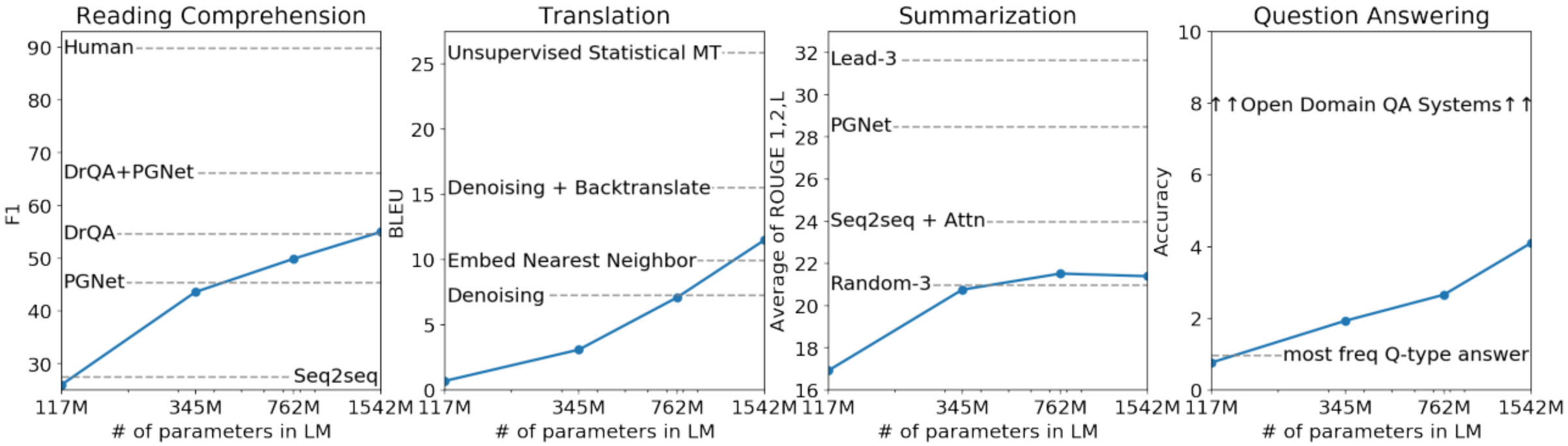
Generalization bounds, scaling, multimodal models

493 / 599 May 18 2023

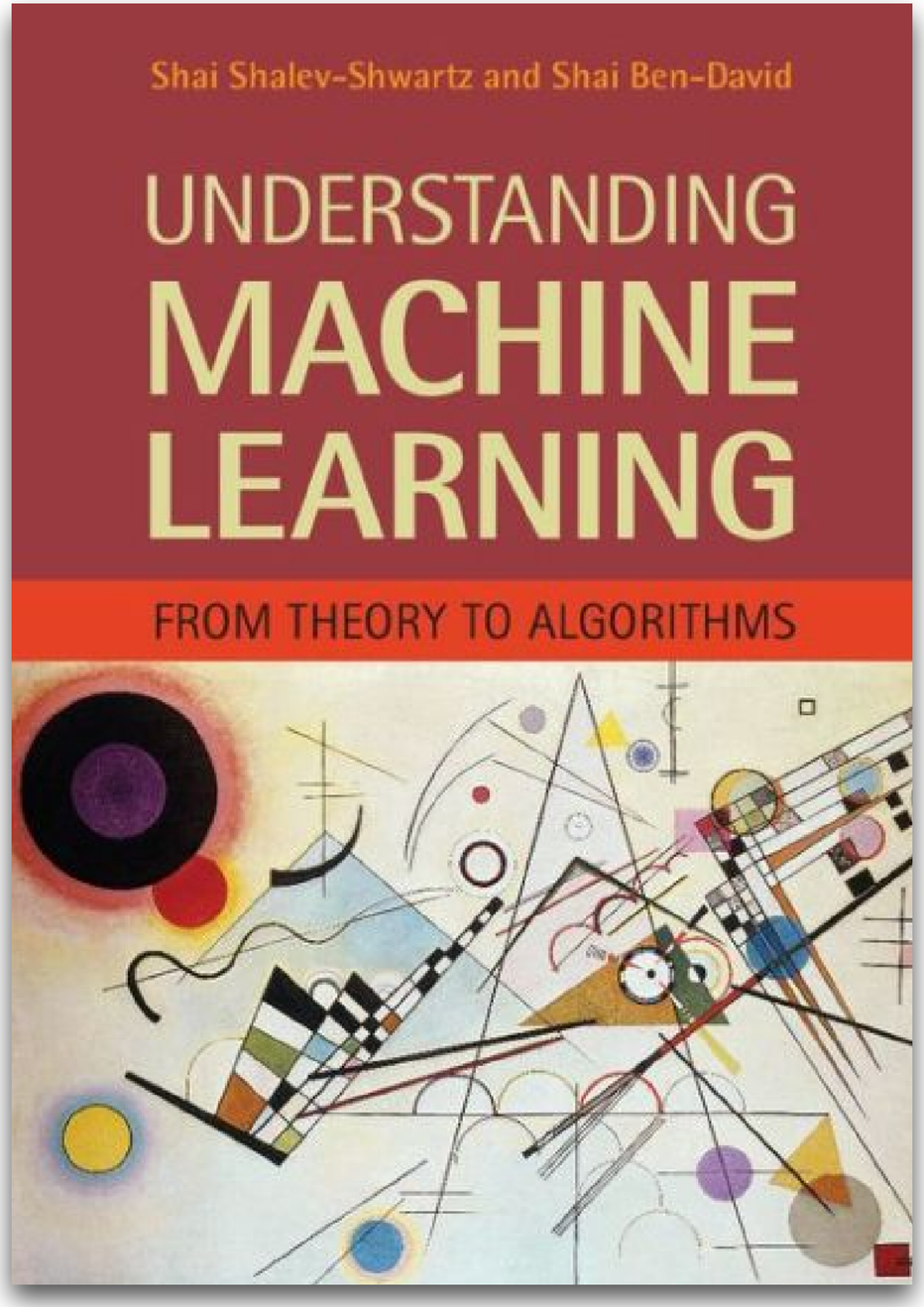
Ludwig Schmidt

Scale is a key ingredient in current models



BUT: large-scale experiments are expensive
+ hyperparameter tuning is difficult (requires many experiments).

Can we predict scaling?


Shai Shalev-Shwartz and Shai Ben-David
UNDERSTANDING
MACHINE
LEARNING
FROM THEORY TO ALGORITHMS

THEOREM 6.8 (The Fundamental Theorem of Statistical Learning – Quantitative Version) *Let $\mathcal{H}$ be a hypothesis class of functions from a domain $\mathcal{X}$ to $\{0,1\}$ and let the loss function be the $0-1$ loss. Assume that $\mathrm{VCdim}(\mathcal{H}) = d < \infty$. Then, there are absolute constants C_1, C_2 such that:*

1. *$\mathcal{H}$ has the uniform convergence property with sample complexity*

$$C_1 \frac{d + \log(1/\delta)}{\epsilon^2} \leq m_{\mathcal{H}}^{UC}(\epsilon, \delta) \leq C_2 \frac{d + \log(1/\delta)}{\epsilon^2}$$

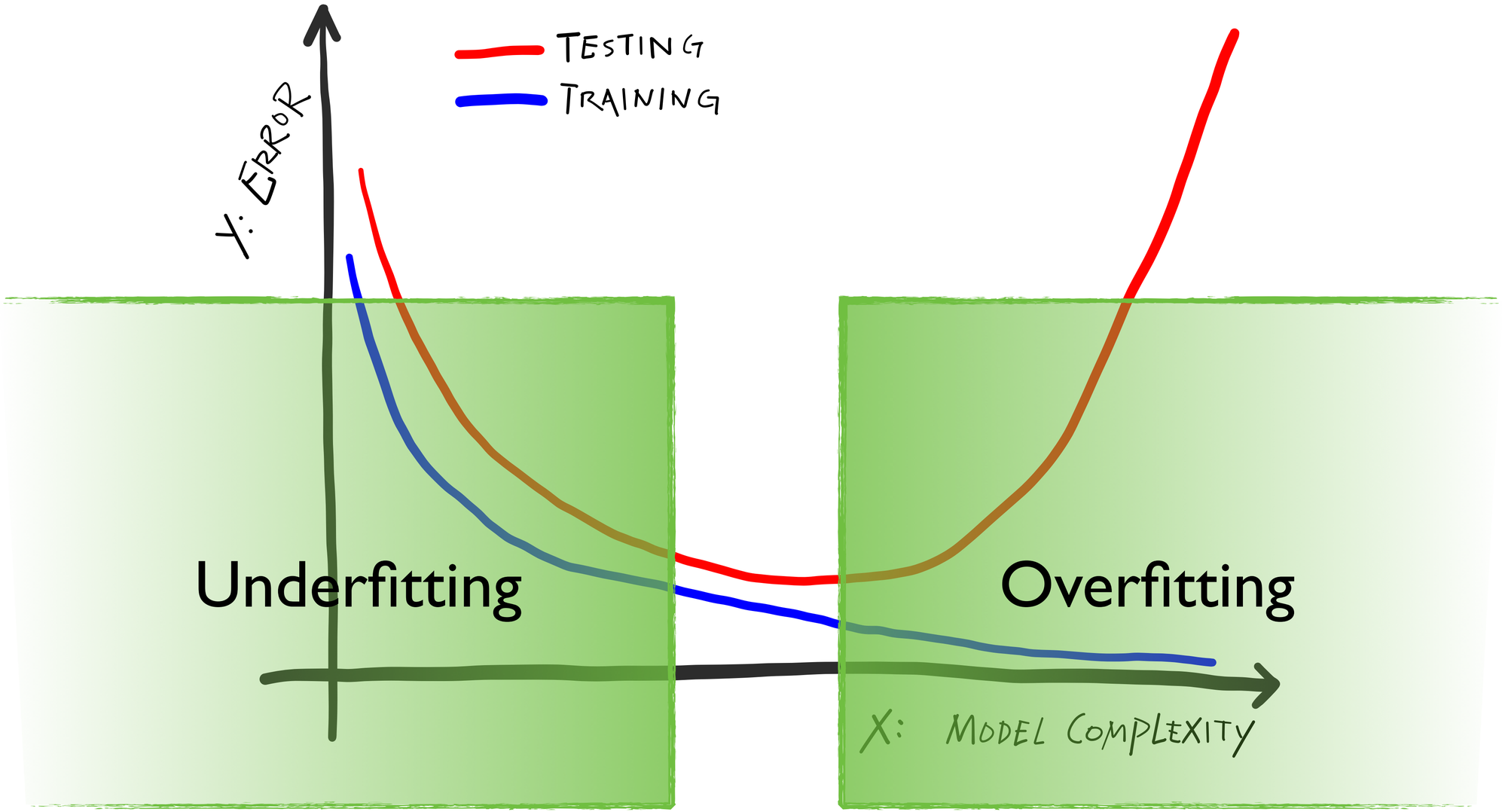
2. *$\mathcal{H}$ is agnostic PAC learnable with sample complexity*

$$C_1 \frac{d + \log(1/\delta)}{\epsilon^2} \leq \ m_{\mathcal{H}}(\epsilon, \delta) \leq C_2 \frac{d + \log(1/\delta)}{\epsilon^2}$$

3. *$\mathcal{H}$ is PAC learnable with sample complexity*

$$C_1 \frac{d + \log(1/\delta)}{\epsilon} \leq m_{\mathcal{H}}(\epsilon, \delta) \leq C_2 \frac{d \log(1/\epsilon) + \log(1/\delta)}{\epsilon}$$

Overfitting and the Bias-Variance Trade-off



Lessons in Neural Network Training: Overfitting May be Harder than Expected

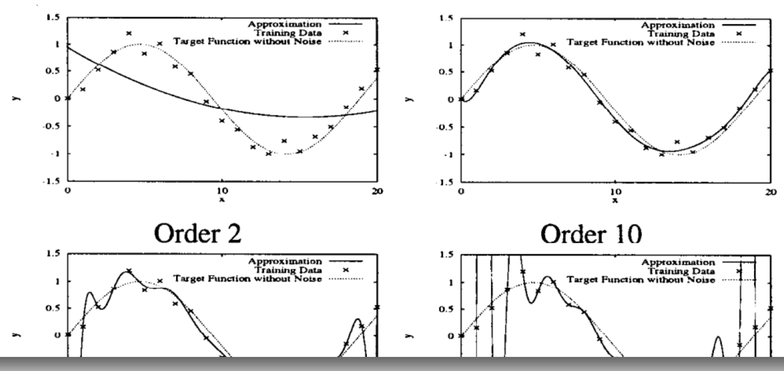
Steve Lawrence[1], **C. Lee Giles**[1], **Ah Chung Tsoi**[2]

[1] NEC Research, 4 Independence Way, Princeton, NJ 08540, [2] Faculty of Informatics, Uni. of Wollongong, Australia
{lawrence,giles}@research.nj.nec.com, Ah_Chung_Tsoi@uow.edu.au

Abstract

For many reasons, neural networks have become very popular AI machine learning models. Two of the most important aspects of machine learning models are how well the model generalizes to unseen data, and how well the model scales with problem complexity. Using a controlled task with known optimal training error, we investigate the convergence of the backpropagation (BP) algorithm. We find that the optimal solution is typically not found. Furthermore, we observe that networks larger than might be expected can result in lower training and generalization error. This result is supported by another real world example. We further investigate the training behavior by analyzing the weights in trained networks (excess degrees of freedom are seen to do little harm and to aid convergence), and contrasting the interpolation characteristics of multi-layer perceptron neural networks (MLPs) and polynomial models (overfitting behavior is very different – the MLP is often biased towards smoother solutions). Finally, we analyze relevant theory outlining the reasons for significant practical differences. These results bring into question common beliefs about neural network training regarding convergence and optimal network size, suggest alternate guidelines for practical use (lower fear of excess degrees of freedom), and help to direct future work (e.g. methods for creation of more parsimonious solutions, importance of the MLP/BP bias and possibly worse performance of "improved" training algorithms).

$y = \sin(x/3) + \nu$ where ν is a uniformly distributed random variable between -0.25 and 0.25. The equation was evaluated at $0, 1, 2, \ldots, 20$. This dataset was then used to fit polynomial models with orders between 2 and 20. For order 2, the approximation is poor. For order 10, the approximation is reasonably good. However, as the order (and number of parameters) increases, significant overfitting and increasingly poor generalization is evident. At order 20, the approximated function fits the training data very well, however the interpolation between training points is very poor. Overfitting can also be a very important problem in MLPs, and much work has been devoted to preventing overfitting with techniques such as model selection, early stopping, weight decay, and pruning.



The stipulation that the number of parameters must be less than the number of examples is typically believed to be true for common datasets. The results here indicate that this is not always the case.

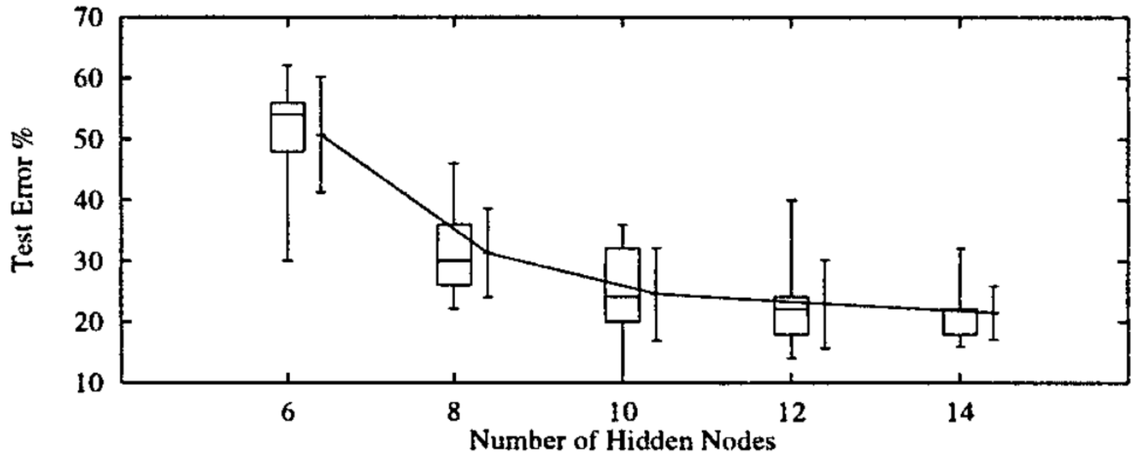
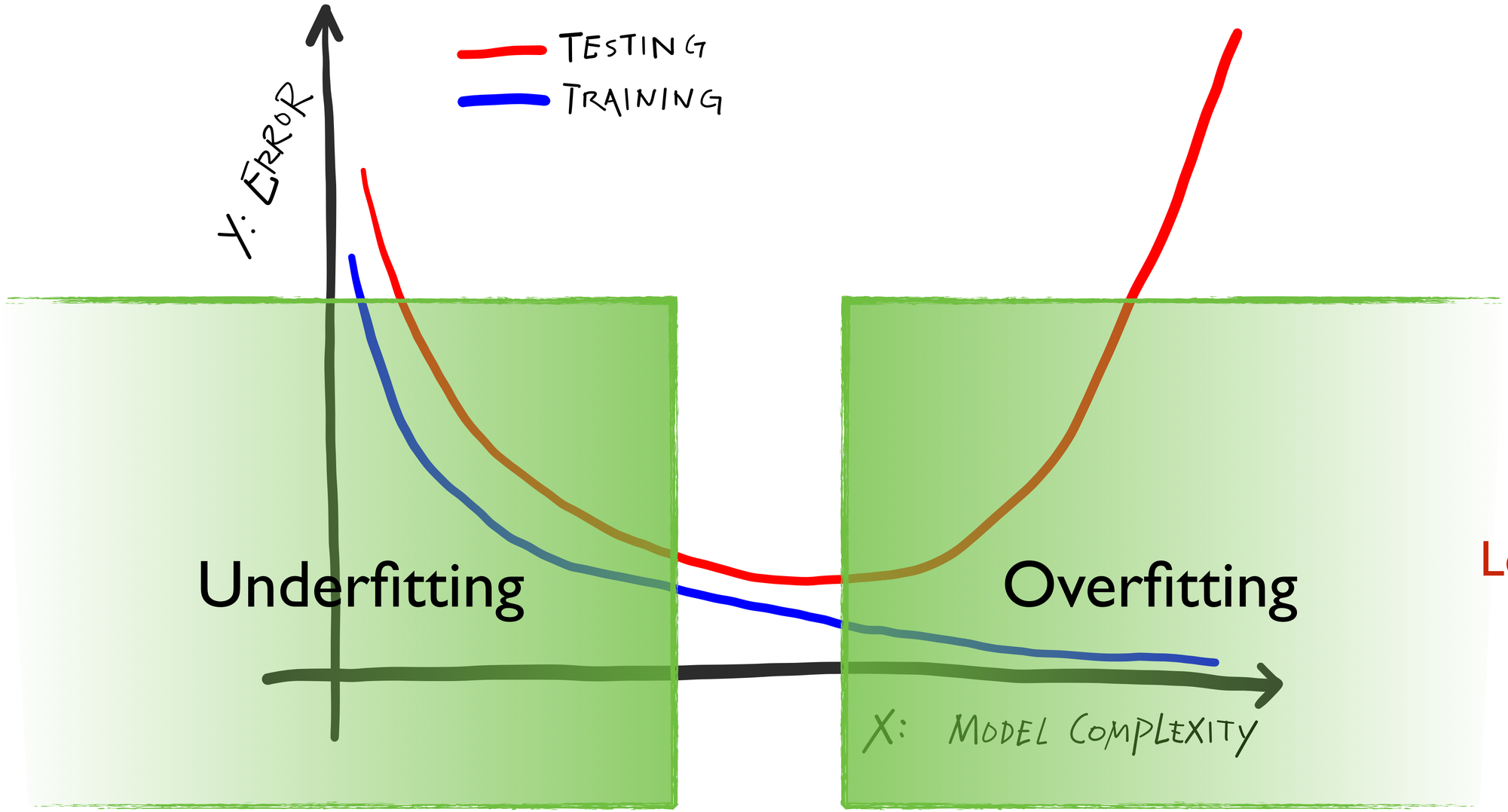


Figure 3. Face recognition example: the best generalizing network has 364 times more parameters than training points (18210 parameters).

Overfitting and the Bias-Variance Trade-off



Deep Learning

The Sample Complexity of Pattern Classification with Neural Networks: The Size of the Weights is More Important than the Size of the Network

Peter L. Bartlett, *Member, IEEE*

***Abstract*—Sample complexity results from computational learning theory, when applied to neural network learning for pattern classification problems, suggest that for good generalization performance the number of training examples should grow at least linearly with the number of adjustable parameters in the network. Results in this paper show that if a large neural network is used for a pattern classification problem and the learning algorithm finds a network with small weights that has small squared error on the training patterns, then the generalization performance depends on the size of the weights rather than the number of weights. For example, consider a two-layer feedforward network of sigmoid units, in which the sum of the magnitudes of the weights associated with each unit is bounded by A and the input dimension is n. We show that the misclassification probability is no more than a certain error estimate (that is related to squared error on the training set) plus $A^3\sqrt{(\log n)/m}$ (ignoring $\log A$ and $\log m$ factors), where m is the number of training patterns. This may explain the generalization performance of neural networks, particularly when the number of training examples is considerably smaller than the number of weights. It also supports heuristics (such as weight decay and early stopping) that attempt to keep the weights small during training. The proof techniques appear to be useful for the analysis of other pattern classifiers: when the input domain is a totally bounded metric space, we use the same approach to give upper bounds on misclassification probability for classifiers with decision boundaries that are far from the training examples.**

VC dimension is a combinatorial complexity measure that is typically at least as large as the number of adjustable network parameters.) These results do not provide a satisfactory explanation of the sample size requirements of neural networks for pattern classification applications, for several reasons. First, neural networks often perform successfully with training sets that are considerably smaller than the number of network parameters (see, for example, [29]). Second, the VC dimension of the class of functions computed by a network is sensitive to small perturbations of the computation unit transfer functions (to the extent that an arbitrarily small change can make the VC dimension infinite, see [39]). That this could affect the generalization performance seems unnatural, and has not been observed in practice.

