Learning Curves for Analysis of Deep Networks: Appendix

A. Implementation Details

We use Pytorch-Lightning (Falcon, 2019) for our implementation with various architectures, weight initializations, data augmentation, and linear or fine-tuning optimization.

Training: We train models with images of size 224×224 for all experiments (to facilitate use of pretrained models) with a batch size of 64 (except for Wide-ResNet101 and Wide-ResNeXt101, where we use a batch size of 32 and performed one optimizer step every two batches). For each experiment setting, we conduct a learning rate search on a subset of the training data and choose the learning rate with the highest validation accuracy, and use it for all other subsets. We determine each fold's training schedule on a mini-train/mini-val split of 2:1 on the train set. Each time the mini-val error stops decreasing for some epochs ("patience"), we revert to the best epoch and decrease the learning rate to 10%, and we perform this twice. Then we use this optimal mini-train learning rate schedule and ending epoch to train on the whole fold. The patience is $\propto 1/\sqrt{n}$, and is 5 at the $n = 400$ samples/class for CIFAR100/Places365 and 15 at the largest training size for other smaller datasets. We use a weight decay value of 0.0001. We use the Ranger optimizer (Wright, 2019), which combines Rectified Adam (Liu et al., 2020), Look Ahead (Zhang et al., 2019), and Gradient Centralization (Yong et al., 2020). In early experiments, we found Ranger to lead to lower error and to reduce sensitivity of hyperparameters, compared to vanilla SGD or Adam (Kingma & Ba, 2015).

Backbone Architecture: We use the default Pytorch implementations of all of the following architectures: AlexNet (Krizhevsky et al., 2012), ResNet-18, ResNet-50, ResNet-101 (He et al., 2016), ResNeXt-50, ResNeXt-100 (Xie et al., 2017), VGG16_BN (Simonyan & Zisserman, 2015), Wide-ResNet-50, and Wide-ResNet-101 (Zagoruyko & Komodakis, 2016). For each architecture, we modify the last layer to match the same number of classes as the test dataset with Kaiming initialization (He et al., 2015).

Number of Training Examples: To compute learning curves for CIFAR and Places365, we vary the number of training examples per class, partition the train set, and train one model per partition. For CIFAR100 (Krizhevsky, 2012), we use $\{25, 50, 100, 200, 400\}$ training examples per class, and the number of models trained for each respectively is $\{16, 8, 4, 2, 1\}$. Similar to (Hestness et al., 2017), we find training sizes smaller than 25 samples per class are strongly influenced by bounded error and deviate from our model. For Places365 dataset, we use $\{25, 50, 100, 200, 400, 1600\}$ training examples per class and $\{16, 8, 4, 3, 3, 1\}$ models each. For other datasets (Fig. 10), we use $\{20\%, 40\%, 80\%\}$ of the full data and train $\{4, 2, 1\}$ models each. We hold out 20% of data from the original training set for testing (a validation set could also be used if available) to discourage meta-fitting on the test set. For example, we hold out 100 samples per class from the original CIFAR100 training set and perform hyperparameter selection and training on the remaining 400 samples.

Pretraining: When pretraining is used, we initialize models with pretrained weights learned through supervised training on ImageNet or Places365, or MOCO self-supervised training on ImageNet (He et al., 2020). Otherwise, weights are randomly initialized with Kaiming initialization.

Data Augmentation: For CIFAR, we pad by 4 pixels and use a random 32×32 crop (test without augmentation), and for Places365 we use random-sized crop (Szegedy et al., 2015) to 224×224 and random flipping (center crop 224×224 test time). For remaining datasets, we follow the pre-processing in Zhai et al. (2020) that produced the best results when training from scratch.

Linear vs. Fine-tuning: For "linear", we only train the final classification layer, with the other weights frozen to initialized values. All weights are trained when "fine-tuning".

B. User's Guide to Learning Curves

B.1. Uses for Learning Curves

- **Comparison**: When comparing two learners, measuring the error and data-reliance provides a better understanding of the differences than evaluating single-point error. We compare curves with e_N and β_N, rather than directly using the curve parameters, because they are more stable under data perturbations and do not depend on the parameterization, instead corresponding to error and rate of change about $n = N$. The difference $e_N - \beta_N$ can be used as a measure of large-sample performance.

