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.

Fine-tuning 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.

Fine-tuning 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 fine-tuning 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

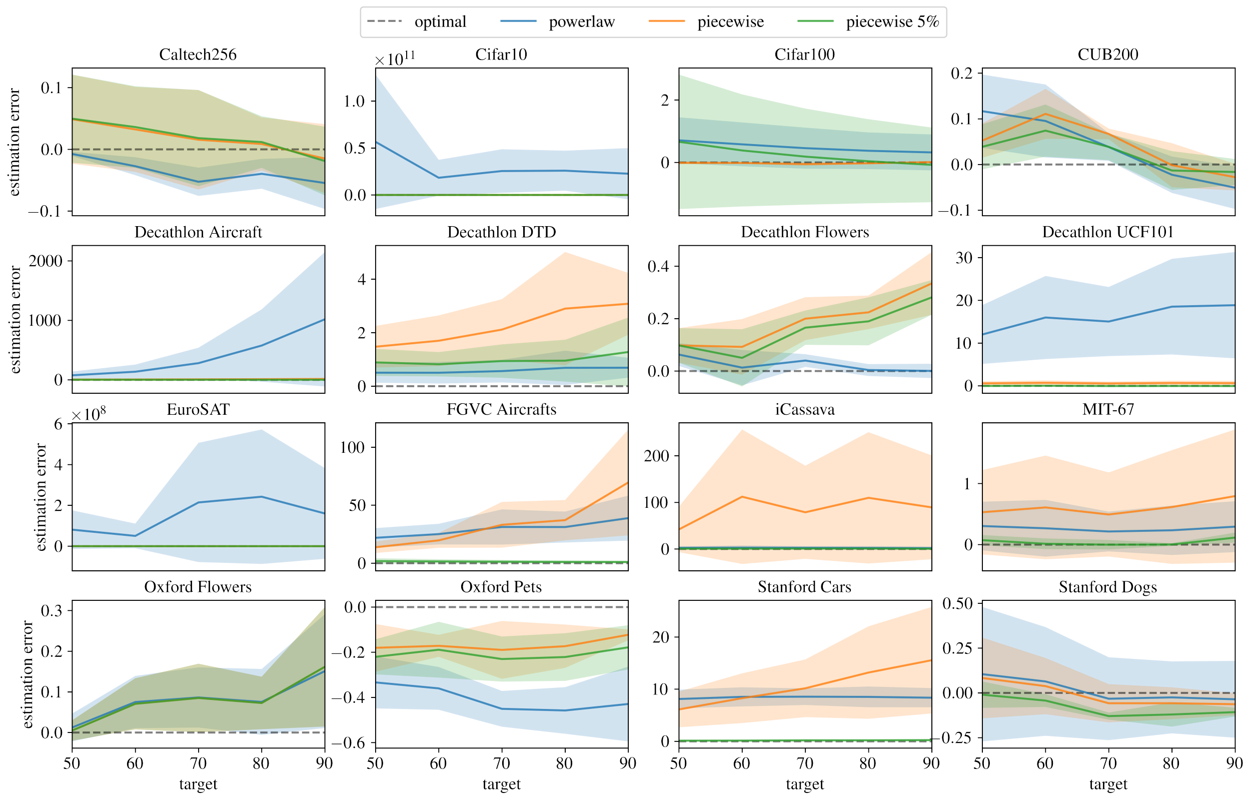

Figure 4. CLASSIFICATION $T = 5$: Data estimation error $\mathcal{E}_{\text{data}}$ (2) to reach different performance targets obtained by using {50, 60, 70, 80, 90}% of the full dataset.