In fact, the sample size bounds in terms of VC dimension are tight in the sense that, for every learning algorithm that selects hypotheses from some class, there is a probability distribution and a target function for which, if training data is chosen independently from the distribution and labeled according to the target function, the function chosen by the learning algorithm will misclassify a random example with probability at least proportional to the VC dimension of the class divided by the number of training examples. However,

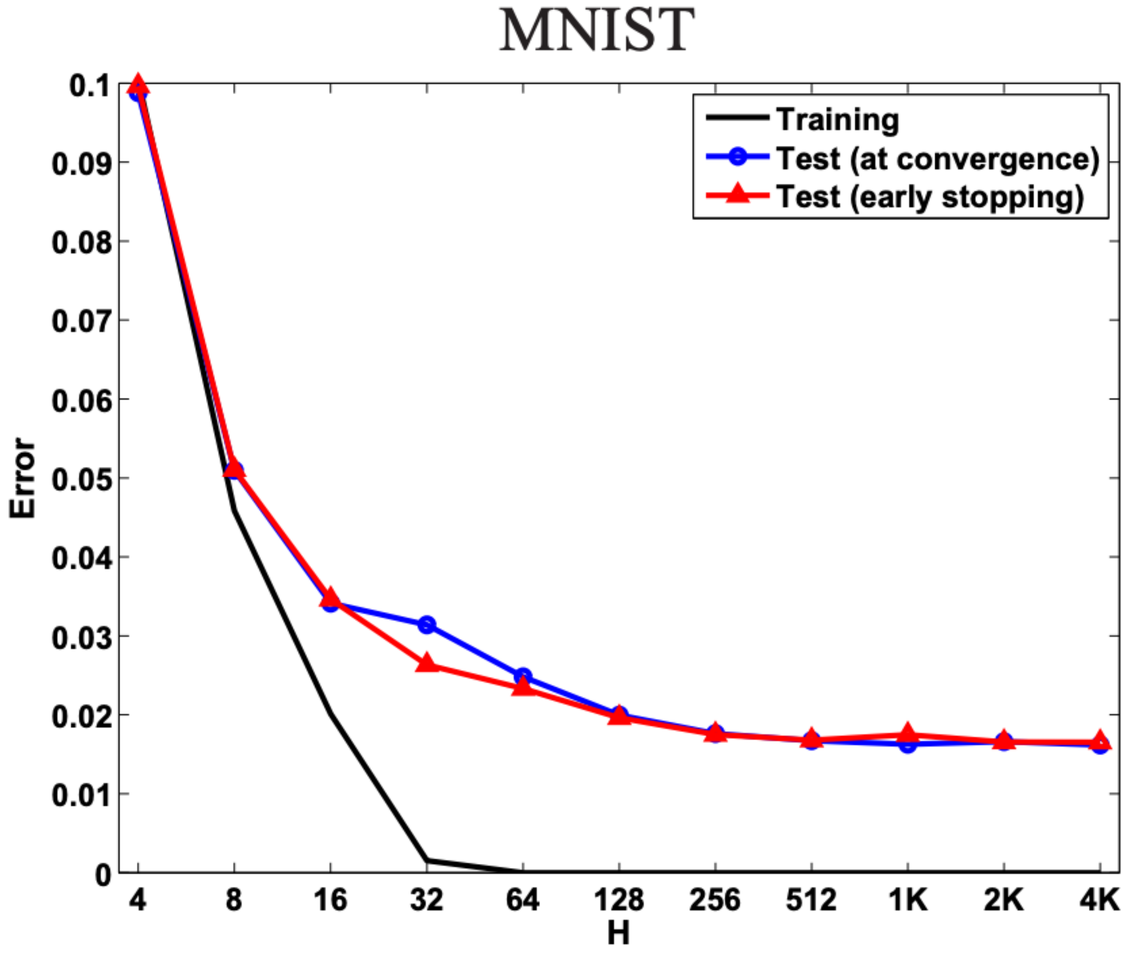
Accepted as a workshop contribution at ICLR 2015

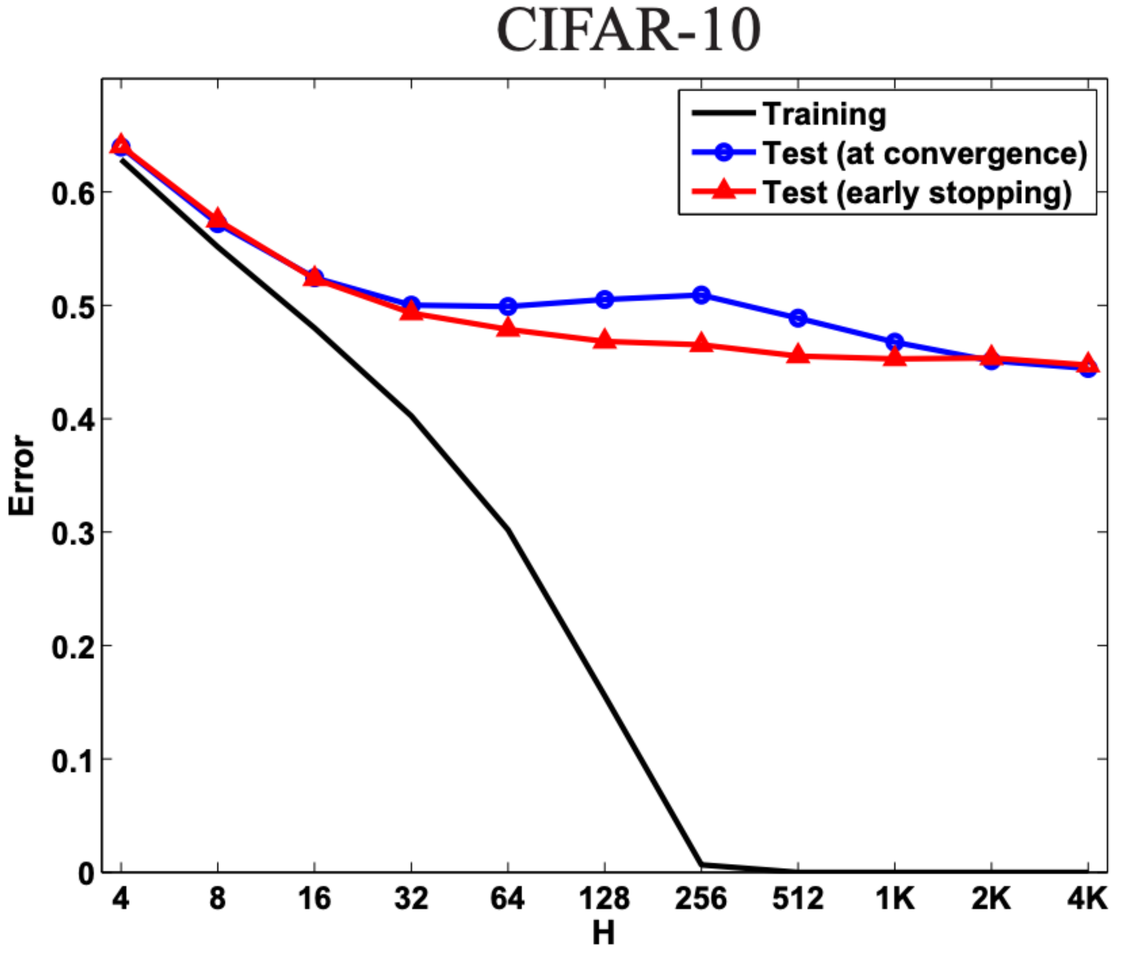
IN SEARCH OF THE REAL INDUCTIVE BIAS: ON THE ROLE OF IMPLICIT REGULARIZATION IN DEEP LEARNING

Behnam Neyshabur, Ryota Tomioka & Nathan Srebro
Toyota Technological Institute at Chicago
Chicago, IL 60637, USA
{bneyshabur,tomioka,nati}@ttic.edu

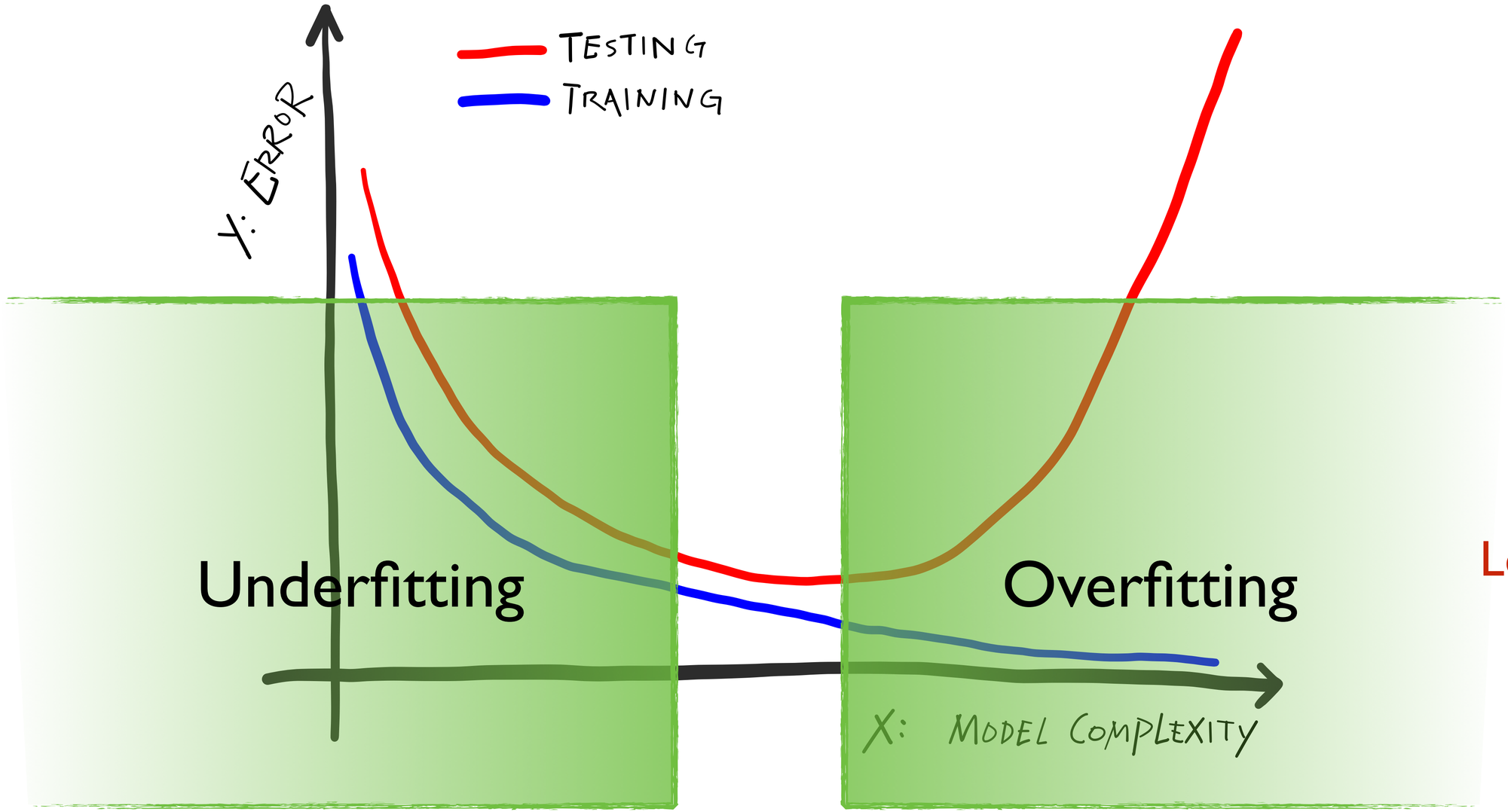
ABSTRACT

We present experiments demonstrating that some other form of capacity control, different from network size, plays a central role in learning multi-layer feed-forward networks. We argue, partially through analogy to matrix factorization, that this is an inductive bias that can help shed light on deep learning.


MNIST
Training
Test (at convergence)
Test (early stopping)
Error
H
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
4
8
16
32
64
128
256
512
1K
2K
4K


CIFAR-10
Training
Test (at convergence)
Test (early stopping)
Error
H
0.6
0.5
0.4
0.3
0.2
0.1
0
4
8
16
32
64
128
256
512
1K
2K
4K

Overfitting and the Bias-Variance Trade-off



Deep Learning

UNDERSTANDING DEEP LEARNING REQUIRES RETHINKING GENERALIZATION

Chiyuan Zhang*
Massachusetts Institute of Technology
chiyuan@mit.edu

Samy Bengio
Google Brain
bengio@google.com

Moritz Hardt
Google Brain
mrtz@google.com

Benjamin Recht†
University of California, Berkeley
brecht@berkeley.edu

Oriol Vinyals
Google DeepMind
vinyals@google.com

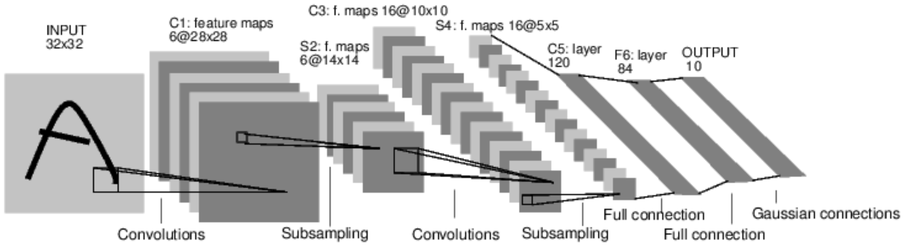
ABSTRACT

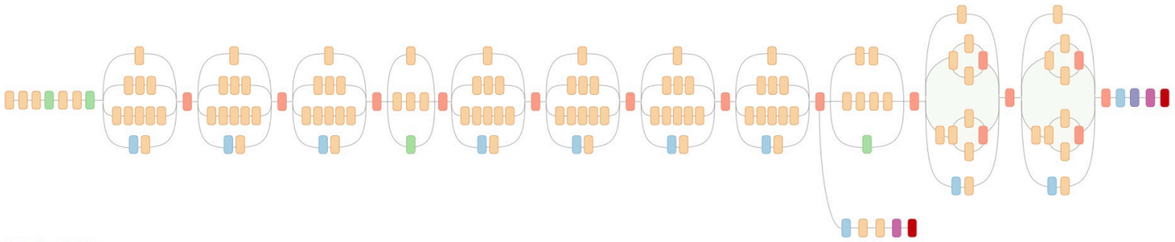
Despite their massive size, successful deep artificial neural networks can exhibit a remarkably small difference between training and test performance. Conventional wisdom attributes small generalization error either to properties of the model family, or to the regularization techniques used during training.

Through extensive systematic experiments, we show how these traditional approaches fail to explain why large neural networks generalize well in practice. Specifically, our experiments establish that state-of-the-art convolutional networks for image classification trained with stochastic gradient methods easily fit a random labeling of the training data. This phenomenon is qualitatively unaffected by explicit regularization, and occurs even if we replace the true images by completely unstructured random noise. We corroborate these experimental findings with a theoretical construction showing that simple depth two neural networks already have perfect finite sample expressivity as soon as the number of parameters exceeds the number of data points as it usually does in practice.

We interpret our experimental findings by comparison with traditional models.

Increasingly large image classification architectures





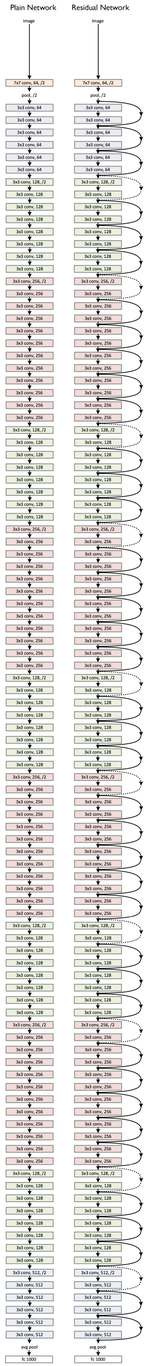


Image classification architectures

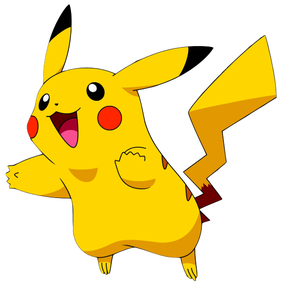
CIFAR-10	# train: 50,000
Inception	1.6M
Alexnet	1.4M
MLP 1x512	1.2M
ImageNet	**# train: ~1,200,000**
Inception V4	43M
Alexnet	61M
Resnet-{18;152}	12M; 60M
VGG-{11;19}	133M; 144M

Parameter Count

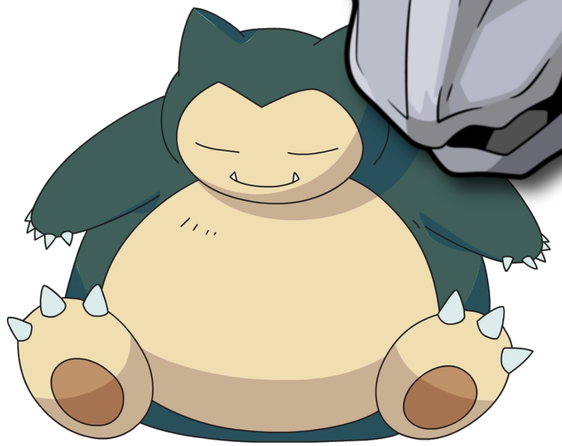
Num Training Samples

MLP 1x512
p/n: 24

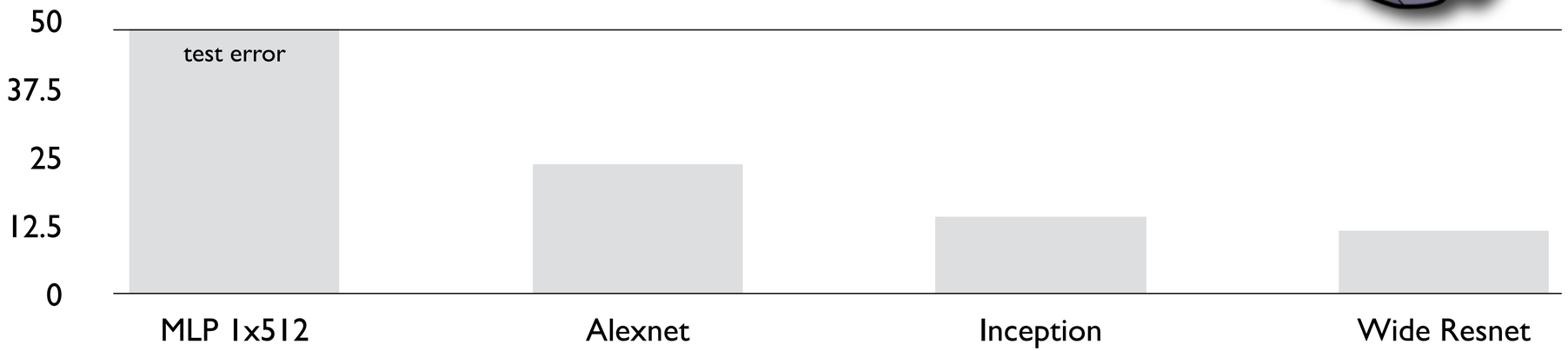
Alexnet
p/n: 28



Inception
p/n: 33



Wide Resnet
p/n: 179



Randomization Test

Deep Neural Networks easily fit random labels.

the Rademacher complexity of $\mathcal{F}$ with respect to S is defined as follows:

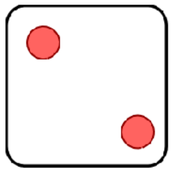
$$R(\mathcal{F} \circ S) \stackrel{\text{def}}{=} \frac{1}{m} \mathop{\mathbb{E}}_{\boldsymbol{\sigma} \sim \{\pm 1\}^m} \left[\sup_{f \in \mathcal{F}} \sum_{i=1}^{m} \sigma_i f(z_i) \right]. \tag{26.4}$$

3. For any $h^\star$, with probability of at least $1 - \delta$,

$$L_{\mathcal{D}}(\mathrm{ERM}_{\mathcal{H}}(S)) - L_{\mathcal{D}}(h^\star) \;\leq\; 2\, R(\ell \circ \mathcal{H} \circ S) + 5\, c \sqrt{\frac{2 \ln(8/\delta)}{m}}.$$

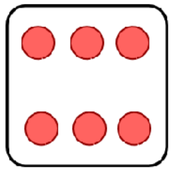
Random Label Dataset

Dog

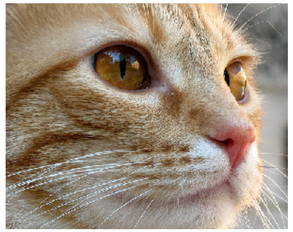


Cat

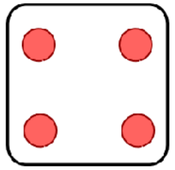
Flower



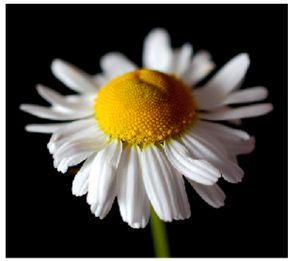
Dog



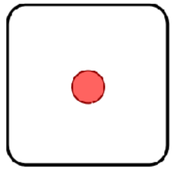
Cat



Bus



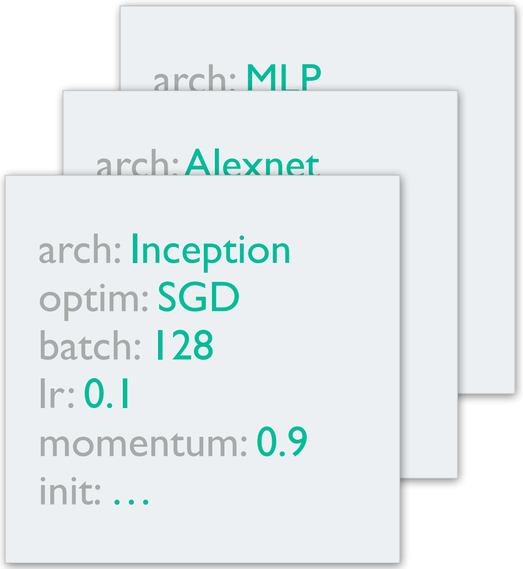
Flower



Bird

⋮

Randomization Test



Recipes of Successful Models

Data with original labels


dog | flower

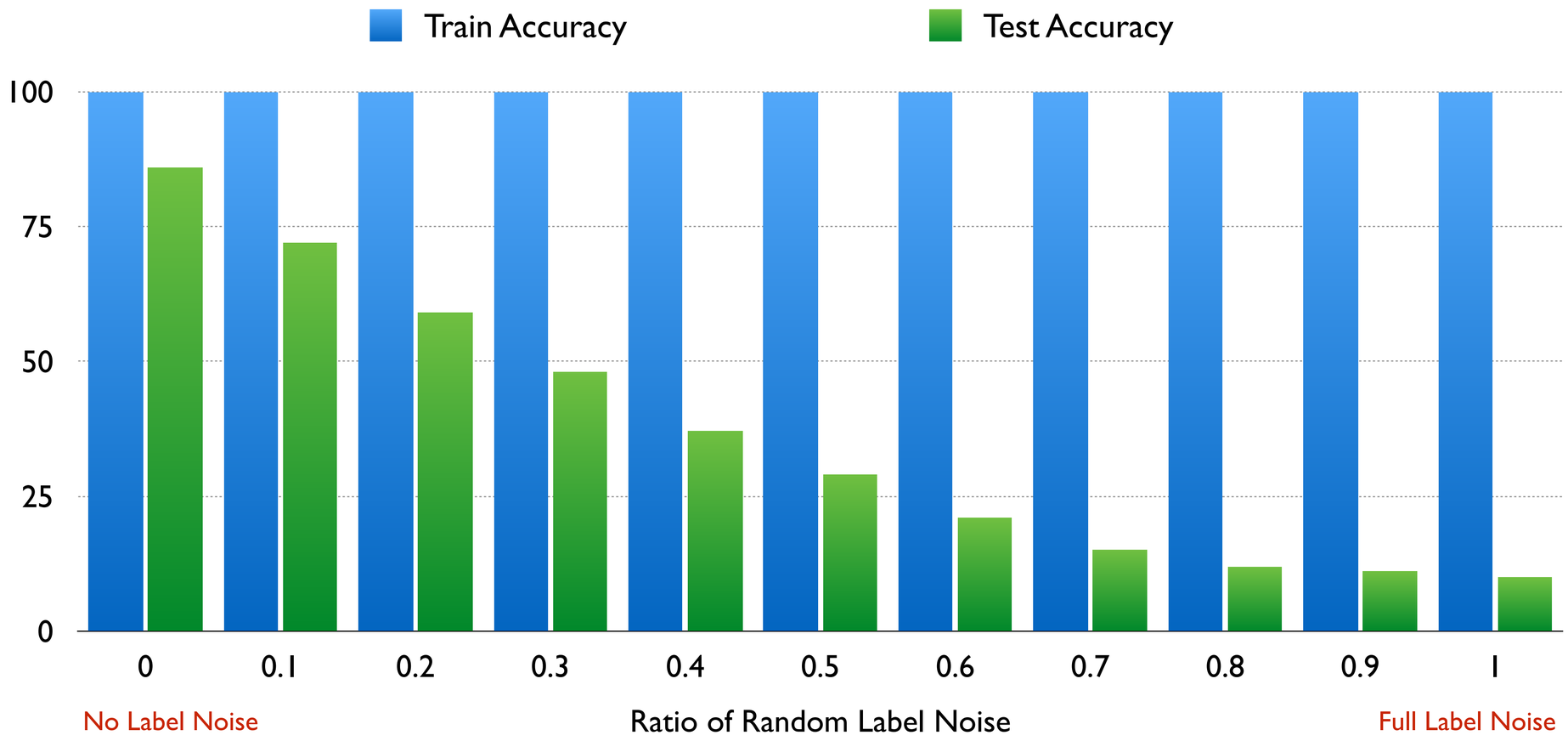
Data with **random** labels


bus | dog

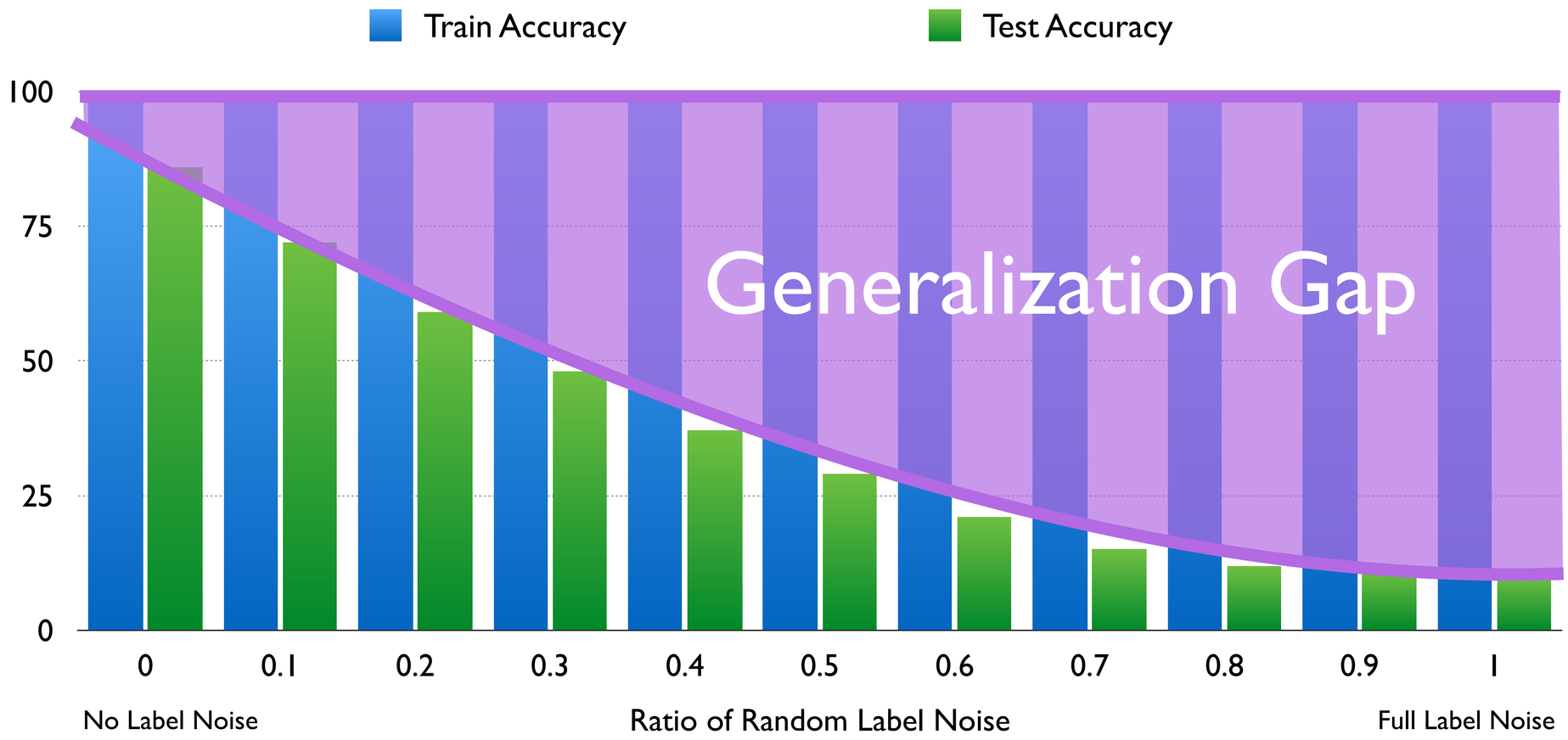
https://www.tensorflow.org/
http://mxnet.io/

