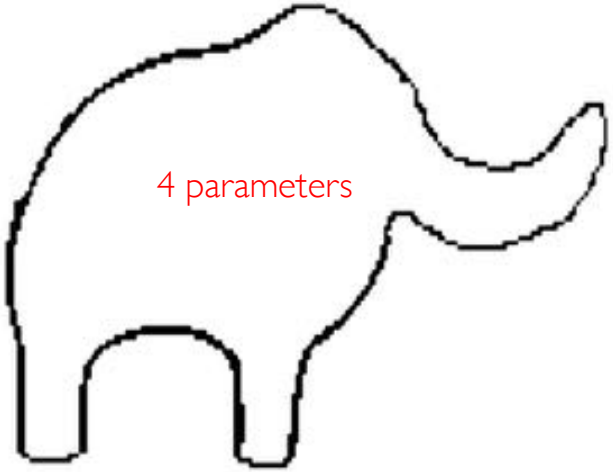
Deep neural networks have an inbuilt Occam's razor

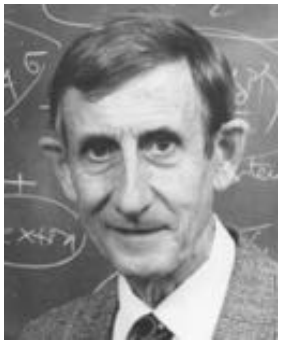
Ard Louis

Physicists are taught: more parameters than data points is bad

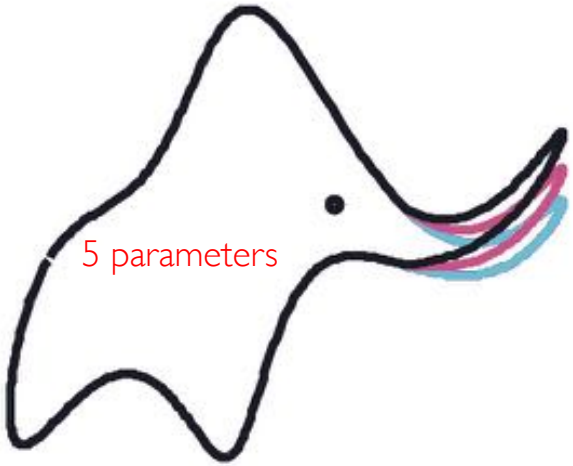
F. Dyson, *A meeting with Enrico Fermi*, Nature. **427**, 287 (2004)



Enrico Fermi
1901-1954



Freeman Dyson
1923-2020

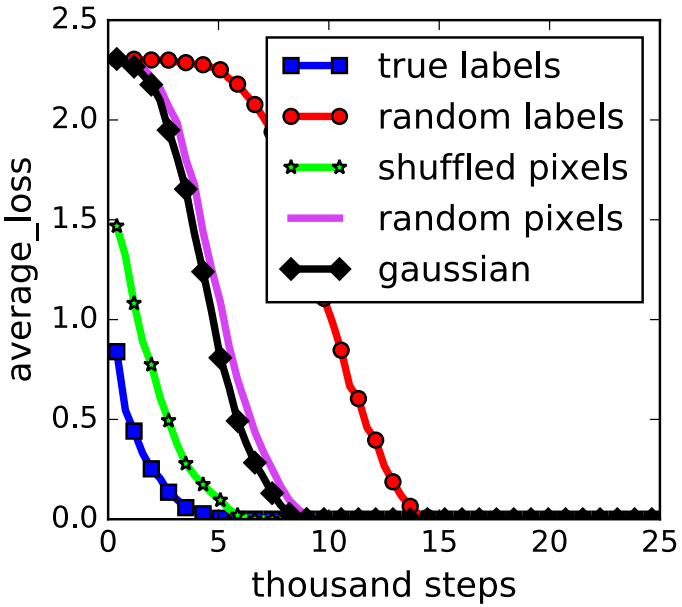


John von Neumann
1903-1957

With four parameters I can fit an elephant, and with five I can make him wiggle his trunk
-- John von Neuman (according to Fermi)

Drawing an elephant with four complex parameters
Jürgen Mayer; Khaled Khairy; Jonathon Howard; American Journal of Physics 78, 648-649 (2010)

Expressivity: DNNs can fit randomized data with zero error

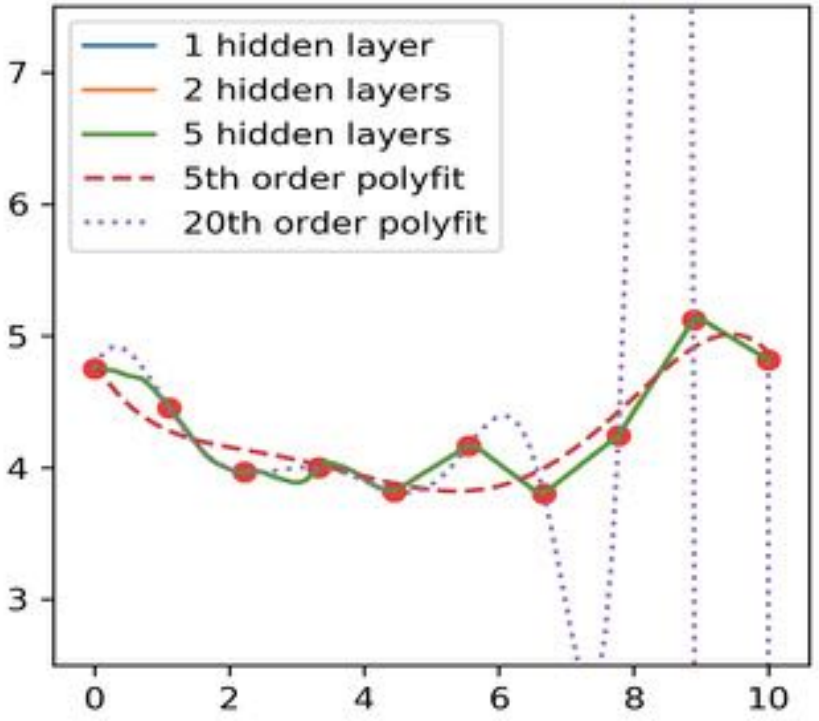


Randomize labels on CIFAR-10

Understanding deep learning requires rethinking generalization, C. Zhang et al, arXiv:1611.03530
(super influential: cited by 2329 –Dec 2020)

DNNs can fit random data with zero error they are highly expressive.
Why don't they memorize data?

Deep neural networks (DNNs) are heavily overparameterized



polynomial fit : $y(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \ldots a_nx^n$

compared to

simple DNNs (FCN with layer width of 1000 units)

CENTRAL THEORETICAL CONUNDRUM of DNNs: Why do they generalise so well?

1) DNNs are used in the over-parameterised regime with many more parameters than data points.
2) DNNs are highly expressive (there is a universal approximation theorem (Cybenko, Hornik etc..)
3) Classical learning theory, based on model capacity, predicts poor generalisation. (bias-variance tradeoff)

Model problem: Supervised learning of a Boolean function with DNNs

Doctor's decision table for COVID-19

Boolean function

Send to hospital?	Fever?	Cough?	Lost sense of smell?	Over 50?	Heart problem?	Obese?	Diabetes?
1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1
0	1	1	1	0	0	0	0
1	1	1	1	1	0	1	1
0	1	1	0	1	0	1	1

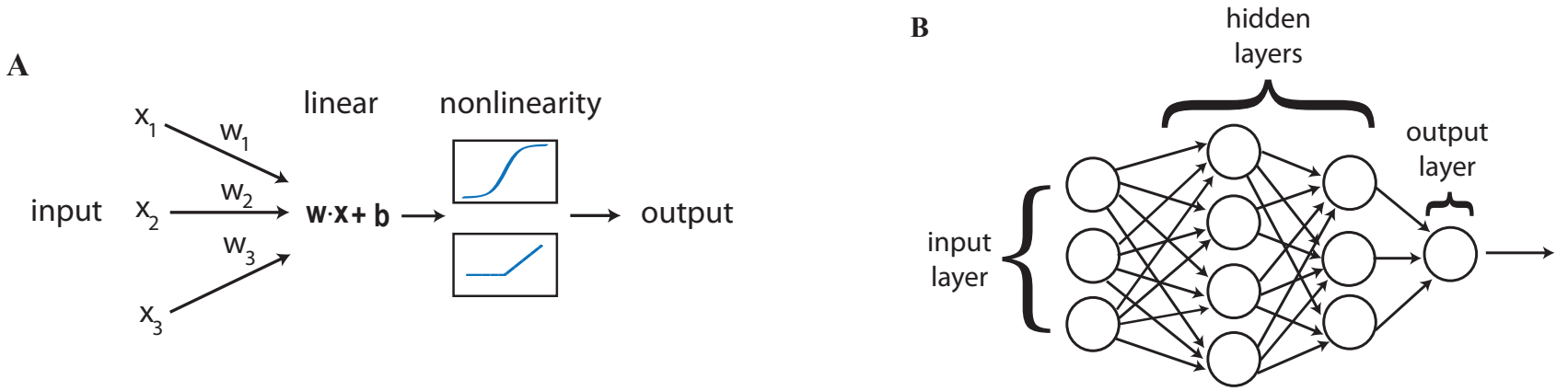
Given some examples, can we learn the rest of the function?

A function maps all possible answers to outputs.

n questions; 2^n possible answers; 2^{2^n} possible Boolean functions

For n=7 $2^7 = 128$ answers; $2^{128} = 3.4 \times 10^{38}$ possible functions

Parameter-function map



Let the space of functions that the model can express be $\mathcal{F}$. If the model has p real valued parameters, taking values within a set $\Theta \subseteq \mathbb{R}^p$,

the parameter-function map, $\mathcal{M}$, is defined as:

$$\mathcal{M} : \Theta \to \mathcal{F}$$
$$\theta \mapsto f_\theta$$

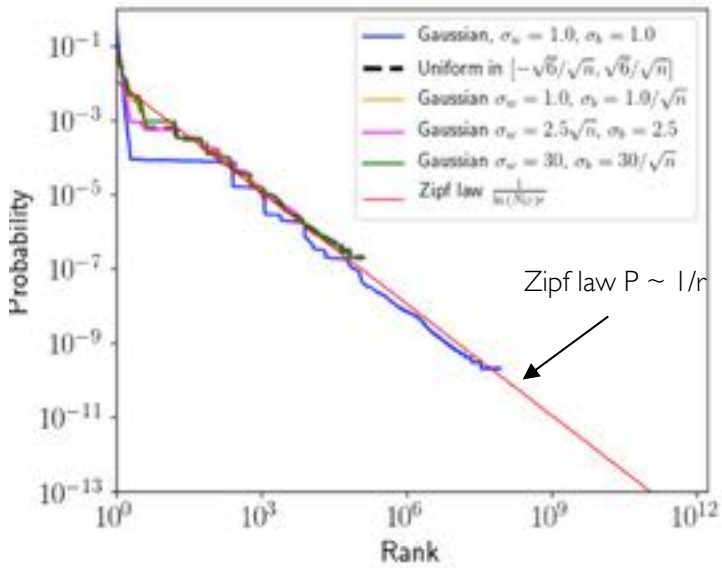
where f_θ is the function implemented by the model with choice of parameter vector θ.

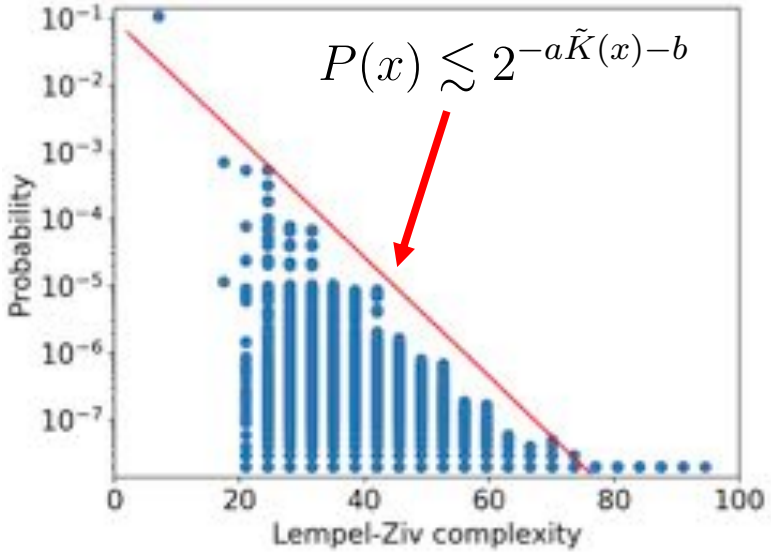
G. Valle-Perez, C. Camargo and A.A. Louis, arxiv:1805.08522 – ICLR 2019

Simplicity bias in the parameter-function map

Prior P(f): upon randomly sampling parameters, how likely to find Boolean function f?

Simple functions exponentially more likely to occur





10^8 samples of parameters (7,40,40,1) DNN (FCN) with ReLU.

Boolean functions for n=7. $2^7 = 128$ possible answers & $2^{128} \backsimeq 3.4 \times 10^{38}$ possible functions.

"Entropy" of simpler functions is larger than that of complex functions.

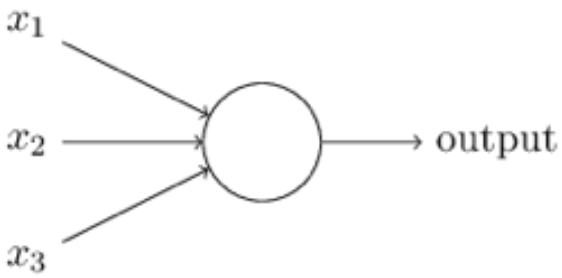
Boolean system is a key simplified model, akin to the Ising model is in physics.

Guillermo Valle Perez

G. Valle Perez, C. Camargo and A.A. Louis, arxiv:1805.08522 – ICLR 2019

Proving simplicity bias in the parameter-function map

P(f): If we randomly sample parameters θ, how likely are we to produce a particular function f?



Chris Mingard

Theorem 4.1. *For a perceptron f_θ with $b = 0$ and weights w sampled from a distribution which is symmetric under reflections along the coordinate axes, the probability measure $P(\theta : \mathcal{T}(f_\theta) = t)$ is given by*

$$P(\theta : \mathcal{T}(f_\theta) = t) = \begin{cases} 2^{-n} & \text{if } 0 \leq t < 2^n \\ 0 & \text{otherwise} \end{cases}.$$

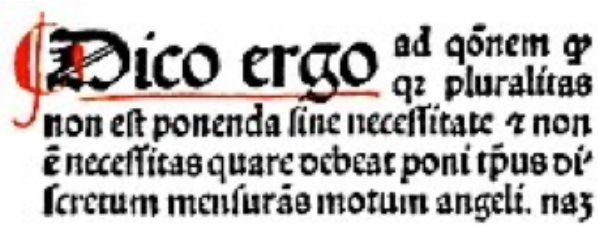
We can also prove theorems that bias towards simple function gets stronger with more layers!

Neural networks are a priori biased towards Boolean functions with low entropy, Chris Mingard, Joar Skalse, Guillermo Valle-Pérez, David Martínez-Rubio, Vladimir Mikulik, Ard A. Louis arxiv:1909.11522

Ockham's Razor and DNNs

Entities are not to be multiplied without necessity"
Ockham, according John Punch's 1639 commentary on Duns Scotus.

