A Meta-Learning Approach to Predicting Performance and Data Requirements

Achin Jain[1*] Gurumurthy Swaminathan[1] Paolo Favaro[1] Hao Yang[1]
Avinash Ravichandran[1] Hrayr Harutyunyan[1,2†] Alessandro Achille[1] Onkar Dabeer[1]
Bernt Schiele[1] Ashwin Swaminathan[1] Stefano Soatto[1]
[1] AWS AI Labs, [2] University of Southern California
{achij,gurumurs,pffavaro,haoyng,ravinash,aachille,onkardab,bschiel,swashwin,soattos}@amazon.com

Abstract

We propose an approach to estimate the number of samples required for a model to reach a target performance. We find that the power law, the de facto principle to estimate model performance, leads to large error when using a small dataset (e.g., 5 samples per class) for extrapolation. This is because the log-performance error against the log-dataset size follows a nonlinear progression in the few-shot regime followed by a linear progression in the high-shot regime. We introduce a novel piecewise power law (PPL) that handles the two data regimes differently. To estimate the parameters of the PPL, we introduce a random forest regressor trained via meta learning that generalizes across classification/detection tasks, ResNet/ViT based architectures, and random/pre-trained initializations. The PPL improves the performance estimation on average by 37% across 16 classification and 33% across 10 detection datasets, compared to the power law. We further extend the PPL to provide a confidence bound and use it to limit the prediction horizon that reduces over-estimation of data by 76% on classification and 91% on detection datasets.

1. Introduction

More data translates to better performance, on average, and higher cost. As data requirements scale, there is a natural desire to predict the cost to train a model and what performance it may achieve, as a function of cost, without training. Towards this goal, neural scaling laws [3, 4, 9, 19, 20, 40] have been proposed that suggest that the performance of a model trained on a given dataset size follows a linear fit when plotted on a logarithmic scale (power law in linear scale).

In practice, however, the power law provides only a family of functions and its parameters must be fitted to each specific case for performance prediction. A common use case is one where a *small* initial dataset is made available and can be used to obtain small-scale performance statistics that are relatively inexpensive to obtain and can be used to fit the power law parameters. Then, the fitted function is used to predict the performance for any dataset size training through extrapolation. This approach is found empirically to generalize across several datasets and deep learning models [40]. However, most applications of power law are done in the high-shot regime. The few-shot regime (e.g., 5 samples/class) is particularly useful given the prevalence of pre-trained initializations available for model training. In the few-shot regime, the performance curve exhibits a nonlinear progression followed by a linear progression, see Figure 1. Thus, data requirements based on the power law can lead to significant errors incurring additional cost for acquiring data.

In this paper, we propose a piecewise power law (PPL) that models the performance as a quadratic curve in the few-shot regime and a linear curve in the high-shot regime in

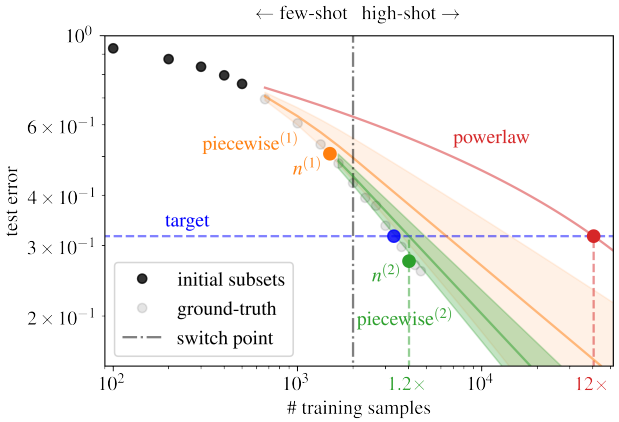


Figure 1. Performance estimation curves using the powerlaw and piecewise power law (PPL) with estimated confidence. The PPL reduces over-estimation of the power law from 12× to 1.9× in 1 step, and further to 1.2× in 2 steps using the estimated confidence bounds to limit the prediction horizon $n^{(1)}$ in the first step.

*Corresponding author. † Work done at AWS.

the log-log domain, while ensuring continuity during the transition. To estimate the parameters of the PPL, we first identify the switching point between the quadratic and linear curves using PowerRF, a random forest regressor trained via meta-learning, and then use the performance statistics to fit the remaining parameters. We show that our approach provides a better estimation of performance than the power law across several datasets, architectures, and initialization settings. We extend the PPL to provide a confidence estimate that is used to further reduce the error in data estimation. In Figure 1, the confidence estimate controls the maximum number of samples in a step such that the PPL uses two steps to achieve the target performance with 1.2× over-estimation compared to 12× with the power law.

Our contributions are as follows. We propose an improved policy for predicting data size needed to achieve a target accuracy with three main innovations: (1) a piecewise power law model (PPL) that approximates the performance error curve from low-to-high-shot regime, (2) a meta-learning approach to estimate the parameters of the PPL, and (3) incorporating the confidence interval of the estimates to limit the prediction horizon and reduce error in data estimation. We demonstrate the generalization of the PPL and the meta-model on 16 classification and 10 detection datasets, improving the (1) performance estimates by 37% on classification and 33% on detection datasets and (2) data estimates by 76% on classification and 91% on detection datasets, with respect to the power law.

2. Related work

The power law is considered as the *de facto* principle for modeling learning curves [1, 3, 9, 19, 20, 30, 40]. It has also been used for analysis and evaluation of design choices such as pre-training, architecture, and data augmentation [20]. The focus of our work, however, is not just on evaluating the fit quality of a predictor on the learning curves but also using it to predict performance through extrapolation and estimate data requirements to reach a target performance. In one of the closely related work, Mahmood et al. [30] evaluate the power law along with other modeling techniques such as algebraic, arctan, and logarithmic in predicting performance and data requirements. They improve the estimates of data requirements via a correction factor and multiple rounds of data collection. In a follow up and *concurrent* work, Mahmood et al. [31] propose stochastic optimization with the power law as a predictor to estimate data requirements. In contrast to these works, this paper focuses on the few-shot setting (e.g., when an initial dataset has 5 samples per class) where the power law breaks as the log-performance versus log-dataset sizes follows a non-linear progression. We not only propose an extension to the power law by modeling the few-shot and high-shot regimes differently but also introduce a robust strategy that uses model confidence to reduce data estimation errors over multiple rounds.

Several works [2, 6, 19, 40] have shown that the learning curves exhibit multiple regimes. Hestness et al. [19] argue about the existence of three data regimes: small data, power law, and irreducible error. Rosenfeld et al. [40] model the transition from the small data regime to the power law regime using a complex envelope function. However, unlike this paper, they mainly focus on the fit quality across different model and data configurations, and the extrapolation results on few-shot datasets show poor generalization. In a *concurrent* work, Caballero et al. [6] propose the broken neural scaling law, which is a smoothly interpolated piecewise linear curve to model different regimes such as initial random fitting, power law, double descent, and saturation. Alabdulmohsin et al. [2] also model the learning curves in multiple regimes. In [2, 6], the setting is different than ours in that (1) a predictor is fit on subsets of JFT300 [44] used for pretraining with evaluation on the downstream datasets, (2) the primary focus is on evaluating the curve fitting and not on estimating data requirements, and (3) the evaluation is done in the very large data regime with millions of samples which is not encountered in real-life models because data is scarce and is usually not enough to saturate the performance of the models. In this paper, we mainly focus on performance extrapolation with a few-shot dataset.

3. Predicting performance and data requirements

We consider the problem of extrapolating the performance (test error) of a trained model, had we trained it on a larger number of samples. This prediction is useful to estimate the data requirements to achieve a given performance target, a fundamental step in the deployment of a model in real-world applications. Specifically, given a model $\mathcal{M}$ trained on a dataset $\mathcal{D}^{(0)}$ with $n^{(0)}$ training samples, we want to predict its performance on dataset sizes $n > n^{(0)}$. We pose this problem in the practical use case of few-shot learning: (1) $\mathcal{D}^{(0)}$ contains only a few samples per class, and (2) $\mathcal{M}$ is fine-tuned after pre-training on ImageNet [10] for classification tasks and COCO [27] for object detection. We denote $v(n)$ as the true performance score of $\mathcal{M}$ obtained by fine-tuning and $\hat{v}(n)$ as the predictor of the performance of $\mathcal{M}$. We choose Top-1 Accuracy for classification and mean Average Precision (mAP) for detection as the score metric.

3.1. Problem statement

Predicting performance. For a given dataset, we construct M subsets $\mathcal{S}_1 \subset \mathcal{S}_2 \subset ... \subset \mathcal{S}_m = \mathcal{D}^{(0)} \subset \mathcal{S}_{m+1} \subset ... \subset \mathcal{S}_M = \mathcal{D}^{(\text{FULL})}$. Here $\mathcal{D}^{(0)}$ is the initial dataset we use for extrapolation in practice and $\mathcal{D}^{(\text{FULL})}$ is a larger dataset obtained by adding samples to $\mathcal{D}^{(0)}$. We fine-tune $\mathcal{M}$ on each subset to obtain $\{n_i, v(n_i)\}_{i=1}^{M}$ sample pairs, where $n_i = |\mathcal{S}_i|$ and $v(n_i)$ is the performance obtained by the model trained on dataset $\mathcal{S}_i$. Then, we fit a predictor

$\hat{v}(n)$ on samples $\{n_i, v(n_i)\}_{i=1}^{m}$ and evaluate its extrapolation performance on $\{n_i, v(n_i)\}_{i=m+1}^{M}$. Our goal is to build a predictor that achieves the smallest mean prediction error

$$\mathcal{E}_{\text{perf}} = \frac{1}{M-m} \sum_{i=m+1}^{M} |v(n_i) - \hat{v}(n_i)| . \tag{1}$$

Predicting data requirements. Next, we estimate the number of samples that should be added to $\mathcal{D}^{(0)}$ to reach a target performance v^*. To do so, we invert the predictor $\hat{v}(n)$ and get the first estimate of the data requirements $n^{(1)} = \hat{v}^{-1}(v^*)$. Due to modeling errors, the first estimate may over/under-estimate the number of samples $n^* \doteq v^{-1}(v^*)$ needed to reach the desired target. In the case of over-estimation, i.e., $v(n^{(1)}) > v^*$ or equivalently $n^{(1)} > n^*$, we collect further $n^{(1)} - n^{(0)}$ samples, where $n^{(0)} = |\mathcal{D}^{(0)}|$, and stop. In the case of under-estimation, it is desirable to update the predictor with the collected data and the corresponding model performance after re-training with the additional performance pair $\{n^{(1)}, v(n^{(1)})\}$, and to take additional steps to reach the target performance. This is referred to as the *data collection problem* in [30]. We consider the same setting as [30], where the maximum number of steps is limited to T since each round of data collection and annotation can be costly to initiate. Specifically, at the k-th step of the data collection, we fit the predictor $\hat{v}_k$ on $\{n_i, v(n_i)\}_{i=1}^{m} \cup \{n^{(i)}, v(n^{(i)})\}_{i=1}^{k-1}$ to obtain the data requirement estimate $n^{(k)}$. Our goal is to achieve a small data estimation error $|\mathcal{E}_{\text{data}}|$ where

$$\mathcal{E}_{\text{data}} = \frac{n^{(K)} - n^*}{n^*}, \tag{2}$$
$$K = \min\Big\{T, \min\{k : \hat{v}_k(n^{(k)}) > v^*\}\Big\}.$$

$\mathcal{E}_{\text{data}} < 0$ represents an under-estimate and $\mathcal{E}_{\text{data}} > 0$ an over-estimate.