TensorFlow™ MXNet

Randomization Test

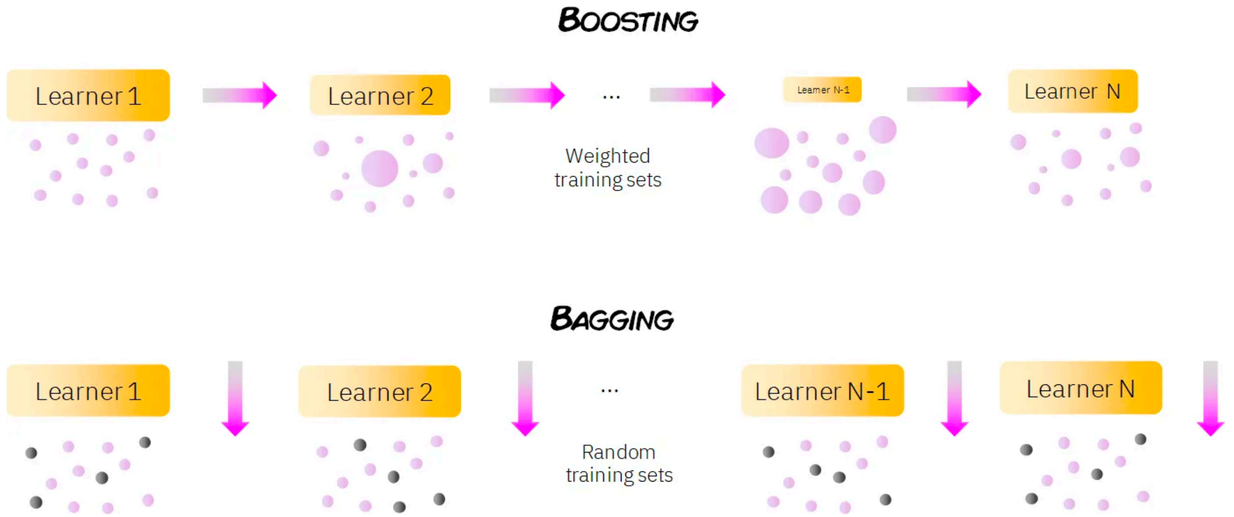


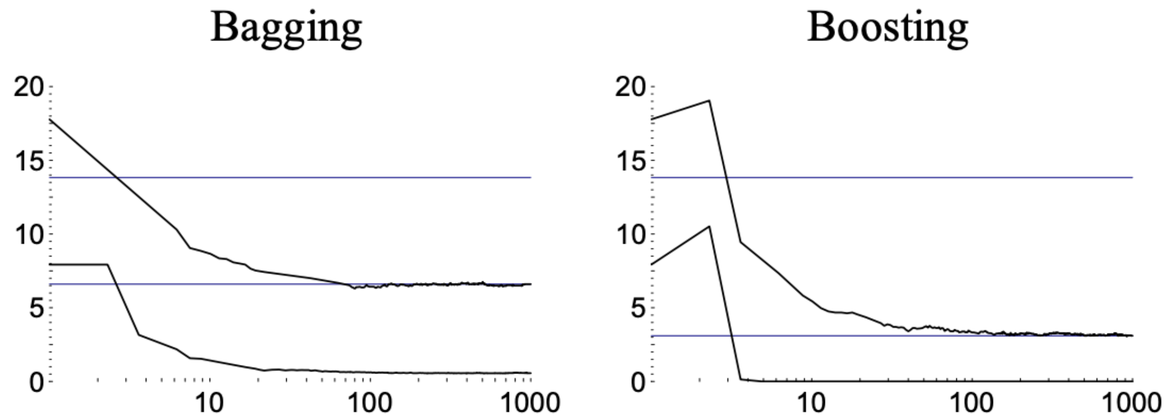
Randomization Test



Similar phenomena for boosting and bagging

Idea: combine many low-accuracy models into one high-accuracy model.





Machine Learning: Proceedings of the Fourteenth International Conference, 1997.

Boosting the margin: A new explanation for the effectiveness of voting methods

Robert E. Schapire **Yoav Freund**
AT&T Labs*
600 Mountain Avenue
Murray Hill, NJ 07974 USA
{schapire, yoav}@research.att.com

Peter Bartlett
Dept. of Systems Engineering
RSISE, Aust. National University
Canberra, ACT 0200 Australia
Peter.Bartlett@anu.edu.au

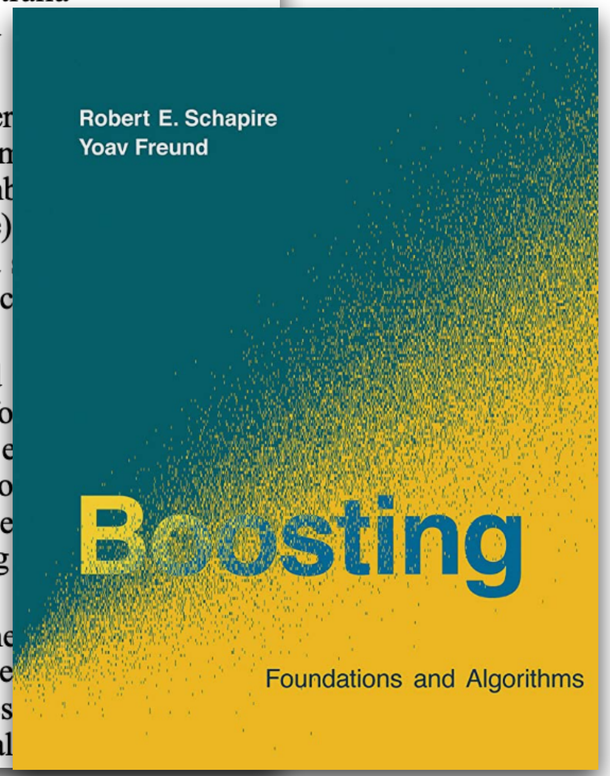
Wee Sun Lee
Electrical Engineering Department
University College UNSW
Australian Defence Force Academy
Canberra ACT 2600 Australia
w-lee@ee.adfa.oz.au

Abstract. One of the surprising recurring phenomena observed in experiments with boosting is that the test error of the generated hypothesis usually does not increase as its size becomes very large, and often is observed to decrease even after the training error reaches zero. In this paper, we show that this phenomenon is related to the distribution of *margins* of the training examples with respect to the generated voting classification rule, where the margin of an example is simply the difference between the number of correct votes and the maximum number of votes received by any incorrect label. We show that techniques used in the analysis of Vapnik's support vector classifiers and of neural networks with small weights can be applied to voting methods to relate the margin distribution to the test error. We also show theoretically and experimentally that boosting is especially effective at increasing the margins of the training examples. Finally, we compare our explanation to those based on the bias-variance decomposition.

Figure 1, we have shown the training and test er
(lower and upper curves, respectively) of the con
pothesis as a function of the number of trees comb
test error of C4.5 on this dataset (run just once)
The test error of bagging 1000 trees is 6.6%, a
improvement. (Both of these error rates are indic
figure as horizontal grid lines.)

In the second experiment, we used Freund
pire's AdaBoost algorithm [12] on top of C4.5 fo
dataset. This algorithm is similar to bagging, e
the subsamples are chosen in a manner which co
on the "hardest" examples. The results of this e
are also shown in Figure 1. Note that boosting
test error down even further to just 3.1%.

These error curves reveal a remarkable phe
first observed by Drucker and Cortes [8], and late
lan [15] and Breiman [5]. Ordinarily, as hypothes
more and more complex, we expect their general



Generalization in deep learning

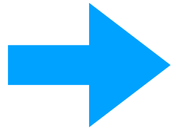
A lot of theoretical work in the past 8 years aims to explain empirical success of DL.

Some approaches:

- Implicit regularization by the learning algorithm
- “Benign overfitting” due to properties of the data generation method
- Simplified theoretical models (e.g., two layer networks)
- Neural tangent kernel (infinite width networks)
- Etc.

→ Many interesting theoretical / mathematical ideas.

But: current mathematical theory is not precise enough to predict empirical phenomena.



What about an experiment-based approach to predict scaling?

DEEP LEARNING SCALING IS PREDICTABLE, EMPIRICALLY

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun,

Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, Yanqi Zhou

{joel,sharan,ardalaninewsha,gregdiamos,junheewoo,hassankianinejad,
patwarymostofa,yangyang62,zhouyanqi}@baidu.com

Baidu Research

ABSTRACT

Deep learning (DL) creates impactful advances following a virtuous recipe: model architecture search, creating large training data sets, and scaling computation. It is widely believed that growing training sets and models should improve accuracy and result in better products. As DL application domains grow, we would like a deeper understanding of the relationships between training set size, computational scale, and model accuracy improvements to advance the state-of-the-art.

This paper presents a large scale empirical characterization of generalization error and model size growth as training sets grow. We introduce a methodology for this measurement and test four machine learning domains: machine translation, language modeling, image processing, and speech recognition. Our empirical results show power-law generalization error scaling across a breadth of factors, resulting in power-law exponents—the "steepness" of the learning curve—yet to be explained by theoretical work. Further, model improvements only shift the error but do *not* appear to affect the power-law exponent. We also show that model size scales sublinearly with data size. These scaling relationships have significant implications on deep learning research, practice, and systems. They can assist model debugging, setting accuracy targets, and decisions about data set growth. They can also guide computing system design and underscore the importance of continued computational scaling.

Empirical scaling behavior

Key question: Can we predict how well a neural network performs at larger scale based on experimentally measuring performance at smaller scale?

If so, we can design architectures and choose hyperparameters with small-scale experiments that are cheaper.

Need an **assumption** to relate performance across scale: **power law**.

$$\varepsilon(m) = \alpha m^{\beta_g}$$

ε: generalization error

m: number of samples in training set

α: problem-specific proportionality constant

β_g: scaling exponent

THEOREM 6.8 (The Fundamental Theorem of Statistical Learning – Quantitative Version) *Let $\mathcal{H}$ be a hypothesis class of functions from a domain $\mathcal{X}$ to $\{0,1\}$ and let the loss function be the $0-1$ loss. Assume that $\mathrm{VCdim}(\mathcal{H}) = d < \infty$. Then, there are absolute constants C_1, C_2 such that:*

1. *$\mathcal{H}$ has the uniform convergence property with sample complexity*

$$C_1 \frac{d + \log(1/\delta)}{\epsilon^2} \leq m_{\mathcal{H}}^{UC}(\epsilon, \delta) \leq C_2 \frac{d + \log(1/\delta)}{\epsilon^2}$$

2. *$\mathcal{H}$ is agnostic PAC learnable with sample complexity*

$$C_1 \frac{d + \log(1/\delta)}{\epsilon^2} \leq \ m_{\mathcal{H}}(\epsilon, \delta) \leq C_2 \frac{d + \log(1/\delta)}{\epsilon^2}$$

3. *$\mathcal{H}$ is PAC learnable with sample complexity*

$$C_1 \frac{d + \log(1/\delta)}{\epsilon} \leq m_{\mathcal{H}}(\epsilon, \delta) \leq C_2 \frac{d \log(1/\epsilon) + \log(1/\delta)}{\epsilon}$$

A note on noise conditions

Not all theoretical bounds have scaling exponents of 1 or 1/2 !

Definition (Tsybakov's noise condition or Mammen-Tsybakov noise condition): The noise in binary classification is said to satisfy Tsybakov's condition if there exists $\alpha \in (0,1)$, $C_0 > 0$ and $t_0 \in (0, \frac{1}{2}]$ such that

$$\mathbb{P}[|\eta(X) - \frac{1}{2}| \leq t] \leq C_0 t^{\frac{\alpha}{1-\alpha}}$$

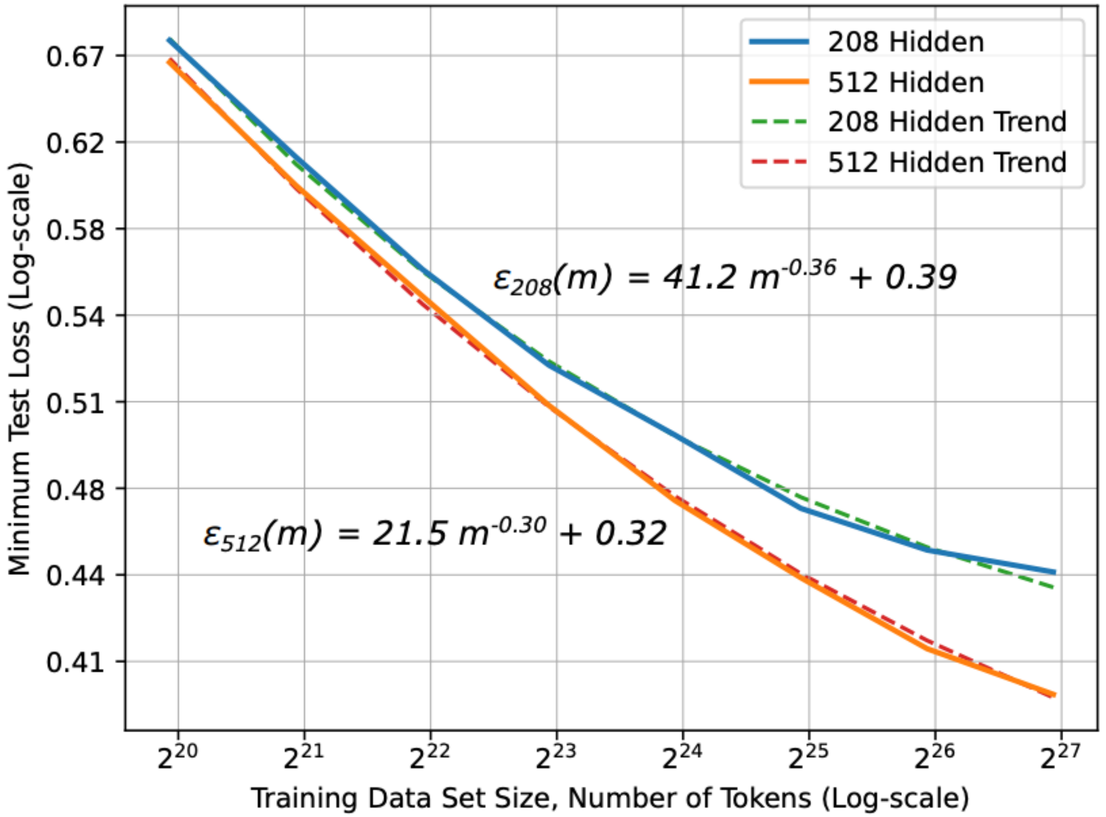
for all $t \in [0, t_0]$.

Theorem: If Tsybakov's noise condition is satisfied with constant α, C_0 and t_0, then there exists a constant $C = C(\alpha, C_0, t_0)$ such that

$$\mathcal{E}(\hat{h}) \leq C\left(\frac{\log(M/\delta)}{n}\right)^{\frac{1}{2-\alpha}}$$

with probability at least $1 - \delta$.

Machine translation



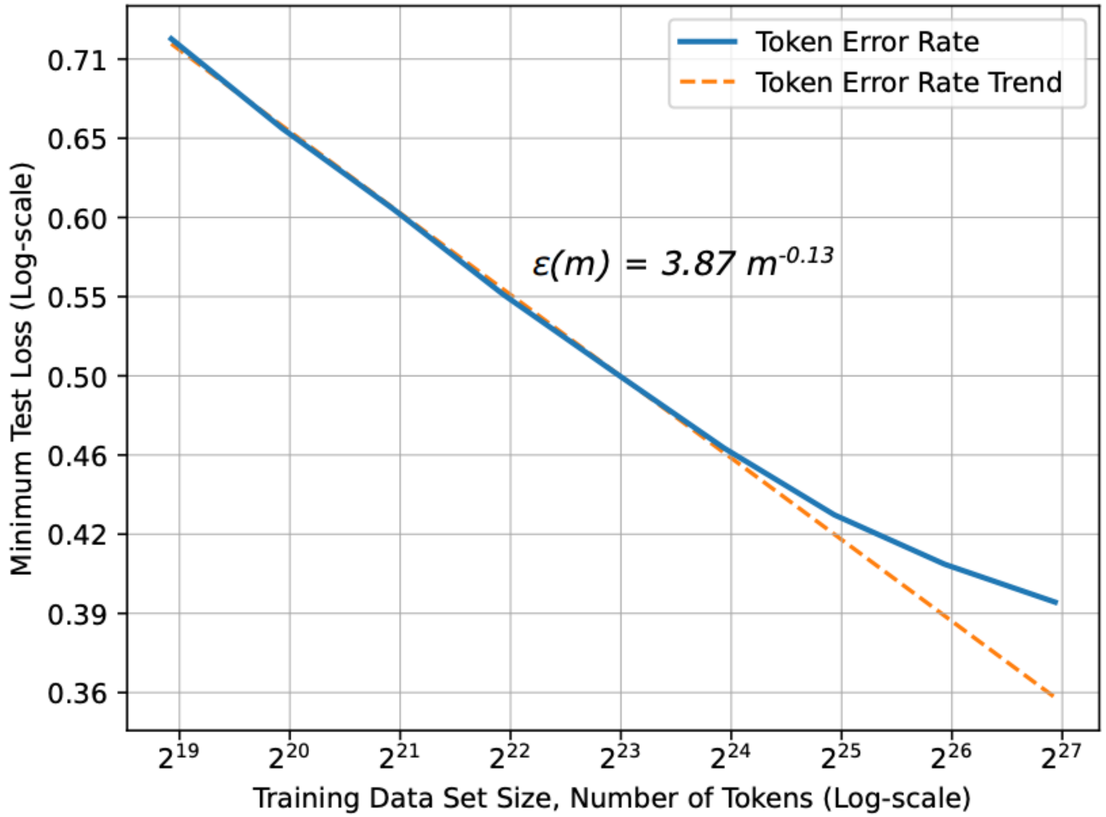
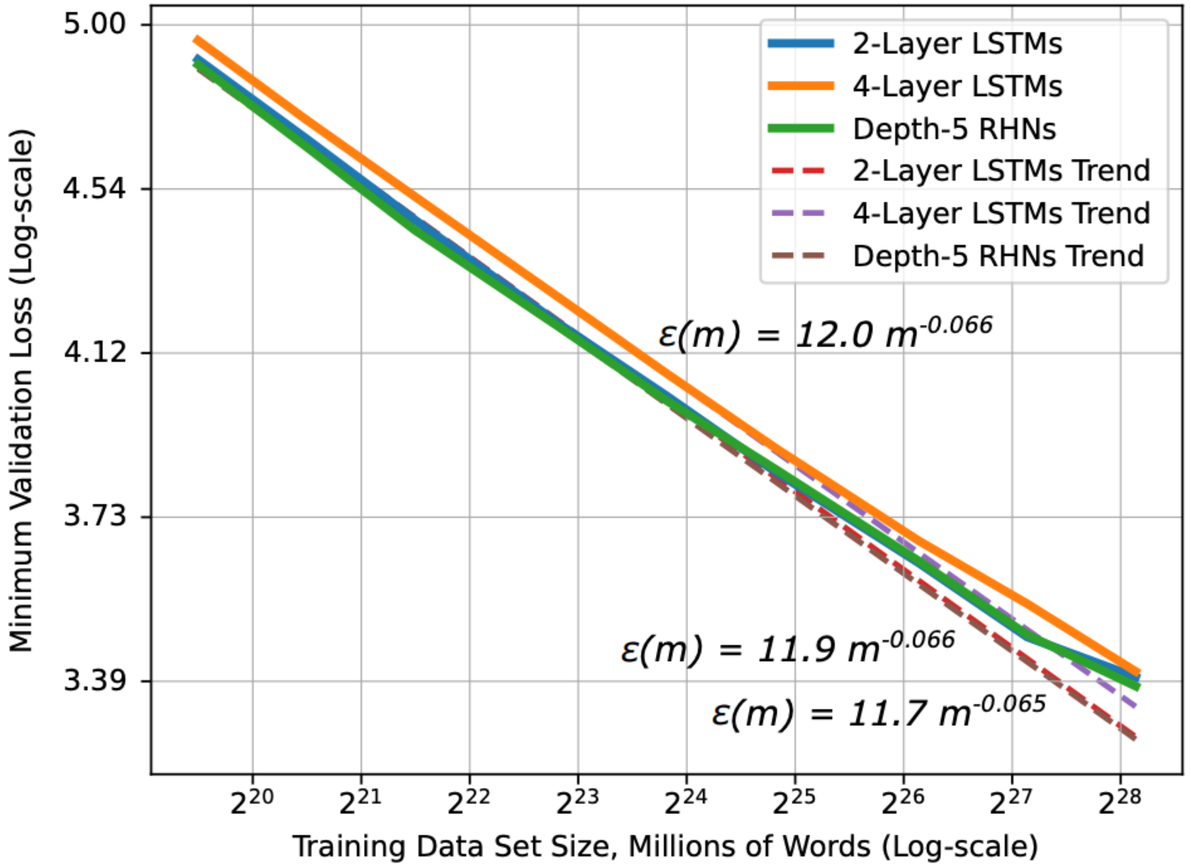


Figure 1: Neural machine translation learning curves. Left: the learning curves for separate models follow $\varepsilon(m) = \alpha m^{\beta_g} + \gamma$. Right: composite learning curve of best-fit model at each data set size.

