Compression, Nonuniform Learnability, and PAC-Bayes

From Shannon's entropy to the implicit bias of deep networks

Roadmap

1. Compression, prediction, and entropy
2. Nonuniform learnability, MDL, and PAC-Bayes (Shalev-Shwartz & Ben-David)
3. Algorithmic probability and the implicit bias of DNNs (Valle-Pérez et al., 2018)

Compression means looking like noise

Claim

An optimally compressed message is statistically indistinguishable from random noise.

- Everyday idea: compression = smaller file.
- We will make this precise, and see it is really a claim about probability.

If it doesn't look like noise...

- Suppose a "compressed" file still has visible structure or repeated patterns.
- Structure means predictability: some parts of the file let us guess other parts.
- Predictable content can be assigned even shorter codes $\Rightarrow$ the file can be compressed further.
- So: still-structured $\Rightarrow$ not yet optimal. Optimal $\Rightarrow$ no structure left $\Rightarrow$ looks like noise.

Compression and prediction are two faces of the same coin

- A good **predictor** of "what comes next" lets you assign very short codes to the outcomes it expects.
- A good **compressor** has, implicitly, learned which outcomes are likely and which are not.
- These are the same skill, viewed from two directions.

predicting well $\iff$ compressing well

This equivalence is the thread that connects entropy, MDL, and PAC-Bayes.

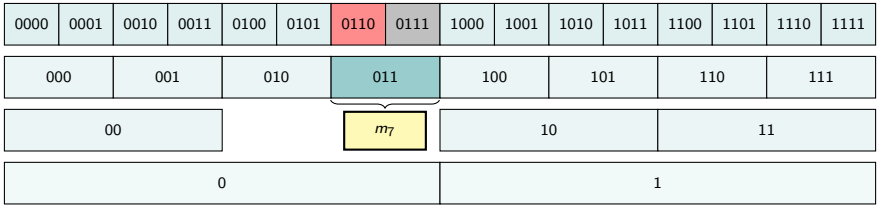
Why random noise is incompressible

Setup

Every message of length n is equally likely: $P(m_i) = 1/2^n$ for $i = 0, \ldots, 2^n - 1$.

- Codes must be **prefix-free**: no codeword may be a prefix of another, so a decoder can read symbols one at a time with no delimiters.
- Suppose we shorten the code for one message m_i by a bit. Its new, shorter code becomes the shared prefix of *two* of the old length-n codewords.
- Prefix-freeness then forbids those two messages from keeping their old codes — they must be moved elsewhere in the tree, to *longer* codewords.
- Since every message is equally likely, this relocation costs more (in expectation) than the single bit we saved.

Shortening one code lengthens others



Merging 0110/0111 into a single 3-bit codeword m_7 saves 1 bit — but m_5 and m_6 no longer fit and must be relocated to 5-bit codes. Net expected length: $-1 + 1 + 1 = +1$ bit. Shortening one code always lengthens others.

Optimal code length: $-\log p(x)$

- We just saw: looking like random noise $\equiv$ being optimally compressed.
- For a "random" message drawn uniformly from 2^n possibilities, $p(x) = 1/2^n$, and its optimal code length is $n = -\log_2 p(x)$.
- This generalizes: for *any* distribution p, the optimal code length is

$$\ell^\star(x) = -\log_2 p(x).$$

  Frequent symbols get short codes, rare symbols get long codes.
- **Entropy** $H(p) = \mathbb{E}_{x\sim p}[-\log p(x)]$: the average number of bits per symbol under the true distribution.
- **Cross-entropy** $H(p, q) = \mathbb{E}_{x\sim p}[-\log q(x)]$: average bits per symbol if we compress p-distributed data using a code built for q. This is exactly the loss we minimize when training predictive models — *training is compression*.

The fundamental limit

Shannon's source coding theorem: no lossless code can achieve an expected length below $H(p)$. Entropy is the hard floor that compression asymptotically approaches but never beats.

Shannon's search for the entropy of English

- Goal: estimate the average information content (bits/character) of English text.
- First attempt: n-gram statistics from books.
- Problem: brittle to the choice of n — breaks down exactly when longer context is needed most.
- Shannon realized he already had a much better language model: the human brain.

