Real Data Distributions Prefer Simplicity and So Do Our Models: Why Machine Learning and Model Selection Are Possible

Anonymous authors
Paper under double-blind review

Abstract

No free lunch theorems for supervised learning state that no learner can solve all problems or that all learners achieve exactly the same accuracy on average over a uniform distribution on learning problems. Accordingly, these theorems are often referenced in support of the notion that individual problems require specially tailored inductive biases. While all but exponentially few uniformly sampled datasets have high complexity, we argue that neural network models share the same preference for low-complexity data that we observe on real-world problems. Notably, we show that architectures designed for a particular domain, such as computer vision, are compressors for labeling functions on a variety of seemingly unrelated domains. From our experiments, we see that pre-trained and even randomly initialized language models prefer to generate low-complexity sequences and can therefore be used for inference. In principle, the use of expert knowledge and bias for simplicity of human practitioners could be folded into the learning algorithm, automating design and selection of models. We explain how typical areas requiring human intervention such as picking the appropriately sized model when labeled data is sparse or plentiful can be automated into a single learning algorithm. These observations help justify the trend in deep learning of unifying seemingly disparate problems with an increasingly small set of machine learning models.

1 Introduction

The problem of justifying inductive reasoning has plagued epistemologists since at least the 1700s (Hume, 1748). More recently, in the late 1990s, *no free lunch theorems* emerged from the computer science community as rigorous versions of arguments showing the impossibility of induction in contexts seemingly relevant to real machine learning problems (Wolpert, 1996; Wolpert & Macready, 1997). One such no free lunch theorem for supervised learning states that no single learner can solve every problem (Shalev-Shwartz & Ben-David, 2014). Another states that, assuming a world where the labeling functions of learning problems are drawn from a uniform distribution, no learner can reliably perform inference at all (Wolpert, 1996). Such a world would be altogether hostile to inductive reasoning. The assumption that labelings are drawn uniformly ensures that training set labels are entirely uninformative about unseen samples.

In contrast to this dismal outlook on machine learning, naturally occurring inference problems involve highly structured data. If we can design learning algorithms with inductive biases that are aligned with this structure, then we may hope to perform inference on a wide range of problems. In this work, we explore structure in both real-world data and modern machine learning models, primarily through the lens of *Kolmogorov complexity*.

The Kolmogorov complexity of some output is defined as the length of the shortest program under a fixed language that produces that output. In Section 3, we explain the connection between Kolmogorov complexity and compressibility. We note that virtually all random data cannot be significantly compressed, yet relevant datasets are highly compressible and hence substantially less complex. In particular, neural networks themselves can be used to create compression of data labelings, upper bounding their Kolmogorov complexity.

Complementing the low complexity of actual data, we then demonstrate in Section 4 that modern neural networks prefer low Kolmogorov complexity too. While models implemented on a computer provably cannot generate data with complexity exceeding the length of their associated program, we find they actually prefer data that is far simpler. We formulate simple languages for generating numerical sequences, under which we can directly measure the Kolmogorov complexity of a sequence. We use these languages to inspect the simplicity bias of both pre-trained and randomly initialized language models. GPT-3 (Brown et al., 2020) reliably favors less complex sequences, and bigger and better GPT-3 variants even more so. Notably, randomly initialized GPT models share this simplicity bias. As such, we can use them to predict the next element in a sequence as long as the target sequence is low-complexity. To further emphasize the universality of this simplicity bias, we cram tabular data from diverse domains, including click prediction and airline delay prediction, into convolutional computer vision architectures and show that these vision architectures prefer correct labelings to random ones, even on data which do not remotely resemble natural images. We use this property to compute cross-domain non-vacuous PAC-Bayes generalization bounds.

A common intuition associated with no free lunch theorems dictates that since a single learner cannot solve all problems, practitioners must inspect data and manually select an appropriate learner for the specific problem at hand. For example, a practitioner might select a more constrained model to avoid overfitting on small datasets, or they may select convolutional architectures to accommodate natural image data. To the contrary, we explain in Section 5 why model selection can be automated from the standpoint of PAC-Bayes theory, namely selecting between learning algorithms bears negligible complexity cost, and this cost is quickly overcome by gains in validation accuracy under reasonably sized validation splits. Moreover, a single learner, which on the one hand supports a variety of functions but on the other hand prefers simple ones, can solve a wide range of problems. We show that flexible models accompanied by a penalty which encourages simple solutions can solve problems with a variety of sample sizes. For example, a single learner which combines a small convolutional neural network and a transformer, while encouraging use of the small model insofar as it fits the training data, achieves strong performance both on small datasets like CIFAR-10 (Krizhevsky, 2009) and also large datasets like ImageNet (Deng et al., 2009). We highlight in Figure 1 that the historic evolution of machine learning systems supports the ability of a single learner to perform diverse tasks as highly task-specific pre-neural algorithms, such as Latent Dirichlet Allocation (Blei et al., 2003) and HOG (Dalal & Triggs, 2005), were replaced by neural architectures such as convolutional or recurrent models, and transformers can now handily perform all tasks listed.

