Why do deep neural networks generalise in the overparameterised regime?

Ard Louis

Guillermo
Valle Perez

Will answer questions in the chat

UNIVERSITY OF OXFORD

Learning machines?

Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's ? If this were then subjected to an appropriate course of education one would obtain the adult brain …..

We have thus divided our problem into two parts. The child-programme and the education process.

Alan Turing, *Computing Machinery and Intelligence,* Mind **59**, 433 (1950)

Hyper-astronomical numbers and Boolean choices

Doctor's decision table for COVID-19

Send to hospital?	Fever?	Cough?	Lost sense of smell?	Body ache?	Recent travel to hotspot?	Over 50?	Heart problem?	Obese?	Diabetes?
Boolean function	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	0	1	1	1
	1	1	1	1	1	0	0	0	0
	1	1	1	0	0	1	0	1	1
	1	1	0	0	0	1	0	1	1
			Can we learn the rest of the function?						

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

For n=9 $2^9 = 512$ answers; $2^{512} = 1.34 \times 10^{154}$ possible functions

(10^{86} elementary particles in observable universe) .

Hyper-astronomical numbers in history

A barley-corn: to a single barley-corn I increased,
2 barley-corns in the 1st day;
4 barley-corns in the 2nd day;
8 barley-corns in the 3rd day;
"
"
"

30 2 'thousand' 7 'hundred' 37 talents 1/2 mina 2 1/3 shekels 4 barley-corns in the 30th day.

Mari 08613 tablet
(Old Babylonian. 1900-1600 BC)

(47 tons of Barley)

Jöran Friberg (2005), *Unexpected Links Between Egyptian and Babylonian Mathematics*,

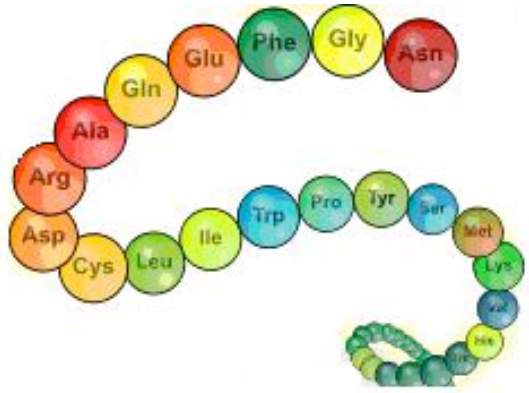
Hyper-astronomical numbers in history

Invetion of Chess: Shah-nama "The Book of Kings" by Ferdowsi, (940 – 1020)

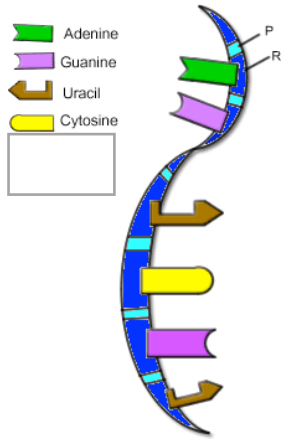
Barley corn and chess board story recounted by Ibn Khallikan c.a. 1254.

2^{64}-1= 18,446,744,073,709,551,615 (18 quintillion) > 1000 times current annual wheat production

Hyper-astronomical numbers in biology



Proteins: 20^L



RNA: 4^L

All

- L=37 proteins weigh more than earth
- L=58 proteins weigh more than the visible universe
- L=476 proteins: 10^{500} times the mass of the visible universe

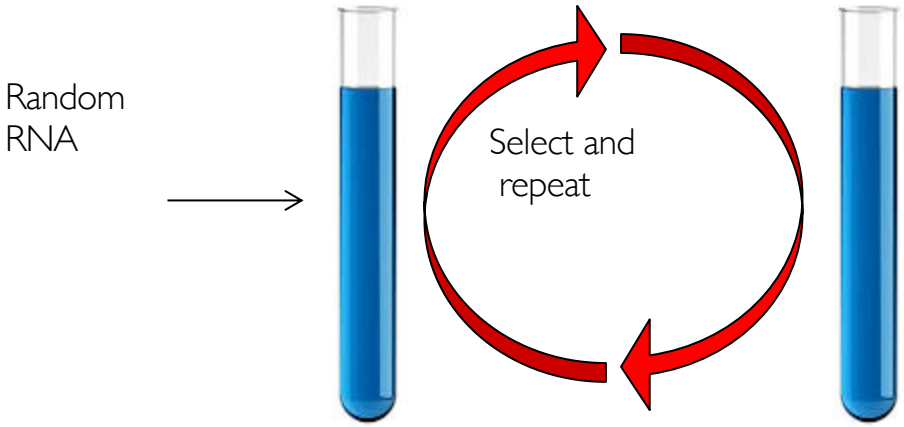
All

- L=55 RNA strands weigh 10^{10} kg
- L=79 RNA strands weigh more than the Earth
- L=126 RNA weigh more than the visible universe

AAL, Contingency, convergence and hyper-astronomical numbers in biological evolution
Studies in History and Philosophy of Biological and Biomedical Sciences 58, 107 (2016)

Evolving the same structure again and again

Question: Convergence: Natural selection or the arrival of variation?



SELEX in-vitro evolution experiments:

Hammerhead ribozyme
appears every time

Hammerhead ribozyme L≈55, so all 4^L combinations weigh 10^{10} kg

Experiment is on ~ 0.01g of material, about ~ $1/10^{15}$ fraction of the space

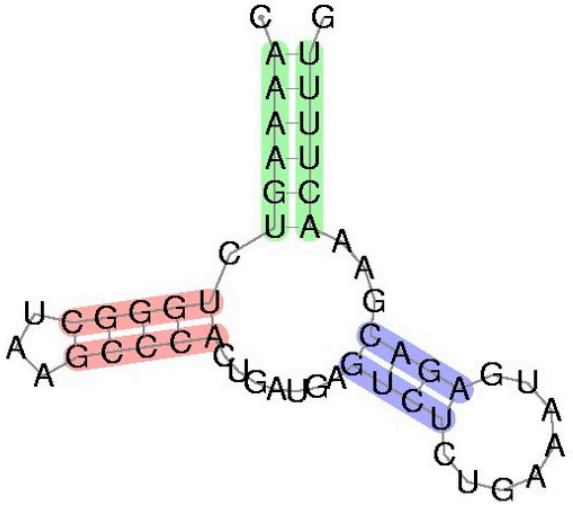
Salehi-Ashtiani K, Szostak JW. In vitro evolution suggests multiple origins for the hammerhead ribozyme. Nature. **414,** 82 (2001).

RNA Genotype → Phenotype map

GAAAGUCUGGGCUAAGCCACUGAUGGUGUCUGAAAUGAGAGGAAAACUUUUG

Hammerhead ribozyme

Folding
GP map

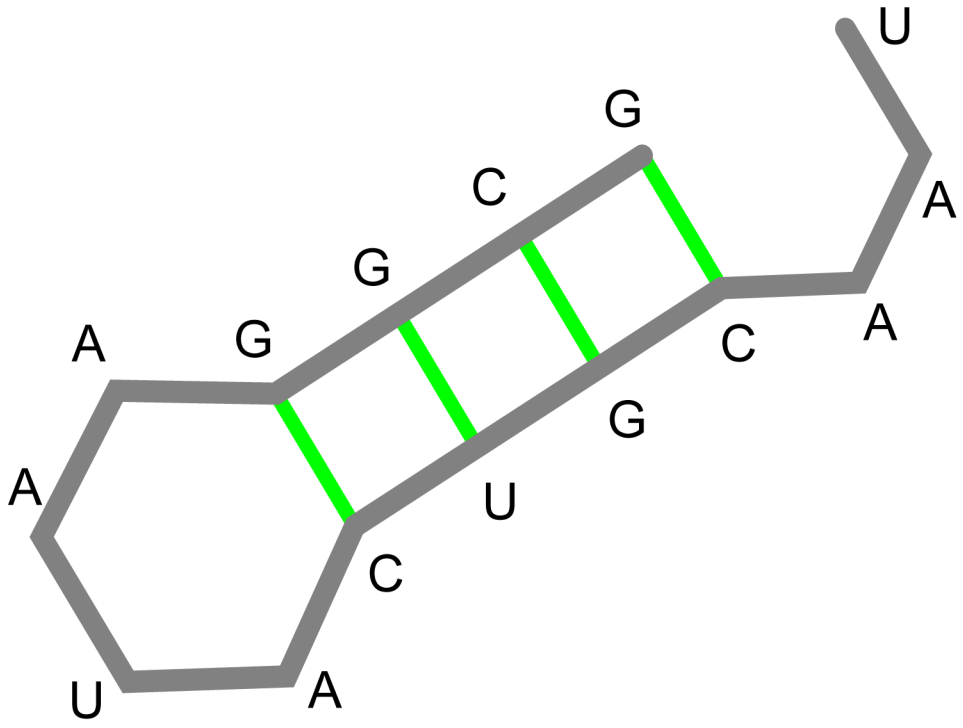


Tertiary structure (3D)

Secondary structure
(who bonds to whom)

Model G→P map: RNA secondary structures

$N_G = 4^L$

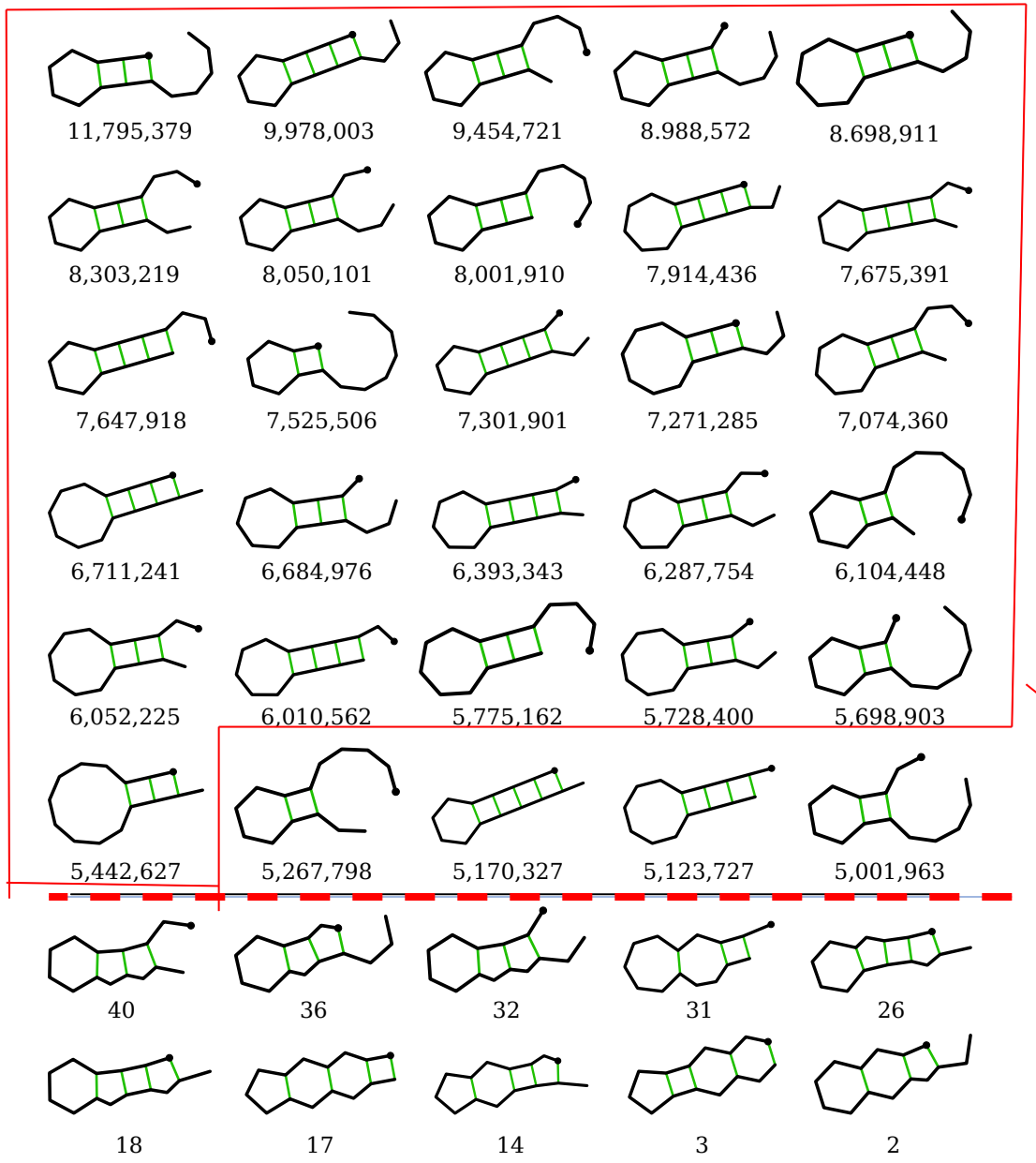


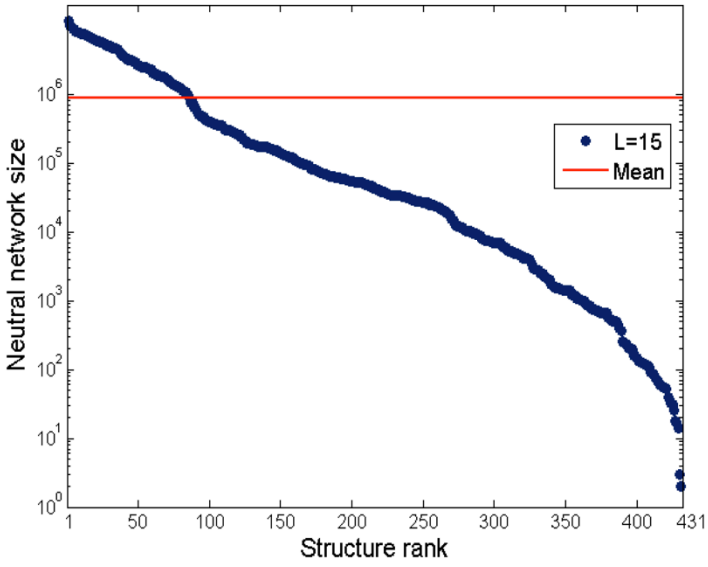
Genotype GCGGAAUACUGCAAU Sequence

Phenotype ((((. . . .)))) . . . Structure

Fast thermodynamic folding algorithms: e.g. Vienna package

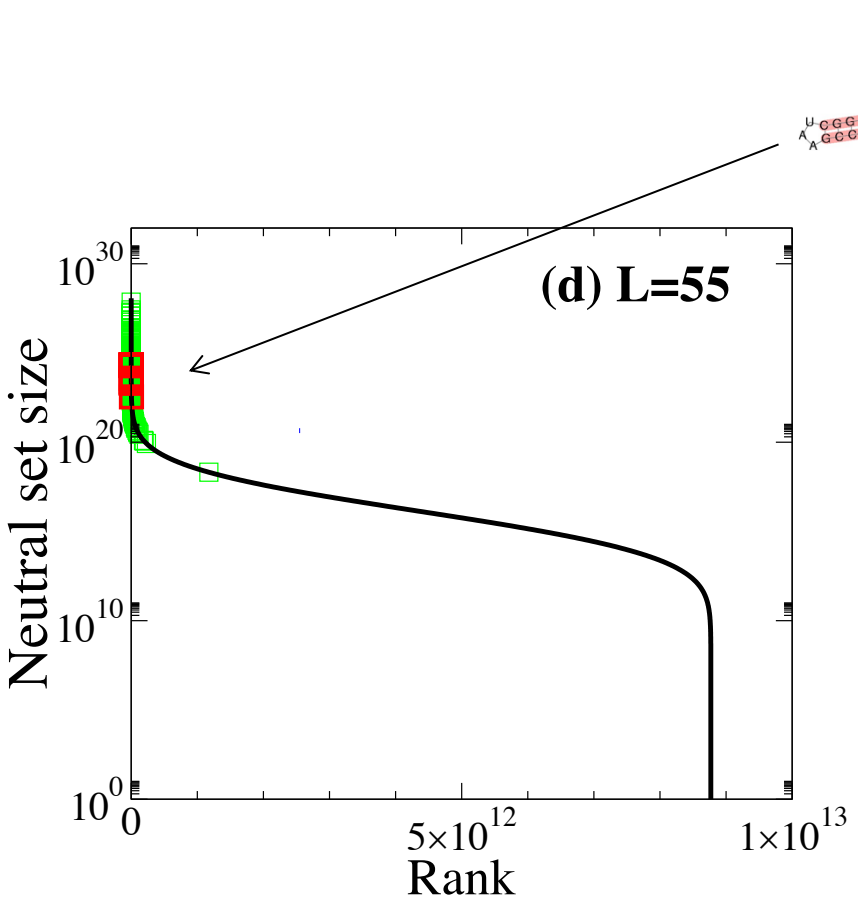
L=15: $4^{15} \sim 10^9$ sequences; these map to 431 phenotypes (note big bias!)