Challenges in the few-shot setting. In the few-shot setting, the largest subset used to fit a predictor $\mathcal{S}_m = \mathcal{D}^{(0)}$ contains only a few samples per class (e.g., we consider 5 for classification and 10 for object detection). We observe that the power law [9] as a predictor breaks down in this setting when $n_M \gg n^{(0)}$, i.e., when we extrapolate its performance to the high-shot regime starting from the few-shot regime. To this end, we propose a piecewise power law that models the two regimes differently. In addition, a direct estimate $n^{(k)} = \hat{v}_k^{-1}(v^*)$ for predicting data requirements is not always the best choice, especially when the size of the initial dataset $n^{(0)} \ll n^*$, i.e, $\hat{v}_1$ is fit to the data samples from the few-shot regime and the target performance can be achieved only in the high-shot regime. We propose a strategy that uses model confidence of the predictor to reduce over-estimation of data requirements, a problem that plagues the prior approach such as the power law.

3.2. The piecewise power law (PPL)

We empirically observe that the test error in logarithmic scale $\log(1 - v(n))$ varies nonlinearly as a function of $\log(n)$ in the few-shot regime and linearly in the high-shot regime, see an illustration in Figure 1. Also, the point of separation of these data regimes differs for every dataset. Thus, we design the predictor $\hat{v}(n)$ to be smooth and monotonic, such as the power law, but also as a piecewise function that can differentiate between the two data regimes. Additionally, to model the stochasticity in training models on the subsets, we design $\hat{v}(n)$ to predict both the mean performance and its uncertainty. The uncertainty estimate is used to predict data requirements as outlined in the following section.

To model the nonlinear behavior in the few-shot regime, we consider a quadratic model. Thus, the piecewise power law is defined as

$$\log(1 - \hat{v}(n; \boldsymbol{\theta})) = \begin{cases} \theta_1 + \theta_2 \log(n) + \theta_3 \log(n)^2, & \text{if } n \leq N \\ \theta_4 + \theta_5 \log(n), & \text{otherwise} \end{cases} \tag{3}$$

where $\{\theta_i\}_{i=1}^{5}$ are the coefficients of the model and N is the switching point from a quadratic to a linear function that differentiates between the few-shot and the high-shot data regimes. By enforcing continuity and differentiability at N, θ_4 and θ_5 become functions of $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]^T$ and N. Thus, the piecewise power law (3) effectively has only 4 parameters. Given the data samples $\{n_i, v(n_i)\}_{i=1}^{m}$, we first obtain N through a meta-model trained on a dictionary of learning curves of several datasets (see Section 3.3 for details), and then derive $\boldsymbol{\theta}$ by fitting the piecewise model (3) on $\{n_i, v(n_i)\}_{i=1}^{m}$ via nonlinear least squares in the logarithmic scale using the Levenberg–Marquardt algorithm on

$$\min_{\boldsymbol{\theta}} \ \|\mathbf{y} - \hat{\mathbf{y}}(\boldsymbol{\theta})\|^2 \tag{4}$$
$$\mathbf{y}_i = y_i = \log(1 - v(n_i)), \quad \forall i \in \{1, 2, ..., m\},$$
$$\hat{\mathbf{y}}_i = \hat{y}_i = \log(1 - \hat{v}(n_i)), \quad \forall i \in \{1, 2, ..., m\}.$$

We estimate the mean $\mu_{\hat{v}}(n)$ and the variance $\sigma^2_{\hat{v}}(n)$ (due to modeling errors) of the predictor as follows. First, we compute of the covariance matrix of the parameters [11] as

$$\boldsymbol{\Sigma}_{\boldsymbol{\theta}} = (\mathbf{J}^T \mathbf{J})^{-1}, \ \mathbf{J} = \left[\frac{\partial \hat{\mathbf{y}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right]_{m \times 3} \tag{5}$$

where $\mathbf{J}$ is the Jacobian matrix of $\hat{\mathbf{y}}$ with respect to $\boldsymbol{\theta}$. By rewriting (3) as $\hat{y}(n; \boldsymbol{\theta}) = \boldsymbol{\alpha}^T(n)\boldsymbol{\theta}$, we obtain the variance $\sigma^2_{\hat{y}}(n) = \boldsymbol{\alpha}^T(n)\boldsymbol{\Sigma}_{\boldsymbol{\theta}}\boldsymbol{\alpha}(n)$, where $\boldsymbol{\alpha}(n)$ is given by

$$\boldsymbol{\alpha}(n) = \begin{cases} [1, \log(n), \log(n)^2], & \text{if } n \leq N \\ [1, \log(n), 2N\log(n) - N^2] & \text{otherwise.} \end{cases} \tag{6}$$

The latter condition for $n > N$ comes from the continuity and differentiability constraints at N. Now, by assuming $\hat{y}$ is normally distributed, $\hat{v} = 1 - \exp(\hat{y})$ is log-normally distributed with mean $\mu_{\hat{v}}$ and variance $\sigma^2_{\hat{v}}$ given by

$$\mu_{\hat{v}}(n) = 1 - \exp\left(\hat{y}(n) + \frac{\sigma^2_{\hat{y}}(n)}{2}\right), \tag{7}$$

$$\sigma^2_{\hat{v}}(n) = \exp\left(\hat{y}(n) + \frac{\sigma^2_{\hat{y}}(n)}{2}\right)\sqrt{\exp(\sigma^2_{\hat{y}}(n) - 1)}. \tag{8}$$

Using confidence bounds to predict data requirements. The test error in logarithmic scale $\log(1 - v(n))$ decays rather slowly in the few-shot regime and we observe that the power law [9] can lead to large over-estimation in data requirements, i.e., $\mathcal{E}_{\text{data}} \gg 0$; for example, the power law over-estimates by $12\times$ in Figure 1. Increasing the number of steps in the data collection process is not helpful since $v(n^{(1)}) > v^*$ in the first step itself. We propose a strategy that exploits the model confidence of the predictor to prevent large estimation errors. Specifically, we limit the number of samples in a step such that $3\sigma_{\hat{v}_k}(n^{(k)}) \leq \tau$, where τ is the confidence threshold. Under this policy, the number of samples in step k of the data collection process is determined by

$$n^{(k)} = \min\left\{\mu^{-1}_{\hat{v}_k}(v^*), \sigma^{-1}_{\hat{v}_k}(\tau/3)\right\} \tag{9}$$

where $\hat{v}_k$ is fit on $\{n_i, v(n_i)\}_{i=1}^m \cup \{n^{(i)}, v(n^{(i)})\}_i^{k-1}$. In Figure 1, we first add $n^{(1)}$ samples using (9) and then we achieve the target in the next step with $n^{(2)}$ samples while satisfying $3\sigma_{\hat{v}_2}(n^{(2)}) \leq \tau$. We empirically show that the same τ works consistently for all datasets across classification and object detection tasks.

Comparison with the power law. The power law [9] predictor of model performance is given by

$$1 - \hat{v}(n; \boldsymbol{\theta}) = \theta_1 n^{\theta_2} + \theta_3. \tag{10}$$

The model comprises of 3 parameters $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]$. The special case of the piecewise power law (3) with $N = 0$ is equivalent to the power law (10) when $\theta_3 = 0$, that is

$$\log(1 - \hat{v}(n; \boldsymbol{\theta})) = \theta_1 + \theta_2 \log(n), \tag{11}$$

whose parameters can simply be obtained by solving linear regression in the logarithmic space. The parameter θ_3 models the asymptotic error of the learning curves. For most datasets we consider in the few-shot setting (with pre-training), we empirically observe that the predictor (11) achieves comparable or smaller $\mathcal{E}_{\text{perf}}$ than predictor (10); see Appendix E.

3.3. Meta-learning for the piecewise power law

We need to estimate four parameters in the piecewise power law (3): $\boldsymbol{\theta} \in \mathbb{R}^3$ and N by fitting on $\{n_i, v(n_i)\}_{i=1}^m$. N is determined by the transition point between the non-linear and the linear regime. However, identifying an optimal N is non-trivial given that many choices of N can produce fit with low errors. We find that a brute-force search that chooses N based on fitting $\{n_i, v(n_i)\}_{i=1}^{m-1}$ and evaluating on $\{n_m, v(n_m)\}$ is better than the power law in several cases, but still far from the upper bound performance given by the piecewise power law.

To bridge the gap, we propose to learn a meta-model that, given data samples $\{n_i, v(n_i)\}_{i=1}^m$, leverages knowledge from a dictionary of learning curves to predict the switching point N between the quadratic and the linear functions. Specifically, we train a Random Forest [5] regressor $\mathcal{F}$ via meta-learning to predict

$$\hat{N} = \mathcal{F}\left(\{\log(n_i), \log(v(n_i))\}_{i=1}^m, \log(C)\right), \tag{12}$$

where C is the number of classes in a dataset. Once the meta-model predicts $\hat{N}$, other parameters of the piecewise power law are determined by fitting on the subsets of performance pairs with dataset size either smaller or larger than $\hat{N}$, via the optimization of Eq. (4) as described in Section 3.2. We choose to predict only the parameter $\hat{N}$ using the meta-model as this can be learnt from the statistics of the learning curves, and the other parameters are better estimated from the data samples $\{n_i, v(n_i)\}_{i=1}^m$. We demonstrate in Section 4.5 that the PPL is robust to errors in N.

Training and inference using the meta-model. To train the meta-model, we use data samples $\{n_i, v(n_i)\}_{i=1}^M$ from all subsets of *both* few-shot and high-shot regimes. For discrete choices of $N \in \{n_1, n_2, \ldots, n_M\}$, we fit the piecewise power law eq. (3) on samples from the few-shot region $\{n_i, v(n_i)\}_{i=1}^m$ and compute the mean prediction error $\mathcal{E}_{\text{perf}}$ as defined in eq. (1). The N that minimizes the evaluation error is considered to be the ground-truth N^* for the dataset. We compute N^* for several datasets and train the meta-model in a leave-one-out (LOO) fashion, i.e., during inference the meta-model used to evaluate on dataset $\mathcal{D}$ is trained on datasets *excluding* $\mathcal{D}$. In our experiments, we show generalization of the meta-model to several datasets, model architectures, and training settings. Lastly, due to very distinct numbers of object categories in the datasets, we train a separate set of meta-models for classification and object detection tasks.

4. Experiments

Datasets. We evaluate the piecewise power law on 16 classification and 10 detection datasets from diverse domains. The datasets vary widely in terms of complexity measured by number of classes and number of training images. Refer to the dataset statistics in Appendix A.

Subsets used for fitting and evaluation. To provide a comprehensive analysis, we consider different subsets for fitting and evaluation of the performance predictors.