Our contributions are summarized as follows:

- We demonstrate the direct connection between compressibility and learnability that is implicit in no free lunch theorems by deriving a new NFL theorem using Kolmogorov complexity.
- We show that the low Kolmogorv complexity of real datasets can be directly derived from the machine learning models used to fit them.
- We provide strong empirical evidence that neural networks have low complexity biases that are relevant even on data far from what they were designed for. To this end, we compute generalization bounds using convolutional vision models on tabular data, we showcase GPT-3's strong preference for sequences generated by shorter expression trees, and we find that even randomly initialized language models have a simplicity bias.
- We argue why in principle model selection can be automated into a larger algorithm, and we provide explicit examples demonstrating how multiple models can be subsumed into a single powerful learner.

2 Background

No free lunch theorems. No free lunch theorems (NFL) state that without making strong assumptions, a single algorithm cannot simultaneously solve all problems well. No free lunch theorems for search and optimization indicate that all optimizers and search algorithms satisfying certain conditions perform exactly the same on average over all such search or optimization problems (Wolpert et al., 1995; Wolpert & Macready, 1997). In this work, we narrow our focus to NFL for supervised learning. Wolpert (1996), and similarly Schaffer (1994), proves an analogous theorem for supervised learning under which every learner—a function that takes in labeled data and outputs a labeling function for the associated domain—achieves exactly the same accuracy of 50% on average over all binary classification problems where accuracy is only evaluated on unseen samples.

In order to prove no free lunch theorems, one needs to place very strong assumptions on the lack of structure in the labeling function such as a uniform distribution, so that conditioning on the training labels does not modify the probability over labelings on unseen points (Rao et al., 1995). To illustrate the severity of this condition, imagine that a person is presented a sequence of one million 1s and asked whether they predict that the next element will be 1 or 0. If labelings of sequence elements were distributed uniformly, then we should assign equal probability to both options, despite the fact that intuition and Bayesian probabilistic models tell us overwhelming to favor 1.

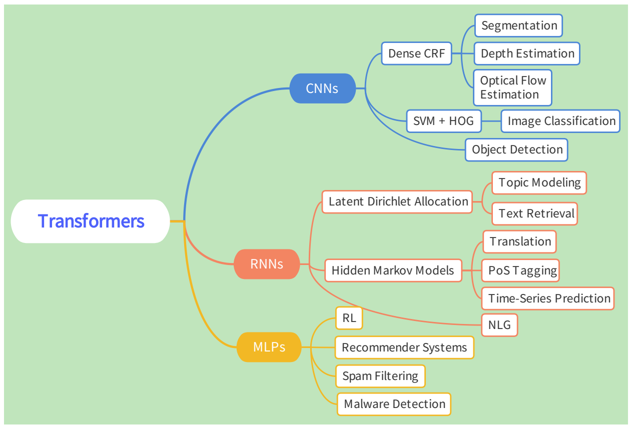


Figure 1: Over time, tasks that were performed by domain-specialized ML systems are increasingly performed by unified neural network architectures.

Shalev-Shwartz & Ben-David (2014) instead do not assume a particular distribution over learning problems and prove that for every learner, there is a task on which the learner achieves poor accuracy with high probability over training splits, whereas another learner solves the same problem with perfect accuracy. Notably, the latter NFL computes accuracy over all data, not just "off-training" samples. While this statement of NFL does not require uniformly distributed data, if the existence of catastrophic failure modes for our learners is to be damning, our learners must encounter them in practice. After all, we do not care if our learners cannot solve problems we do not want to solve. Thus, the practical relevance of this theorem again hinges on the distribution over real-world learning problems and how well it aligns with the inductive bias of a learner. In this paper, we argue that the real-world learning problems we care about are highly structured, and the inductive biases of neural networks are well-aligned with such problems.

Kolmogorov complexity and compression. Kolmogorov complexity is a quantitative measure of complexity. For any fixed programming language L, the Kolmogorov complexity of data x, $K(x)$, is the length of the shortest program in that language that outputs x (Kolmogorov, 1963). We can similarly define $K(y|x)$ as the length of the shortest program which inputs x and outputs y. For definiteness, we assume that all programs and outputs are finite bitstrings.

Despite the fact that one cannot prove that any particular string has high complexity (Chaitin, 1974), all but exponentially few sequences of a given length have near maximal Kolmogorov complexity. Since there can only be less than 2^{n+1} programs of length less than or equal to n, then by the pigeonhole principle, there are at most 2^{n+1} distinct bitstrings x with $K(x) \leq n$. Equivalently, drawing a random bitstring x of length n, there is at most a 2^{1-k} chance that $K(x) \leq n-k$. As this probability decays exponentially in k, this bound shows that the vast majority of bitstrings of length n have complexity close to n. One way of upper bounding $K(x)$ is to compress x, but then we must include both the size of the compressed file and the size of the program required to decompress it. Alternatively, if we can construct a short program which directly outputs x, this program also forms a compression of x. The above discussion then demonstrates that the vast majority of bitstrings do not admit a non-trivial compression.