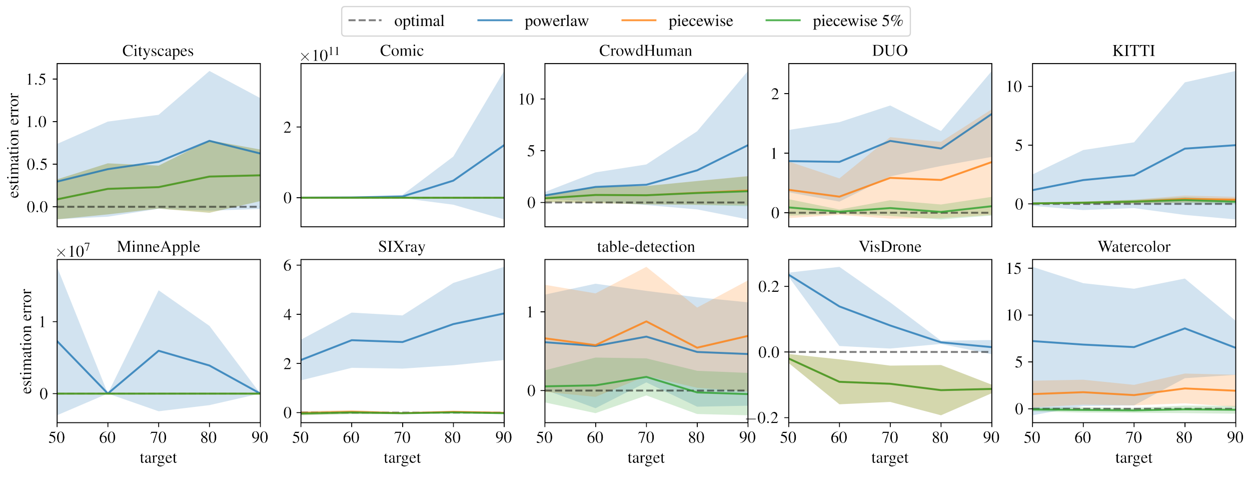

Figure 5. DETECTION $T = 5$: Data estimation error $\mathcal{E}_{\text{data}}$ (2) to reach different performance targets obtained by using {50, 60, 70, 80, 90}% of the full dataset.