Word language modeling



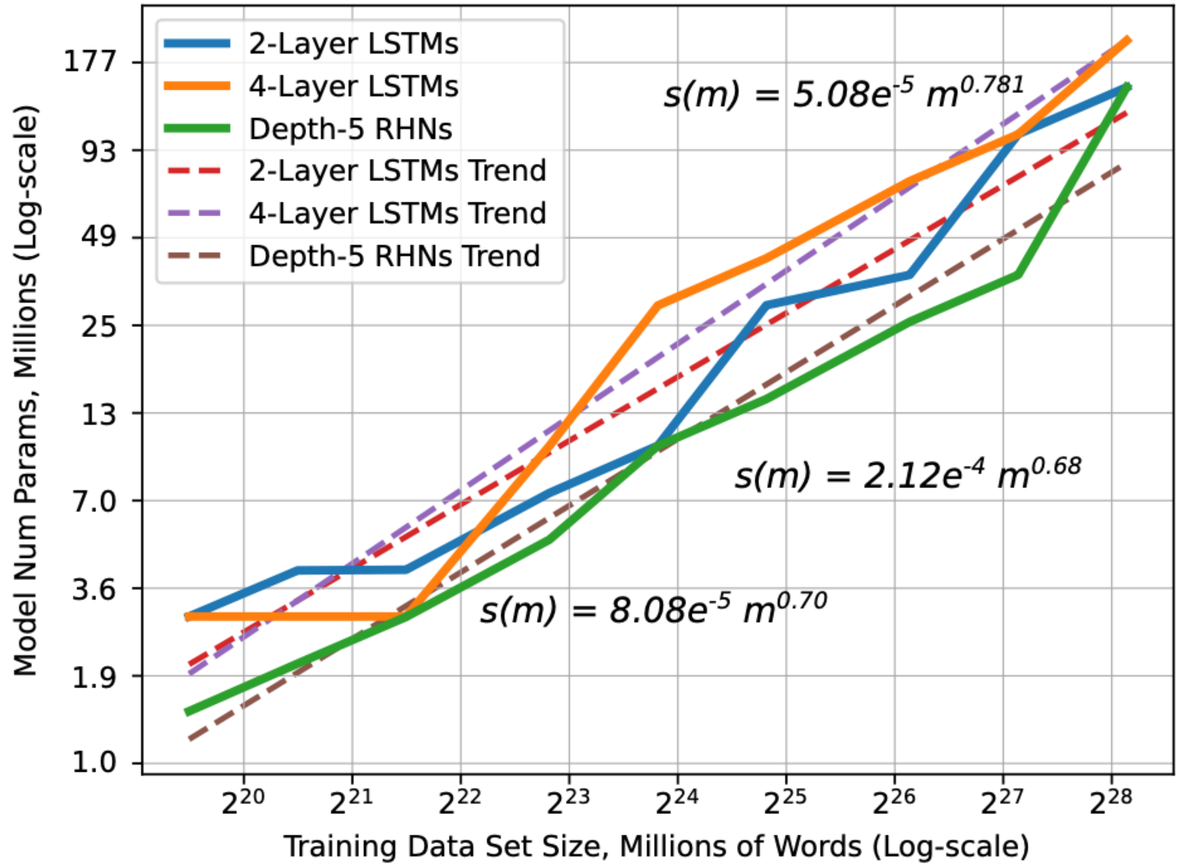


Figure 2: Learning curve and model size results and trends for word language models.

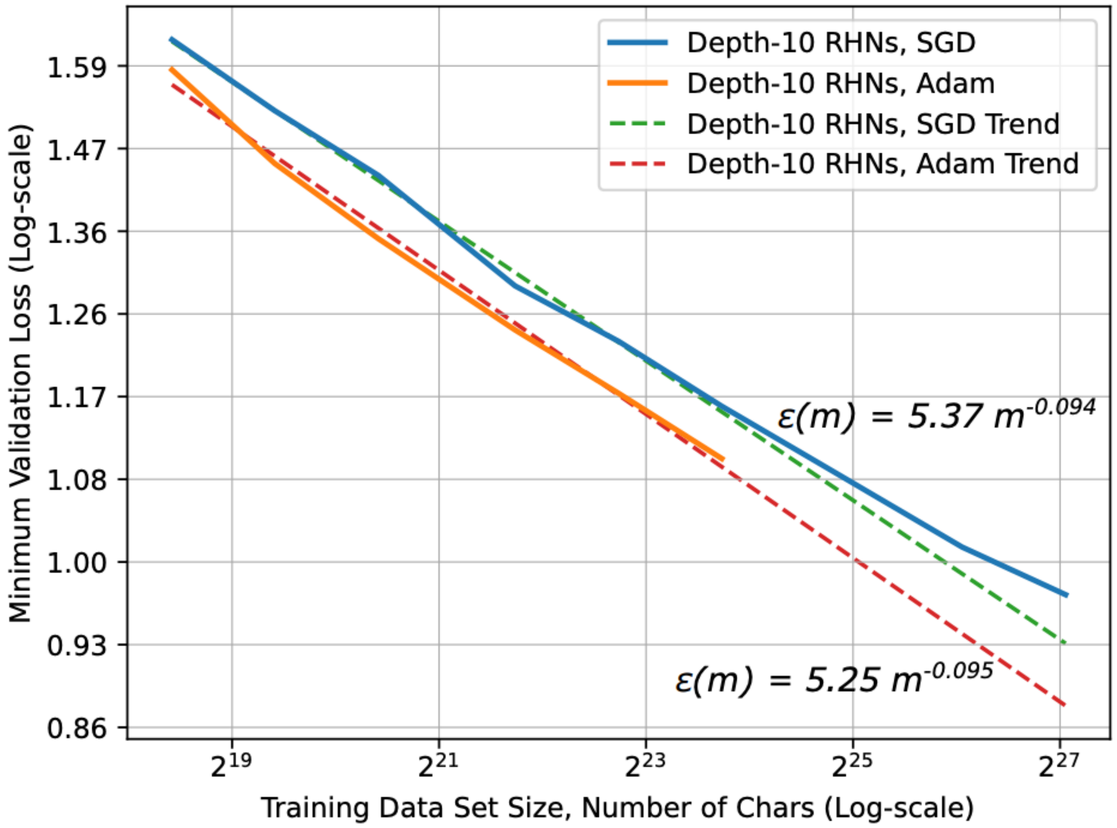
Model size results show that best-fit models grow sublinearly in the training shard size. Specifically, the best-fit 2-layer LSTM and depth-5 RHNs model sizes grow roughly with $\beta_p = 0.69 \pm 5\%$. The 4-layer LSTMs show slightly worse scaling with $\beta_p = 0.78$, suggesting they make less effective use of extra parameters on larger data sets. Despite the model size scaling differences, for a given model architecture, we can accurately predict the model size that will best fit increasingly larger data sets.

$$\text{model size} = \alpha |T_i|^{\beta_p}$$

Free parameter in power law

Training set size

Character language modeling



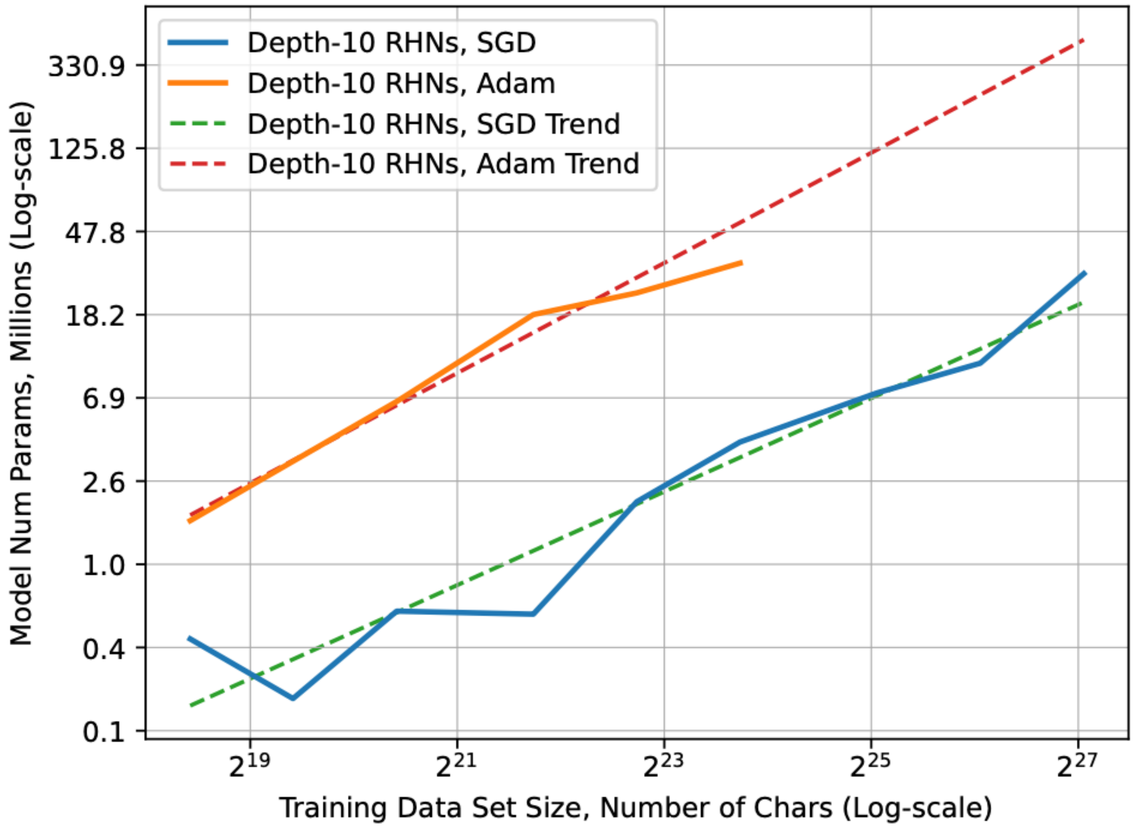
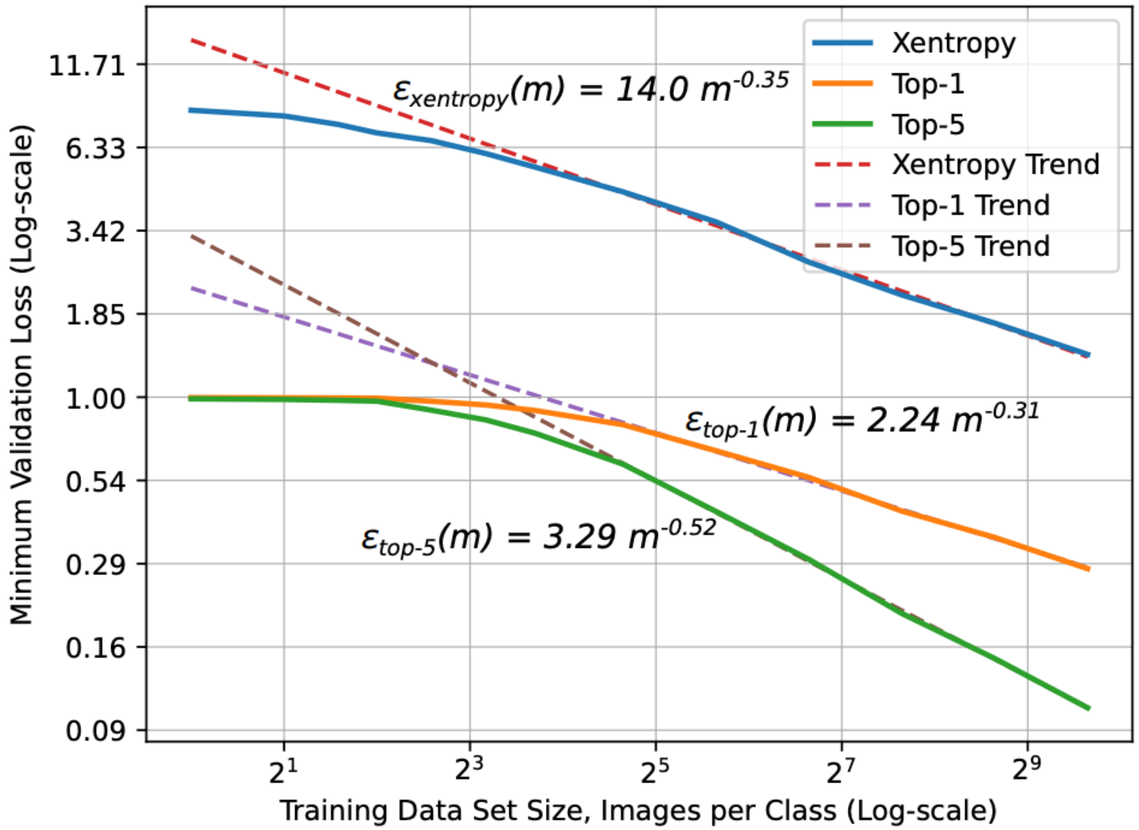


Figure 3: Learning curve and model size results and trends for character language models.

Image classification



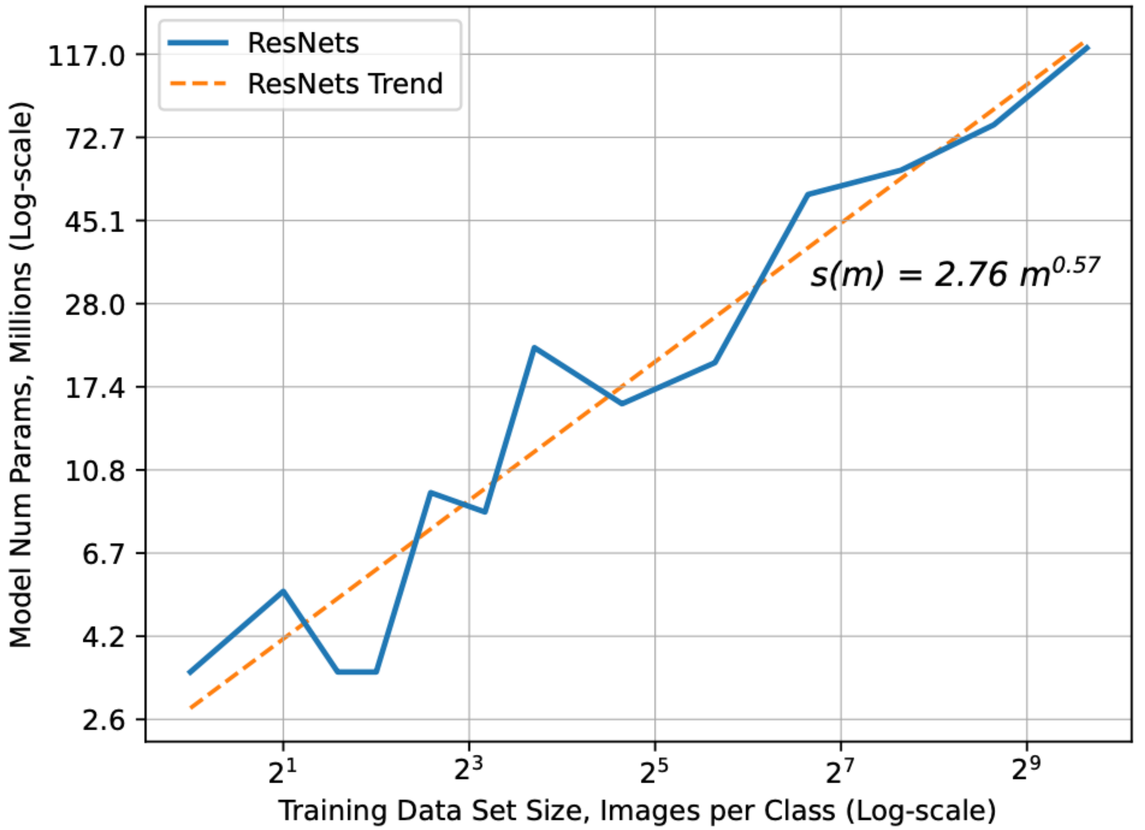
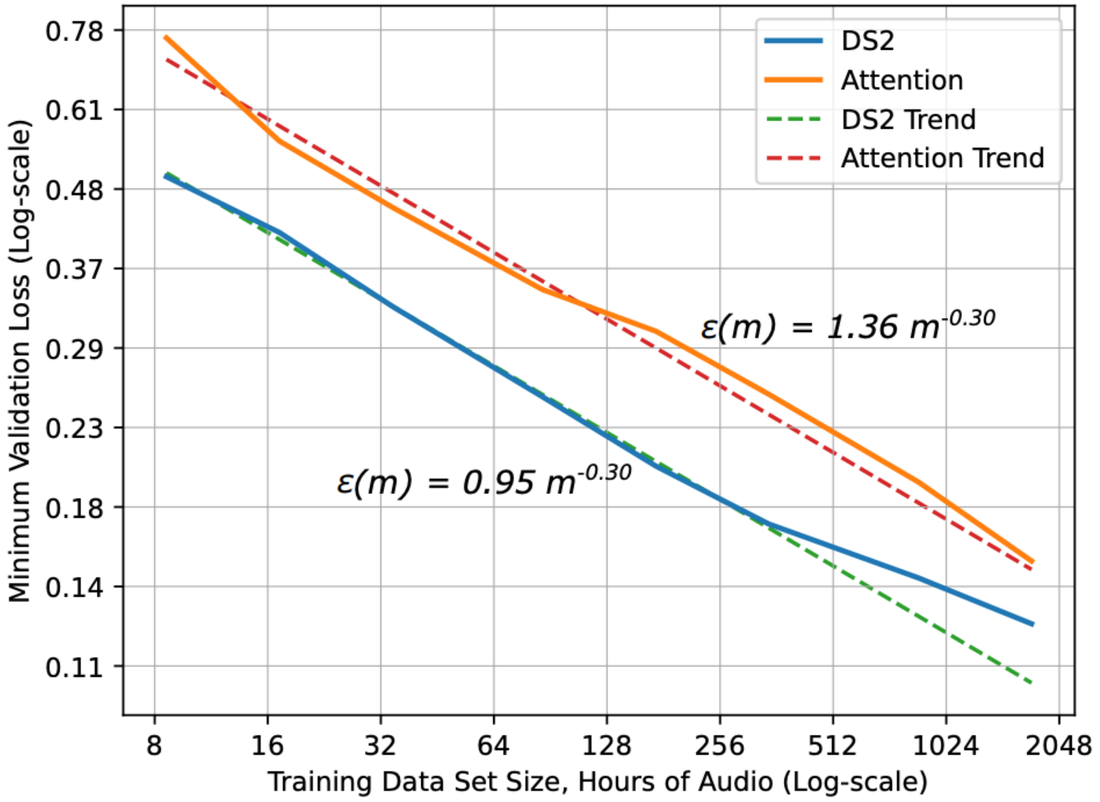


Figure 4: Learning curve and model size results and trends for ResNet image classification.

Speech recognition



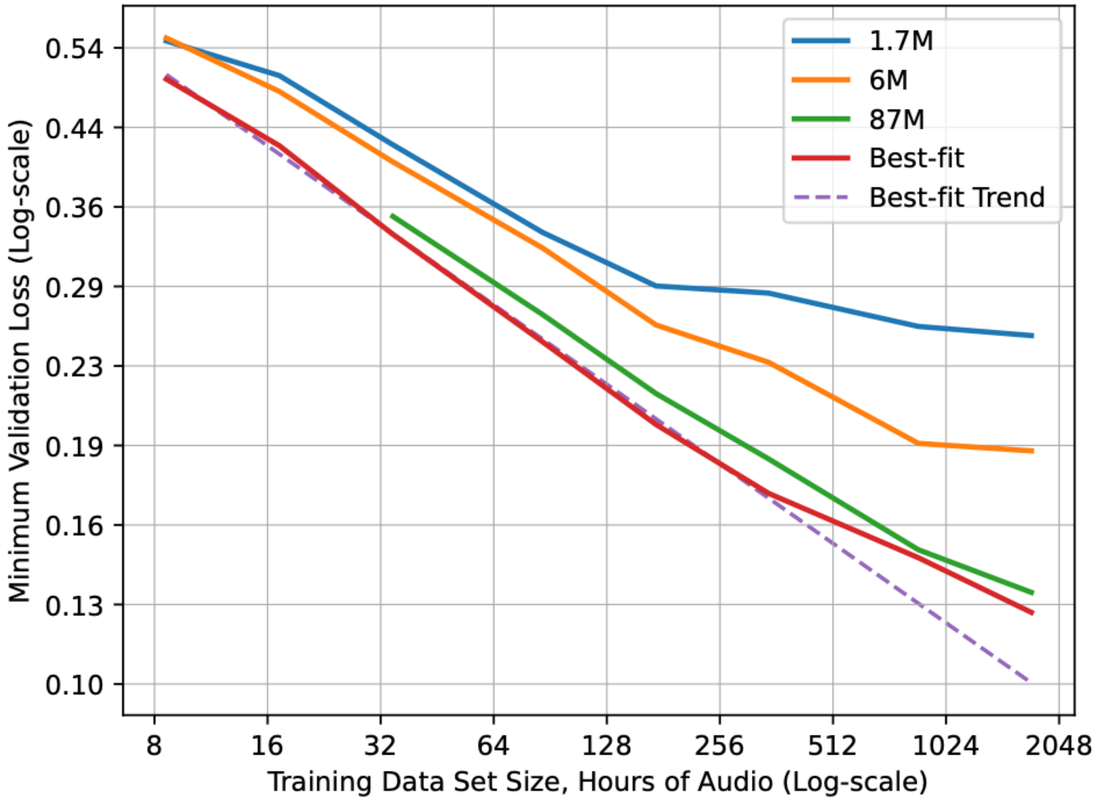


Figure 5: Learning curves for DS2 and attention speech models (left), and learning curves for various DS2 model sizes, 1.7M to 87M parameters (right).

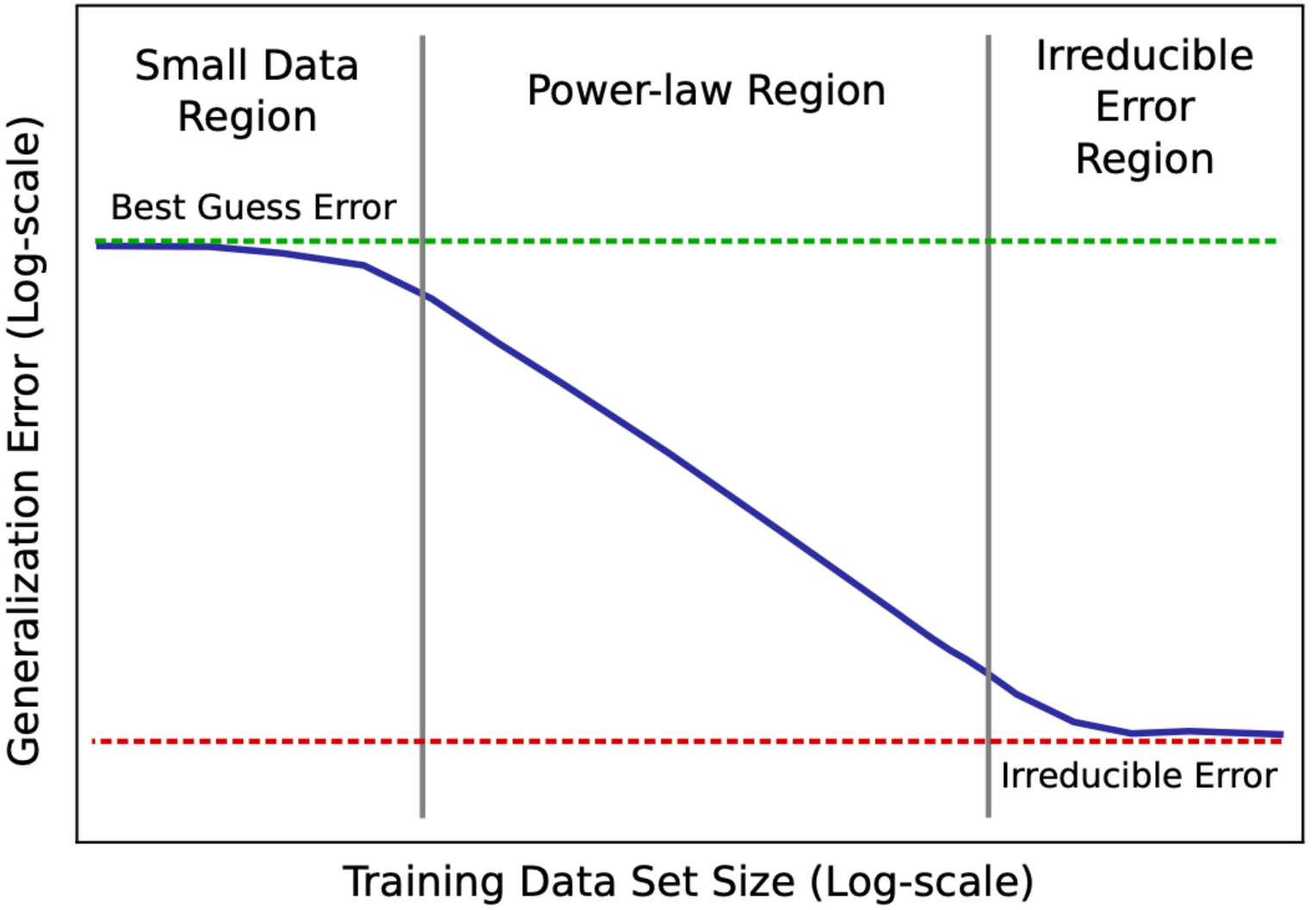


Figure 6: Sketch of power-law learning curves

Scaling Laws for Neural Language Models

Jared Kaplan *
Johns Hopkins University, OpenAI
jaredk@jhu.edu

Sam McCandlish*
OpenAI
sam@openai.com

Tom Henighan
OpenAI
henighan@openai.com

Tom B. Brown
OpenAI
tom@openai.com

Benjamin Chess
OpenAI
bchess@openai.com

Rewon Child
OpenAI
rewon@openai.com

Scott Gray
OpenAI
scott@openai.com

Alec Radford
OpenAI
alec@openai.com

Jeffrey Wu
OpenAI
jeffwu@openai.com

Dario Amodei
OpenAI
damodei@openai.com

Abstract

We study empirical scaling laws for language model performance on the cross-entropy loss. The loss scales as a power-law with model size, dataset size, and the amount of compute used for training, with some trends spanning more than seven orders of magnitude. Other architectural details such as network width or depth have minimal effects within a wide range. Simple equations govern the dependence of overfitting on model/dataset size and the dependence of training speed on model size. These relationships allow us to determine the optimal allocation of a fixed compute budget. Larger models are significantly more sample-efficient, such that optimally compute-efficient training involves training very large models on a relatively modest amount of data and stopping significantly before convergence.