Notions of Kolmogorov complexity and simplicity bias have also been discussed in the context of universal induction algorithms and machine learning more broadly. Notably, one can use Kolmogorov complexity to construct the Solomonoff prior (Solomonoff, 1964), $P(x) = 2^{-[K(x)+2\log K(x)]}/Z$ (with $Z < 1$), a formalization of the principle of Occam's razor favoring simplicity. One can perform inference using just this prior via universal induction algorithms, and we discuss these matters in more detail along with complexity in deep learning in Appendix A.

PAC-Bayes compression bounds. Generalization bounds limit how a model's expected risk $R(h)$ will differ from its train risk $\hat{R}(h)$. PAC-Bayes generalization bounds (McAllester, 1998; 2003; 2013; Alquier, 2021) can be considered a generalization and improvement of the finite hypothesis bounds (Mohri et al., 2018) which provide a generalization guarantee even when the number of hypotheses is large (or even infinite) so long as the given hypothesis is likely under a *prior* $P(h)$. The simplest version of the bound (McAllester, 1999) applied with a point mass posterior $Q(h) = \mathbf{1}_{[h=h^*]}$ states that

$$R(h) \leq \hat{R}(h) + \sqrt{\frac{-\log P(h) + \log(n/\delta) + 2}{2n-1}} \tag{1}$$

with probability at least $1-\delta$, where n is the size of the training set. Choosing the Solomonoff prior $P(h) = 2^{-[K(h)+2\log K(h)]}/Z$ as done in Lotfi et al. (2022b) and noting that the normalization constant $Z \leq 1$, the prior likelihood of a given hypothesis can be bounded via

$$-\log_2 P(h) \leq K(h) + 2\log_2 K(h) \leq C(h) + 2\log_2 C(h)$$

for any compression scheme C. Combining these two bounds provides a theoretical basis for the PAC-Bayes compression approach explored in other work (Zhou et al., 2018; Lotfi et al., 2022b) and can be used to derive nonvacuous generalization bounds even for large neural networks when using a strong compression scheme.

3 Structure in Real-World Data

The often cited no free lunch theorem of Wolpert (1996) states that all learners perform the same when averaged over a uniform distribution on all possible datasets. However, the assumption of uniform samples is very strong and extremely unnatural. This assumption subtly selects high complexity incompressible data, a set of problems on which learning is fundamentally impossible. Through means of hypothesis test we are able to conclusively rule out the possibility that real datasets are drawn from the uniform distribution, and therefore assertions that the NFL applies to problems of interest. We bring to the surface the centrality of incompressibility in NFL theorems by deriving a new NFL theorem directly from ideas of data compression and Kolmogorov complexity.

3.1 An Exercise in Bounding the Complexity of Datasets

We first consider the hypothesis that unlabeled machine learning datasets are drawn uniformly at random and use a bound on the Kolmogorov complexity as a test statistic. Using `bzip2`, and including the size of the compression program, we compress text dataset Amazon Review Full (McAuley & Leskovec, 2013) and audio dataset LibriSpeech (Panayotov et al., 2015) to 393.2 MB and 8.36 GB respectively, providing upper bounds on the Kolmogorov complexity with respect to the python programming language. Supposing the datasets were in fact uniformly randomly sampled datasets, the probability of observing complexities this size or smaller is less than $2^{-2^{32}}$ and $2^{-2^{36}}$, astronomically low p-values, conclusively ruling out the possibility that they were sampled in this way. If we randomly shuffle the datasets, we instead obtain bounds of only 836.7 MB and 9.69 GB, considerably larger, showing that the compressibility results not just from an inefficient encoding, but from structure in the dataset. Other works have also examined Kolmogorov complexity in data, for example EEG patterns (Petrosian, 1995) or animal behavior (Zenil et al., 2015), and likewise confirm that such naturally occurring data is simple.

3.2 Neural Networks as Compressors of the Labeling Function

More relevant to the context of supervised learning, we show that not only are the unlabeled datasets compressible—the labeling functions are too. Further, this can be demonstrated using trained models as compressors. Given a labeled dataset $\mathcal{D} = (X, Y) = \{(x_i, y_i)\}_{i=1}^n$, any likelihood model $p(y|x)$—regardless of whether the top predictions are correct—can be used to generate a lossless compression scheme to encode the dataset labels Y given the inputs X. Using a stream code such as arithmetic coding (Witten et al., 1987) in combination with the probability model $p(y|x)$, the labels can be encoded in $K(Y|X,p) \leq -\sum_i^n \log_2 p(y_i|x_i) + 3$ bits. Models which maximize the log likelihood of the data also implicitly minimize the length of this encoding.