What Ockham actually said:


Dico ergo ad qōnem q
qz pluralitas
non eſt ponenda ſine neceſſitate ⁊ non
ē neceſſitas quare debeat poni tp̄us dī
ſcretum menſurās motum angeli. naz

Pluralitas non est ponenda sine necessitate"
"Plurality is not to be posited without necessity"

William of Ockham
1287-1347

-possibly at Merton?

Modern approaches (the rabbit hole is deep …)

Bayes (e.g. David MacKay "*Information Theory, Inference, and Learning Algorithms*", ch 28)

AIT (e.g. Solomonoff, Hutter etc.. (AIT), but see Tom Sterkenberg for a critique)

Philosophers disagree …..Aristotle → Elliot Sober

Is this simplicity bias more universal ?

Why do DNNs exhibit an inbuilt Occam's razor?

(why the simplicity bias?)

MONKEY INTUITION:

What is the probability that a monkey types out M digits of π on an N key typewriter ?

$$P(X) = (1/N)^{(M+1)}$$

3.14159265358979323846264338327950288419716939
93751058209749445923078164062862089986280348253421170679821480865132823066470938446095505822317253594081284811174502841027019385211055596446229489549303819644288109756659334461284756482337867831652712019091456485669234603486104543266482133936072602491412737245870066063155881748815209209628292540917153643678925903600113305305488204665213841469519415116094330572703657595919530921861173819326117931051185480744623799627495673518857527248912279381830119491298336733624406566430860213949463952247371907021798609437027705392171762931767523846748184676694051320005681271452635608277857713427577896091736371787214684409012249534301465495853710507922796892589235420199561121290219608640344181598136297747713099605187072113499999983729780499510597317328160963185950244594553469083026425223082533446850352619311881710100031378387528865875332083814206171776691473035982534904287554687311595628638823537875937519577818577805321712268066130019278766111959092164201989380952572010654858632788659361533818279682303019520353018529689957736225994138912497217752834791315155748572424541506959508295331168617278558890750983817546374649393192550604009277016711390098488240128583616035637076601047101819429555961989467678374494482553797747268471040475346462080466842590694912933136770289891521047521620569660240580381501935112533824300355876402474964732639141992726042699227967823547816360093417216412199245863150302861829745557067498385054945885869269956909272107975093029553211653449872027559602364806654991198818347977535663698074265425278625518184175746728909777727938000816470600161452491921732172147723501414419735685481613611573525521334757418494684385233239073941433345477624168625189835694855620992192221842725502542568876717904946016534668049886272327917860857843838279679766814541009538837863609506800642251252051173929848960841284886269456042419652850222106611863067442786220391949450471237137869609563643719172874677646575739624138908658326459958133904780275901

But what if the monkey types into C ?

$$P(M) \lesssim (1/N)^{133}$$

133 character (obfuscated) C code to calculate first 15,000 digits of π

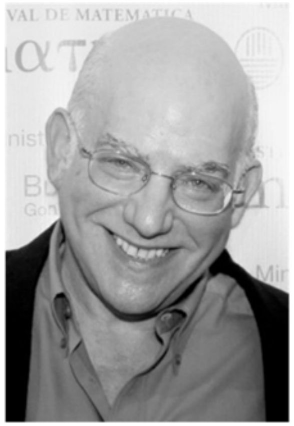
```
a[52514],b,c=52514,d,e,f=1e4,g,h;
main(){for(;b=c-=14;h=printf("%04d", e+d/f))
for(e=d%=f;g=--b*2;d/=g)d=d*b+f*(h?a[b]:f/5),a[b]=d%--g;}
```

$$\pi = \sum_{i=0}^{\infty} \frac{(i!)^2 2^{i+1}}{(2i+1)!}$$

C program due to Dik Winter and Achim Flammenkamp (See Unbounded Spigot Algorithms for the Digits of Pi, by Jeremy Gibbons (Oxford CS), Math. Monthly, April 2006, pages 318-328.)

Formalising the Monkey Intuition AIT: Kolmogorov Complexity

AIT = Algorithmic Information Theory



A.N. Kolgomorov
1903-1987

G.J. Chaitin
1947--

Kolmogorov/Chaitin complexity K(X) is the length in bits of the shortest program on a UTM that generates X

K is universal, (not UTM dependent) because you can always write a compiler => O(1) terms.

$K\_U(X) = K\_W(X) + O(1) \approx K(X)$ asymptotically

K is not computable due to Halting problem.

simple

0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101

complex

0111010100110010111101111011010100001000101110101010011010111110111010010100011101110110110111010

Warning: you don't know for sure that it is complex, t could be encoding π= 3.14159265358979323846 2 …..

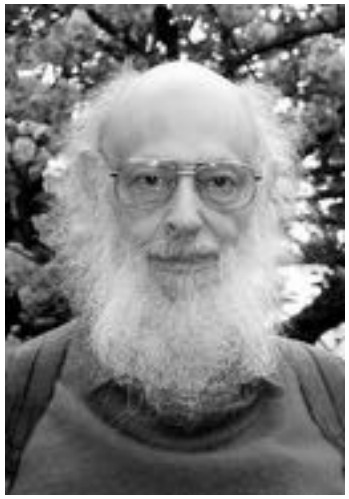
new intuitions

-- A random number is one for which $K(X) \gtrsim |X|$

-- The complexity of a set can be << than complexity of elements of the set

Formalising the Monkey Intuition using AIT: Algorithmic Probability

Algorithmic Probability P(x) = probability a random program on a (prefix) UTM generates x



R. Solomonoff
1926-2009

$$P_U(X) = \sum_{l:U(l)=X} 2^{-l} = 2^{-K(X)} + ...$$

First term is the biggest one

Sum all binary codes that generate X on a prefix machine

Intuitively: simpler (small K(X)) outputs are much more likely to appear

It seems to me that the most important discovery since Gödel was the discovery by Chaitin, Solomonoff and Kolmogorov of the concept called Algorithmic Probability,. Everybody should learn all about that and spend the rest of their lives working on it.
Marvin Minsky (2014)
https://www.youtube.com/watch?v=DfY-DRsE86s&feature=youtu.be&t=1h30m02s

Solomonoff, R., "A Preliminary Report on a General Theory of Inductive Inference", Report V-131, Zator Co., Cambridge, Ma. Feb 4, 1960, revision, Nov., 1960.

Formalising the Monkey Intuition using AIT: Levin's Coding Theorem

L. Levin, 1948 --

We should teach this much more widely!

$$2^{-K(x)} \leq P(x) \leq 2^{-K(x)+O(1)}$$

Intuitively: simpler (small K(X)) outputs are much more likely to appear

Serious problems for applying coding theorem

1) Many systems of interest are not Universal Turing Machines
2) Kolmogorov complexity K(x) is formally incomputable
3) Only holds in in the asymptotic limit of large x…

L. A. Levin. Laws of information conservation (non-growth) and aspects of the foundation of probability theory. Problems of Information Transmission, 10:206–210, 1974.

Formalising the Monkey Intuition using AIT: a new coding theorem for non UTM maps

Proof sketch:

1) For simple maps **f,** with input size ***n*** we can calculate the whole set of input → output pairs at *O(1)* cost (complexity of a set << elements of set)
2) Encode this with a Shannon-Fano-Elias (SFE) code for which P(x) ~ ½^length
3) This procedure gives a bound on the Kolmogorov complexity, **given f and n:** K(x|f,n)

$$K(x|f,n) \leq l(E(x)) + O(1)$$

$$= \log_2 \left(\frac{1}{P(x)} \right) + O(1)$$

$$\Rightarrow \boxed{P(x) \leq 2^{-K(x|f,n)+O(1)}}$$

← NOTE: upper bound only!

K. Dingle, C. Camargo and A.AL, Nature Communications 9, 761 (2018)

Simplicity bias for computable input-output maps

Kamal Dingle

(2 Dphils of work)

$$P(x) \lesssim 2^{-a\tilde{K}(x)-b}$$

NOTE: upper bound only!

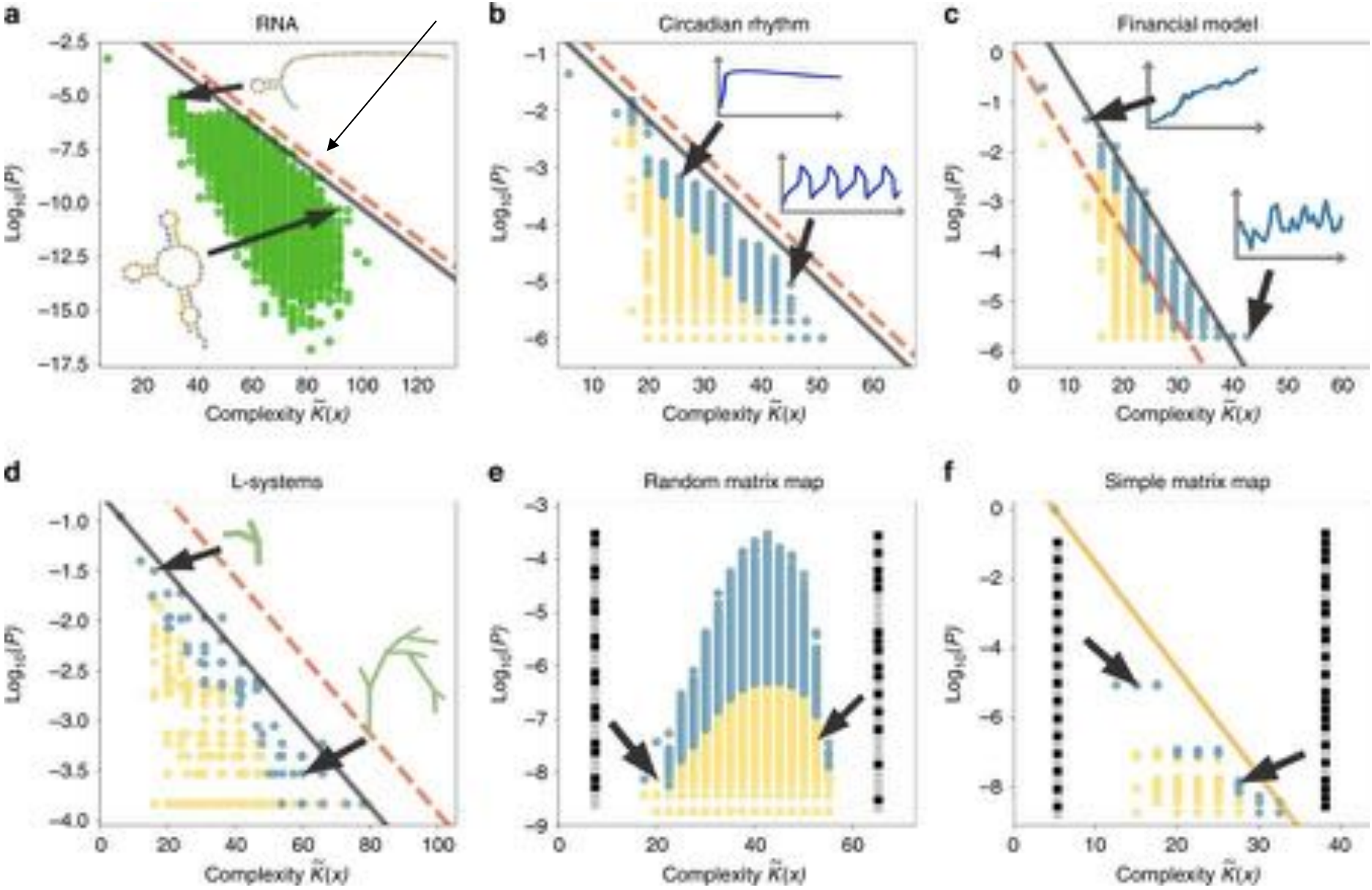
Chico Camargo

1) Computable input-output map f: I → O
2) Map **f** must be simple – e.g. K(f) grows slowly with system size – then K(x|f,n) ≈ K(x) + O(1)
3) K(x) is approximated, for example by Lempel Ziv compression or some other suitable measure.
4) Bound is tight for most inputs, but not most outputs.
5) Maps must be a) simple, b) redundant, c) non-linear, d) well-behaved (e.g. not a pseudorandom number generator) – many maps satisfy these conditions.
6) There is also a statistical lower bound.

K. Dingle, C. Camargo and A.AL, Nature Comm 9, 761 (2018); K. Dingle, G. Valle-Perez, AAL, Sci. Rep. 10, 4415 (2020)

Simpliciy bias works in many different maps

$P(x) \lesssim 2^{-a\tilde{K}(x)-b}$ = black line (red dashed with b=0)

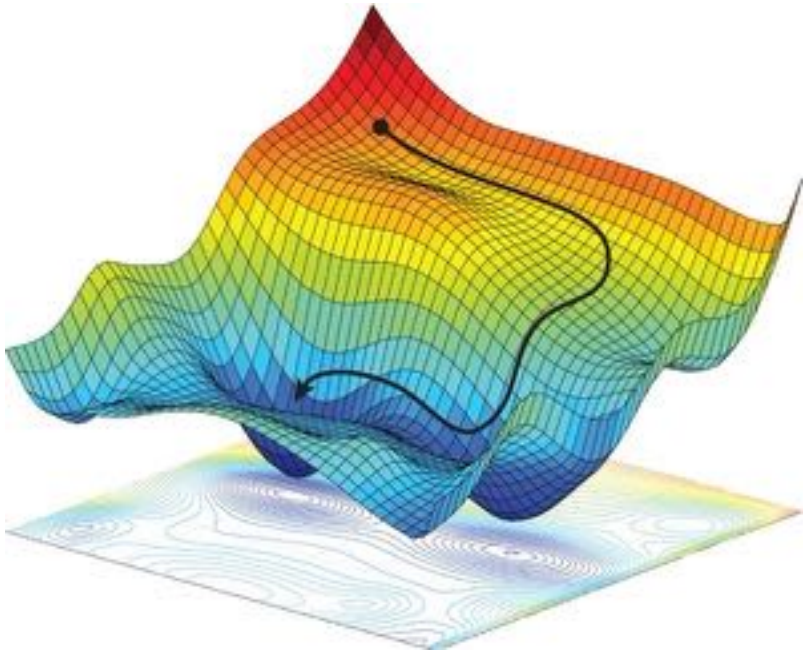


K. Dingle, C. Camargo and A.AL, Nature Communications 9, 761 (2018)

What about SGD?

Hold on: why should parameter function map predict DNN outcomes?

Hold on: why should parameter function map predict DNN outcomes?



DNNs are trained using Stochastic gradient descent (SGD) on a loss function.

~~Dominant hypothesis in the field is that SGD has special properties that enhance generalization~~

Is SGD a Bayesian sampler? Well, almost, Chris Mingard, Guillermo Valle-Pérez, Joar Skalse, Ard A. Louis, arxiv.org: 2006.15191

A function based picture

Definition 2.2 (Representation of Functions)**.** Consider a DNN $\mathcal{N}$, a training set $S = \{(x_i, y_i)\}_{i=1}^{m}$ and test set $E = \{(x'_i, y'_i)\}_{i=1}^{|E|}$. We *represent* the function $f(\mathbf{w})$ with parameters $\mathbf{w}$ associated with $\mathcal{N}$ as a string of length $(|S| + |E|)$, where the values are the labels $\hat{y}_i$ and $\hat{y}'$ that $\mathcal{N}$ produces on the concatenation of training inputs and testing inputs.