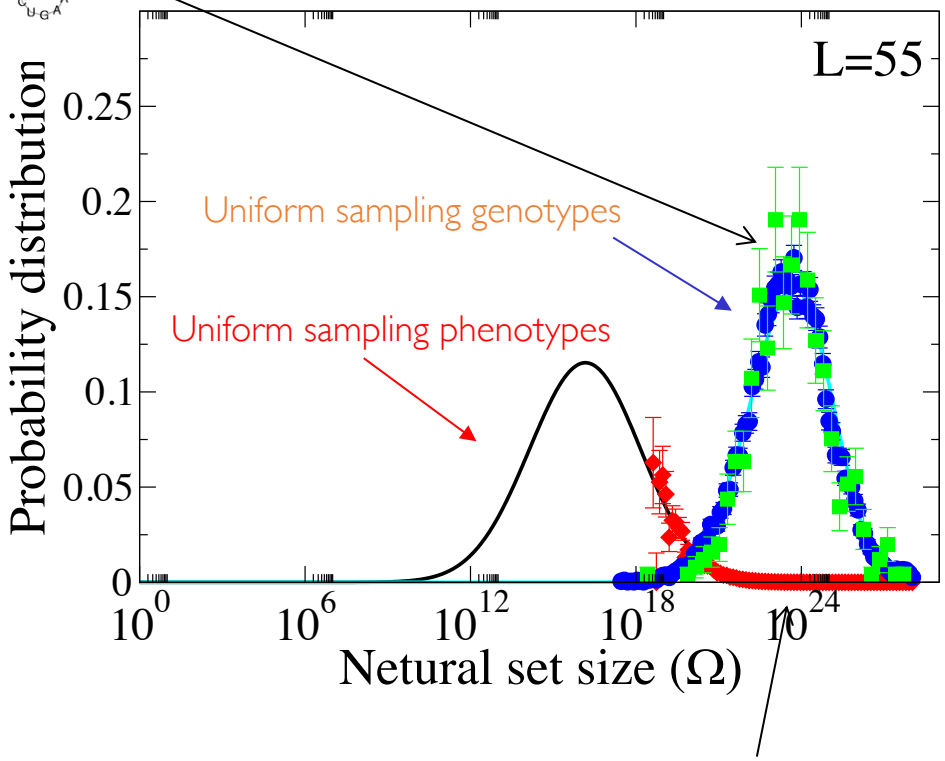


26/431 take up 50% of G-space

Hammerhead ribosome is anomalously many sequences mapping to it (large neutral set)



hammerhead ribozyme



0.001 = fraction of all possible structures take up > 50% of genotypes

10^{33} sequences → 10^{13} structures.

Green: Set of 504 sequences from functional RNA database

Kamaludin Dingle, Steffen Schaper, and Ard A. Louis, Interface Focus 5 (6), 20150053 (2015)

Why the strong bias?

INTUITION:

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

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

3.14159265358979323846264338327950288419716939
93751058209749445923078164062862089986280348 2
53421170679821480865132823066470938446095505 8
22317253594081284811174502841027019385211055 5
964462294895493038196442

But what if the monkey types into C ?

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

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

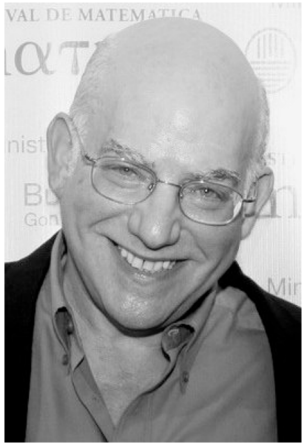
```
a[52514],b,c=52514,d,e,f=1e4,g,h;
main(){for(;b=c-=14;h=printf("%04d", e+d/f))
for(e=d%=f;g=--b*2;d/=g)d=d*b+f*(h?a[b]:f/5),a[b]=d%--g;}
```

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

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.)

Formalizing the monkey intuition with AIT: Kolmogorov complexity

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

0101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101010101

complex

0111010100110010111101111011010100001000101110101010011010111110111010010100011101101101101110

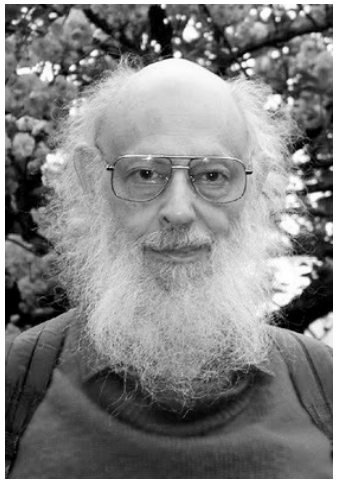
Warning: you don't know for sure that it is complex, t could be encoding **π= 3.**14159265358979323846 …..

new intutions

-- 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

Formalizing the monkey intuition with AIT: Algorithmic Probability



R. Solomonoff
1926-2009

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

First term is the biggest one

Sum all 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.

Formalizing the monkey intuition with AIT: The Coding Theorem

L. Levin, 1948 --

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

We should teach this much more widely!

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

Serious problems for applying coding theorem

1) Many systems 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.

AIT coding (like) 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 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 – *O(1)*
3) This procedure gives a bound on the Kolmogorov complexity **given f and 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)

AIT coding (like) theorem for 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)
1) K(x) is approximated, for example by Lempel Ziv compression or some other suitable measure.
2) Constants a and b depend on mapping only and can be approximated fairly easily.
3) Bound is tight for most inputs, but not most outputs.
4) Maps must be a) simple, b) redundant, c) non-linear, d) well-behaved (e.g. not a pseudorandom number generator)
5) 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

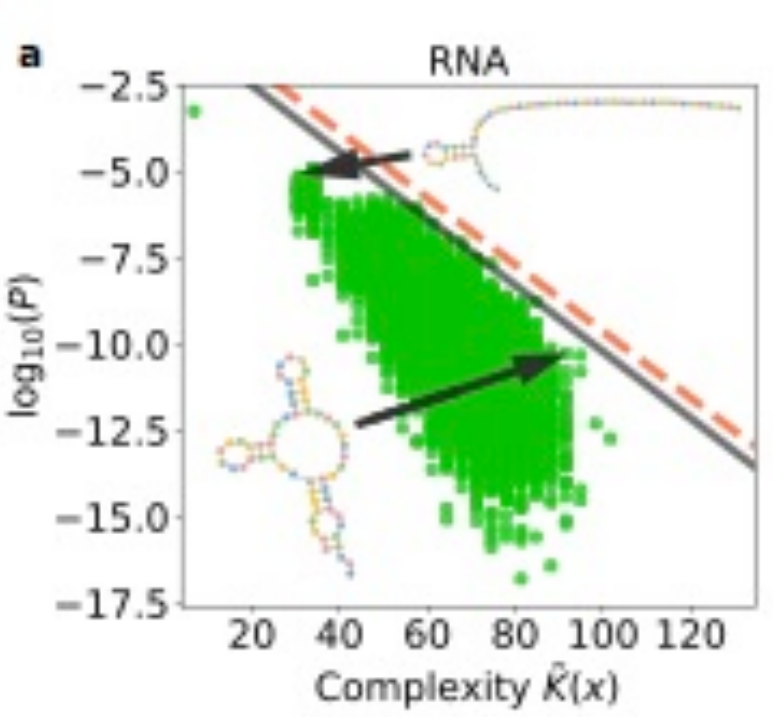
RNA map shows simplicity bias

Frequent

```
[[7.0, -0.13, '.................................................'],
 [31.0, -2.01, '.....................................((((....))))).....'],
 [34.0, -2.01, '..((((((.....))))))..................................'],
 [37.0, -3.38, '................(((((..(((((.......)))))..)))))........'],
 [41.0, -2.95, '....................(((..(((........)))..)))..........'],
 [44.0, -3.08, '((((.(((((........)))))))))..........................'],
 [44.0, -3.15, '.............((((.(((.....)))))))....................'],
 [47.0, -2.81, '..........................((((..(((.....)))..))))..'],
 [51.0, -2.99, '........................((((.(((........))).))))).....'],
 [54.0, -2.95, '..........................((..((((.....))))..)).....'],
 [58.0, -3.48, '..(((.((((((..............)))))).)))..................'],
 [61.0, -3.86, '...........((((..((((((.......))))))..))))............'],
 [61.0, -5.06, '(((.(((...))).)))...........(((((...)))))............'],
 [64.0, -4.21, '..........(((((.((....(((((........))))....)).)))))....'],
 [64.0, -4.38, '.(((((.((....((((......))))...)).)))))................'],
 [68.0, -4.29, '..............(((.((..(((((.....)))))..)).)))........'],
 [71.0, -4.55, '(((.((((...((((((.........))))))..))))..)))..........'],
 [75.0, -5.32, '...........(((((.(..((....(((......)))....))..).)))))'],
 [78.0, -5.84, '(((.((.....(((...(((((....)))))..)))....)).)))........'],
 [78.0, -5.28, '............(((.((((..((.((((((....))))))..))..))))).)))'],
 [81.0, -6.01, '((((..((((((....(((.(......).)))....))))))...))))......'],
 [85.0, -6.41, '(((((...(((..(....)..)))...))))).((((.((....)).))))....'],
 [88.0, -6.54, '...((.((((((..(..(((..((((....))))..))))..)..))))))..))..'],
 [88.0, -7.67, '.(((((...(((((((.....))))..)))..))))).....((.(((....))).))'],
 [92.0, -7.94, '((..((....))..)).((((((......))))))((.(((....))).))....'],
 [95.0, -7.46, '((..(.....(((((((..(.((((....)))).)..)))))))).....)..)).']]
```

Complexity →

More rare

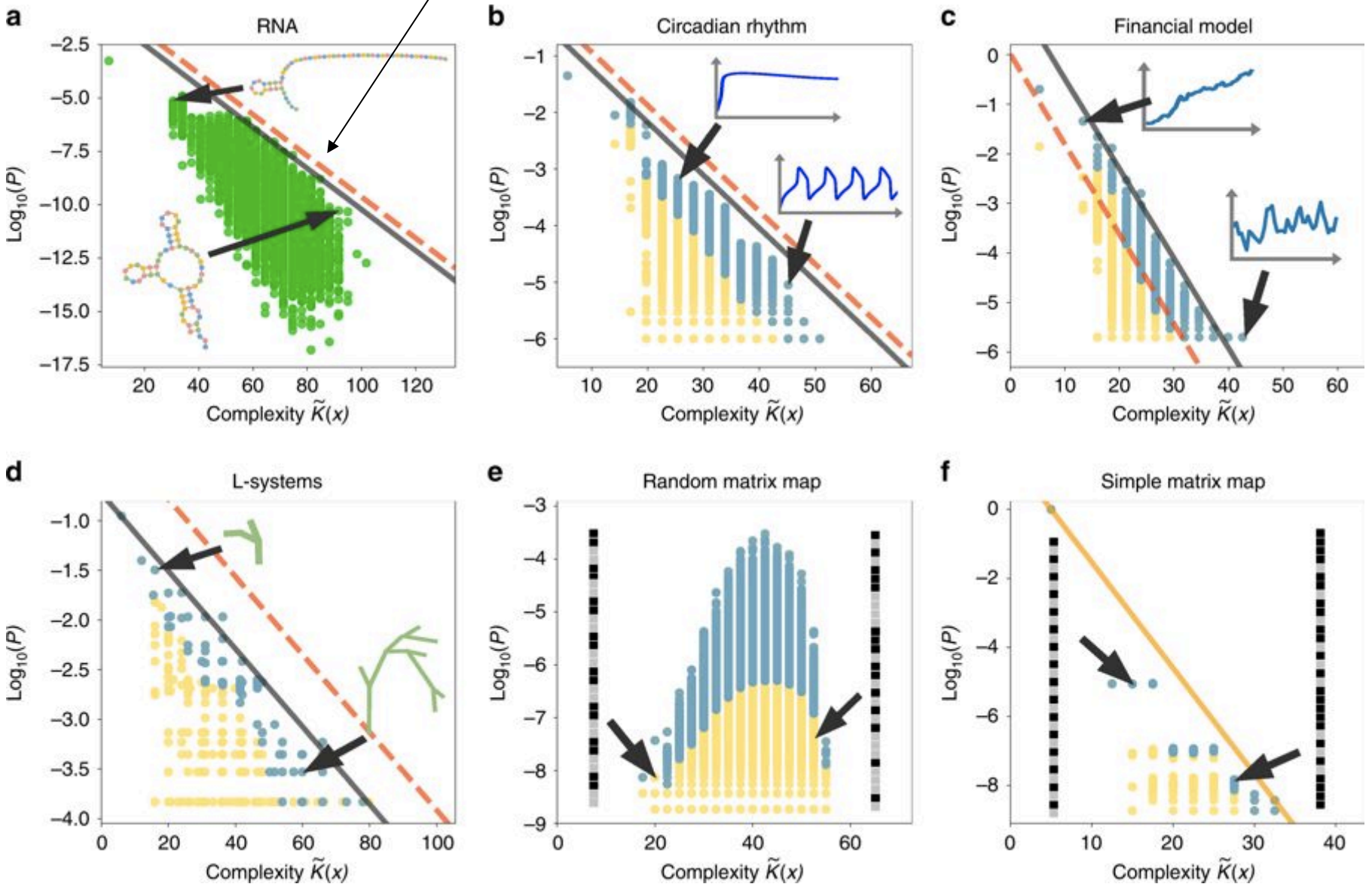


We compress dot-bracket notation

This trend holds for other systems as well

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

Is black line (red dashed with b=0)



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

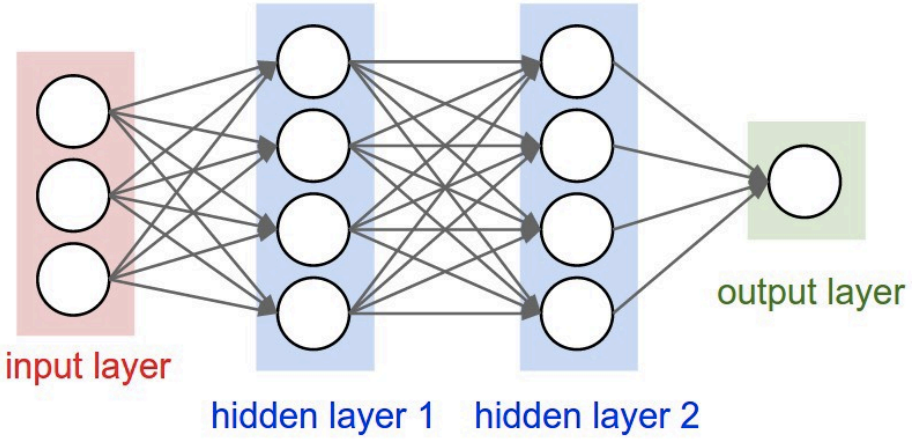
Can simplicity bias be applied to supervised learning with deep neural networks?