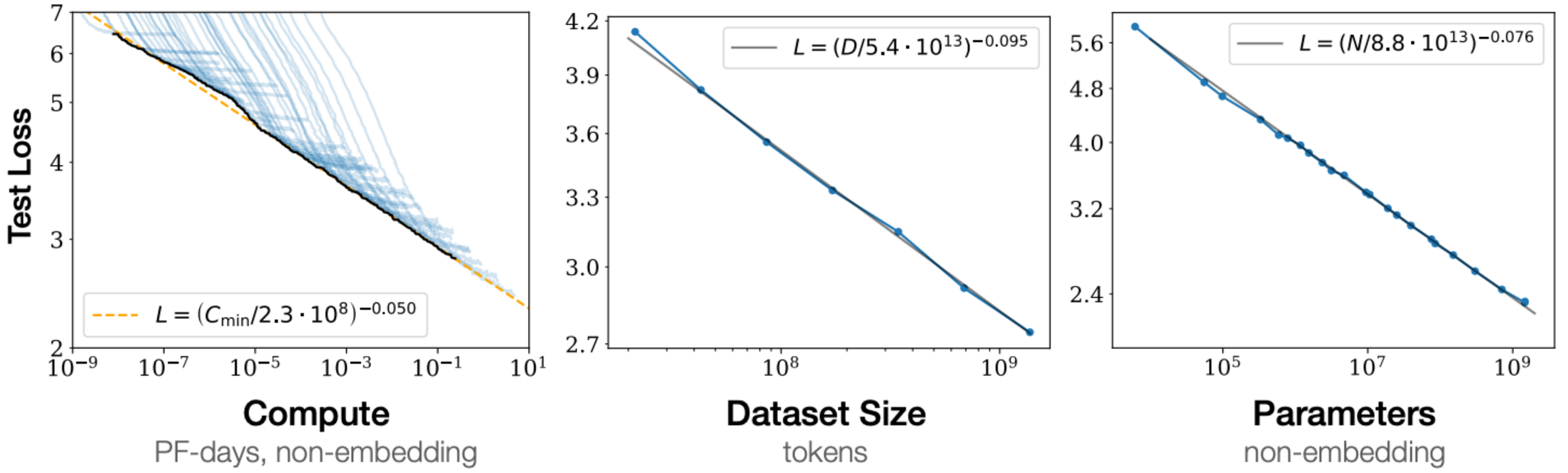


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

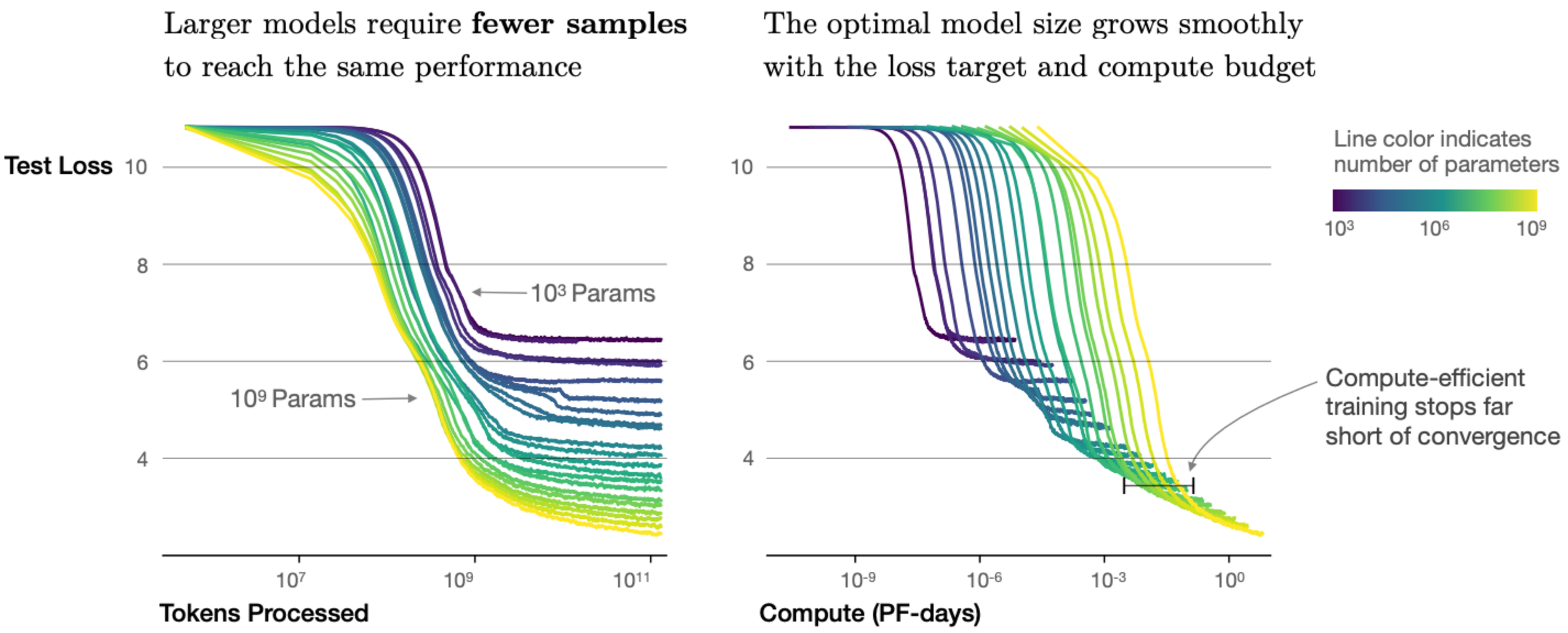


Figure 2 We show a series of language model training runs, with models ranging in size from 10^3 to 10^9 parameters (excluding embeddings).

Universality of overfitting: Performance improves predictably as long as we scale up N and D in tandem, but enters a regime of diminishing returns if either N or D is held fixed while the other increases. The performance penalty depends predictably on the ratio $N^{0.74}/D$, meaning that every time we increase the model size 8x, we only need to increase the data by roughly 5x to avoid a penalty. (Section 4)

Convergence is inefficient: When working within a fixed compute budget C but without any other restrictions on the model size N or available data D, we attain optimal performance by training *very large models* and stopping *significantly short of convergence* (see Figure 3). Maximally compute-efficient training would therefore be far more sample efficient than one might expect based on training small models to convergence, with data requirements growing very slowly as $D \sim C^{0.27}$ with training compute. (Section 6)

Convergence is inefficient: When working within a fixed compute budget C but without any other restrictions on the model size N or available data D, we attain optimal performance by training *very large models* and stopping *significantly short of convergence* (see Figure 3). Maximally compute-efficient training would therefore be far more sample efficient than one might expect based on training small models to convergence, with data requirements growing very slowly as $D \sim C^{0.27}$ with training compute. (Section 6)

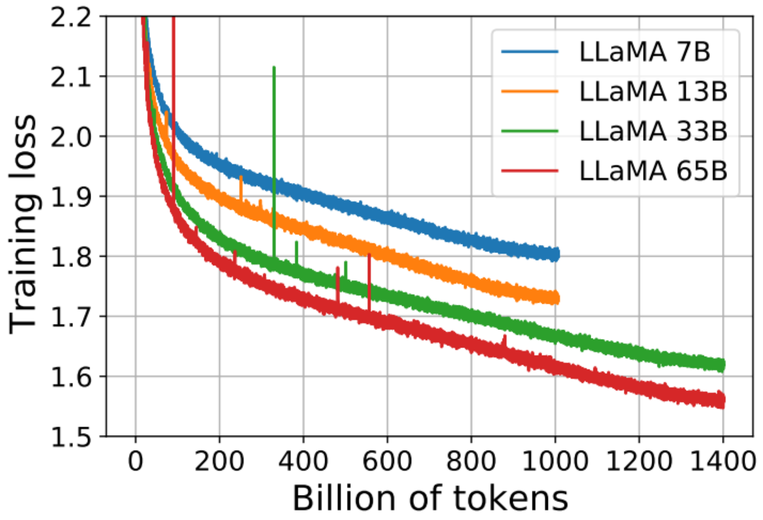


Figure 1: **Training loss over train tokens for the 7B, 13B, 33B, and 65 models.** LLaMA-33B and LLaMA-65B were trained on 1.4T tokens. The smaller models were trained on 1.0T tokens. All models are trained with a batch size of 4M tokens.

DeepMind

Training Compute-Optimal Large Language Models

Jordan Hoffmann*, Sebastian Borgeaud*, Arthur Mensch*, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals and Laurent Sifre*

*Equal contributions

We investigate the optimal model size and number of tokens for training a transformer language model under a given compute budget. We find that current large language models are significantly under-trained, a consequence of the recent focus on scaling language models whilst keeping the amount of training data constant. By training over 400 language models ranging from 70 million to over 16 billion parameters on 5 to 500 billion tokens, we find that for compute-optimal training, the model size and the number of training tokens should be scaled equally: for every doubling of model size the number of training tokens should also be doubled. We test this hypothesis by training a predicted compute-optimal model, *Chinchilla*, that uses the same compute budget as *Gopher* but with 70B parameters and 4× more more data. *Chinchilla* uniformly and significantly outperforms *Gopher* (280B), GPT-3 (175B), Jurassic-1 (178B), and Megatron-Turing NLG (530B) on a large range of downstream evaluation tasks. This also means that *Chinchilla* uses substantially less compute for fine-tuning and inference, greatly facilitating downstream usage. As a highlight, *Chinchilla* reaches a state-of-the-art average accuracy of 67.5% on the MMLU benchmark, greater than a 7% improvement over *Gopher*.

Table 1 | **Current LLMs**. We show five of the current largest dense transformer models, their size, and the number of training tokens. Other than LaMDA (Thoppilan et al., 2022), most models are trained for approximately 300 billion tokens. We introduce *Chinchilla*, a substantially smaller model, trained for much longer than 300B tokens.

Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
Chinchilla	70 Billion	1.4 Trillion

	Chinchilla	*Gopher*	GPT-3	MT-NLG 530B	Supervised SOTA
HellaSWAG	**80.8%**	79.2%	78.9%	80.2%	93.9%
PIQA	81.8%	81.8%	81.0%	**82.0%**	90.1%
Winogrande	**74.9%**	70.1%	70.2%	73.0%	91.3%
SIQA	**51.3%**	50.6%	-	-	83.2%
BoolQ	**83.7%**	79.3%	60.5%	78.2%	91.4%

Table 8 | **Zero-shot comparison on Common Sense benchmarks.** We show a comparison between *Chinchilla*, *Gopher*, and MT-NLG 530B on various Common Sense benchmarks. We see that *Chinchilla* matches or outperforms *Gopher* and GPT-3 on all tasks. On all but one *Chinchilla* outperforms the much larger MT-NLG 530B model.

Random	25.0%
Average human rater	34.5%
GPT-3 5-shot	43.9%
Gopher 5-shot	60.0%
***Chinchilla* 5-shot**	**67.6%**
Average human expert performance	*89.8%*
June 2022 Forecast	57.1%
June 2023 Forecast	63.4%

Table 6 | **Massive Multitask Language Understanding (MMLU).** We report the average 5-shot accuracy over 57 tasks with model and human accuracy comparisons taken from Hendrycks et al. (2020). We also include the average prediction for state of the art accuracy in June 2022/2023 made by 73 competitive human forecasters in Steinhardt (2021).

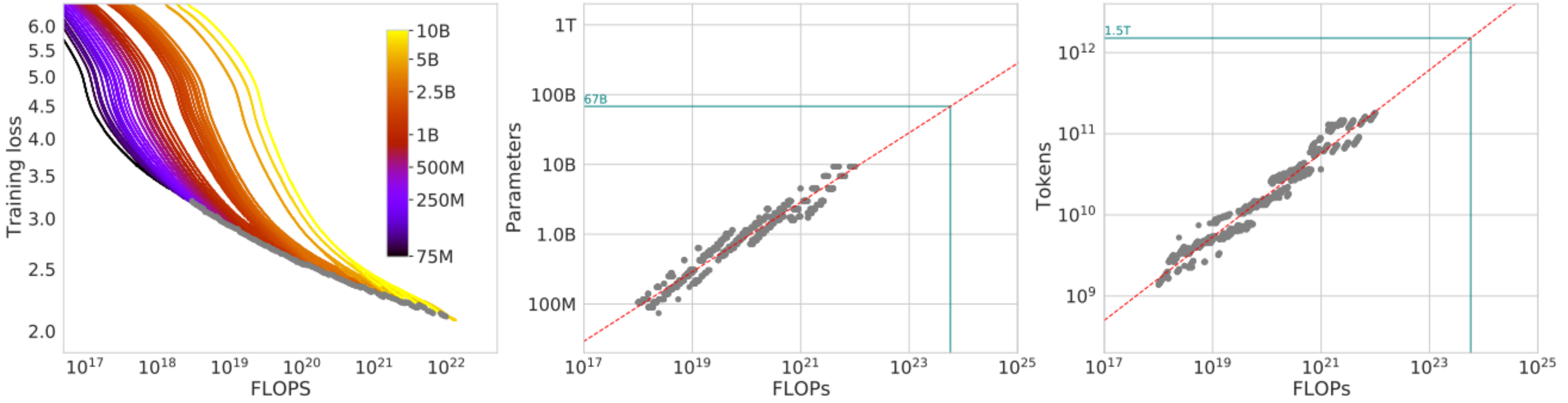
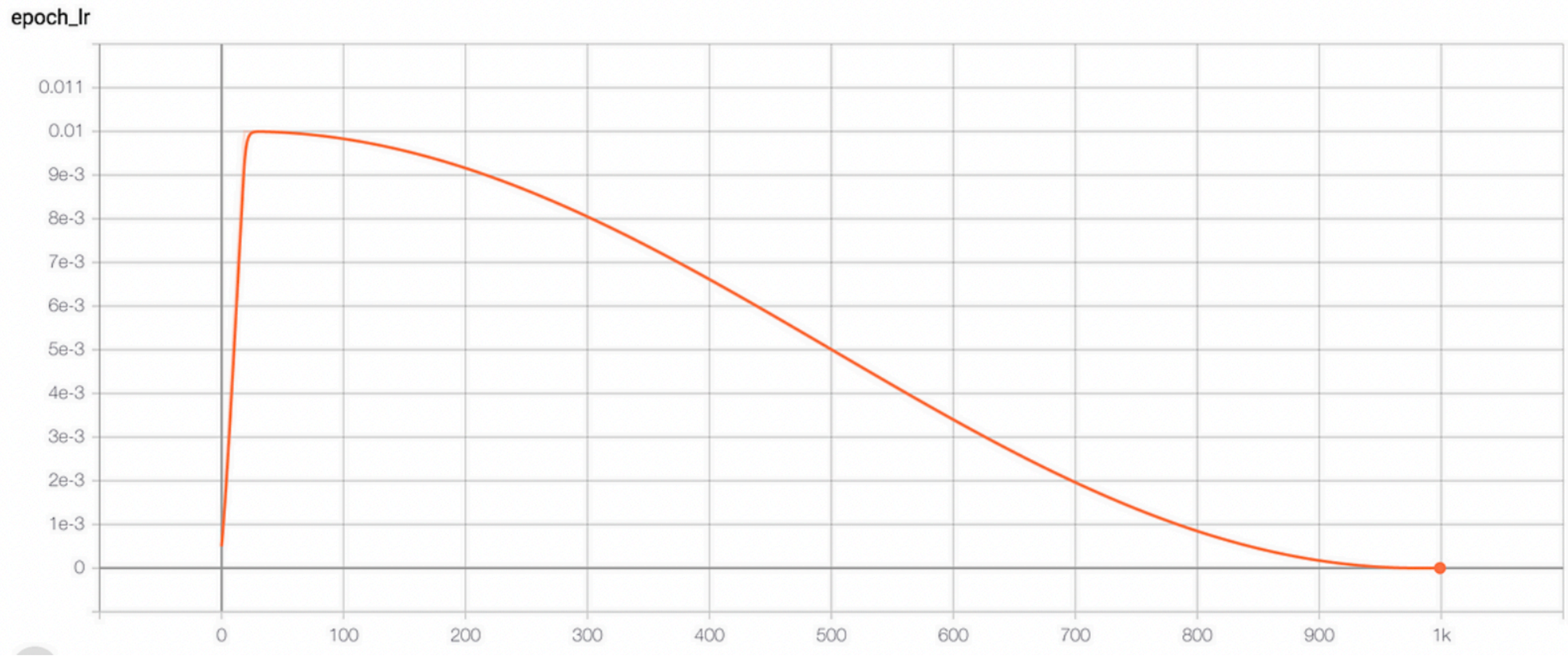


Figure 2 | **Training curve envelope.** On the **left** we show all of our different runs. We launched a range of model sizes going from 70M to 10B, each for four different cosine cycle lengths. From these curves, we extracted the envelope of minimal loss per FLOP, and we used these points to estimate the optimal model size (**center**) for a given compute budget and the optimal number of training tokens (**right**). In green, we show projections of optimal model size and training token count based on the number of FLOPs used to train *Gopher* (5.76×10^{23}).

Cosine learning rate depends on training duration!



Early stopping a training run does **not** give the optimal performance for the shorter training duration → need to run a separate experiment for each training duration.

Table 2 | **Estimated parameter and data scaling with increased training compute.** The listed values are the exponents, a and b, on the relationship $N_{opt} \propto C^a$ and $D_{opt} \propto C^b$. Our analysis suggests a near equal scaling in parameters and data with increasing compute which is in clear contrast to previous work on the scaling of large models. The 10^{th} and 90^{th} percentiles are estimated via bootstrapping data (80% of the dataset is sampled 100 times) and are shown in parenthesis.

Approach	Coeff. a where $N_{opt} \propto C^a$	Coeff. b where $D_{opt} \propto C^b$
1. Minimum over training curves	0.50 (0.488, 0.502)	0.50 (0.501, 0.512)
2. IsoFLOP profiles	0.49 (0.462, 0.534)	0.51 (0.483, 0.529)
3. Parametric modelling of the loss	0.46 (0.454, 0.455)	0.54 (0.542, 0.543)
Kaplan et al. (2020)	0.73	0.27

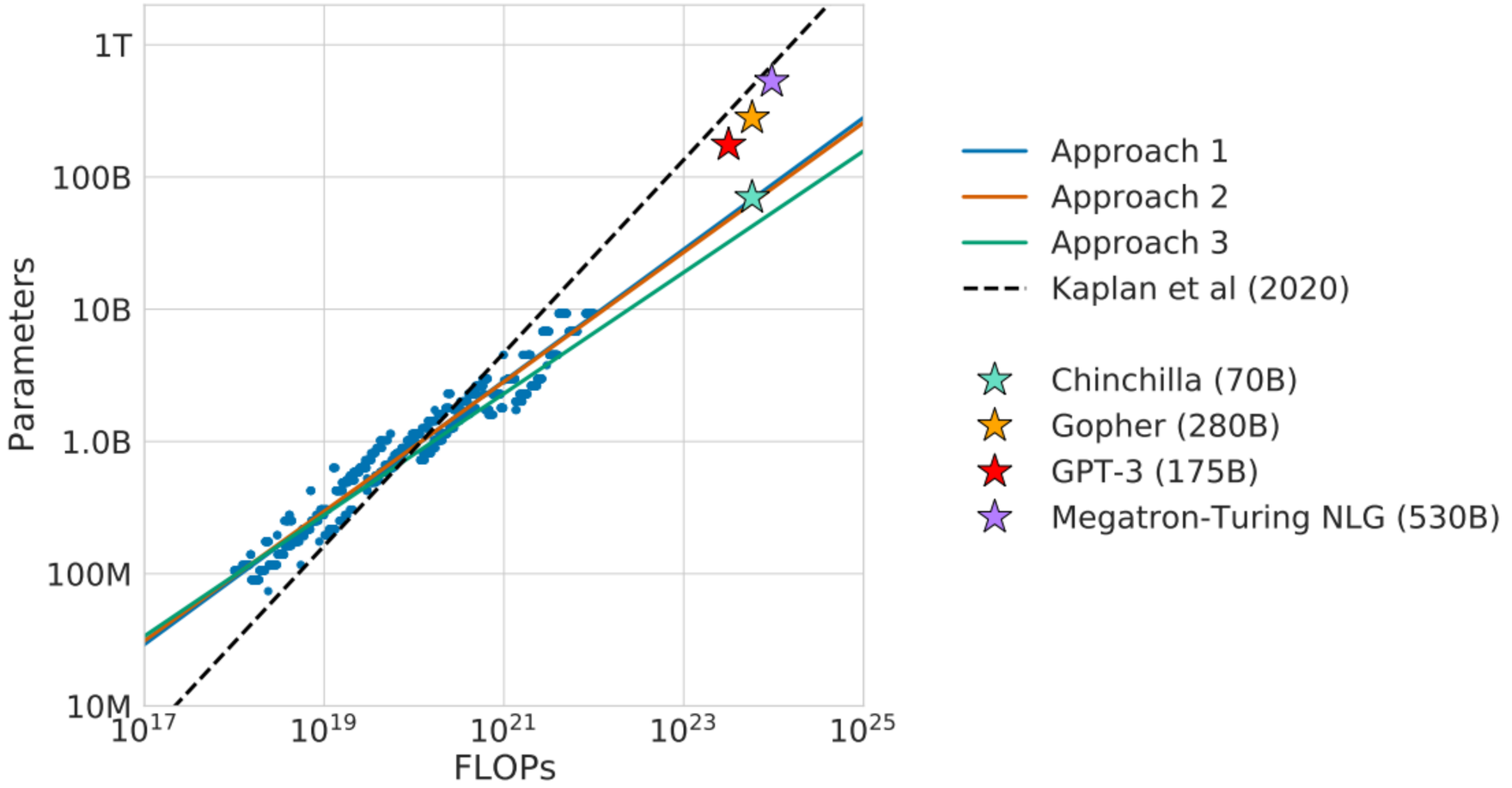


Figure 1 | **Overlaid predictions.** We overlay the predictions from our three different approaches, along with projections from Kaplan et al. (2020). We find that all three methods predict that current large models should be substantially smaller and therefore trained much longer than is currently done. In Figure A3, we show the results with the predicted optimal tokens plotted against the optimal number of parameters for fixed FLOP budgets. ***Chinchilla* outperforms *Gopher* and the other large models (see Section 4.2).**

Table 3 | **Estimated optimal training FLOPs and training tokens for various model sizes.** For various model sizes, we show the projections from Approach 1 of how many FLOPs and training tokens would be needed to train compute-optimal models. The estimates for Approach 2 & 3 are similar (shown in Section D.3)

Parameters	FLOPs	FLOPs (in *Gopher* unit)	Tokens
400 Million	1.92e+19	1/29,968	8.0 Billion
1 Billion	1.21e+20	1/4,761	20.2 Billion
10 Billion	1.23e+22	1/46	205.1 Billion
67 Billion	5.76e+23	1	1.5 Trillion
175 Billion	3.85e+24	6.7	3.7 Trillion
280 Billion	9.90e+24	17.2	5.9 Trillion
520 Billion	3.43e+25	59.5	11.0 Trillion
1 Trillion	1.27e+26		
10 Trillion	1.30e+28		

LLaMA:

params	dimension	n heads	n layers	learning rate	batch size	n tokens
6.7B	4096	32	32	$3.0e^{-4}$	4M	1.0T
13.0B	5120	40	40	$3.0e^{-4}$	4M	1.0T
32.5B	6656	52	60	$1.5e^{-4}$	4M	1.4T
65.2B	8192	64	80	$1.5e^{-4}$	4M	1.4T

Table 2: **Model sizes, architectures, and optimization hyper-parameters.**

Multimodal learning

[Deng, Dong, Socher, Li, Li, Fei-Fei'09]

[Russakovsky, Deng, Su, Krause, Satheesh, Ma, Huang, Karpathy, Khosla, Bernstein, Berg Fei-Fei'15]

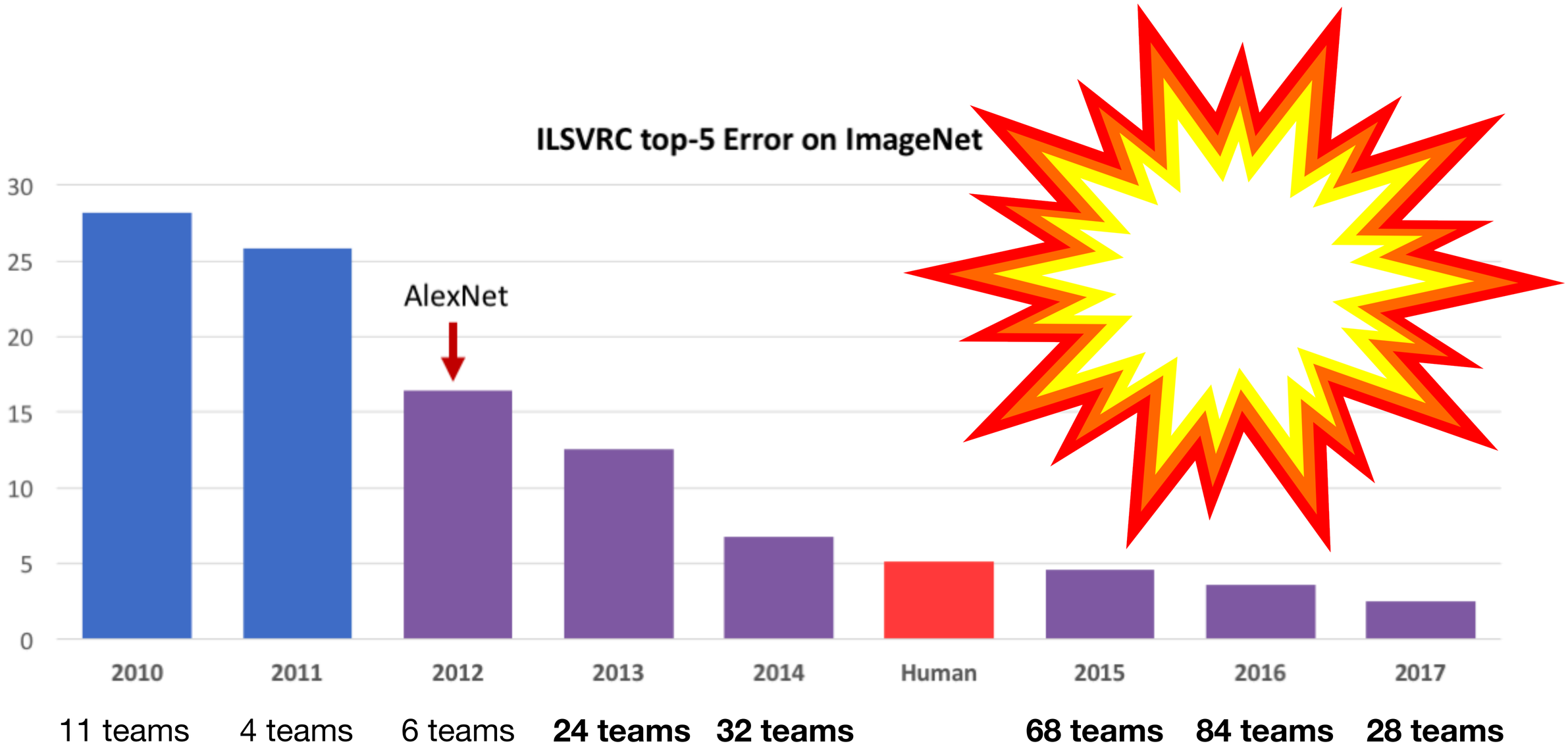
ImageNet

Large **image classification** dataset: 1.2 mio training images, 1,000 image classes.