The dash experiment

- Shannon asked his wife to guess the next character from context; correctly guessed characters were replaced with a dash.

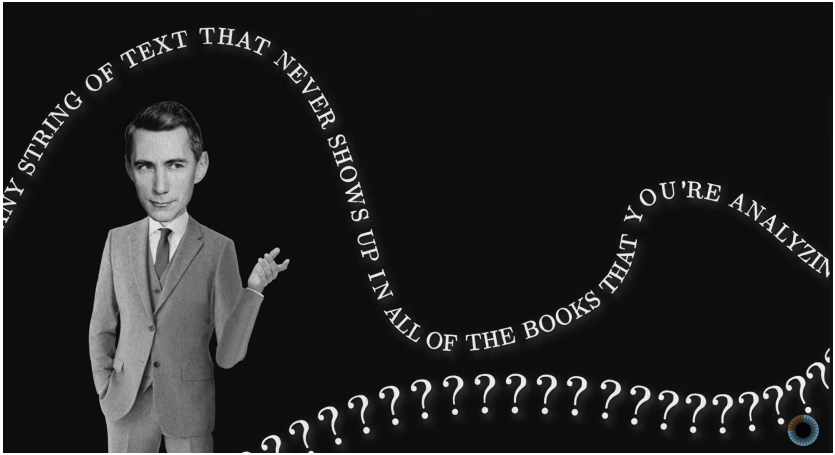
Example

Original: `THE STUDENT IS READING A BOOK`
Dashed: `TH- ST-D-NT -S R-AD-NG A B--K`

- If we replicated his wife — someone with the same predictive model — the dashed sequence would already be enough for her to reconstruct, and recognize, the entire original passage.
- So the two sequences carry *the same information*.

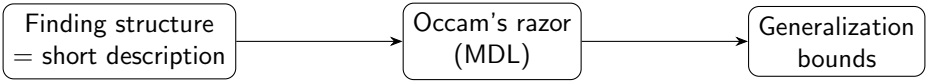
Predictability $\equiv$ Compression.

For more intuition and visualizations



Check: https://www.youtube.com/watch?v=I6DKRf-fAAM

Bridge to learning theory



A learning algorithm that compresses the data has found a short description of it — this is exactly what Occam's razor and MDL will formalize.

Why go beyond agnostic PAC learning

- Uniform convergence needs finite VC-dimension and gives *one* sample-complexity bound for the whole class.
- Puzzle: deep nets appear to have effectively unbounded VC-dimension, yet generalize well.
- Idea: let sample complexity depend on *which* hypothesis h we land on — not every $h \in \mathcal{H}$ is equally "probable" a priori.

Nonuniform learnability

Definition

$\mathcal{H}$ is nonuniformly learnable if for every $h \in \mathcal{H}$ and ϵ, δ there is a sample size $m_{\mathcal{H}}(\epsilon, \delta; h)$, depending on h, above which an ERM-style algorithm is (ϵ, δ)-competitive with h.

- Strictly weaker than agnostic PAC learnability (e.g. a countable union of finite classes).
- **Characterization:** $\mathcal{H}$ is nonuniformly learnable $\iff$ $\mathcal{H}$ is a countable union of uniformly convergent classes.

Weighting the pieces of $\mathcal{H}$

Write $\mathcal{H} = \bigcup_{n \in \mathbb{N}} \mathcal{H}_n$, each $\mathcal{H}_n$ uniformly convergent with sample complexity $m^{UC}_{\mathcal{H}_n}(\epsilon, \delta)$. Define

$$\epsilon_n(m, \delta) = \min\{\epsilon \in (0,1) : m^{UC}_{\mathcal{H}_n}(\epsilon, \delta) \leq m\}.$$

So for a fixed budget m, with probability $\geq 1 - \delta$ over $S \sim D^m$:

$$\forall h \in \mathcal{H}_n, \quad |L_D(h) - L_S(h)| \leq \epsilon_n(m, \delta).$$

- Let $w : \mathbb{N} \to [0,1]$ with $\sum_n w(n) \leq 1$ — a **weight function** reflecting how much we favor each $\mathcal{H}_n$ a priori.
- Equal weights $\Rightarrow$ no preference; larger $w(n) \Rightarrow$ stronger prior belief that the target lives in $\mathcal{H}_n$.