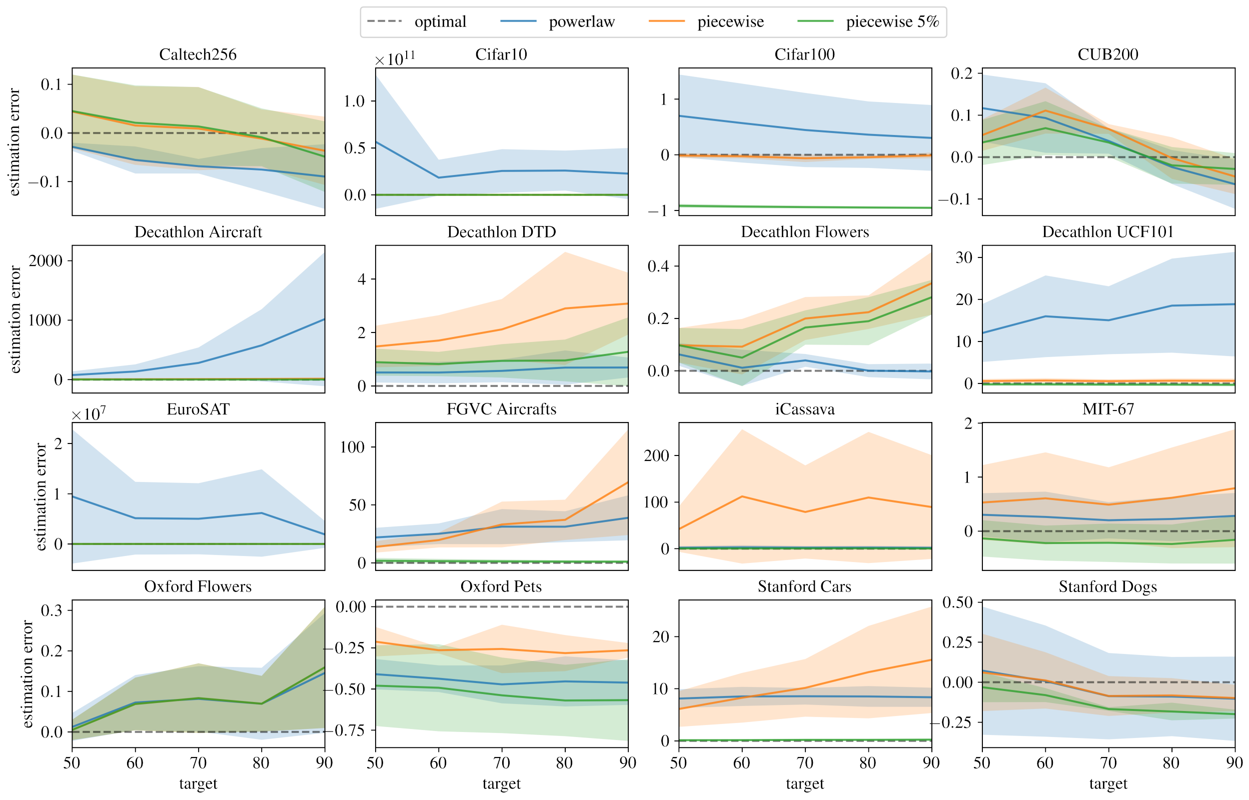

Figure 6. CLASSIFICATION $T = 3$: Data estimation error $\mathcal{E}_{\texttt{data}}$ (2) to reach different performance targets obtained by using {50, 60, 70, 80, 90}% of the full dataset.

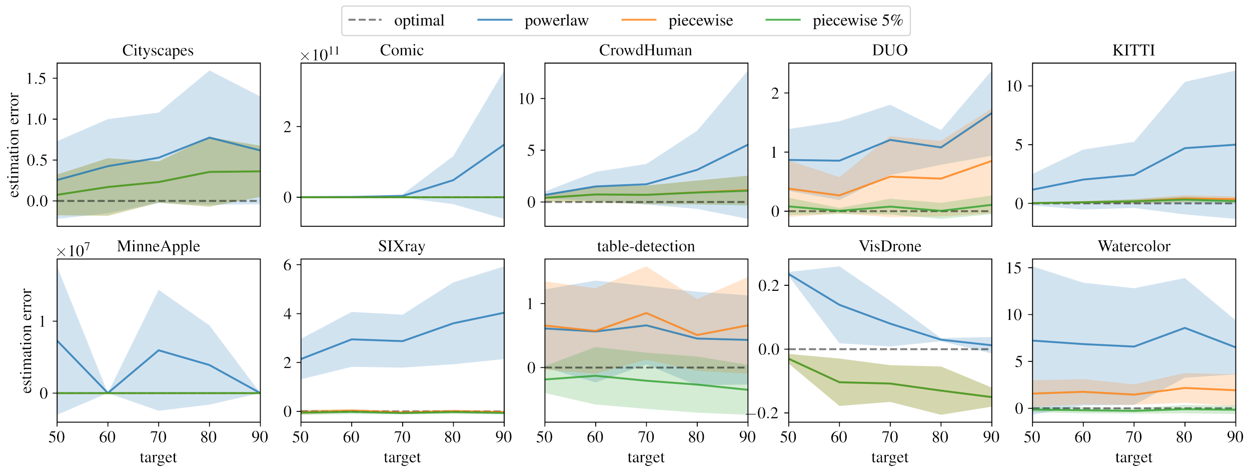

Figure 7. DETECTION $T = 3$: Data estimation error $\mathcal{E}_{\texttt{data}}$ (2) to reach different performance targets obtained by using {50, 60, 70, 80, 90}% of the full dataset.

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 $\mathcal{E}_{\mathrm{perf}}$ (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 $\mathcal{E}_{\mathrm{perf}}$ (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