Example on 5 MNIST inputs:

f(**w**) = (5,0,4,1,9) (0 errors)

f(w) = (5,0,4,7,9) (1 error)

Bayesian function picture for supervised learning on S

Posterior for functions conditioned on training set S follows from Bayes rule

$$P(f|S) = \frac{P(S|f)P(f)}{P(S)},$$

Prior over functions $P(f)$

If we wish to infer (i.e. no noise) at some points, then we need a 0-1 likelihood on training data $S = \{(x_i, y_i)\}_{i=1}^m$

$$P(S|f) = \begin{cases} 1 \text{ if } \forall i, \quad f(x_i) = y_i \\ 0 \text{ otherwise .} \end{cases}$$

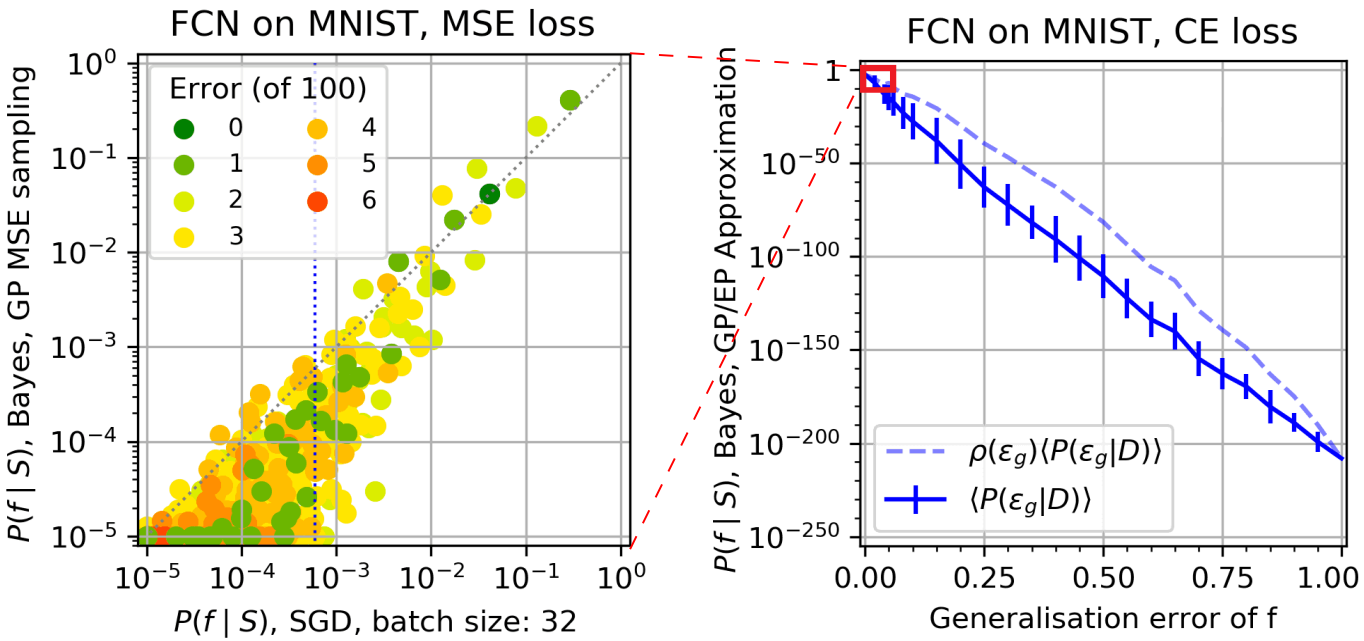
P(S) = marginal likelihood or evidence

Functions that fit S

$$P(S) = \sum_f P(S|f)P(f) = \sum_{f \in C(S)} P(f)$$

P(f|S) = P(f)/P(S) or 0, so bias in prior translates over to bias in posterior

SGD acts like a Bayesian optimiser ….



(a) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{SGD}}(f|S)$ (b) $P_{\mathrm{B}}(f|S)$ v.s. ϵ_G

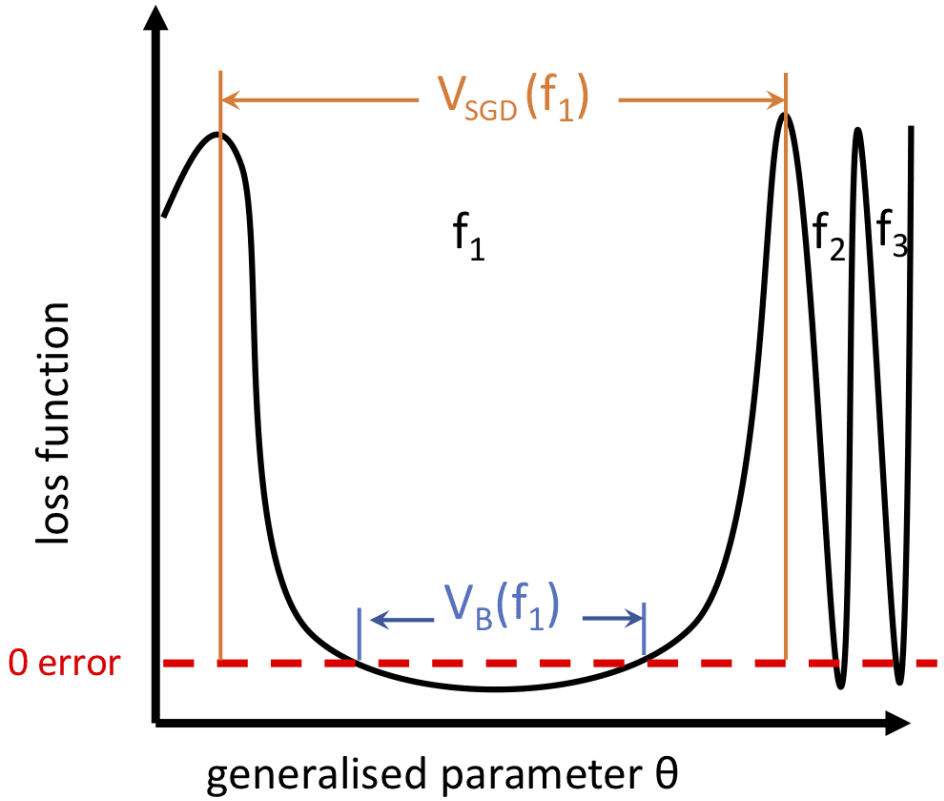
FCN on binarized MNIST – training set=10,000, test set=100 images $2^{100} = 10^{30}$ possible functions fit the test set.

We use Gaussian Processes (GP)s to calculate $P_B(f|S)$ –

Is SGD a Bayesian sampler? Well, almost, Chris Mingard, Guillermo Valle-Pérez, Joar Skalse, Ard A. Louis, arxiv.org: JMLR **22(79):1–64, 2021**

Problem: why should parameter function map predict outcomes?

Intuition: for very strong bias: Basin of attraction ~ Basin size (P(f))



Similar effect in evolutionary theory under strong bias:
The arrival of the frequent: how bias in genotype-phenotype maps can steer populations to local optima
Steffen Schaper and Ard A. Louis, PLoS ONE 9 (2): e86635 (2014)

Is SGD a Bayesian sampler? Well, almost, Chris Mingard, Guillermo Valle-Pérez, Joar Skalse, Ard A. Louis, arxiv.org: 2006.15191

Two kinds of questions about generalisation:

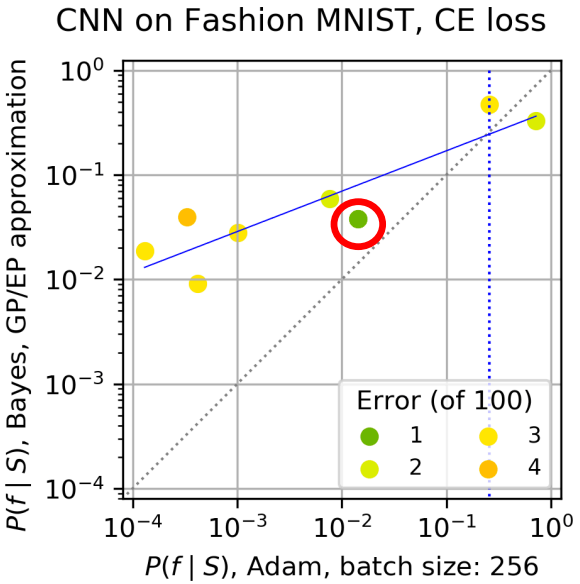
1) Why do DNNs generalise at all in the overparameterised regime?

Because the parameter-function map is highly biased towards simple solutions.

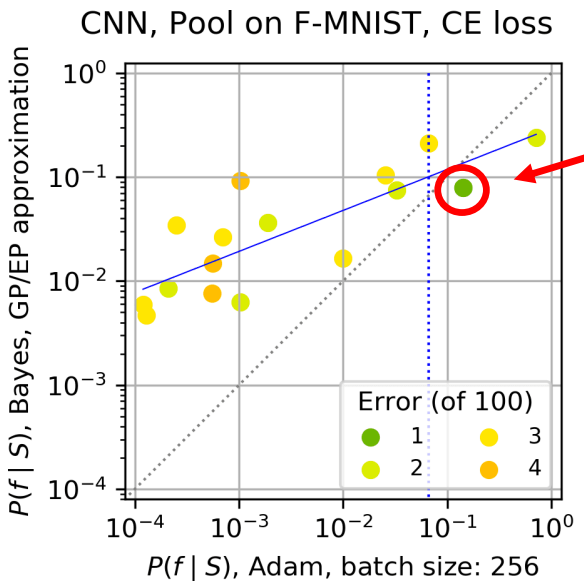
2) Given DNNs that generalise, can we further fine-tune the hyperparameters to improve generalisation? (engineers).

Is SGD a Bayesian sampler? Well, almost, C. Mingard, G.Valle-Pérez, J. Skalse, AAL, arxiv.org: 2006.15191

2nd order effects beyond simplicity bias: changing the network



(b) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{Adam}}(f|S)$



(c) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{Adam}}(f|S)$

With max-pooling probability of low error function increases

CNN on binarized Fashion-MNIST – training set=10,000, test set=100 images
$2^{100} = 10^{30}$ possible functions fit the test set.

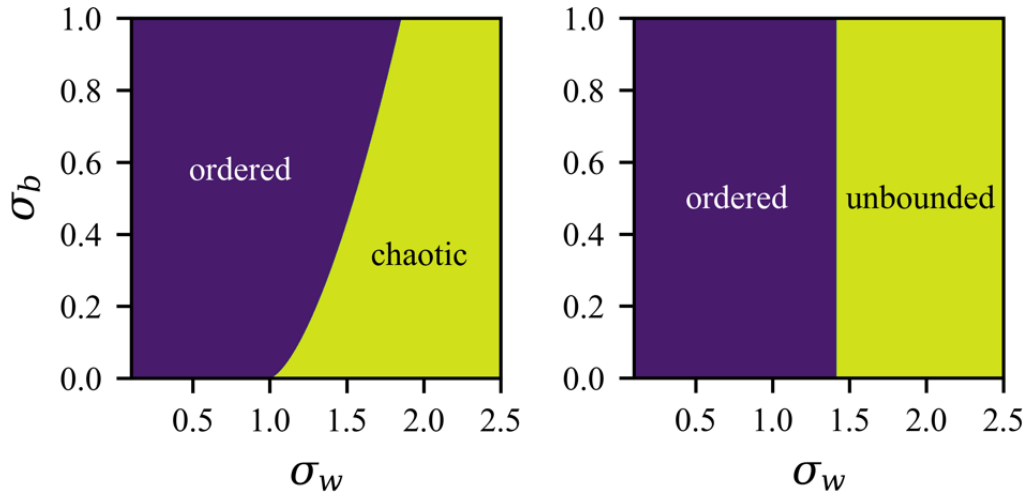
Similar results for CNN, LSTM, other data sets, etc….

Is SGD a Bayesian sampler? Well, almost, Chris Mingard, Guillermo Valle-Pérez, Joar Skalse, Ard A. Louis, arxiv.org: 2006.15191

Can we do any control experiments?

1)Can we break the simplicity bias?

DNNs can exhibit an order-to-chaos transition



(a) *Tanh* **(b)** *ReLU*

Figure 3: *Mean field phase diagrams for tanh and ReLU activation functions showing various phase regimes as a function of* σ_w *and* σ_b.

Chaotic regime for some activation functions (not ReLU!) – for wider initial parameters

Deep Information Propogation, S. S. Schoenholz et al. arXiv:1611.01232

Chaotic regime reduces bias in prior P(f)

FCN on Boolean system

J. **Empirical probability versus LZ complexity plots**

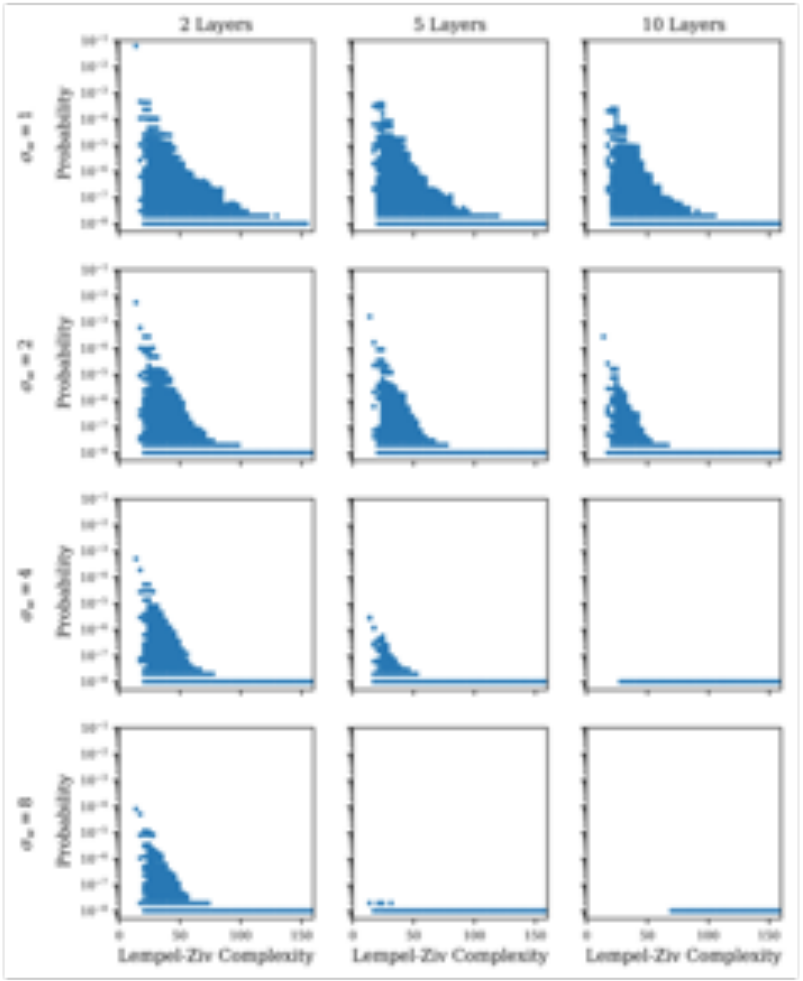


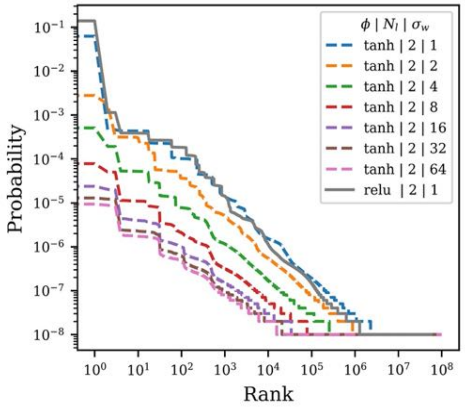
Figure 17: *Empirical probability of individual functions versus their LZ complexity for networks initialized with various σ_w and numbers of layers. Despite suffering from finite-size effects, points with a probability of 10^{-5} are not removed since in plots ($\sigma_w = 4, d = 10$) and ($\sigma_w = 8, d = 10$) only points of this type are found. Details are the same as Figure 6.*

More biased

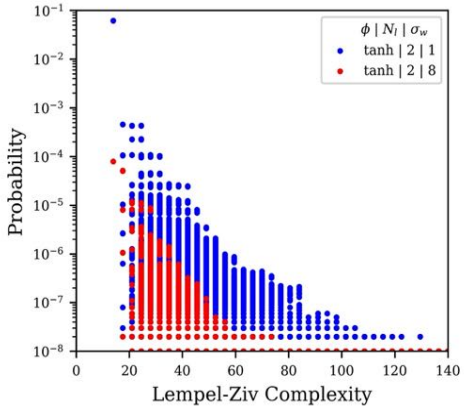
Less biased
No Occam

Greg Yang and Hadi Salman. A fine-grained spectral perspective on neural networks. arXiv preprint arXiv:1907.10599, 2019.