The SRM bound

Theorem (structural risk minimization bound)

For every δ and D, with probability $\geq 1-\delta$ over $S \sim D^m$, simultaneously for every n and every $h \in \mathcal{H}_n$:

$$|L_D(h) - L_S(h)| \;\leq\; \epsilon_n\big(m,\, w(n)\,\delta\big).$$

- **Idea of proof:** apply uniform convergence to each $\mathcal{H}_n$ at its own confidence level $w(n)\delta$, then a union bound over n; since $\sum_n w(n) \leq 1$, the total failure probability stays $\leq \delta$.
- **SRM rule:** pick h minimizing the bound itself,

$$h = \arg\min_{h'} \Big[L_S(h') + \epsilon_{n(h')}\big(m,\, w(n(h'))\,\delta\big) \Big].$$

Countable $\mathcal{H}$: SRM becomes an MDL-like rule

- If $\mathcal{H}$ is countable, write it as a union of singletons $\{h_n\}$.
- Hoeffding's inequality gives each singleton $\epsilon_n(m, \delta) = \sqrt{\log(2/\delta)/(2m)}$.
- The SRM rule becomes, writing w directly as a function of h:

$$\arg\min_{h \in \mathcal{H}} \left[L_S(h) + \sqrt{\frac{-\log w(h) + \log(2/\delta)}{2m}} \right].$$

$-\log w(h)$ is a **complexity penalty** in bits — this is exactly the description-length term MDL will use.

Minimum Description Length and Occam's razor

Kraft's inequality

If $S \subseteq \{0,1\}^\star$ is prefix-free, then $\sum_{\sigma \in S} 2^{-|\sigma|} \leq 1$.

- **Why:** toss a fair coin repeatedly until the outcomes match some $\sigma \in S$. Prefix-freeness means each σ is reached with probability exactly $2^{-|\sigma|}$, and these events are disjoint — so their probabilities sum to at most 1.
- Any prefix-free description language $d : \mathcal{H} \to \{0,1\}^\star$ therefore gives a valid weighting $w(h) = 2^{-|d(h)|}$.

MDL bound

For every m, δ, D, with probability $\geq 1 - \delta$: $\forall h \in \mathcal{H}$,

$$L_D(h) \;\leq\; L_S(h) + \sqrt{\frac{|d(h)| + \ln(2/\delta)}{2m}}.$$

Shorter description $\Rightarrow$ tighter bound $\Rightarrow$ Occam's razor, formalized.

PAC-Bayes: from one hypothesis to a distribution

- A prior P over $\mathcal{H}$ is just a weight function, $P(h) = w(h)$ — MDL is the special case $P(h) = 2^{-|d(h)|}$.
- Bayesian idea: why should the learner output a single h? Instead, learn a whole **posterior** Q over $\mathcal{H}$.
- Randomized prediction rule: given a new x, draw $h \sim Q$ and predict $h(x)$.
- Loss of Q on example z: $\ell(Q, z) := \mathbb{E}_{h \sim Q}[\ell(h, z)]$, and by linearity

$$L_D(Q) := \mathbb{E}_{h \sim Q}[L_D(h)], \qquad L_S(Q) := \mathbb{E}_{h \sim Q}[L_S(h)].$$

The PAC-Bayes bound

Theorem (PAC-Bayes)

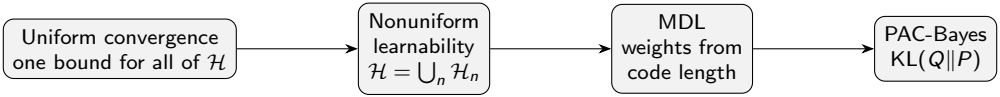
Let P be a prior over $\mathcal{H}$, $\delta \in (0,1)$. With probability $\geq 1-\delta$ over $S \sim D^m$, *simultaneously for every* distribution Q over $\mathcal{H}$ (even one chosen after seeing S):

$$L_D(Q) \;\leq\; L_S(Q) + \sqrt{\frac{\mathrm{KL}(Q\|P) + \ln(m/\delta)}{2(m-1)}},$$

where $\mathrm{KL}(Q\|P) := \mathbb{E}_{h\sim Q}\big[\ln(Q(h)/P(h))\big]$.

