Optimizing Data Collection for Machine Learning

Rafid Mahmood[1] **James Lucas**[1] **Jose M. Alvarez**[1] **Sanja Fidler**[1,2,3] **Marc T. Law**[1]
[1]NVIDIA [2]University of Toronto [3]Vector Institute
{rmahmood, jalucas, josea, sfidler, marcl}@nvidia.com
Project Page: https://nv-tlabs.github.io/LearnOptimizeCollect/

Abstract

Modern deep learning systems require huge data sets to achieve impressive performance, but there is little guidance on how much or what kind of data to collect. Over-collecting data incurs unnecessary present costs, while under-collecting may incur future costs and delay workflows. We propose a new paradigm for modeling the data collection workflow as a formal *optimal data collection problem* that allows designers to specify performance targets, collection costs, a time horizon, and penalties for failing to meet the targets. Additionally, this formulation generalizes to tasks requiring multiple data sources, such as labeled and unlabeled data used in semi-supervised learning. To solve our problem, we develop Learn-Optimize-Collect (LOC), which minimizes expected future collection costs. Finally, we numerically compare our framework to the conventional baseline of estimating data requirements by extrapolating from neural scaling laws. We significantly reduce the risks of failing to meet desired performance targets on several classification, segmentation, and detection tasks, while maintaining low total collection costs.

1 Introduction

When deploying a deep learning model in an industrial application, designers often mandate that the model must meet a pre-determined baseline performance, such as a target metric over a validation data set. For example, an object detector may require a certain minimum mean average precision before being deployed in a safety-critical setting. One of the most effective ways of meeting target performances is by collecting more training data for a given model.

Determining how much data is needed to meet performance targets can impact costs and development delays. Overestimating the data requirement incurs excess costs from collection, cleaning, and annotation. For instance, annotating segmentation masks for a driving data set takes between 15 to 40 seconds per object. For 100,000 images the annotation could require between 170 and 460 days-equivalent of time [1, 2]. On the other hand, collecting too little data may incur future costs and workflow delays from having to collect more later. For example, in medical imaging applications, this means further clinical data acquisition rounds that require expensive clinician time. In the worst case, designers may even realize that a project is infeasible only after collecting insufficient data.

The growing literature on sample complexity in machine learning has identified neural scaling laws that scale model performance with data set sizes according to power laws [3–10]. For instance, Rosenfeld et al. [6] fit power law functions on the performance statistics of small data sets to extrapolate the learning curve with more data. In contrast, Mahmood et al. [2] consider estimating data requirements and show that even small errors in a power law model of the learning curve can translate to massively over- or underestimating how much data is needed. Beyond this, different data sources have different costs and scale differently with performance [11]. For example, although unlabeled data may be easier to collect than labeled data, some semi-supervised learning tasks may need an order of magnitude more unlabeled data to match the performance of a small labeled set. Thus, collecting more data based only on estimation will fail to capture uncertainty and collection costs.

Preprint. Under review.

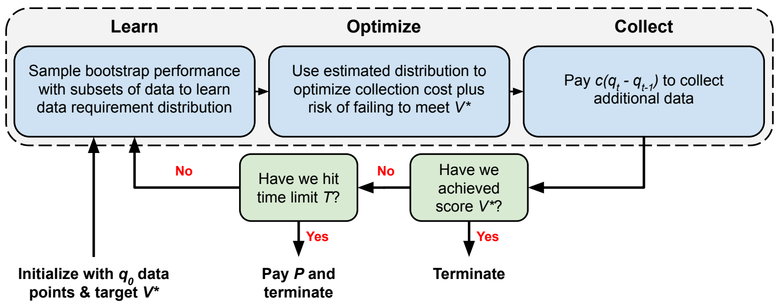
Learn
Optimize
Collect
Sample bootstrap performance with subsets of data to learn data requirement distribution
Use estimated distribution to optimize collection cost plus risk of failing to meet V*
Pay c(q_t - q_{t-1}) to collect additional data
Have we achieved score V*?
Have we hit time limit T?
No
No
Yes
Yes
Initialize with q_0 data points & target V*
Pay P and terminate
Terminate

Figure 1: In the optimal data collection problem, we iteratively determine the amount of data that we should have, pay to collect the additional data, and then re-evaluate our model. Our approach, Learn-Optimize-Collect, optimizes for the minimum amount of data q_t^* to collect.

In this paper, we propose a new paradigm for modeling the data collection workflow as an *optimal data collection problem*. Here, a designer must minimize the cost of collecting enough data to obtain a model capable of a desired performance score. They have multiple collection rounds, where after each round, they re-evaluate the model and decide how much more data to order. The data has per-sample costs and moreover, the designer pays a penalty if they fail to meet the target score within a finite horizon. Using this formal framework, we develop an optimization approach for minimizing the expected future collection costs and show that this problem can be optimized in each collection round via gradient descent. Furthermore, our optimization problem immediately generalizes to decisions over multiple data sources (e.g., unlabeled, long-tail, cross-domain, synthetic) that have different costs and impacts on performance. Finally, we demonstrate the value of optimization over naïvely estimating data set requirements (e.g., [2]) for several machine learning tasks and data sets.

Our contributions are as follows. (1) We propose the optimal data collection problem in machine learning, which formalizes data collection workflows. (2) We introduce Learn-Optimize-Collect (LOC), a learning-and-optimizing framework that minimizes future collection costs, can be solved via gradient descent, and has analytic solutions in some settings. (3) We generalize the data collection problem and LOC to a multi-variate setting where different types of data have different costs. To the best of our knowledge, this is the first exploration of data collection with general multiple data sets in machine learning, covering for example, semi-supervised and long-tail learning. (4) We perform experiments over classification, segmentation, and detection tasks to show, on average, approximately a 2× reduction in the chances of failing to meet performance targets, versus estimation baselines.

2 Related work

Neural Scaling Laws. According to the neural scaling law literature, the performance of a model on a validation set scales with the size of the training data set q via a power law $V \propto \theta_0 q^{\theta_1}$ [5, 6, 8–10, 12–16]. Hestness et al. [5] observe this property over vision, language, and audio tasks, Bahri et al. [9] develop a theoretical relationship under assumptions on over-parametrization and the Lipschitz continuity of the loss, model, and data, and Rosenfeld et al. [6] estimate power laws using smaller data sets and models to extrapolate future performance. Multi-variate scaling laws have also been considered for some specific tasks, for example in transfer learning from synthetic to real data sets [11]. Finally, Mahmood et al. [2] explore data collection by estimating the minimum amount of data needed to meet a given target performance over multiple rounds. Our paper extends these prior studies by developing an optimization problem to minimize the expected total cost of data collected. Specifically, we incorporate the uncertainty in any regression estimate of data requirements and further generalize to multiple data sources with different costs.

Active Learning. In active learning, a model sequentially collects data by selecting new subsets of an unlabeled data pool to label under a pre-determined labeling budget that replenishes after each round [17–21]. In contrast, our work focuses on systematically determining an optimal collection budget. After determining how much data to collect, we can use active learning techniques to collect the desired amount of data.

Statistical Learning Theory. Theoretical analysis of the sample complexity of machine learning models is typically only tight asymptotically, but some recent work have empirically analyzed these relationships [22, 23]. Particularly, Bisla et al. [10] study generalization bounds for deep neural networks, provide empirical validation, and suggest using them to estimate data requirements. In contrast, our paper formally explores the consequences of collection costs on data requirements.

Optimal Experiment Design. The topic of how to collect data, select samples, and design scientific experiments or controlled trials is well-studied in econometrics [24–26]. For example, Bertsimas et al. [27] optimize the assignment of samples into control and trial groups to minimize inter-group variances. Most recently, Carneiro et al. [28] optimize how many samples and covariates to collect in a statistical experiment by minimizing a treatment effect estimation error or maximizing t-test power. However, our focus on industrial machine learning applications differs from experiment design by having target performance metrics and continual rounds of collection and modeling.

3 Main Problem

In this section, we give a motivating example before introducing the formal data collection problem. We include a table of notation in Appendix A.

Motivating Example. *A startup is developing an object detector for use in autonomous vehicles within the next $T = 5$ years. Their model must achieve a mean Average Precision greater than $V^* = 95\%$ on a pre-determined validation set or else they will lose an expected profit of $P = \$1,000,000$. Collecting training data requires employing drivers to record videos and annotators to label the data, where the marginal cost of obtaining each image is approximately $c = \$1$. In order to manage annual finances, the startup must plan how much data to collect at the beginning of each year.*

Let $z \sim p(z)$ be data drawn from a distribution p. For instance, $z := (x, y)$ may correspond to images x and labels y. Consider a prediction problem for which we train a model with a data set $\mathcal{D}$ of points sampled from $p(z)$. Let $V(\mathcal{D})$ be a score function evaluating the model trained on $\mathcal{D}$.

Optimal Data Collection. We possess an initial data set $\mathcal{D}_{q_0} := \{z_i\}_{i=1}^{q_0}$ of q_0 points; we omit the subscript on $\mathcal{D}$ referring to its size when it is obvious. Our problem is defined by a target score $V^* > V(\mathcal{D}_{q_0})$, a cost-per-sample c of collection, a horizon of T rounds, and a penalty P for failure. At the end of each round $t \in \{1, \dots, T\}$, let q_t be the current amount of data collected. Our goal is to minimize the total cost of collection while building a model that can achieve the target score:

$$\min_{q_1,\dots,q_T} \; c\sum_{t=1}^{T}(q_t - q_{t-1}) + P\mathbb{1}\{V(\mathcal{D}_{q_T}) < V^*\} \qquad \text{s.t. } q_0 \le q_1 \le \cdots \le q_T \tag{1}$$

We collect training data iteratively over multiple rounds (see Figure 1), where in each round, we

1. Decide to grow the data set to $q_t \ge q_{t-1}$ points by sampling $\hat{\mathcal{D}} := \{\hat{z}_i\}_{i=1}^{q_t - q_{t-1}} \sim p(z)$. Pay a cost $c(q_t - q_{t-1})$ and update $\mathcal{D} \leftarrow \mathcal{D} \cup \hat{\mathcal{D}}$.
2. Train the model and evaluate the score. If $V(\mathcal{D}) \ge V^*$, then terminate.
3. If $t = T$, then pay the penalty P and terminate. Otherwise, repeat for the next round.

The model score typically increases monotonically with data set size [5, 6]. This means that the minimum cost strategy for (1) is to collect just enough data such that $V(\mathcal{D}_{q_T}) = V^*$. We can estimate this minimum data requirement by modeling the score function as a stochastic process. Let $V_q := V(\mathcal{D}_q)$ and let $\{V_q\}_{q \in \mathbb{Z}_+}$ be a stochastic process whose indices represent training set sizes in different rounds. Then, collecting data in each round yields a sequence of subsampled data sets $\mathcal{D}_{q_{t-1}} \subset \mathcal{D}_{q_t}$ and their performances $V(\mathcal{D}_{q_t})$. The minimum data requirement is the stopping time

$$D^* := \arg\min_q \{q \mid V_q \ge V^*\}. \tag{2}$$

which is a random variable giving the first time that we pass the target. Note that $q_1^* = \cdots = q_T^* = D^*$ is a minimum cost solution to the optimal data collection problem, incurring a total cost $c(D^* - q_0)$[1].

Estimating D^* using past observations of the learning curve is difficult since we have only T rounds. Further, Mahmood et al. [2] empirically show that small errors in fitting the learning curve can cause massive over- or under-collection. Thus, robust policies must capture the uncertainty of estimation.

[1]We assume that $c(D^* - q_0) < P$, since otherwise the optimal strategy would be to collect no data.

4 Learn-Optimize-Collect (LOC)

Our solution approach, which we refer to as Learn-Optimize-Collect (LOC), minimizes the total collection cost while incorporating the uncertainty of estimating D^*. Although D^* is a discrete random variable, it is realized typically on the order of thousands or greater. To simplify our problem and ensure differentiability, we assume that D^* is continuous and has a well-defined density.

Assumption 1. *The random variable D^* is absolutely continuous and has a cumulative density function (CDF) $F(q)$ and probability density function (PDF) $f(q) := dF(q)/dq$.*

In Section 4.1, we first develop an optimization model when given access to the CDF $f(q)$ and PDF $F(q)$. In Section 4.2, we estimate these distributions and combine them with the optimization model. Finally in Section 4.3, we delineate our optimization approach from prior regression methods.