- **Performance extrapolation**: A 10x increase in training data can require a large investment, sometimes millions of dollars. Learning curves can predict how much performance will improve with the additional data to judge whether the investment is worthwhile.

- **Model selection**: When much training data is available, architecture, hyperparameters, and losses can be designed and selected using a small subset of the data to minimize the extrapolated error of the full training set size. Higher-parameter models such as in (Kaplan et al., 2020) and (Rosenfeld et al., 2020) may be more useful as a mechanism to simultaneously select scale parameters and extrapolate performance, though fitting those models is much more computationally expensive due to the requirement of sampling error/loss at multiple scales and data sizes.

- **Hyperparameter validation**: A poor fitting learning curve (or one with γ far from -0.5) is an indication of poor choice of hyperparameters, as pointed out by (Hestness et al., 2017).

B.2. Estimating and Displaying Learning Curves

Use validation set: We recommend computing learning curves on a validation set, rather than a test set, according to best practice of performing a single evaluation on the test set for the final version of the algorithm. All of our experiments are on a validation set, which is carved from the official training set if necessary.

Generate at least four data points: In most of our experiments on CIFAR100, we train a 31 models: 1 on 400 images, 2 on 200 images, 4 on 100 images, 8 on 50 images, and 16 on 25 images. Each trained model provides one data point, the average validation error. In each case, the training data is partitioned so that the image sets within the same size are non-overlapping. Training multiple models at each size enables estimating the standard deviation for performing weighted least squares and producing confidence bounds. However, our experiments indicate that learning curves are highly stable, so a minimal experiment of training four models on the full, half, quarter, and eighth-size training set may be sufficient as part of a standard evaluation. See Fig. 9 It may be necessary to train more models if attempting to distinguish fine differences.

Set hyperparameters: The learning rate and learning schedule are key parameters to be set. We have not experimented with changes to weight decay, momentum, or other hyperparameters.

Fit learning curves: If more than one data point is available for the same training size, the standard deviation can be estimated. As described in Sec. 3, we recommend fitting a model of $\sigma_i^2 = \sigma_0^2 + \hat{\sigma}^2/n$, where σ_0^2. σ_0^2 is the variance due to randomness in initialization and optimization. The fitting is not highly sensitive to this parameter, so we recommend setting $\sigma_0^2 = 0.01$ and fitting $\hat{\sigma}$ to observations, since estimating both from experiments to generate a single learning curve introduces high variance and instability.

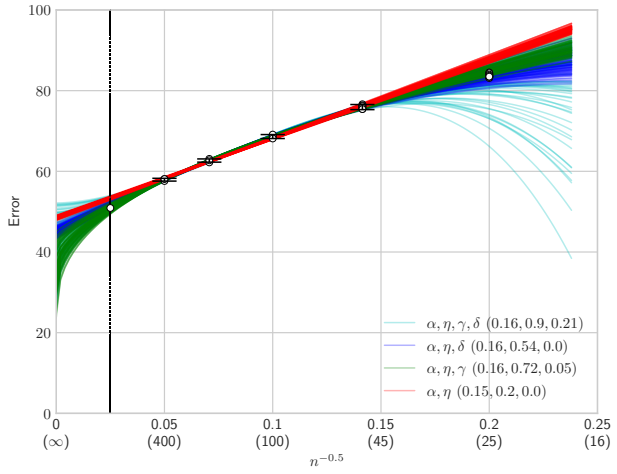

Figure 9: Stability under sparse measurements: Sampled learning curves for Places365 fine-tuned without pretraining are shown for four different learning curve parameterizations. In each case, means and standard deviations (shown by error bars) are estimated for $n = 50$, $n = 100$, $n = 200$, $n = 400$, using all the data points shown as white circles. Then, 100 times, we sample one point each from a Guassian distribution and fit a learning curve to the four points. In parantheses, the legend shows the standard deviation of e_N, β_N, and γ. Note that the parameterization of $\{\alpha, \eta, \gamma\}$ extrapolates best to lower and higher data sizes while still producing stable estimates of e_N and β_N. Asymptotic error, however, varies widely.