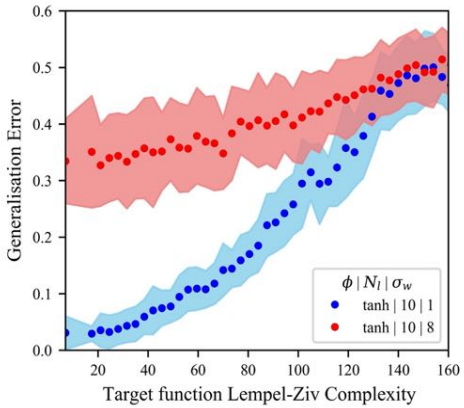
Chaotic regime changes the bias in prior P(f)



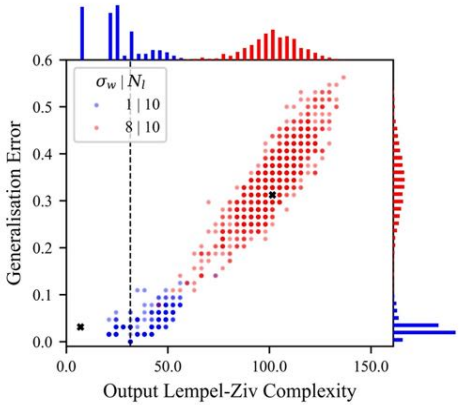
(a)



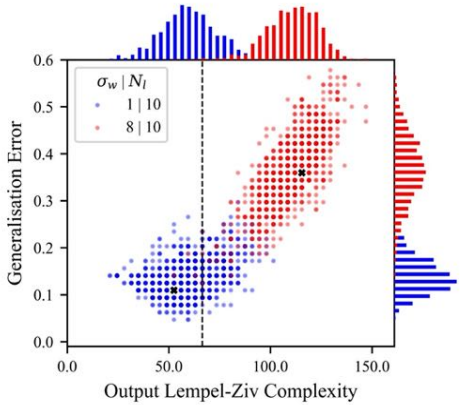
(b)



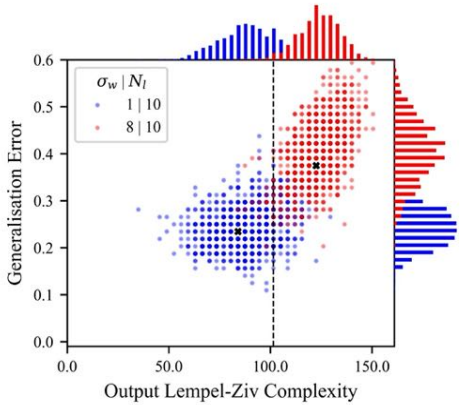
(c)



(d) Target function LZ = 31.5



(e) Target function LZ = 66.5



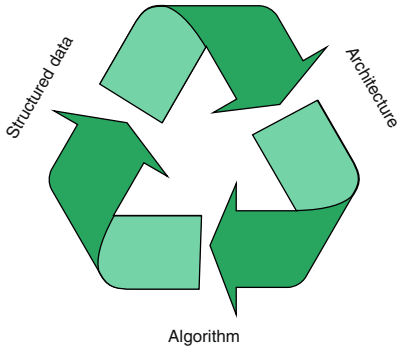
(f) Target function LZ = 101.5

Greg Yang and Hadi Salman. A fine-grained spectral perspective on neural networks. arXiv preprint arXiv:1907.10599, 2019.

Bayesian picture and the data

$$\langle P(f|\mathcal{S})\rangle_S = P(f)\left\langle \frac{P(\mathcal{S}_i|f)}{P(\mathcal{S}_i)} \right\rangle_{\mathcal{S}_i} \approx \frac{P(f)(1-\epsilon(f))^m}{\langle P(\mathcal{S})\rangle}$$

$$(1-\varepsilon(f))$$



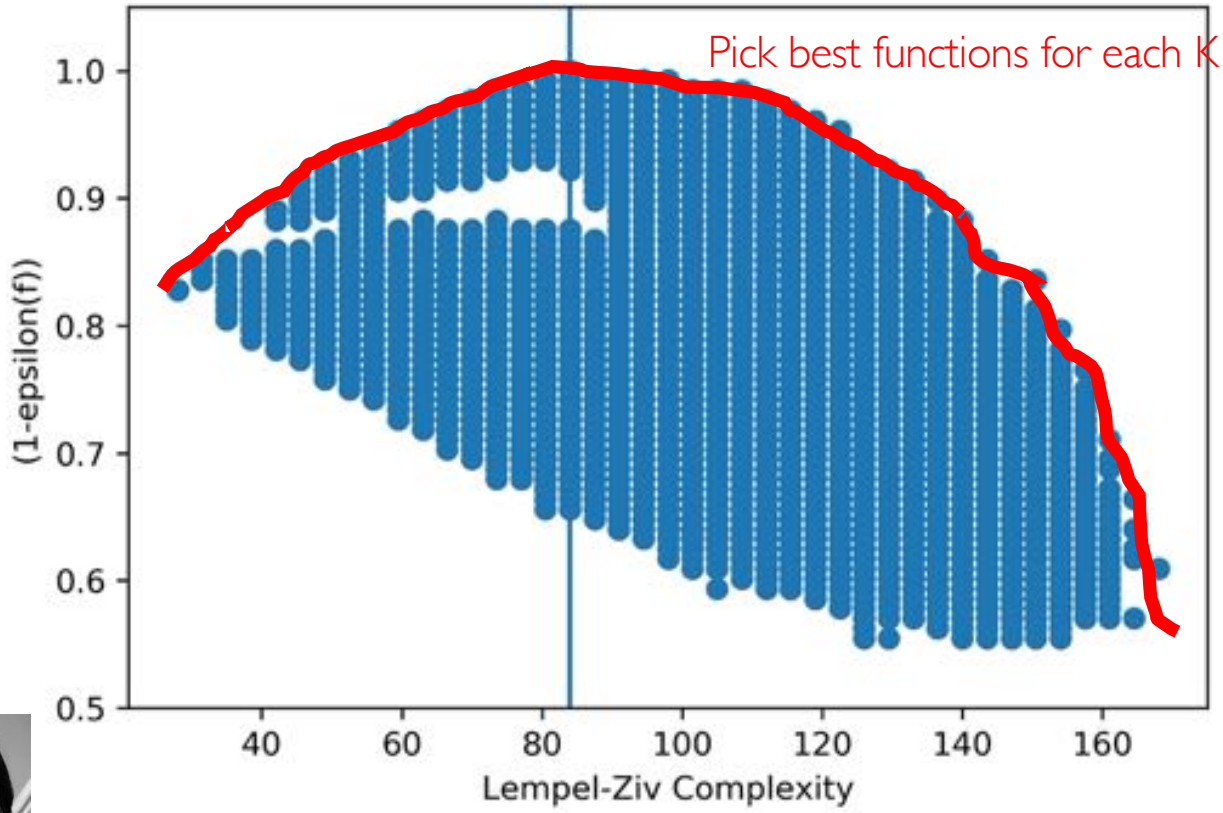
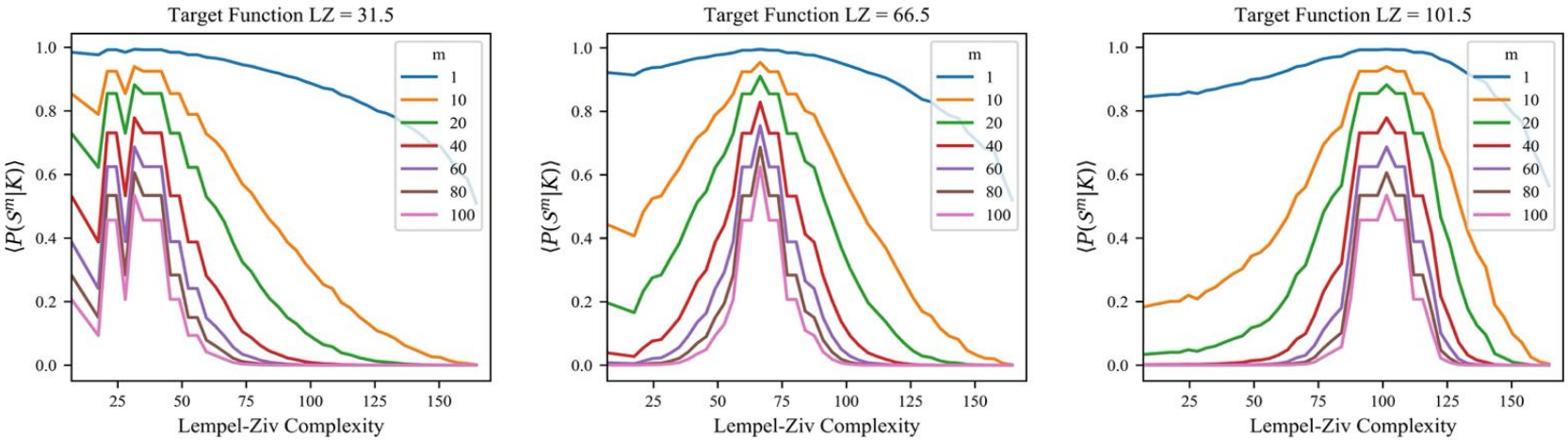


Fig. 1 | Interplay of key ingredients. Building theory of deep learning requires an understanding of the intrinsic interplay between the architecture of the neural network, the behaviour of the algorithm used for learning and the structure in the data.

Lenka Zdeborová. Nature Physics **16,** 602 (2020)

Bayesian picture and data

$$\langle P(f|\mathcal{S})\rangle_S = P(f)\left\langle \frac{P(\mathcal{S}_i|f)}{P(\mathcal{S}_i)}\right\rangle_{\mathcal{S}_i} \approx \frac{P(f)(1-\epsilon(f))^m}{\langle P(\mathcal{S})\rangle} =$$



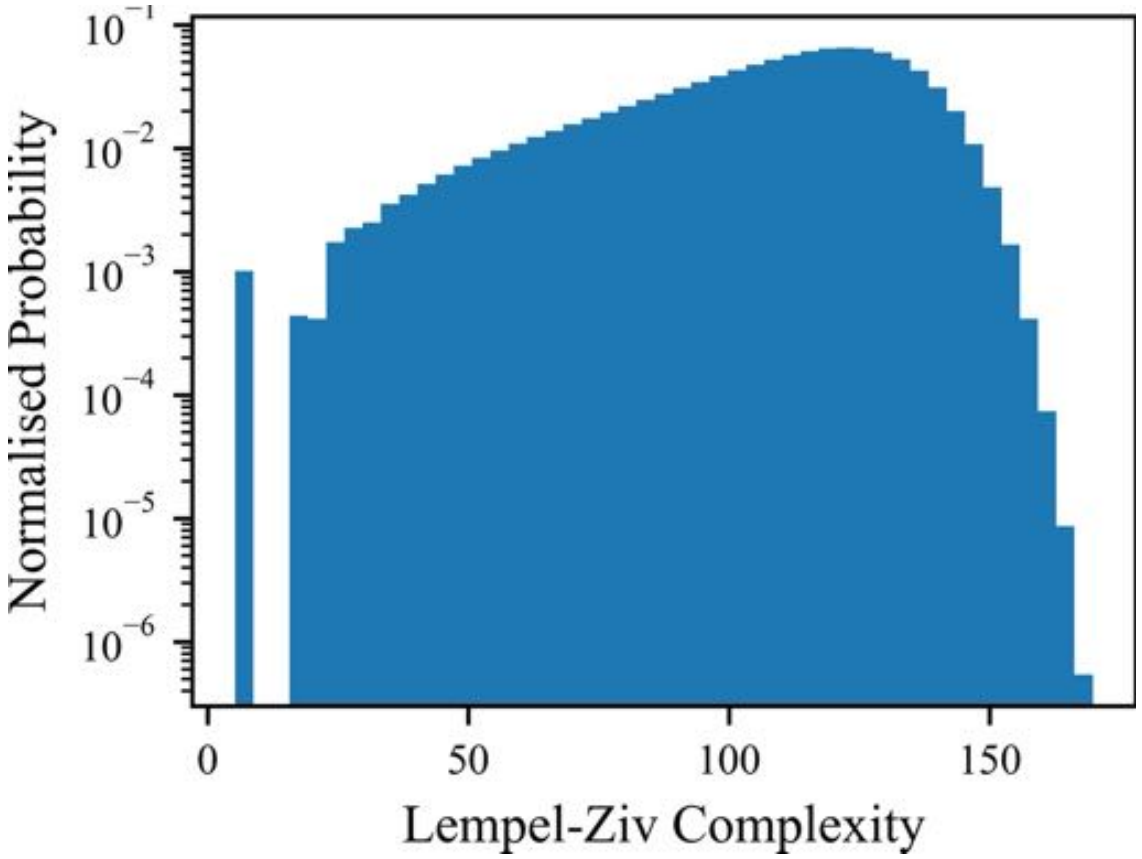
Bayesian picture and prior P(K)

$$\langle P(f|\mathcal{S})\rangle_S = P(f)\left\langle \frac{P(\mathcal{S}_i|f)}{P(\mathcal{S}_i)} \right\rangle_{\mathcal{S}_i} \approx \frac{P(f)(1-\epsilon(f))^m}{\langle P(\mathcal{S})\rangle} =$$