The **few-shot setting** is the main focus of this paper and is most widely used across all experiments. For most classification datasets with the exceptions listed below, we fit a performance predictor on the 5 subsets comprising of {1, 2, 3, 4, 5} samples per class. For Cifar10 and EuroSAT, the subsets for fitting comprise of {5, 10, 15, 20, 25} samples per class, and for iCassava {10, 15, 20, 25, 30} samples per class to ensure the number of training images in the smallest subset is more than the batch size of 32 used universally across all classification experiments. For all classification datasets, we compute $\mathcal{E}_{\text{perf}}$ (1) on the subsets comprising of {10, 15, ..., 100}% of the dataset. For all object detection datasets, we fit a performance predictor on the 5 subsets

Table 1. Mean prediction error $\mathcal{E}_{\text{perf}}$ (1) for extrapolating performance from few-shot to high-shot. Piecewise GT denotes the upper bound. Piecewise outperforms powerlaw on 12/16 datasets.

CLASSIFICATION	powerlaw [9]	arctan [30]	piecewise (ours)	piecewise (GT)
Caltech256 [13]	10.3±3.6	3.0±1.5	**2.0±0.9**	1.2±0.5
Cifar10 [25]	6.7±2.2	6.0±6.2	**0.9±0.5**	0.5±0.4
Cifar100 [25]	6.5±3.1	17.2±7.3	**6.1±3.5**	5.3±3.8
CUB200 [45]	**2.6±0.3**	14.1±3.9	4.0±0.1	0.7±0.1
Decathlon Aircraft [32]	18.0±1.8	23.5±15.9	**11.1±4.2**	11.1±4.1
Decathlon DTD [7]	**3.2±1.9**	4.7±2.5	5.6±1.7	2.1±1.1
Decathlon Flowers [36]	**1.0±0.3**	1.5±0.3	2.0±0.3	1.1±0.0
Decathlon UCF101 [43]	14.5±1.9	15.5±4.9	**4.1±3.0**	4.1±2.9
EuroSAT [17, 18]	2.6±0.6	4.2±2.4	**0.9±0.2**	0.9±0.2
FGVC Aircrafts [32]	25.8±1.6	**9.2±7.4**	19.1±1.4	11.1±1.8
iCassava [35]	9.2±6.7	14.6±4.8	**6.9±2.4**	6.9±2.4
MIT-67 [38]	**4.2±1.7**	8.2±5.3	4.3±2.5	3.9±2.3
Oxford Flowers [36]	1.5±0.4	1.5±0.3	**1.2±0.3**	1.1±0.4
Oxford Pets [37]	9.2±0.4	**1.1±0.5**	5.6±0.8	1.7±0.5
Stanford Cars [24]	26.4±1.3	17.6±2.9	**17.3±2.7**	7.7±3.3
Stanford Dogs [22]	6.1±5.4	7.8±2.7	**2.3±1.0**	1.2±0.1
AVERAGE	9.2±2.1	9.3±4.3	**5.8±1.6**	3.8±1.5

Table 2. Mean prediction error $\mathcal{E}_{\text{perf}}$ (1) for extrapolating performance from few-shot to high-shot. Piecewise GT denotes the upper bound. Piecewise outperforms powerlaw on 9/10 datasets.

DETECTION	powerlaw [9]	arctan [30]	piecewise (ours)	piecewise (GT)
Cityscapes [8]	1.3±0.8	1.5±0.6	**1.1±0.5**	0.9±0.6
Comic [21]	4.4±2.9	28.0±33.9	**4.1±1.0**	3.4±2.0
CrowdHuman [42]	0.8±0.2	1.5±0.5	**0.7±0.3**	0.5±0.3
DUO [28]	3.9±1.8	4.5±1.6	**2.4±0.5**	1.8±0.9
KITTI [12]	2.6±2.1	**1.5±1.0**	1.6±0.7	1.5±1.4
MinneApple [14]	4.7±2.3	**1.1±0.3**	1.9±1.0	0.6±0.1
SIXray [33]	6.9±0.9	8.3±9.9	**2.7±2.7**	2.4±1.1
table-detection [41]	5.9±2.7	7.8±2.2	**5.5±0.8**	5.5±2.2
VisDrone [47]	**0.3±0.1**	0.7±0.3	0.8±0.3	0.4±0.1
Watercolor [21]	5.2±1.5	6.7±2.7	**3.2±1.4**	3.1±1.7
AVERAGE	3.6±1.5	6.2±5.3	**2.4±0.9**	2.0±1.0

Table 3. Mean prediction error $\mathcal{E}_{\text{perf}}$ (1) for extrapolating performance from mid-shot to high-shot. Piecewise works better on 14/16 and 12/16 datasets using 30% and 50% data, respectively.

CLASSIFICATION	using 30% data		using 50% data	
	powerlaw	piecewise	powerlaw	piecewise
Caltech256	0.8±0.6	**0.6±0.3**	0.5±0.2	**0.3±0.0**
Cifar10	0.3±0.3	**0.1±0.0**	0.3±0.1	**0.1±0.0**
Cifar100	1.1±0.8	**0.6±0.1**	**0.3±0.1**	**0.3±0.1**
CUB200	4.5±2.4	**2.5±1.3**	1.6±0.6	**0.9±0.0**
Decathlon Aircraft	8.5±1.6	**4.1±2.0**	4.2±0.3	**1.5±1.1**
Decathlon DTD	**1.2±0.4**	1.7±0.8	1.4±0.8	**1.3±0.4**
Decathlon Flowers	**1.4±0.4**	2.6±1.7	**1.0±0.3**	1.8±0.3
Decathlon UCF101	2.7±1.5	**2.2±1.2**	1.0±0.4	**0.8±0.3**
EuroSAT	0.3±0.2	**0.1±0.0**	0.2±0.1	**0.1±0.0**
FGVC Aircrafts	6.4±4.7	**2.0±0.8**	2.6±0.9	**0.9±0.4**
iCassava	2.4±0.7	**1.2±0.5**	0.5±0.3	**0.5±0.1**
MIT-67	2.3±1.3	**1.1±0.4**	1.2±0.7	**0.9±0.5**
Oxford Flowers	3.6±1.8	**1.9±0.7**	**0.6±0.3**	0.7±0.5
Oxford Pets	2.2±0.9	**1.1±0.2**	1.2±0.8	**0.7±0.4**
Stanford Cars	9.7±0.1	**1.1±0.4**	4.3±0.9	**0.4±0.1**
Stanford Dogs	3.1±2.0	**2.1±0.3**	**1.2±1.2**	1.6±0.3
AVERAGE	3.2±1.2	**1.6±0.7**	1.4±0.5	**0.8±0.3**

comprising of $\{1, 5, 10, 15, 20\}$ samples per class and compute $\mathcal{E}_{\text{perf}}$ (1) on the subsets comprising of $\{25, 30, ..., 100\}$ samples per class. To construct the subsets, we follow the natural k-shot sampling protocol [26]. Due to high variance in the training statistics, we choose a larger number of samples per class for fitting for the detection tasks compared to the classification tasks. We also evaluate our approach in the **mid-shot setting** for the classification datasets where a predictor is (1) fit on $\{10, 15, 20, 25, 30\}$% data and evaluated on $\{35, 40, ..., 100\}$% data, and (2) fit on $\{10, 20, 30, 40, 50\}$% data and evaluated on $\{55, 60, ..., 100\}$% data.

We repeat all experiments with three different random seeds selecting different set of images for fitting and evaluation. For training recipe used for finetuning and linear probing the subsets, see Appendix B.

4.1. Extrapolation from few-shot to high-shot

We first evaluate the piecewise model in the few-shot setting with the mean prediction error $\mathcal{E}_{\text{perf}}$ (1) as the metric for evaluation. We compare the piecewise power law (3) with the power law [9, 30] on classification tasks in Table 1 and object detection tasks Table 2. Other predictors such as algebraic [30], arctan [30], and logarithmic [30] did not perform better in our experiments; we compare with "arctan" here, see others in Appendix C. The upper bound performance of the piecewise power law is denoted by "piecewise GT". This corresponds to the piecewise model using the ground-truth switching point N^* as described in Section 3.3. Note that the upper bound is not always close to zero because the derivation of N^* utilizes only five data samples from the few-shot regime for fitting the piecewise power law. The "piecewise" power law with the meta-model performs better on 12/16 classification tasks and 9/10 object detection tasks reducing the average mean prediction error of the "powerlaw" by 37% and 33%, respectively. It also achieves lower variance consistently across most datasets. We note that there is still a gap between the performance of the meta-model with respect to the upper bound.

4.2. Extrapolation from mid-shot to high-shot

We next evaluate the "piecewise" power law with the meta-model in the mid-shot setting. We show a comparison against the "powerlaw" for classification datasets in Table 3. We consider two scenarios where upto 30% and 50% data is used for fitting the predictors. The meta-model used with the piecewise power law is the *same* as in Section 4.1. Again, the piecewise power law is better than the power law reducing the average mean prediction errors by 50.0% and 42.8% when using 30% and 50% data for fitting, respec-

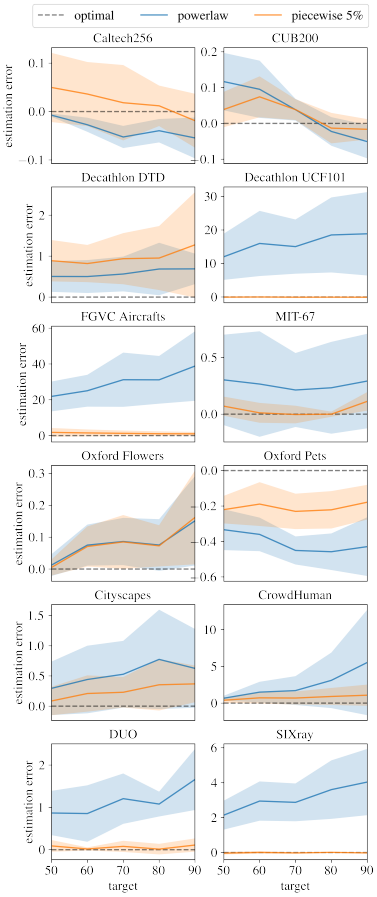
Table 4. Left: Data estimation error $\mathcal{E}_{\text{data}}$ to reach the target performance corresponding to 90% samples (of the full dataset). $\mathcal{E}_{\text{data}} < 0$ represents an under-estimate and $\mathcal{E}_{\text{data}} > 0$ an over-estimate. PPL with confidence threshold (5%) achieves the lowest error on 13/16 classification and 9/10 detection tasks. Right: PPL with confidence (5-step) achieves smaller $\mathcal{E}_{\text{data}}$ (closer to 0) than powerlaw (5-step) for different performance targets obtained by using {50, 60, 70, 80, 90}% of the full dataset (more datasets in Appendix D).