4.1 Optimization Model

We first propose an optimization problem that at any given round t can simultaneously solve for the optimal amounts of data to collect $q_t, \dots, q_T$ in all future rounds. Consider the initial setting at $t = 1$. In order to develop intuition, let us first suppose that we know a priori the exact stopping time D^* Then, problem (1) can be re-written as

$$\min_{q_1, \cdots q_T} \; L(q_1, \dots, q_T; D^*) \qquad \text{s.t. } q_0 \le q_1 \le \cdots \le q_T \tag{3}$$

where the objective function is defined recursively as follows

$$\begin{aligned} L(q_1, \dots, q_T; D^*) &:= c(q_1 - q_0) + \mathbb{1}\{q_1 < D^*\}\Big(c(q_2 - q_1) + \mathbb{1}\{q_2 < D^*\}\Big(c(q_3 - q_2) \dots \\ &\qquad \cdots + \mathbb{1}\{q_{T-1} < D^*\}\Big(c(q_T - q_{T-1}) + P\mathbb{1}\{q_T < D^*\}\Big) \cdots\Big)\Big) \\ &= c\sum_{t=1}^{T}(q_t - q_{t-1}) \prod_{s=1}^{t-1} \mathbb{1}\{q_s < D^*\} + P \prod_{t=1}^{T} \mathbb{1}\{q_s < D^*\} \\ &= c\sum_{t=1}^{T}(q_t - q_{t-1})\mathbb{1}\{q_{t-1} < D^*\} + P\mathbb{1}\{q_T < D^*\}. \end{aligned}$$

The second line follows from gathering the terms. The third line follows from observing that since $q_1 \le q_2 \le \cdots \le q_T$ are constrained, the product of the indicators is equal to the maximum.

In practice, we do not know D^* a priori since it is an unobserved random variable. Instead, suppose we have access to the CDF $F(q)$. Then, we take the expectation over the objective $\mathbb{E}[L(q_1, \dots, q_T; D^*)]$ to formulate a *stochastic optimization problem* for determining how much data to collect:

$$\min_{q_1, \cdots q_T} \; c\sum_{t=1}^{T}(q_t - q_{t-1})\left(1 - F(q_{t-1})\right) + P\left(1 - F(q_T)\right) \quad \text{s.t. } q_0 \le q_1 \le \cdots \le q_T. \tag{4}$$

Note that the collection variables should be discrete $q_1, \dots, q_T \in \mathbb{Z}_+$, but similar to the modeling of D^*, we relax the integrality requirement, optimize over continuous variables, and round the final solutions. Furthermore, although problem (4) is constrained, we can re-formulate it with variables $d_t := q_t - q_{t-1}$; this consequently replaces the current constraints with only non-negativity constraints $d_t \ge 0$. Finally due to Assumption 1, problem (4) can be optimized via gradient descent.

4.2 Learning and Optimizing the Data Requirement

Solving problem (4) requires access to the true distribution $F(q)$, which we do not have in reality. In each round, given a current training data set $\mathcal{D}_{q_t}$ of q_t points, we must estimate these distribution functions $F(q)$ and $f(q)$ and then incorporate them into our optimization problem.

Given a current data set $\mathcal{D}_{q_t}$, we may sample an increasing sequence of R subsets $\mathcal{D}_{q_t/R} \subset \mathcal{D}_{2q_t/R} \subset \cdots \subset \mathcal{D}_{q_t}$, fit our model to each subset, and compute the scores to obtain a data set of the learning curve $\mathcal{R} := \{(rq_t/R, V(\mathcal{D}_{rq_t/R}))\}_{r=1}^{R}$. In order to model the distribution of D^*, we can take B bootstrap resamples of $\mathcal{R}$ to fit a series of regression functions and obtain corresponding estimates

$\{\hat{D}_b\}_{b=1}^B$. Given a set of estimates of the data requirement, we estimate the PDF via Kernel Density Estimation (KDE). Finally to fit the CDF, we numerically integrate the PDF.

In our complete framework, LOC, we first estimate $F(q)$ and $f(q)$. We then use these models to solve problem (4). Note that in the t-th round of collection, we fix the prior decision variables $q_1, \dots q_{t-1}$ constant. Finally, we collect data as determined by the optimal solution q_t^* to problem (4). Full details of the learning and optimization steps, including the complete Algorithm, are in Appendix B.

4.3 Comparison to Mahmood et al. [2]

Our prediction model extends the previous approach of Mahmood et al. [2], who consider only point estimation of D^*. They (i) build the set $\mathcal{R}$, (ii) fit a parametric function $\hat{v}(q;\boldsymbol{\theta})$ to $\mathcal{R}$ via least-squares minimization, and (iii) solve for $\hat{D} = \arg\min_q\{q \mid \hat{v}(q;\boldsymbol{\theta}) \geq V^*\}$. They use several parametric functions from the neural scaling law literature, including the power law function, $\hat{v}(q;\boldsymbol{\theta}) := \theta_0 q^{\theta_1} + \theta_2$ [8, 2], and use an ad hoc correction factor obtained by trial and error on past tasks to help decrease the failure rate. Instead, we take bootstrap samples of $\mathcal{R}$ to fit multiple regression functions, estimate a distribution for $\hat{D}$, and incorporate them into our novel optimization model. Finally, we show in the next two sections that our optimization problem has analytic solutions and extends to multiple sources.

5 Analytic Solutions for the $T = 1$ Setting

In this section, we explore analytic solutions for problem (4). The unobservable D^* and sequential decision-making nature suggest this problem can be formulated as a Partially Observable Markov Decision Process (POMDP) with an infinite state and action space (see Appendix C.1), but such problems rarely permit exact solution methods [29]. Nonetheless, we can derive exact solutions for the simple case of a single $T = 1$ round, re-stated below

$$\min_{q_1} \; c(q_1 - q_0) + P(1 - F(q_1)) \qquad \text{s.t. } q_0 \leq q_1 \tag{5}$$

Theorem 1. *Assume $F(q)$ is strictly increasing and continuous. If there exists $q_1 \geq q_0, \hat{\epsilon} \geq 0$ where*

$$\frac{c}{P} \leq \frac{F(q_1) - F(q_0)}{q_1 - q_0}, \qquad \hat{\epsilon} \leq 1 - F(q_0), \qquad P = c/f(F^{-1}(1-\hat{\epsilon})) \tag{6}$$

then there exists an $\epsilon \leq 1 - F(q_0)$ that satisfies $P = c/f(F^{-1}(1-\epsilon))$ and an optimal solution to the corresponding problem (5) is $q_1^* := F^{-1}(1-\epsilon)$. Otherwise, the optimal solution is $q_1^* := q_0$.

When the penalty P is specified via a failure risk ϵ, the optimal solution to problem (5) is equal to a quantile of the distribution of D^*. We defer the proof and some auxiliary results to Appendix C.2.

Theorem 1 further provides guidelines on choosing values for the cost and penalty parameters. While c is the dollar-value cost per-sample, which includes acquisition, cleaning, and annotation, P can reflect their inherent regret or opportunity cost of failing to meet their target score. A designer can accept a risk ϵ of failing to collect enough data $\Pr\{q^* < D^*\} = \epsilon$. From Theorem 1, their optimal strategy should be to collect $F^{-1}(1-\epsilon)$ points, which is also the optimal solution to problem (5).

6 The Multi-variate LOC: Collecting Data from Multiple Sources

So far, we have assumed that a designer only chooses how much data to collect and must pay a fixed per-sample collection cost. We now explore the multi-variate extension of the data collection problem where there are different types of data with different costs. For example, consider long-tail learning where samples for some rare classes are harder to obtain and thus, more expensive [30], semi-supervised learning where labeling data may cost more than collecting unlabeled data [31], or domain adaptation where a source data set is easier to obtain than a target set [32]. In this section, we highlight our main formulation and defer the complete multi-variate LOC to Appendix D.

Consider $K \in \mathbb{N}$ data sources (e.g., $K = 2$ with labeled and unlabeled) and for each $k \in \{1, \dots, K\}$, let $z^k \sim p_k(z^k)$ be data drawn from their distribution. We train a model with a data set $\mathcal{D} := \cup_{k=1}^K \mathcal{D}^k$ where each $\mathcal{D}^k$ contains points of the k-th source. The performance or score function of our model is

$V(\mathcal{D}^1, \dots, \mathcal{D}^K)$. For each k, we initialize with q_0^k points. Let $\mathbf{q}_0 = (q_0^1, \dots, q_0^K)^\mathsf{T}$ denote the vector of data set sizes and let $\mathbf{c} = (c^1, \dots, c^K)^\mathsf{T}$ denote costs (i.e., c^k is the cost of collecting data from $p_k(z^k)$). Given a target V^*, penalty P, and T rounds, we want to minimize the total cost of collection

$$\min_{\mathbf{q}_1, \dots, \mathbf{q}_T} \; \mathbf{c}^\mathsf{T} \sum_{t=1}^{T} (\mathbf{q}_t - \mathbf{q}_{t-1}) + P \mathbb{1}\{V(\mathcal{D}_{q_T^1}, \dots, \mathcal{D}_{q_T^K}) < V^*\} \qquad \text{s.t. } \mathbf{q}_0 \leq \mathbf{q}_1 \leq \mathbf{q}_2 \leq \cdots \leq \mathbf{q}_T$$

We follow the same steps shown in Section 4 for this problem. First, the learning curve is now a stochastic process $\{V_\mathbf{q}\}_{\mathbf{q} \in \mathbb{Z}_+^K}$ indexed in K dimensions. Next, the multi-variate analogue of the minimum data requirement in (2) is the minimum cost amount of data needed to meet the target:

$$\mathbf{D}^* := \arg\min_{\mathbf{q}} \left\{ \mathbf{c}^\mathsf{T}\mathbf{q} \mid V_\mathbf{q} \geq V^* \right\}$$

We randomly pick a unique solution to break ties. From Assumption 1, $\mathbf{D}^*$ is a random vector with a PDF $f(\mathbf{q})$ and a CDF $F(\mathbf{q}) := \int_\mathbf{0}^\mathbf{q} f(\hat{\mathbf{q}}) d\hat{\mathbf{q}}$. Finally, the multi-variate analogue of problem (4) is

$$\min_{\mathbf{q}_1, \cdots, \mathbf{q}_T} \; \mathbf{c}^\mathsf{T} \sum_{t=1}^{T} (\mathbf{q}_t - \mathbf{q}_{t-1}) \left(1 - F(\mathbf{q}_{t-1})\right) + P \left(1 - F(\mathbf{q}_T)\right) \text{ s.t. } \mathbf{q}_0 \leq \mathbf{q}_1 \leq \cdots \leq \mathbf{q}_T \qquad (7)$$

The Multi-variate LOC requires multi-variate PDFs, which we can fit in the same way as discussed in Section 4.2. However, we now need multi-variate regression functions that can accommodate different types of data. In Appendix D, we propose an additive family of power law regression functions that can handle an arbitrary number of K sources. In our experiments, we also generalize the estimation approach of Mahmood et al. [2] to the multi-source setting for comparison.

7 Empirical Results

We explore the data collection problem over two sets of experiments covering single-variate $K = 1$ (Section 4) and multi-variate $K = 2$ (Section 6) problems. We consider image classification, segmentation, and object detection tasks. For every data set and task, LOC significantly reduces the number of instances where we fail to meet a data requirement V^*, while incurring a competitive cost with respect to the conventional baseline of naïvely estimating the data requirement [2].

In this section, we summarize the main results. We detail our data collection and experiment setup in Appendix E. We expand our full results and experiments with additional baselines in Appendix F .

7.1 Data and Methods

When $K = 1$, the designer decides how much data to sample without controlling the type of data. We explore classification on CIFAR-10 [33], CIFAR-100 [33], and ImageNet [34], where we train ResNets [35] to meet a target validation accuracy. We explore semantic segmentation using Deeplabv3 [36] on BDD100K [37], which is a large-scale driving data set, as well as Bird's-Eye-View (BEV) segmentation on nuScenes [38] using the 'Lift Splat' architecture [39]; for both tasks, we desire a target mean intersection-over-union (IoU). We explore 2-D object detection on PASCAL VOC [40, 41] using SSD300 [42], where we evaluate mean average precision (mAP).

When $K = 2$, the designer collects two types of data with different costs. We first divide CIFAR-100 into two subsets containing data from the first and last 50 classes, respectively. Here, we assume that the first 50 classes are more expensive to collect than the last; this mimics a real-world scenario where collecting data for some classes (e.g., long-tail) is more expensive than others. We then explore semi-supervised learning on BDD100K where the labeled subset of this data is more expensive than the unlabeled data; the cost difference between these two types is equal to the cost of data annotation.

We use a simulation model of the deep learning workflow following the procedure of Mahmood et al. [2], to approximate the true problem while simplifying the experiments (see Appendix E for full details). To avoid repeatedly sampling data, re-training a model, and evaluating the score, each simulation uses a piecewise-linear approximation of a 'ground truth' learning curve that returns model performance as a function of data set size. In our problems, we initialize with $q_0 = 10\%$ of the full data set (we use 20% for VOC). Then in each round, we solve for the amount of data to collect and then call the piecewise-linear learning curve to obtain the current score.

