1-\lambda\eta)^T)\sqrt{n(d+1)}}{\lambda} \quad (7)$$

and for the case of $\lambda = 0$, we have

$$\left\|\frac{\partial\hat{\theta}_k}{\partial D}\right\| \leq \sqrt{n(d+1)}\eta T(\beta + B_1/n). \quad (8)$$

5 Experiments

5.1 Experimental setup

Dataset. We evaluate the proposed OptLearn on four datasets: Iris (Fisher and Creator [1988]) with input feature dimension $d = 4$, Spam (Hopkins et al. [1999], $d = 215$), HIGGS (Baldi et al. [2014], $d = 250$) and MNIST (LeCun et al. [1998], $d = 128$). Using the binary version of Iris, we have three datasets (Iris, Spam, HIGGS) with two classes and one dataset (MNIST) with multiple classes. 300 training data points and 500 testing data points are randomly selected from Spam, HIGGS, and MNIST, and 2/3 points of the binary Iris are selected to generate the training sample set Φ, respectively. The preprocessing details can be found in the supplemental material.

Base model. To demonstrate that our proposed OptLearn is a model-agnostic approach, two widely used convex models (i.e., Logistic Regression (LR) and Support Vector Machine (SVM)) are chosen as the base models from which OptLearn will learn to refit. For both models, a regularization term with L_2-norm coefficient $\lambda = 1$ is added to the loss function. We adopted L-BFGS-B [Zhu et al., 1997] as the solver for LR and LIBLINEAR [Fan et al., 2008] as the solver for SVM with a squared hinge loss.

Sampling Distribution. To train the OptLearn model, 15000 subsets are sampled from each dataset following the permutation sampling procedures in Algorithm 1 to construct the training sample set Φ.

Evaluation Metrics and Baseline. To evaluate the performance of the proposed OptLearn, baseline comparisons are conducted in two main scenarios: *dataset deletion* and *dataset addition*. During the deletion, the size of training subsets is decreased from 100% of the full size to 50% (i.e, for MNIST, from 300 to 150). During the addition, the size of training subsets is increased from 50% of the full size to 100% (i.e, for MNIST, from 150 to 300). In both scenarios, 10 subsets are generated randomly with each size. To illustrate the generalizability of the proposed OptLearn to new data points from the same distribution, previously unseen data points are incorporated to form a larger subset in the dataset addition scenario.

In these two scenarios, influence function is selected as the baseline. With the ability to approximate the effect of single data point deletion or addition on model parameters, influence function can also be extended to evaluate the subset change [Koh et al., 2019]. Besides, DeltaGrad [Wu et al., 2020b] can also serve as a baseline for batch deletion or addition. However, it requires the application of SGD to optimize the base model, which cannot provide the optimal solution in a timely manner and cannot solve the primal problem of SVM accurately. Therefore, we do not include DeltaGrad in the experiments.

Three metrics are adopted to evaluate the effectiveness of the proposed method: 1) the Euclidean distance between the estimated model parameters and the exact parameters θ^* provided by the solver, i.e., $\left\|\hat{\theta} - \theta^*\right\|$; 2) the Normalized-Root-Mean-Squared Error (NRMSE) of the estimated utility calculated by the estimated model parameters; 3) the Spearman rank-order correlation (Myers and Sirois [2006]) between the ground-truth utility and the estimated utility on the testing set. The first metric evaluates the accuracy of parameter estimation. Here, we use the parameters optimized by retraining the model with the solver as the ground-truth. We adopt the Spearman correlation in addition to the NRMSE because most of the applications of subsets utility [Akhbardeh et al., 2019] desire that the utility be ranked in the correct order.

To assess the efficiency of OptLearn for the repeated training of the base model on a large number of subsets, we record the computation time required for influence function, OptLearn and the solver to update the model parameters, and calculated the average running time per subset.

Machine configuration. We run experiments with one Intel(R) Xeon(R) Gold 5218 CPU and use one GeForce RTX 2080 Ti for DNN training and inference.

5.2 Experimental results.

Table 1 summarizes results of proposed OptLearn and the baseline in scenarios of dataset deletion and addition with Logistic Regression as the base model.