One can show by delimiting, as is done in Fortnow (2000), that $K(Y|X) \leq K(Y|X,p) + K(p) + 2\log_2 K(p) + c$, where c is some small constant depending on the language. Writing the negative log likelihood in terms of the empirical cross entropy, combining our two inequalities, and dividing by the size of the dataset n yields

$$\tfrac{1}{n}K(Y|X) \leq \mathrm{CE}/\ln 2 + n^{-1}(K(p) + 2\log_2 K(p) + c), \tag{2}$$

where CE is the cross entropy of the classifier p averaged over dataset $\mathcal{D}$. This implies that, regardless of how large the model is—so long as CE is better than random guess—if the size n of the dataset is large enough, the model provides a non-trivial compression of the dataset. To demonstrate this fact, we employ the compression scheme from Lotfi et al. (2022b) in order to find a compressed representation of MLPs on several class balanced tabular classification datasets[1]. As shown in Figure 2 (middle), we are able to compress the labels on most of the datasets by well over the naive $n\log_2 C$ encoding length where C is the number of classes. We also apply the method with convolutional architectures to compress labels on CIFAR-10 and CIFAR-100 in Figure 2 (right), allowing us to reject the hypothesis that the labeling functions are drawn uniformly at random with extremely high confidence.

3.3 A Kolmogorov-Style No Free Lunch Theorem

A corollary of Equation 2 is that if the dataset is incompressible, then no model can do better than random chance in the limit of a large dataset. Since compressible datasets in uniformly sampled data are exponentially unlikely, we can prove our own version of the no free lunch theorem directly from this constraint on complexity. With very high probability, on any given uniformly sampled dataset, learning is impossible.

Theorem 1. *Let (X,Y) be a labeled dataset with n data points and uniformly sampled random labels from C classes. Then, with probability at least $1-\delta$, for every classifier $p(y|x)$,*

$$\mathrm{CE}(p) \geq \ln C - \frac{\ln 2}{n}\left(K(p) + 2\log_2 K(p) + \log_2\frac{1}{\delta} + c\right) \tag{3}$$

where CE(p) is the empirical cross entropy of the classifier $p(y|x)$ on the data. Thus for any model of bounded size, if the size of the dataset is large enough, the model cannot represent any classifier with cross entropy appreciably smaller than that attained from random guess.

Proof: *See Appendix B*

Like any of the no free lunch theorems, this initially seems limiting, but here we see that the incompressiblility of the datasets from the uniform distribution is the fundamental issue. On compressible datasets (ones with less than maximal complexity), learning is possible.

Takeaway: Real datasets are highly unlike the high complexity samples from the uniform distribution where learning is impossible.

4 Low-Complexity Bias in Machine Learning Models

In the previous section, we saw that real-world data distributions across domains are highly structured and therefore have low Kolmogorov complexity. If we can construct models which prefer the same low-complexity data, then we can hope to perform inference with a single model across many such domains. While early machine learning systems incorporated highly domain-specific design elements, such as handcrafted image features (Dalal & Triggs, 2005) or graphical models for language modeling (Mnih & Hinton, 2007), modern neural network architectures across domains are converging on transformers (Vaswani et al., 2017; Dosovitskiy et al., 2020; Gulati et al., 2020; Somepalli et al., 2021), some of which can simultaneously achieve impressive performance on a variety of data types with a single architecture (Jaegle et al., 2021).

In this section, we argue that neural networks have a generic simplicity bias that extends well beyond the datasets for which they are designed. To this end, we: (1) feed tabular datasets from diverse

[1] https://www.openml.org/

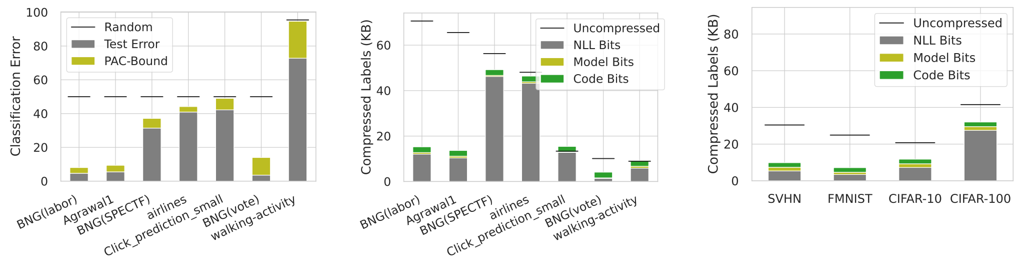


Figure 2: (Left): PAC-Bayes generalization bounds for CNNs trained on tabular data. (Middle): Compression of tabular labels via a trained MLP model vs direct encoding of labels ($n \log_2 C$). (Right): Compression of image classification datasets using CNNs. Note the breakdown of the total compressed size of the labels into model fit (NLL bits), compressed parameters (Model Bits), and architecture and decompressor (Code Bits).

domains such as click prediction and airline delay prediction into neural networks designed specifically for computer vision and find that they prefer naturally occurring labelings to random ones, (2) formulate a language with respect to which we can measure the Kolmogorov complexity of numerical sequences and observe that GPT-3 generates low-complexity sequences with exponentially higher probability than complex ones, (3) predict the next term in a sequence with randomly initialized language models. Whereas the no free lunch theorem of Wolpert (Wolpert, 1996) ensures that such an inference procedure cannot outperform a random guess on average, we find that randomly initialized neural networks prefer sequence completions which generate low-complexity completed sequences, demonstrating that they can make accurate guesses as long as the ground truth sequence distribution also favors low complexity.