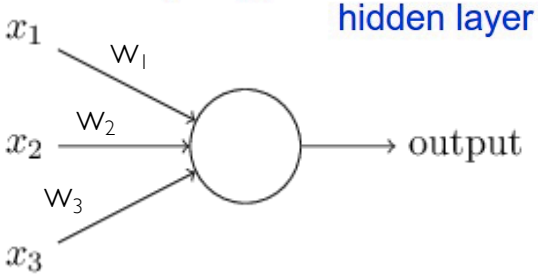
Neural networks are highly expressive

We have thus divided our problem into two parts. The child-programme and the education process.

- A Turing (1950)

Child-programme: Neural Network





$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

Universal approximation theorem for NN

Neural networks are fundamentally function approximators. The following theorem holds:

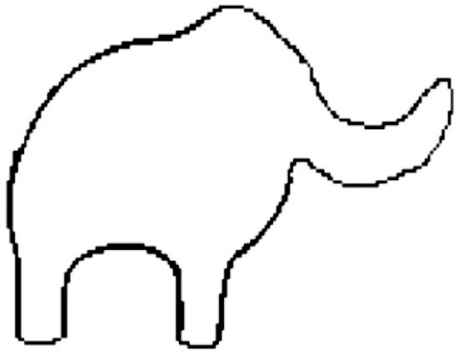
For any Lebesgue-integrable function $f : \mathbb{R}^n \to \mathbb{R}$ and any $\epsilon > 0$, there exists a fully-connected ReLU network $\mathcal{A}$ with width $d_m \leq n + 4$, such that the function $F_{\mathcal{A}}$ represented by this network satisfies

$$\int_{\mathbb{R}^n} |f(x) - F_{\mathcal{A}}(x)| \, \mathrm{d}x < \epsilon$$

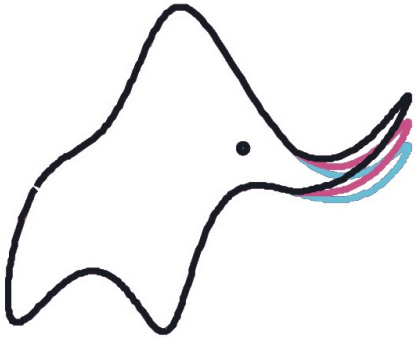
Neural networks are highly expressive -

B. Hanin Approximating Continuous Functions by ReLU Nets of Minimal Width. arXiv preprint arXiv:1710.11278.

Neural networks are typically highly over-parameterized: number of parameters >> number of data points



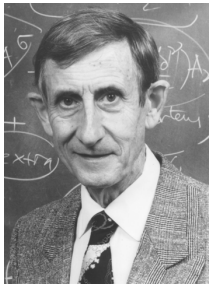
4 parameters



5 parameters

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

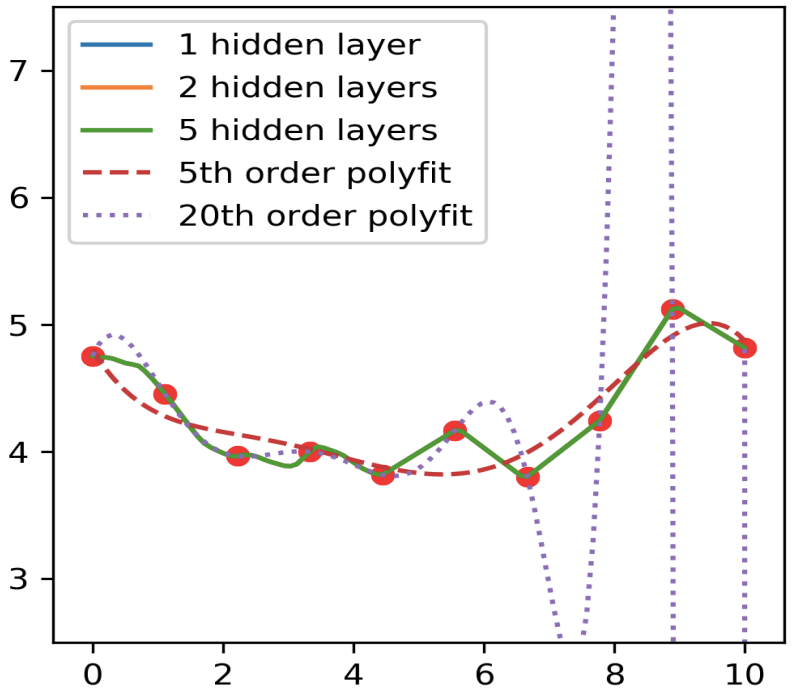
F. Dyson, *A meeting with Enrico Fermi.* Nature. 427, 287 (2004)



1923-2020

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

Neural networks are typically highly over-parameterized: number of parameters >> number of data points



Why do the DNNs not over-fit?

Comparison of a polynomial fit to a DNN fit (with thousands of parameters)

$$y(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots\ldots\ldots a_n x^n$$

Bias-Variance tradeoff in Machine Learning

Why does this concept not work well for DNNs?

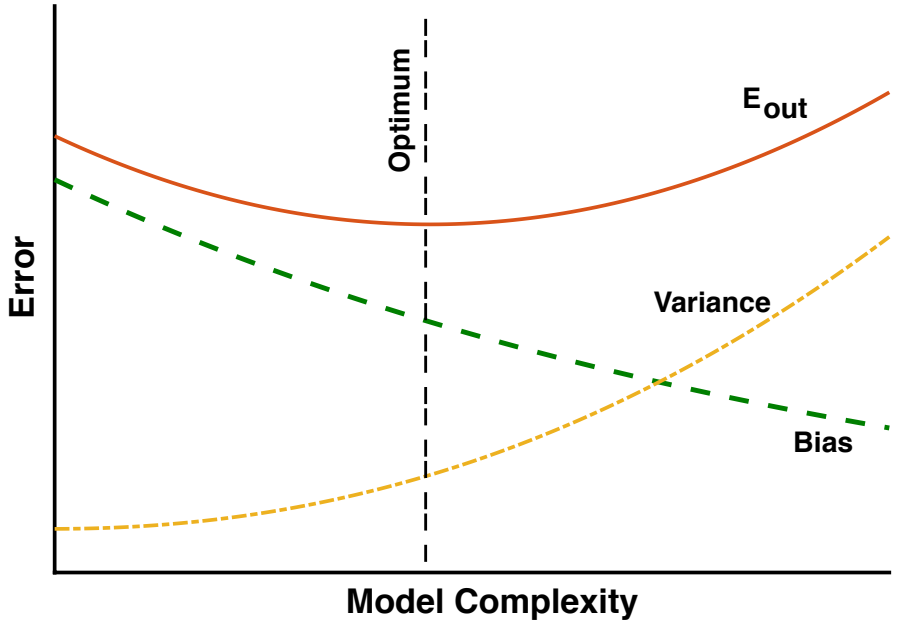


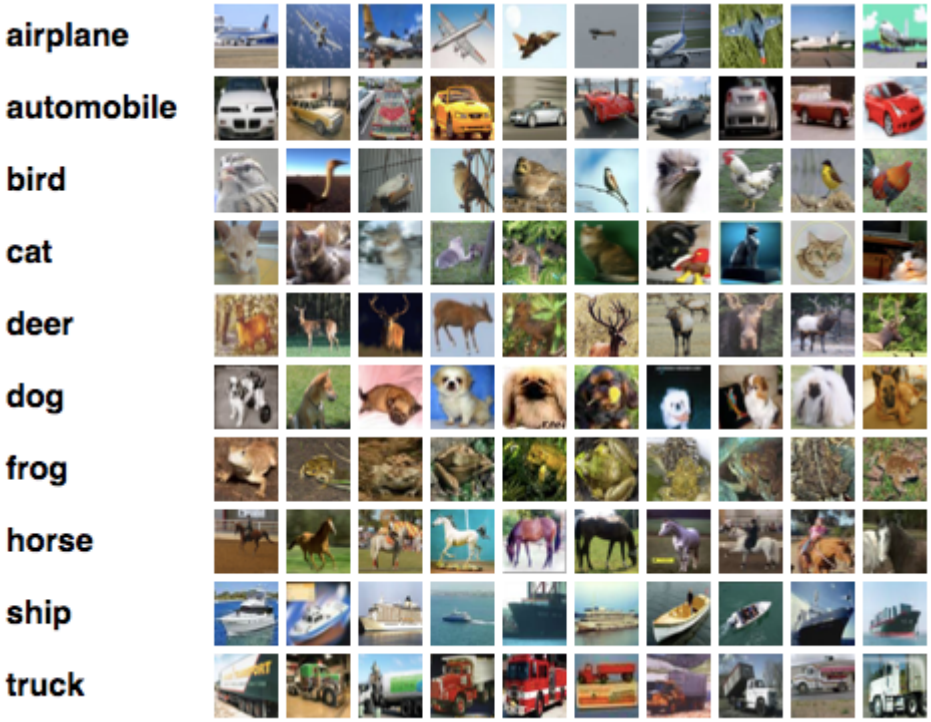
FIG. 5 **Bias-Variance tradeoff and model complexity.**

From **A high-bias, low-variance introduction to Machine Learning for physicists**

Pankaj Mehta, Ching-Hao Wang, Alexandre G. R. Day, and Clint Richardson

Arxiv/1803.08823

Conundrum: if DNNs are highly expressive, why do they pick functions that generalize so well?



CIFAR-10 dataset

C. Zhang et al., **Understanding deep learning requires rethinking generalization**.
arXiv:1611.03530 (2016)
Showed that you could randomise the labels, and still easily train to zero training error.

If a DNN can "memorize" a dataset, why does it pick functions that generalise so well?

Supervised learning of a Boolean function with DNNs

Doctor's decision table for COVID-19

Send to hospital?	Fever?	Cough?	Lost sense of smell?	Body ache?	Recent travel to hotspot?	Over 50?	Heart problem?	Obese?	Diabetes?
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	0	0	0	0
1	1	1	1	0	0	1	0	1	1
0	1	1	0	0	0	1	0	1	1
	Given some examples, can we learn the rest of the function?								

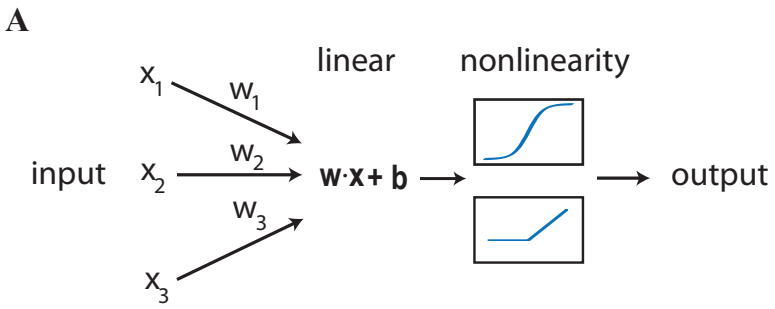
Boolean function

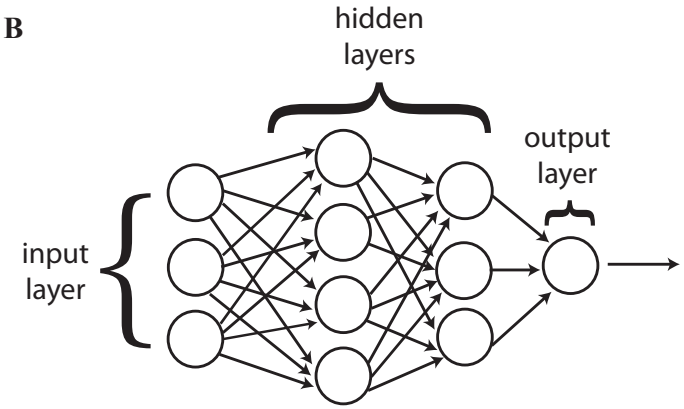
n questions; 2^n possible answers; 2^{2^n} possible Boolean functions

For n=9 $2^9 = 512$ answers; $2^{512} = 1.34 \times 10^{154}$ possible functions

(10^{86} elementary particles in observable universe) .

Parameter-function map for deep learning





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:

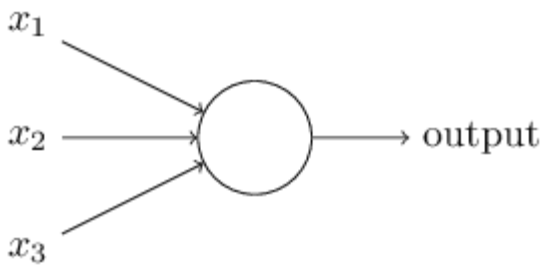
$$\mathcal{M} : \Theta \to \mathcal{F}$$
$$\theta \mapsto f_\theta$$

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

A-Priori probability: 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*

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

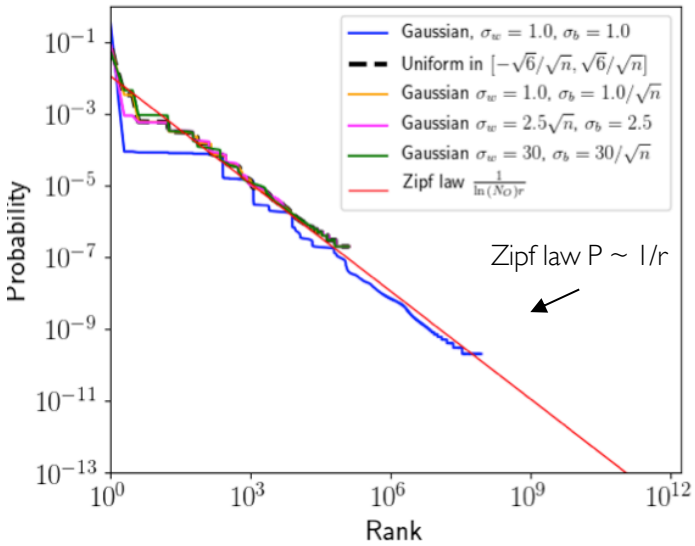
We also prove that this bias towards simple function gets stronger with more layers

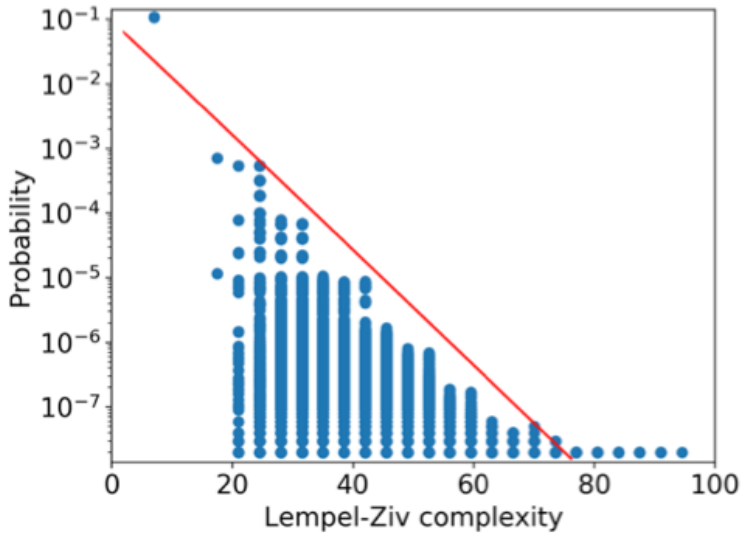
Neural networks are a priori biased towards Boolean functions with low entropy, Chris Mingard, Joar Skalse, Guillermo Valle-Pérez, David Martínez-Rubio, Vladimir Mikulik, Ard A. Louis arxiv:1909.11522

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

Model problem for a 7 bit string, study all Boolean functions f.
There are $2^7 = 128$ different strings, and $2^{128} \backsimeq 10^{38}$ different functions.
You might expect a 10^{-38} chance of finding any function.
Instead, we find strong simplicity bias.