In general, OptLearn has a significant advantage in accurate prediction of parameters if we compare $\left\|\hat{\theta} - \theta^*\right\|$. Comparing the results of different dataset, the Euclidean distance grows slightly with the dimension of the input feature dimension, but the estimated utility is unaffected. It can also been shown the pre-

		Deletion				Addition			
		Parameter	Utility		Time (sec)	Parameter	Utility		Time (sec)
Dataset	Algorithm	$\left\|\hat{\theta} - \theta^*\right\|$	NRMSE	Spearman		$\left\|\hat{\theta} - \theta^*\right\|$	NRMSE	Spearman	
Iris	**OptLearn**	**9.67E-02**	**4.23%**	**0.9786**	**5.11E-04**	**1.34E-01**	**5.10%**	**0.9882**	**6.83E-04**
	Influence function	8.97E-01	74.62%	0.4071	7.64E-04	3.56E-01	63.07%	0.0779	1.30E-04
	Exact solver	-	-	-	3.90E-03	-	-	-	2.54E-03
Spam	**OptLearn**	**3.72E-01**	**6.48%**	**0.9856**	**1.10E-03**	**5.96E-01**	**10.22%**	**0.9919**	**1.04E-03**
	Influence function	1.07E+00	50.60%	0.1841	3.92E-02	1.11E+00	65.44%	0.1886	2.10E-02
	Exact solver	-	-	-	5.77E-03	-	-	-	6.21E-03
HIGGS	**OptLearn**	**2.41E-01**	**12.58%**	**0.7980**	**5.43E-04**	**7.23E-01**	**20.66%**	**0.8467**	**7.86E-04**
	Influence function	3.32E-01	22.68%	0.7504	1.12E-02	7.30E-01	17.34%	0.7095	6.32E-03
	Exact solver	-	-	-	5.13E-03	-	-	-	5.13E-03
MNIST	**OptLearn**	**9.92E-01**	**2.28%**	**0.9978**	**1.09E-03**	**1.81E+00**	**5.46%**	**0.9969**	**8.89E-04**
	Influence function	1.04E+01	48.85%	0.0278	1.93E-01	1.31E+01	125.26%	-0.8431	1.13E-01
	Exact solver	-	-	-	3.14E-02	-	-	-	3.10E-02

Table 1: A summary of OptLearn results and baseline comparison results in the scenario of dataset deletion and addition with LR as the base model. The best result is highlighted in **bold**. The solver provides the optimal parameter for comparison, so the evaluation of its error is omitted as '-'.

dicted parameters well maintain the utility by checking the NRMSE of utility and the Spearman correlation (i.e., over 95% for most datasets) between the estimated utility and groud-truth. However, it is seen that both the parameter prediction accuracy and the utility estimation accuracy are relative low on the HIGGS dataset. This might be caused by the low performance of the base model (i.e., LR) on this dataset, which generates similar optimal parameters as training samples in the training phase.

We visualize the result on MNIST in Figure 2. The x axis of Figure 2 a) and b) is the ground-truth utility (i.e., testing loss) and the y axis shows the estimated utility. The red line represents the ground-truth utility and each circle represents the estimated utility of one subset, where the circle size is proportional to the subset size. It clearly shows that the error of loss predicted by influence function increases with the scale of the change on dataset (either deletion or addition) since the utility estimated by influence function rapidly deviates from the red line, whereas OptLearn consistently provides accurate utility prediction.

Figure 2 c) and d) demonstrate the Euclidean distance $\left\|\hat{\theta} - \theta^*\right\|$ with error bar where the x axis in c) is the size of the training subset after addition and the x axis in d) is the size of the remaining training subset after deletion, both of which are formatted as the percentage of the size of full training set(i.e., $n = 300$ for MNIST). From Figure 2 c) and d), it is shown that influence function can make accurate approximation of the parameter change with a small scale of change (i.e., at the beginning of data addition and deletion). However, the distance grows with the scale of change rapidly due to nature of linear approximation of the influence function. In summary, OptLearn demonstrates its superior performance in predicting the model parameters

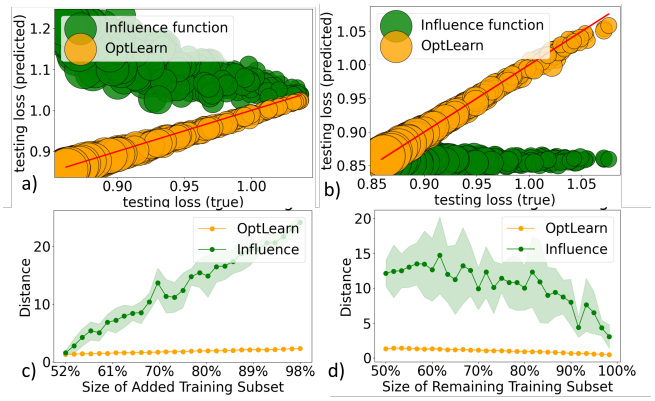

Figure 2: a) and b): the estimated loss by predicted parameters of each subset in the scenario of dataset deletion and addition; c) and d): the Euclidean distance between predicted parameters and optimal parameters in the same scenario.

with subsets of different sizes.

