Supplementary Content

A. Experiment setup

We first summarize the regression problem before detailing the data collection and training process for each data set and task. All models were implemented using PyTorch and trained on machines with up to eight NVIDIA V100 GPU cards.

We fit each regression function by minimizing a least squares problem using the Levenberg-Marquardt algorithm as implemented by Scipy [27, 37]. 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 n.

Image classification tasks. For all experiments with CIFAR10 and CIFAR100, we use a ResNet18 [16] following the same procedure as in [8]. For ImageNet, we use a ResNet34 [16] using the procedure in [8]. All models are trained with cross entropy loss using SGD with momentum. We evaluate all models on Top-1 Accuracy.

For all experiments, we first create 10 subsets $\mathcal{S}_0 \subset \mathcal{S}_1 \subset \mathcal{S}_2 \subset \cdots \subset \mathcal{S}_9 = \mathcal{D}_0$ containing $2\%, 4\%, 6\%, \ldots 20\%$ of the training data set, respectively. For example on CIFAR10, $\mathcal{S}_0$ contains 1000 images, $\mathcal{S}_1$ contains 2000 images, and so on. This data is used to build our initial regression models. Thus, when we use $n_0 = 10\%$ of the training data, our initial regression data contains five points evaluating the score from training with $1000, 2000, \ldots, 5000$ images. For evaluation, we sample $\mathcal{D}_1 \subset \mathcal{D}_2 \subset \mathcal{D}_3 \subset \cdots \subset \mathcal{D}_8$ containing $30\%, 40\%, 50\%, \ldots, 100\%$ of the training data set, respectively. In regression, we evaluate our estimators on predicting $V_f(\mathcal{D}_i)$ for each of these data sets.

VOC. We use the Single-Shot Detector 300 (SSD300) [25] based on a VGG16 backbone [33], following the same procedure as in [10]. All models are trained using SGD with momentum. We evaluate all models on mean AP.

For all experiments, we create 8 regression subsets $\mathcal{S}_0 \subset \mathcal{S}_1 \subset \mathcal{S}_2 \subset \cdots \subset \mathcal{S}_7 = \mathcal{D}_0$ containing approximately $2.5\%, 5\%, 7.5\%, \ldots 20\%$ of the training data set, meaning $n_0 = 10\%$ of the data corresponds to an initial regression data set of four points. For evaluation, we sample $\mathcal{D}_1 \subset \mathcal{D}_2 \subset \mathcal{D}_3 \subset \cdots \subset \mathcal{D}_8$ containing approximately $30\%, 40\%, 50\%, \ldots, 100\%$ of the full training data, respectively.

nuSenes (Detection). We use the FCOS3D network architecture [38], which received first place in the NeurIPS 2020 nuScenes 3-D detection challenge. We follow the same procedure from the original paper for training using SGD. We evaluate on mean AP.

Because training this 3-D detector is computationally expensive, we only use a small number of points for these experiments. We first create initial subsets $\mathcal{S}_0 \subset \mathcal{S}_1 \subset \mathcal{S}_2 \subset \mathcal{S}_3$ containing $5\%, 10\%, 15\%, 20\%$ of the training data set, respectively. For evaluation, we sample $\mathcal{D}_1 \subset \mathcal{D}_2 \subset \mathcal{D}_3 \subset \mathcal{D}_4$ containing $25\%, 50\%, 75\%, 100\%$ of the training data set, respectively.

BDD100K. We use Deeplabv3 [6] with ResNet50 backbone. We use random initialization for the backbone. We use the original dataset split from [40] with 7k train and 1k validation set. The evaluation metrics is mean Intersection over Union (IoU).

For all experiments, we first create 10 regression subsets $\mathcal{S}_0 \subset \mathcal{S}_1 \subset \mathcal{S}_2 \subset \cdots \subset \mathcal{S}_9 = \mathcal{D}_0$ containing $2\%, 4\%, 6\%, \ldots, 20\%$ of the training data set, respectively. For evaluation, we sample $\mathcal{D}_1 \subset \mathcal{D}_2 \subset \mathcal{D}_3 \subset \cdots \subset \mathcal{D}_8$ containing $30\%, 40\%, 50\%, \ldots, 100\%$ of the training data set, respectively.

nuScenes (Segmentation). We use the "Lift Splat" architecture [28], which is used for BEV segmentation from driving scenes, following the steps from the original paper to train this model. We evaluate on mean IoU. Our data collection procedure follows the same steps and percentages of the data set as used for BDD100K.