4.1 Neural Networks Prefer Naturally Occurring Labelings Across Domains

The inductive biases of even specialized architectures like convolutional neural networks facilitate broad learning abilities. To demonstrate this fact, we take tabular classification datasets and encode the tabular features as an image by simply forming images with one channel so that each pixel corresponds to a different feature, and padding the edges with zeros as needed. We train a small 9 layer convolutional network using this input data to predict the classification labels. Since the data has no local or translation equivariant structure, learning with the convolutional network requires breaking these structures. Even in spite of this mismatch, the convolutional networks perform much better than random chance. Furthermore, using the PAC-Bayes compression methodology from Lotfi et al. (2022b), we are able to find a compressible set of parameters providing nonvacuous generalization bounds on the model's performance, as shown in Figure 2. Despite violating the locality and translation equivariance inductive biases of the architecture, the model is still highly compressible when fitting the tabular data and provably generalizes, suggesting that a core part of these inductive biases are general and shared with this tabular data.

Takeaway: While architectures such as CNNs have inductive biases tailored for specific problems, much of their inductive bias is general, extending across domains.

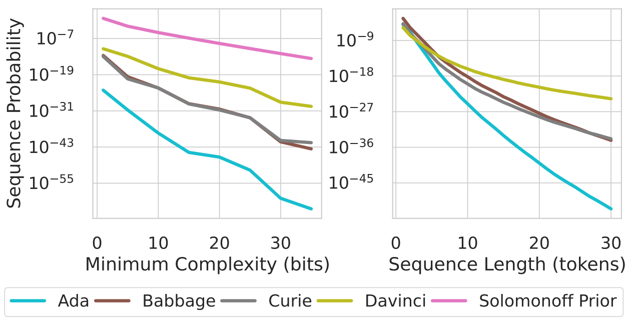
4.2 GPT-3 Assigns Exponentially Higher Probability to Simpler Sequences

We now study the preference of GPT-3—a line of high-performance autoregressive text generators—has for simpler sequences. The ability of language models to solve reasoning problems has recently been studied by Zelikman et al. (2022), who develop an effective prompting framework, and d'Ascoli et al. (2022), who develop transformers for predicting symbolic expressions directly from the sequence. To perform our own study, we need a well-defined, computable notion of complexity. We thus define a simple, non-Turing-complete language and measure Kolmogorov complexity with respect to this simple language. Namely, we generate integer sequences with binary expression trees. We then define the complexity of a sequence as the size of the smallest expression tree, measured as the number of internal nodes—or equivalently the number of operators in the expression represented by the tree—that generates that sequence. By using a small set of terms for the leaves and binary operators for the nodes, we can enumerate over all possible expression trees for sufficiently small sizes at most L and compute all sequences with complexities 0 through L.

In our experiments, we use operations $+$, $\times$, and $//$, where $//$ denotes integer division. For leaves, we use 2 and i, where i is the index within the sequence. For example, $(2 + i) \times i$ could be implemented with a tree of size 2 and would generate the sequence $a_i = 0, 3, 8, 15, ...$ Using this setup, we are able to generate sequences of varying complexity, according to a well-defined metric, and quantify the preference of GPT-3 models for simpler sequences over more complex ones. We provide details on how we tokenize sequences and extract their probabilities in Appendix D.

In Figure 3, we measure the average log-probability GPT-3 models assign to sequences of a given Kolmogorov complexity, where we fix the number of numerical tokens input into the model to be 30, and we observe that the probabilities assigned by these language models decrease exponentially with sequence complexity, similar to the Solomonoff prior discussed in Section 2. In contrast, a uniform prior would be described by a flat line. Note that for very complex sequences, we cannot easily measure their minimum description length, so we limit our experiments to expression trees with at most 7 operators. In this low-complexity regime, we observe that bigger GPT-3 models which are better at language modeling, such as `Davinci` which contains 175 billion parameters, assign higher probability to these simple sequences than much smaller GPT-3 models such as `Ada`. The legend lists GPT-3 variants in order of increasing size.

We can also examine the decay of such log-probabilities as we feed more tokens, corresponding to digits of sequence elements, into the model. As the sequences get longer, we see in Figure 3 that the probabilities assigned to sequences decay sub-exponentially, indicating that these models, especially bigger variants such as `Davinci`, become more and more confident about later sequence elements.