To evaluate the efficiency of OptLearn, we compute the average running time of estimating the parameter for each subset as the time recorded in Table 1. Note that this time does not include the computation time for offline training of OptLearn. Compared to the baseline and the solver, OptLearn demonstrates a significant advantage in efficiency since it only evaluates a deterministic NN to predict the model parameters. Given the offline training time, OptLearn gains its advantage in the application which requires the model retraining on a large number of subsets.

Figure 3 shows the total computation time consumed by OptLearn and the solver on model retraining with different number of subsets for MNIST, where we sample 1000 training samples to train $\mathcal{A}_{DNN}$. Despite the

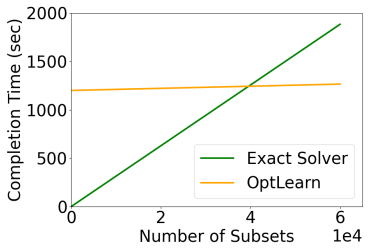

Figure 3: Total computation time consumed by OptLearn and the solver on different number of training subsets.

relative high computation cost during the offline training phase, the computation time of OptLearn grows much slower compared to the solver. It will be more efficient when the number of subsets to train exceeds roughly 40,000, which is a small number for applications such as SV estimation.

Due to space constraint, we defer results of two scenarios with SVM as the base model to Appendix, which demonstrate similar performance of OptLearn in terms of efficiency and effectiveness.

5.3 Applications

We investigate OptLearn performance in applications which require a large number of model retraining.

Data Valuation. Quantifying the value of each training data point to a learning task is a fundamental problem in ML. Shapley value is a widely used data value notion nowadays [Ghorbani and Zou, 2019, Jia et al., 2019a,b, Wang et al., 2020, Jia et al., 2021]. However, the exact SV calculation for a training dataset with n points involves computing the marginal utility of every point in all subsets, which requires 2^n times of utility evaluation by retraining the model. Since OptLearn is capable of efficient utility evaluation by predicting the model parameters on a new training subset, OptLearn can speed up the SV calculation.

To validate the performance of OptLearn on calculating SV, we use permutation sampling [Maleki, 2015] to generate the subsets to be evaluated from the full training set D in sequence. We compare the SV results calculated by OptLearn, and UtlLearn (i.e., predicted SV) with the SV calculated by the solver as ground truth, and we also record the total time consumed by each method for subset utility evaluation. Here, we create UtlLearn as a baseline which has the same neural network structure as OptLearn but directly predicts the utility of a subset, and thus, not including KKT loss. We set the number of permutations $T \in \{10, 50, 100, 500, 1000\}$ for Iris, Spam, and HIGGS. For MNIST, we set $T \in \{10, 50, 100, 200\}$ due to the long computation time of the solver caused by high feature dimension. Similarly, we use the NRMSE and Spearman correlation as performance metrics to evaluate the predicted SV.

Table 2 summarizes the results for Iris, Spam and HIGGS. It can be observed that OptLearn outperforms UtlLearn in terms of SV prediction under all scenarios. Compared to UtlLearn, OptLearn has more network parameters to train due to the higher output dimension d_{param}. However, OptLearn gains its advantage from the KKT regularizer which enforces the optimality of predicted parameters, thus, achieving accurate parameter and utility prediction.

Additionally, comparing the results of varied number of permutations, it is shown that the Spearman correlation tends to increase with number of permutations T. When T is small, the ground-truth SV calculated by permutation sampling is very sensitive to the utility of small subsets. Therefore, the relative inaccurate prediction on small subsets results in the high NRMSE and low Spearman correlation. With the increase of T, the SV predicted by OptLearn can better represent the true SV with the increasing correlation.

Figure 4 presents the trends of Spearman Correlation of predicted SV by OptLearn, and compares the total computation time of SV calculation with the solver on MNIST and HIGGS. It demonstrates the advantage of OptLearn over the solver in computation time, where time consumed by OptLearn has a slow growth with the increasing number of permutations.