- Proof: see Chapter 31 of the book (not reproduced here).
- $\mathrm{KL}(Q\|P)$ is a continuous, soft analogue of description length: minimizing the bound trades off empirical loss of Q against how far Q strays from the prior P.

Synthesis: one axis, three tools



Each step relaxes "uniform" further and measures hypothesis complexity in bits — the same currency as Part 1.

From MDL to Algorithmic Information Theory

- MDL depends on a *chosen* description language.
- AIT asks: what if we use the most universal language possible — a universal Turing machine?
- **Kolmogorov complexity** $K(x)$: length of the shortest program producing x.
- **Solomonoff's algorithmic probability**: $P(x)$ = probability a random program on a prefix UTM outputs x.

Formalizing simplicity: algorithmic probability

Algorithmic probability

$P(x)$ = probability a random program on a (prefix) universal Turing machine U generates x.

$$P_U(x) = \sum_{\ell\,:\,U(\ell)=x} 2^{-|\ell|} \;=\; 2^{-K(x)} + \cdots$$

- Sum over all binary programs ℓ that make U output x.
- The shortest such program (length $K(x)$) dominates the sum.

$\Rightarrow$ simple outputs (small $K(x)$) are exponentially more likely to appear.

Formalized by R. Solomonoff (1926–2009); Marvin Minsky called algorithmic probability one of the most important ideas since Gödel.

The algorithmic monkey

Question: what is the probability a monkey types the first M digits of π on an N-key typewriter?

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

But what if the monkey types into a C compiler instead?

- A 133-character (obfuscated) C program computes the first 15,000 digits of π, via

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

- So the probability of typing *some* program that outputs those digits is

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

vastly larger than $(1/N)^{M+1}$ once $M \gg 133$.

Punchline

- Simple, structured outputs are reached by disproportionately many random programs.
- Levin's coding theorem formalizes this:

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

- Intuition: probability mass is dominated by the shortest program, so

$$P(x) \approx 2^{-K(x)}.$$

- Same logic as noise-incompressibility in Part 1, now with a universal description language instead of a fixed code.

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

The puzzle this paper addresses

- **Valle-Pérez, Camargo & Louis, ICLR 2019** — "Deep learning generalizes because the parameter-function map is biased towards simple functions."
- Zhang et al. (2017) showed DNNs can memorize CIFAR-10 with *randomly permuted labels* just as fast as real labels $\Rightarrow$ classical complexity measures (VC-dim, Rademacher) give vacuous bounds.
- So capacity alone cannot explain generalization. **Something about which functions DNNs actually reach must be biased.**

The parameter-function map

Definition (parameter-function map)

For a model with parameters $\theta \in \Theta \subseteq \mathbb{R}^p$, the map $\mathcal{M} : \Theta \to \mathcal{F}$, $\theta \mapsto f_\theta$, sends parameters to the function the network computes.

- SGD searches in parameter space Θ; generalization depends on where it lands in function space $\mathcal{F}$.
- Building on Dingle et al. (2018, *Nature Communications*), a general **probability–complexity bound** holds for simple, redundant, nonlinear input–output maps:

$$P(x) \;\leq\; 2^{-a\tilde{K}(x)+b}$$

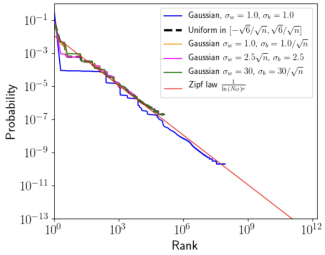
for constants a, b depending on the map but not on x, where $\tilde{K}(x)$ approximates Kolmogorov complexity.

- Claim: the DNN parameter-function map satisfies these conditions $\Rightarrow$ it should be exponentially biased toward simple f.