$2 \rightarrow$
$2, 2, 2, 2, 2, 2, 2, ...$

$i \times i \rightarrow$
$0, 1, 4, 9, 16, 25, ...$

$2 \times (i + 2) \rightarrow$
$4, 6, 8, 10, 12, 14, ...$

$i//(i \times (i//2)) \rightarrow$
$0, 1, 1, 1, 0, 0, 0, 0, ...$

$i + i + (i//(i//2)) \rightarrow$
$0, 3, 6, 9, 10, 12, 14, ...$

Figure 3: GPT-3 prefers low-complexity sequences generated by expression trees. Left: Average log-probability of sequences by complexity. **Middle:** Average log-prob. by sequence length, restricted to decimal digit tokens. **Right:** Example sequences of complexity 0 through 4 with minimal generating expressions.

4.3 Even Randomly Initialized Language Models Prefer Low Complexity

The previous section solely examined pre-trained language models, but these models have been trained on massive corpora of data. Do they prefer low complexity before they have even seen any data at all? While initializers of such models follow highly diffuse distributions over parameters, such random parameters can induce highly structured functions.

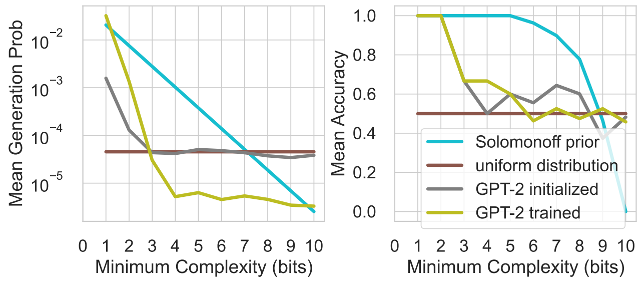


Figure 4: Randomly initialized language models prefer low-complexity sequences generated by bitstring repetition. Left: Average log-probability of sequences by complexity. **Right:** Average accuracy by complexity.

Trained language models are known to repeat themselves when generating text (Holtzman et al., 2020; Fu et al., 2021). One might think that this behavior is learned from training data which often contains repeated text, but we show in this section that randomly initialized GPT models repeat themselves too. Interestingly, we can formalize the preference for repetition as a preference for low Kolmogorov complexity. In order to disentangle the impact of initialization from training, we adopt a simple language for generating binary sequences under which we can quickly measure Kolmogorov complexity. We consider a program to be a bitstring, and then the program upon execution simply repeats the bitstring until output reaches length 10. Under this language, the sequence $0, 0, 0, ...$ has Kolmogorov complexity 1, and $0, 1, 0, 1, ...$ has complexity 2, yet randomly generated sequences are exponentially more likely to have high complexity. We conduct our evaluations exhaustively on all such sequences of length 10.

We now generate sequences of length 10 with randomly initialized GPT-2 language models (Radford et al., 2019), such that each initialization is used to generate one sequence, and we measure the frequency with which each such sequence is generated. The estimated generation probabilities by Kolmogorov complexity are found in Figure 4, where we see again that low-complexity sequences are assigned exponentially higher probabilities. Here, we see that pre-trained checkpoints exhibit an even stronger preference for low complexity as they are trained on structured text. We can also use the randomly initialized language models to perform next element prediction by estimating the probabilities they assign to the next element in a sequence given having correctly generated the previous terms. While Wolpert's no free lunch theorem (Wolpert, 1996) ensures that the average completion accuracy over all possible length 10 bitstrings is exactly 0.5, we verify in Figure 4 that randomly initialized networks can be used for sequence completion when the sequence is of low complexity. Additional details and experiments with other GPT-2 architecture variants are found in Appendix E.

We can further generate very long length-100 sequences with randomly initialized and also pre-trained GPT-2 models and run a simple hypothesis test, demonstrating both randomly initialized and pre-trained models generate lower Kolmogorov complexity sequences on average than a uniform distribution. Over $100{,}000$ samples from each of these three generative distributions and with a one-tailed t-test on the null hypothesis that $\mu(K(S_{\text{GPT}})) \geq \mu(K(S_{\mathcal{U}}))$, where S_{GPT} and $S_{\mathcal{U}}$ respectively denote random sequences generated by the language model or a uniform distribution, we reject this null hypothesis in both randomly initialized and pre-trained models with an extremely low p-value, indicating that language models are indeed more likely to generate simple sequences, pre-trained models even moreso. Further details can be found in Appendix E. We conclude that neural networks for language generation, both trained and randomly initialized, express a bias towards low Kolmogorov complexity which mirrors that of data as demonstrated in Section 3.

Takeaway: Language models, both pre-trained and randomly initialized, prefer to generate low-complexity sequences. As a result, we can use even such randomly initialized models to predict the next element in a sequence, as long as the sequence is low-complexity.

5 Model Selection with a Preference for Simplicity

In typical industrial workflows, practitioners examine their data and select an appropriate learner. For example, if data are linearly separable, then a practitioner might use logistic regression, whereas if the data is more complex, the same practitioner may instead choose a neural network. We can then consider the human model selector and the model they select as a single meta-learner. While early works conjectured that automated meta-learners are ruled out by no free lunch theorems (Giraud-Carrier & Provost, 2005), subsequent works show that model selection can in fact be automated in practice (Vilalta & Drissi, 2002). Giraud-Carrier & Provost (2005) shows that with minimal assumptions, the defeating conclusion of the no free lunch theorem can be escaped by meta-learners. Subsequent works prove the existence of meta-induction algorithms, for selecting amongst all induction methods, which are near-optimal among learners (Schurz, 2008; Schurz & Thorn, 2017). In this section, we argue why in principle, model selection can be automated from the view of complexity.