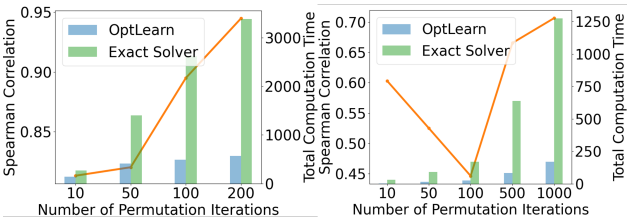

Figure 4: Total computation time consumed by OptLearn and the solver on training different number of subsets in SV calculation for MNIST (left) and HIGGS (right).

Model Calibration. Classification models' confidence calibration performance is crucial in mission-critical tasks [Guo et al., 2017], and a well-calibrated model should have a confidence (i.e., probability associated with the predicted class label) matching with its ground truth accuracy. Expected Calibration Error (ECE) [Naeini et al., 2015] is the primary metric of model calibration performance, which is calculated by partitioning predictions (with range [0, 1]) into M equally-spaced bins and calculate the weighted average

Permutation		50			100			500			1000		
		Shapley Value		Time (sec)	Shapley Value		Time (sec)	Shapley Value		Time (sec)	Shapley Value		Time (sec)
Dataset	Algorithm	NRMSE	Spearman		NRMSE	Spearman		NRMSE	Spearman		NRMSE	Spearman	
Iris	**OptLearn**	**8.52%**	**0.9265**	**1.05E+01**	**7.69%**	**0.9463**	**2.22E+01**	**6.82%**	**0.9747**	7.84E+01	**6.22%**	**0.9585**	2.27E+02
	DeepUtility	25.97%	-0.2526	1.08E+01	28.46%	-0.1118	2.21E+01	36.14%	0.0388	**7.87E+01**	42.79%	-0.0480	**2.22E+02**
	solver	–	–	4.05E+01	–	–	8.64E+01	–	–	2.70E+02	–	–	8.57E+02
Spam	**OptLearn**	**13.46%**	**0.3774**	**1.00E+02**	**18.16%**	**0.3686**	1.23E+02	**15.57%**	**0.7069**	1.43E+02	**18.51%**	**0.7772**	2.86E+02
	DeepUtility	15.21%	-0.2081	1.14E+02	20.76%	-0.2091	**1.26E+02**	18.72%	-0.1187	**1.42E+02**	22.96%	-0.0600	**2.75E+02**
	solver	–	–	3.07E+02	–	–	6.50E+02	–	–	8.96E+02	–	–	4.73E+03
HIGGS	**OptLearn**	**12.15%**	**0.5249**	**1.37E+01**	**15.60%**	**0.4463**	**2.42E+01**	**17.54%**	**0.6655**	**8.30E+01**	**17.45%**	**0.7066**	**1.69E+02**
	DeepUtility	43.12%	-0.2494	1.27E+01	61.82%	-0.3029	2.15E+01	86.85%	-0.5729	1.02E+02	90.52%	-0.6742	1.75E+02
	solver	–	–	9.17E+01	–	–	1.70E+02	–	–	6.39E+02	–	–	1.27E+03

Table 2: A summary of OptLearn results and baseline comparison results in SV prediction with LR on IRIS, SPAM and HIGGS with permutation number $T \in \{50, 100, 500, 1000\}$. The best result is highlighted in **bold**.

of differences between bins' accuracy and confidence:

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left|\mathrm{acc}\left(B_m\right) - \mathrm{conf}\left(B_m\right)\right|,$$

where n is the number of training samples, B_m is the set of indices of samples whose prediction confidence falls into the interval $I_m = (\frac{m-1}{M}, \frac{m}{M})$ for m $\in \{1,\ldots,$ M$\}$. Accuracy and confidence are calucated below:

$$\mathrm{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{1}[\hat{y}_i = y_i], \tag{9}$$

$$\mathrm{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i, \tag{10}$$

where $\hat{y}_i$ and y_i are the predicted and true class labels of sample i, respectively, and $\hat{p}_i$ is the confidence for sample i on label y_i. A model with better calibration performance should have a lower ECE.