CLASSIFICATION

	powerlaw [9]		**piecewise (ours)**			
	1-step	5-step	1-step	5-step	5-step 5%	avg. steps
Caltech256	-0.6±0.1	-0.1±0.0	-0.2±0.2	-0.0±0.1	**-0.0±0.1**	3.7±1.9
Cifar10	inf	inf	0.4±0.7	0.5±0.6	**0.5±0.6**	2.3±1.9
Cifar100	0.2±0.7	0.3±0.6	-0.6±0.2	0.0±0.0	**-0.1±1.2**	4.7±0.5
CUB200	-0.1±0.1	-0.1±0.0	-0.3±0.0	-0.0±0.0	**-0.0±0.0**	4.3±0.9
Decathlon Aircraft	inf	inf	13.3±11.4	13.3±11.4	**0.2±0.0**	3.7±0.5
Decathlon DTD	0.7±0.4	**0.7±0.4**	3.1±1.2	3.1±1.2	1.3±1.3	1.7±0.9
Decathlon Flowers	-0.0±0.0	**0.0±0.0**	0.3±0.1	0.3±0.1	0.3±0.1	1.3±0.5
Decathlon UCF101	18.8±12.4	18.8±12.4	0.6±0.6	0.6±0.5	**-0.1±0.1**	4.7±0.5
EuroSAT	inf	inf	-0.7±0.1	0.0±0.1	**0.0±0.1**	3.3±1.2
FGVC Aircrafts	38.8±19.4	38.8±19.4	69.6±45.6	69.6±45.6	**1.0±1.0**	2.0±0.8
iCassava	1.7±3.3	1.7±3.3	89.3±111.4	89.3±111.4	**0.5±1.4**	3.7±1.9
MIT-67	0.1±0.6	0.3±0.4	0.6±1.2	0.8±1.1	**0.1±0.1**	3.0±0.8
Oxford Flowers	0.1±0.2	**0.2±0.1**	0.1±0.2	0.2±0.1	**0.2±0.1**	2.3±1.9
Oxford Pets	-0.8±0.0	-0.4±0.2	-0.8±0.0	-0.1±0.0	**-0.2±0.1**	5.0±0.0
Stanford Cars	8.4±1.8	8.4±1.8	15.5±10.2	15.5±10.2	**0.2±0.2**	3.0±0.0
Stanford Dogs	-0.3±0.4	**-0.0±0.2**	-0.3±0.2	-0.1±0.1	-0.1±0.0	5.0±0.0

DETECTION

	powerlaw [9]		**piecewise (ours)**			
	1-step	5-step	1-step	5-step	5-step 5%	avg. steps
Cityscapes	0.4±0.8	0.6±0.7	0.1±0.4	0.4±0.3	**0.4±0.3**	2.7±1.7
Comic	inf	inf	2.4±2.7	2.7±2.4	**2.7±2.5**	2.3±1.9
CrowdHuman	5.5±7.2	5.5±7.2	1.0±1.5	1.1±1.4	**1.1±1.4**	1.3±0.5
DUO	1.7±0.7	1.7±0.7	0.8±1.0	0.8±0.9	**0.1±0.2**	3.3±1.2
KITTI	5.0±6.3	5.0±6.3	0.2±0.4	0.4±0.2	**0.2±0.2**	2.3±1.9
MinneApple	-0.3±0.7	-0.0±0.6	-0.6±0.2	-0.2±0.2	**0.0±0.5**	4.3±0.9
SIXray	4.0±1.9	4.0±1.9	-0.2±0.2	-0.0±0.0	**-0.0±0.0**	5.0±0.0
table-detection	0.4±0.7	0.5±0.7	0.6±0.8	0.7±0.7	**-0.0±0.3**	4.3±0.9
VisDrone	0.0±0.0	**0.0±0.0**	-0.3±0.1	-0.1±0.0	-0.1±0.0	5.0±0.0
Watercolor	6.5±2.9	6.5±2.9	1.9±1.8	1.9±1.7	**-0.1±0.4**	4.0±1.4



tively, and achieving lower variance at the same time.

4.3. Prediction of data requirements to reach target performance

We provide results for maximum number steps $T \in \{1, 5\}$ and compute data estimation error $\mathcal{E}_{\text{data}}$ (2) which is a measure of under- or over-estimation of number of samples after T steps of data collection. We compare the "piecewise" power law with the meta-model against the "powerlaw" for $v^* = v(n_{90\%})$ for both classification and detection tasks in Table 4. Here, $n_{90\%}$ corresponds to the subset with 90% samples of the full dataset $\mathcal{D}^{(\text{FULL})}$. We also show a visualization of under-/over-estimation for different choices of the target performance corresponding to {50, 60, 70, 80, 90}% data for some datasets here (and others in Appendix D). In the few-shot regime, the test error decays slowly in the logarithmic scale as compared to the high-shot regime. As a result, the "powerlaw" overestimates data requirements by a huge margin in several cases across both classification and detection tasks even with 1-step ($T = 1$); all estimates more than 1000 are denoted as "inf" in Table 4. In the cases where the power law under-estimates the performance, increasing the number of steps to $T = 5$ helps reduce the data estimation error; as previously suggested in [30]. We can further reduce the data estimation error by two complementary ways. First, we improve the quality of the predictor, i.e., replace the power law by the piecewise power law. We immediately see that "piecewise" reduces large overestimation by improving the quality of extrapolation. Since the piecewise predictor is still not perfect, we further reduce the error by controlling the step sizes (9) using the confidence bounds. As a result, "piecewise 5-step 5%" (referring to $\tau = 5\%$) demonstrates lower data estimation error on 13/16 classification tasks and 9/10 detection tasks compared to "powerlaw 5-step", and consistently achieves small error with exceptions of Decathlon DTD (1.3) and Comic (2.7). "avg. steps" denotes the average number of steps taken across different random seeds with "piecewise 5-step 5%". We compute the aver-

Table 5. Generalization of the meta-model trained on ResNet-18 finetuning to different scenarios: ResNet-50 finetuning, ViT-B/16 finetuning, ResNet-18 linear probing, and ResNet-18 finetuning with a fixed learning rate.

	ResNet-50 finetune		ViT-B/16 finetune		ResNet-18 linear		ResNet-18 fixed LR	
	powerlaw	**piecewise**	**powerlaw**	**piecewise**	**powerlaw**	**piecewise**	**powerlaw**	**piecewise**
Caltech256	2.9±2.2	**1.1±0.8**	**1.0±0.4**	5.1±0.4	4.7±3.9	**3.2±1.5**	12.8±7.2	**4.2±1.6**
Cifar10	18.9±2.2	**1.9±1.4**	4.1±4.2	**3.6±4.2**	15.4±0.5	**8.3±1.3**	8.9±9.4	**0.7±0.3**
Cifar100	15.1±0.1	**11.8±0.9**	11.5±3.3	**10.3±3.1**	18.8±1.8	**12.4±4.1**	27.9±3.0	**6.4±2.6**
CUB200	4.9±1.5	**4.0±1.6**	6.5±2.2	**2.6±0.4**	3.1±0.9	**0.5±0.1**	11.9±1.1	**1.6±0.9**
Decathlon Aircraft	20.3±0.6	**10.6±1.1**	14.4±2.1	**3.9±2.1**	**2.8±0.6**	3.5±0.6	13.7±1.2	**9.9±1.4**
Decathlon DTD	5.6±3.0	**5.4±1.3**	5.0±0.9	**1.9±1.1**	1.7±0.9	**1.3±0.0**	2.4±0.7	**4.3±1.8**
Decathlon Flowers	**2.0±1.1**	2.6±0.7	1.7±0.6	**1.2±0.4**	**0.7±0.1**	0.8±0.2	3.6±0.5	**1.4±0.1**
Decathlon UCF101	15.0±1.0	**3.5±1.9**	11.5±0.3	**6.0±3.4**	**2.5±2.6**	5.4±2.3	21.2±1.1	**11.2±1.6**
EuroSAT	4.8±4.3	**0.3±0.1**	3.0±3.0	**2.5±2.8**	3.6±2.4	**3.3±0.3**	5.1±3.9	**1.5±0.2**
FGVC Aircrafts	26.6±1.8	**15.2±5.0**	21.8±1.1	**15.1±2.9**	3.8±0.7	**2.1±1.3**	29.0±1.5	**14.0±2.6**
iCassava	14.3±5.8	**1.8±0.7**	11.0±3.3	**2.1±0.7**	12.0±6.6	**4.4±0.9**	16.4±2.3	**6.0±3.6**
MIT-67	7.7±2.2	**5.9±2.9**	5.0±2.8	**3.2±1.5**	7.6±4.7	**5.0±3.0**	3.4±0.4	**3.2±0.6**
Oxford Flowers	1.5±1.0	**1.1±0.4**	0.9±0.3	**0.5±0.1**	1.6±0.3	**1.3±0.1**	3.6±0.1	**2.1±0.5**
Oxford Pets	4.1±2.8	**2.0±0.5**	14.6±1.9	**10.6±3.5**	7.8±2.5	**5.3±0.5**	10.3±0.2	**9.0±0.4**
Stanford Cars	23.9±1.1	**14.7±1.4**	20.4±0.7	**13.9±1.4**	7.1±0.8	**4.4±0.9**	33.5±0.6	**27.6±1.1**
Stanford Dogs	8.7±3.1	**6.8±0.5**	**3.5±3.2**	5.0±1.0	5.4±2.6	**5.1±0.9**	17.9±0.1	**11.0±1.6**
AVERAGE	11.0±2.1	**5.5±1.3**	8.5±1.9	**5.5±1.8**	6.2±2.0	**4.2±1.1**	13.8±2.1	**7.1±1.3**

age improvement by the PPL only on the datasets where the power law predicts less than 10× over-estimation, and yet the PPL improves the estimates by 76% on classification and 91% on detection datasets.

4.4. Generalization of the meta-model

In this section, we show generalization of the meta-model to several datasets, model architectures, and training settings. We use the *same* meta-model trained according to the procedure described in Section 3.3 across all scenarios listed below. Unless noted otherwise, we follow the few-shot setting to construct the subsets for fitting and evaluation, with the mean prediction error $\mathcal{E}_{\text{perf}}$ (1) as the metric for evaluation.

Different architectures. The meta-model is trained on learning curves of ResNet-18 [16]. Here, we show its generalization to a more complex architecture ResNet-50 and a different architecture ViT-B/16-224 [23]. All networks are initialized with ImageNet pretrained weights. The piecewise power law works better than the power law on 15/16 classification tasks with ResNet-50 and 14/16 tasks with ViT-B/16; see "ResNet-50 finetune" and "ViT-B/16 finetune" in Table 5.

Linear probing. The meta-model is trained on the data samples from finetuning on the subsets with ImageNet pre-trained weights. We show that the piecewise power law with same meta-model also generalizes while linear probing on the subsets (again with ImageNet pre-trained weights) performing better than the power law on 13/16 classification tasks; see "ResNet-18 linear" in Table 5.

Fixed learning rate. The meta-model is trained after running HPO over 3 different learning rates in {0.001, 0.005, 0.01} while finetuning on the subsets; see more details in Appendix B. However, should it be desirable to use a fixed learning rate (we choose 0.001) during finetuning on the subsets due to limited compute, we show that the meta-model also generalizes in this scenario; see "ResNet-18 fixed LR" in Table 5.

Table 6. Generalization of the meta-model trained on *finetuning ResNet-18* to *training ResNet-18 from scratch*.

	powerlaw	**arctan**	**piecewise**
Cifar10 [30]	39.02±20.3	7.98±7.1	-
Cifar10 (ours)	0.9±0.8	2.9±0.7	**0.3±0.1**
Cifar100 [30]	34.98±35.1	13.3±5.3	-
Cifar100 (ours)	4.0±0.5	19.4±5.3	**2.5±0.3**

Training from scratch. We next show that the meta-model trained with data samples from finetuning ResNet-18 network with pre-trained models (ImageNet initialization) generalizes to training ResNet-18 network from scratch (random initialization). In this case, we follow the settings by Mahmood et al [30] for constructing the subsets for fitting and evaluation of the predictors. Specifically, we fit on subsets with {2, 4, 6, 8, 10}% data and do evaluation on {20, 30, ..., 100}% data using the root mean squared error (RMSE) as the metric for evaluation. The "piecewise" power law with the meta-model achieves 67% error reduction on Cifar10 and 37% on Cifar100 with respect to the "powerlaw"; see Table 6. For completeness, we also reproduce results from [30]; however, we note that the numbers may not be comparable due to different subsets and training recipe. We provide a comparison with algebraic [30] and logarithmic [30] in Appendix F.