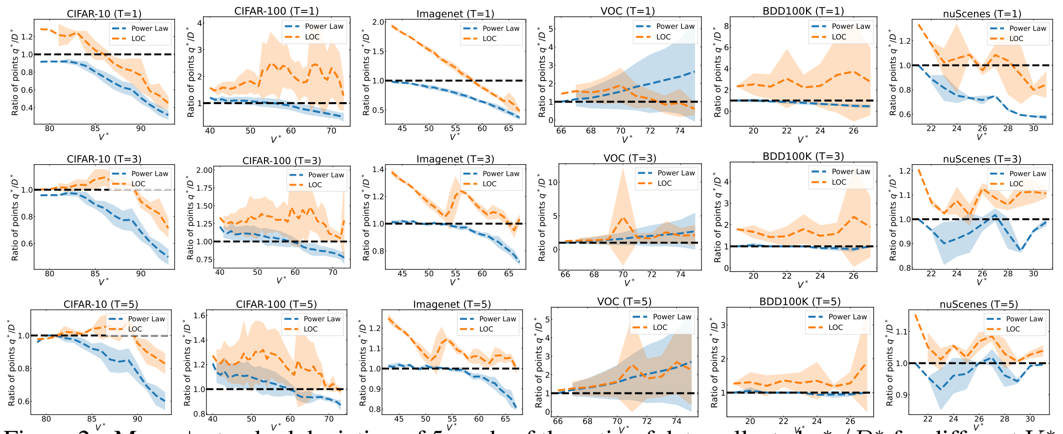
Figure 2: Mean $\pm$ standard deviation of 5 seeds of the ratio of data collected q_T^*/D^* for different V^*. The rows correspond to $T = 1, 3, 5$ and the columns to different data sets. The black line corresponds to collecting exactly the minimum data requirement. LOC almost always remains slightly above the black line, meaning we rarely fail to meet the target.

	Data set	T	Scaling Law Regression		LOC	
			Failure rate	Cost ratio	Failure rate	Cost ratio
Class.	CIFAR-10	1	100%	–	**60%**	0.19
		3	95%	0.00	**32%**	0.05
		5	86%	0.00	**29%**	0.03
	CIFAR-100	1	56%	0.12	**4%**	0.99
		3	48%	0.10	**3%**	0.31
		5	48%	0.10	**2%**	0.19
	Imagenet	1	99%	0.00	**37%**	0.49
		3	75%	0.01	**5%**	0.16
		5	56%	0.01	**2%**	0.10
Seg.	BDD100K	1	77%	0.03	**12%**	2.03
		3	31%	0.00	**0%**	0.72
		5	23%	0.01	**0%**	0.35
	nuScenes	1	95%	0.00	**52%**	0.16
		3	71%	0.01	**0%**	0.09
		5	62%	0.00	**0%**	0.04
Det.	VOC	1	36%	1.24	**25%**	0.56
		3	8%	0.88	**0%**	1.10
		5	6%	0.86	**0%**	0.84

Table 1: Average cost ratio $\mathbf{c}^\top(\mathbf{q}_T^* - \mathbf{q}_0)/\mathbf{c}^\top(\mathbf{D}^* - \mathbf{q}_0) - 1$ and failure rate measured over a range of V^* for each T and data set. We fix $c = 1$ and $P = 10^7$ ($P = 10^6$ for VOC and $P = 10^8$ for ImageNet). The best performing failure rate for each setting is bolded. The cost ratio is measured only for instances that achieve V^*. LOC consistently reduces the average failure rate, often down to 0%, while keeping the average cost ratio almost always below 1 (i.e., spending at most $2\times$ the optimal amount).

We compare LOC against the conventional estimation approach of Mahmood et al. [2], who fit a regression model to the learning curve statistics, extrapolate the learning curve for larger data sets, and then solve for the minimum data requirement under this extrapolation. There are many different regression models that can be used to fit learning curves [12, 14, 5, 8]. Since power laws are the most commonly studied approach in the neural scaling law literature, we focus on these. In Appendix F.4, we show that our optimization approach can be incorporated with other regression models.

7.2 Main Results

We consider $T = 1, 3, 5$ rounds and $V^* \in [V(\mathcal{D}_{q_0}) + 1, V(\mathcal{D})]$ targets, where $\mathcal{D}$ is the entire data set. We evaluate all methods on (i) the failure rate, which is how often the method fails to achieve the given V^* within T rounds, and (ii) the cost ratio, which is the suboptimality of an algorithm for solving problem (4), i.e., $\mathbf{c}^\top(\mathbf{q}_T^* - \mathbf{q}_0)/\mathbf{c}^\top(\mathbf{D}^* - \mathbf{q}_0) - 1$. Note that the suboptimality does not count the penalty for failure since this would distort the average metrics. For $K = 1$, we also measure the ratio of points collected q_T^*/D^*. Although there is a natural trade-off between low cost ratio (under-collecting) and failure rate (over-collecting), we emphasize that our goal is to have low cost but with zero chance of failure.

The Value of Optimization over Estimation when $K = 1$. Figure 2 compares LOC versus the corresponding power law regression baseline when $c = 1$ and $P = 10^7$ ($P = 10^6$ for VOC and $P = 10^8$ for ImageNet). If a curve is below the black line, then it failed to collect enough data to

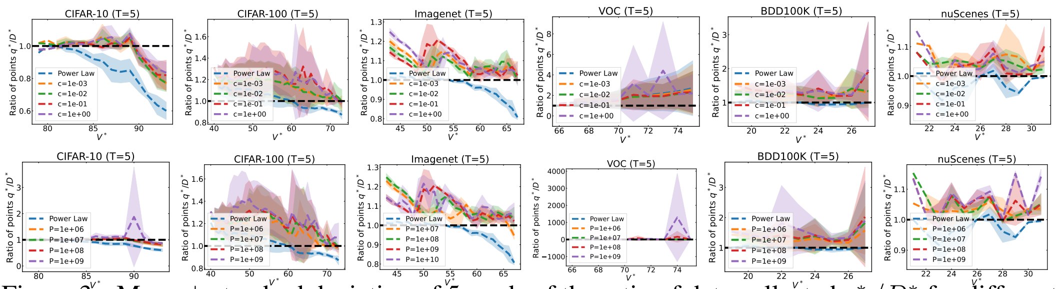
Figure 3: Mean $\pm$ standard deviation of 5 seeds of the ratio of data collected q_T^*/D^* for different V^* and fixed $T = 5$. *Top:* We sweep the cost parameter from 0.001 to 1 and fix $P = 10^7$. *Bottom:* We sweep the penalty parameter from 10^6 to 10^9 and fix $c = 1$. The dashed black line corresponds to collecting exactly the minimum data requirement. See Appendix F for all T.

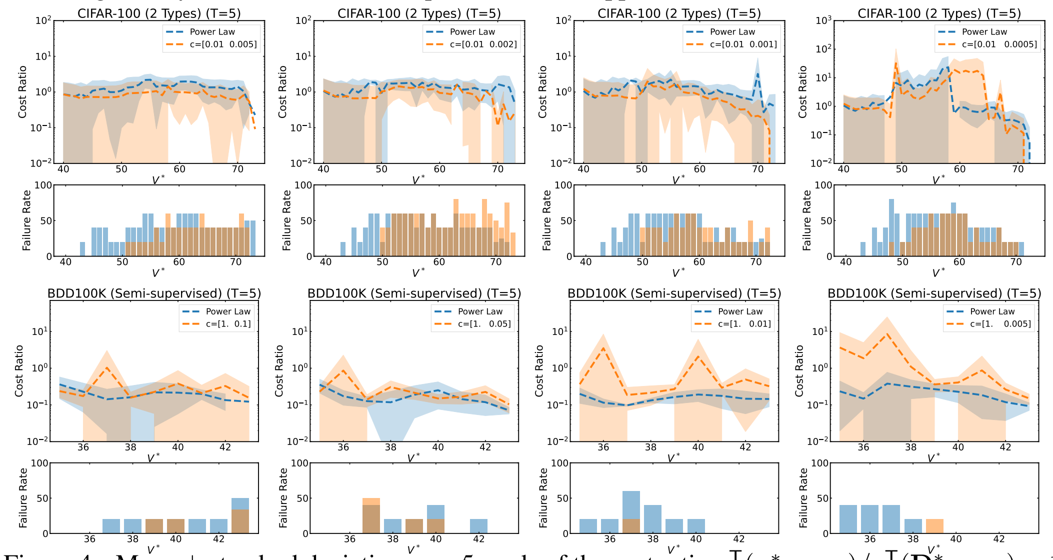
Figure 4: Mean $\pm$ standard deviation over 5 seeds of the cost ratio $\mathbf{c}^\top(\mathbf{q}_T^* - \mathbf{q}_0)/\mathbf{c}^\top(\mathbf{D}^* - \mathbf{q}_0) - 1$ and failure rate for different V^*, after removing 99-th percentile outliers. The columns correspond to scenarios where the first set c^1 costs increasingly more than the second c^2. See Appendix F for all T.

meet the target. LOC consistently remains above this black line for most settings. In contrast, even with up to $T = 5$ rounds, collecting data based only on regression estimates leads to failure.

Table 1 aggregates the failure rates and cost ratios for each setting. To summarize, LOC fails at less than 10% of instances for $12/18$ settings, whereas regression fails over 30% for $15/18$ settings. In particular, regression nearly always under-collects data when given a single $T = 1$ round. Here, LOC reduces the risk of under-collecting by 40% to 90% over the baseline. While this leads to a marginal increase in costs, our cost ratios are consistently less than 0.5 for $12/18$ settings, meaning that we spend at most 50% more than the true minimum cost.

We remark that previously, Mahmood et al. [2] observed that incorrect regression estimates necessitated real machine learning workflows to collect data over multiple rounds. Instead, with LOC, we can make significantly improved data collection decisions even with a single round.

Robustness to Cost and Penalty Parameters (see Appendix F.2 for details). Figure 3 evaluates the ratio of points collected for $T = 5$ when the cost and the penalty of the optimization problem are varied. Our algorithm is robust to variations in these parameters, as LOC retains the same shape and scale for almost every parameter setting and data set. Further, LOC consistently remains above the horizontal 1 line, showing that even after varying c and P, we do not fail as frequently as the baseline. Finally, validating Theorem 1, the penalty parameter P provides natural control over the amount of data collected. As we increase P, the ratio of data collected increases consistently.

The Value of Optimization over Estimation when $K = 2$ (Appendix F.3). Figure 4 compares LOC versus regression at $T = 5$ with different costs, showing that we maintain a similar cost ratio to the regression alternative, but with lower failure rates. Table 2 aggregates failure rates and cost ratios

Data set	T	Cost	Power Law Regression		LOC	
			Failure rate	Cost ratio	Failure rate	Cost ratio
CIFAR-100 (2 Types)	1	$(0.01, 0.0005)$	62%	0.89	**40%**	41.80
		$(0.01, 0.001)$	58%	1.19	**46%**	9.85
		$(0.01, 0.002)$	56%	1.55	**54%**	6.98
		$(0.01, 0.005)$	54%	1.65	**33%**	4.43
	3	$(0.01, 0.0005)$	43%	3.47	**30%**	4.88
		$(0.01, 0.001)$	45%	1.22	**43%**	1.31
		$(0.01, 0.002)$	45%	1.47	**44%**	1.21
		$(0.01, 0.005)$	38%	1.31	**36%**	1.17
	5	$(0.01, 0.0005)$	38%	3.31	**24%**	5.19
		$(0.01, 0.001)$	35%	1.22	**24%**	0.79
		$(0.01, 0.002)$	**37%**	1.33	38%	0.90
		$(0.01, 0.005)$	36%	1.30	**24%**	0.82
BDD100K (Semi-supervised)	1	$(1, 0.005)$	86%	0.11	**44%**	7.02
		$(1, 0.01)$	79%	0.15	**30%**	13.47
		$(1, 0.05)$	72%	0.19	**49%**	1.02
		$(1, 0.1)$	70%	0.19	**65%**	0.40
	3	$(1, 0.005)$	23%	0.18	**7%**	1.20
		$(1, 0.01)$	21%	0.15	**7%**	2.57
		$(1, 0.05)$	26%	0.18	**23%**	0.50
		$(1, 0.1)$	26%	0.21	**30%**	0.15
	5	$(1, 0.005)$	16%	0.22	**2%**	1.91
		$(1, 0.01)$	21%	0.15	**2%**	0.86
		$(1, 0.05)$	16%	0.17	**9%**	0.27
		$(1, 0.1)$	16%	0.20	**7%**	0.32

Table 2: Average cost ratio $\mathbf{c}^\intercal(\mathbf{q}_T^* - \mathbf{q}_0)/\mathbf{c}^\intercal(\mathbf{D}^* - \mathbf{q}_0) - 1$ and failure rate over different V^* for each T and $\mathbf{c}$, after removing 99-th percentile outliers. We fix $P = 10^{13}$ for CIFAR-100 and $P = 10^8$ for BDD100K. The best performing failure rate for each setting is bolded. The cost ratio is measured over instances that achieve V^*. LOC consistently reduces the average failure rate, and for $T > 1$, preserves the cost ratio. Further, LOC is more robust to uneven costs than regression.

for all settings, showing LOC consistently achieves lower failure rates for nearly all settings of T. When $T = 5$, LOC also achieves lower cost ratios versus regression on CIFAR-100, meaning that with multiple rounds of collection, we can ensure meeting performance requirements while paying nearly the optimal amount of data. However, solving the optimization problem is generally more difficult as K increases, and we sometimes over-collect data by large margins. In practice, these outliers can be identified from common sense (e.g.,

consequently, we report these results after removing the 99-th percentile outliers with respect to total cost for both methods. Nonetheless, this challenge remains when $T = 1$, particularly for CIFAR-100.