B. Further regression analysis

In this section, we provide regression plots visualizing each of the functions from Table 1 as well as error measurements using the log of the prediction ratio, which is an alternative metric to RMSE. This analysis supports the three challenges observed in Section 3.2 and reinforces the necessity of our simulation analysis.

Visualizing the regression models. Figure 6 plots the regression functions versus the ground truth for each of the regression experiments in Section 4.2. These results support the RMSE errors given in Table 6. That is, when $n_0 = 50\%$ of the full data set, all of the functions are close to the ground truth curve, but when $n_0 = 10\%$, the regression functions can diverge significantly. Moreover, the Arctan function is often the closest function to the ground truth, as reflected by RMSE.

We observe that the regression functions are almost always either optimistic (*i.e.* their curve is above the ground truth) or pessimistic (*i.e.* their curve is below) over the entire range of the data set size. In particular, the Arctan function is one of the two most pessimistic estimators for $19/21$ plots, Algebraic Root for $13/21$ plots, Logarithm for $7/21$ plots, and Power Law

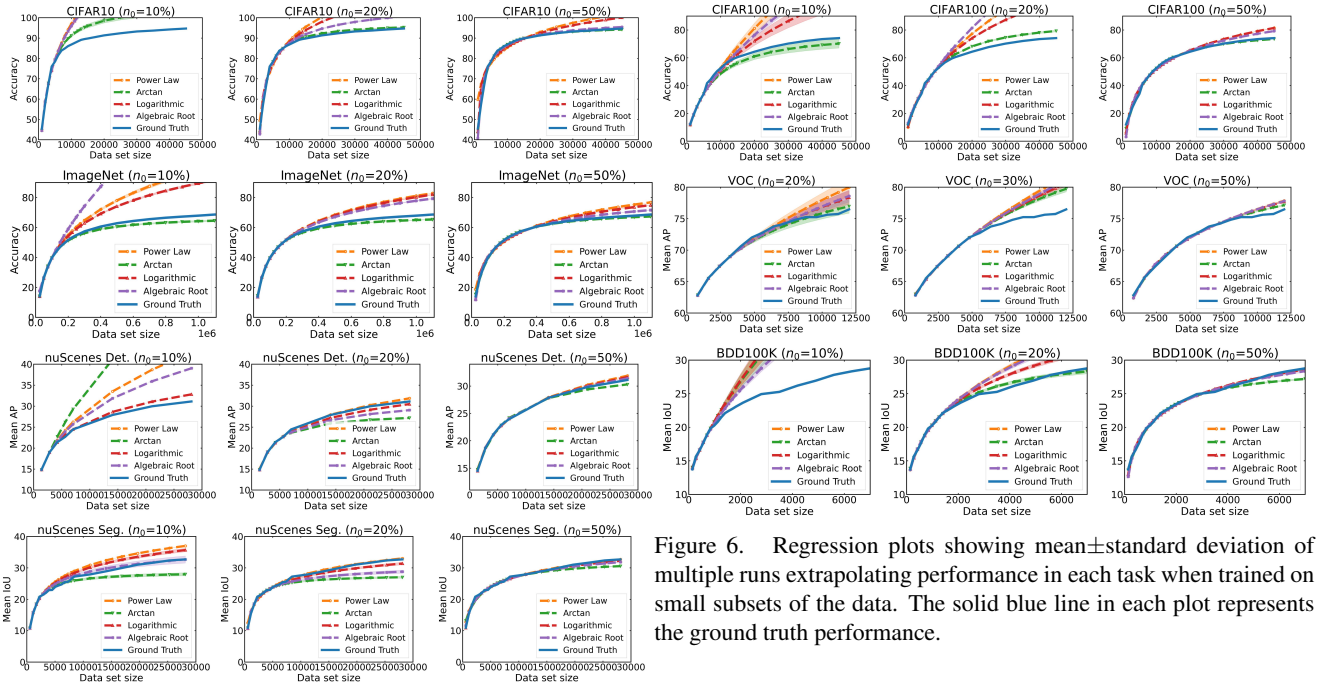

Figure 6. Regression plots showing mean±standard deviation of multiple runs extrapolating performance in each task when trained on small subsets of the data. The solid blue line in each plot represents the ground truth performance.