Dataset	Regular	**OptLearn**
Iris	0.0562	**0.0508 ± 0.0002**
SPAM	0.2693	**0.1661 ± 0.0054**
HIGGS	0.0626	**0.3082 ± 0.0151**
MNIST	0.5492	**0.5227 ± 0.0014**

Table 3: Expected Calibration Error (ECE). 'Regular' refers to an ensemble of LR models that obtained during the training of $\mathcal{A}_{DNN}$; 'OptLearn' refers to an ensemble of a combination of these models and 5000 models generated by the trained $\mathcal{A}_{DNN}$.

One way to improve a model's calibration performance is bagging [Breiman, 1996], which generates an ensemble of models on bootstrapped datasets and aggregates over each prediction to obtain final prediction results. However, training a large number of models can be time-consuming. We show here that OptLearn provides an efficient way to perform bagging, which effectively reduce the ECE. Specifically, we use LR base models for training the parameter function as a baseline of ensemble. Further, we combine these models with models predicted by the trained DNN, $\mathcal{A}_{DNN}$, to get a larger ensemble. For generating models using $\mathcal{A}_{DNN}$, each time we randomly sample a subset of instances with a ratio 0.6 from the training set with replacement and obtain estimated models predicted by $\mathcal{A}_{DNN}$. As shown in Table 3, combining more models generated by $\mathcal{A}_{DNN}$ effectively lower the ECE on Iris, Spam and MNIST dataset. ECE on HIGGS of OptLearn is higher than Regular, because the parameter function learned on HIGGS has a relative low performance, which may be caused by the low performance of the base model, LR, on this dataset, similar to the results of dataset deletion and addition.

6 Conclusion

Many essential applications in ML require retraining models on a large number of subsets of the same dataset, which has posed significant computational challenges. Existing methods are either incapable of estimating the large change on the dataset accurately or incapable of completing the repeated training efficiently. To address these challenges, we propose OptLearn to learn to refit models for convex learning problems by DNNs. The expressiveness of DNNs enables the effective approximation of the parameter function, and the proposed KKT and utility regularizers further improve the generalizability of OptLearn on subsets with a large change. We evaluate the effectiveness and efficiency of OptLearn on two dataset change scenarios and show that OptLearn outperforms state-of-the-art by providing accurate parameter estimation as well as achieving fast model retraining on a large number of subsets. The advantage of OptLearn is further demonstrated in two ML applications.

Several future work directions can be identified from this work. First, we will further explore the learnability of parameter function by DNNs. We would also like to generalize the proposed method by transferring the learned parameter function from one convex learning model to others, or even to ensemble models [Jin and Deng, 2015], which can further reduce the computation time.

References

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gurel, Bo Li, Ce Zhang, Costas J Spanos, and Dawn Song. Efficient task-specific data valuation for nearest neighbor algorithms. *arXiv preprint arXiv:1908.08619*, 2019a.

Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1167–1176. PMLR, 2019b.

Ruoxi Jia, Xuehui Sun, Jiacen Xu, Ce Zhang, Bo Li, and Dawn Song. An empirical and comparative analysis of data valuation with scalable algorithms. *arXiv preprint arXiv:1911.07128*, 2019c.

Tianhao Wang, Johannes Rausch, Ce Zhang, Ruoxi Jia, and Dawn Song. A principled approach to data valuation for federated learning. In *Federated Learning*, pages 153–167. Springer, 2020.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pages 2242–2251. PMLR, 2019.

Ryan Giordano, William Stephenson, Runjing Liu, Michael Jordan, and Tamara Broderick. A swiss army infinitesimal jackknife. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1139–1147. PMLR, 2019a.

Ryan Giordano, Michael I Jordan, and Tamara Broderick. A higher-order swiss army infinitesimal jackknife. *arXiv preprint arXiv:1907.12116*, 2019b.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR, 2017.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*, pages 3981–3989, 2016.

Ke Li and Jitendra Malik. Learning to optimize. *arXiv preprint arXiv:1606.01885*, 2016.

Yutian Chen, Matthew W Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Timothy P Lillicrap, Matt Botvinick, and Nando Freitas. Learning to learn without gradient descent by gradient descent. In *International Conference on Machine Learning*, pages 748–756. PMLR, 2017.

Kaifeng Lv, Shunhua Jiang, and Jian Li. Learning gradient descent: Better generalization and longer horizons. In *International Conference on Machine Learning*, pages 2247–2255. PMLR, 2017.

Olga Wichrowska, Niru Maheswaranathan, Matthew W Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Nando Freitas, and Jascha Sohl-Dickstein. Learned optimizers that scale and generalize. In *International Conference on Machine Learning*, pages 3751–3760. PMLR, 2017.