$P(K)$

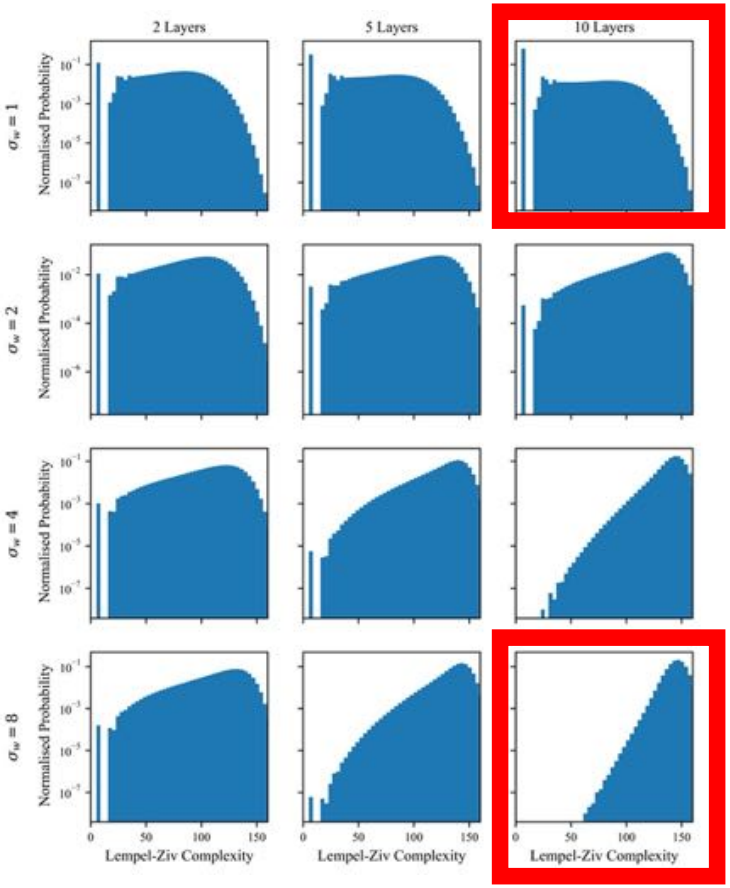
Instead of

$P(f)$



Bayesian picture and prior P(K)

$$\langle P(f|\mathcal{S})\rangle_S = P(f)\left\langle \frac{P(\mathcal{S}_i|f)}{P(\mathcal{S}_i)} \right\rangle_{\mathcal{S}_i} \approx \frac{P(f)(1-\epsilon(f))^m}{\langle P(\mathcal{S})\rangle} =$$



Ordered Regime:

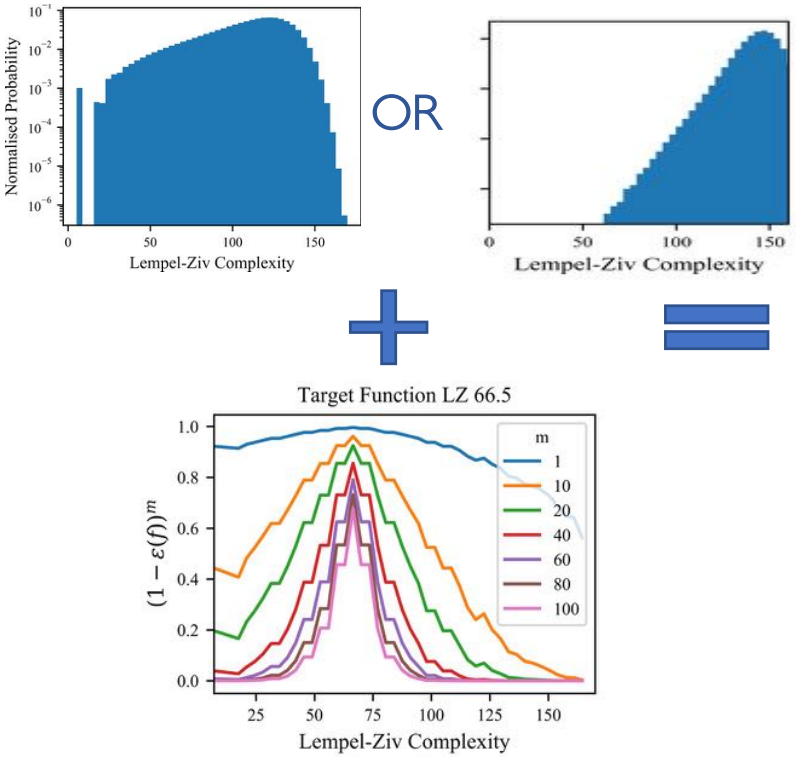
10 Layers, $\sigma_w = 1.0$

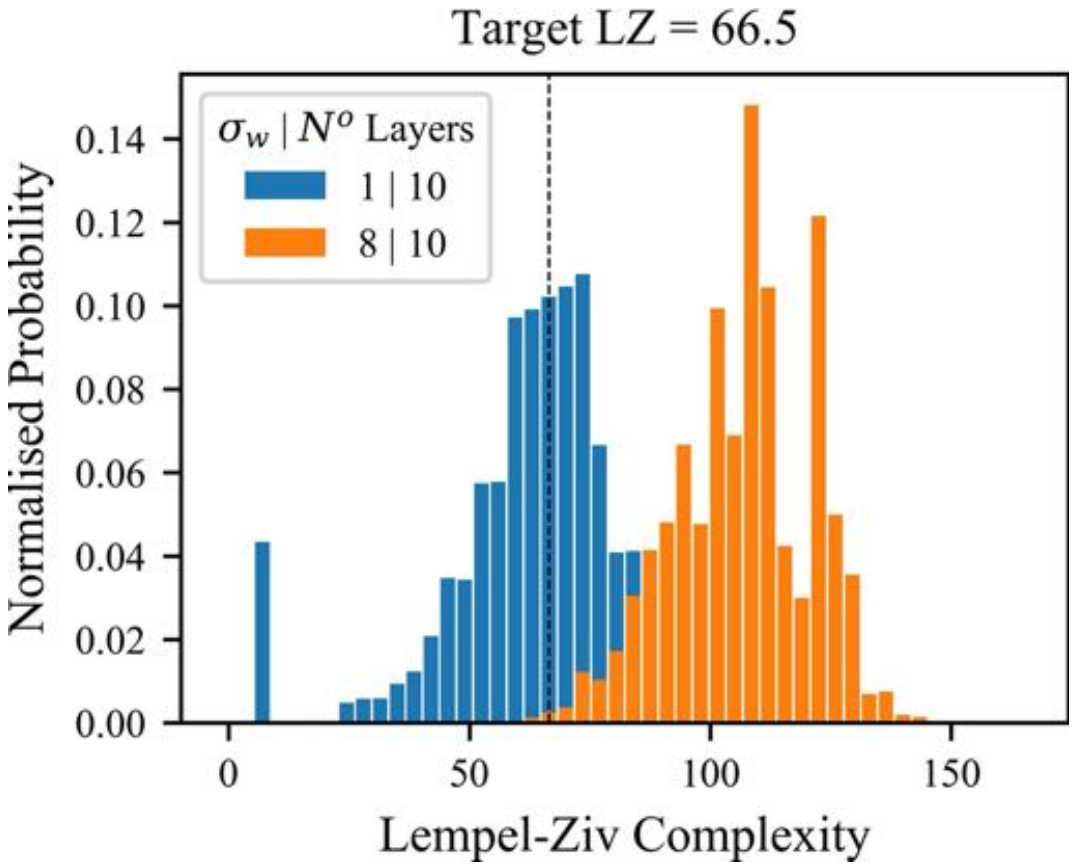
Chaotic Regime:

10 Layers, $\sigma_w = 8.0$

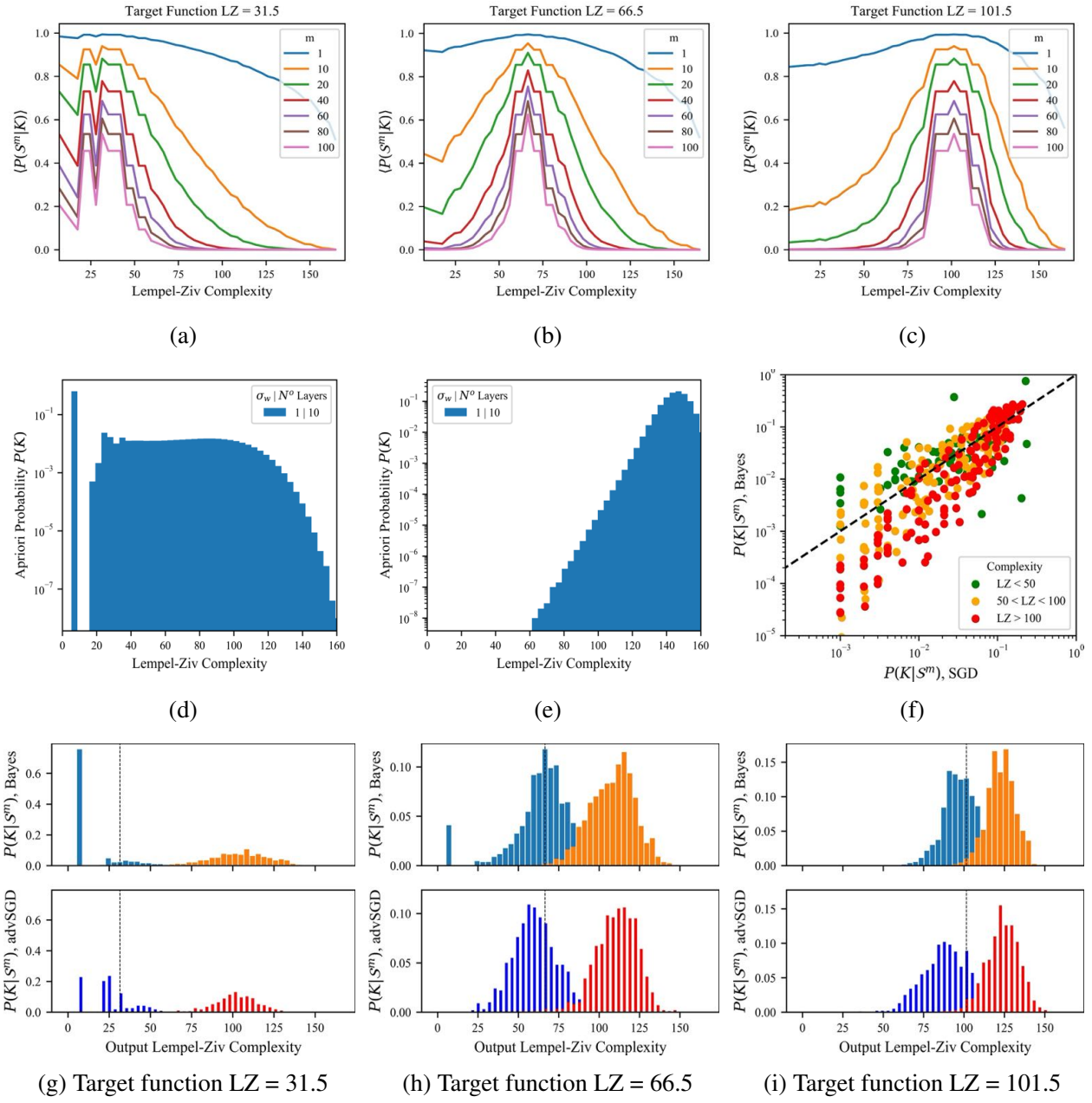
Bayesian picture: combining data and prior

$$\langle P(f|S)\rangle_S = P(f)\left\langle \frac{P(S_i|f)}{P(S_i)} \right\rangle_{S_i} \approx \frac{P(f)\,(1-\epsilon(f))^m}{\langle P(S)\rangle} \propto P(K)(1-\epsilon(f))^m$$

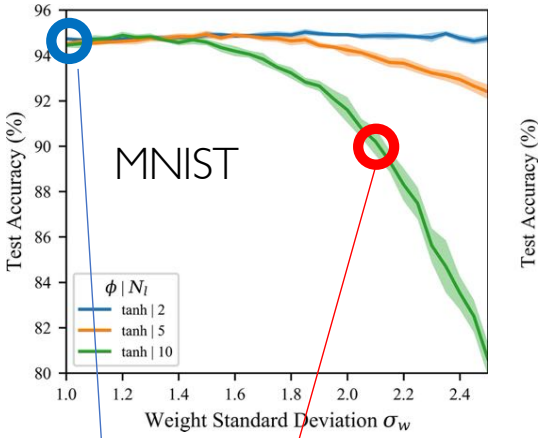




Bayesian picture combining data and prior

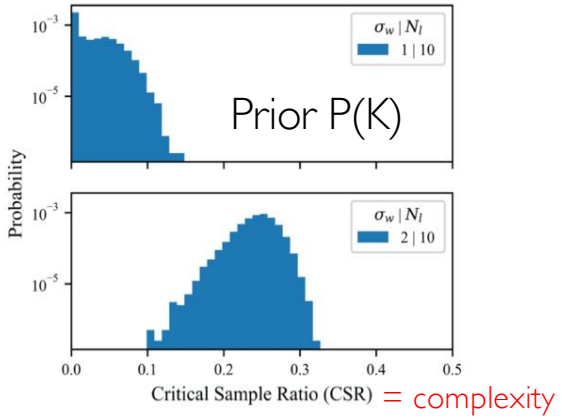


Bayesian picture: prior and data for MNIST/CIFAR-10



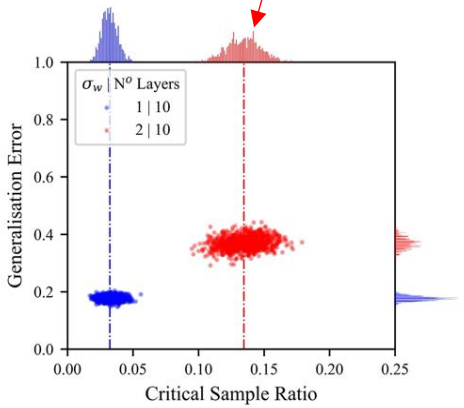
(a)

(b)

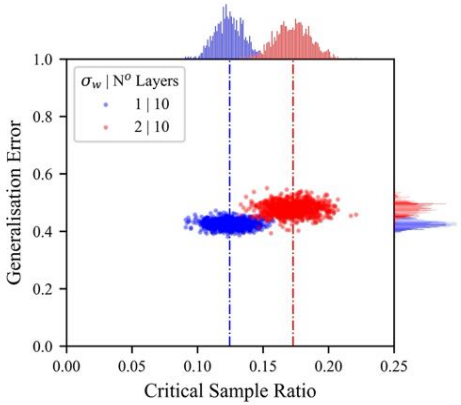


= complexity

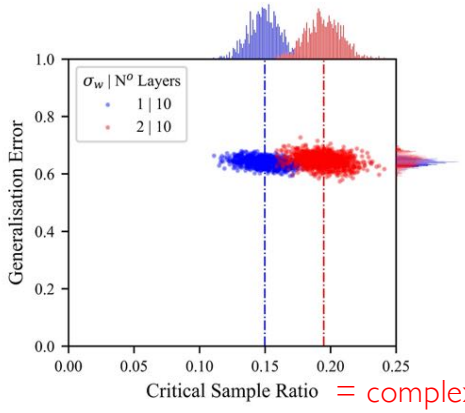
(c)