4.5. Ablation studies

Comparison of the meta-model with baselines. A simple baseline is to choose the switch point as the number

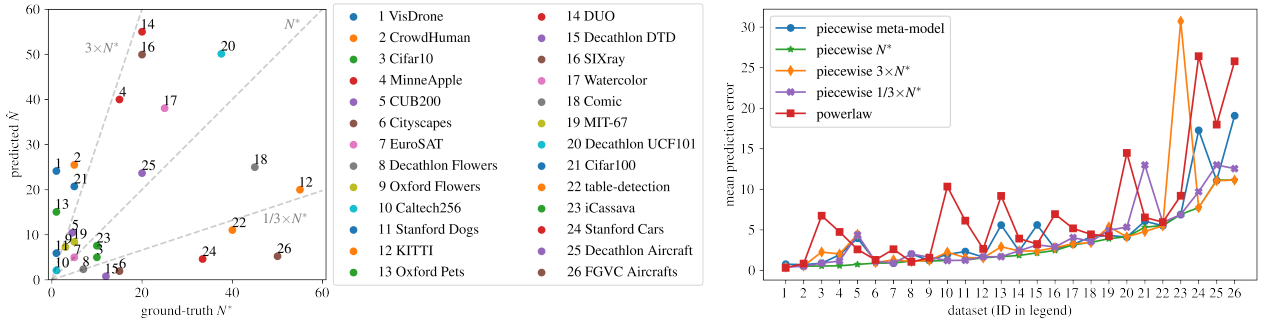


Figure 2. Left: Comparison of predicted versus ground-truth switch point (normalized by number of classes). Right: Mean prediction error $\mathcal{E}_{\text{perf}}$ (1) obtained with powerlaw, and the piecewise power law using N^*, $1/3\times N^*$, $3\times N^*$, and the meta-model.

of samples in the smallest subset, i.e., $N = n_1$. We refer to it as "linear" baseline since the piecewise power law reduces to a linear predictor in logarithmic scale; see (11). A second baseline is to find a "brute-force" solution that minimizes the error of the piecewise power law fit on the first 4 subsets $\{n_i, v(n_i)\}_{i=1}^4$ and evaluated on the last subset $\{n_5, v(n_5)\}$. Note the subtle difference with respect to the procedure to compute the ground-truth N^* in Section 3.3 where the piecewise power law is fit on all 5 subsets $\{n_i, v(n_i)\}_{i=1}^5$ and evaluated on the subsets in the high-shot regime. We observe that different methods work better for different tasks but on average the "meta-model" works best reducing the average mean prediction error by 21.6% and 19.1% on ResNet-18 and ResNet-50, respectively, compared to the "brute-force" (next best) method. See results in Appendix G. We also note that even the "brute-force" method works better than the power law (see Table 1 for comparison) thus demonstrating the strength of the piecewise power law.

Quality of predictions of the meta-model. We compare the predictions of the meta-model against the ground-truth (GT) N^* for all classification and detection tasks in Figure 2 (left). We normalize the values by the number of classes in the dataset to plot them on a similar scale. Most predictions lie within $[1/3N^*, 3N^*]$. We observe that despite large errors in predictions, the meta-model performs better compared to the baselines in Table 1 because the piecewise power law has high tolerance to the errors in the switch point N. To demonstrate this, we evaluate two more choices of N corresponding to $\{1/3N^*, 3N^*\}$ and compare the mean prediction error in Figure 2 (right) (see an elaborate comparison in Appendix H). Both of them perform better than the power law on most datasets.

Adaptability of the meta-model. For a given dataset, we expect the training statistics on the subsets to change if we finetune on the subsets differently, for example if we choose a different network architecture. Hence the learning curves in these cases are also expected to be different. In Figure 3, we show an example where the learning curve starts to exhibit linear behavior for a smaller N with ViT-B/16 as compared to ResNet-18 on CUB200. The meta-model adapts to this change and predicts a smaller switching point for ViT-B/16 as compared to ResNet-18.

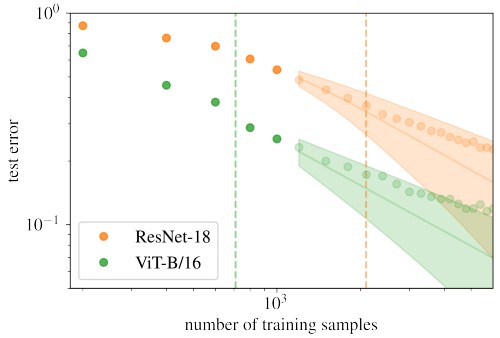


Figure 3. The meta-model adapts to different inputs for the same subsets finetuned with different model architectures. The piecewise power law is fit on the first 5 points (denoted in dark) and evaluated on the remaining points (denoted in light). Solid lines denote the mean prediction and the confidence bound by the shaded band. Vertical lines denote the predicted switch points.

5. Conclusion

Learning curves are better modeled by a non-linear function in the few-shot regime and a linear function in the high-shot regime. The widely used power law best explains only the high-shot regime of the curve. Our results show that by modeling both regimes differently with a piecewise power law (PPL) leads to lower extrapolation error on average. Further, using the confidence bound of the PPL prevents large data estimation errors to reach a target performance. However, the transition between the non-linear and linear regimes varies depending on the dataset, initializations, and even network architectures. We find that a meta learning based approach that uses statistics of the learning curves is useful to estimate the transition point. Learning curves could exhibit other phenomena such as double descent and saturation in the very high data regimes. We do not observe these phenomena with datasets used in this paper. Potentially, the proposed PPL can be extended to model double descent and saturation. We leave that for future work.

References

[1] Samira Abnar, Mostafa Dehghani, Behnam Neyshabur, and Hanie Sedghi. Exploring the limits of large scale pre-training. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. 2

[2] Ibrahim Alabdulmohsin, Behnam Neyshabur, and Xiaohua Zhai. Revisiting neural scaling laws in language and vision. *arXiv preprint arXiv:2209.06640*, 2022. 2

[3] Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *arXiv preprint arXiv:2102.06701*, 2021. 1, 2

[4] Sara Beery, Grant van Horn, and Pietro Perona. Recognition in terra incognita. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 1

[5] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 4

[6] Ethan Caballero, Kshitij Gupta, Irina Rish, and David Krueger. Broken neural scaling laws. *arXiv preprint arXiv:2210.14891*, 2022. 2

[7] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613, 2014. 5

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[9] Corinna Cortes, Lawrence D Jackel, Sara Solla, Vladimir Vapnik, and John Denker. Learning curves: Asymptotic values and rate of convergence. *Advances in Neural Information Processing Systems (NeurIPS)*, 1993. 1, 2, 3, 4, 5, 6, 11, 13, 16

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 2, 11

[11] Henri P. Gavin. The levenberg-marquardt method for nonlinear least squares curve-fitting problems ©. 2013. 3

[12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 5

[13] G. Griffin, AD. Holub, and Pietro Perona. The caltech 256. 5

[14] Nicolai Häni, Pravakar Roy, and Volkan Isler. MinneApple: a benchmark dataset for apple detection and segmentation. *IEEE Robotics and Automation Letters*, 5(2):852–858, 2020. 5

[15] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 11

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 7, 11

[17] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 204–207. IEEE, 2018. 5

[18] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019. 5

[19] 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. 1, 2

[20] Derek Hoiem, Tanmay Gupta, Zhizhong Li, and Michal Shlapentokh-Rothman. Learning curves for analysis of deep networks. In *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2021. 1, 2

[21] Naoto Inoue, Ryosuke Furuta, Toshihiko Yamasaki, and Kiyoharu Aizawa. Cross-domain weakly-supervised object detection through progressive domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5

[22] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011. 5

[23] Alexander Kolesnikov, Alexey Dosovitskiy, Dirk Weissenborn, Georg Heigold, Jakob Uszkoreit, Lucas Beyer, Matthias Minderer, Mostafa Dehghani, Neil Houlsby, Sylvain Gelly, Thomas Unterthiner, and Xiaohua Zhai. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. 7, 11

[24] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 5

[25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5

[26] Kibok Lee, Hao Yang, Satyaki Chakraborty, Zhaowei Cai, Gurumurthy Swaminathan, Avinash Ravichandran, and Onkar Dabeer. Rethinking few-shot object detection on a multi-domain benchmark. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 5, 11

[27] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. *arXiv preprint arXiv:1405.0312*, 2014. 2, 11

[28] Chongwei Liu, Haojie Li, Shuchang Wang, Ming Zhu, Dong Wang, Xin Fan, and Zhihui Wang. A dataset and benchmark of underwater object detection for robot picking. In *Proceedings of the IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, 2021. 5

[29] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. 11

[30] 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 I need? Estimating requirements for downstream tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3, 5, 6, 7, 11, 13, 16

[31] Rafid Mahmood, James Lucas, Jose M Alvarez, Sanja Fidler, and Marc T Law. Optimizing data collection for machine learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2

[32] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013. 5

[33] Caijing Miao, Lingxi Xie, Fang Wan, Chi Su, Hongye Liu, Jianbin Jiao, and Qixiang Ye. Sixray: A large-scale security inspection x-ray benchmark for prohibited item discovery in overlapping images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 5

[34] Thomas Moreau, Mathurin Massias, Alexandre Gramfort, Pierre Ablin, Pierre-Antoine Bannier, Benjamin Charlier, Mathieu Dagréou, Tom Dupré la Tour, Ghislain Durif, Cassio F. Dantas, Quentin Klopfenstein, Johan Larsson, En Lai, Tanguy Lefort, Benoit Malézieux, Badr Moufad, Binh T. Nguyen, Alain Rakotomamonjy, Zaccharie Ramzi, Joseph Salmon, and Samuel Vaiter. Benchopt: Reproducible, efficient and collaborative optimization benchmarks. 2022. 11

[35] Ernest Mwebaze, Timnit Gebru, Andrea Frome, Solomon Nsumba, and Jeremy Tusubira. icassava 2019 fine-grained visual categorization challenge. *arXiv preprint arXiv:1908.02900*, 2019. 5

[36] Maria-Elena Nilsback and Andrew Zisserman. A visual vocabulary for flower classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1447–1454, 2006. 5

[37] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 5

[38] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–420. IEEE, 2009. 5

[39] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proceedings of the Twenty-ninth Conference on Neural Information Processing Systems (NeurIPS)*, 2015. 11

[40] Jonathan S. Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. A constructive prediction of the generalization error across scales. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. 1, 2

[41] sgrpanchal31. table-detection-dataset. https://github.com/sgrpanchal31/table-detection-dataset, 2018. 5

[42] Shuai Shao, Zijian Zhao, Boxun Li, Tete Xiao, Gang Yu, Xiangyu Zhang, and Jian Sun. CrowdHuman: A benchmark for detecting human in a crowd. *arXiv preprint arXiv:1805.00123*, 2018. 5

[43] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. A dataset of 101 human action classes from videos in the wild. *Center for Research in Computer Vision*, 2(11), 2012. 5

[44] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the The International Conference on Computer Vision (ICCV)*, 2017. 2

[45] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 5

[46] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 11

[47] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Heng Fan, Qinghua Hu, and Haibin Ling. Detection and tracking meet drones challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 5

Appendix

A. Datasets

We use 16 classification and 10 detection datasets for evaluation, see statistics in Table 7. For classification, we randomly sample images according to subsets in Table 7 for training and we use the original test splits for evaluation. For detection, we choose natural k-shot sampling to construct subsets in Table 7 following the few-shot object detection (FSOD) setting [26].