Guillermo Valle Perez

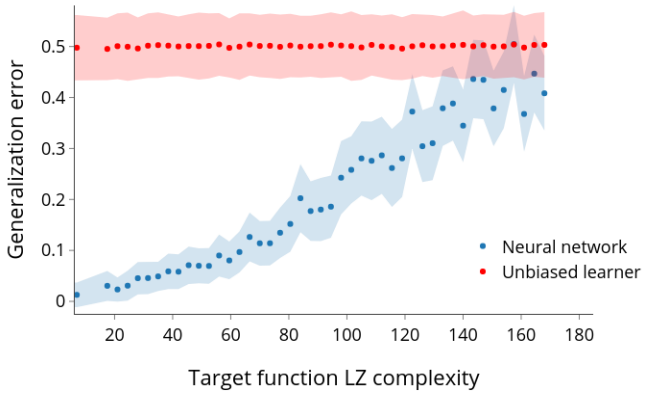




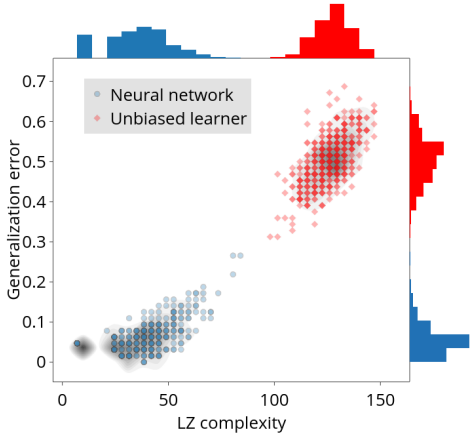
10^8 samples of parameters for (7,40,40,1) vanilla fully connected DNN system (FCN) with ReLU.

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

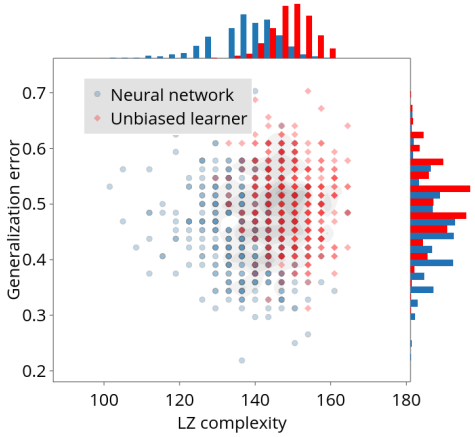
Does simplicity bias help generalisation?



DNN works much better than random learner



(a) Target function LZ complexity: 38.5

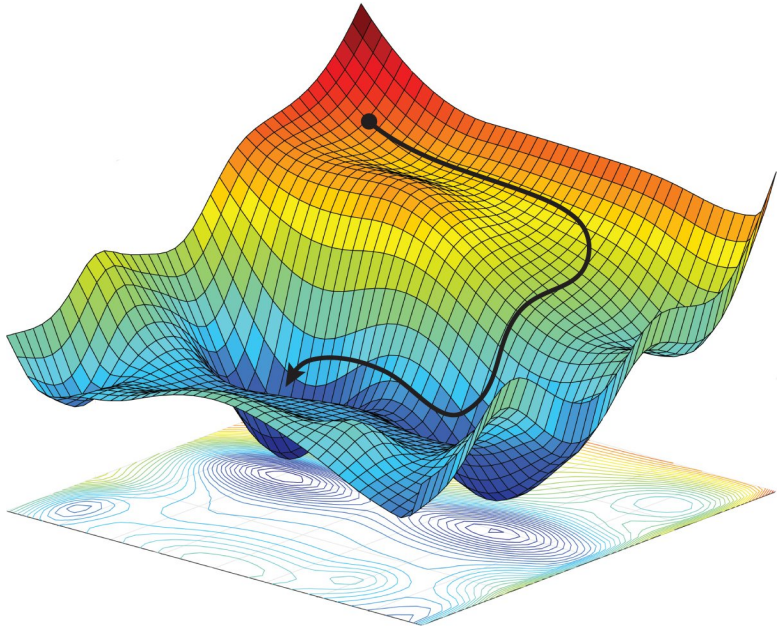


(b) Target function LZ complexity: 164.5

DNN works well on simple functions,
but less well on complex functions

Trained on 64 of 128 possible functions.

Problem; DNNs are not trained by randomly sampling parameters

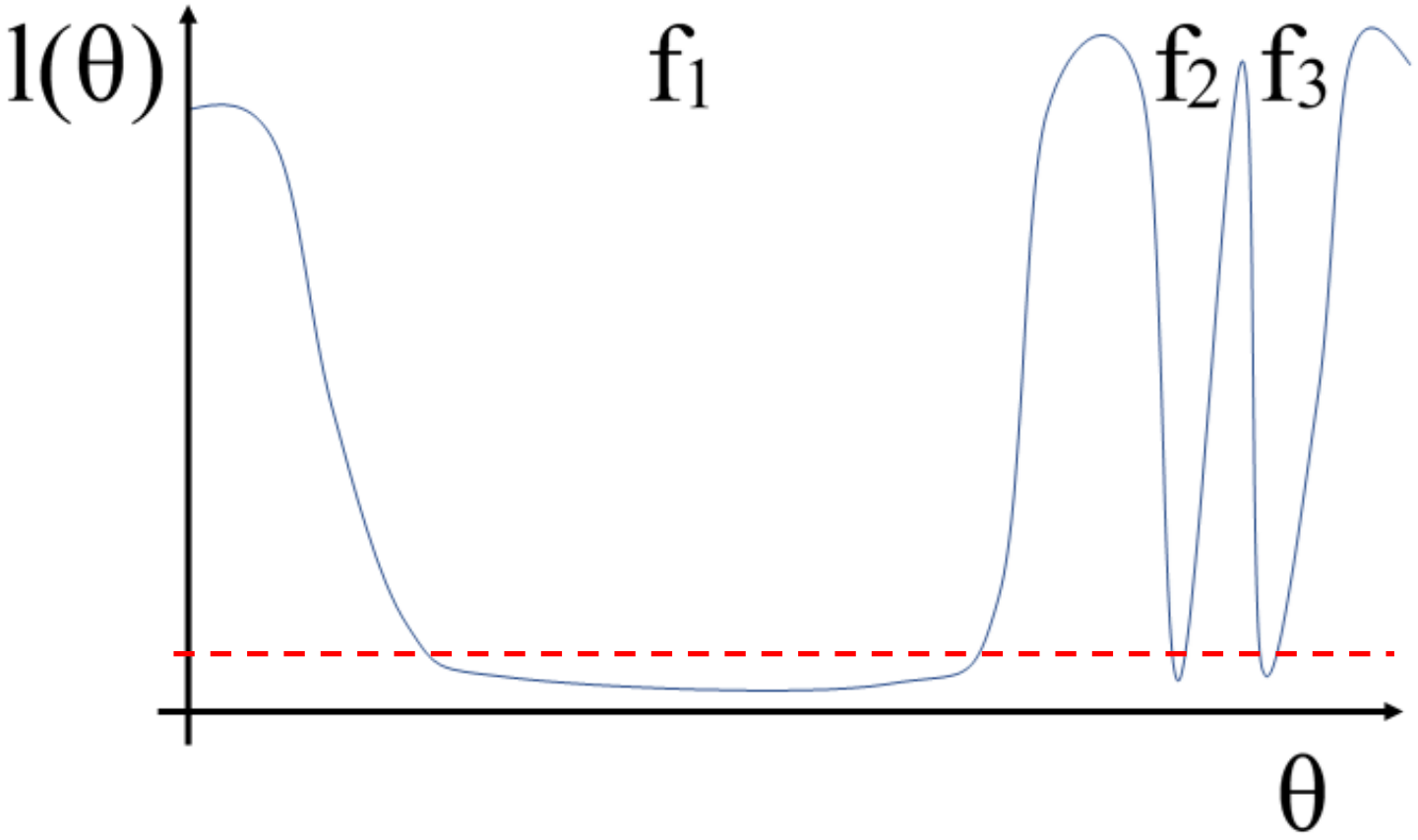


DNNs are trained using Stochastic gradient descent (SGD) on a loss function.

Problem; DNNs are not trained by randomly sampling parameters

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

Chris Mingard



Bayesian picture of bias for training set 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 $S = \{(x_i, y_i)\}_{i=1}^{m}$

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

Posterior follows from Bayes rule

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

Where the marginal likelihood or evidence is

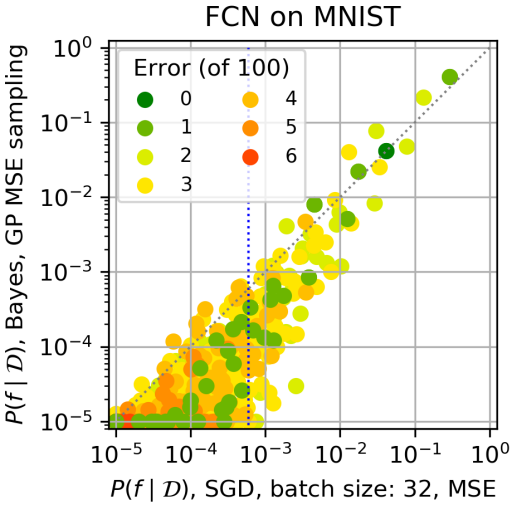
Functions that fit S

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

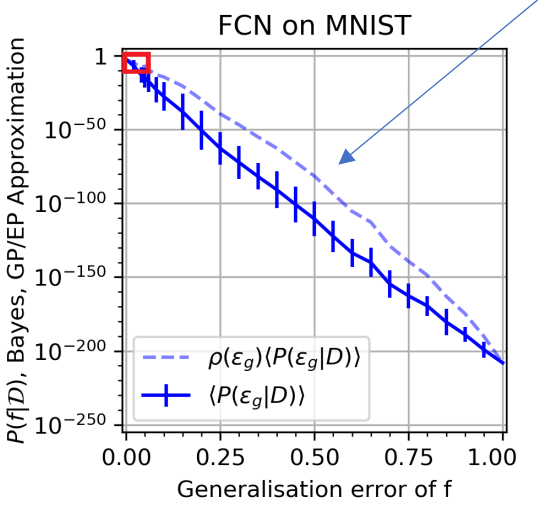
For fixed S, bias in P(f|S) translates over from bias in prior P(f)

SGD acts (almost) like a Bayesian sampler

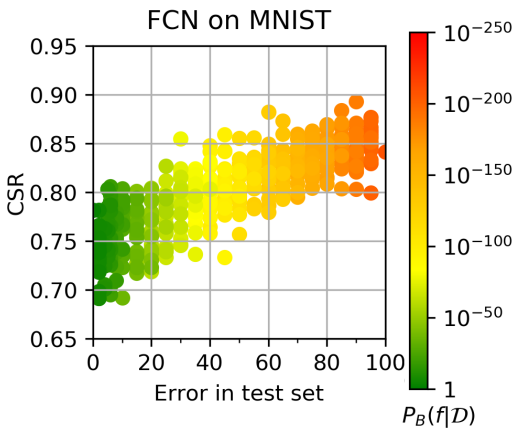
$\rho(\epsilon_G) = M!/((M-\epsilon_G)!\epsilon_G!)$



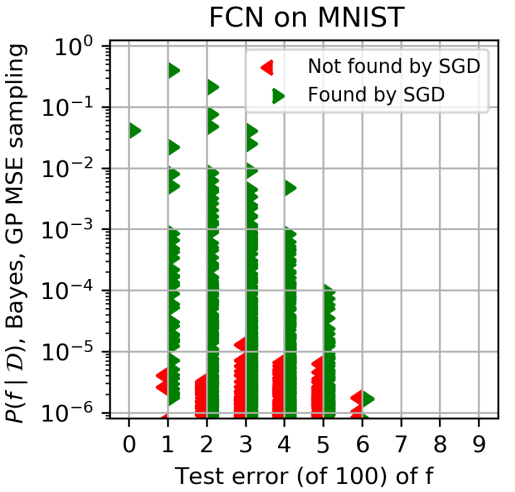
(a) $P_B(f|S)$ v.s.$P_{\text{SGD}}(f|S)$



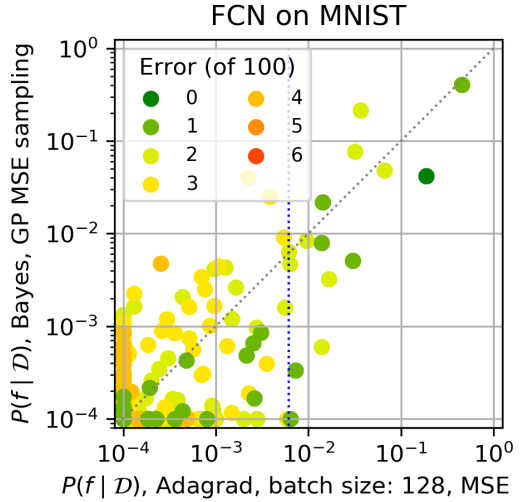
(b) $P_B(f|S)$ v.s. ϵ_G



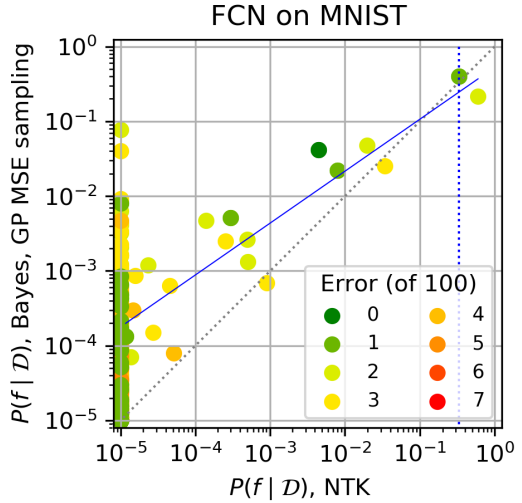
(c) CSR complexity v.s. ϵ_G for (b)



(d) f found by NNGP in (a).