Golden retriever

Great white shark

Minibus



Large improvement, new method → Tremendous interest from the community

Image Classification (pre 2021)

- Training data with labels {(x, y)}
  - For instance, x is an image of a digit and y is a class {0,…9}
- Use training set to learn a new model *f* which takes in an image and outputs one of the classes.
- At test time, see new images x’ and classify as *f*(x’).

2012 - 2021: Models larger, researchers trained for longer, training set sizes increased.

But: overall dataset-specific training paradigm stayed largely the same.

(Each image classification problem requires its own independent dataset)

One exception: ImageNet pre-training

CNN Features off-the-shelf: an Astounding Baseline for Recognition

Ali Sharif Razavian Hossein Azizpour Josephine Sullivan Stefan Carlsson
CVAP, KTH (Royal Institute of Technology)
Stockholm, Sweden
{razavian,azizpour,sullivan,stefanc}@csc.kth.se

Abstract

Recent results indicate that the generic descriptors extracted from the convolutional neural networks are very powerful. This paper adds to the mounting evidence that this is indeed the case. We report on a series of experiments conducted for different recognition tasks using the publicly available code and model of the `OverFeat` *network which was trained to perform object classification on ILSVRC13. We use features extracted from the* `OverFeat` *network as a generic image representation to tackle the diverse range of recognition tasks of object image classification, scene recognition, fine grained recognition, attribute detection and image retrieval applied to a diverse set of datasets. We selected these tasks and datasets as they gradually move further away from the original task and data the* `OverFeat` *network was trained to solve. Astonishingly, we report consistent superior results compared to the highly tuned state-of-the-art systems in all the visual classification*

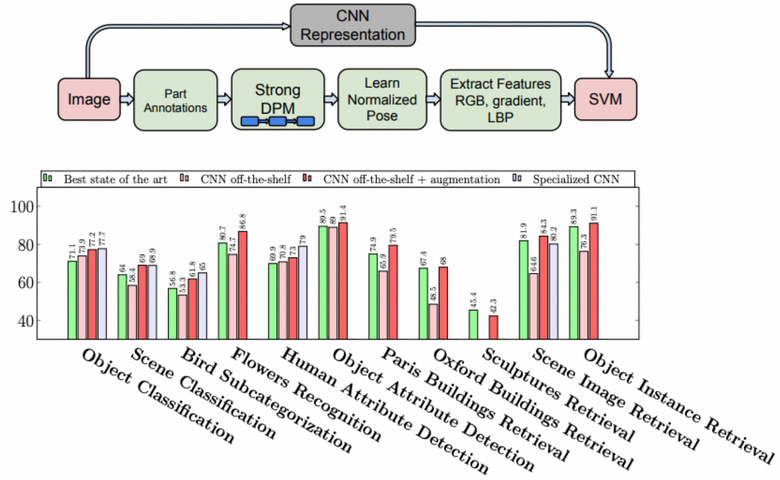
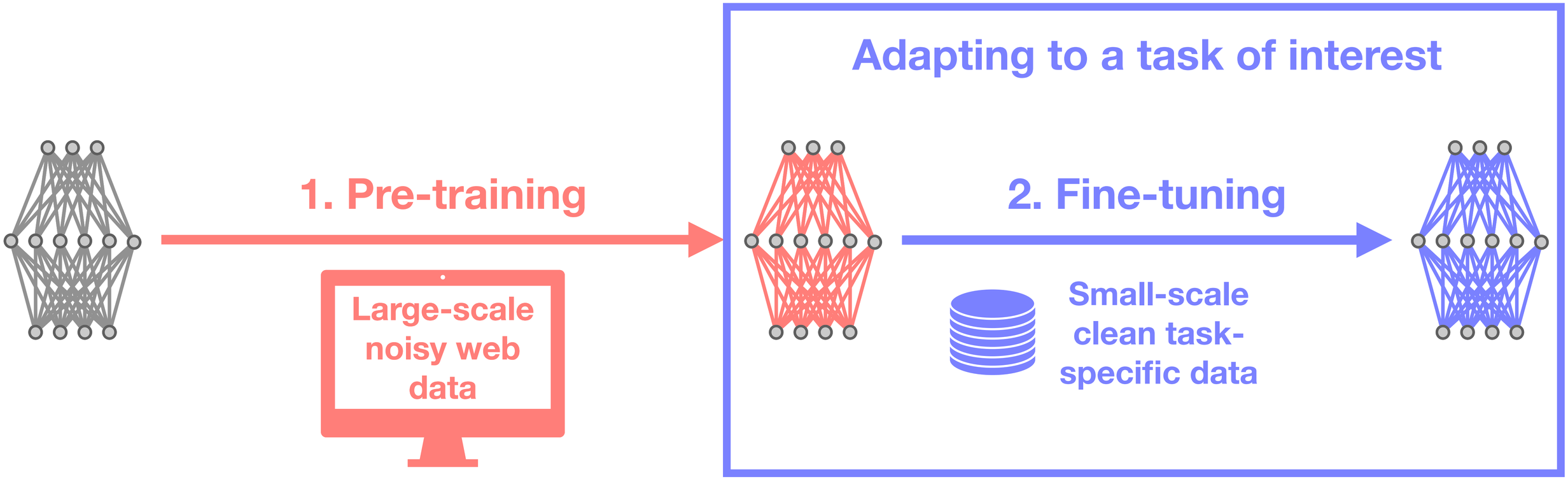


Figure 1: **top**) CNN representation replaces pipelines of s.o.a methods and achieve better results. e.g. DPD [50].
bottom) Augmented CNN representation with linear SVM consistently outperforms s.o.a. on multiple tasks. Specialized CNN refers to other works which specifically designed the CNN for their task

Fine-tuning

State-of-the-art ML models often come from a two-step process.



Analogy: pre-GPT-2 era in NLP

Improving Language Understanding by Generative Pre-Training

Alec Radford OpenAI alec@openai.com

Karthik Narasimhan OpenAI karthikn@openai.com

Tim Salimans OpenAI tim@openai.com

Ilya Sutskever OpenAI ilyasu@openai.com

Abstract

Natural language understanding comprises a wide range of diverse tasks such as textual entailment, question answering, semantic similarity assessment, and document classification. Although large unlabeled text corpora are abundant, labeled data for learning these specific tasks is scarce, making it challenging for discriminatively trained models to perform adequately. We demonstrate that large gains on these tasks can be realized by *generative pre-training* of a language model on a diverse corpus of unlabeled text, followed by *discriminative fine-tuning* on each specific task. In contrast to previous approaches, we make use of task-aware input transformations during fine-tuning to achieve effective transfer while requiring minimal changes to the model architecture. We demonstrate the effectiveness of our approach on a wide range of benchmarks for natural language understanding. Our general task-agnostic model outperforms discriminatively trained models that use architectures specifically crafted for each task, significantly improving upon the state of the art in 9 out of the 12 tasks studied. For instance, we achieve absolute improvements of 8.9% on commonsense reasoning (Stories Cloze Test), 5.7% on question answering (RACE), and 1.5% on textual entailment (MultiNLI).

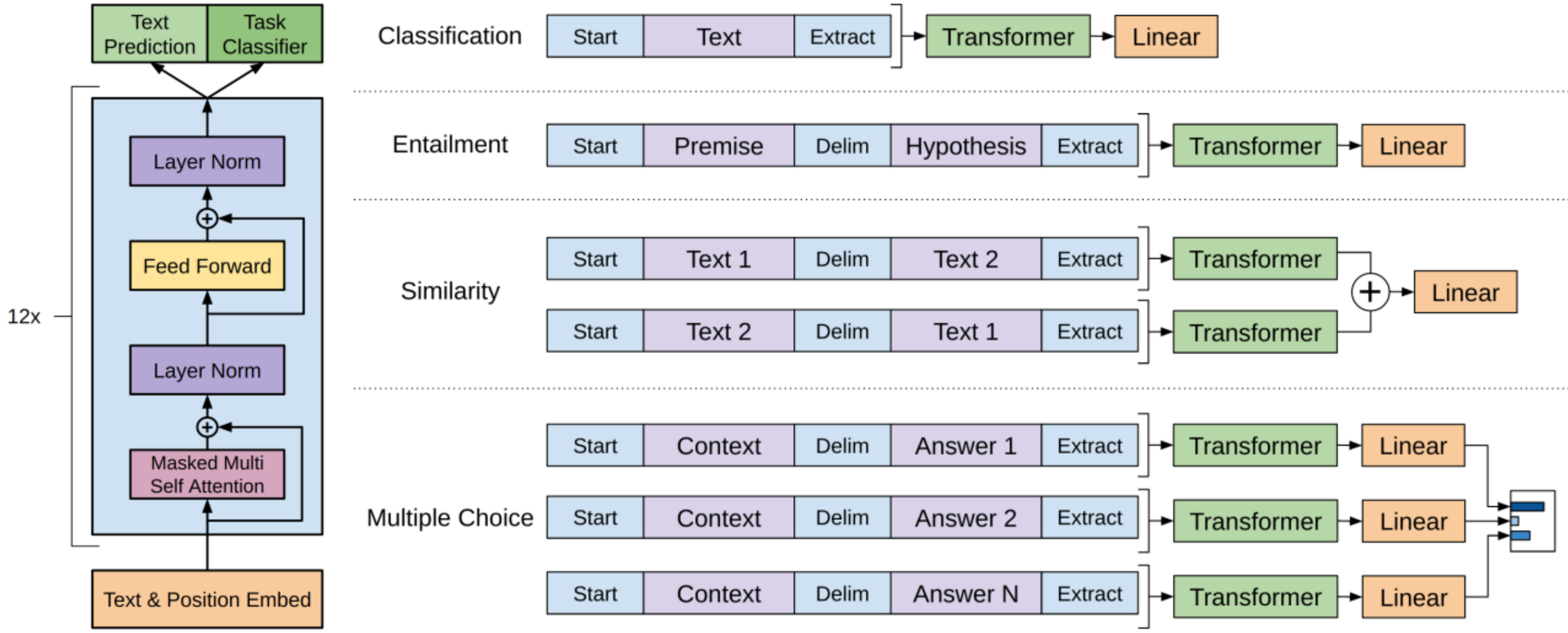


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

Limitations of the pre-training / fine-tuning paradigm

Language Models are Unsupervised Multitask Learners

Alec Radford [*1] Jeffrey Wu [*1] Rewon Child [1] David Luan [1] Dario Amodei [**1] Ilya Sutsk

The GPT-2 paper (2019)

1. Introduction

Machine learning systems now excel (in expectation) at tasks they are trained for by using a combination of large datasets, high-capacity models, and supervised learning (Krizhevsky et al., 2012) (Sutskever et al., 2014) (Amodei et al., 2016). Yet these systems are brittle and sensitive to slight changes in the data distribution (Recht et al., 2018) and task specification (Kirkpatrick et al., 2017). Current systems are better characterized as narrow experts rather than

Two key problems:

- Need **new labeled data** for each new problem
- Models are not robust under **distribution shift**

Do CIFAR-10 Classifiers Generalize to CIFAR-10?

Benjamin Recht
UC Berkeley

Rebecca Roelofs
UC Berkeley

Ludwig Schmidt
MIT

Vaishaal Shankar
UC Berkeley

Do ImageNet Classifiers Generalize to ImageNet?

Benjamin Recht
UC Berkeley

Rebecca Roelofs
UC Berkeley

Ludwig Schmidt
UC Berkeley

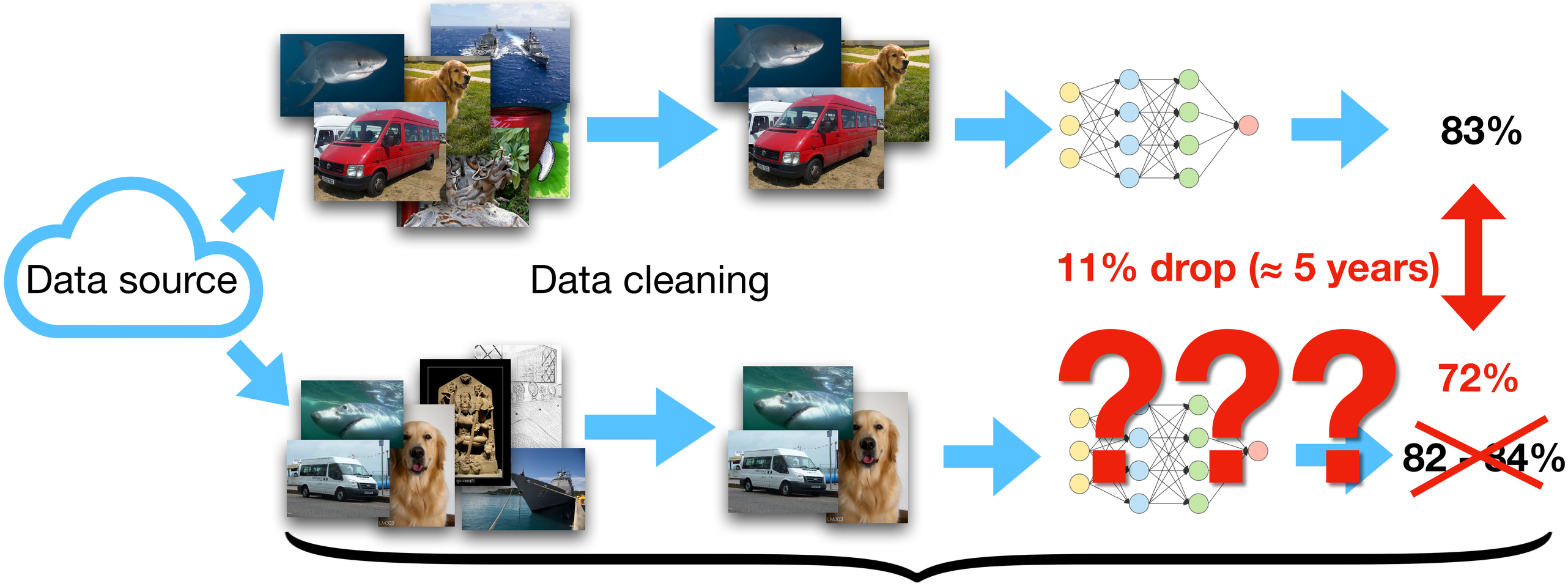
Vaishaal Shankar
UC Berkeley

Abstract

We build new test sets for the CIFAR-10 and ImageNet datasets. Both benchmarks have been the focus of intense research for almost a decade, raising the danger of overfitting to excessively re-used test sets. By closely following the original dataset creation processes, we test to what extent current classification models generalize to new data. We evaluate a broad range of models and find accuracy drops of 3% – 15% on CIFAR-10 and 11% – 14% on ImageNet. However, accuracy gains on the original test sets translate to larger gains on the new test sets. Our results suggest that the accuracy drops are not caused by adaptivity, but by the models' inability to generalize to slightly "harder" images than those found in the original test sets.

Generalization

At least, the classifiers should perform similarly well on new data from the **same source**.



Our experiment: sample a new ImageNet test set *nearly* i.i.d.

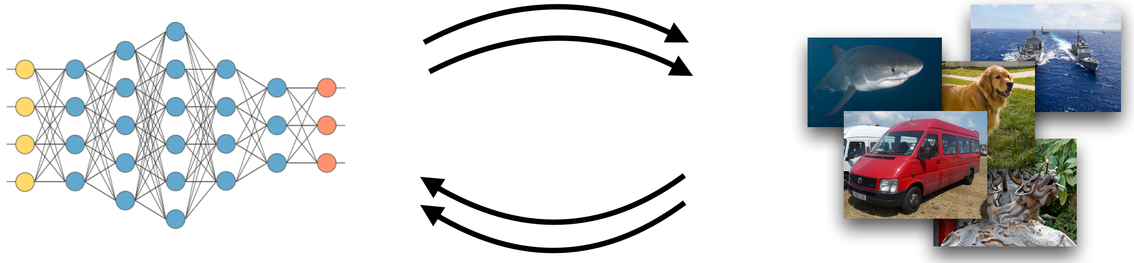
Overfitting

?

Three Forms of Overfitting

1. Test error ≥ training error

2. Overfitting through test set re-use



Model

Test Set

3. Distribution shift

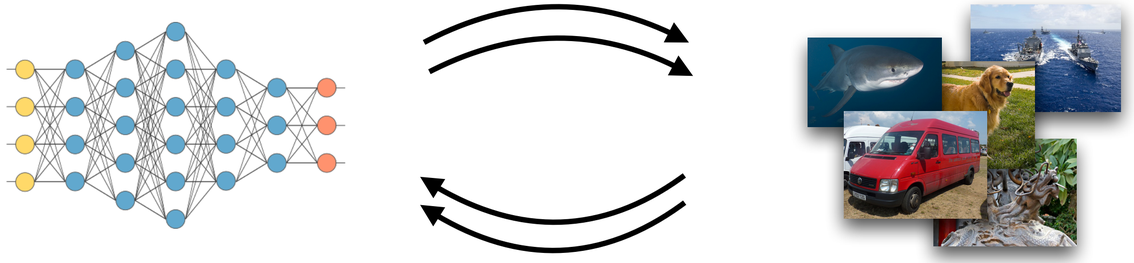
Original Test Set

New Test Set

Three Forms of Overfitting

1. Test error ≥ training error

2. Overfitting through test set re-use



Model

Test Set

3. Distribution shift

Original Test Set

New Test Set

Two Possible Causes

New test accuracy

Overfitting through test set re-use

Distribution shift

$$\underbrace{\widehat{\text{acc}}_S(f) - \widehat{\text{acc}}_{S'}(f)}_{\approx 11\%} =$$

Original test accuracy (orig. test set S, new S')

Generalization error (≈ 1%)

$$\widehat{\text{acc}}_S(f) = \frac{1}{|S|} \sum_{(x,y) \in S} \mathbb{1}[f(x) = y]$$

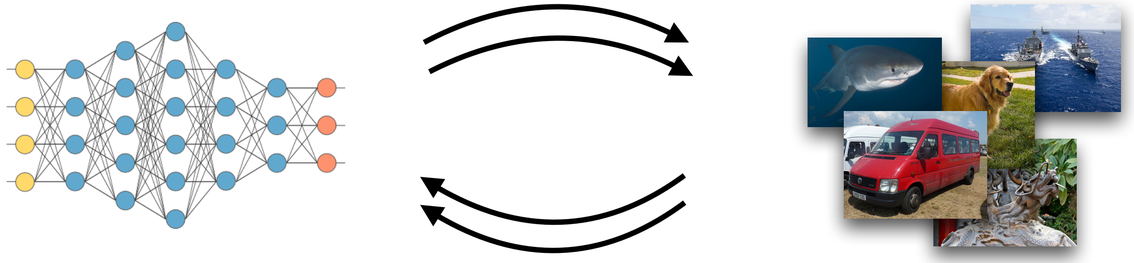
$$\text{acc}_D(f) = \mathbb{E}_{(x,y) \sim D} \mathbb{1}[f(x) = y]$$

(S is drawn from D)