8 Discussion

We develop a rigorous framework for optimizing data collection workflows in machine learning applications, by introducing an optimal data collection problem that captures the uncertainty in estimating data requirements. We generalize this problem to more realistic settings where multiple data sources incur different collection costs. We validate our solution algorithm, LOC, on six data sets covering classification, segmentation, and detection tasks to show that we consistently meet pre-determined performance metrics regardless of costs and time horizons.

Our approach relies on estimating the CDF and PDF of the minimum data requirement, which is a challenging problem, especially with multiple data sources. Nonetheless, LOC can be deployed on top of future advances in estimating neural scaling laws. Further, we allow practitioners to input problem-specific costs and penalties, but these quantities may not always be readily available. We provide some theoretical insight into parameter selection and show that LOC is robust to these parameters. Finally, our empirical analysis focuses on computer vision, but we expect our approach to be viable in other domains governed by scaling laws.

Improving data collection practices yields potentially positive and negative societal impacts. LOC reduces the collection of extraneous data, which can, in turn, reduce the environmental costs of training models. On the other hand, equitable data collection should also be considered in real-world data collection practices that involve humans. We envision a potential future work to incorporate privacy and fairness constraints to prevent over- or under-sampling of protected groups. Finally, our method is guided by a score function on a held-out validation set. Biases in this set may be exacerbated when optimizing data collection to meet target performance.

There is a folklore observation that over 80% of industry machine learning projects fail to reach production, often due to insufficient, noisy, or inappropriate data [43, 44]. Our experiments verify this by showing that naïvely estimating data requirements will often yield failures to meet target performances. We believe that robust data collection policies obtained via LOC can reduce failures while further guiding practitioners on how to manage both costs and time.

Acknowledgments and Disclosure of Funding

We thank Jonathan Lorraine, Daiqing Li, Jonah Philion, David Acuna, and Zhiding Yu, as well as the anonymous reviewers and meta-reviewers for valuable feedback on earlier versions of this paper.

References

[1] David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[2] Rafid Mahmood, James Lucas, David Acuna, Daiqing Li, Jonah Philion, Jose M. Alvarez, Zhiding Yu, Sanja Fidler, and Marc T. Law. How much more data do we need? estimating requirements for downstream tasks. In *2022 IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2022.

[3] Lewis J Frey and Douglas H Fisher. Modeling decision tree performance with the power law. In *Seventh International Workshop on Artificial Intelligence and Statistics*. PMLR, 1999.

[4] Baohua Gu, Feifang Hu, and Huan Liu. Modelling classification performance for large data sets. In *International Conference on Web-Age Information Management*, pages 317–328. Springer, 2001.

[5] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Patwary, Mostofa Ali, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

[6] Jonathan S Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. A constructive prediction of the generalization error across scales. In *International Conference on Learning Representations*, 2020.

[7] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[8] Derek Hoiem, Tanmay Gupta, Zhizhong Li, and Michal Shlapentokh-Rothman. Learning curves for analysis of deep networks. In *International Conference on Machine Learning*, pages 4287–4296. PMLR, 2021.

[9] Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *arXiv preprint arXiv:2102.06701*, 2021.

[10] Devansh Bisla, Apoorva Nandini Saridena, and Anna Choromanska. A theoretical-empirical approach to estimating sample complexity of dnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3270–3280, 2021.

[11] Hiroaki Mikami, Kenji Fukumizu, Shogo Murai, Shuji Suzuki, Yuta Kikuchi, Taiji Suzuki, Shin-ichi Maeda, and Kohei Hayashi. A scaling law for synthetic-to-real transfer: How much is your pre-training effective? *arXiv preprint arXiv:2108.11018*, 2021.

[12] S Jones, S Carley, and M Harrison. An introduction to power and sample size estimation. *Emergency Medicine Journal: EMJ*, 20(5):453, 2003.

[13] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 843–852, 2017.

[14] Rosa L Figueroa, Qing Zeng-Treitler, Sasikiran Kandula, and Long H Ngo. Predicting sample size required for classification performance. *BMC Medical Informatics and Decision Making*, 12(1):1–10, 2012.

[15] Tom Viering and Marco Loog. The shape of learning curves: a review. *arXiv preprint arXiv:2103.10948*, 2021.

[16] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[17] Burr Settles. Active learning literature survey. 2009.

[18] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018.

[19] Donggeun Yoo and In So Kweon. Learning loss for active learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 93–102, 2019.

[20] Samarth Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational adversarial active learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5972–5981, 2019.

[21] Rafid Mahmood, Sanja Fidler, and Marc T. Law. Low-budget active learning via wasserstein distance: An integer programming approach. In *International Conference on Learning Representations*, 2022.

[22] Yiding Jiang, Pierre Foret, Scott Yak, Daniel M Roy, Hossein Mobahi, Gintare Karolina Dziugaite, Samy Bengio, Suriya Gunasekar, Isabelle Guyon, and Behnam Neyshabur. Neurips 2020 competition: Predicting generalization in deep learning. *arXiv preprint arXiv:2012.07976*, 2020.

[23] Yiding Jiang, Parth Natekar, Manik Sharma, Sumukh K Aithal, Dhruva Kashyap, Natarajan Subramanyam, Carlos Lassance, Daniel M Roy, Gintare Karolina Dziugaite, Suriya Gunasekar, et al. Methods and analysis of the first competition in predicting generalization of deep learning. In *NeurIPS 2020 Competition and Demonstration Track*, pages 170–190. PMLR, 2021.

[24] Kirstine Smith. On the standard deviations of adjusted and interpolated values of an observed polynomial function and its constants and the guidance they give towards a proper choice of the distribution of observations. *Biometrika*, 12(1/2):1–85, 1918.

[25] David Cohn. Neural network exploration using optimal experiment design. *Advances in neural information processing systems*, 6, 1993.

[26] Ashley F Emery and Aleksey V Nenarokomov. Optimal experiment design. *Measurement Science and Technology*, 9(6):864, 1998.

[27] Dimitris Bertsimas, Mac Johnson, and Nathan Kallus. The power of optimization over randomization in designing experiments involving small samples. *Operations Research*, 63(4):868–876, 2015.

[28] Pedro Carneiro, Sokbae Lee, and Daniel Wilhelm. Optimal data collection for randomized control trials. *The Econometrics Journal*, 23(1):1–31, 2020.

[29] Hao Zhang. Dynamic learning and decision making via basis weight vectors. *Operations Research*, 2022.

[30] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert systems with applications*, 73:220–239, 2017.

[31] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine Learning*, 109 (2):373–440, 2020.

[32] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

[33] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[34] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee, 2009.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[36] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[37] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 2636–2645, 2020.

[38] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020.

[39] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *Proceedings of the European Conference on Computer Vision*, 2020.

[40] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html, .

[41] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html, .

[42] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*, pages 21–37. Springer, 2016.

[43] Rob van der Meulen and Thomas McCall. Gartner says nearly half of cios are planning to deploy artificial intelligence, Feb 2018. URL `https://www.gartner.com/en/newsroom/press-releases/2018-02-13-gartner-says-nearly-half-of-cios-are-planning-to-deploy-artificial-intelligence`.

[44] Why do 87% of data science projects never make it into production?, Jul 2019. URL `https://venturebeat.com/2019/07/19/why-do-87-of-data-science-projects-never-make-it-into-production/`.

[45] Jorge J Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[46] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[47] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[48] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 1. Athena scientific, 2012.

[49] David Easley and Nicholas M Kiefer. Controlling a stochastic process with unknown parameters. *Econometrica: Journal of the Econometric Society*, pages 1045–1064, 1988.

[50] Richard D Smallwood and Edward J Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5):1071–1088, 1973.

[51] Eric Zhao, Anqi Liu, Animashree Anandkumar, and Yisong Yue. Active learning under label shift. In *International Conference on Artificial Intelligence and Statistics*, pages 3412–3420. PMLR, 2021.

[52] 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.

[53] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations*, 2020.

[54] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2015.

[55] Ismail Elezi, Zhiding Yu, Anima Anandkumar, Laura Leal-Taixe, and Jose M Alvarez. Not all labels are equal: Rationalizing the labeling costs for training object detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See the Conclusion.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See the Conclusion.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Assumption 1 and the theorem statements.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix C.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The code is proprietary. The data is publicly available. See Algorithm 1 and Appendix F for implementation details.

   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix E and F for details.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All experiments were over five seeds. See the for standard deviation ranges.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix E.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite all data sets and model architectures in Appendix E.
   (b) Did you mention the license of the assets? [N/A] All data sets and models are publicly available.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Table of Notation

$T \in \mathbb{N}$	Total number of rounds of collection.
$V^* \in \mathbb{R}$	Target performance.
$P \in \mathbb{N}$	Penalty for failing to reach target performance.
$c \in \mathbb{N}$	Cost of per-item collection.
$q_t \in \mathbb{N},\ t \in \{0, \dots, T\}$	Number of data points in round t.
$\mathcal{D}_q$	Data set of size q.
$V(\mathcal{D})$	Valuation of model trained on data set $\mathcal{D}$.
$V_q := V(\mathcal{D}_q)$	Short-hand notation for model valuation.
$D^* := \arg\min_q \{q \mid V_q \geq V^*\}$	The minimum data requirement.
$F(q) = P(D^* < q)$	The CDF of the minimum data requirement.
$f(q) = dF(q)/dq$	The PDF of the minimum data requirement.
$L(q_1, \dots, q_T; D^*)$	The stochastic objective function for data collection.
$d_t \geq 0$	Amount of data to collect in round t.
$\mathcal{R} := \left\{(rq_t/R, V(\mathcal{D}_{rq_t/R}))\right\}_{r=1}^{R}$	The regression set representing the learning curve of a given model and data set.
$K \in \mathbb{N}$	Number of data sources in multi-source setting.
$\mathbf{q}_t \in \mathbb{N}^K$	Vector containing number of data points per-source in round t.
$\mathbf{c} \in \mathbb{N}^K$	Vector of per-source costs.

Table 3: Table of notation used throughout paper.

B Learn-Optimize-Collect

Algorithm 1 summarizes the complete workflow of LOC within the data collection problem. Given a target score V^*, we collect data until we have met the target or until T rounds have passed. In this section, we first expand our complete LOC algorithm. We then provide further details on the regression procedure used to estimate the data requirement distributions. Finally, we introduce a more practical reformulation of our optimization problem (4), which is what we later use in our numerical experiments.

B.1 Details on the Learning and Optimizing Approach

Our approach for determining how much data to collect consists of three steps. The first step is to collect performance statistics by measuring the training dynamics of the current data set $\mathcal{D}_{q_t}$. Here, we sample subsets of the current data set, train our model on the subset, and evaluate the score. We repeat this process for up to R subsets, where R denotes the size of our performance statistics regression data set $\mathcal{R}$.

The second step is to model the data requirement distribution D^*. We perform this by creating bootstrap samples from $\mathcal{R}$, and then fitting a regression model $\hat{v}(q; \boldsymbol{\theta})$ that can estimate the model score as a function of data set size. We then invert this regression function by solving for the minimum $\hat{q}$ for which the model will predict that we have exceeded the target performance. We repeat this process to obtain B bootstrap estimates of D^*. We can then fit any density estimation model to approximate a probability density and a cumulative density function. In this paper, we focus on Kernel Density Estimation (KDE) and Gaussian Mixture models since such models can be easily fit and provide functional forms of the PDF.

Finally in the third step, we solve our optimization problem (4) via gradient descent. This problem yields the optimal data set sizes $q_1^*, \dots, q_T^*$ that we should have at the end of each round. Furthermore, if we are in the t-th round for $t > 1$, we freeze the values for $q_1, \dots, q_{t-1}$ to the data set sizes that we have observed in the previous rounds. Upon solving this problem, we then collect data until we have q_t samples, and then re-train our model to evaluate our current state.