(e) $P_B(f|S)$ v.s.$P_{Adagrad}(f|S)$

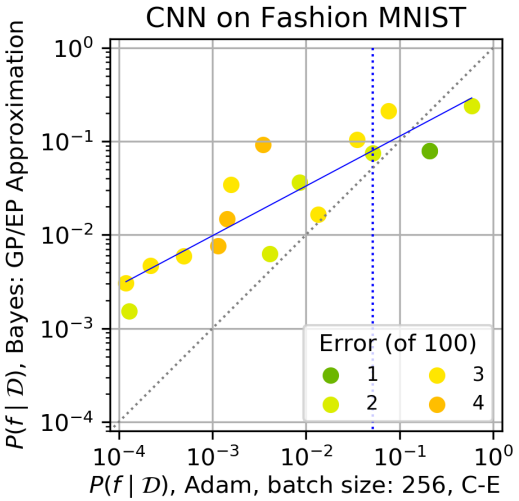


(f) $P_B(f|S)$ v.s.$P_{NTK}(f|S)$

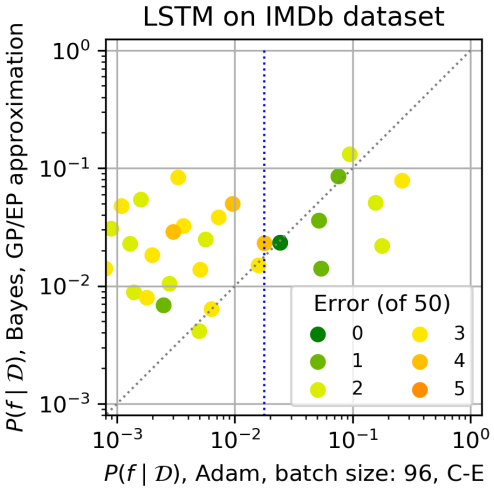
FCN on binarized MNIST – training set 10,000, test set 10,000 – $2^{100} = 10^{30}$ possible functions fit the test set.

SGD acts (almost) like a Bayesian sampler

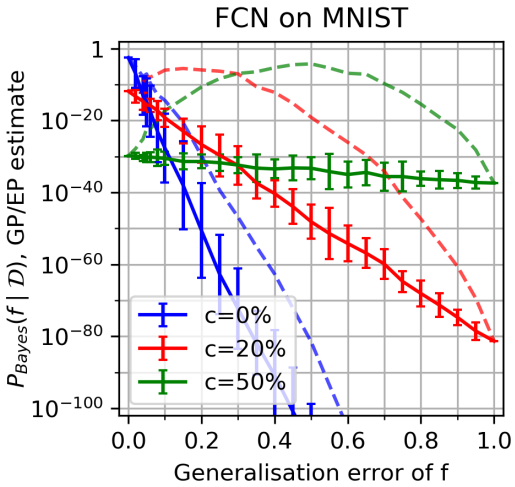
(d) f found by NNGP in (a).



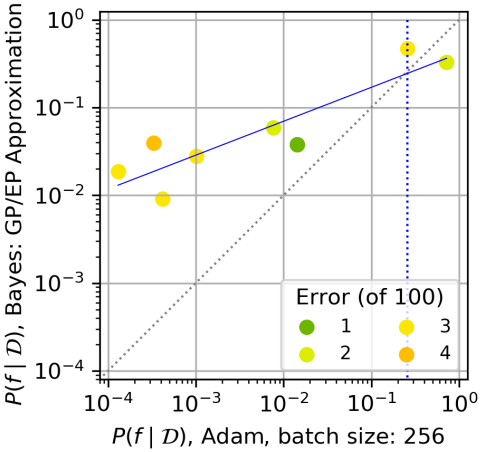
(e) $P_B(f|S)$ v.s. $P_{Adagrad}(f|S)$



(f) $P_B(f|S)$ v.s. $P_{NTK}(f|S)$



(g) $P_B(f|S)$ v.s. $P_{\text{adam}}(f|S)$



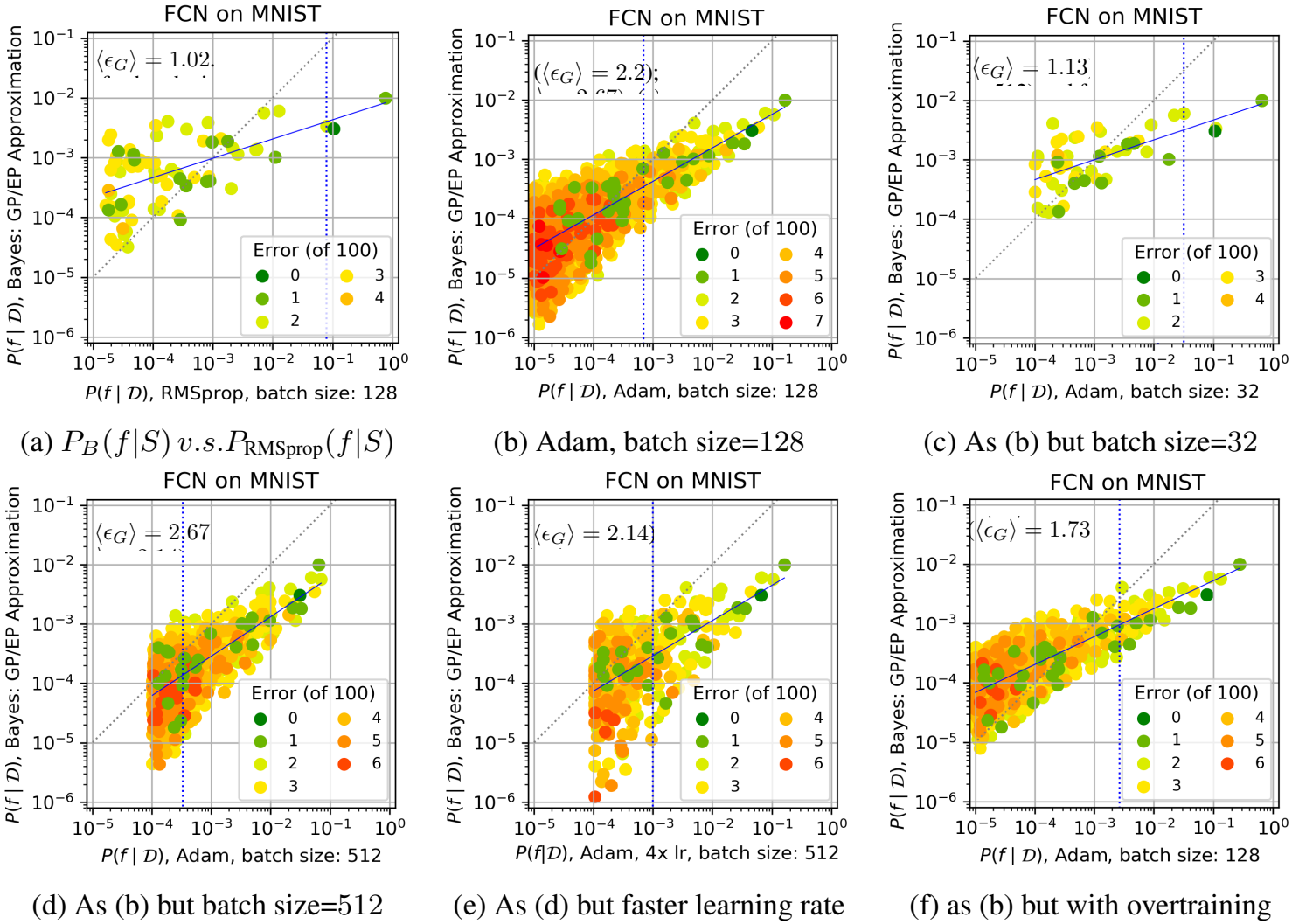
(b) vanilla CNN

(h) $P_B(f|S)$ v.s. $P_{\text{adam}}(f|S)$

(i) Corrupted data

Other data sets and architectures

SGD acts (almost) like a Bayesian sampler

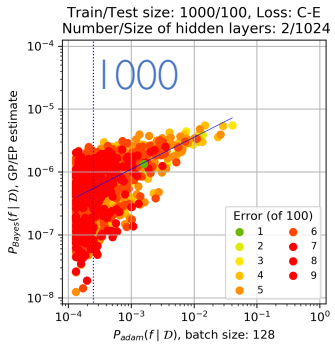


(a) $P_B(f|S)$ $v.s.$ $P_{\text{RMSprop}}(f|S)$ (b) Adam, batch size=128 (c) As (b) but batch size=32

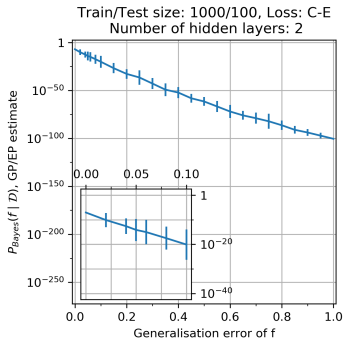
(d) As (b) but batch size=512 (e) As (d) but faster learning rate (f) as (b) but with overtraining

Figure 2: **Effects of hyperparameter changes:** All examples are for a 2-layer 1024 node FCN trained on MNIST with CE loss. (a) shows RMSprop which for batch size = 128 achieves an error on

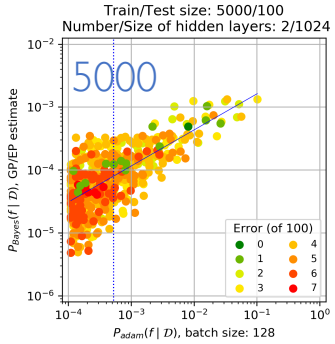
Training set size dependence



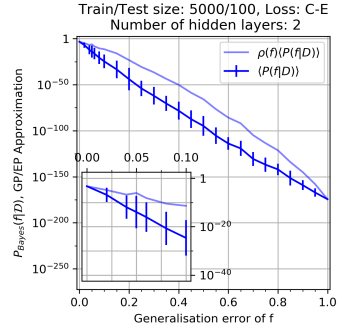
(a) Experiment 1. 1000 training examples. $\langle \epsilon_G \rangle = 6.65\%$



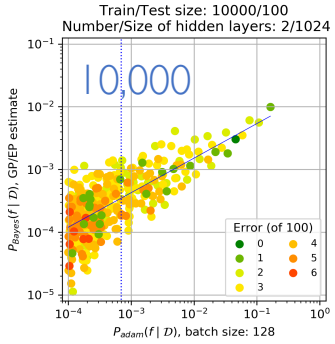
(b) Experiment 3. 1000 training examples

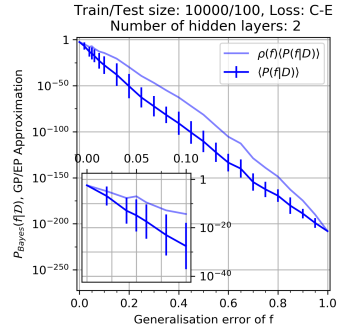


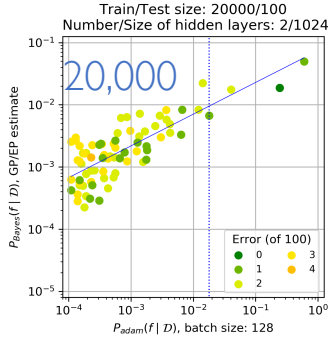
(c) Experiment 1. 5000 training examples. $\langle \epsilon_G \rangle = 3.33\%$



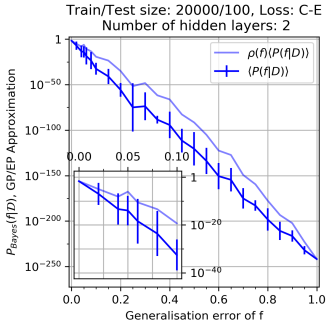
(d) Experiment 3. 5000 training examples





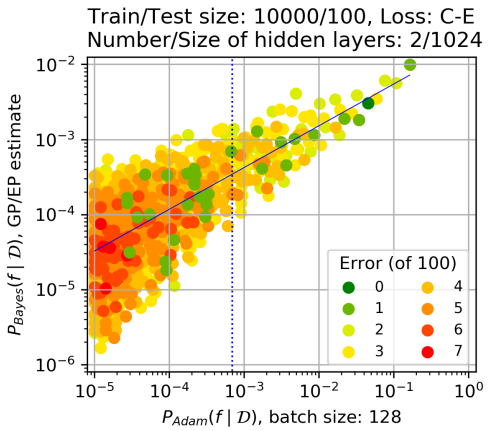


(g) Experiment 1. 20000 training examples. $\langle \epsilon_G \rangle = 0.89\%$

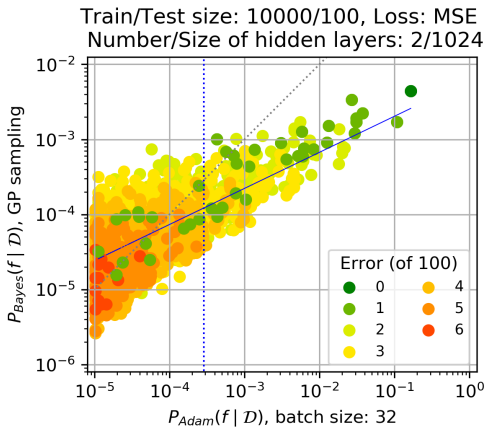


(h) Experiment 3. 20000 training examples

Test set dependence



(a) Generalisation error: 2.20%.



(b) Generalisation error: 2.62%

Upshot so far:

- The probabilities that functions are found by SGD is remarkably close to a Bayesian sampler.
- Exact reason not understood, but heuristically this must stem from large bias. (for unbiased systems the correlations are very weak)
- Nice, because we can use Bayesian picture to understand generalisation.
- Also, we observe very strong bias (consistent with simplicity bias)

Two questions in generalisation

1) Why do DNNs generalise at all in the overparameterised regime?
2) Given DNNs that generalise, can we further fine-tune the hyperparameters to improve generalisation?

Two questions in generalisation

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

Can we derive generalisation bounds ?

PAC-Bayes bounds

Guillermo
Valle Perez

Corollary 1. *(Realizable PAC-Bayes theorem (for Bayesian classifier)) Under the same setting as in Theorem 1, with the extra assumption that $\mathcal{D}$ is realizable, we have:*

$$-\ln\left(1-\epsilon(Q^*)\right) \leq \frac{\ln\frac{1}{P(U)} + \ln\left(\frac{2m}{\delta}\right)}{m-1}$$

where $Q^*(c) = \frac{P(c)}{\sum_{c\in U} P(c)}$, U is the set of concepts in $\mathcal{H}$ consistent with the sample S, and where $P(U) = \sum_{c\in U} P(c)$

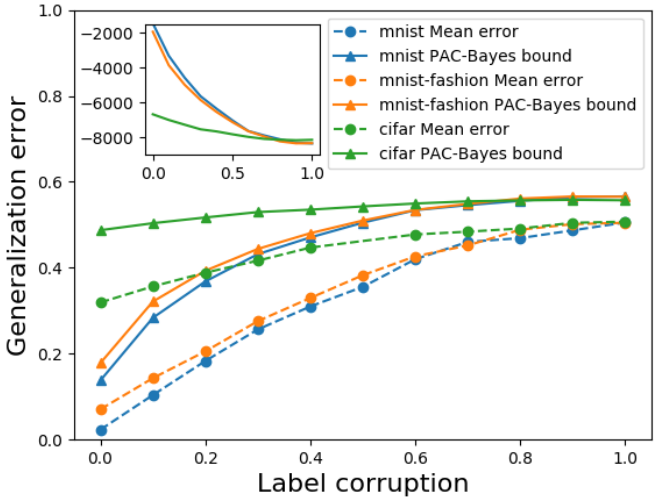
Adapted from David McAllester

P(U) – Marginal likelihood we saw before.