5.1 Model Selection and Generalization Bounds

When developing a machine learning approach for a given application, it is often helpful to leverage domain knowledge in constructing or choosing the right model for the task. One might start by choosing from architecture families like MLPs, CNNs, GNNs, PointNets, or Transformers and then decide on the appropriate way of featurizing the inputs, possibly incorporating knowledge of data symmetries in via hard-coded equivariances or data augmentations. Even just enumerating the possible models a practitioner would consider and choosing based on their expertise and prior knowledge, the number is not tremendously large. Even if we are extremely generous and suppose that the prac-

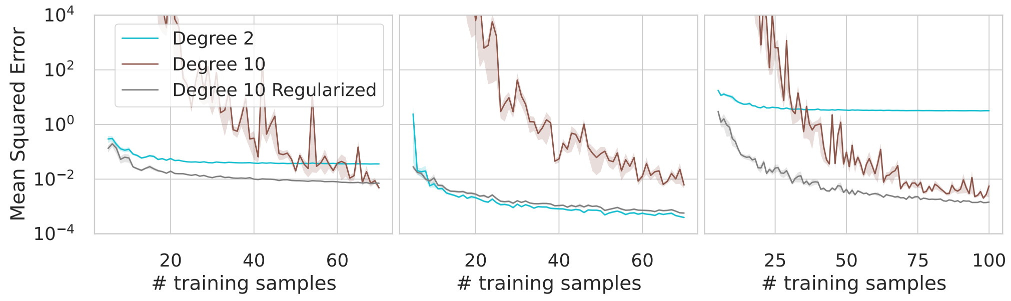


Figure 5: High-order polynomials with a complexity penalty can solve problems at a variety of sample sizes. **Left:** Cosine target function. **Middle:** Degree 2 polynomial target function. **Right:** Degree 10 polynomial target function.

titioner is choosing from 100 million models, we can consider the impractical algorithm of selecting between them via cross validation. While one might expect that such a procedure would overfit, in fact even finite hypothesis bounds show this is not the case. Using cross validation on a validation set of size $n = 20000$ for a classification problem, plugging in a uniform prior $P(h) = 10^{-8}$ to Equation 1, we get that the gap between *validation* and test error will be less than 2.9% with probability greater than 99%. Ultimately, this is the case because we only need a number of data points proportional to the log of the size of the hypothesis space for finite hypothesis bounds.

Considering even a more general class of models, one may consider the number of bits of prior knowledge needed to specify the model architectures like MLPs, CNNs, or GNNs as well as symmetry groups and any other required information. In each of these cases, the model architectures can in fact be expressed with a small number of bits. A near state-of-the-art computer vision model can be expressed in only 280 characters (Trockman & Kolter, 2022) of PyTorch. Similarly, important symmetry groups like translations, rotations, reflections, and other matrix groups can be expressed in only a few lines of code (Finzi et al., 2021) and can be used to encode equivariances or used for augmentation. Therefore, even in selecting from all possible models that can be expressed in that short amount of code, we can expect to generalize with only tens of thousands of data points.

Takeaway: In principle, automating model selection directly via cross validation provably generalizes well across millions of models with only thousands of data points.

5.2 One Model for Big and Small Training Sets

It is commonly believed that small training datasets demand compact neural network architectures, whereas large training datasets can accommodate flexible architectures. Accordingly, practitioners hand select appropriate models for their datasets. In this section, we argue that a single learner can be effective for all data sizes as long as we encode our a priori preference for simplicity. Our prior should prefer simple functions we believe are more likely yet also support a wide variety of functions in case training data rules out our preferences. We begin with a simple illustration.

Polynomial regression. Common intuition dictates that high degree polynomials easily overfit their training data and should be avoided on small training sets. In contrast, low degree polynomials cannot fit complicated functions so they should be avoided when training data is plentiful. However, we demonstrate here that we can rely on lower order terms insofar as they can explain the data while still allowing higher order terms if they are necessary.

To this end, we adopt Tikhonov regularization with a Tikhonov matrix given by $\mathrm{diag}(\{\alpha k^2\}_{k=0}^{d})$, that is we impose an ℓ_2 penalty which increases quadratically with the order of the corresponding monomial. We consider three models: non-regularized degree 2 and degree 10 polynomials as well as a degree 10 polynomial regularized as mentioned above. In Figure 5, we perform regression on the cosine target function (left) and degree 2 (center) and degree 10 (right) polynomial target functions. We confirm that even though a high degree polynomial model performs poorly with few training samples, it eventually outperforms its lower degree counterpart. Meanwhile, a high degree polynomial model with the above Tikhonov regularization outperforms both non-regularized models across most sample sizes on cosine and degree 10 polynomial regression problems, and it notably matches the performance of a degree 2 model even on data generated by a degree 2 polynomial. We include details regarding the example target functions and the data sampling process in Appendix F.