	Data set	n_0	r	Power Law	Arctan	Logarithmic	Algebraic Root
Classification	CIFAR10	10%	5	0.19 ± 0.1	$\mathbf{0.06 \pm 0.1}$	0.17 ± 0.1	0.17 ± 0.1
	CIFAR10	20%	10	0.1 ± 0.0	$\mathbf{0.01 \pm 0.0}$	0.08 ± 0.0	0.04 ± 0.0
	CIFAR10	50%	17	0.06 ± 0.0	-0.01 ± 0.0	0.04 ± 0.0	$\mathbf{0.01 \pm 0.0}$
	CIFAR100	10%	5	0.1 ± 0.2	-0.12 ± 0.1	$\mathbf{0.01 \pm 0.2}$	0.06 ± 0.1
	CIFAR100	20%	10	0.21 ± 0.0	$\mathbf{0.05 \pm 0.0}$	0.15 ± 0.0	0.27 ± 0.0
	CIFAR100	50%	17	0.07 ± 0.0	-0.01 ± 0.0	0.07 ± 0.0	$\mathbf{0.05 \pm 0.0}$
	ImageNet	10%	4	0.18 ± 0.1	-0.03 ± 0.0	$\mathbf{0.14 \pm 0.0}$	0.38 ± 0.0
	ImageNet	20%	8	$\mathbf{0.1 \pm 0.0}$	-0.03 ± 0.0	0.09 ± 0.0	0.07 ± 0.0
	ImageNet	50%	15	0.07 ± 0.0	-0.01 ± 0.0	0.05 ± 0.0	$\mathbf{0.02 \pm 0.0}$
Detection	VOC	20%	4	0.02 ± 0.0	-0.0 ± 0.0	$\mathbf{0.01 \pm 0.0}$	$\mathbf{0.01 \pm 0.0}$
	VOC	30%	6	0.04 ± 0.0	$\mathbf{0.02 \pm 0.0}$	0.03 ± 0.0	0.03 ± 0.0
	VOC	50%	10	$\mathbf{0.01 \pm 0.0}$	$\mathbf{0.01 \pm 0.0}$	$\mathbf{0.01 \pm 0.0}$	$\mathbf{0.01 \pm 0.0}$
	nuScenes	10%	2	0.15 ± 0.0	0.3 ± 0.0	$\mathbf{0.02 \pm 0.0}$	0.11 ± 0.0
	nuScenes	20%	4	0.06 ± 0.1	-0.03 ± 0.1	0.04 ± 0.1	$\mathbf{0.03 \pm 0.1}$
	nuScenes	50%	6	0.02 ± 0.0	-0.02 ± 0.0	$\mathbf{0.01 \pm 0.0}$	$\mathbf{0.01 \pm 0.0}$
Segmentation	BDD100K	10%	5	0.2 ± 0.1	$\mathbf{0.17 \pm 0.2}$	0.19 ± 0.2	0.14 ± 0.1
	BDD100K	20%	10	0.08 ± 0.0	$\mathbf{0.01 \pm 0.0}$	0.05 ± 0.0	0.08 ± 0.0
	BDD100K	50%	15	0.05 ± 0.0	-0.03 ± 0.0	$\mathbf{0.03 \pm 0.0}$	0.04 ± 0.0
	nuScenes	10%	5	0.09 ± 0.0	-0.04 ± 0.0	0.07 ± 0.0	$\mathbf{0.03 \pm 0.1}$
	nuScenes	20%	10	$\mathbf{0.01 \pm 0.0}$	-0.1 ± 0.0	-0.02 ± 0.0	-0.07 ± 0.0
	nuScenes	50%	15	$\mathbf{0.01 \pm 0.0}$	-0.08 ± 0.0	-0.01 ± 0.0	0.03 ± 0.1