Display learning curves or parameters: As in this paper, learning curves can be plotted linearly with the x-axis as $n^{-0.5}$ and the y-axis as error. We choose this rather than log-linear because it reveals prediction of asymptotic error and yields a linear plot when $\gamma = -0.5$. Since space is often a premium, the learning curve parameters can be displayed instead, as illustrated in Table 2. Although γ is not useful for direct comparison, including it enables recovery of α, η, and γ to plot the original learning curve.

Table 2: Results: model_1 and model_2 have similar percent test error when training on the full set. Fitting a learning curve on the validation set, we see that model_2 has higher data-reliance, so may outperform for larger training sets. *This is a hypothetical example to illustrate use of learning curves in a table.*

	e_N	β_N	γ
model_1	25.3 %	4.6	-0.36
model_2	25.2 %	8.4	-0.47

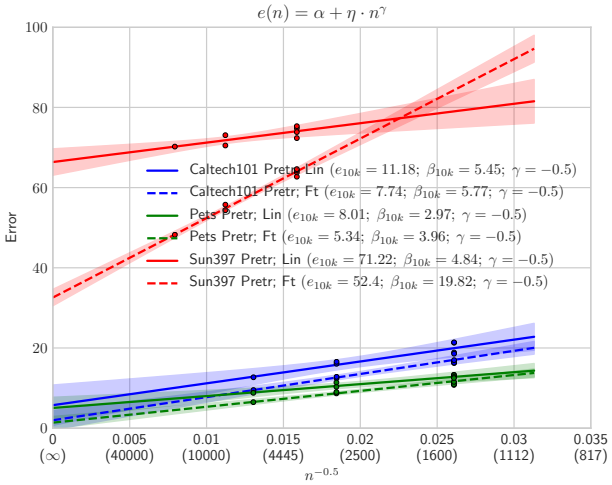

Figure 10: **Additional datasets**

C. Additional Results

Additional datasets: In Fig. 10, we verify that our learning curve model fits to multiple other datasets (chosen from natural tasks in (Zhai et al., 2020)), comparing fine-tuned vs. linear with Resnet-18. For these plots only, n is the total number of samples. The γ values are estimated from data, but the prior has more effect here due to fewer error measurements. We see fine-tuning consistently outperforms linear, though the difference is most dramatic for Sun397.

D. Table of Learning Curves

Table 3 shows experimental settings and fit parameters for learning curves under two parameterizations. We can see that similar e_{400} and β_{400} values are obtained when fixing $\gamma = -0.5$ and fitting to errors with only three training sizes (RMS difference in e_{400} and γ_{400} are 0.42 and 0.95, respectively). This means that learning curves can be fit and compared without training a large number of additional models.

Table 3: Experiment settings and parameters: We show the datasets, architectures, settings, and learning rate (set by mini-train/val) used to train and test our classifiers. Next, we show the parameters fit using the extended power law model $e(n) = \alpha + \eta n^{-\gamma}$. Next to that, we show the model resulting from setting $\gamma = -0.5$ and fitting to only the three training sizes with highest n.