Three Forms of Overfitting

1. Test error ≥ training error

2. Overfitting through test set re-use

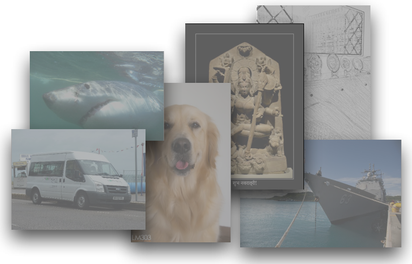


Model

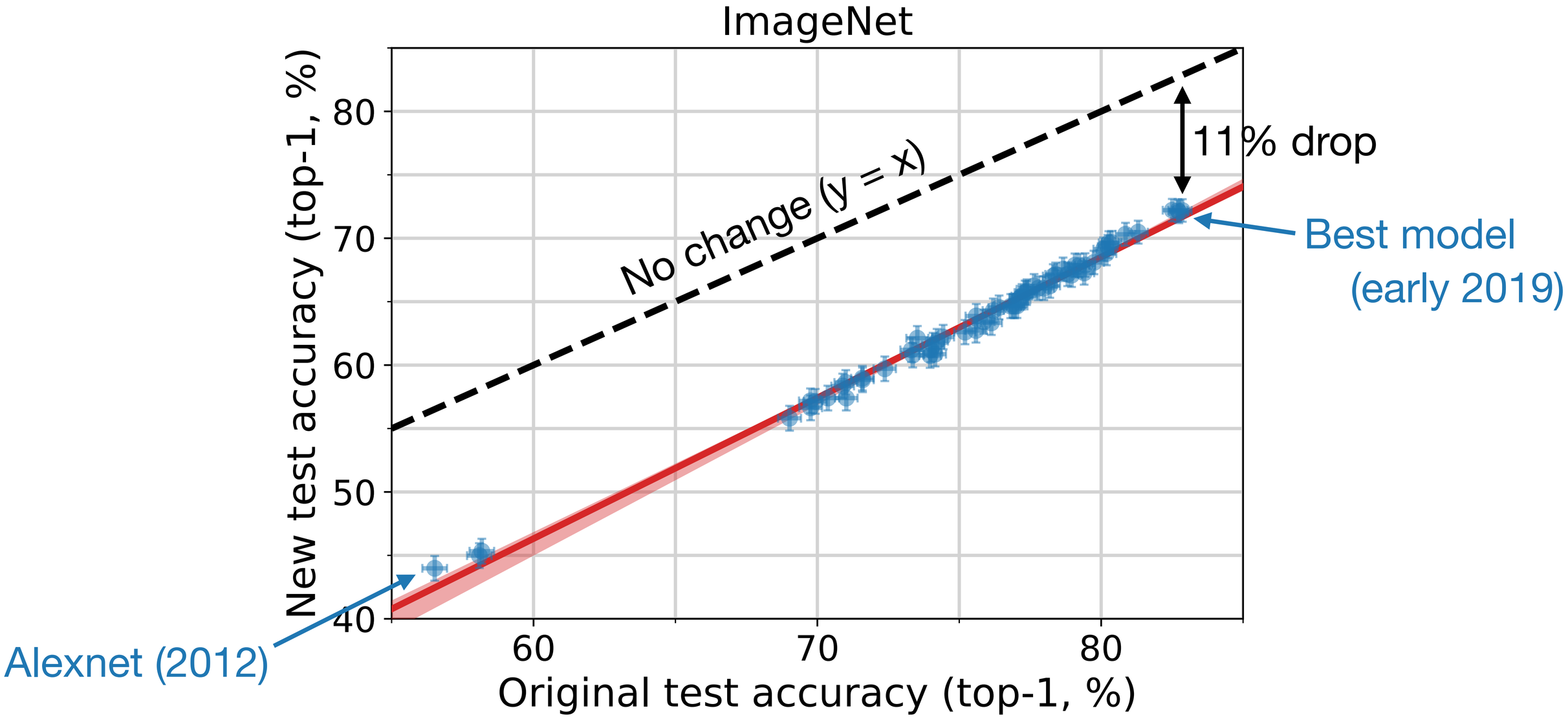
Test Set

3. Distribution shift

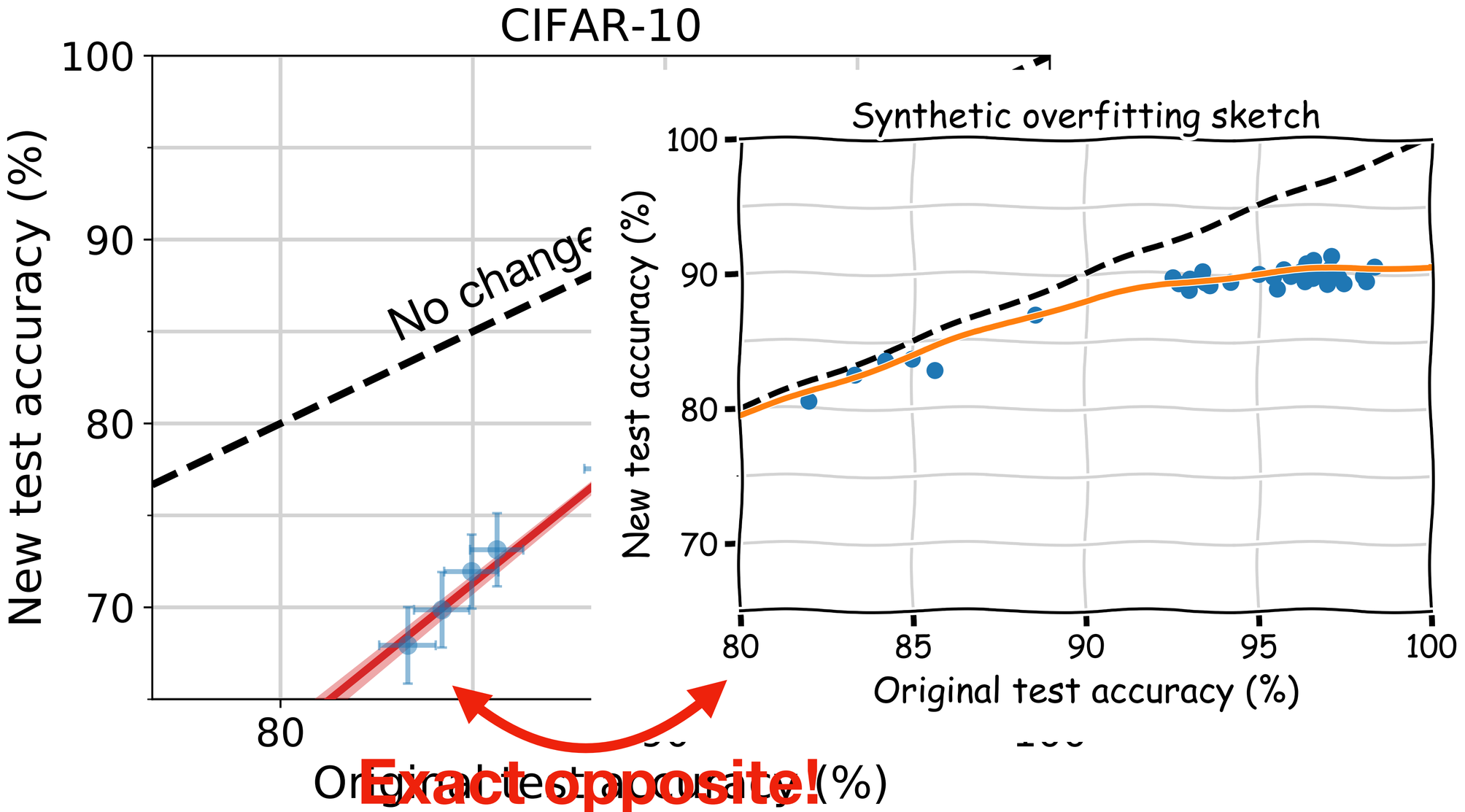
Original Test Set



New Test Set



- The best models on the original test set stay the best models on the new test set.
- All models see a substantial drop in accuracy.



Later models see a **smaller** drop in accuracy.

AutoAugment vs. ResNet: 4.9% difference on CIFAR-10

AutoAugment vs. ResNet: 10.3% difference on CIFAR-10.1

Overfitting Is Surprisingly Absent

No overfitting despite 10 years of test set re-use on CIFAR-10 and ImageNet.

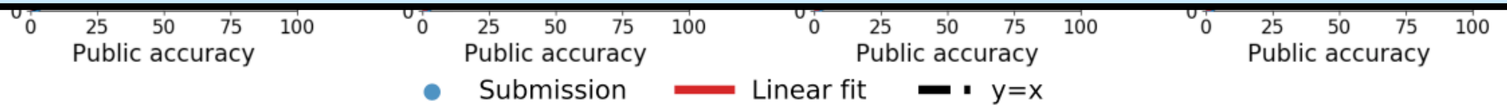
➡ Relative ordering preserved. Progress is real!

MNIST: similar conclusions in [Yadav, Bottou'19]

no overfitting after 20+ years of MNIST

Kaggle: Meta-analysis of 120 ML competitions [Roelofs, Fridovich-Keil, Miller, Shankar, Hardt, Recht, **Schmidt** '19]

Our results unambiguously confirm the trends observed by Recht et al. [2018, 2019]: although the misclassification rates are slightly off, classifier ordering and model selection remain broadly reliable.



Two Possible Causes

New test accuracy

Overfitting through test set re-use **(≈ 0%)**

Distribution shift

$$\underbrace{\widehat{\text{acc}}_S(f) - \widehat{\text{acc}}_{S'}(f)}_{\approx 11\%} = \cancel{\widehat{\text{acc}}_S(f) - \text{acc}_D(f)} + \text{acc}_D(f) - \text{acc}_{D'}(f) + \text{acc}_{D'}(f) - \widehat{\text{acc}}_{S'}(f)$$

Original test accuracy (orig. test set S, new S')

Generalization error (≈ 1%)

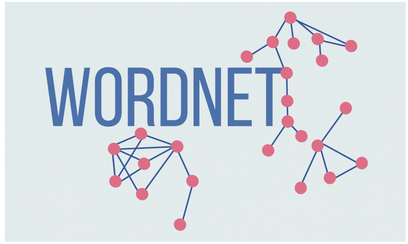
$$\widehat{\text{acc}}_S(f) = \frac{1}{|S|} \sum_{(x,y) \in S} \mathbb{1}[f(x) = y]$$

$$\text{acc}_D(f) = \mathbb{E}_{(x,y) \sim D} \mathbb{1}[f(x) = y]$$

(S is drawn from D)

ImageNet Creation Process

Detailed description in [Deng, Dong, Socher, Li, Li, Fei-Fei'09]:



1. Find relevant search keywords for each class from **WordNet** (e.g., "goldfish", "Carassius auratus" for wnid "n01443537")

2. Search for images on **Flickr**

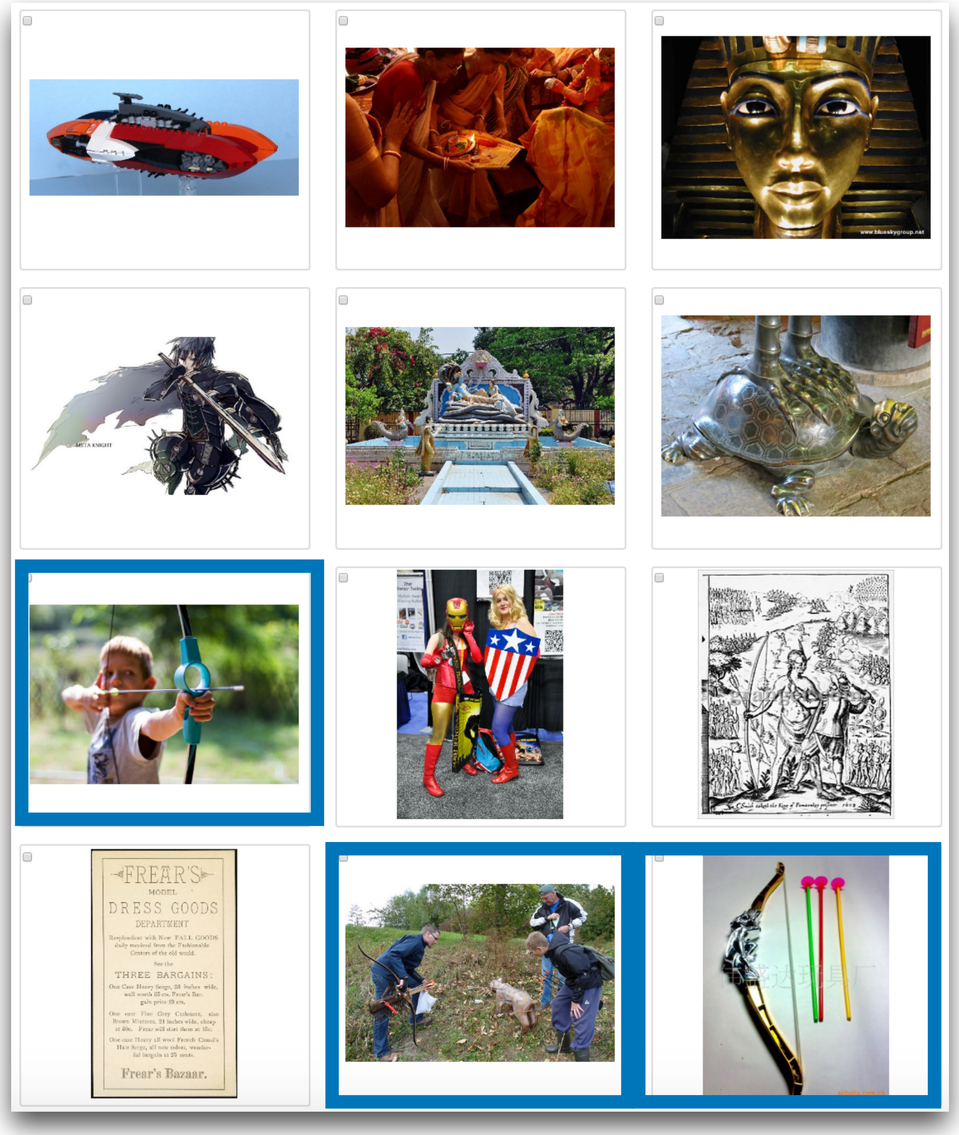
3. Show images to **MTurk** workers ← Likely source of distribution shift

4. Sample a class-balanced dataset

We replicated this process as closely as possible.

Data Cleaning With MTurk

Instructions: Select all images containing a bow.

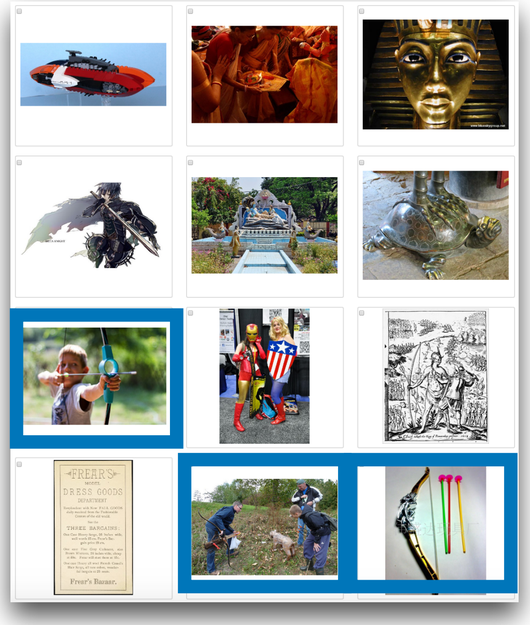


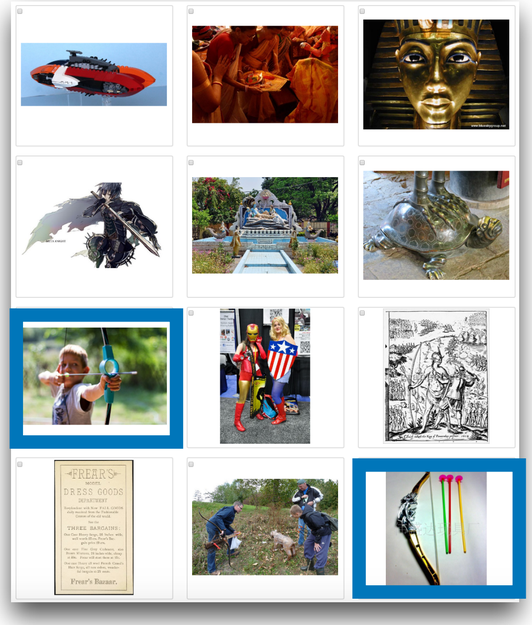
Data Cleaning With MTurk


Worker 1

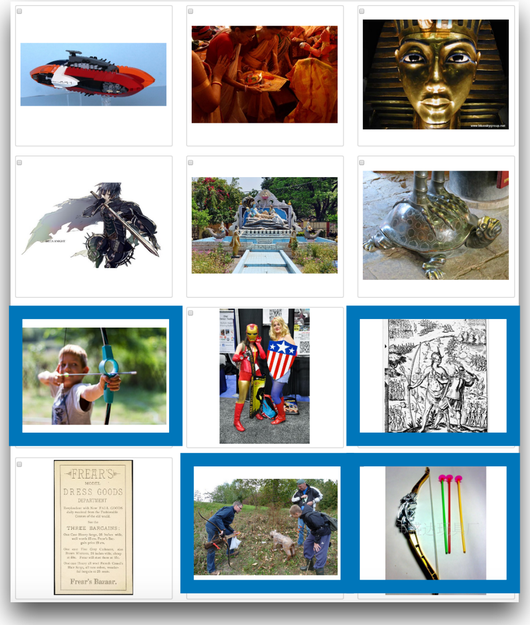

Worker 2


Worker 10

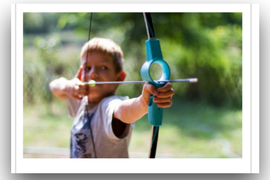




...



Main quantity: **selection frequency** $= \dfrac{\textbf{Number of workers who selected image } i}{\textbf{Number of workers who saw image } i}$


: 1.0


: 1.0


: 0.67


: 0.33


: 0.0

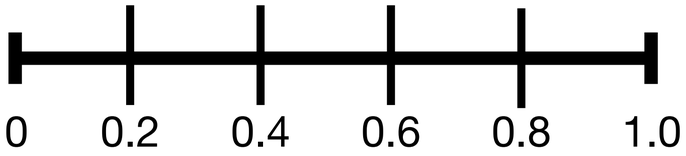
Sampling Strategy for a New Test Set

Input: Selection frequencies from MTurk
(= fraction of workers selecting the image)

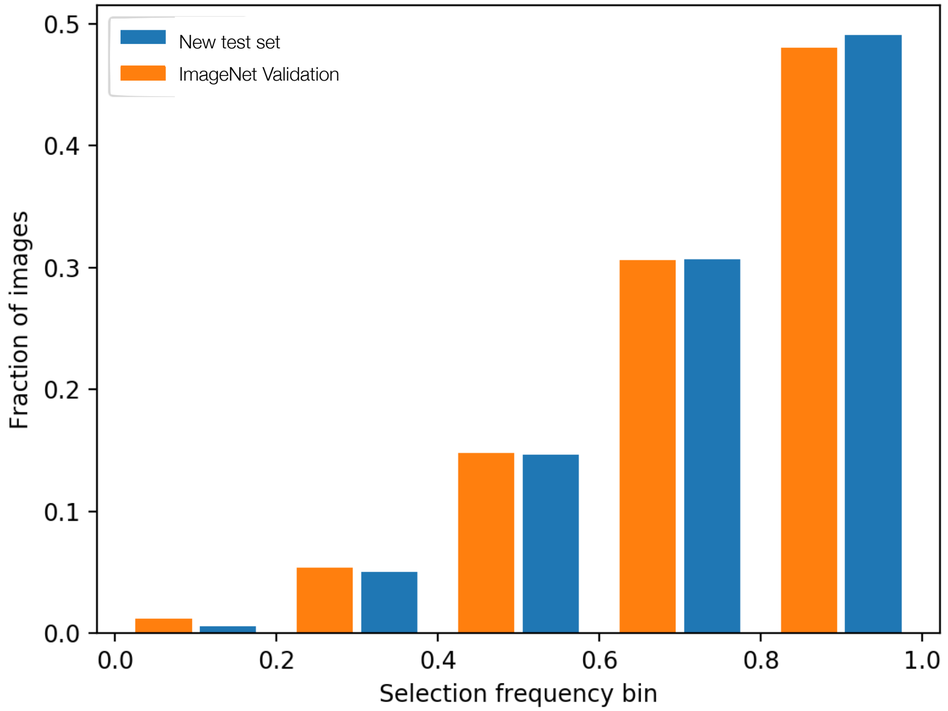
Output: representative & correct subset

Our approach:

1. Bin the existing validation images by selection frequency.



2. Sample images from our candidate pool to match the selection frequency distribution.



Three New Test Sets

ApproxCalibrated: Selection frequencies comparable to the original test set (**0.71**).

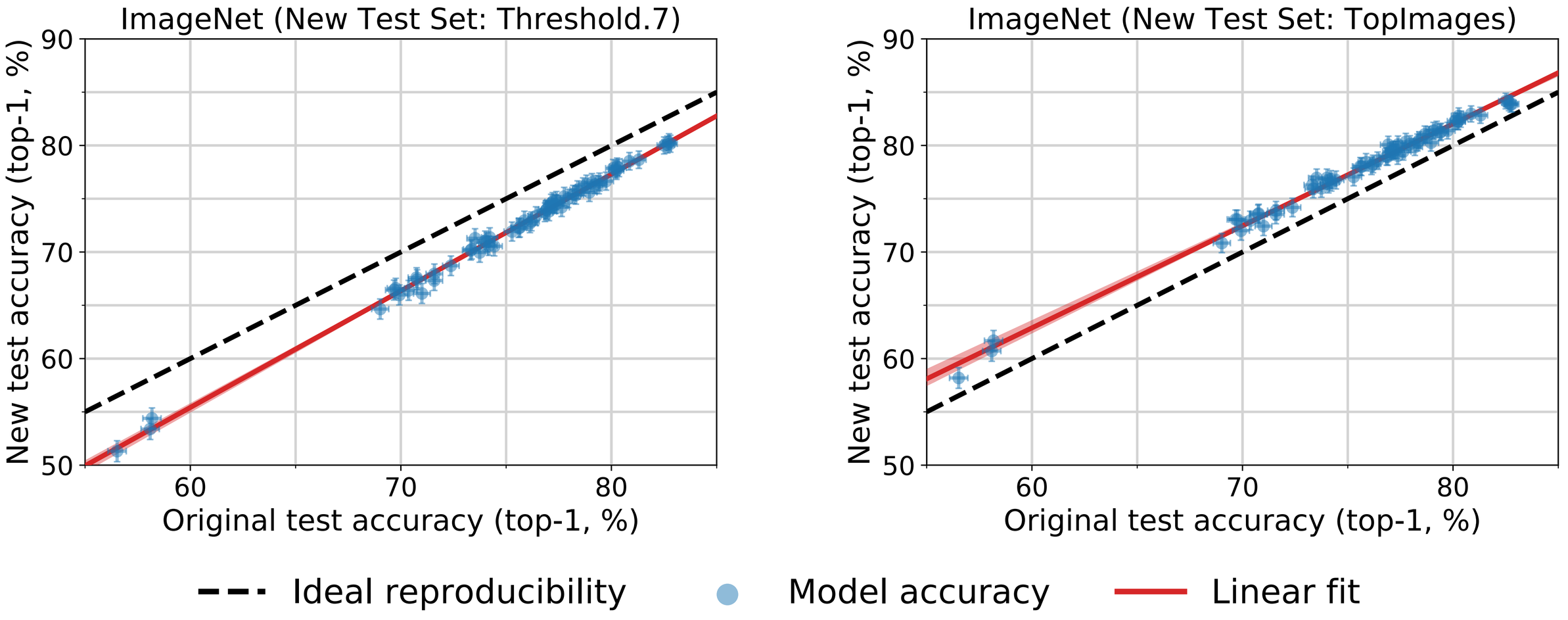
Easier: Different sampling strategy, higher selection frequencies.

Easiest: Highest selection frequencies in our candidate pool.

All correctly labeled!

Test Set	Average MTurk Selection Frequency	Average Top-1 Accuracy Change
ApproxCalibrated	**0.73**	- 12%

Selection frequencies have large impact on classification accuracies.



Relative ordering is stable, absolute accuracies are brittle.

Robustness on ImageNet

Lots of progress on ImageNet over the past 10 years, but models are still not robust.