Table 5. Mean±standard deviation of multiple runs evaluating the mean log relative ratio for extrapolating performance in each task when trained on small subsets of the data. We report n_0 in terms of the percentage of the true data set. The lowest error (*i.e.* smallest positive value) for each setting is bolded. Given 50% of the data, there is always at least one regression function achieving a log ratio less than 0.03, whereas with $n_0 = 10\%$ of the data, we may achieve ratios as low as -0.12 or as high as 0.2 (*i.e.* an order of magnitude higher error compared to performance with $n_0 = 50\%$).

for $0/21$. This leads us to conclude that Arctan is generally a pessimistic estimator whereas Logarithmic and Power Law are generally optimistic. Algebraic Root lies in the middle. We hypothesize that this is because when θ_3 is held constant, both Arctan and Algebraic Root converge to a finite value as $n \to +\infty$ (*i.e.* they flatten), but Power Law and Logarithm are unbounded.

Finally, we remark on the shape of the ground truth curves. Recall from Section 3.2 that we observe the ground truth score function $v(n)$ to be piecewise linear, concave, and monotonically increasing. Figure 6 shows that this observation generally holds for CIFAR10, CIFAR100, ImageNet, and the two nuScenes tasks. However for VOC and BDD100K, we observe that the curves are not always concave and monotonically increasing. Nonetheless, our observations about using regression to evaluate data requirement estimation persists. Moreover, our proposed techniques for estimating data requirements remain effective, emphasizing that the initial observation of $v(n)$ from Section 3.2 is not a theoretical requirement but a motivating

Architecture	n_0	r	Power Law	Arctan	Logarithmic	Algebraic Root
ResNet18 (baseline)	10%	5	34.38 ± 35.1	$\mathbf{13.3 \pm 5.3}$	17.25 ± 21.8	26.29 ± 16.8
ResNet18 (baseline)	20%	10	29.52 ± 3.9	$\mathbf{4.71 \pm 2.0}$	19.87 ± 2.5	40.33 ± 1.5
ResNet18 (baseline)	50%	17	5.49 ± 0.2	$\mathbf{0.69 \pm 0.2}$	5.42 ± 0.2	3.65 ± 0.3
ResNet50	10%	5	275.98 ± 521.1	44.94 ± 41.7	49.43 ± 72.1	$\mathbf{24.01 \pm 7.1}$
ResNet50	20%	10	31.04 ± 13.1	$\mathbf{4.14 \pm 4.4}$	21.48 ± 8.3	37.73 ± 3.4
ResNet50	50%	17	6.57 ± 1.0	$\mathbf{1.07 \pm 0.8}$	6.71 ± 1.4	4.8 ± 1.7
ResNet101	10%	5	88.47 ± 115.3	$\mathbf{25.53 \pm 17.1}$	46.73 ± 60.9	26.96 ± 4.5
ResNet101	20%	10	47.34 ± 12.1	$\mathbf{10.23 \pm 4.7}$	32.35 ± 8.5	40.89 ± 1.5
ResNet101	50%	17	8.07 ± 0.3	$\mathbf{0.78 \pm 0.3}$	7.95 ± 0.3	5.57 ± 0.5
WideResNet-16-4	10%	5	34.38 ± 35.1	$\mathbf{13.3 \pm 5.3}$	17.25 ± 21.8	26.29 ± 16.8
WideResNet-16-4	20%	10	13.78 ± 1.8	$\mathbf{0.99 \pm 0.1}$	12.86 ± 0.6	12.55 ± 1.8
WideResNet-16-4	50%	17	5.35 ± 0.3	$\mathbf{1.61 \pm 0.4}$	4.24 ± 0.3	1.55 ± 0.5
WideResNet-16-8	10%	5	57.7 ± 19.3	$\mathbf{5.0 \pm 3.8}$	33.34 ± 10.2	65.31 ± 2.2
WideResNet-16-8	20%	10	14.01 ± 1.8	$\mathbf{1.04 \pm 0.3}$	13.62 ± 1.2	12.74 ± 1.7
WideResNet-16-8	50%	17	5.57 ± 0.2	1.87 ± 0.1	4.32 ± 0.2	$\mathbf{1.4 \pm 0.2}$

Table 6. For different architectures with CIFAR100, mean±standard deviation of multiple runs evaluating the RMSE on extrapolating performance when trained on small subsets of the data. The lowest error for each architecture is bolded. These results reinforce the initial results for CIFAR100 in Table 3, as the Arctan function consistently dominates in nearly every setting.