								extended power law					$n^{-0.5}$ **linear fit to last 3 points**			
	dataset	**arch**	**# param**	**pretrain/init**	**fine-tune?**	**data aug?**	**lrnRate**	α	η	γ	e_{400}	β_{400}	α	η	e_{400}	β_{400}
PRETRAIN_IN2CIFAR																
No Pretr; Linear	CIFAR	Resnet-18	51K	Random	No	Yes	0.01	78.51	120.13	-0.84	79.29	1.32	78.06	26.12	79.36	1.31
No Pretr; Finetune	CIFAR	Resnet-18	11.7M	Random	Yes	Yes	0.01	5.68	259.29	-0.41	27.91	18.23	11.21	336.13	28.02	16.81
Pretr; Linear	CIFAR	Resnet-18	51K	ImageNet	No	Yes	0.0003	24.4	65.28	-0.35	32.42	5.61	27.33	102.16	32.44	5.11
Pretr; Finetune	CIFAR	Resnet-18	11.7M	ImageNet	Yes	Yes	0.001	12.48	194.19	-0.57	18.86	7.28	11.37	150.73	18.91	7.54
PRETRAIN_IN2PLACES																
No Pretr; Linear	Places	Resnet-18	187K	Random	No	Yes	0.03	91.84	19.13	-0.5	92.79	0.96	91.09	29.31	92.55	1.47
No Pretr; Finetune	Places	Resnet-18	11.7M	Random	Yes	Yes	0.001	33.16	117.45	-0.26	57.89	12.86	44.39	263.63	57.57	13.18
Pretr; Linear	Places	Resnet-18	187K	ImageNet	No	Yes	0.0003	54.43	53.39	-0.38	59.91	4.16	56.11	76.95	59.95	3.85
Pretr; Finetune	Places	Resnet-18	11.7M	ImageNet	Yes	Yes	0.0003	40.92	70.04	-0.28	54	7.33	44.82	174.69	53.55	8.73
PRETRAIN_IN_PLACES_MOCO2CIFAR																
No Pretr	CIFAR	Resnet-50	25.6M	Random	Yes	Yes	0.01	-2.23	243.44	-0.35	27.66	20.93	9.18	372.11	27.79	18.61
Pretr on Imagenet	CIFAR	Resnet-50	25.6M	ImageNet	Yes	Yes	0.001	15.11	178.61	-0.67	18.33	4.32	13.3	99.88	18.29	4.99
Pretr on Places	CIFAR	Resnet-50	25.6M	Places	Yes	Yes	0.001	-5.61	109.92	-0.24	20.49	12.53	9.05	195.43	18.82	9.77
Pretr on Imagenet with MOCO	CIFAR	Resnet-50	25.6M	ImageNet (MOCO)	Yes	Yes	0.0003	0.07	112.69	-0.3	18.74	11.21	10.07	210.67	20.61	10.53
DEPTH_FT																
Resnet-18	CIFAR	Resnet-18	11.7M	ImageNet	Yes	Yes	0.001	12.48	194.19	-0.57	18.86	7.28	11.37	150.73	18.91	7.54
Resnet-34	CIFAR	Resnet-34	21.8M	ImageNet	Yes	Yes	0.001	15.76	237.19	-0.73	18.75	4.36	13.51	104.15	18.72	5.21
Resnet-50	CIFAR	Resnet-50	25.6M	ImageNet	Yes	Yes	0.001	15.11	178.61	-0.67	18.33	4.32	13.3	99.88	18.29	4.99
Resnet-101	CIFAR	Resnet-101	44.5M	ImageNet	Yes	Yes	0.0003	10.91	166.44	-0.62	14.97	5.03	8.95	117.22	14.81	5.86
DEPTH_LINEAR																
Resnet-18	CIFAR	Resnet-18	51K	ImageNet	No	Yes	0.001	24.4	65.28	-0.35	32.42	5.61	27.33	102.16	32.44	5.11
Resnet-34	CIFAR	Resnet-34	51K	ImageNet	No	Yes	0.001	21.77	59.56	-0.33	30.02	5.44	25.11	98.33	30.03	4.92
Resnet-50	CIFAR	Resnet-50	205K	ImageNet	No	Yes	0.0003	13.5	56.54	-0.22	28.63	6.66	23.05	112.08	28.65	5.6
Resnet-101	CIFAR	Resnet-101	205K	ImageNet	No	Yes	0.0003	11.17	51.7	-0.21	25.86	6.17	20.98	99.32	25.95	4.97
WIDTH_FT																
Resnet-50	CIFAR	Resnet-50	25.6M	ImageNet	Yes	Yes	0.001	15.11	178.61	-0.67	18.33	4.32	13.3	99.88	18.29	4.99