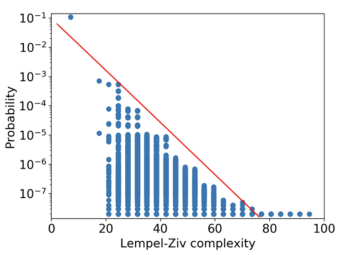
Testing this on Boolean functions

- Toy model: 7 Boolean inputs, two hidden layers of 40 ReLU units, one Boolean output — small enough to enumerate probabilities by sampling.
- Sample $\sim 10^7$–10^8 parameter settings (Gaussian or uniform), record the empirical frequency of each function f_θ.
- Complexity measure: **Lempel–Ziv complexity** $K_{LZ}(x)$ of the function's truth table (a computable stand-in for Kolmogorov complexity).

Simplicity bias: probability vs rank and complexity



(a)



(b)

Simplicity bias: probability vs rank and complexity

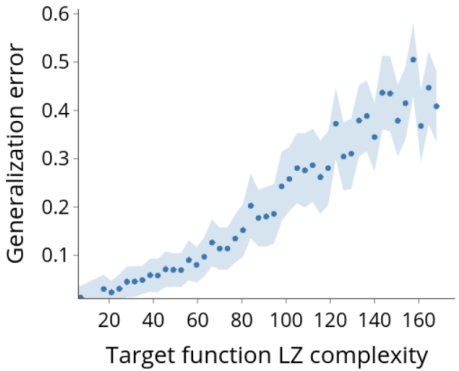
- **(a) Rank-frequency (Zipf-like):** Function probabilities follow a heavy-tailed distribution.
  - A small number of simple functions dominate the probability mass.
- **(b) Probability vs. complexity:**

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

  - Probability decays exponentially with Lempel–Ziv complexity.
  - The red line is the simplicity-bias upper bound (Levin-style).
- **Takeaway:** High-probability $\Rightarrow$ low complexity (strong simplicity bias).

Valle-Pérez et al., arXiv:1805.08522.

Simplicity bias ⇒ generalization



- **(c) Generalization vs. complexity:**
  - Targets with higher LZ complexity have higher generalization error.
- **Interpretation:**
  - Neural networks implicitly sample functions with probability $\propto$ simplicity.
  - Training is biased toward low-complexity solutions.
- **Key link:**
  - Simplicity bias ⇒ better generalization.
- Matches Levin's intuition: $P(x) \approx 2^{-K(x)}$.

From bias to a generalization bound: PAC-Bayes

PAC-Bayes theorem (Langford & Seeger, 2001)

For prior P over $\mathcal{H}$ and any posterior Q, with probability $\geq 1-\delta$:

$$L_D(Q) \ln \frac{L_D(Q)}{L_S(Q)} + (1 - L_D(Q)) \ln \frac{1 - L_D(Q)}{1 - L_S(Q)} \leq \frac{\mathrm{KL}(Q\|P) + \ln(2m/\delta)}{m-1}$$

$L_D(Q) = \mathbb{E}_{h\sim Q}[L_D(h)], \quad L_S(Q) = \mathbb{E}_{h\sim Q}[L_S(h)]$

From bias to a generalization bound: PAC-Bayes

Realizable case

There exists a hypothesis h with zero training error:

$$\exists\, h \in \mathcal{H} \text{ such that } L_S(h) = 0$$

Let $U = \{h : L_S(h) = 0\}$ (all hypotheses perfectly fitting the sample), and define:

$$Q^\star(h) \propto P(h) \quad \text{on } U$$

Then:

$$\mathrm{KL}(Q^\star \| P) = \ln \frac{1}{P(U)}$$

$$-\ln(1 - L_D(Q^\star)) \;\leq\; \frac{\ln \frac{1}{P(U)} + \ln(2m/\delta)}{m-1}$$

Takeaway

- **Key takeaway:**
  - Generalization is controlled by $P(U)$
  - $\Rightarrow$ how much prior mass is on hypotheses fitting the data

From Parameters to Functions

We start with a distribution over parameters:

$$\tilde{P}(\theta) \tag{1}$$

This induces a distribution over functions via the map M:

$$P(f) = \tilde{P}(M^{-1}(f)) \tag{2}$$

Key point:

$$\text{We care about } P(f), \text{ not } \tilde{P}(\theta) \tag{3}$$

because generalisation depends on the *function*, not the parameters.

Why Focus on $P(U)$?