(d) Uncorrupted



(e) 25% corruption



= complexity

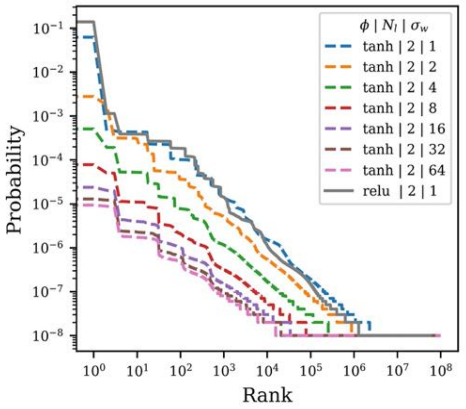
(f) 50% corruption

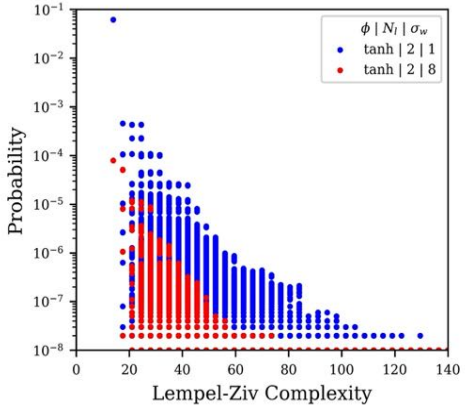
CSR complexity, Arpit et al, arXiv:1706.05394

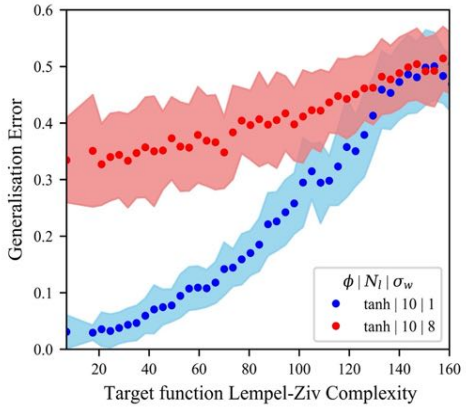
Summary: Bayesian picture: prior and data

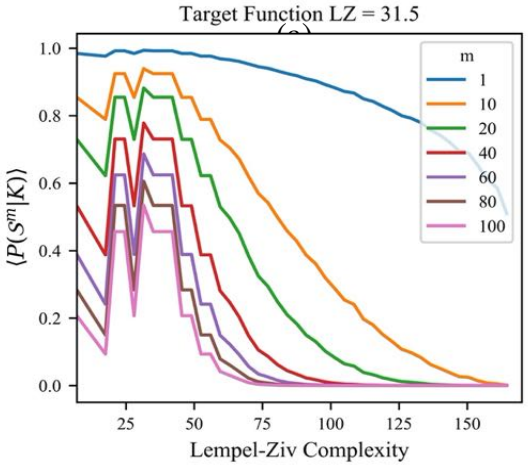
Average posterior over training sets

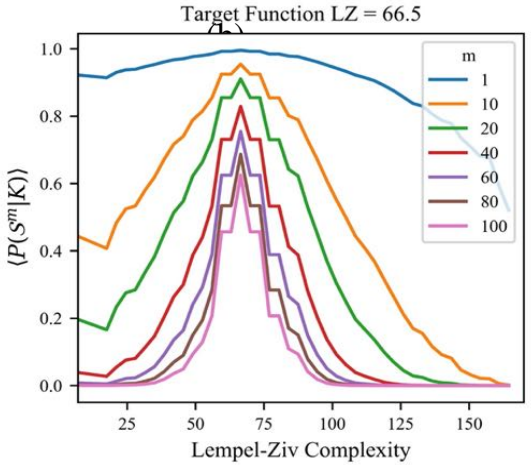
$$\langle P(f|\mathcal{S})\rangle_S = P(f)\left\langle \frac{P(\mathcal{S}_i|f)}{P(\mathcal{S}_i)}\right\rangle_{\mathcal{S}_i} \approx \frac{P(f)\,(1-\epsilon(f))^m}{\langle P(\mathcal{S})\rangle}$$

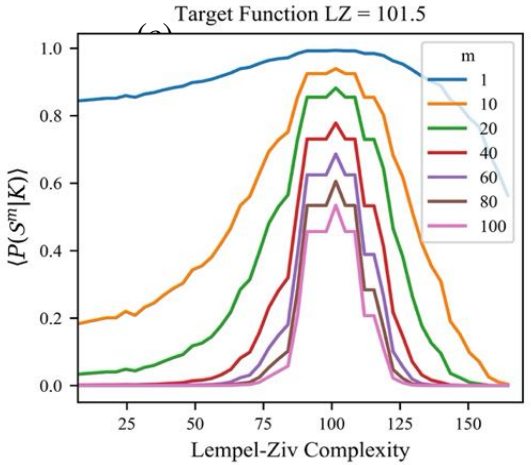






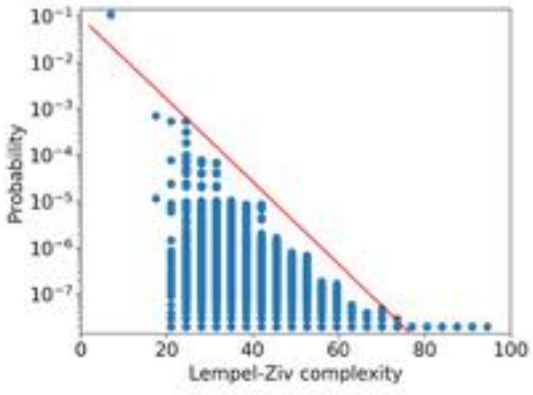






Thanks!

$$P(x) \lesssim 2^{-a\tilde{K}(x)-b}$$



Occam's razor

Conclusions:

1) DNNs generalize because they have an implicit bias towards simple functions, as predicted by AIT
2) SGD acts as a Bayesian optimizer, it is not the source of the good generalization performance
3) Many common intuitions from learning theory, such as bias-variance tradeoff etc… don't work for DNNs, but:
4)Our marginal-likelihood PAC-Bayes bound performs well


Kamal Dingle


Chico Camargo


Guillermo Valle Perez


Shuofeng Zhang


Yoonsoo Nam


David Martinez


Ouns El Harzli


Henry Rees

Hertford College undergraduates


Chris Mingard


Joar Skalse


Vlad Mikulik


Isaac Reid

Generalization theory of deep learning

How can we quantify how the inductive bias leads to generalization?

Setting: supervised learning, binary classification

Supervised learning deals with the problem of predicting outputs from inputs, given a set of example input-output pairs. The inputs live in an input domain $\mathcal{X}$, and the outputs belong to an output space $\mathcal{Y}$. We will mostly focus on the binary classification setting where $\mathcal{Y} = \{0, 1\}$. We assume that there is a *data distribution* $\mathcal{D}$ on the set of input-output pairs $\mathcal{X} \times \mathcal{Y}$. The *training set* S is a sample of m input-output pairs sampled i.i.d. from $\mathcal{D}$, $S = \{(x_i, y_i)\}_{i=1}^m \sim \mathcal{D}^m$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. We will refer to the data distribution as *noiseless* if it can be factored as $\mathcal{D}((x, y)) = \mathcal{D}_X(x)\mathcal{D}_{Y|X}(y|x)$ where $\mathcal{D}_{Y|X}$ is deterministic, that is it has support on a single value $y = f(x)$. In this case f is called the *target function*.

Generalization error:

$$\epsilon(h) = \mathbf{P}_{(x,y)\sim\mathcal{D}}[h(x) \neq y]$$

How to predict generalization?

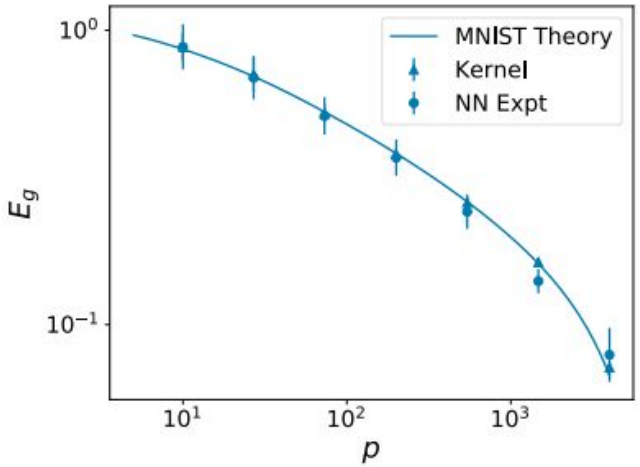
- **Bayesian approach.** Makes assumption on the data distribution

- **Frequentist approach.** Makes no (or very few) assumptions on the data distribution. We can be more confident that our predictions will work.

Bayesian approach

Make strong assumptions on data distribution. Can use this to very accurately predict generalization error. Bordelon et al. 2020 recently used the theory developed by Sollich et al. 2002 to analyze the error of some DNNs on NTK regime.

Main problem: Need to know what the data distribution D is beforehand, and no theoretical guarantees if we are wrong about this. Open empirical and theoretical question: how accurately do we need to know D for this to give useful predictions?



(b) 3-Layer NN on MNIST, $N = 800$

Bordelon et al. 2020

Relaxing the assumption that we know the data distribution, in a Bayesian analysis 😮

- Kernel Alignment Risk Estimator: Risk Prediction from Training Data. Jacot et al. 2020. Makes assumptions, but promising

Another promising approach?

- Exhaustive Learning. Schwartz et al. 1990 (link: https://www.gwern.net/docs/ai/1990-schwartz.pdf). Relate generalization to distribution of errors at initialization!

(We focus on this one)

Frequentist bounds: the general way to not make assumptions on data distribution

This is the approach taken by the probably approximately correct (PAC) family of results

Agnostic

$$\forall \mathcal{D},\ \mathbf{P}_{S\sim\mathcal{D}^m}[D(\epsilon(\mathcal{A}(S))), \hat{\epsilon}(\mathcal{A}(S))) \leq \frac{f_{\mathcal{A}}(S)}{m^{\alpha}}] \geq 1-\delta$$

Realizable

$$\forall \mathcal{D},\ \mathbf{P}_{S\sim\mathcal{D}^m}\left[\epsilon(\mathcal{A}(S)) \leq \frac{f_{\mathcal{A}}(S)}{m}\right] \geq 1-\delta$$

Classification of Frequentist Bounds

	Algorithm-independent (section 4.1)		Algorithm-dependent (section 4.2)	
	Based on uniform convergence	Based on non-uniform convergence		Other
Data-independent	VC dimension bound* (section 4.1.1)	SRM-based bounds† (section 4.2.1.1)	-	uniform stability bounds‡ and compression bounds§ (section 4.3.1)
Data-dependent	Rademacher complexity bound¶ (section 4.1.2)	data-dependent SRM-based bounds** (section 4.2.1.1)	margin bounds†† (4.2.1.2), sensitivity-based bounds‡‡ (section 4.2.1.4), NTK-based bounds§§ (section 4.2.1.3), other PAC-Bayes bounds¶¶ (section 4.2.2)	non-uniform stability bounds*** (section 4.3.1), marginal-likelihood PAC-Bayes bound††† (section 5)

Reference: Generalization bounds for deep learning. VP&L2020 (https://arxiv.org/abs/2012.04115)

Algorithm-independent bounds

$$\forall \mathcal{D},\ \mathbf{P}_{S\sim\mathcal{D}^m}\left[\forall h \in \mathcal{H},\ D(\epsilon(h), \hat{\epsilon}(h)) \leq \frac{f(S,h)}{m^\alpha}\right] \geq 1-\delta \tag{4}$$

Algorithm-independent bounds are commonly classified in two classes, according to the dependence of the capacity $f(S,h)$ on h:

- **Uniform convergence bounds** (or *uniform bounds* for short) are algorithm-independent bounds where the capacity is independent of h. This includes VC dimension bounds (section 4.1.1) and Rademacher complexity bounds (section 4.1.2). Here the nomenclature "uniform" means the value of the bound is independent of the hypothesis.
- **Non-uniform convergence bounds** (or *non-uniform bounds* for short) are algorithm-independent bounds where the capacity depends on h. Common examples include bounds for structural risk minimization (section 4.2.1.1).

Algorithm-dependent bounds

Algorithm-dependent bounds. This type of bounds generalizes the above class of bounds by considering stronger, or more general, assumptions on $\mathcal{A}$, as well as more general dependence of the capacity on $\mathcal{A}$. We can express this general class of bounds as

$$\forall \mathcal{A} \in \mathfrak{A},\ \forall \mathcal{D},\ \mathbf{P}_{S\sim\mathcal{D}^m}\left[D(\epsilon(\mathcal{A}(S))), \hat{\epsilon}(\mathcal{A}(S))) \leq \frac{f_{\mathcal{A}}(S)}{m^{\alpha}}\right] \geq 1-\delta \tag{5}$$

where $\mathfrak{A}$ is a set of algorithms that represents our assumptions on $\mathcal{A}$. An example

Algorithm-dependent bounds based on non-uniform convergence

$$\forall \mathcal{D},\ \mathbf{P}_{S\sim\mathcal{D}^m}\left[\forall h \in \mathcal{H}(S),\ D(\epsilon(h),\hat{\epsilon}(h)) \leq \frac{f(S,h)}{m^\alpha}\right] \geq 1-\delta \tag{6}$$

where $\mathcal{H}(S)$ is a set which includes $\mathcal{A}(S)$ for the class of algorithms $\mathfrak{A}$ which we are considering.

$$\forall \mathcal{D},\ \mathbf{P}_{S\sim\mathcal{D}^m}\left[\forall h \in \mathcal{H},\ D(\epsilon(h),\hat{\epsilon}(h)) \leq \frac{f(h)}{m^\alpha}\right] \geq 1-\delta \longrightarrow \forall \mathcal{D},\ \mathbf{P}_{S\sim\mathcal{D}^m}\left[D(\epsilon(h),\hat{\epsilon}(h)) \leq \frac{f(\mathcal{A}(S))}{m^\alpha}\right] \geq 1-\delta$$

Nagarajan et al. 2019 (https://arxiv.org/abs/1902.04742) showed that for such type of bounds, there are data distributions, for which the optimal such bounds, are vacuous

Desiderata for generalization theories of DL

Scale right

1. with data distribution complexity.
2. with training set size.
3. with architecture changes.
4. with optimiser changes.

and should be

5. non-vacuous/tight.
6. efficiently computable.
7. rigorous.

Desiderata for VC dimension theory

Scale right

1. with data distribution complexity. **No**
2. with training set size. **No**. It's 1/m while empirical is 1/m^alpha
3. with architecture changes. **No**. E.g. bigger NNs generalize better
4. with optimiser changes. **No.** Only depends on the algorithm via hypothesis class.

and should be

5. non-vacuous/tight. **No**
6. efficiently computable. **Yes**
7. rigorous. **Yes**

Norm-based and margin-based bounds

Scale right

1. with data distribution complexity. **Yes**
2. with training set size. **Maybe/partially**
3. with architecture changes. **Maybe/partially**
4. with optimiser changes. **Maybe/partially**

and should be

5. non-vacuous/tight. **No**
6. efficiently computable. **Yes**
7. rigorous. **Yes**

Classification of Frequentist Bounds

	Algorithm-independent (section 4.1)		Algorithm-dependent (section 4.2)	
	Based on uniform convergence	Based on non-uniform convergence		Other
Data-independent	VC dimension bound* (section 4.1.1)	SRM-based bounds† (section 4.2.1.1)	-	uniform stability bounds‡ and compression bounds§ (section 4.3.1)
Data-dependent	Rademacher complexity bound¶ (section 4.1.2)	data-dependent SRM-based bounds** (section 4.2.1.1)	margin bounds†† (4.2.1.2), sensitivity-based bounds‡‡ (section 4.2.1.4), NTK-based bounds§§ (section 4.2.1.3), other PAC-Bayes bounds¶¶ (section 4.2.2)	non-uniform stability bounds*** (section 4.3.1), marginal-likelihood PAC-Bayes bound††† (section 5)

Reference: Generalization bounds for deep learning. VP&L2020 (https://arxiv.org/abs/2012.04115)

PAC-Bayes theory - frequentist but Bayesian flavor

A powerful theory for predicting generalization error given an inductive bias. It is framed in the PAC framework, and thus it gives bounds which are true regardless of data distribution.