Algorithm 1 Learn-Optimize-Collect (LOC)

```
1: Input: Initial data set $\mathcal{D}_{q_0}$, Sampling distribution $p(z)$, Score function $V(\mathcal{D})$, Target score $V^*$, Maximum
   rounds $T$, Cost $c$, Penalty $P$, Regression model $\hat{v}(q;\boldsymbol{\theta})$, Regression set size $R$, Density Estimation model
   $f(q)$, Number of bootstrap samples $B$.
2: Initialize round $t = 0$, loss $L = 0$
3: repeat
4:     COLLECT PERFORMANCE STATISTICS
5:         Initialize $\mathcal{R} = \emptyset$, $\mathcal{D}_0 = \emptyset$
6:         for $r \in \{1, \dots, R\}$ do
7:             Sub-sample $\mathcal{D}_{q_t r/R} \subset \mathcal{D}_{q_t}$ by augmenting to $\mathcal{D}_{q_t (r-1)/R}$
8:             Evaluate $V(\mathcal{D}_{q_t r/R})$ and update $\mathcal{R} \leftarrow \mathcal{R} \cup \{(q_t r/R, V(\mathcal{D}_{q_t r/R}))\}$
9:         end for
10:    end
11:    ESTIMATE DATA REQUIREMENT DISTRIBUTION
12:        Initialize $\mathcal{Q} = \emptyset$
13:        for $b \in \{1, \dots, B\}$ do
14:            Create bootstrap $\mathcal{R}_b$ by sub-sampling with replacement from $\mathcal{R}$
15:            Fit regression model $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{(q,v)\in\mathcal{R}_b} (v - \hat{v}(q;\boldsymbol{\theta}))^2$
16:            Estimate data requirement $\hat{q}_b = \arg\min_q \{q \mid \hat{v}(q;\boldsymbol{\theta}^*) \geq V^*\}$
17:            Update $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{\hat{q}_b\}$
18:        end for
19:        Fit Density Estimation model $f(q)$ using the empirical distribution $\mathcal{Q}$ and let $F(q) = \int_0^q f(q)dq$
20:    end
21:    COLLECT DATA
22:        Solve problem (4) using Density Estimation models $f(q)$ and $F(q)$ to obtain $q_1^*, \dots, q_T^*$.
23:        Sample $\hat{\mathcal{D}} \sim p(z)$ until $|\hat{\mathcal{D}}| = q_t - q_{t-1}$
24:        Update loss $L \leftarrow L + c(q_t - q_{t-1})$
25:        Update $\mathcal{D}_{q_t} = \mathcal{D}_{q_{t-1}} \cup \hat{\mathcal{D}}$
26:    end
27:    Evaluate $V(\mathcal{D}_{q_t})$ and update $t \leftarrow t + 1$
28: until $V(\mathcal{D}_{q_t}) \geq V^*$ or $t = T$
29: if $V(\mathcal{D}_{q_t}) < V^*$ then
30:     Update loss $L \leftarrow L + P$
31: end if
32: Output: Final collected data set $\mathcal{D}_{q_t}$, loss $L$
```

B.2 Regression Model for D^*

To estimate the data requirement, we build a regression model of the learning curve and then invert this curve. The classical learning curve literature proposes using structured functions to regress on the learning curve. For example with neural scaling laws, the most common function is a power law model $\hat{v}(q;\boldsymbol{\theta}) := \theta_0 q^{\theta_1} + \theta_2$ [5, 15, 13, 6, 9, 10, 16, 8, 2]. As a result, the main experiments of our paper use this model.

We fit each regression function by minimizing a least squares problem using the Levenberg-Marquardt algorithm as implemented by Scipy [45, 46]. The parameters for each function are initialized to either 1 or 0 depending on if they are product or bias terms. To further help fit the data, we use weighted least squares where each subsequent point is weighted twice as much as the previous point. This ensures that our regression model is tuned to better fit the curve for larger q.

B.3 Alternate Re-formulation of Problem (4)

Although problem (4) has a differentiable objective function, it includes a set of lower bound constraints, meaning that a vanilla gradient descent algorithm may not immediately apply. A naive solution algorithm may be to use projected gradient descent. Instead, we show that the problem can be reformulated to remove these constraints.

For each t, let $d_t = q_t - q_{t-1}$ denote the additional amount of data collected in each round. Note that we can recursively re-define $q_t = q_0 + \sum_{r=1}^{t} d_r$. Furthermore, the ordering inequality constraints can all be re-written to non-negativity constraints $d_t \geq 0$. We now re-write problem (4) using these new variables:

$$\begin{aligned} \min_{d_1,\cdots d_T} \quad & c\sum_{t=1}^{T} d_t \left(1 - F\left(q_0 + \sum_{r=1}^{t-1} d_r\right)\right) + P\left(1 - F\left(q_0 + \sum_{r=1}^{T} d_r\right)\right) \\ \text{s.t.} \quad & d_1, \ldots, d_T \geq 0. \end{aligned}$$

The objective in the above formulation remains differentiable. Moreover, the non-negativity constraints can be removed by re-writing $d_t \leftarrow \text{softplus}(d_t)$ for all t. As a result, we can solve this problem by using an off-the-shelf gradient descent algorithm. Finally we note that in our experiments, we implemented a projected gradient descent based on (4) and the above gradient descent algorithm. We found that the above streamlined approach to generate slightly better final solutions.

C Details on the Main Theory

In this section, we explore mathematical properties and challenges of the optimal data collection problem. In Section C.1, we discuss the drawbacks of alternative approaches to our modeling framework. Then in Section C.2, we prove our main theorem demonstrating an analytic solution in the single-round problem.

C.1 Markov Decision Process Alternatives

The data collection problem requires sequential decision-making in terms of collecting additional data in each round. A natural modeling approach for such problems is via Markov Decision Processes (MDPs). However, MDP techniques are challenging for this problem due to (i) the unobservability of the state, and (ii) an infinite state space. That is, until we have collected enough data to meet the target, we do not know how much data is the minimum, which can furthermore be arbitrarily large. Here, we sketch a potential alternative MDP framework and highlight the core challenge.

Our sequential decision-making problem can be written as a Partially Observable Markov Decision Process (POMDP) [47, 48]. Furthermore, because this state variable is constant throughout the collection problem, we can write it as an EK 'Learning-and-Doing' model [49]. Such POMDPs are defined by the tuple $(\Theta, \mathcal{A}, \mathcal{S}, p, r_t)$, where the state space characterizes the data requirement $D^* \in \Theta := \mathbb{R}_+$, the action space characterizes the additional data collected $d_t := (q_t - q_{t-1}) \in \mathcal{A} := \mathbb{R}_+$, and the observation set $\mathcal{S} := \{0, 1\}$ characterizes a binary variable $\mathbb{1}\{V(\mathcal{D}_{q_t}) \geq V^*\} = \mathbb{1}\{q_t \geq D^*\}$. Furthermore, $p(\cdot|D^*, d_t)$ is the observation transition probability and $r_t(\cdot)$ is the reward function where $r_t(q_t, q_{t-1}, D^*) := -c(q_t - q_{t-1})$ for $t \leq T$ and $r_{T+1}(q_t, q_{t-1}, D^*) := -P\mathbb{1}\{q_T < D^*\}$.

Because the state variable is unobserved, POMDPs are typically solved by using a belief distribution of the state variable to average the reward in the value function. When both the state and the action space of a POMDP are finite, Smallwood and Sondik [50] show that this value function is piecewise-linear and can be solved by exact methods, albeit under a curse of dimensionality with respect to these spaces. Unfortunately, for a general POMDP with an infinite state and action space, these methods do not apply and we most often resort to approximation techniques [51]. Moreover in our case, approximations based on discretizing the state and action space fall prey to the curse of dimensionality.

Alternatively, we may naively consider applying reinforcement learning. However, note that real-world data collection tasks do not contain the requisite sizes of learning data or generalizable simulation mechanisms that are staples in reinforcement learning techniques. These challenges, coupled with the goal of delivering practical managerial guidelines for data collection operations, motivate us to explore easy-to-implement techniques for optimizing data collection.

C.2 Proof of Theorem 1

Our main theorem states that the one-round problem has an analytic solution. However, the proof requires several auxiliary results. For clarity, we first reproduce the theorem.

Theorem 1 (Repeated). *Consider the one-round problem*

$$\min_{q_1} \; c(q_1 - q_0) + P(1 - F(q_1)) \qquad \text{s.t. } q_0 \leq q_1$$

Assume $F(q)$ is strictly increasing and continuous. For any ϵ such that $F(q_0) < 1 - \epsilon$, let $P := c/f(F^{-1}(1-\epsilon))$. The optimal solution to the corresponding problem (5) *is $q_1^* = F^{-1}(1-\epsilon)$. Furthermore, the optimal solution satisfies $F(q_1^*) = 1 - \epsilon$.*

The assumption of a strictly increasing and continuous cumulative density function is needed to ensure that the data requirement distribution has a well-defined quantile function $F^{-1}(p) := \inf_q\{q \mid F(q) \geq p\}$, where the optimal solution for any $p \in (0,1)$ is unique.

The proof for Theorem 1 relies on equating the one-round optimization problem to the following constrained optimization problem:

$$\min_{q_1}\ c(q_1 - q_0) \qquad \text{s.t.}\ F(q_1) \geq 1 - \epsilon\ ,\ \ q_0 \leq q_1 \tag{8}$$

where $\epsilon > 0$ is a pre-determined parameter. This above problem (8) is solving for the least amount of additional data to collect such that with probability at least $1-\epsilon$, we collect above the minimum data requirement. We first characterize the properties of problem (8).

Lemma 1. *Problem* (8) *is a convex optimization problem.*

Proof. We only need to prove that the set $\{q_1 \mid F(q_1) \geq 1-\epsilon\}$ is a convex set, since the objective and remaining constraint are convex. Since $F(q)$ is a monotonically non-decreasing function in q, for any $\theta \in [0,1]$ and $q < \hat{q}$ that satisfy the CDF constraint, we have

$$F(\theta q + (1-\theta)\hat{q}) \geq F(q) \geq 1 - \epsilon.$$

Because the convex combination of any two points is in the set, the set must be convex. □

Lemma 2. *The optimal solution to problem* (8) *is*

$$q^* = \begin{cases} F^{-1}(1-\epsilon), & \textit{if } F(q) < 1-\epsilon \\ q_0 & \textit{otherwise} \end{cases}$$

Proof. First consider the case where $F(q_0) \geq 1-\epsilon$. Then, q_0 is a feasible solution to problem (8). Furthermore due to the second constraint, any $q < q_0$ is infeasible. Since q_0 minimizes the objective, it is optimal.

Next consider the case where $F(q_0) < 1-\epsilon$. Then, let $q_1 = F^{-1}(1-\epsilon) = \inf\{q \mid F(q) \geq 1-\epsilon\}$ be the smallest solution that satisfies the CDF constraint. By the monotonicity of $F(q)$, it follows that $q_1 > q_0$. Therefore, this solution minimizes the objective. □

We are now ready to prove Theorem 1.

Proof of Theorem 1. We prove this result by first developing two different characterizations of the optimal solution set of problem (5) and then applying their equivalence.

First note that problem (5) is an optimization problem with one variable q_1 over a constrained domain $[q_0, \infty)$. Furthermore, the objective function is continuous everywhere, meaning that there are two possible scenarios:

- q_1^* satisfies $f(q_1) = c/P$.
- $q_1^* := q_0$ and the optimal value is $P(1 - F(q_0))$.

The first scenario is obtained by taking the derivative of the objective function and setting it to 0. The second scenario comes from the boundary condition. Moreover, note that the objective is unbounded as $q_1 \to +\infty$ meaning the above scenario is the only boundary condition that we need to consider.

Next, note that problem (5) is equivalent to the following optimization problem

$$\begin{aligned} \min_{q_1,\epsilon}\quad & c(q_1 - q_0) + P\epsilon \\ \text{s.t.}\quad & \epsilon \geq 1 - F(q_1) \\ & q_1 \geq q_0 \end{aligned} \tag{9}$$

$$\min_{\epsilon} c(F^{-1}(1-\epsilon) - q_0) + P\epsilon \qquad \text{s. t.}\ \epsilon \geq 1 - F(q_0)$$

For any fixed ϵ, problem (9) is equivalent to problem (8), and therefore by Lemma 1, the problem is convex optimization problem.

We can optimize problem (9) by breaking into two cases. First, for any fixed $\epsilon \geq 1 - F(q_0)$, setting $q_1^*(\epsilon) = 0$ attains a feasible solution and mnimizes the objective to $P\epsilon$.

Second, for any fixed $\epsilon \leq 1 - F(q_0)$, Lemma 2 states that $q_1^*(\epsilon) = F^{-1}(1-\epsilon)$ is a corresponding optimal solution and the objective function reduces to

$$c\left(F^{-1}(1-\epsilon) - q_0\right) + P\epsilon.$$

Moreover, from the original formulation (5), we can substitute $q_1^*(\epsilon)$ and obtain $f(F^{-1}(1-\epsilon) = c/P$.

Finally, problem (9) is optimized via the second case if and only if there exists a feasible $\epsilon \leq 1-F(q_0)$ that satisfies

$$c\left(F^{-1}(1-\epsilon) - q_0\right) + P\epsilon \leq P(1 - F(q_0)).$$

We can rewrite this condition as follows. Let $q_1 > q_0$ and assume that (6) holds. Then,

$$\begin{aligned} & c(q_1 - q_0) \leq PF(q_1) - PF(q_0) \\ \Rightarrow \quad & c(q_1 - q_0) - PF(q_1) \leq -PF(q_0) \\ \Rightarrow \quad & c(q_1 - q_0) + P(1 - F(q_1)) \leq P(1 - F(q_0)). \end{aligned}$$

Let $\epsilon = 1 - F(q_1)$. Since $F(q_1) \geq F(q_0)$, we have $\epsilon \leq 1 - F(q_0)$, meaning that there is a feasible $q_1 > q_0$ to problem (9) with lower objective function value than q_0. Thus, assumption (6) guarantees that problem (9) has an optimal solution $q_1^* = F^{-1}(1-\epsilon^*)$ where ϵ^* must satisfy $f(F^{-1}(1-\epsilon^*) = c/P$. Conversely, if (6) is not satisfiable for any $q_1 > q_0$, then we can use the same steps to show that $q_1^* = q_0$ is an optimal solution to the problem. $\square$

D Optimal Data Collection with Multiple Sources

The multi-variate data collection problem considers multiple sources delivering different types of data required to train a model. Consider K data sets with $q^1, \dots, q^K$ points in each, respectively. Rather than collecting up to q_t data points in each round, we optimize a vector $\mathbf{q}_t \in \mathbb{R}_+^K$ where each element q_t^k refers to how much data we need from the k-th source. Furthermore, the minimum data requirement is now a vector $\mathbf{D}^*$.