Let U be the set of zero-training-error functions:

$$U = \{f : f(x_i) = y_i \ \forall (x_i, y_i) \in \mathcal{D}\} \tag{4}$$

We want:

$$P(U) = \sum_{f \in U} P(f) \tag{5}$$

Why this matters:

$$P(f \mid f \in U) = \frac{P(f)}{P(U)} \tag{6}$$

- This is the **Bayesian posterior over functions**
- Needed to compare with SGD

Why is $P(U)$ Intractable?

$$P(U) = \sum_{f \in U} P(f) \tag{7}$$

Problems:

- The number of functions is **astronomical**
- Each $P(f)$ depends on a complicated pre-image:

$$M^{-1}(f) = \{\theta : f_\theta = f\} \tag{8}$$

- Volumes in parameter space are highly irregular

Conclusion:

$$\text{Direct computation of } P(U) \text{ is impossible} \tag{9}$$

Key Idea: Replace Network with GP

In wide-network limit:

$$f(x) \sim \mathcal{GP}(0, K(x, x')) \tag{10}$$

$$K(x, x') = \mathbb{E}_\theta[f_\theta(x) f_\theta(x')] \tag{11}$$

Crucial advantage:

$$P(f \mid \mathcal{D}) \text{ becomes tractable} \tag{12}$$

because everything is Gaussian.

How GP Helps Compute $P(U)$

Under GP:

$$\mathbf{f} = (f(x_1), \ldots, f(x_n)) \sim \mathcal{N}(0, K) \tag{13}$$

Then:

$$P(U) = P(\mathbf{f} = \mathbf{y}) \tag{14}$$

$$= \frac{1}{\sqrt{(2\pi)^n |K|}} \exp\left(-\frac{1}{2}\mathbf{y}^T K^{-1}\mathbf{y}\right) \tag{15}$$

Now computable!

SGD and Function-Space Bias

Given a parameter distribution:

$$\tilde{P}(\theta) \tag{16}$$

It induces:

$$P(f) = \tilde{P}(M^{-1}(f)) \tag{17}$$

Key assumption:

$$\tilde{P}(\theta) \approx \text{uniform} \tag{18}$$

Then:

$$P(f) \propto \text{Vol}(M^{-1}(f)) \tag{19}$$

Origin of the Bias

Core insight:

$$\text{Different functions occupy vastly different volumes in parameter space} \tag{20}$$

$$\Rightarrow P(f) \text{ varies exponentially} \tag{21}$$

Implication:

- SGD samples functions with probability proportional to volume
- Large-volume functions dominate

$$\Rightarrow \text{Strong inductive bias towards simple hypotheses} \tag{22}$$

SGD Behaviour

Assumption:

$$\text{SGD samples the zero-error region nearly uniformly} \tag{23}$$

$$P_{\text{SGD}}(f \mid f \in U) \approx \text{const} \tag{24}$$

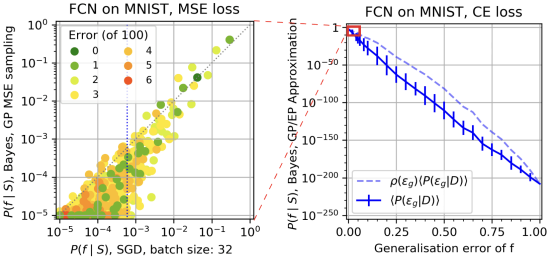
But:

$$P(f \mid f \in U) \propto P(f) \tag{25}$$

Conclusion:

$$\text{SGD} \approx \text{Bayesian sampler weighted by } P(f) \tag{26}$$

Bayesian vs SGD Distribution



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

$$(a)\ P_B(f|S) \text{ vs } P_{\mathrm{SGD}}(f|S) \tag{27}$$

$$(b)\ P_B(f|S) \text{ vs generalisation error} \tag{28}$$

Interpretation of Results

Left plot:

$$P_B(f|S) \approx P_{\text{SGD}}(f|S) \tag{29}$$

- Strong correlation across many orders of magnitude

Right plot:

$$P(f|S) \downarrow \quad \text{as generalisation error} \uparrow \tag{30}$$

- High-probability functions generalise better

Takeaway:

$$\text{Generalisation emerges from function-space bias} \tag{31}$$