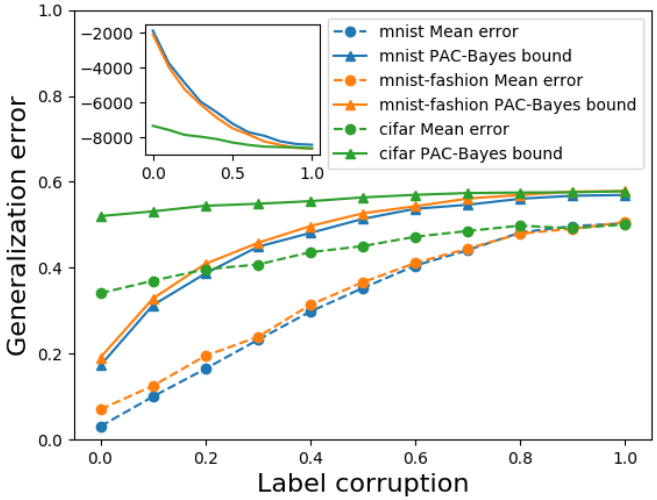
G. Valle Perez, C. Camargo and A.A. Louis, arxiv:1805.08522 – ICLR 2019

Tight PAC-Bayes bounds:

Guillermo Valle Perez



(a) for a 4 hidden layers convolutional network



(b) for a 1 hidden layer fully connected network

Corollary 1. *(Realizable PAC-Bayes theorem (for Bayesian classifier)) Under the same setting as in Theorem 1, with the extra assumption that $\mathcal{D}$ is realizable, we have:*

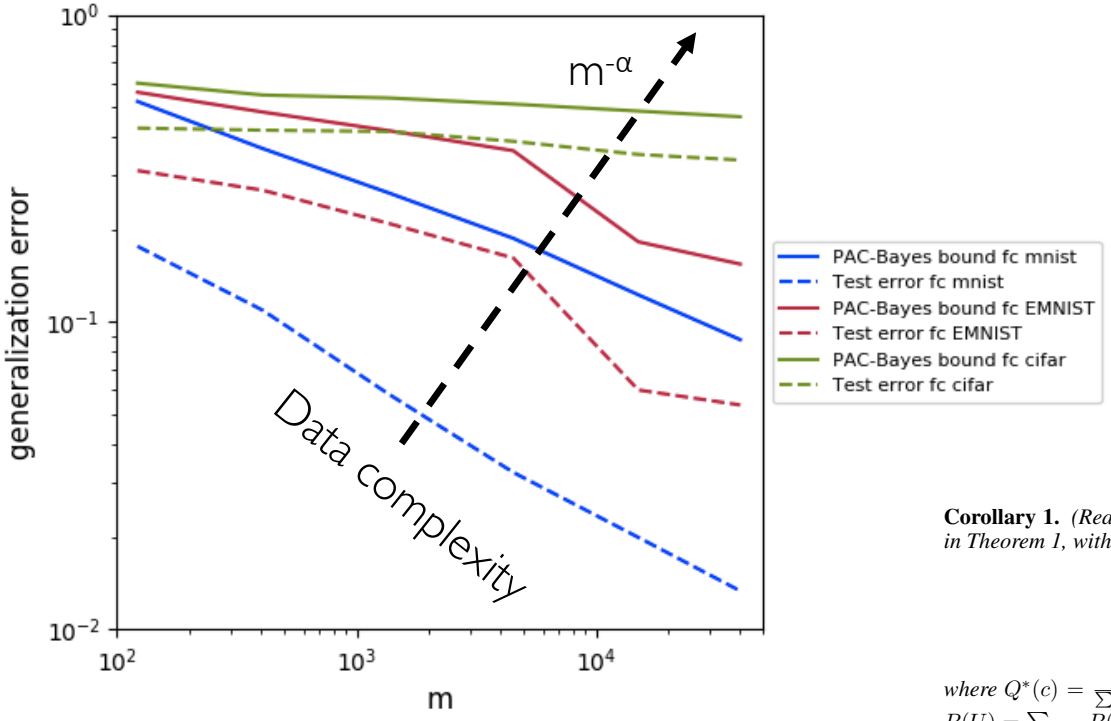
$$-\ln\left(1-\epsilon(Q^*)\right) \leq \frac{\ln\frac{1}{P(U)} + \ln\left(\frac{2m}{\delta}\right)}{m-1}$$

where $Q^*(c) = \frac{P(c)}{\sum_{c\in U} P(c)}$, *$U$ is the set of concepts in $\mathcal{H}$ consistent with the sample S, and where* $P(U) = \sum_{c\in U} P(c)$

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

Tight PAC-Bayes bounds for scaling of error with training set size m (learning curves)

Guillermo
Valle Perez



Observed: error ~ $m^{-\alpha}$

1) α decreases with data complexity (bad news for machine learning)
2) α appears independent of algorithm
3) We can reproduce this scaling with PAC-Bayes theory approach we have derived.

But, WHY this scaling?

Corollary 1. *(Realizable PAC-Bayes theorem (for Bayesian classifier)) Under the same setting as in Theorem 1, with the extra assumption that $\mathcal{D}$ is realizable, we have:*

$$-\ln\left(1-\epsilon(Q^*)\right) \leq \frac{\ln\frac{1}{P(U)} + \ln\left(\frac{2m}{\delta}\right)}{m-1}$$

where $Q^*(c) = \frac{P(c)}{\sum_{c\in U} P(c)}$, U is the set of concepts in $\mathcal{H}$ consistent with the sample S, and where $P(U) = \sum_{c\in U} P(c)$

Bayesian predictions

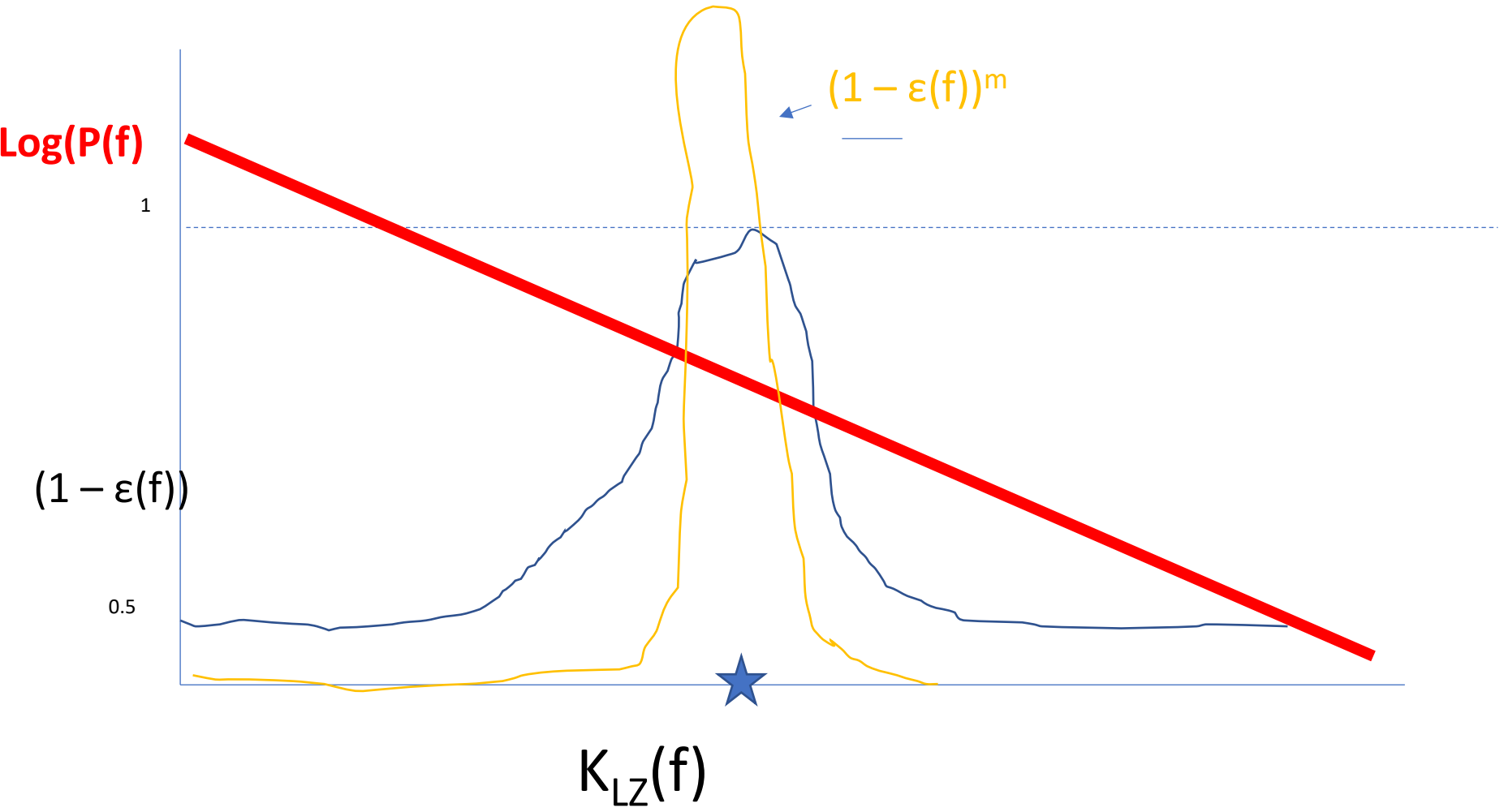
Averaged over datasets

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

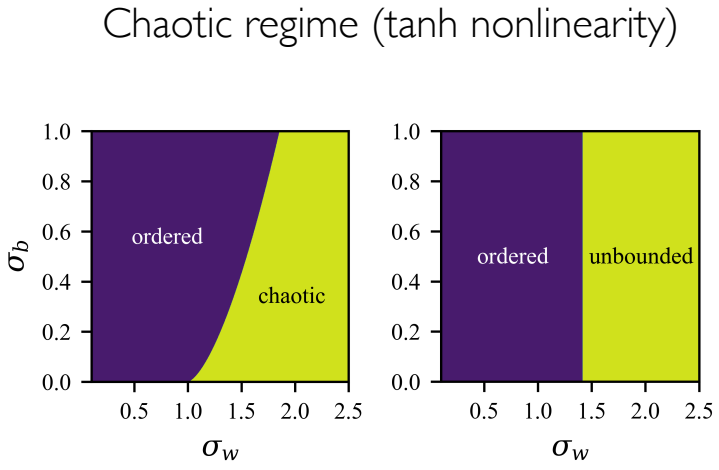
Training factor tells you how much the function probability is affected by training

$$t_F(f,m) = \frac{\left(1-\epsilon(f)\right)^m}{\langle P(\mathcal{D}^m)\rangle} \approx \frac{e^{-m\epsilon(f)}}{\langle P(\mathcal{D}^m)\rangle} = \frac{e^{-m\epsilon_G}}{\langle P(\mathcal{D}^m)\rangle} e^{-m(\epsilon(f)-\epsilon_G)}$$

Error distribution on data

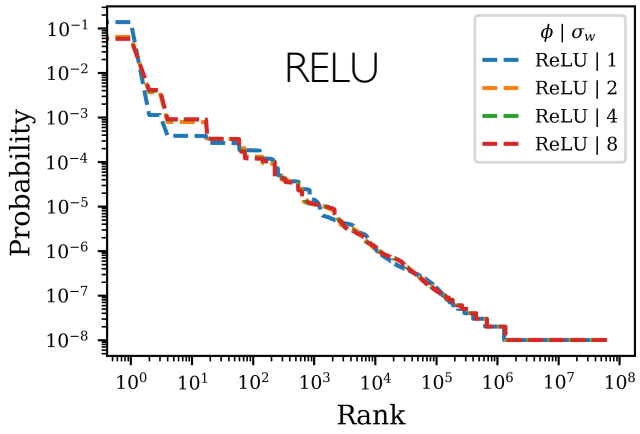


$$\langle P(f|\mathcal{D}^m)\rangle = P(f)\left\langle \frac{P(\mathcal{D}_i^m|f)}{P(\mathcal{D}_i^m)}\right\rangle_{\mathcal{D}_i^m} \approx \frac{P(f)\,(1-\epsilon(f))^m}{\langle P(\mathcal{D}^m)\rangle} = t_F(f,m)P(f) \approx \frac{P(f)e^{-m\epsilon(f)}}{\langle P(\mathcal{D}^m)\rangle}$$

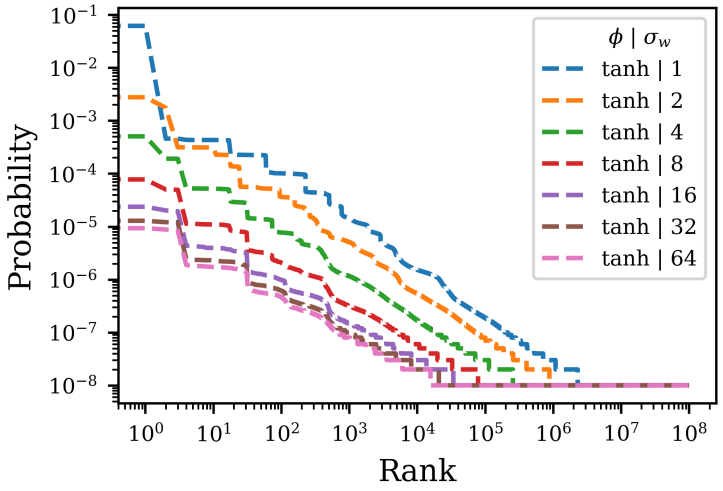


(a) *Tanh* **(b)** *ReLU*

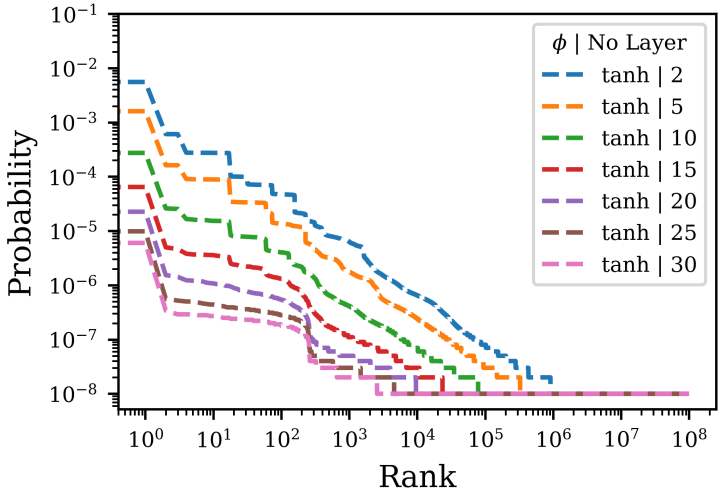
Figure 3: *Mean field phase diagrams for tanh and ReLU activation functions showing various phase regimes as a function of σ_w and σ_b.*