2xWide-Resnet-50	CIFAR	Wide_Resnet-50_2	68.9M	ImageNet	Yes	Yes	0.0003	8.78	160.14	-0.57	14.04	6	7.83	124.55	14.06	6.23
Resnet-101	CIFAR	Resnet-101	44.5M	ImageNet	Yes	Yes	0.0003	10.91	166.44	-0.62	14.97	5.03	8.95	117.22	14.81	5.86
2xWide-Resnet-101	CIFAR	Wide_Resnet-101_2	126.9M	ImageNet	Yes	Yes	0.0003	7.56	116.21	-0.5	13.37	5.81	7.55	116.26	13.36	5.81
WIDTH_LINEAR																
Resnet-50	CIFAR	Resnet-50	205K	ImageNet	No	Yes	0.0003	13.5	56.54	-0.22	28.63	6.66	23.05	112.08	28.65	5.6
2xWide-Resnet-50	CIFAR	Wide_Resnet-50_2	205K	ImageNet	No	Yes	0.0001	21.78	56.88	-0.35	28.77	4.89	24.45	87.19	28.81	4.36
Resnet-101	CIFAR	Resnet-101	205K	ImageNet	No	Yes	0.0003	11.17	51.7	-0.21	25.86	6.17	20.98	99.32	25.95	4.97
2xWide-Resnet-101	CIFAR	Wide_Resnet-101_2	205K	ImageNet	No	Yes	0.0003	16.95	48.35	-0.25	27.76	5.41	23.27	90.85	27.81	4.54
AUG_NO_PRETR_FT																
Resnet-18 w/ Data-Aug	Places	Resnet-18	11.7M	Random	Yes	Yes	0.001	35.79	118.94	-0.28	58.01	12.44	44.39	263.63	57.57	13.18
Resnet-18 w/o Data-Aug	Places	Resnet-18	11.7M	Random	Yes	No	0.003	36.62	114.2	-0.24	63.73	13.01	48.84	288.46	63.26	14.42
AUG_PRETR_FT																
Resnet-18 w/ Data-Aug	Places	Resnet-18	11.7M	ImageNet	Yes	Yes	0.0003	35.72	67.85	-0.22	53.88	7.99	44.82	174.69	53.55	8.73
Resnet-18 w/o Data-Aug	Places	Resnet-18	11.7M	ImageNet	Yes	No	0.001	39.43	68.44	-0.23	56.68	7.94	47.34	183.99	56.53	9.2
AUG_PRETR_LINEAR																
Resnet-18 w/ Data-Aug	Places	Resnet-18	187K	ImageNet	No	Yes	0.0003	55.55	58.56	-0.43	60.01	3.83	56.11	76.95	59.95	3.85
Resnet-18 w/o Data-Aug	Places	Resnet-18	187K	ImageNet	No	No	0.001	54.2	54.27	-0.36	60.48	4.52	55.62	96.08	60.42	4.8
ARCHITECTURES_IN2CIFAR																
AlexNet	CIFAR	AlexNet	61.1M	ImageNet	Yes	Yes	0.001	7.52	131.77	-0.32	26.89	12.4	15.97	219.36	26.94	10.97
VGG-16(bn)	CIFAR	VGG-16BN	138.4M	ImageNet	Yes	Yes	0.0003	6.21	125.57	-0.38	19.1	9.79	10.07	181.1	19.13	9.06
ResNet-50	CIFAR	Resnet-50	25.6M	ImageNet	Yes	Yes	0.001	15.11	178.61	-0.67	18.33	4.32	13.3	99.88	18.29	4.99
ResNeXt-50(32x4d)	CIFAR	ResNeXt-50(32x4d)	25.0M	ImageNet	Yes	Yes	0.001	12.67	185.51	-0.65	16.45	4.91	11.37	102.91	16.52	5.15
ResNet-101	CIFAR	Resnet-101	44.5M	ImageNet	Yes	Yes	0.0003	10.91	166.44	-0.62	14.97	5.03	8.95	117.22	14.81	5.86
ENSEMBLE																
1xResnet-18	CIFAR	Resnet-18	11.7M	ImageNet	Yes	Yes	0.001	12.48	194.19	-0.57	18.86	7.28	11.37	150.73	18.91	7.54
6xResnet-18	CIFAR	Resnet-18	70.1M	ImageNet	Yes	Yes	0.001	8.73	136.26	-0.49	15.96	7.09	8.55	147.16	15.91	7.36
1xResnet-50	CIFAR	Resnet-50	25.6M	ImageNet	Yes	Yes	0.001	15.11	178.61	-0.67	18.33	4.32	13.3	99.88	18.29	4.99
3xResnet-50	CIFAR	Resnet-50	76.7M	ImageNet	Yes	Yes	0.001	12.72	150.13	-0.64	15.96	4.15	11.43	90.59	15.96	4.53