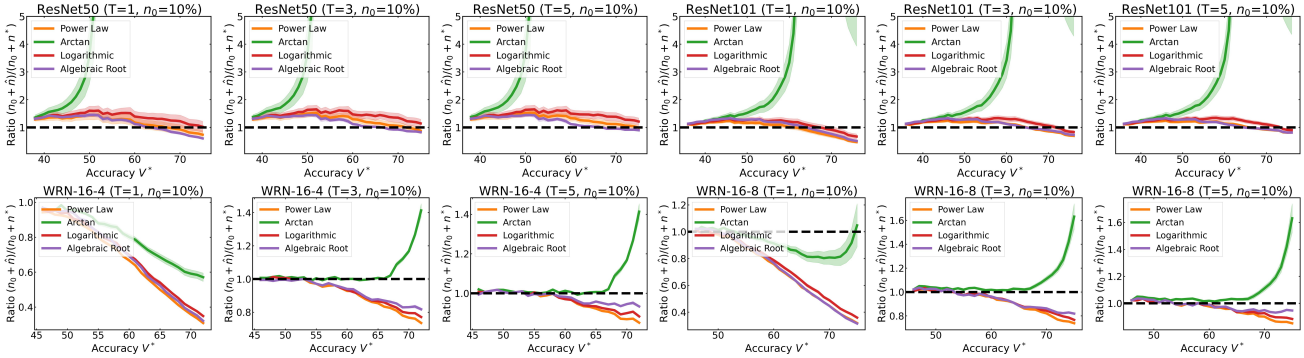

Figure 7. The ratio of the amount of data collected versus the minimum data needed (y-axis) for different target V^* (x-axis) in simulations initializing with $n_0 = 10\%$ of the data set. For each data set, we show simulations for $T = 1, 3, 5$ maximum rounds. The dashed black line corresponds to collecting the least amount of data needed to reach V^*.

trend.

Alternative metrics to RMSE. When evaluating regression functions on their ability to estimate data requirements, we must be able to differentiate between estimators that over- versus under-estimate model performance (and therefore under- or over-estimate data requirements). Since RMSE is not a signed metric, it does not provide this information, and we consequently explore alternative signed metrics. Table 5 evaluates each regression function on the log of the relative ratio of estimation error $\log \hat{v}(n; \boldsymbol{\theta}^*) - \log v^*(n)$ when extrapolating to larger data sets. The log relative ratio is a signed metric where negative values means that we are pessimistic (*i.e.* $\hat{v}(n; \boldsymbol{\theta}^*) < v^*(n)$ on average) and positive values means that we are optimistic. Ideally, we want the smallest positive log relative ratio.

Unlike Table 3, the Arctan function does not consistently dominate on any individual task when evaluating on the log relative ratio. However, Table 5 also does not reveal any clear best performing regression function. Specifically, Power Law, Arctan, Logarithmic, and Algebraic Root each rank the best 4, 7, 7, and 8 times, respectively. This result supports our belief that evaluating regression error alone does not permit us to identify a good data collection policy.

C. Ablations on different architectures

In this section, we further explore CIFAR100 and repeat the previous experiments but with different architectures, to show that our results are consistent even with larger models. Specifically, we observe that with small data sets, estimating data requirements is more difficult and that moreover, regression error does not give a complete picture in terms of determining a good data collection policy. Comparing against the baseline ResNet18, we consider ResNet50 and ResNet101 as well as WideResNet-16-4 and WideResNet-16-8 [41]. All models are trained in the exact same setup.