(a) *Increasing σ_w has no significant effect on the bias in a simple ReLU DNN. $\sigma_b = 0.2$. The DNN has 2 hidden layers each containing 40 ReLU neurons.*



(a) *Increasing σ_w decreases the bias in a simple DNN. $\sigma_b = 0.2$. The DNN has 2 hidden layers of 40 tanh neurons.*



(b) *In the chaotic regime, increasing the number of layers decreases the bias in a simple DNN. $\sigma_b = 0.2$ and $\sigma_w = 2$. Each hidden layer contains 40 tanh neurons.*

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

Increasing chaos decreases bias

Henry Rees

J. Empirical probability versus LZ complexity plots

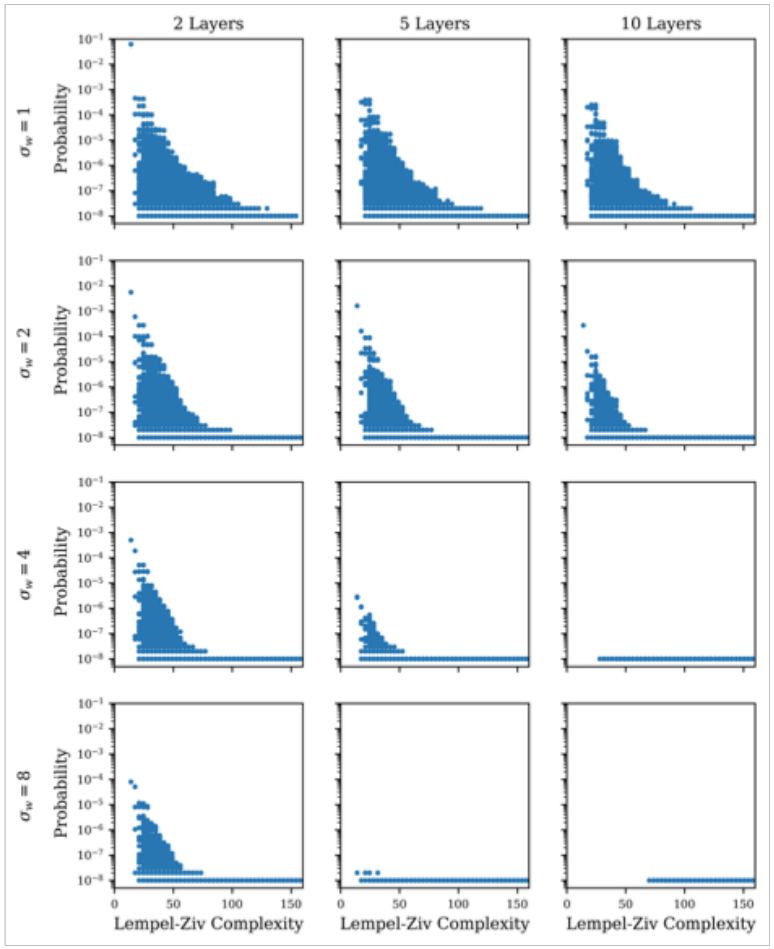


Figure 17: *Empirical probability of individual functions versus their LZ complexity for networks initialised with various σ_w and numbers of layers. Despite suffering from finite-size effects, points with a probability of 10^{-8} are not removed since in plots ($\sigma_w = 4, d = 10$) and ($\sigma_w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

K. Generalisation versus LZ complexity plots

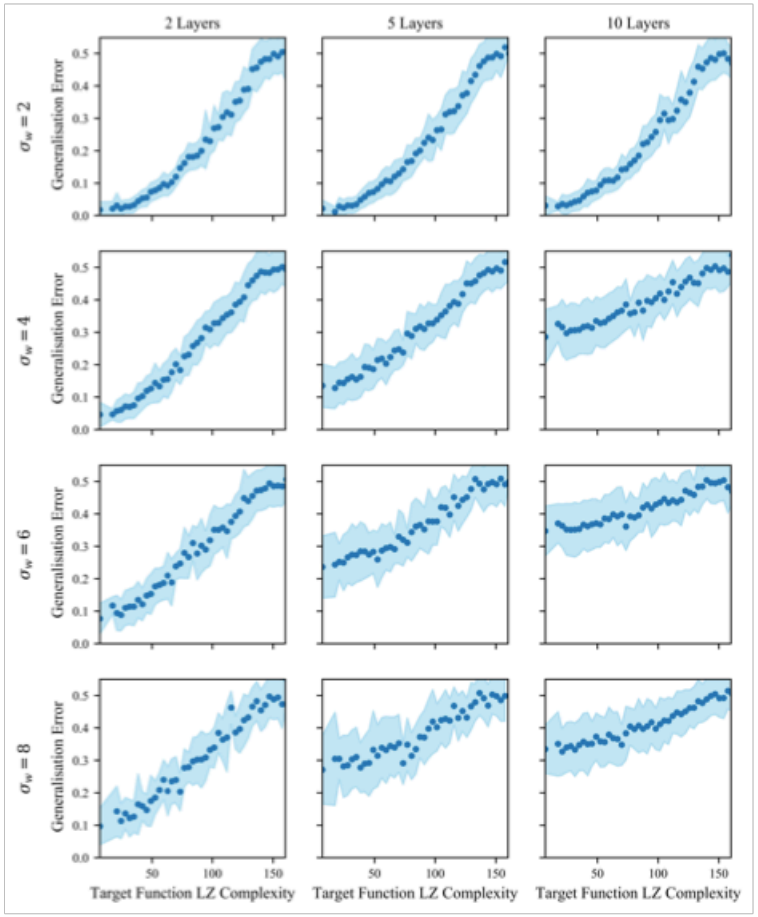


Figure 19: *Effect of increasing σ_w and network depth on the generalising ability of Boolean networks trained on functions $\{0,1\}^7 \to \{0,1\}$. Details are the same as Figure 7.*

Bayesian Approach

Henry Rees

$$\langle P(f|\mathcal{D}_m)\rangle = P(f)\left\langle \frac{P(\mathcal{D}_m|f)}{P(\mathcal{D}_m)} \right\rangle \approx \frac{P(f)\,(1-\epsilon(f))^m}{\langle P(\mathcal{D}_m)\rangle}$$

Target Function: 00000101000100000000100000010000000110000…

Test function: 00000101000000000000100010010000000110000…

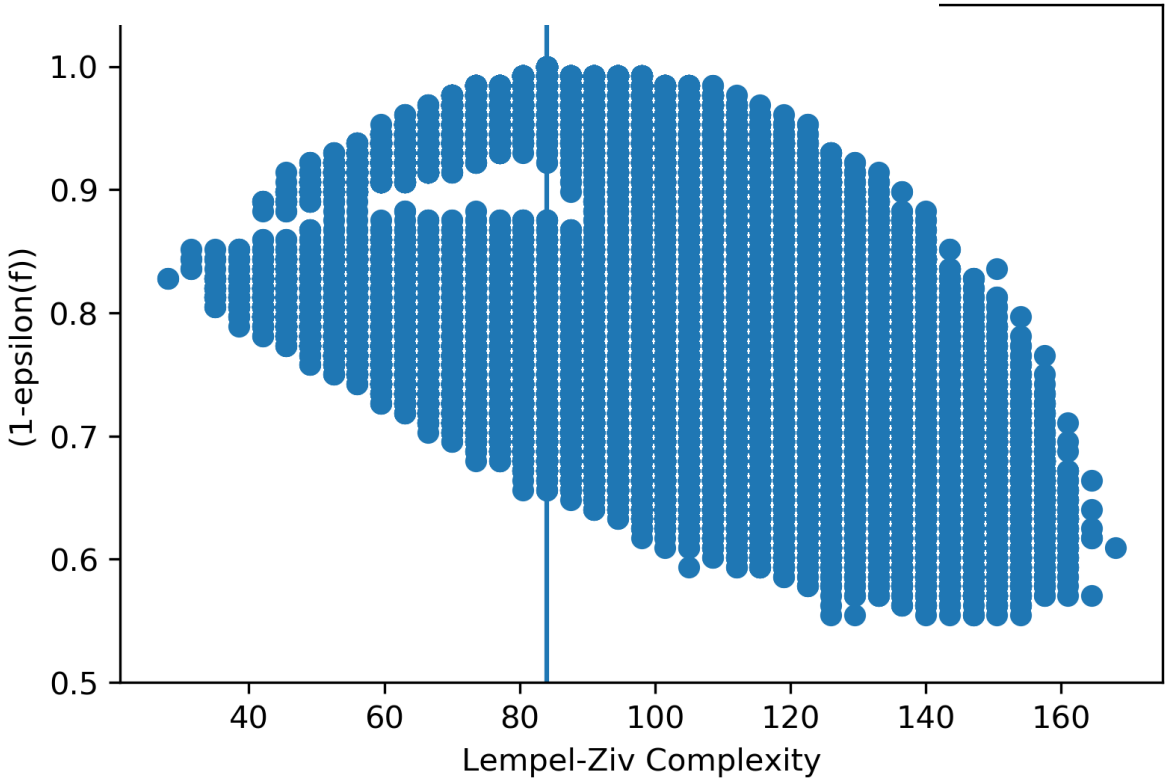
$(1 - \varepsilon(f)) = 1 - 2/128 = 0.984…$

Bayesian Approach -

Henry Rees

$$\langle P(f|\mathcal{D}_m)\rangle = P(f)\left\langle \frac{P(\mathcal{D}_m|f)}{P(\mathcal{D}_m)} \right\rangle \approx \frac{P(f)\,(1-\epsilon(f))^m}{\langle P(\mathcal{D}_m)\rangle}$$

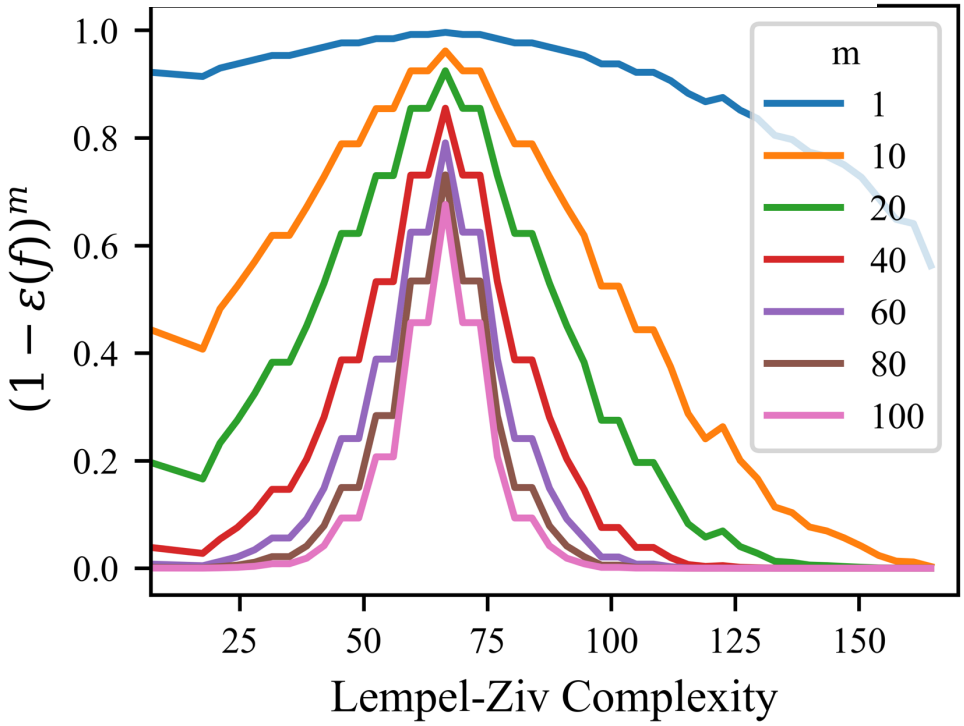
$(1-\varepsilon(f))$



Bayesian Approach

$$\langle P(f|\mathcal{D}_m)\rangle = P(f)\left\langle \frac{P(\mathcal{D}_m|f)}{P(\mathcal{D}_m)} \right\rangle \approx \frac{P(f)\,(1-\epsilon(f))^m}{\langle P(\mathcal{D}_m)\rangle}$$