B. Implementation details for training models

We train all models on single GPU with the following training recipe.

Finetuning on classification datasets. We use ImageNet [10] pre-trained weights to initialize models for all architectures - ResNet-18 [16], ResNet-50 [16], and ViT-B/16 [23], and train for 30 epochs with a batch size of 32. We perform HPO over 3 different learning rates (LR) $\in \{0.001, 0.005, 0.01\}$ (with the exception of ViT for which we tried a fixed LR of 0.0005) with SGD + momentum of 0.9 + weight decay of 0.0001 and LR decay by 0.1 at $\{15, 25\}$ epochs. We choose Top-1 Accuracy as the metric for performance $v(n)$.

Linear probing on classification datasets. We show experiments for ResNet-18 with ImageNet [10] pre-trained weights. We freeze the backbone and train a linear layer with batch norm [15]. We use a batch size of 32 and train the network for 30 epochs performing HPO over 3 different learning rates $\in \{0.001, 0.005, 0.01\}$ with SGD + momentum of 0.9 + weight decay of 0.0001 and LR decay by 0.1 at $\{15, 25\}$ epochs. We choose Top-1 Accuracy as the metric for performance $v(n)$.

Training from scratch on Cifar10/100. We show experiments for ResNet-18. We use the hyperparameter settings from [34] that achieves the state-of-the-art results on Cifar10. Specifically, we use a batch size of 128 and train for 100 epochs using a learning rate of 0.1 with cosine annealing [29] and linear warmup with SGD + momentum of 0.9 + weight decay of 0.0005. For the full dataset, we obtained Top-1 accuracy of 0.95±0.003 on Cifar10 and 0.78±0.002 on Cifar100.

Finetuning on detection datasets. We train a Faster R-CNN [39] detector with ResNet-50+FPN backbone with COCO [27] pre-trained weights. We use a batch size of $\min\{4, |\mathcal{S}_i|\}$ and train for 2000 iterations. We choose the best learning rate $\in \{0.0025, 0.005\}$ with decay by 0.1 at 1600 iterations. We tested the official implementation in Detectron2 [46] and default settings unless noted otherwise. We choose mean Average Precision (mAP) (averaged over IoU 0.5-0.95) as the metric for performance $v(n)$.

C. Extrapolation from few-shot to high-shot

In Section 4.1, we show the evaluation of the piecewise power law against two baselines: the power law [9] and arctan [30]. Here, we show results for two more baselines: algebraic [30] and logarithmic [30]. On average, the piecewise power law performs the best, followed by the power law and arctan on the classification tasks (see Table 8), and followed by the power law and logarithmic on the detection tasks (see Table 9).

D. Estimating data requirements to reach target performance

In Section 4.3, we show a visualization of data estimation error (2) for different choices of the target performance corresponding to $\{50, 60, 70, 80, 90\}\%$ data for only some datasets due to space limitation. Here, we provide complete results for all datasets with maximum steps $T = 5$ in Figure 4 and Figure 5, and additionally with maximum steps $T = 3$ in Figure 6 and Figure 7. The piecewise power law with "piecewise 5%" (referring to $\tau = 5\%$) demonstrates lower data estimation error compared to "powerlaw" in most cases both $T = 3$ and $T = 5$.

E. Comparison with the power law

In Section 3.2, we discuss the connection between the piecewise power law (3) and the power law (10). Specifically, the linear term of the PPL is equivalent to the power law with its asymptotic term set to zero (11). We reproduce the expression here

$$\log(1 - \hat{v}(n; \boldsymbol{\theta})) = \theta_1 + \theta_2 \log(n). \tag{13}$$

We refer to this predictor as "linear" since its parameters can simply be obtained by solving linear regression in the log-log space. In Table 10, we empirically show that "linear" predictor works better in mid-shot regime since the learning curve also exhibits linear behavior in the log-log space.

F. Generalization to training from scratch

We provide a comparison with algebraic [30] and logarithmic [30] to show generalization of the meta-model trained on finetuning ResNet-18 to training ResNet-18 from scratch in Table 11.

G. Comparison of meta-model with baselines

In the first ablation study in Section 4.5, we compare the meta-model to two different baselines, namely (1) "linear" baseline (same as (11)) that uses $N = n_1$ in the piecewise power law, and (2) "brute-force" baseline that greedily optimizes N based on the available data samples $\{n_i, v(n_i)\}_{i=1}^{5}$. We show the results in Table 12. We observe that different methods work better for different tasks but on average the "meta-model" works best reducing the average mean prediction error by 21.6% and 19.1% on ResNet-18 and ResNet-50, respectively, compared to the "brute-force" (next best) method.

Table 7. Datasets used for experiments with the sizes of subsets used for fitting and evaluation in the few-shot regime.

CLASSIFICATION				
	# classes	# train samples in largest subset	fitting (samples per class)	evaluation (% of full data)
Caltech256	257	15418	{1, 2, 3, 4, 5}	{10, 15, ..., 100}
Cifar10	10	50000	{5, 10, 15, 20, 25}	{10, 15, ..., 100}
Cifar100	100	50000	{1, 2, 3, 4, 5}	{10, 15, ..., 100}
CUB200	200	5994	{1, 2, 3, 4, 5}	{10, 15, ..., 100}
Decathlon Aircraft	100	3334	{1, 2, 3, 4, 5}	{10, 15, ..., 100}
Decathlon DTD	47	1880	{1, 2, 3, 4, 5}	{10, 15, ..., 100}
Decathlon Flowers	102	1020	{1, 2, 3, 4, 5}	{10, 15, ..., 100}
Decathlon UCF101	101	7585	{1, 2, 3, 4, 5}	{10, 15, ..., 100}
EuroSAT	10	20250	{5, 10, 15, 20, 25}	{10, 15, ..., 100}
FGVC Aircrafts	100	6667	{1, 2, 3, 4, 5}	{10, 15, ..., 100}
iCassava	5	4242	{10, 15, 20, 25, 30}	{10, 15, ..., 100}
MIT-67	67	5360	{1, 2, 3, 4, 5}	{10, 15, ..., 100}
Oxford Flowers	102	1020	{1, 2, 3, 4, 5}	{10, 15, ..., 100}
Oxford Pets	37	3680	{1, 2, 3, 4, 5}	{10, 15, ..., 100}
Stanford Cars	195	8144	{1, 2, 3, 4, 5}	{10, 15, ..., 100}
Stanford Dogs	120	12000	{1, 2, 3, 4, 5}	{10, 15, ..., 100}
DETECTION				
	# classes	# train samples in largest subset	fitting (samples per class)	evaluation (samples per class)
Cityscapes	8	800	{1, 5, 10, 15, 20}	{25, 30, ..., 100}
Comic	6	600	{1, 5, 10, 15, 20}	{25, 30, ..., 100}
CrowdHuman	2	200	{1, 5, 10, 15, 20}	{25, 30, ..., 100}
DUO	4	400	{1, 5, 10, 15, 20}	{25, 30, ..., 100}
KITTI	4	400	{1, 5, 10, 15, 20}	{25, 30, ..., 100}
MinneApple	1	100	{1, 5, 10, 15, 20}	{25, 30, ..., 100}
SIXray	5	500	{1, 5, 10, 15, 20}	{25, 30, ..., 100}
table-detection	1	100	{1, 5, 10, 15, 20}	{25, 30, ..., 100}
VisDrone	10	1000	{1, 5, 10, 15, 20}	{25, 30, ..., 100}
Watercolor	6	600	{1, 5, 10, 15, 20}	{25, 30, ..., 100}

H. Quality of predictions of meta-model

We provide results to support the second ablation study in Section 4.5. We observe that the piecewise power law has high tolerance to the errors in the switch point N. To demonstrate this, we evaluate two more choices of N corresponding to $\{1/3N^*, 3N^*\}$ and compare the mean prediction error in Table 13. Both of them perform better than the power law on most datasets.

Table 8. Mean prediction error $\mathcal{E}_{\text{perf}}$ (1) for extrapolating performance from few-shot to high-shot. Piecewise GT denotes the upper bound obtained with the piecewise model.

CLASSIFICATION						
	powerlaw [9]	**algebraic** [30]	**arctan** [30]	**logarithmic** [30]	**piecewise (ours)**	**piecewise (GT)**
Caltech256	10.3±3.6	9.1±5.3	3.0±1.5	12.3±3.5	**2.0±0.9**	1.2±0.5
Cifar10	6.7±2.2	7.6±3.3	6.0±6.2	5.2±0.1	**0.9±0.5**	0.5±0.4
Cifar100	6.5±3.1	22.5±0.1	17.2±7.3	22.2±0.4	**6.1±3.5**	5.3±3.8
CUB200	**2.6±0.3**	18.5±0.9	14.1±3.9	16.8±1.3	4.0±0.1	0.7±0.1
Decathlon Aircraft	18.0±1.8	17.8±13.8	23.5±15.9	11.5±2.1	**11.1±4.2**	11.1±4.1
Decathlon DTD	3.2±1.9	5.3±1.2	4.7±2.5	**3.2±0.9**	5.6±1.7	2.1±1.1
Decathlon Flowers	**1.0±0.3**	1.1±0.4	1.5±0.3	1.2±0.5	2.0±0.3	1.1±0.0
Decathlon UCF101	14.5±1.9	16.5±14.6	15.5±4.9	12.3±10.7	**4.1±3.0**	4.1±2.9
EuroSAT	2.6±0.6	4.3±3.2	4.2±2.4	2.1±0.2	**0.9±0.2**	0.9±0.2
FGVC Aircrafts	25.8±1.6	18.1±2.4	**9.2±7.4**	10.5±6.7	19.1±1.4	11.1±1.8
iCassava	9.2±6.7	12.4±7.9	14.6±4.8	12.2±6.6	**6.9±2.4**	6.9±2.4
MIT-67	**4.2±1.7**	15.8±6.4	8.2±5.3	11.7±6.5	4.3±2.5	3.9±2.3
Oxford Flowers	1.5±0.4	1.6±0.2	1.5±0.3	1.4±0.5	**1.2±0.3**	1.1±0.4
Oxford Pets	9.2±0.4	8.4±0.7	**1.1±0.5**	9.7±0.2	5.6±0.8	1.7±0.5
Stanford Cars	26.4±1.3	18.8±0.4	17.6±2.9	**14.2±0.6**	17.3±2.7	7.7±3.3
Stanford Dogs	6.1±5.4	5.2±3.1	7.8±2.7	12.5±2.1	**2.3±1.0**	1.2±0.1
AVERAGE	9.2±2.1	11.4±4.0	9.3±4.3	9.9±2.7	**5.8±1.6**	3.8±1.5

Table 9. Mean prediction error $\mathcal{E}_{\text{perf}}$ (1) for extrapolating performance from few-shot to high-shot. Piecewise GT denotes the upper bound obtained with the piecewise model.