Luke Metz, Niru Maheswaranathan, Jeremy Nixon, Daniel Freeman, and Jascha Sohl-Dickstein. Understanding and correcting pathologies in the training of learned optimizers. In *International Conference on Machine Learning*, pages 4556–4565. PMLR, 2019.

Yue Cao, Tianlong Chen, Zhangyang Wang, and Yang Shen. Learning to optimize in swarms. *Advances in neural information processing systems*, 32:15044, 2019.

Xinshi Chen, Yu Li, Ramzan Umarov, Xin Gao, and Le Song. Rna secondary structure prediction by learning unrolled algorithms. *arXiv preprint arXiv:2002.05810*, 2020a.

Xinshi Chen, Hanjun Dai, Yu Li, Xin Gao, and Le Song. Learning to stop while learning to predict. In *International Conference on Machine Learning*, pages 1520–1530. PMLR, 2020b.

Antonio Ginart, Melody Y Guan, Gregory Valiant, and James Zou. Making ai forget you: Data deletion in machine learning. *arXiv preprint arXiv:1907.05012*, 2019.

Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. Variational bayesian unlearning. *Advances in Neural Information Processing Systems*, 33, 2020.

Jonathan Brophy and Daniel Lowd. Machine unlearning for random forests. In *International Conference on Machine Learning*, pages 1092–1104. PMLR, 2021.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159. IEEE, 2021.

Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. *Advances in neural information processing systems*, pages 409–415, 2001.

Sebastian Schelter. amnesia–towards machine learning models that can forget user data very fast. In *1st International Workshop on Applied AI for Database Systems and Applications (AIDB19)*, 2019.

Yinjun Wu, Val Tannen, and Susan B Davidson. Priu: A provenance-based approach for incrementally updating regression models. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 447–462, 2020a.

Yinjun Wu, Edgar Dobriban, and Susan Davidson. Deltagrad: Rapid retraining of machine learning models. In *International Conference on Machine Learning*, pages 10355–10366. PMLR, 2020b.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9304–9312, 2020.

Samyadeep Basu, Philip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. *arXiv preprint arXiv:2006.14651*, 2020.

Ananth Mahadevan and Michael Mathioudakis. Certifiable machine unlearning for linear models. *arXiv preprint arXiv:2106.15093*, 2021.

Diogo Almeida, Clemens Winter, Jie Tang, and Wojciech Zaremba. A generalizable approach to learning optimizers. *arXiv preprint arXiv:2106.00958*, 2021.

Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017.

R.A. Fisher and Test Creator. Iris. UCI Machine Learning Repository, 1988.

Mark Hopkins, Erik Reeber, George Forman, and Jaap Suermondt. Spambase. UCI Machine Learning Repository, 1999.

Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5(1): 1–9, 2014.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86 (11):2278–2324, 1998.

Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, 23(4):550–560, 1997.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *the Journal of machine Learning research*, 9:1871–1874, 2008.

Pang Wei Koh, Kai-Siang Ang, Hubert HK Teo, and Percy Liang. On the accuracy of influence functions for measuring group effects. *arXiv preprint arXiv:1905.13289*, 2019.

Leann Myers and Maria J. Sirois. *Spearman Correlation Coefficients, Differences between*. American Cancer Society, 2006. ISBN 9780471667193. doi: https://doi.org/10.1002/0471667196.ess5050.pub2. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/0471667196.ess5050.pub2.

Alireza Akhbardeh, Hersh Sagreiya, Ahmed El Kaffas, Jürgen K Willmann, and Daniel L Rubin. A multi-model framework to estimate perfusion parameters using contrast-enhanced ultrasound imaging. *Medical physics*, 46(2):590–600, 2019.

Ruoxi Jia, Fan Wu, Xuehui Sun, Jiacen Xu, David Dao, Bhavya Kailkhura, Ce Zhang, Bo Li, and Dawn Song. Scalability vs. utility: Do we have to sacrifice one for the other in data importance quantification? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8239–8247, 2021.

Sasan Maleki. *Addressing the computational issues of the Shapley value with applications in the smart grid*. PhD thesis, University of Southampton, 2015.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Ran Jin and Xinwei Deng. Ensemble modeling for data fusion in manufacturing process scale-up. *IIE Transactions*, 47(3):203–214, 2015.