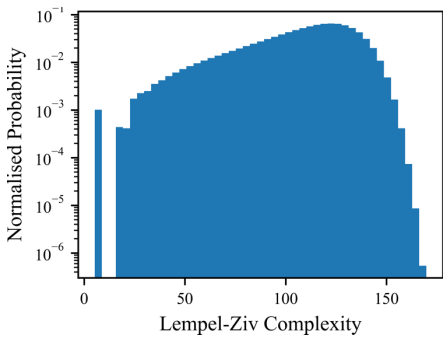
$$(1-\langle \varepsilon(f)\rangle)^m$$

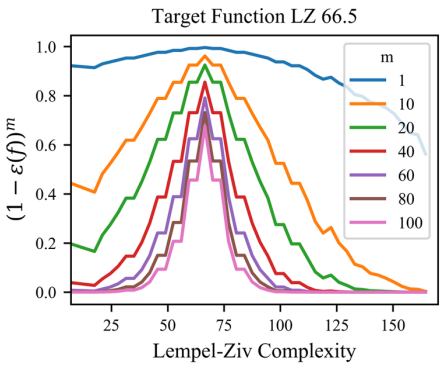


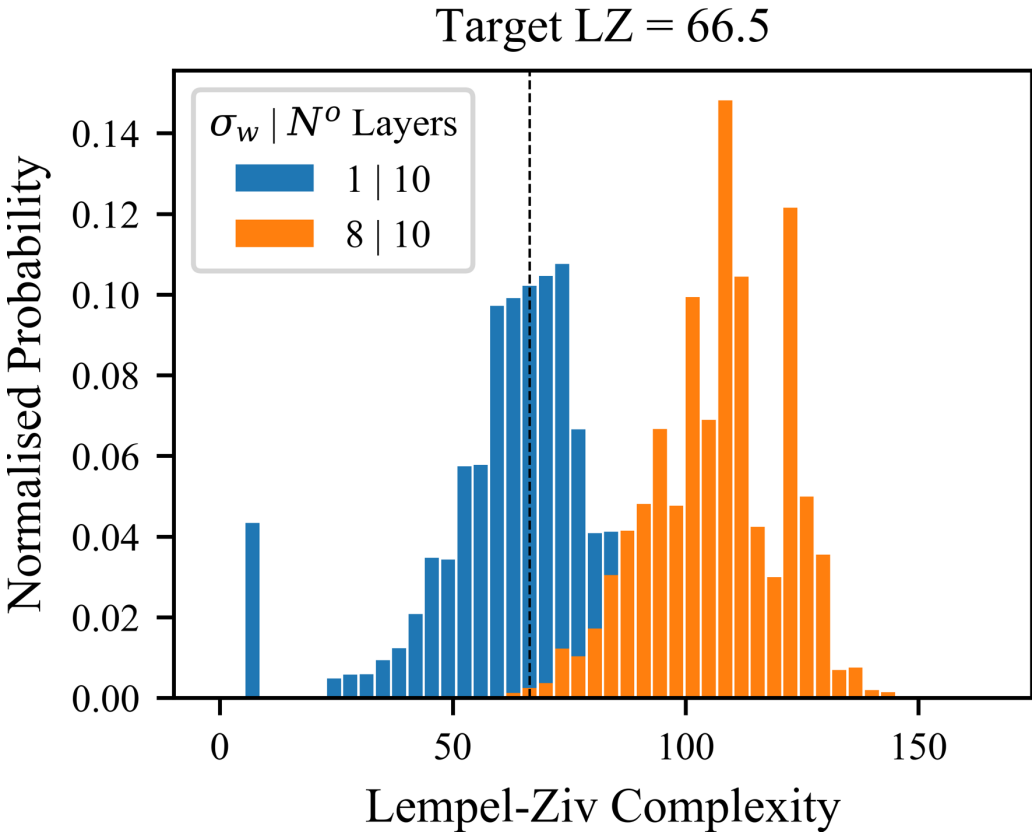
Bayesian Approach

Henry Rees

$$P(K|D^{64})$$

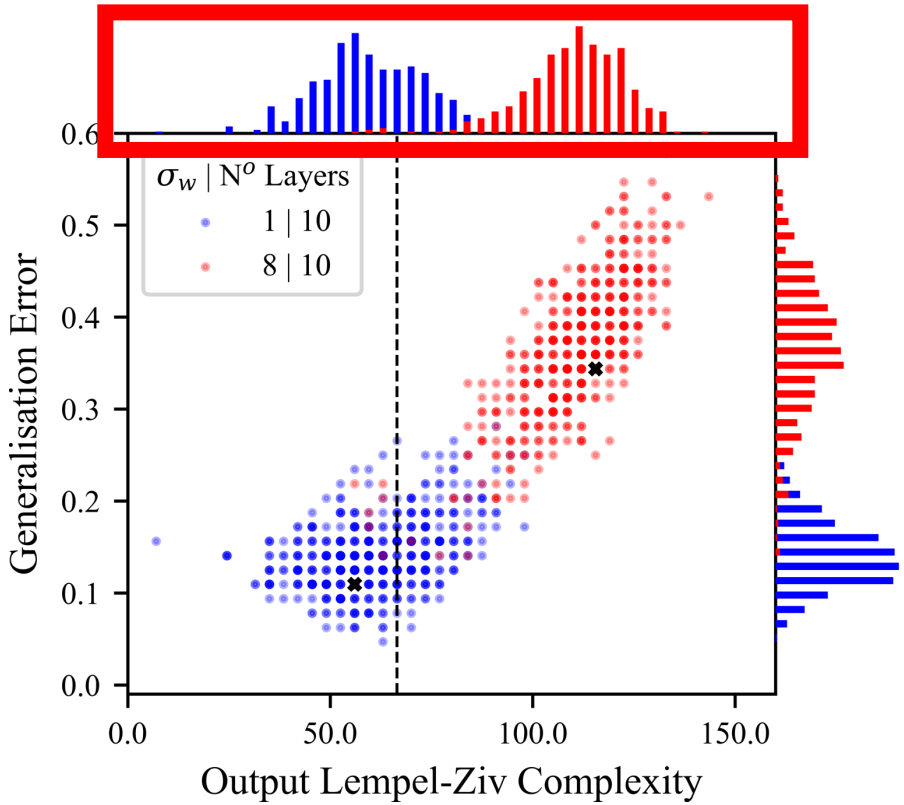




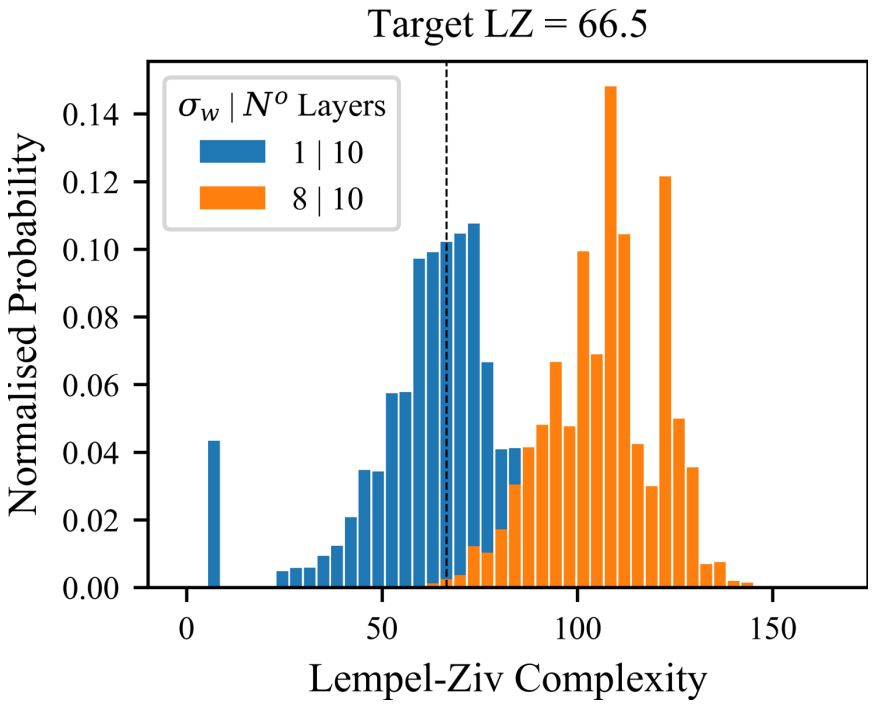


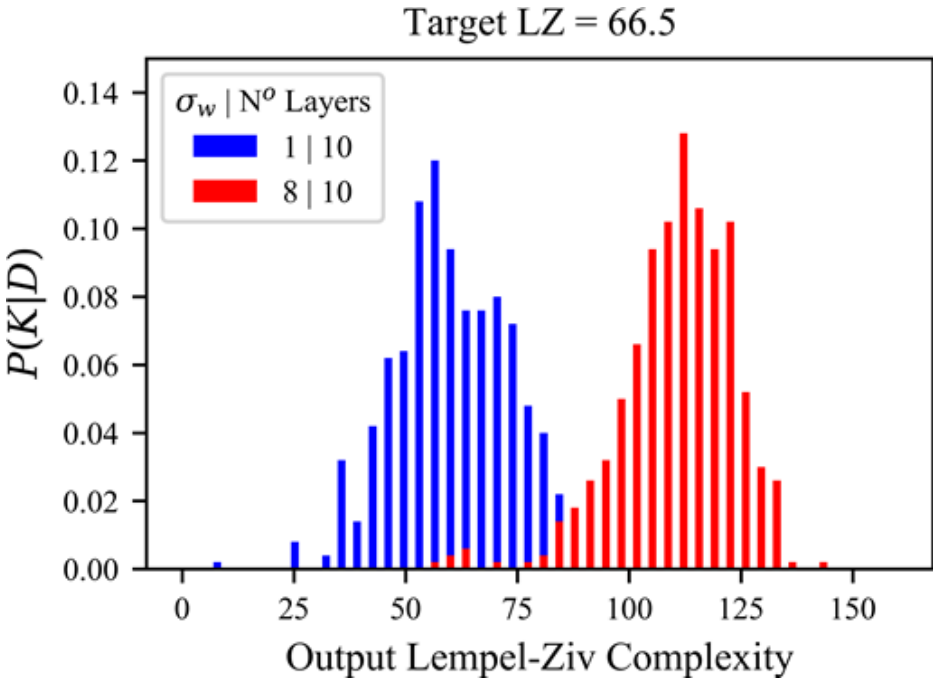
Experiments with SGD

- Train on a Specific Target Function with training set of size 64
- Calculate the complexity and Generalisation Error of function implemented by model
- Repeat many times – randomising the examples in the training set each time.



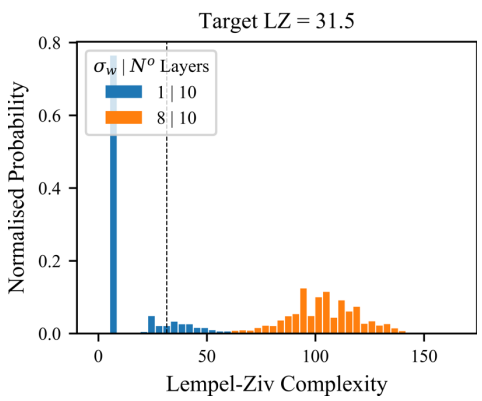
Comparison of SGD to Bayesian

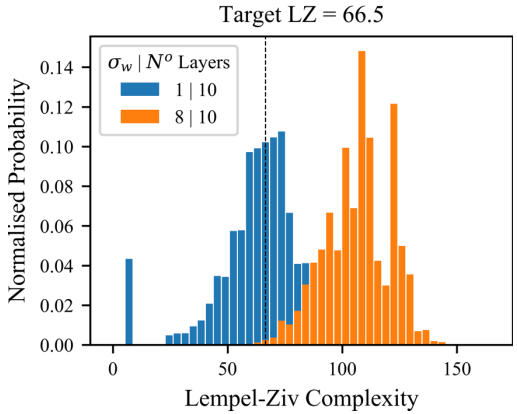


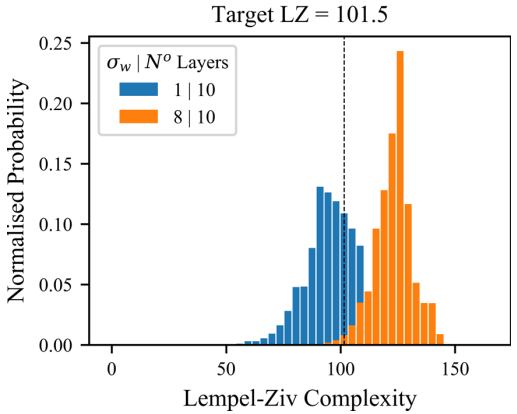


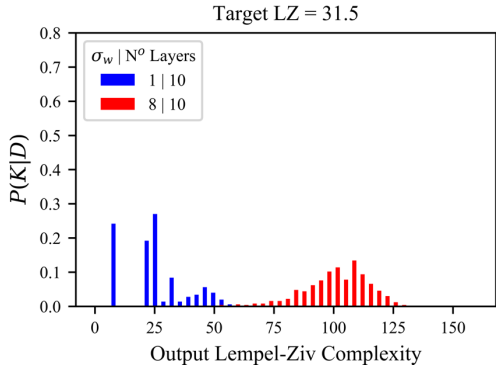
Henry Rees

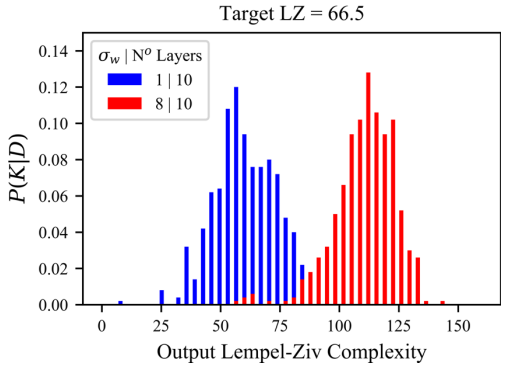
Comparison of SGD to Bayesian

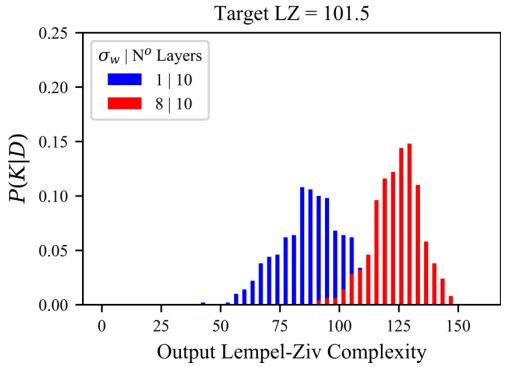






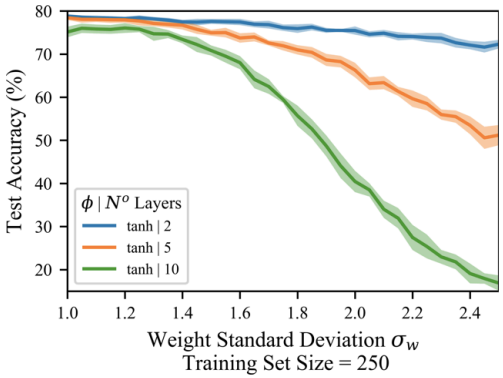


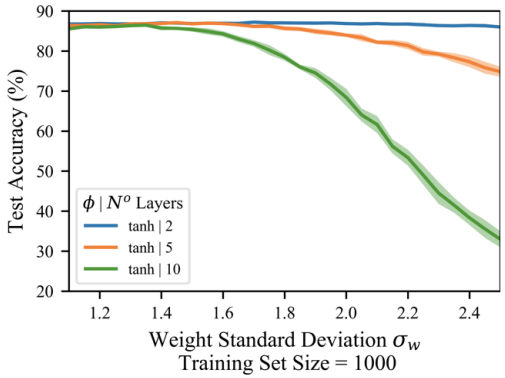




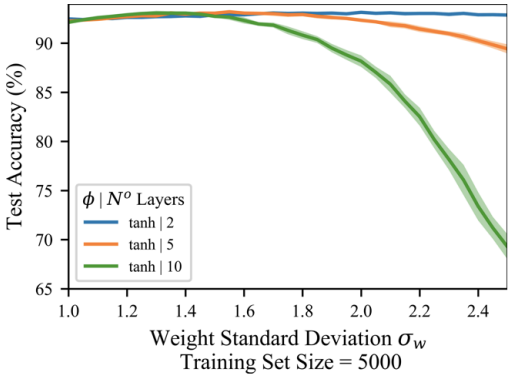
Bayesian Approach

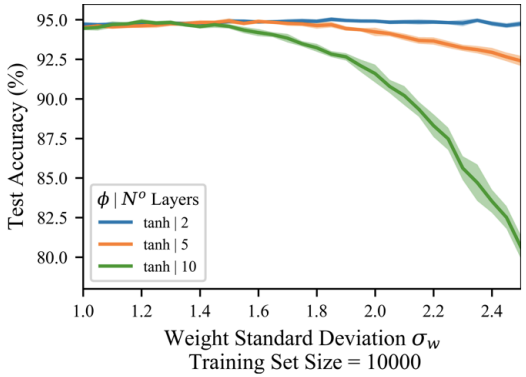
Henry Rees

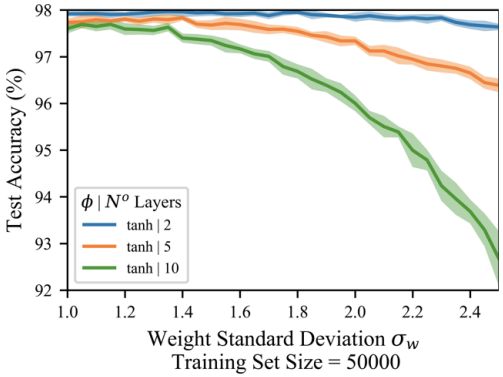




MNIST







THANK YOU

Conclusions

- Deep learning may work because they have a natural bias towards simple functions (Occam's razor)
- We show evidence that SGD behaves like a Bayesian optimiser
- PAC-Bayes provides tight bounds
- A Bayesian approach can help explain how generalisation works
- Work done by Guillermo Valle Perez, Chris Mingard, Joar Skalse, Henry Rees and more