However, there are an infinite family of PAC-Bayes bounds, and which one will produce good results depends on assumptions on data distribution, so we still benefit from making assumptions about the data!

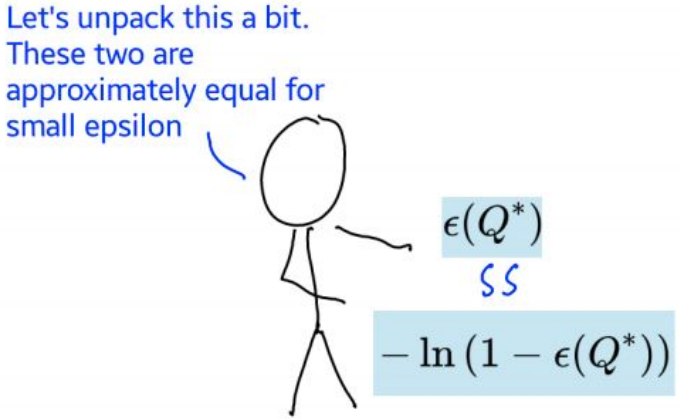
PAC-Bayes theorem (for Bayesian predictor)

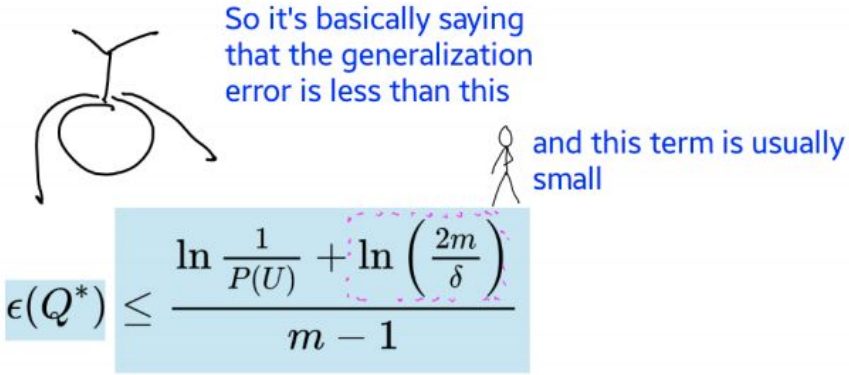
In particular, we use a (variation of) McAllister theorem for Bayesian predictors with prior P

Theorem 4.1 *(Deterministic realizable PAC-Bayes theorem) For any distribution P on any concept space $\mathcal{H}$ and any realizable distribution $\mathcal{D}$ on a space of instances we have, for $0 < \delta \leq 1$, and $0 < \gamma \leq 1$, that with probability at least $1 - \delta$ over the choice of sample S of m instances, that with probability at least $1 - \gamma$ over the choice of c:*

$$-\ln(1 - \epsilon(c)) < \frac{\ln \frac{1}{P(C(S))} + \ln m + \ln \frac{1}{\delta} + \ln \frac{1}{\gamma}}{m - 1}$$

where c is sampled from the posterior distribution $Q(c) = \frac{P(c)}{\sum_{c \in C(S)} P(c)}$, $C(S)$ is the set of concepts in $\mathcal{H}$ consistent with the sample S (with 0 training error), and where $P(C(S)) = \sum_{c \in C(S)} P(c)$





P(U) the probability, under the prior, of the labelling of the training data, is also called the **marginal likelihood**

Bayesian optimality of a frequentist bound

A frequentist bound is defined to be optimal with respect to a prior Pi over data distributions, and algorithm A, if it has the smallest expected value under Pi, among all valid frequentist bounds for algorithm A.

Theorem 6.4. *Assume* $\langle \epsilon(m) \rangle \sim bm^{-\alpha}$ *as* $m \to \infty$, *with* $0 < \alpha < 1$. *Then, assuming that for some* $E < 1$, *for all* S_{m+1}, $P_e(m) \leq E$, *there exists a constant* C', *such that, as* $m \to \infty$,

$$-\langle \log P(S_m) \rangle \sim C' b m^{1-\alpha} \tag{21}$$

Furthermore, if $Var(P_e(m)) = o(\langle \epsilon \rangle)$, *then* $C' = \frac{1}{1-\alpha}$.

$$\mathbf{E}_{S_m \sim \mathcal{D}^m} [\mathrm{PB}(m)] \sim \frac{-\langle \log P(S_m) \rangle}{m} \sim C' b m^{-\alpha} \sim C' \langle \epsilon(m) \rangle,$$

This means the PAC-Bayes bound has an expected value that asymptotically matches the true error, if it follows a power law learning curve, up to a constant, and is therefore asymptotically optimal in that sense

Evaluating the PAC-Bayes bound

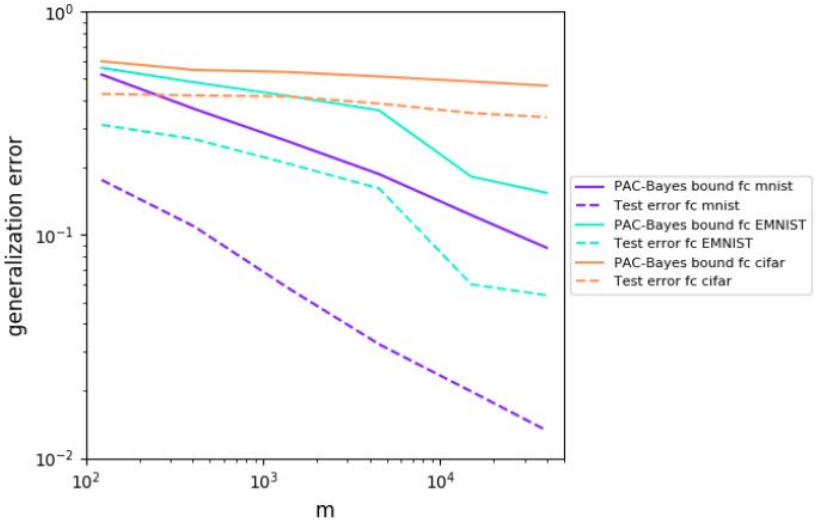
We evaluate the PAC-Bayes bound by approximating the NNGP for different architectures, datasets, and training set sizes, and using the EP approximation to estimate marginal likelihood of the data, P(U)

We estimate the NNGP kernel by sampling and computing the empirical covariance matrix of the output of the DNN (we actually use a trick introduced by Novak et al. 2018)

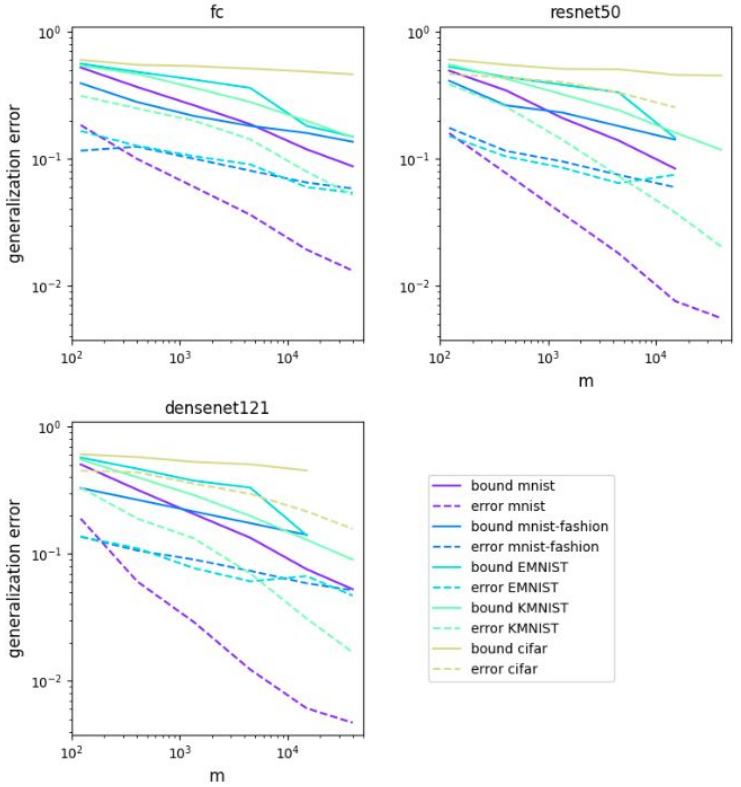
$$\left[K^l_{n,M}\right]_{\alpha,\alpha'}(x,x') \equiv \frac{1}{Mn}\sum_{m=1}^{M}\sum_{c=1}^{n} y^l_{c\alpha}(x;\theta_m)\, y^l_{c\alpha'}(x';\theta_m)$$

In the following, we present a long series of experiments comparing the bound versus the true error, for networks trained with the Adam optimiser, with batch size 32, unless I say otherwise.

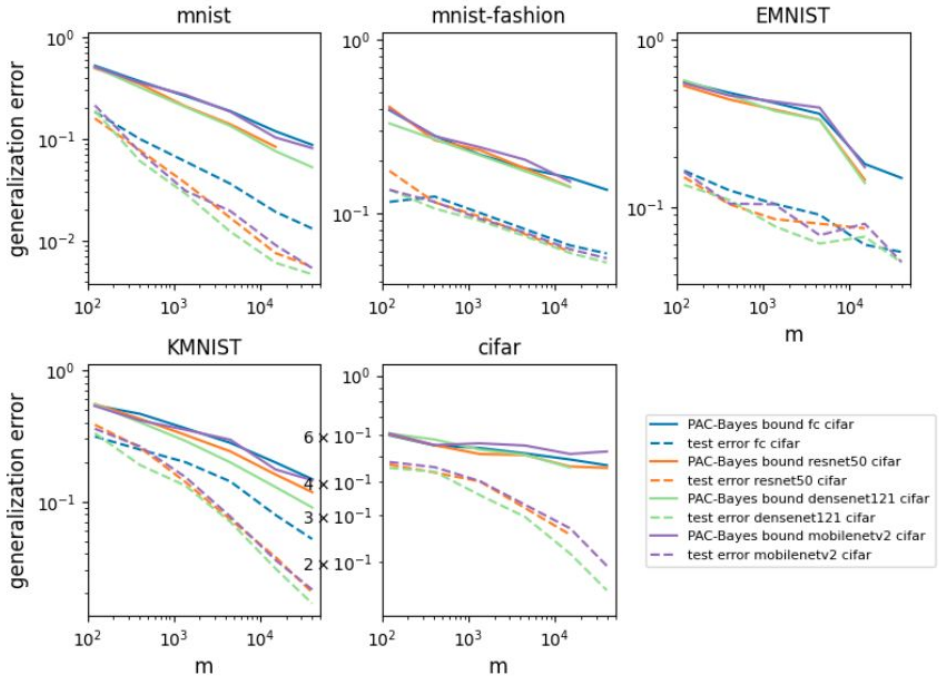
The bound predicts learning curve exponents of different datasets



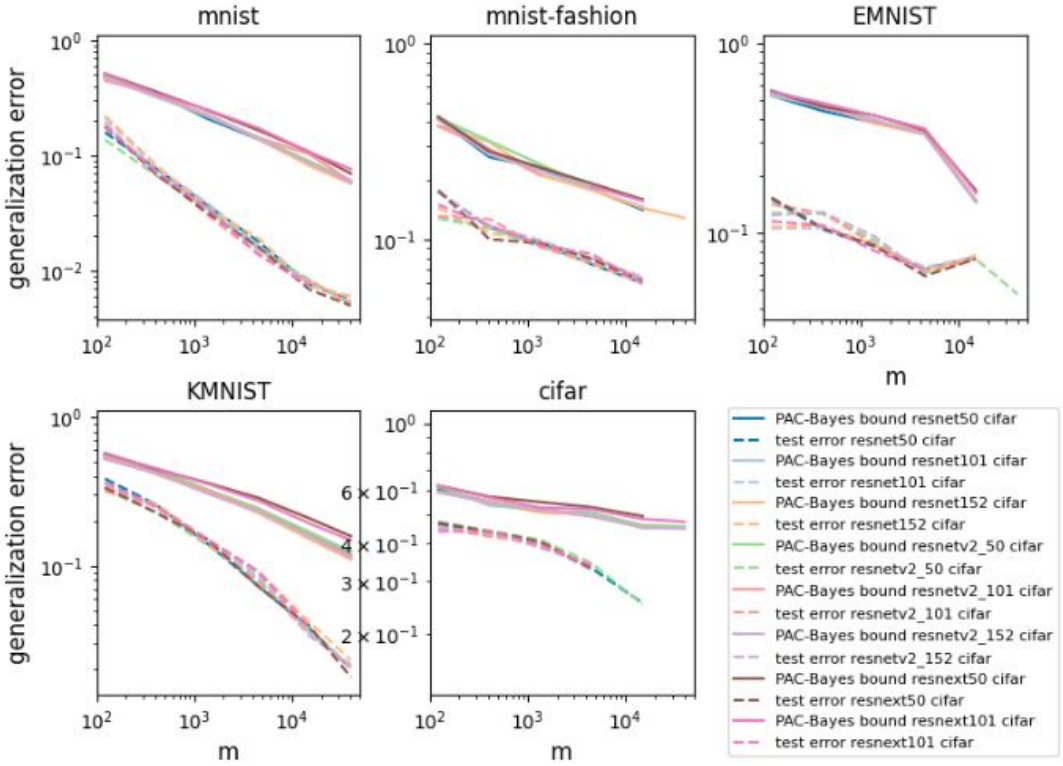
This one uses batch size 256



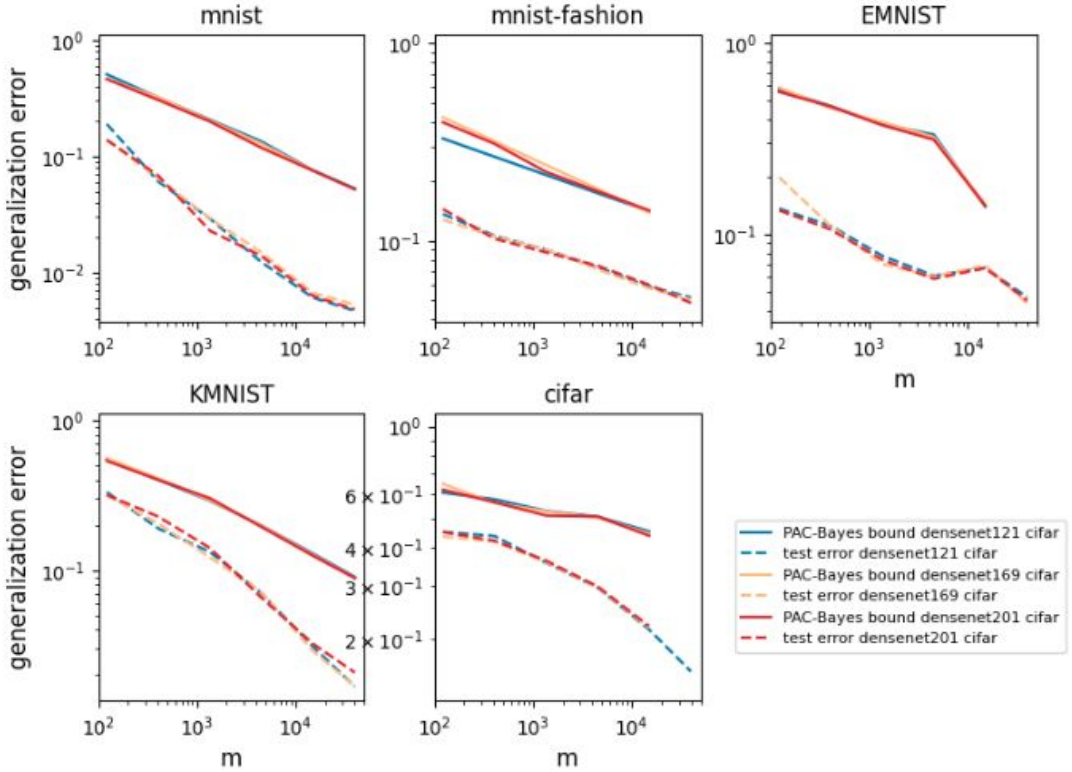
It also predicts that different architectures have similar learning curve exponents