This problem can be solved using the same general approach outlined in Algorithm 1, but with two changes. First, we require a multi-variate version of the PDF and CDF of $\mathbf{D}^*$. This necessitates new neural scaling law regression models for dealing with multiple data sources. Second, we modify the optimization problem from Appendix B to accommodate decision vectors. We highlight the above two steps in this section.

D.1 A General Multi-variate Neural Scaling Law

In order to construct a PDF and CDF in the multi-variate setting, we follow the same general steps as in Algorithm 1. We first collect a data set of performance statistics $\mathcal{R} := \{(\mathbf{q}_r, V(\mathcal{D}^1_{q^1_r}, \mathcal{D}^2_{q^2_r}, \cdots, \mathcal{D}^K_{q^K_r})\}_{r=1}^R$ as before. We then use bootstrap resamples of this data set to fit parameters $\boldsymbol{\theta}^*$ to a regression model $\hat{v}(q^1, \dots, q^K; \boldsymbol{\theta})$ and then solve for

$$\hat{\mathbf{q}} := \arg\min_{\mathbf{q}} \{\mathbf{c}^\top \mathbf{q} \mid \hat{v}(q^1, \dots, q^K; \boldsymbol{\theta}^*) \geq V^*\}.$$

Finally, we fit a density estimation model over our data set of $\hat{\mathbf{q}}$.

The key challenge to this approach however is in designing a multi-variate regression function. To the best of our knowledge, the neural scaling law literature has not explored general power law models that can accommodate K different types of data for arbitrary tasks.

We propose an easy-to-implement baseline regression model by adding the contributions of each data set being used. Then, our additive regression model is

$$\hat{v}(q^1, \ldots, q^K; \boldsymbol{\theta}) := \sum_{k=1}^{K} \hat{v}_k(q^k; \boldsymbol{\theta}_k)$$

where $\hat{v}_k(q_k; \boldsymbol{\theta}_k)$ can be any single-variate regression model for estimating score. For instance, consider $K = 2$ data types with power law regression models for each data type. Our multi-variate regression model becomes

$$\hat{v}(q^1, q^2; \boldsymbol{\theta}) = \theta_{1,0}(q^1)^{\theta_{1,1}} + \theta_{2,0}(q^2)^{\theta_{2,1}} + \theta_3.$$

We may also incorporate other base models in the same way, such as the logarithmic or arctan functions introduced in Mahmood et al. [2].

The benefit of this additive regression model is that we can easily fit it via least squares minimization using a regression data set $\mathcal{R} := \{(q_r^1, q_r^2, V(\mathcal{D}^1_{q_r^1}, \mathcal{D}^2_{q_r^2})\}_{r=1}^R$. Furthermore, additive models are simple and offer interpretable explanations on the contributions of each data type to performance by assuming that each data set has an independent effect. Finally, additive models are common in many other tasks, such as when estimating the valuation of specific data points [52].

We remark that some recent research has explored neural scaling laws for specific tasks with multiple data types. For instance, Mikami et al. [11] explore a $K = 2$ power law for transfer learning from synthetic to real domains, where they use a multiplicative component that captures an interaction between real and synthetic data sets. Because scaling laws in multi-variate settings remain an open area of study, if there exist specific structural regression functions for a given application with different types of data, then such functions should be used in place of the additive model. Moreover, our downstream optimization model operates independently of the regression model, as long as the regression model can be re-trained with bootstrap samples in order to facilitate density estimation.

D.2 The Optimization Problem with Multiple Decisions

Just as in the single-variate case, problem (7) has a differential objective function but a series of lower bound constraints. We can use the same approach highlighted in Appendix B to reformulate this problem and remove the constraints. We summarize this reformulation below.

For each t, let $\mathbf{d}_t = \mathbf{q}_t - \mathbf{q}_{t-1}$ be the additional data collected in each round. Then, we recursively re-define $\mathbf{q}_t = \mathbf{q}_0 + \sum_{r=1}^{t} \mathbf{d}_r$ and re-write the problem to

$$\begin{aligned} \min_{\mathbf{d}_1, \cdots \mathbf{d}_T} \quad & \mathbf{c}^\mathsf{T} \sum_{t=1}^{T} \mathbf{d}_t \left(1 - F\left(\mathbf{q}_0 + \sum_{r=1}^{t-1} \mathbf{d}_r\right)\right) + P\left(1 - F\left(\mathbf{q}_0 + \sum_{r=1}^{T} \mathbf{d}_r\right)\right) \\ \text{s.t.} \quad & \mathbf{d}_1, \ldots, \mathbf{d}_T \geq 0. \end{aligned}$$

The above problem can now be solved using off-the-shelf gradient descent.

E Simulation Experiment Setup

The most intuitive approach of validating our data collection problem is by repeatedly sampling from a data set, training a model, and solving the optimization problem. However, performing a large set of such experiments over many data sets becomes computationally intractable. Instead, we follow the approach introduced in Mahmood et al. [2], which proposes a simulation model of the data collection problem. This section summarizes the simulation setup.

The simulation replicates the steps in Algorithm 1 except with one key difference. In the simulation, we replace the score function $V(\mathcal{D})$ with a *ground truth* function $v_{\text{gt}}(q)$ that serves as an oracle which reports the expected score of the model trained with q data points. Thus, rather than having to collect data and train a model in each round, we evaluate $v_{\text{gt}}(q_t)$ and treat this as the current model score. The optimization and regression models do not have access to $v_{\text{gt}}(q)$.

E.1 A Piecewise-Linear Ground Truth Approximation

In order to build a ground truth function, we first use the sub-sampling procedure in Algorithm 1 to collect performance statistics over subsets of the entire training data set. Using these observed statistics, we then build a piecewise-linear model of the ground truth. Below, we first highlight how to construct a piecewise-linear model when given a set of data set sizes and their corresponding scores. In the next subsection, we will detail the exact data collection process.

The Single-variate ($K = 1$) Case. Mahmood et al. [2] develop a ground truth function as follows. Let $q_0 \leq q_1 \leq q_2 \leq \cdots$ be a series of data set sizes and let $\mathcal{D}_{q_0} \subset \mathcal{D}_{q_1} \subset \mathcal{D}_{q_2} \subset \cdots$ be their corresponding sets. Then, consider the following piecewise-linear function:

$$v_{\text{gt}}(q) := \begin{cases} \dfrac{V(\mathcal{D}_{q_0})}{q_0} n, & q \leq q_0 \\ \dfrac{V(\mathcal{D}_{q_t}) - V(\mathcal{D}_{q_{t-1}})}{q_t - q_{t-1}} (q - q_t) + V(\mathcal{D}_{q_{t-1}}), & q_{t-1} \leq q \leq q_t \end{cases}$$

This function is concave and monotonically increasing, which follows the general trend of real learning curves [5]. Furthermore Mahmood et al. [2] show that given sufficient resolution, i.e., enough data subsets, this piecewise linear function is an accurate approximation of the true learning curve $V(\mathcal{D})$.

The Multi-variate ($K = 2$) **Case.** In the previous $K = 1$ case, the ground truth was formed by taking linear approximations between different subset sizes. When $K > 1$, we have multiple subsets that are used to evaluate the score $V(\mathcal{D}^1, \dots, \mathcal{D}^K)$.

We focus specifically on $K = 2$ in our numerical experiments and propose a generalization of the previous piecewise-linear function. Here, rather than building lines on the intervals between subsequent sets, we build planes on triangular intervals. Specifically, let $q_0^1 \leq q_1^1 \leq q_2^1 \leq \cdots$ and $q_0^2 \leq q_1^2 \leq q_2^2 \leq \cdots$ be two series of data set sizes, and consider the grid

$$\begin{matrix} (q_0^1, q_0^2) & (q_1^1, q_0^2) & (q_2^1, q_0^2) & \cdots \\ (q_0^1, q_1^2) & (q_1^1, q_1^2) & (q_2^1, q_1^2) & \cdots \\ (q_0^1, q_2^2) & (q_1^1, q_2^2) & (q_2^1, q_2^2) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{matrix}$$

For each tuple (q_s^1, q_t^2) in the above grid, let $V(\mathcal{D}^1_{q_s^1}, \mathcal{D}^2_{q_t^2})$ be the score of a model trained on two data sets of the corresponding respective sizes.

For each index (s, t) we fit linear models on the corresponding lower right and upper left triangles. First, let $(\underline{\alpha}(s,t), \underline{\beta}(s,t), \underline{\gamma}(s,t))$ be the parameters of the plane defined by the lower triangle $\{(q_{s-1}^1, q_t^2), (q_s^1, q_{t-1}^2), (q_s^1, q_t^2)\}$, i.e., the unique solution to the following linear system:

$$\begin{pmatrix} q_{s-1}^1 & q_t^2 & 1 \\ q_s^1 & q_{t-1}^2 & 1 \\ q_s^1 & q_t^2 & 1 \end{pmatrix} \begin{pmatrix} \underline{\alpha}(s,t) \\ \underline{\beta}(s,t) \\ \underline{\gamma}(s,t) \end{pmatrix} = \begin{pmatrix} V(\mathcal{D}^1_{q_{s-1}^1}, \mathcal{D}^2_{q_t^2}) \\ V(\mathcal{D}^1_{q_s^1}, \mathcal{D}^2_{q_{t-1}^2}) \\ V(\mathcal{D}^1_{q_s^1}, \mathcal{D}^2_{q_t^2}) \end{pmatrix}$$

Thus, for any data set sizes (q^1, q^2) in this triangle, we evaluate the ground truth by the linear model $\underline{\alpha}(s,t)q^1 + \underline{\beta}(s,t)q^2 + \underline{\gamma}(s,t)$. Similarly, let $(\overline{\alpha}(s,t), \overline{\beta}(s,t), \overline{\gamma}(s,t))$ be the parameters of the plane defined by the upper triangle $\{(q_s^1, q_t^2), (q_{s+1}^1, q_t^2), (q_s^1, q_{t+1}^2)\}$, i.e., the unique solution to the following linear system:

$$\begin{pmatrix} q_s^1 & q_t^2 & 1 \\ q_{s+1}^1 & q_t^2 & 1 \\ q_s^1 & q_{t+1}^2 & 1 \end{pmatrix} \begin{pmatrix} \overline{\alpha}(s,t) \\ \overline{\beta}(s,t) \\ \overline{\gamma}(s,t) \end{pmatrix} = \begin{pmatrix} V(\mathcal{D}^1_{q_s^1}, \mathcal{D}^2_{q_t^2}) \\ V(\mathcal{D}^1_{q_{s+1}^1}, \mathcal{D}^2_{q_t^2}) \\ V(\mathcal{D}^1_{q_s^1}, \mathcal{D}^2_{q_{t+1}^2}) \end{pmatrix}$$

Similarly for any (q^1, q^2) in this triangle, the ground truth is obtained by the linear model $\overline{\alpha}(s,t)q^1 + \overline{\beta}(s,t)q^2 + \overline{\gamma}(s,t)$.

Finally, we define our ground truth function $v_{\text{gt}}(q^1, q^2)$. For any $q^1 \geq q_0^1, q^2 \geq q_0^2$, this function first identifies the interval $[q_s^1, q_{s+1}^1] \times [q_t^2, q_{t+1}^2]$ in which the point lies. Then, the function assigns a

Data set	Task	Score	Full data set size	
CIFAR-10 [33]	Classification	Accuracy	$50,000$	
CIFAR-100 [33]	Classification	Accuracy	$50,000$	
ImageNet [34]	Classification	Accuracy	$1,281,167$	
BDD100K [37]	Semantic Segmentation	Mean IoU	$7,000$	
nuScenes [38]	BEV Segmentation	Mean IoU	$28,130$	
VOC [40, 41]	2-D Object Detection	Mean AP	$16,551$	
CIFAR-100 [33]	Classification	Accuracy	$25,000$ (Classes 0-49)	$25,000$ (Classes 50-99)
BDD100K [37]	Semantic Segmentation	Mean IoU	$7,000$ (Labeled)	$70,000$ (Unlabeled)

Table 4: Data sets, tasks, and score functions considered.

score based on whether the point lies in the upper left or the lower right triangle in this interval. We write this function as

$$v_{\text{gt}}(q^1, q^2) := \begin{cases} \overline{\alpha}(s,t)q^1 + \overline{\beta}(s,t)q^2 + \overline{\gamma}(s,t) \\ \qquad \text{if } \left\|(q^1,q^2) - (q^1_s, q^2_t)\right\| \le \left\|(q^1,q^2) - (q^1_{s+1}, q^2_{t+1})\right\|, \\ \underline{\alpha}(s+1,t+1)q^1 + \underline{\beta}(s+1,t+1)q^2 + \underline{\gamma}(s+1,t+1) \\ \qquad \text{otherwise,} \end{cases}$$
$$\text{for } q^1_s \le q^1 \le q^1_{s+1}\ ,\ q^2_t \le q^2 \le q^2_{t+1}.$$

For $K > 2$. The piecewise linear approximations grow increasingly complex as the dimension K increases. Furthermore, the number of subsets of data set sizes required to create a piecewise linear approximation increases exponentially with K. Specifically for $k \in \{1, \dots, K\}$, let M_k denote the number of subsets (i.e., $|\{q^k_0, q^k_1, \dots, q^k_{M_k}\}|$) of a data set that we consider when creating subsets. For each combination of K subsets, we must then train a model and evaluate it's performance to record $V(\mathcal{D}^1, \dots, \mathcal{D}^K)$. Thus, we must subsample and train our model for $O(\prod_k M_k)$ combinations. Since this quickly becomes computationally prohibitive, we avoid experiments for $K > 2$.