Table 6 reports RMSE from fitting each regression function with $n_0 = 10\%, 20\%, 50\%$ of the data. Similar to the baseline, the Arctan function is almost always the best performing regression function. Figure 7 plots the ratios of the amount of data collected versus the minimum data required for each of the alternative architectures. ResNet50 and ResNet101 follows the

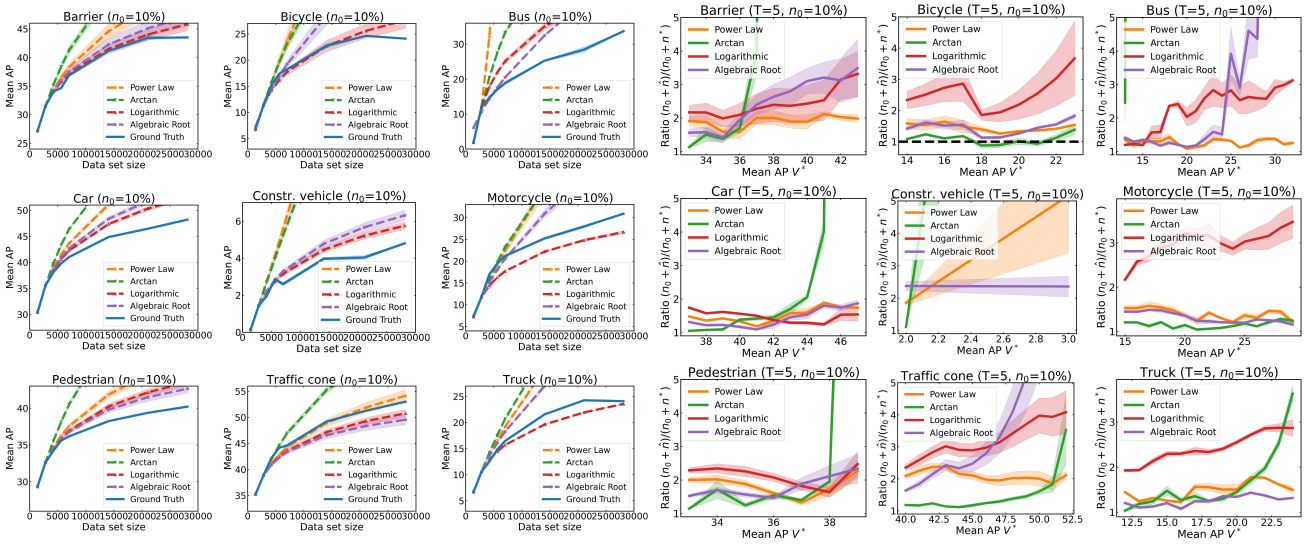

Figure 8. Experiments where the score function $V_f(\mathcal{D})$ corresponds to the AP of each individual class from the nuScenes data set. All plots show mean±standard deviation. (Left columns) Regression plots extrapolating performance of each individual class. (Right columns) The ratio of the amount of data collected versus the minimum data needed for different target V^* in simulations initializing with $n_0 = 10\%$ of the data set. We fix $T = 5$ and apply the correction factor fit using CIFAR10. The dashed black line corresponds to collecting the least amount of data needed to reach V^*.

same trends as the baseline ResNet18 (see Figure 3) in that Arctan significantly over-estimates the data requirement, even though it presents the lowest RMSE among the regression functions. The WideResNets show a slightly different picture. Here, although Arctan does not always over-estimate the requirement for all V^*, it is nonetheless, the most pessimistic estimator and recommends collecting more data than the other functions.

D. The data collection problem for class specific metrics

In many applications, we may be motivated to collect data in order to improve on a specific class. In this section, we explore scenarios where the score function $V_f(\mathcal{D})$ is a class-specific metric. We consider 3-D object detection over the nuScenes data set. For every class, we perform regression and simulation over the entire data set after setting $V_f(\mathcal{D})$ to be the AP for that specific class. That is, we use the same setup and data collection as described in Appendix A but now fit our regression models to the class-specific metric.

Figure 8 (left) plots regression analysis for each class when $n_0 = 10\%$ of the data. We first observe that the ground truth $v(n)$ are not always concave, monotonically increasing functions. In particular, the bicycle, construction vehicle, and truck classes contain situations where performance slightly decreases after increasing data. Moreover, these three classes are also the classes with the lowest AP even after training with the full data set. Finally, all of the regression functions, including the Arctan function, tend to be optimistic for most of the classes. In particular, only the traffic cone class features multiple pessimistic regression functions. These results suggest that in general, it can be more difficult to fit regression curves for individual class-specific metrics as opposed to fitting for mean AP.