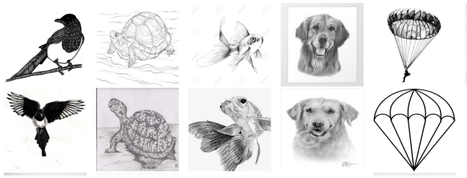
Evaluation: **new test sets**

ImageNetV2

[Recht, Roelofs, Schmidt, Shankar ’19]

ObjectNet

[Barbu, Mayo, Alverio, Luo, Wang, Gutfreund, Tenenbaum, Katz ’19]



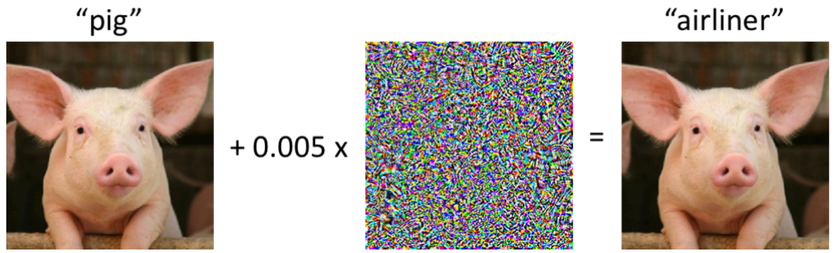
ImageNet-Sketch

[Wang, Ge, Lipton, Xing ’19]

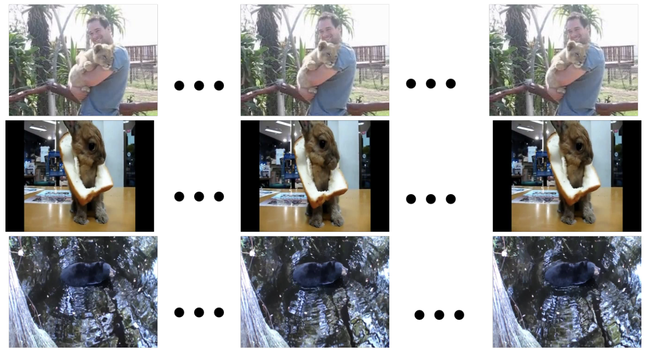
ImageNet-R

[Hendrycks, Basart, Mu, Kadavath, Wang, Dorundo, Desai, Zhu, Parajuli, Guo, Song, Steinhardt, Gilmer ’20]

Many different types of robustness



Adversarial examples



Video perturbations

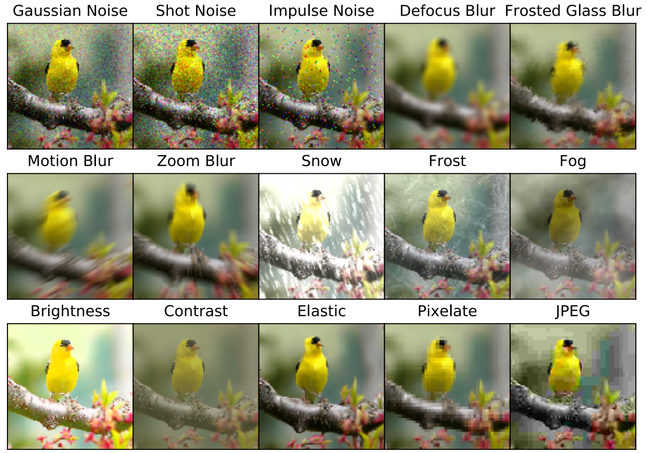
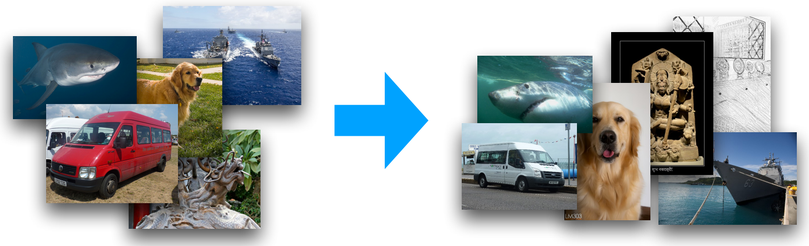
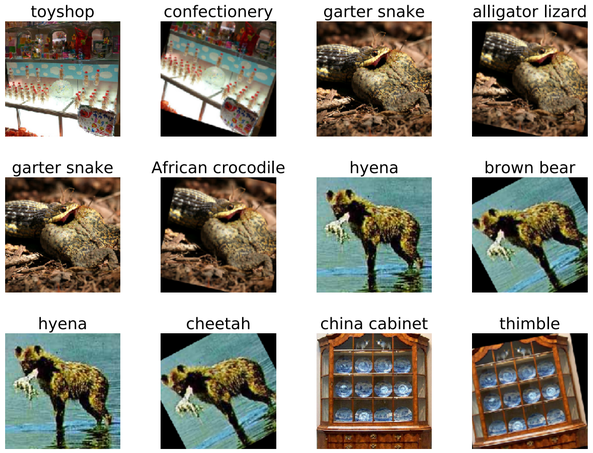


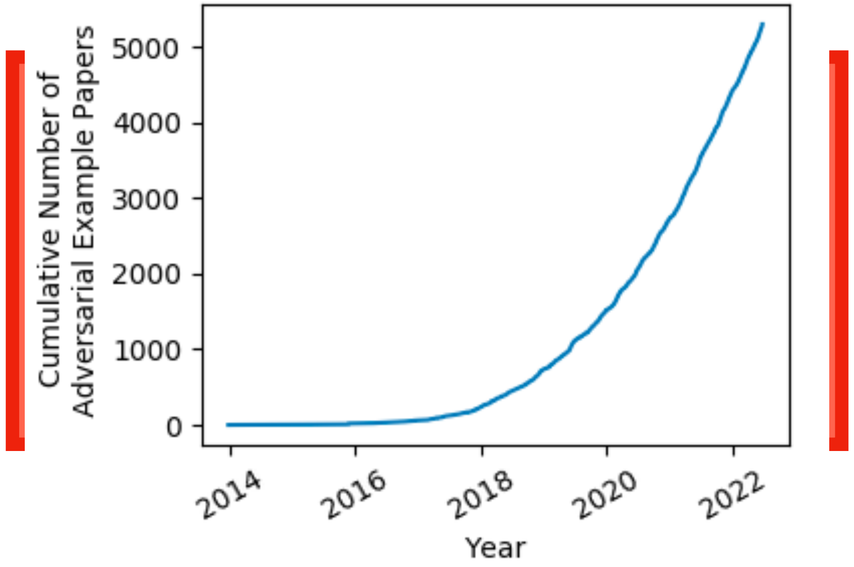
Image corruptions



Dataset shift



Geometric transformations



Measuring Robustness to Natural Distribution Shifts in Image Classification

Rohan Taori
UC Berkeley

Achal Dave
CMU

Vaishaal Shankar
UC Berkeley

Nicholas Carlini
Google Brain

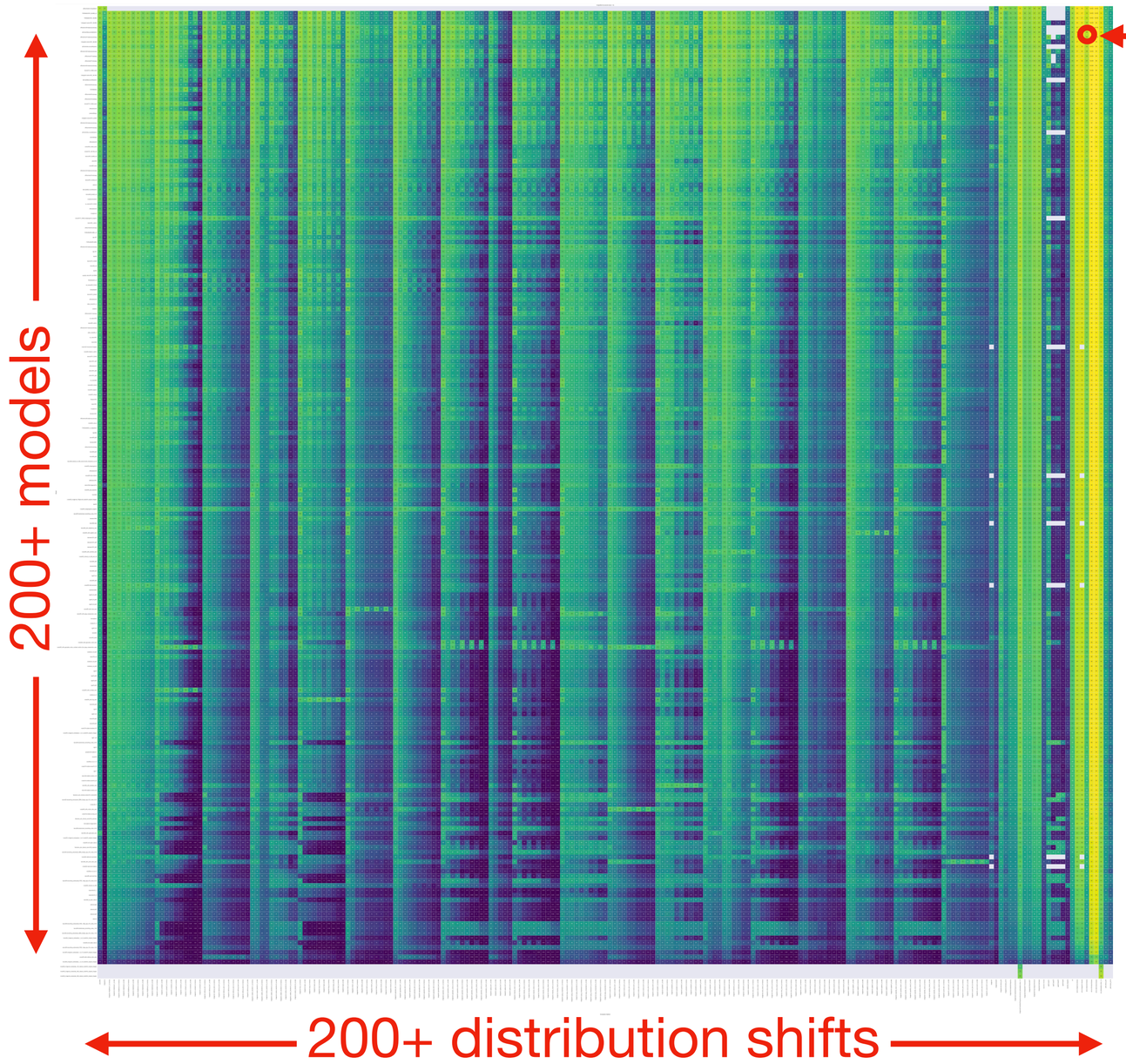
Benjamin Recht
UC Berkeley

Ludwig Schmidt
UC Berkeley

Abstract

We study how robust current ImageNet models are to distribution shifts arising from natural variations in datasets. Most research on robustness focuses on synthetic image perturbations (noise, simulated weather artifacts, adversarial examples, etc.), which leaves open how robustness on synthetic distribution shift relates to distribution shift arising in real data. Informed by an evaluation of 204 ImageNet models in 213 different test conditions, we find that there is often little to no transfer of robustness from current synthetic to natural distribution shift. Moreover, most current techniques provide no robustness to the natural distribution shifts in our testbed. The main exception is training on larger and more diverse datasets, which in multiple cases increases robustness, but is still far from closing the performance gaps. Our results indicate that distribution shifts arising in real data are currently an open research problem. We provide our testbed and data as a resource for future work at https://modestyachts.github.io/imagenet-testbed/.

Our approach: evaluate everything



1 cell = 1 model evaluation on 1 dataset (total 10^9 image evaluations).

Models:

- "**Standard**" models (focus on ImgNet acc.)
- **Robust** models (adversarially robust models, models with special data augmentation, etc.)
- Models trained on **more data**

Distribution shifts

- ImageNet-V2
- ObjectNet
- ImageNet-R
- ImageNet-Sketch
- ImageNet-A
- ImageNetVid-Robust
- Adversarial attacks (L_p-norms)
- Image corruptions
- …

Quantifying Robustness

Often in-distribution (“standard”) accuracy acts as a **confounder**.

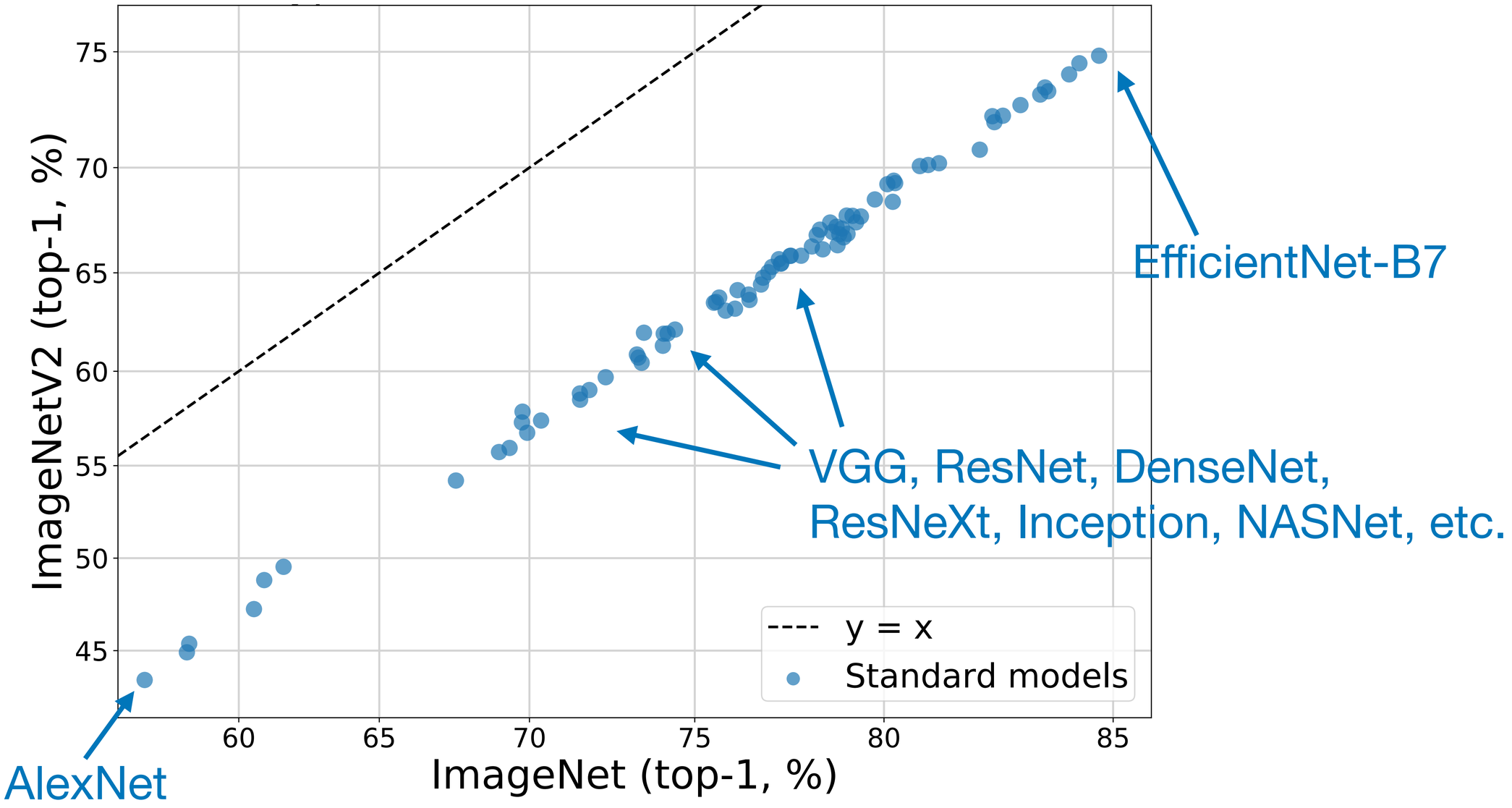
	In-distribution (Source) Accuracy	Out-of-distribution (Target) Accuracy
Model A	80%	75%
Model B	90%	77%

Quantifying Robustness

Often in-distribution (“standard”) accuracy acts as a **confounder**.

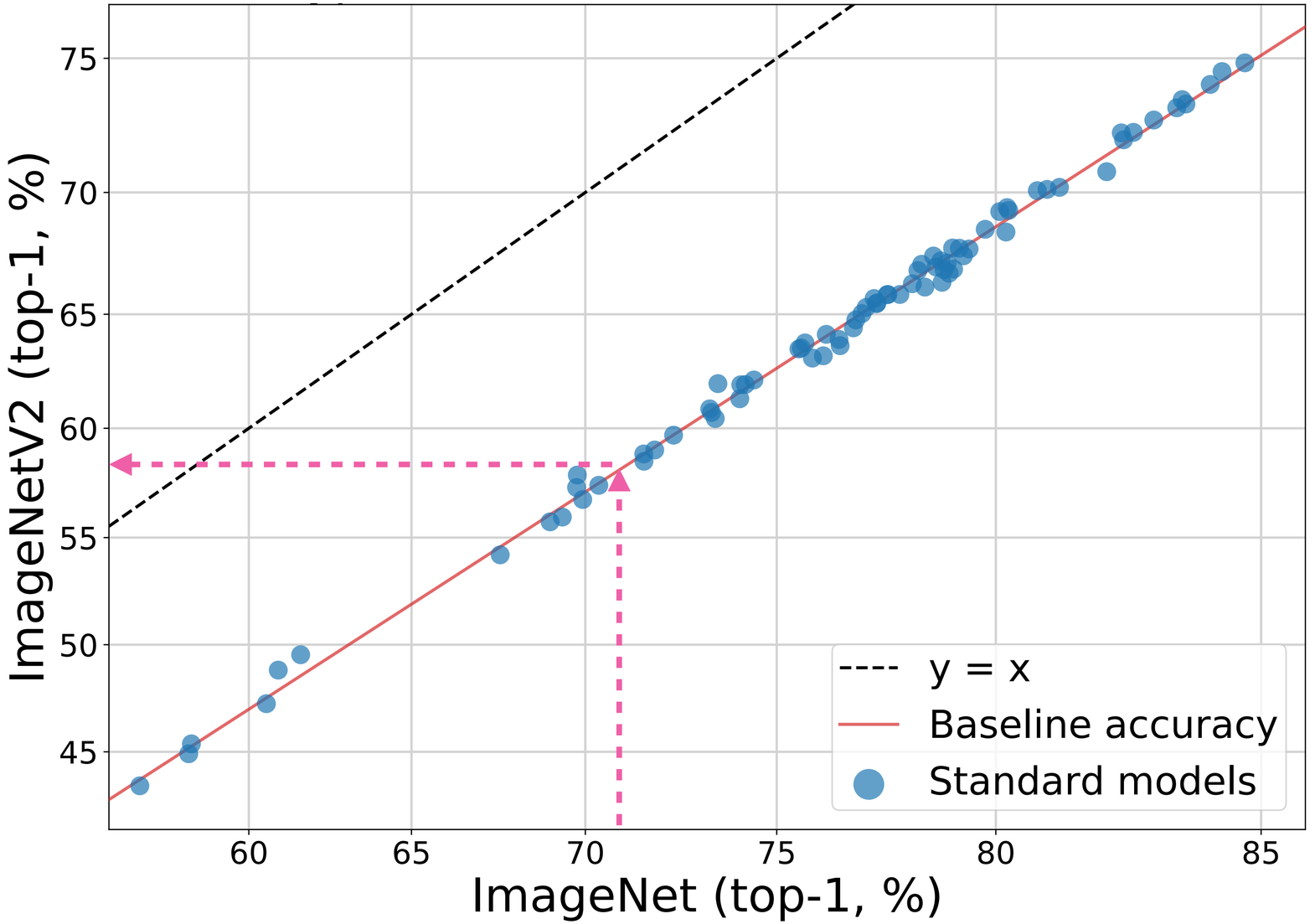
	In-distribution (Source) Accuracy	Out-of-distribution (Target) Accuracy	Accuracy Drop
Model A	80%	75%	5%
Model B	90%	77%	13%

How do we compare models with different in-distribution accuracy?



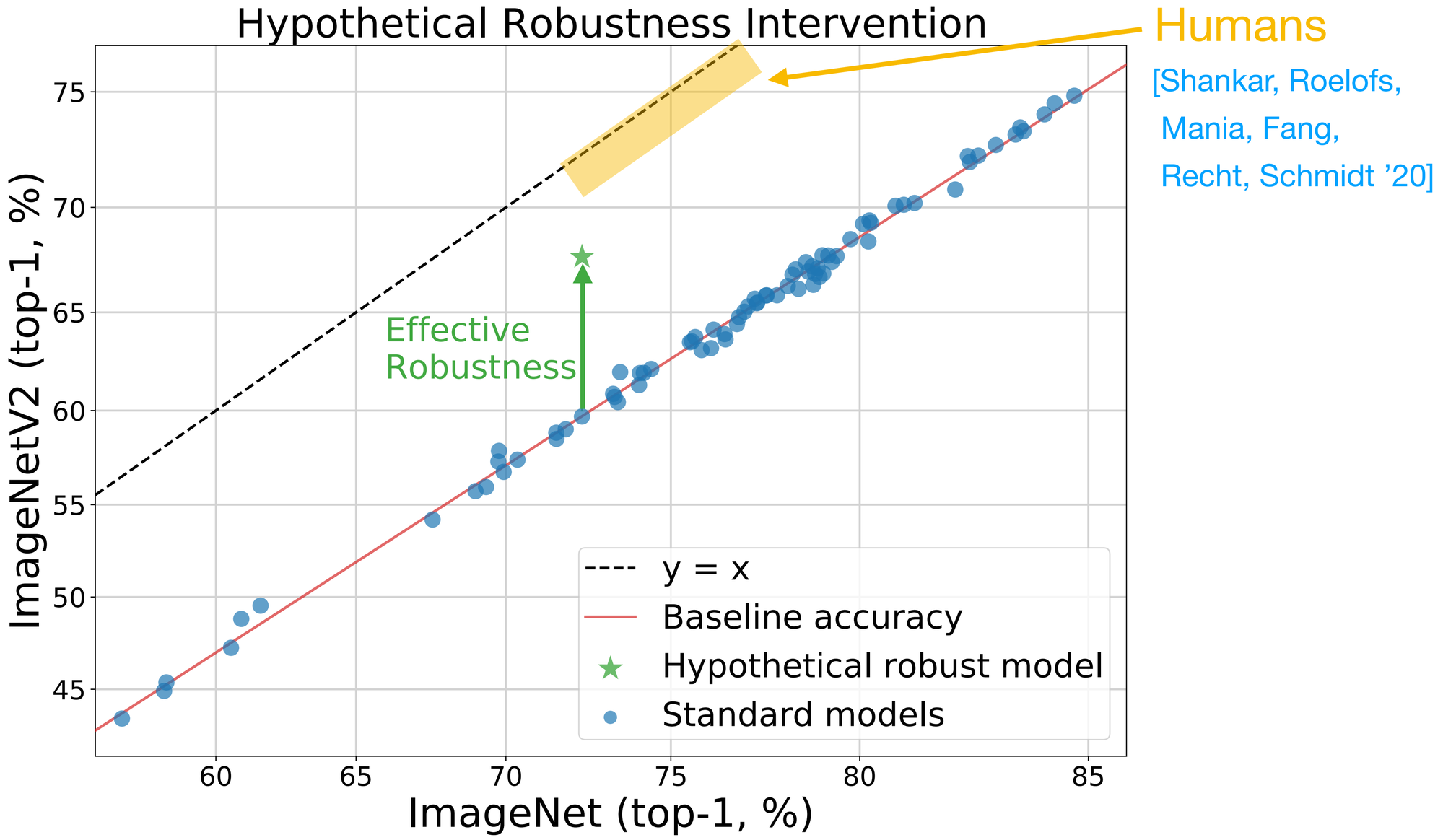
[Taori, Dave, Shankar, Carlini, Recht, Schmidt ’20]

Expected out-of-distribution accuracy

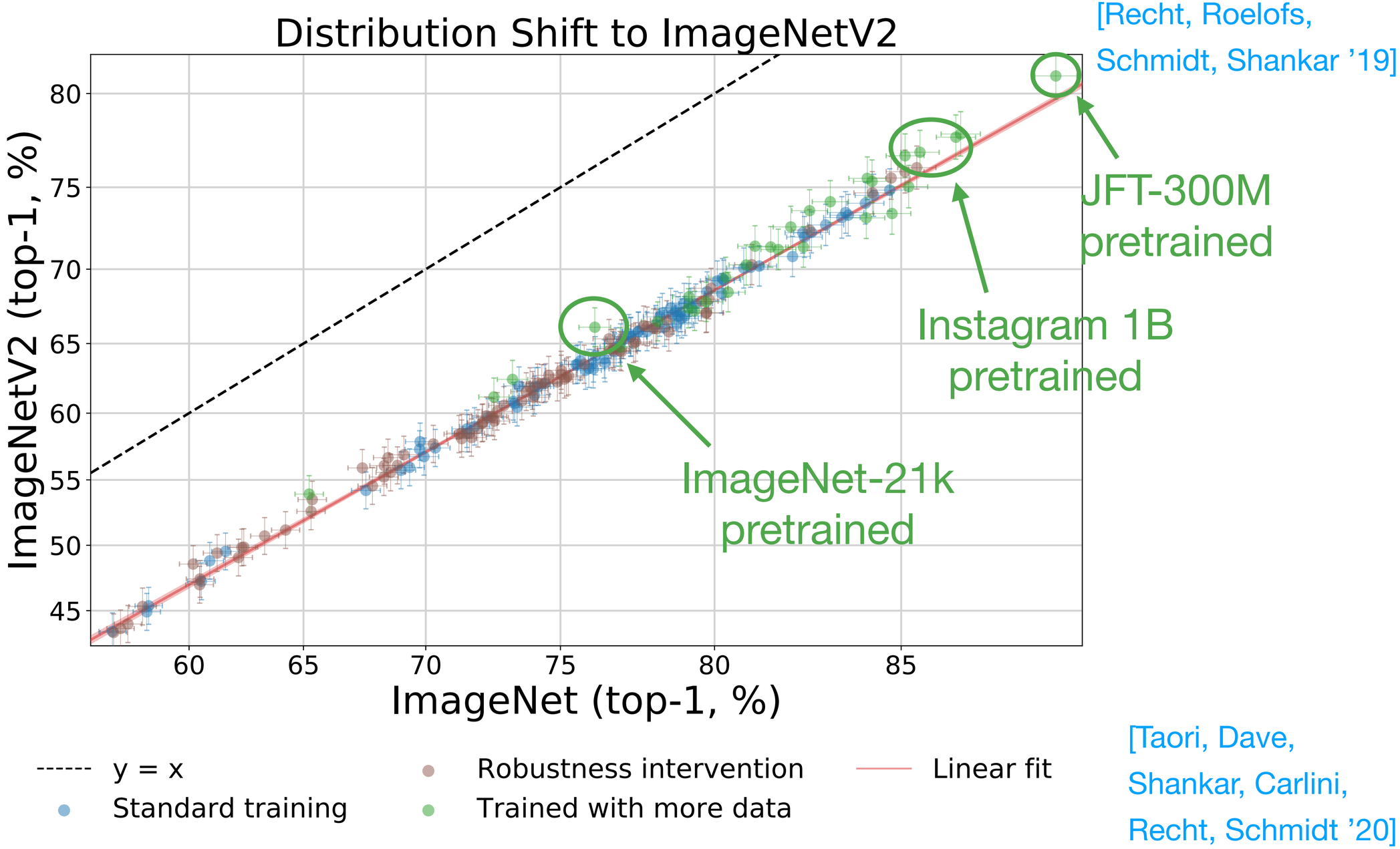


In-distribution accuracy

Baseline **out-of-distribution accuracy** from **in-distribution accuracy.**



Do current robustness interventions achieve effective robustness?



[Recht, Roelofs, Schmidt, Shankar '19]

[Taori, Dave, Shankar, Carlini, Recht, Schmidt '20]

Only training on (a lot) **more data** gives a small amount of effective robustness.