E.2 Data Collection

We now summarize the data collection and training process used to create the above piecewise-linear functions for each data set and task. All models were implemented using PyTorch and trained on machines with up to eight NVIDIA V100 GPU cards. Table 4 details each task and data set size.

Image Classification Tasks. For all experiments with CIFAR-10 and CIFAR-100, we use a ResNet18 [35] following the same procedure as in Coleman et al. [53]. For ImageNet, we use a ResNet34 [35] using the procedure in Coleman et al. [53]. All models are trained with cross entropy loss using SGD with momentum. We evaluate all models on Top-1 Accuracy.

For all experiments, we set the initial data set at $q_0 = 10\%$ of the data. In data collection, we create five subsets containing $2\%, 4\%, \cdots, 10\%$ of the training data, five subsets containing $12\%, 14\%, \cdots, 20\%$ of the training data, and eight subsets containing $30\%, 40\%, \cdots, 100\%$ of the data. Note that we use higher granularity in the early stage as this is where the dynamics of the learning curve vary the most. With more data, the learning curve eventually has a nearly zero slope. For each subset, we train our respective model and evaluate performance.

VOC. We use the Single-Shot Detector 300 (SSD300) [42] based on a VGG16 backbone [54], following the same procedure as in Elezi et al. [55]. All models are trained using SGD with momentum. We evaluate all models on mean AP.

For all experiments, we set the initial data set at $q_0 = 10\%$ of the data. In data collection, we sample twenty subsets at 5% intervals, i.e., $5\%, 10\%, 15\%, \cdots, 100\%$ of the training data.

BDD100K. We use Deeplabv3 [36] with ResNet50 backbone. We use random initialization for the backbone. We use the original data set split from Yu et al. [37] with $7,000$ and $1,000$ data points in the train and validation sets respectively. The evaluation metrics is mean Intersection over Union (IoU). We follow the same protocol used in the Image classification tasks to create our subsets of data.

nuScenes. We use the "Lift Splat" architecture [39], which is used for BEV segmentation from driving scenes, following the steps from the original paper to train this model. We evaluate on mean

Parameter	Setting
Optimizer	GD with Momentum ($\beta = 0.9$), Adam ($\beta_0, \beta_1 = 0.9, 0.999$)
Learning rate	$0.005, \ldots, 500$
Number of bootstrap samples B	500
Number of regression subsets R	See Appendix E.2
Density Estimation Model	KDE for $K = 1$, GMM for $K = 2$
KDE Bandwidth	$20000, \ldots, 20000000$ for ImageNet $200, \ldots, 4000$ for all others
GMM number of clusters	$4, \ldots, 10$

Table 5: Summary of hyperparameters used in our experiments.

IoU. Our data collection procedure follows the same steps and percentages of the data set as used for BDD100K and the Image classification tasks.

CIFAR-100 (2 Types). We partition this data set into two subsets $\mathcal{D}^1$ and $\mathcal{D}^2$ of $25,000$ images each containing the first 50 and last 50 classes, respectively. We then train a ResNet18 [35] using different fractions of the two subsets. We follow the same training procedure as in the single-variate case except with one difference. Since some of the data sets will naturally be imbalanced (e.g., if we train with half of the first subset and all of the second subset), we employ a class-balanced cross entropy loss using the inverse frequency of samples per class.

For each $\mathcal{D}^k$ subsets, respectively, we follow the same subsampling procedure used in the single-variate case. That is, we let $q_0^1 = 10\%$ of the first data subset and $q_0^2 = 10\%$ of the second data subset. For each subset, we create 10 subsampled sets at intervals of $2\%, 4\%, 6\%, \cdots, 20\%$ of the respective data subset. We then create eight further subsampled sets at $30\%, 40\%, \cdots, 100\%$ of the respective data subset. Finally, we train our model and evaluate the score on every combination of the subsampled subsets of $\mathcal{D}^1 \times \mathcal{D}^2$.

BDD100K (Semi-supervised). For this task, we consider semi-supervised segmentation via pseudo-labeling the unlabeled data set in BDD100K. The data is partitioned into two subsets $\mathcal{D}^1$ and $\mathcal{D}^2$ containing $7,000$ labeled and $70,000$ unlabeled scenes. As before, we use the Deeplabv3 [36] architecture with a ResNet50 backbone. Here however, we:

1. First train with a labeled subset of $\mathcal{D}^1$ via supervised learning.
2. Pseudo-label an unlabeled subset of $\mathcal{D}^2$ using the trained model.
3. Re-train the model with the labeled subset and the pseudo-labeled subset.

We follow the same procedure as in the single-variate case for both training steps, except we weigh the unlabeled data by 0.2 to reduce it's contribution to the loss.

Training via semi-supervised learning on BDD100K requires long compute times, so we reduce the number of subsets used in this experiment. For the labeled set $\mathcal{D}^1$, we create subsets with $5\%, 10\%, 15\%, 20\%, 40\%, 60\%, 80\%, 100\%$ of the data. For the unlabeled set $\mathcal{D}^2$, we create subsets with $0\%, 10\%, 25\%, 50\%, 100\%$ of the data. Note that we have five settings of unlabeled data since we include the case of training with no unlabeled data as well.

E.3 LOC Implementation

For all experiments, we initialize with 10% of the training data set. We consider $T = 1, 3, 5$ rounds and sweep a range of V^*. We provide a summary of parameters in Table 5.

For the experiments with $K = 1$, we model the data requirement PDF $f(q)$ in each round of the problem as follows. We first draw $B = 500$ bootstrap resamples of the current training statistics $\mathcal{R}$, where $\mathcal{R} = \{(rq_0/R, V(\mathcal{D}_{rq_0/R}))\}_{r=1}^R \cup \{(q_s, V(\mathcal{D}_{q_s}))\}_{s=1}^t$ contains all of the measured statistics up to the initial data set (e.g., for CIFAR-10, this includes performance with $2\%, 4\%, \cdots, 10\%$ of the data), and the previous collected data. The latter is obtained by calling our piecewise-linear ground truth approximation. For each bootstrap resample, we fit a power regression model $\hat{v}(q; \boldsymbol{\theta}) = \theta_0 q^{\theta_1} + \theta_2$ and solve for the estimated minimum data requirement. We then use our set of estimates to fit a Kernel Density Estimation (KDE) model after gridsearching for the best bandwidth parameter.

For the experiments with $K = 2$, we use the same above procedure but fit Gaussian Mixture Models (GMM) due to their having an easily computable CDF via the Gaussian $\mathrm{erf}(\cdot)$, rather than numerically integrating the PDF. We grid-search over the number of mixture components for the GMM model.

We optimize over problems (4) and (7) using gradient descent techniques. Depending on the current state and data set, different hyperparameters perform better. As a result, we perform extensive hyperparameter tuning every time we need to solve the optimization problem. Here, we sweep over gradient descent with momentum and Adam with learning rates between 0.005 to 500.

We initialize each problem with $\mathbf{q}_t$ equal to the baseline regression solution and $\mathbf{q}_{t+s} = \mathbf{q}_t/(s+1)$ for all $1 \le s \le T - t$. That is, we set the initial value for future collection amounts to be fractions of the initial value of the immediate amount of data to collect. We identified this initialization by manually inspecting the solutions found by LOC, consequently it improves the conditioning of the loss landscape relative to other random initialization schemes.

F Additional Numerical Results

This section contains expanded results of our numerical experiments and further ablations. Our key results include:

- In Appendix F.1, we evaluate the effectiveness of estimating $F(q)$ by plotting the estimated learning curves as well as the empirical histograms used to model the data requirement distribution.
- In Appendix F.2, we explore the sensitivity of our optimization algorithm to variations in the cost and penalty parameters. In all except one instance, LOC consistently maintains a low total cost and failure rate.
- In Appendix F.3, we explore the multi-variate LOC (i.e., $K = 2$) for problems where we have a small number of $T = 1, 3$ rounds. The baseline fails for almost all instances of $T = 1$, whereas LOC maintains a low failure rate.
- In Appendix F.4, we consider variants of LOC where we use different regression functions to estimate the data requirement distribution. Our optimization framework can be deployed on top of any regression function to reduce the failure rate.

F.1 Estimating the Data Requirement Distribution $F(q)$

To estimate $F(q)$, we first create an ensemble of estimated learning curves, which we then invert to obtain an empirical distribution of estimated values for D^*. Figure 5 plots our bootstrap resampled estimated learning curves versus the ground truth performance for the first round of data collection when we have access to an initial $\mathcal{D}_{q_0}$ containing 10% of the full data set. As noted in Mahmood et al. [2], the mean estimated learning curve diverges from the ground truth. However, by bootstrap resampling an ensemble of learning curves, we can cover the ground truth with some probability.

Figure 6 plots the empirical histograms of estimated D^* as well as the estimated $F(q)$ obtained via KDE on CIFAR-10 with three different values for V^*. Although the mode of the estimated distribution is far from the ground truth D^*, the estimated distribution assigns some probability to the ground truth region. LOC optimizes over this estimated $F(q)$, which allows us to conservatively collect data and reduce the chances of failure.

F.2 Robustness to the Cost and Penalty Parameters

Figure 7 expands the cost parameter sweep from Figure 3 (Top row) to the settings of $T = 1, 3$. For nearly all settings, LOC remains stable to variations in the cost parameter. Nonetheless, careful parameter selection becomes important as T decreases. This is due to the fact that for low costs, the total amount of data collected increases as T decreases (e.g., $c = 0.001$ for BDD100K). Furthermore, Figure 8 expands the penalty parameter sweep from Figure 3 (Bottom row). Here, we observe similar properties to the cost parameter sweep.

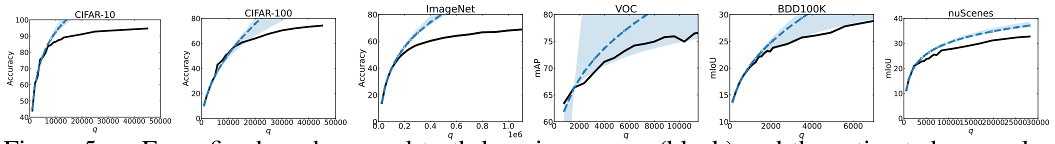
Figure 5: For a fixed seed, ground truth learning curves (black) and the estimated power law learning curves (blue) obtained via bootstrapping and ensembling. The shaded region represents the 95 percentile of the ensemble and the dashed blue line represents the mean of the regression functions. The mean is consistently higher than the unknown ground truth, whereas the shaded region can at times cover it.

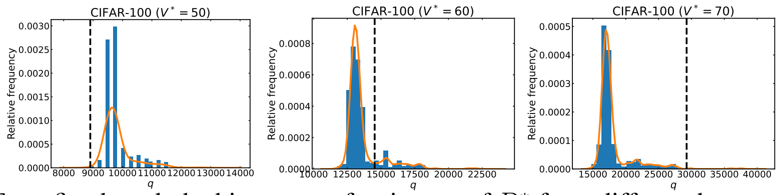
Figure 6: For a fixed seed, the histogram of estimates of D^* from different bootstrapped models (blue bars), the estimated $F(q)$ (orange curve), and the ground truth D^* (black dashed line). Each plot corresponds to a different V^* for CIFAR-100 (see Figure 5 for the learning curve). With higher targets, regression (i.e., collecting the mean of the distribution) will lead to larger under-estimations.

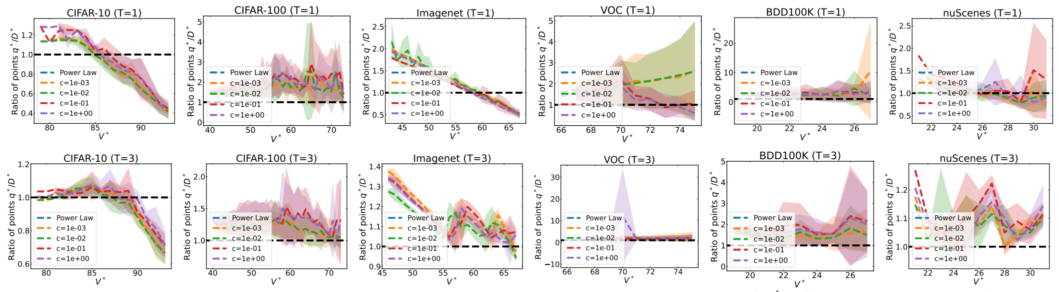
Figure 7: Mean $\pm$ standard deviation of the ratio of data collected q_T^*/D^* for different V^* when we sweep the cost parameter from 0.001 to 1 and fix $P = 10^7$. We show $T = 1, 3$ and refer to the main paper for $T = 5$. The dashed black line corresponds to collecting exactly the minimum data requirement.