Figure 8 (right) plots simulation results after employing the τ fit from CIFAR10 data for $T = 5$ rounds. Here, we demonstrate the effectiveness of our general recipe (*i.e.* to use $T = 5$ rounds and incorporate a correction factor τ). That is, for most of the classes, the Power Law, Logarithmic, and Algebraic Root functions consistently achieve ratios between 1 to 3. These methods achieve their poorest performance for the bicycle and construction vehicle classes, as these classes are naturally the most noisy and thus more challenging for estimating the data requirement.

It is worth noting that Arctan continues to over-estimate the data requirement by large margins for six out of nine classes, even though the regression plots themselves show that the Arctan function is optimistic. This behavior is due to the fact that we have $T = 5$ rounds to meet the data requirement. For instance in the first round when our initial data set contains $2,813$ images, we will significantly under-estimate the data requirement. After the first round however, we will have added a new data point into our regression set and re-fit the Arctan function. It is likely that after this first round, the next Arctan function will be pessimistic. As a result in the second round, we would over-estimate the data requirement, leading to the corresponding curve. Since the simulations show Arctan to over-estimate, we conclude the pessimistic nature of the Arctan

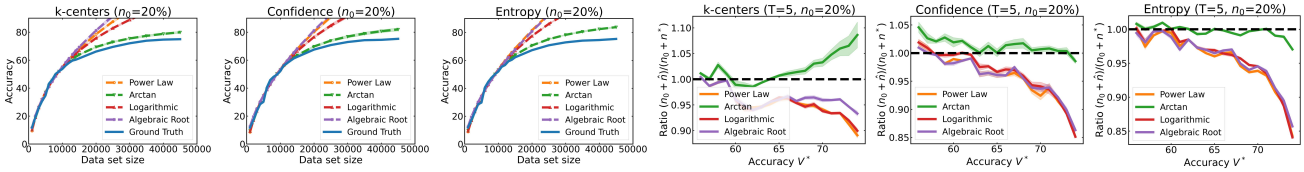

Figure 9. Experiments evaluating three different active learning strategies on CIFAR100 with $n_0 = 20\%$ of the data set and $T = 5$ rounds. (Left) regression plots showing mean±standard deviation extrapolating performance. (Right) The ratio of data collected versus the minimum data needed (y-axis) for different target V^* (x-axis) in simulations.

function is generally consistent across data sets and tasks, likely being a factors of the function and the model fitting process.

E. The data collection problem with active learning

Although all of our experiments so far have considered collecting data by randomly sampling data points from a fixed training data set, we can also consider structured techniques such as active learning, where data is collected and labeled using sample efficient strategies. In conventional active learning, we would collect a pre-determined budget of B data points each round over T rounds to attain the best possible final model. Instead, if we are given a performance target V^*, we can use our data collection framework to dynamically choose a budget B_t to collect in each round for up to T rounds.

Figure 9 plots experiments of data collection for CIFAR100 when using three active learning strategies, k-centers [31], Least Confidence [32], and Max Entropy [32], rather than random sampling. Figure 9 (left) demonstrate regression analysis for each strategy when initializing with $n_0 = 20\%$ of the data set. We use a larger initial set than in the previous experiments because the ground truth learning curves are not always concave monotonically increasing, especially in the lower data regimes. By giving a sufficiently large initial data set, the regime of extrapolation is stable and does not contain erratic trends.

Figure 9 (right) show our simulation analysis for $T = 5$ rounds of data collection using each active learning strategy. We set $\tau = 0$ since the previous correction factor values were designed from random sampling and with an initial $n_0 = 10\%$ of the data set. Nonetheless, this simulation approximates the scenario where we would dynamically choose the budget before each active learning round for up to T rounds. These plots validate that our main empirical findings all hold regardless of the specific techniques used in collecting data. For each active learning strategy, all four of the regression functions achieve ratios between 0.8 and 1.1. Furthermore, Arctan typically over-estimates the data requirement whereas the others are more likely to under-estimate the requirement.