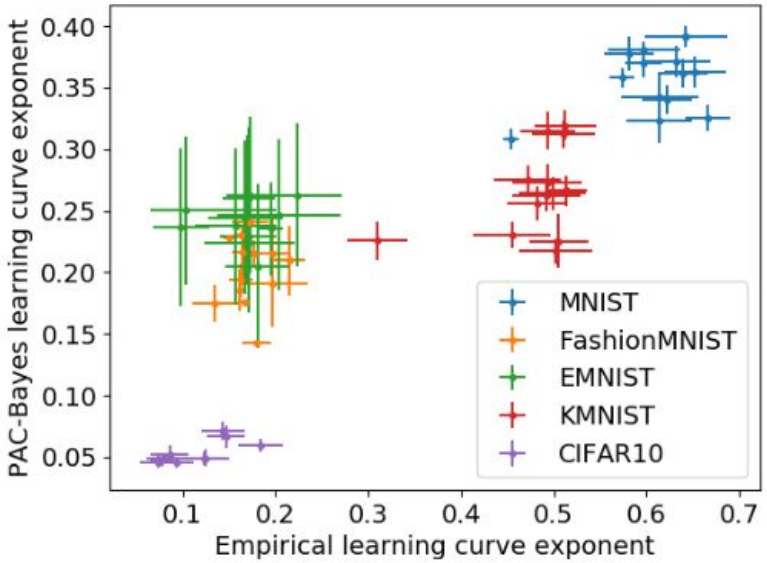
For resnets


mnist
mnist-fashion
EMNIST
KMNIST
cifar
generalization error
m
PAC-Bayes bound resnet50 cifar
test error resnet50 cifar
PAC-Bayes bound resnet101 cifar
test error resnet101 cifar
PAC-Bayes bound resnet152 cifar
test error resnet152 cifar
PAC-Bayes bound resnetv2_50 cifar
test error resnetv2_50 cifar
PAC-Bayes bound resnetv2_101 cifar
test error resnetv2_101 cifar
PAC-Bayes bound resnetv2_152 cifar
test error resnetv2_152 cifar
PAC-Bayes bound resnext50 cifar
test error resnext50 cifar
PAC-Bayes bound resnext101 cifar
test error resnext101 cifar

And densenets



The bound predicts learning curve exponents of different datasets



Bayesian optimality of a frequentist bound (reminder)

A frequentist bound is defined to be optimal with respect to a prior Pi over data distributions, and algorithm A, if it has the smallest expected value under Pi, among all valid frequentist bounds for algorithm A.

Theorem 6.4. *Assume* $\langle \epsilon(m) \rangle \sim bm^{-\alpha}$ *as* $m \to \infty$, *with* $0 < \alpha < 1$. *Then, assuming that for some* $E < 1$, *for all* S_{m+1}, $P_e(m) \leq E$, *there exists a constant* C', *such that, as* $m \to \infty$,

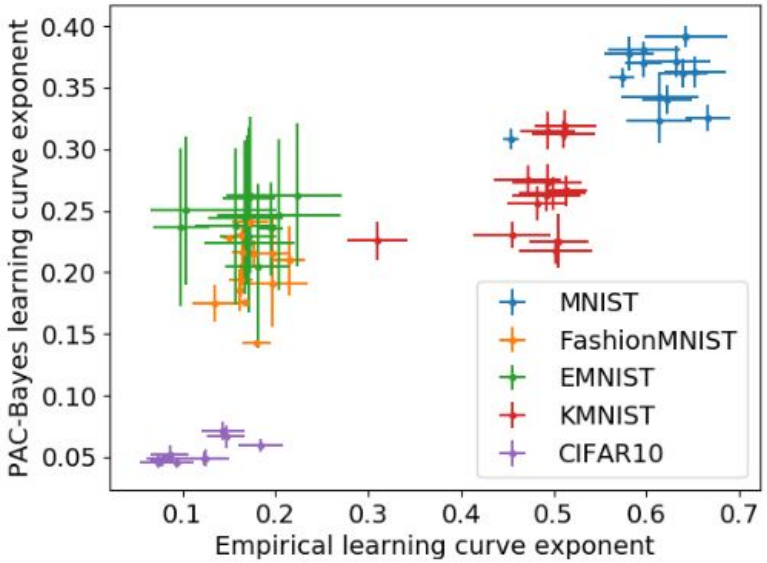
$$-\langle \log P(S_m) \rangle \sim C' b m^{1-\alpha} \tag{21}$$

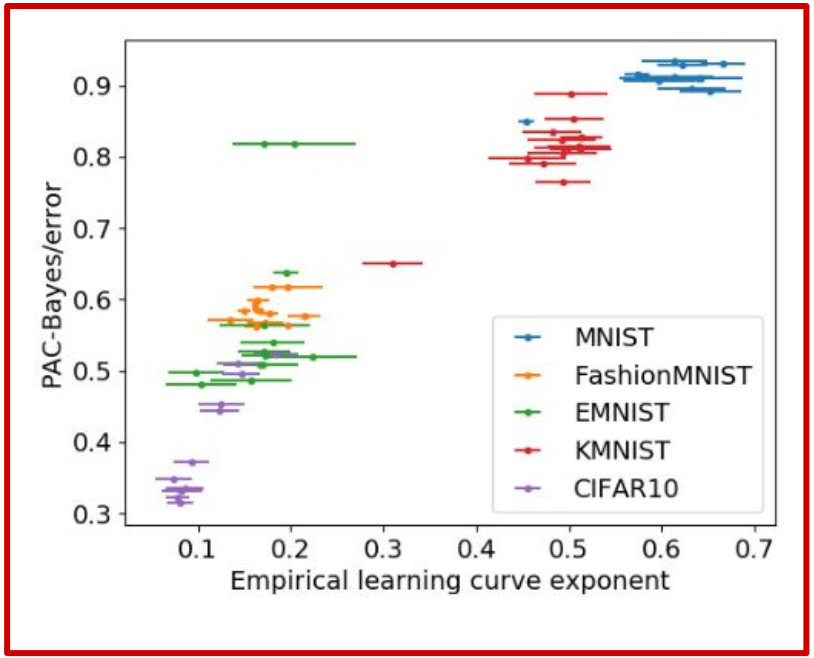
Furthermore, if $Var(P_e(m)) = o(\langle \epsilon \rangle)$, *then* $C' = \frac{1}{1-\alpha}$.

$$\mathbf{E}_{S_m \sim \mathcal{D}^m} [\mathrm{PB}(m)] \sim \frac{-\langle \log P(S_m) \rangle}{m} \sim C' b m^{-\alpha} \sim C' \langle \epsilon(m) \rangle,$$

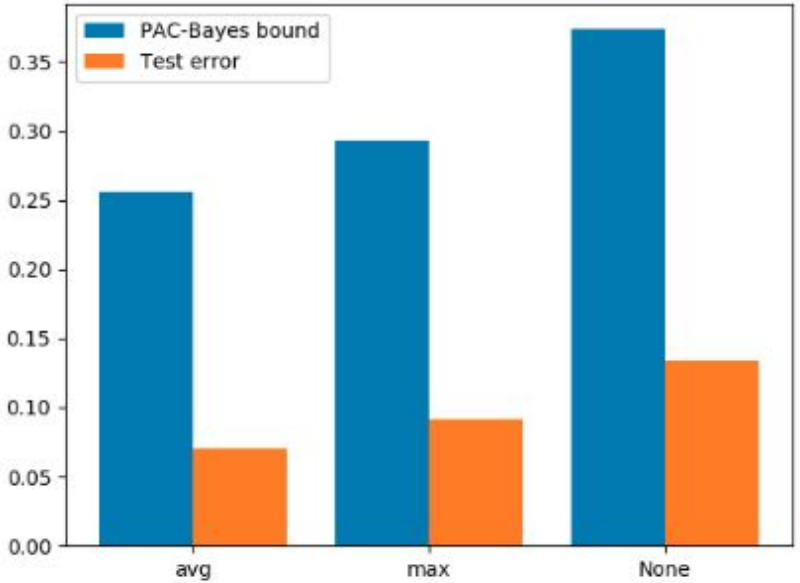
This means the PAC-Bayes bound has an expected value that asymptotically matches the true error, if it follows a power law learning curve, up to a constant, and is therefore asymptotically optimal in that sense

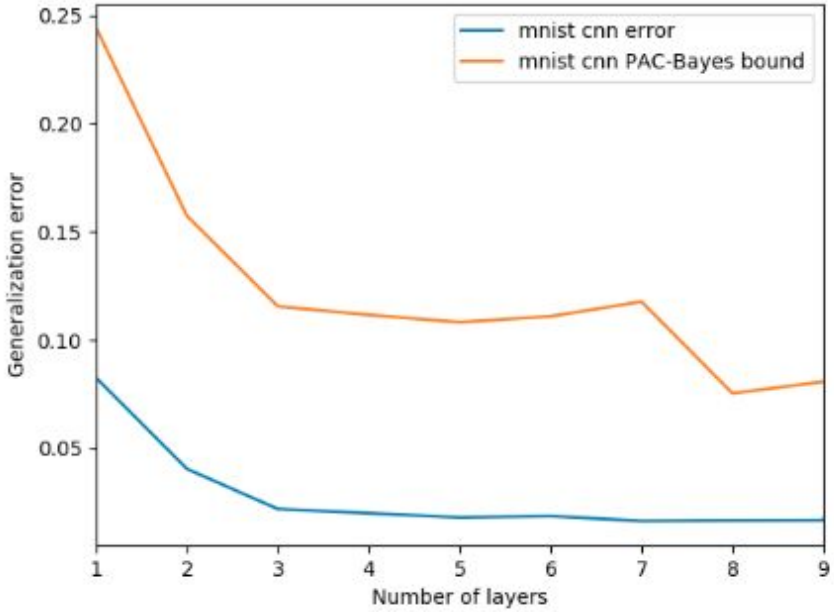
The bound predicts learning curve exponents of different datasets



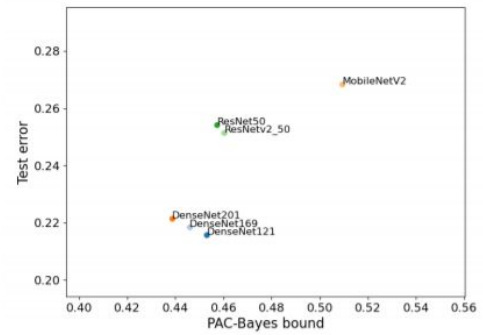


It also predicts well the difference generalization performance among different architectures

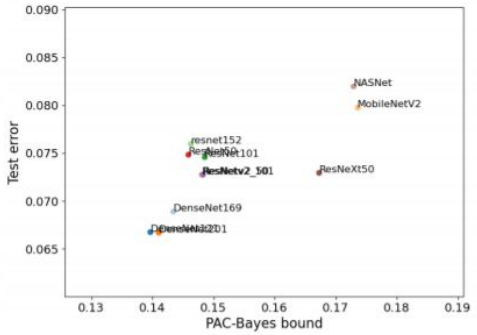




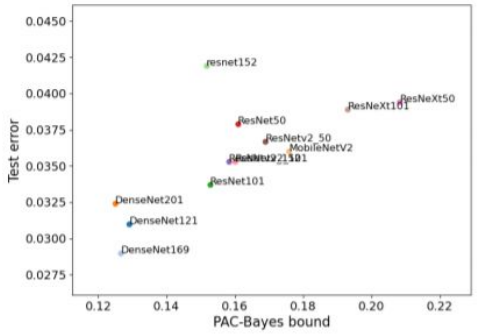
Bound vs error for different architectures



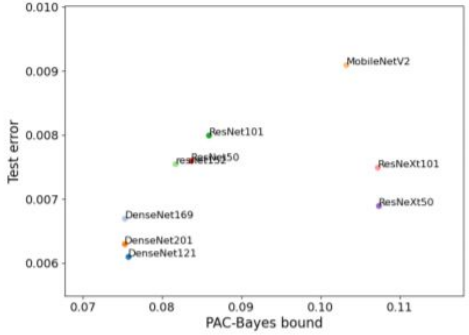
(a) CIFAR10



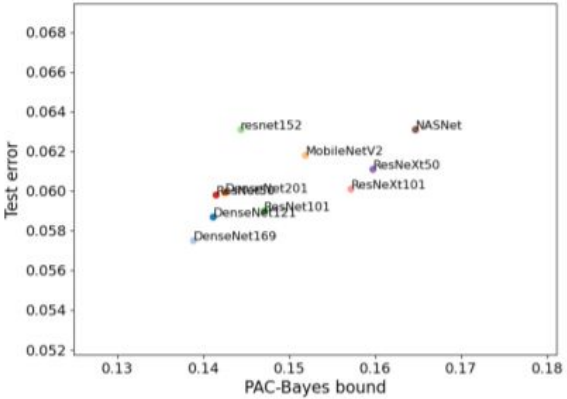
(b) EMNIST



(c) KMNIST



(d) MNIST



(e) Fashion-MNIST

Desiderata for PAC-Bayesian theory of DNNs

Scale right

1. with data distribution complexity. **Yes**
2. with training set size. **Yes**
3. with architecture changes. **Yes**, except changes to width
4. with optimiser changes. **No**

and should be

5. non-vacuous/tight. **Yes**
6. efficiently computable. **Not very, but doable**
7. rigorous. **Close**

Conclusion

This bound works remarkably well as a predictive theory of generalization.

Generalization theories should be tested extensively over a wide range of architectures, datasets, and optimizers

Future work:

Explore application of the bound, e.g. to neural architecture search, using the learning curve predictions

Explore connections, and alternative Bayesian approaches.