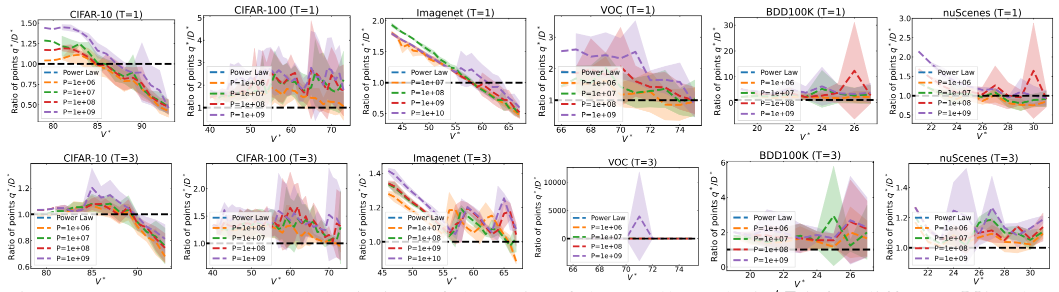
Figure 8: Mean $\pm$ standard deviation of the ratio of data collected q_T^*/D^* for different V^* when we sweep the penalty parameter from 10^6 to 10^9 and fix $c = 1$. We show $T = 1, 3$ and refer to the main paper for $T = 5$. The dashed black line corresponds to collecting exactly the minimum data requirement.

Although LOC is relatively stable on all other data sets, our results demonstrate some extreme results for VOC, potentially due to noise in the simulation. For example in Figure 8, setting $P = 10^9$, $V^* = 71$, and $T = 3$ led to collecting $10,000$ times the minimum data requirement. Such a situation is unrealistic in a production-level implementation, since in a real implementation, we could impose further constraints onto problem (4), such as upper bounds on the total amount of data permissible.

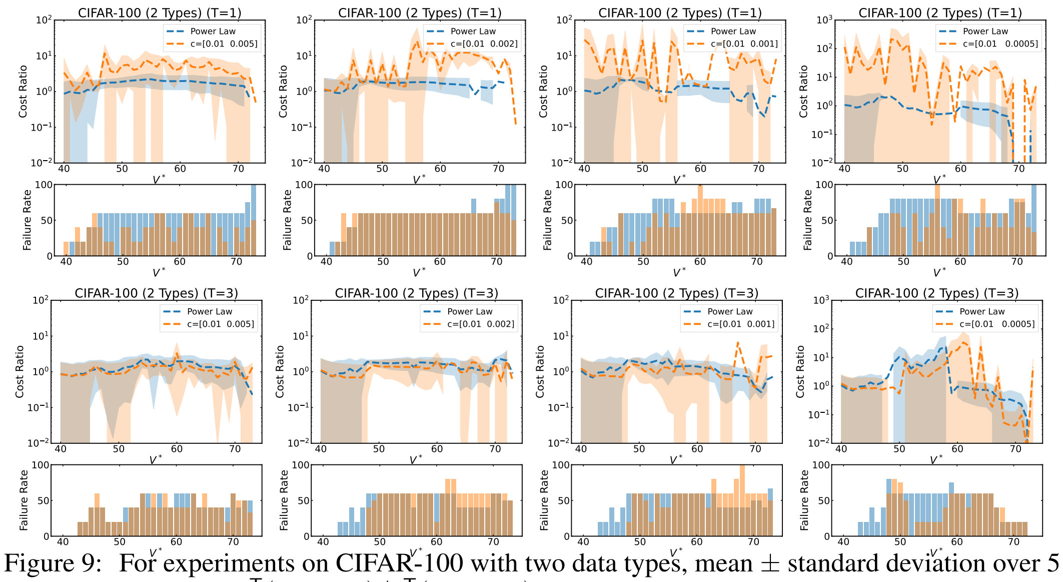
Figure 9: For experiments on CIFAR-100 with two data types, mean $\pm$ standard deviation over 5 seeds of the cost ratio $\mathbf{c}^\top(\mathbf{q}_T^* - \mathbf{q}_0)/\mathbf{c}^\top(\mathbf{D}^* - \mathbf{q}_0) - 1$ and failure rate for different V after removing 99-th percentile outliers. We fix $c_0 = 1$ and $P = 10^{13}$. The rows correspond to $T = 1, 3$ (see the main paper for $T = 5$) and the columns correspond to $c_1 = c_0/2, c_0/5, c_0/10, c_0/20$.

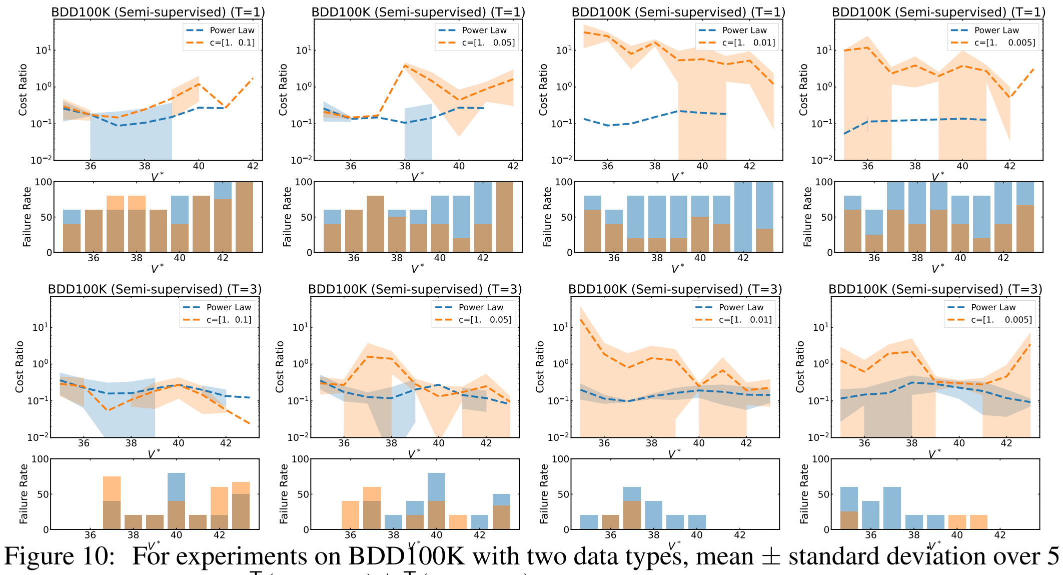
Figure 10: For experiments on BDD100K with two data types, mean $\pm$ standard deviation over 5 seeds of the cost ratio $\mathbf{c}^\top(\mathbf{q}_T^* - \mathbf{q}_0)/\mathbf{c}^\top(\mathbf{D}^* - \mathbf{q}_0) - 1$ and failure rate for different V after removing 99-th percentile outliers. We fix $c_0 = 1$ and $P = 10^{13}$. The rows correspond to $T = 1, 3$ (see the main paper for $T = 5$) and the columns correspond to $c_1 = c_0/2, c_0/5, c_0/10, c_0/20$.

F.3 The Value of Optimization over Estimation when $K = 2$

Figure 9 and Figure 10 expand Figure 4 to $T = 1, 3$ rounds. The results validate the summary observations from Table 2 in that the baseline has considerably higher failure rates versus LOC. In particular for BDD100K at $T = 1$, the baseline fails consistently for four out of five random seeds. On the other hand, recall that LOC admits a higher cost ratio compared to the baseline when $T = 1$. We can observe now that this high cost ratio is due to the method incurring high cost for a few target V^* values. This behavior is similar to the observation above on VOC with high penalties at $T = 3$.

Regression Function	T	Regression		LOC	
		Failure rate	Cost ratio	Failure rate	Cost ratio
Logarithmic $\hat{v}(q;\boldsymbol{\theta}) = \theta_0 \log(q+\theta_1) + \theta_2$	1	43%	0.19	**2%**	1.17
	3	37%	0.17	**2%**	0.54
	5	34%	0.16	**1%**	0.39
Arctan $\hat{v}(q;\boldsymbol{\theta}) = \frac{200}{\pi} \arctan(\theta_0 \frac{\pi}{2} q + \theta_1) + \theta_2$	1	23%	3.31	**0%**	5.56
	3	15%	3.01	**0%**	3.92
	5	12%	2.90	**0%**	3.60
Algebraic Root $\hat{v}(q;\boldsymbol{\theta}) = \frac{100q}{1+\lvert\theta_0 q\rvert^{\theta_1})^{1/\theta_1}} + \theta_2$	1	52%	0.11	**23%**	0.81
	3	44%	0.1	**2%**	0.87
	5	44%	0.1	**2%**	0.54

Table 6: For experiments on CIFAR-100, average cost ratio $\mathbf{c}^\mathsf{T}(\mathbf{q}_T^* - \mathbf{q}_0)/\mathbf{c}^\mathsf{T}(\mathbf{D}^* - \mathbf{q}_0) - 1$ and failure rate measured over a range of V^* and T. We fix $c = 1$ and $P = 10^7$. The best performing failure rate for each setting is bolded. The cost ratio is measured only for instances that achieve V^*. LOC consistently reduces the average failure rate, almost consistently down to 0%.

	Data set	T	Regression With Correction [2]		LOC	
			Failure rate	Cost ratio	Failure rate	Cost ratio
Class.	CIFAR-100	1	14%	<u>0.94</u>	**4%**	0.99
		3	**1%**	<u>0.23</u>	3%	0.31
		5	**0%**	<u>0.17</u>	2%	0.19
	Imagenet	1	**7%**	1.03	37%	<u>0.49</u>
		3	**0%**	0.21	5%	<u>0.16</u>
		5	**0%**	0.14	2%	<u>0.10</u>
Seg.	BDD100K	1	**4%**	4.03	12%	<u>2.03</u>
		3	**0%**	1.02	**0%**	<u>0.72</u>
		5	**0%**	0.62	**0%**	<u>0.35</u>
	nuScenes	1	**0%**	27.2	52%	<u>0.16</u>
		3	**0%**	0.75	**0%**	<u>0.09</u>
		5	**0%**	0.30	**0%**	<u>0.04</u>
Det.	VOC	1	**0%**	44.6	25%	<u>0.56</u>
		3	**0%**	7.02	**0%**	<u>1.10</u>
		5	**0%**	3.98	**0%**	<u>0.84</u>

Table 7: Comparison against the correction factor-based Power Law Regression of Mahmood et al. [2] using the same setup as in Table 1. The best performing cost ratio is underlined and the best performing failure rate for each setting is bolded. Although the baseline is designed specifically to achieve low failure rates, LOC often can achieve competitive failure rates while reducing the cost ratios by an order of magnitude.

F.4 LOC with Alternative Regression Functions

Mahmood et al. [2] show that we can use other regression functions instead of the power law to estimate the data requirement. Moreover, some functions tend to consistently over- or under-estimate the requirement. LOC can be deployed on top of any such regression function, since the regression function is only used to generate bootstrap samples.

Table 6 highlights experiments on CIFAR-100 with three alternative regression functions that were used by Mahmood et al. [2]. For both functions, we observe the same trends seen in Table 1. That is, LOC reduces the failure rate down to approximately zero, at a marginal relative increase in cost.

Noting that Power Law Regression often leads to failure, Mahmood et al. [2] also propose a correction factor heuristic wherein they learn a parameter τ such that if the data collection problem requires a target performance V^*, we should instead aim to collect enough data to meet $V^* + \tau$. In order to learn this correction factor, we require a pre-existing data set upon which we can simulate a data collection policy. Mahmood et al. [2] suggest setting τ such that we can achieve the data requirement V^* for any V^* on the pre-existing data set, and then fixing this parameter for new data sets.

Table 7 compares LOC (i.e., repeating Table 1) with the Correction factor-based Power Law regression baseline of Mahmood et al. [2]. Following the original paper, we tune τ using CIFAR-10 and apply it on all other data sets. The correction factor is designed to minimize the failure rate and thus, achieves nearly 0% failure rate for all settings, but often at high cost ratios. On the other hand, LOC achieves generally low failure rates and low cost ratios. Specifically, for $T = 3, 5$, we are competitive with the baseline on failure rates for most tasks while obtaining up to an order of magnitude decrease in costs. For $T = 1$, we typically admit higher failure rates; however for the segmentation and detection tasks, we obtain up multiple orders of magnitude lower costs. Finally, note that this baseline requires a similar prior task to be effective. For example, the baseline outperforms us on cost and failure rate both only on CIFAR-100, since it is tuned on CIFAR-10. On the other hand, LOC does not require this prior data set to be effective as evidence by its performance on non-classification tasks.