DETECTION						
	powerlaw [9]	**algebraic** [30]	**arctan** [30]	**logarithmic** [30]	**piecewise (ours)**	**piecewise (GT)**
Cityscapes	1.3±0.8	1.2±0.5	1.5±0.6	1.1±0.8	**1.1±0.5**	0.9±0.6
Comic	4.4±2.9	10.0±6.3	28.0±33.9	**3.9±1.7**	4.1±1.0	3.4±2.0
CrowdHuman	0.8±0.2	1.0±0.2	1.5±0.5	0.8±0.1	**0.7±0.3**	0.5±0.3
DUO	3.9±1.8	5.9±4.4	4.5±1.6	2.9±2.2	**2.4±0.5**	1.8±0.9
KITTI	2.6±2.1	3.3±2.0	**1.5±1.0**	2.2±1.1	1.6±0.7	1.5±1.4
MinneApple	4.7±2.3	1.2±0.5	**1.1±0.3**	1.3±0.6	1.9±1.0	0.6±0.1
SIXray	6.9±0.9	28.3±17.7	8.3±9.9	9.6±9.9	**2.7±2.7**	2.4±1.1
table-detection	5.9±2.7	8.5±5.1	7.8±2.2	6.3±3.7	**5.5±0.8**	5.5±2.2
VisDrone	**0.3±0.1**	1.0±0.4	0.7±0.3	0.9±0.3	0.8±0.3	0.4±0.1
Watercolor	5.2±1.5	19.4±22.1	6.7±2.7	6.7±3.0	**3.2±1.4**	3.1±1.7
AVERAGE	3.6±1.5	8.0±5.9	6.2±5.3	3.6±2.3	**2.4±0.9**	2.0±1.0

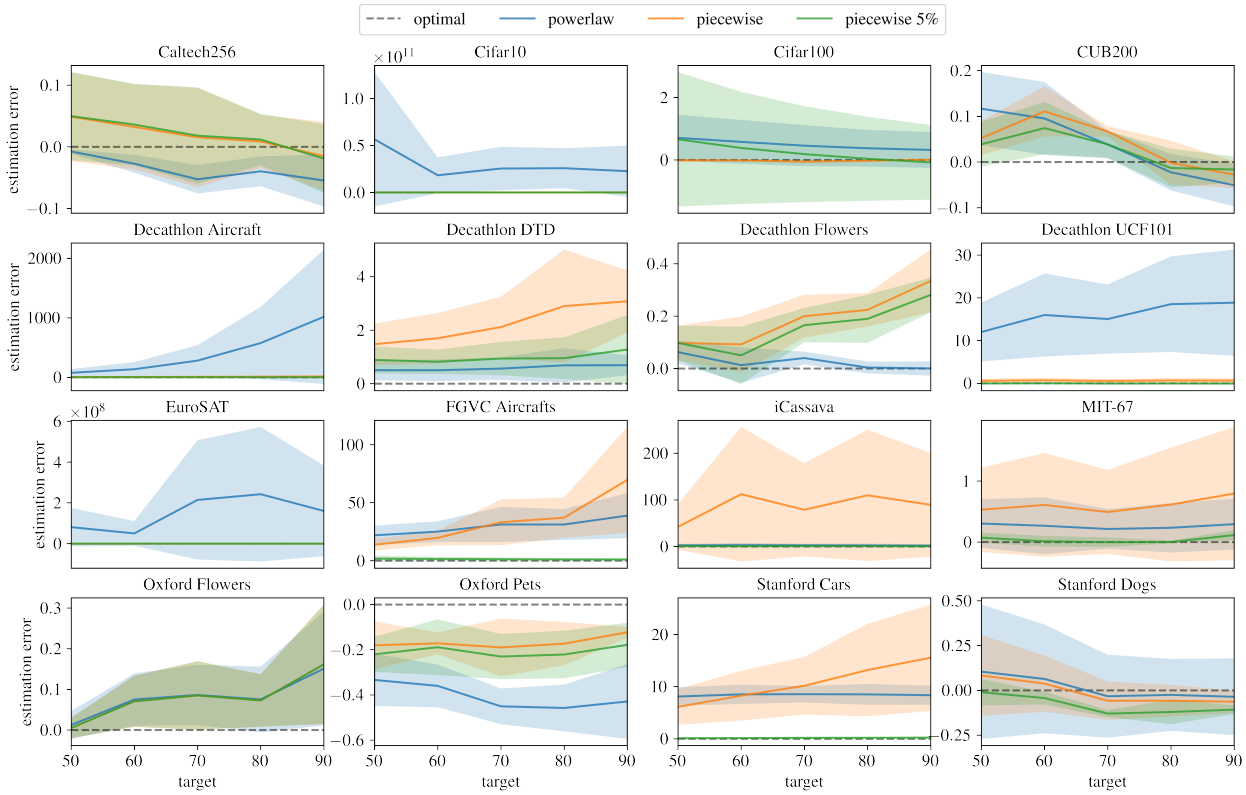


Figure 4. CLASSIFICATION $T = 5$: Data estimation error $\mathcal{E}_{\texttt{data}}$ (2) to reach different performance targets obtained by using {50, 60, 70, 80, 90}% of the full dataset.

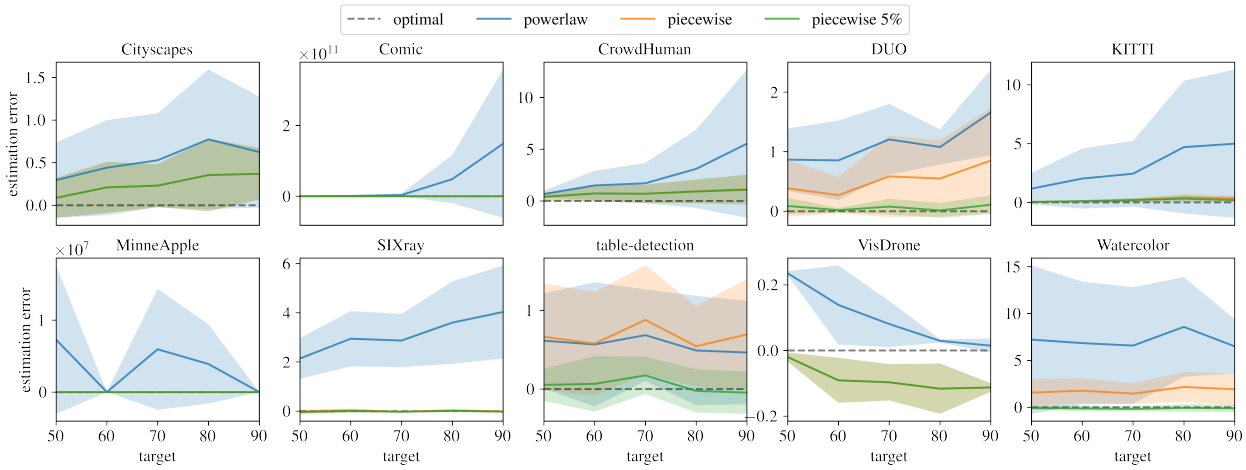


Figure 5. DETECTION $T = 5$: Data estimation error $\mathcal{E}_{\texttt{data}}$ (2) to reach different performance targets obtained by using {50, 60, 70, 80, 90}% of the full dataset.

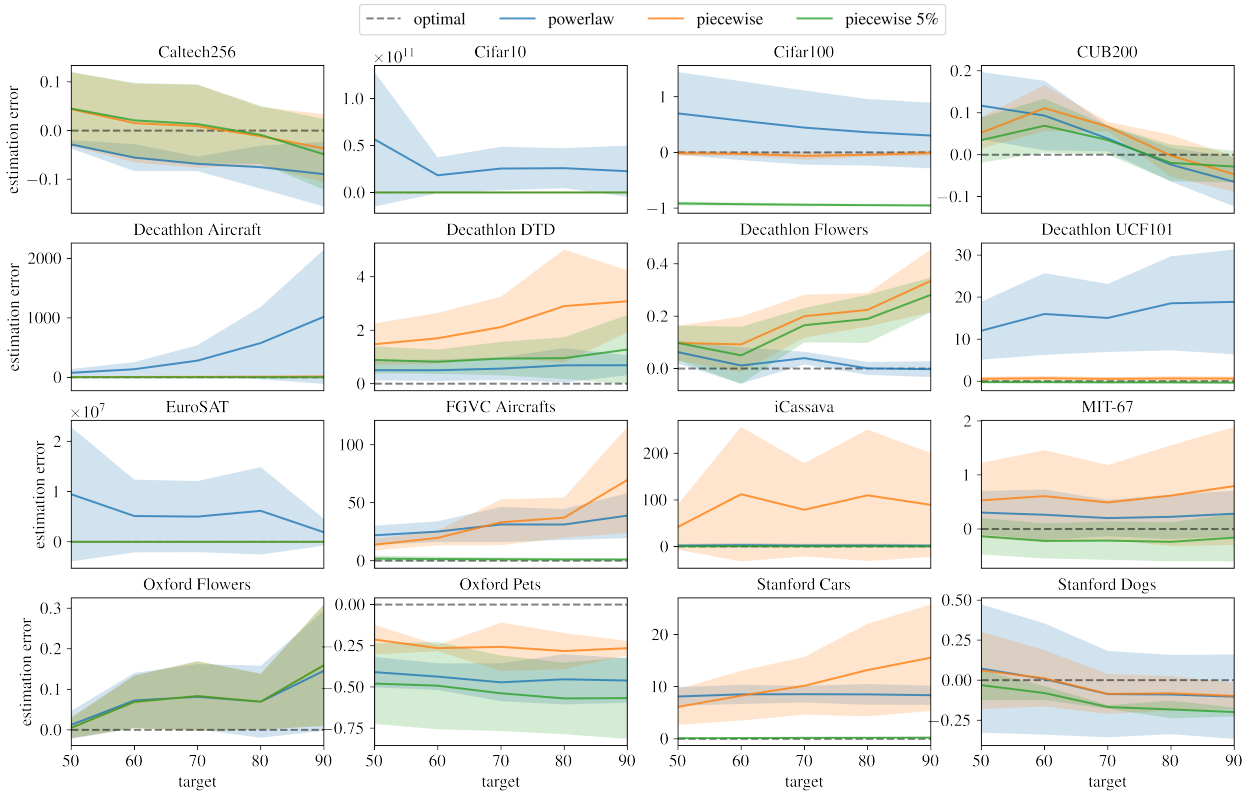


Figure 6. CLASSIFICATION $T = 3$: Data estimation error $\mathcal{E}_{\texttt{data}}$ (2) to reach different performance targets obtained by using {50, 60, 70, 80, 90}% of the full dataset.

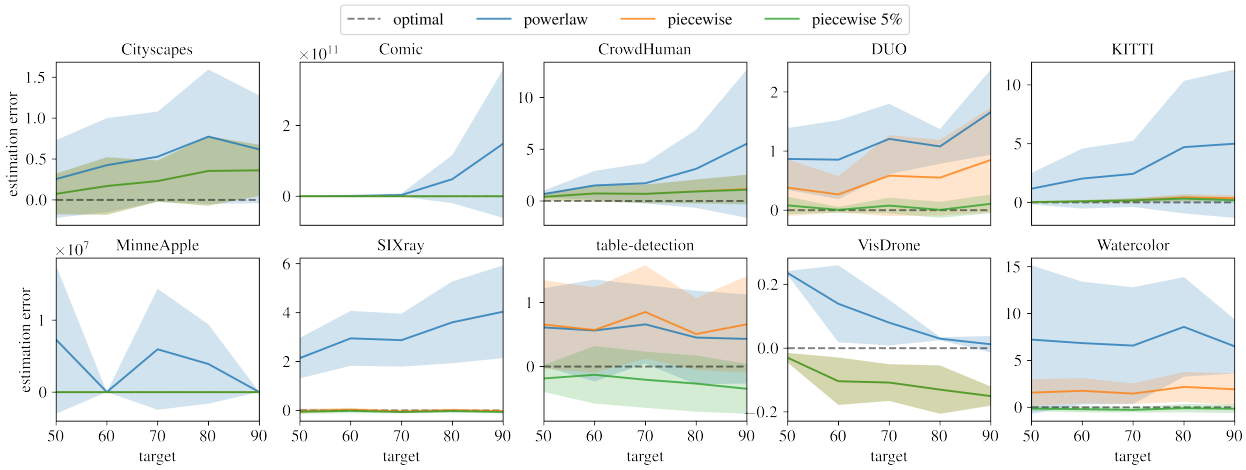


Figure 7. DETECTION $T = 3$: Data estimation error $\mathcal{E}_{\texttt{data}}$ (2) to reach different performance targets obtained by using {50, 60, 70, 80, 90}% of the full dataset.

Table 10. Extrapolating performance for classification tasks. Between the "powerlaw" and the "linear", we mark the better performing predictor in bold.

CLASSIFICATION									
	few-shot			**mid-shot 30%**			**mid-shot 50%**		
	powerlaw	**linear**	**piecewise**	**powerlaw**	**linear**	**piecewise**	**powerlaw**	**linear**	**piecewise**
Caltech256	10.3±3.6	**1.2±0.5**	2.0±0.9	0.8±0.6	**0.6±0.3**	0.6±0.3	**0.5±0.2**	**0.5±0.2**	0.3±0.0
Cifar10	6.7±2.2	**0.9±0.5**	0.9±0.5	0.3±0.3	**0.1±0.0**	0.1±0.0	0.3±0.1	**0.1±0.0**	0.1±0.0
Cifar100	**6.5±3.1**	19.0±3.0	6.1±3.5	1.1±0.8	**0.6±0.1**	0.6±0.1	**0.3±0.1**	**0.3±0.1**	0.3±0.1
CUB200	**2.6±0.3**	8.8±0.9	4.0±0.1	4.5±2.4	**0.8±0.2**	2.5±1.3	1.6±0.6	**0.7±0.2**	0.9±0.0
Decathlon Aircraft	**18.0±1.8**	18.5±1.7	11.1±4.2	**8.5±1.6**	10.0±1.4	4.1±2.0	**4.2±0.3**	5.9±0.2	1.5±1.1
Decathlon DTD	**3.2±1.9**	5.6±1.7	5.6±1.7	**1.2±0.4**	1.7±0.8	1.7±0.8	1.4±0.8	**1.3±0.4**	1.3±0.4
Decathlon Flowers	**1.0±0.3**	3.0±0.3	2.0±0.3	**1.4±0.4**	5.9±0.8	2.6±1.7	**1.0±0.3**	3.4±0.9	1.8±0.3
Decathlon UCF101	**14.5±1.9**	16.1±1.6	4.1±3.0	2.7±1.5	**0.9±0.5**	2.2±1.2	1.0±0.4	**0.5±0.1**	0.8±0.3
EuroSAT	2.6±0.6	**0.9±0.2**	0.9±0.2	0.3±0.2	**0.1±0.0**	0.1±0.0	**0.2±0.1**	0.1±0.0	0.1±0.0
FGVC Aircrafts	**25.8±1.6**	28.9±1.2	19.1±1.4	6.4±4.7	**4.0±0.5**	2.0±0.8	2.6±0.9	**2.2±0.5**	0.9±0.4
iCassava	9.2±6.7	**6.9±2.4**	6.9±2.4	2.4±0.7	**1.2±0.5**	1.2±0.5	0.5±0.3	**0.5±0.1**	0.5±0.1
MIT-67	**4.2±1.7**	7.3±1.6	4.3±2.5	2.3±1.3	**0.8±0.2**	1.1±0.4	1.2±0.7	**0.4±0.0**	0.9±0.5
Oxford Flowers	**1.5±0.4**	1.6±0.4	1.2±0.3	3.6±1.8	**3.4±0.6**	1.9±0.7	**0.6±0.3**	2.0±0.8	0.7±0.5
Oxford Pets	9.2±0.4	**1.7±0.5**	5.6±0.8	2.2±0.9	**1.1±0.2**	1.1±0.2	1.2±0.8	**0.7±0.4**	0.7±0.4
Stanford Cars	**26.4±1.3**	30.8±1.1	17.3±2.7	9.7±0.1	**3.4±0.1**	1.1±0.4	4.3±0.9	**1.1±0.1**	0.4±0.1
Stanford Dogs	6.1±5.4	**1.2±0.1**	2.3±1.0	3.1±2.0	**2.1±0.3**	2.1±0.3	**1.2±1.2**	1.6±0.3	1.6±0.3
AVERAGE	**9.2±2.1**	9.5±1.1	5.8±1.6	3.2±1.2	**2.3±0.4**	1.6±0.7	1.4±0.5	**1.3±0.3**	0.8±0.3

Table 11. Generalization of the meta-model trained on *finetuning ResNet-18* to *training ResNet-18 from scratch*.

	powerlaw [9]	**algebraic** [30]	**arctan** [30]	**logarithmic** [30]	**piecewise (ours)**
Cifar10 [30]	39.02±20.3	33.63±22.1	7.98±7.1	32.28±13.1	-
Cifar10 (ours)	0.9±0.8	1.3±0.5	2.9±0.7	5.8±0.3	**0.3±0.1**
Cifar100 [30]	34.98±35.1	26.29±16.8	13.3±5.3	17.25±21.8	-
Cifar100 (ours)	4.0±0.5	25.1±1.0	19.4±5.3	23.5±1.5	**2.5±0.3**

Table 12. Comparison of performance of the meta-model against the baselines, measured by the mean prediction error $\mathcal{E}_{\text{perf}}$ (1).

CLASSIFICATION						
	ResNet-18			**ResNet-50**		
	linear	**brute-force**	**meta-model**	**linear**	**brute-force**	**meta-model**
Caltech256	**1.2±0.5**	2.4±1.3	2.0±0.9	**0.7±0.2**	**0.7±0.2**	1.1±0.8
Cifar10	0.9±0.5	**0.5±0.1**	0.9±0.5	**1.9±1.4**	6.5±6.4	**1.9±1.4**
Cifar10	19.0±3.0	**5.3±3.8**	6.1±3.5	7.7±0.2	**1.2±0.3**	11.8±0.9
CUB200	8.8±0.9	**0.8±0.2**	4.0±0.1	6.5±1.3	**1.9±1.2**	4.0±1.6
Decathlon Aircraft	18.5±1.7	14.8±2.5	**11.1±4.2**	21.0±0.5	14.8±0.7	**10.6±1.1**
Decathlon DTD	5.6±1.7	**4.9±0.9**	5.6±1.7	5.4±1.3	**3.4±0.9**	5.4±1.3
Decathlon Flowers	3.0±0.3	**1.5±0.0**	2.0±0.3	3.0±0.4	**2.6±0.6**	**2.6±0.7**
Decathlon UCF101	16.1±1.6	12.9±3.7	**4.1±3.0**	16.7±0.8	8.1±1.7	**3.5±1.9**
EuroSAT	**0.9±0.2**	3.0±3.3	**0.9±0.2**	**0.3±0.1**	2.8±3.3	**0.3±0.1**
FGVC Aircrafts	28.9±1.2	20.1±2.2	**19.1±1.4**	31.3±1.2	20.2±2.4	**15.2±5.0**
iCassava	**6.9±2.4**	25.5±18.3	**6.9±2.4**	**1.8±0.7**	19.3±24.8	**1.8±0.7**
MIT-67	7.3±1.6	4.3±3.3	**4.3±2.5**	4.0±1.4	**2.5±2.6**	5.9±2.9
Oxford Flowers	1.6±0.4	**1.2±0.3**	**1.2±0.3**	1.9±0.5	1.2±0.4	**1.1±0.4**
Oxford Pets	**1.7±0.5**	2.4±1.0	5.6±0.8	2.5±0.0	2.5±0.0	**2.0±0.5**
Stanford Cars	30.8±1.1	**16.2±2.8**	17.3±2.7	30.2±0.9	**13.6±1.4**	14.7±1.4
Stanford Dogs	**1.2±0.1**	2.0±0.8	2.3±1.0	6.9±0.4	6.9±0.4	**6.8±0.5**
AVERAGE	9.5±1.1	7.4±2.8	**5.8±1.6**	8.9±0.7	6.8±3.0	**5.5±1.3**

Table 13. Effect on performance of choosing different switch points in the piecewise power law, measured by the mean prediction error $\mathcal{E}_{\text{perf}}$ (1).

CLASSIFICATION					
	powerlaw	**piecewise** meta-model	**piecewise** N^*	**piecewise** $3\times N^*$	**piecewise** $1/3\times N^*$
Caltech256	10.3±3.6	2.0±0.9	1.2±0.5	2.2±1.2	1.2±0.5
Cifar10	6.7±2.2	0.9±0.5	0.5±0.4	2.3±0.3	0.9±0.5
Cifar10	6.5±3.1	6.1±3.5	5.3±3.8	4.8±3.1	13.0±3.6
CUB200	2.6±0.3	4.0±0.1	0.7±0.1	4.4±0.2	4.4±0.8
Decathlon Aircraft	18.0±1.8	11.1±4.2	11.1±4.1	11.1±4.2	13.0±3.5
Decathlon DTD	3.2±1.9	5.6±1.7	2.1±1.1	2.4±1.0	3.1±1.6
Decathlon Flowers	1.0±0.3	2.0±0.3	1.1±0.0	1.1±0.0	2.0±0.3
Decathlon UCF101	14.5±1.9	4.1±3.0	4.1±2.9	4.1±2.8	5.3±3.4
EuroSAT	2.6±0.6	0.9±0.2	0.9±0.2	1.3±0.7	0.9±0.2
FGVC Aircrafts	25.8±1.6	19.1±1.4	11.1±1.8	11.1±1.8	12.5±1.7
iCassava	9.2±6.7	6.9±2.4	6.9±2.4	30.7±19.7	6.9±2.4
MIT-67	4.2±1.7	4.3±2.5	3.9±2.3	5.4±2.2	5.0±3.3
Oxford Flowers	1.5±0.4	1.2±0.3	1.1±0.4	1.2±0.3	1.6±0.4
Oxford Pets	9.2±0.4	5.6±0.8	1.7±0.5	2.9±0.6	1.7±0.5
Stanford Cars	26.4±1.3	17.3±2.7	7.7±3.3	7.7±3.3	9.7±3.3
Stanford Dogs	6.1±5.4	2.3±1.0	1.2±0.1	1.6±0.4	1.2±0.1
AVERAGE	9.2±2.1	5.8±1.6	3.8±1.5	5.9±2.6	5.2±1.6