Neural networks. We illustrate a similar concept with neural networks. We consider a small network, GoogLeNet (Szegedy et al., 2015), which performs well on small datasets such as CIFAR-10 and CIFAR-100 (Krizhevsky, 2009) but poorly on larger datasets like ImageNet (Deng et al., 2009). We also consider a large network, ViT-B/16 (Dosovitskiy et al., 2020), which performs significantly worse on CIFAR variants but much better on ImageNet. As in the polynomial regression example above, we can combine these two architectures, specifying our preference for the simpler GoogLeNet to the extent that it fits the training data. To learn on a dataset, we train both models and then take a convex combination of their logits, $c * \text{logits}_{\text{ViT}} + (1 - c) * \text{logits}_{\text{G}}$, controlled by a single parameter c with ℓ_2 regularization in favor of GoogLeNet. In Figure 6, we observe that while GoogLeNet and ViT each have strengths and weaknesses, combining them with a preference for simplicity achieves the best of both worlds. While aggressively restricting our architectures can decrease computational cost, it is unnecessary for generalization. Details and additional experiments with Swin Transformer (Liu et al., 2021) are found in Appendix F.

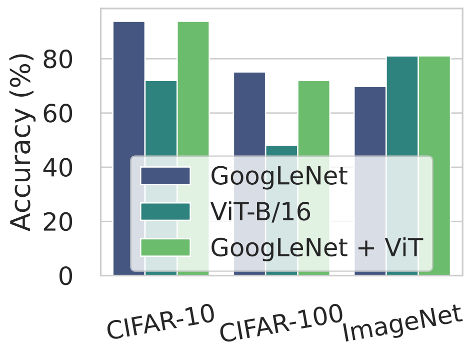


Figure 6: A single learner, which is at least as expressive as a vision transformer, but also prefers simple solutions representable by a small GoogLeNet, can simultaneously solve small and large scale problems.

6 Discussion

While large ML models are highly flexible, we saw in this work that they reliably prefer low Kolmogorov complexity solutions—aligning well with relevant learning problems—despite not being designed with complexity in mind. This observation raises the question: why exactly do neural networks encode such a strong preference for low complexity and how can we tune this preference? Complementing the above observation, we also saw that a single expressive model which simultaneously supports a wide variety of solutions but prefers simple ones can simultaneously solve simple and hard problems over sample sizes. Such learners present clear advantages over the current paradigm in deep learning in which we manually select small constrained architectures or large ones with mild inductive biases, depending on the problem. Keeping this possibility in mind, how can we design expressive yet simplicity-biased models but with affordable computational costs?

References

Pierre Alquier. User-friendly introduction to PAC-Bayes bounds. *arXiv preprint arXiv:2110.11216*, 2021.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Gregory J Chaitin. Information-theoretic limitations of formal systems. *Journal of the ACM (JACM)*, 21(3):403–424, 1974.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pp. 886–893. Ieee, 2005.

Alex Damian, Tengyu Ma, and Jason D Lee. Label noise sgd provably prefers flat global minimizers. *Advances in Neural Information Processing Systems*, 34:27449–27461, 2021.

Stéphane d'Ascoli, Pierre-Alexandre Kamienny, Guillaume Lample, and François Charton. Deep symbolic regression for recurrent sequences. *arXiv preprint arXiv:2201.04600*, 2022.

Giacomo De Palma, Bobak Kiani, and Seth Lloyd. Random deep neural networks are biased towards simple functions. *Advances in Neural Information Processing Systems*, 32, 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. In *International Conference on Machine Learning*, pp. 3318–3328. PMLR, 2021.

Lance Fortnow. Kolmogorov complexity. In *Aspects of Complexity (Short Courses in Complexity from the New Zealand Mathematical Research Institute Summer 2000 Meeting, Kaikoura)*, volume 4, pp. 73–86, 2000.

Zihao Fu, Wai Lam, Anthony Man-Cho So, and Bei Shi. A theoretical analysis of the repetition problem in text generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 12848–12856, 2021.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

Christophe Giraud-Carrier and Foster Provost. Toward a justification of meta-learning: Is the no free lunch theorem a show-stopper. In *Proceedings of the ICML-2005 Workshop on Meta-learning*, pp. 12–19, 2005.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *Proc. Interspeech 2020*, pp. 5036–5040, 2020.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020.

W Ronny Huang, Zeyad Ali Sami Emam, Micah Goldblum, Liam H Fowl, JK Terry, Furong Huang, and Tom Goldstein. Understanding generalization through visualizations. In *"I Can't Believe It's Not Better!"NeurIPS 2020 workshop*, 2020.

David Hume. *Philosophical Essays Concerning Human Understanding*. A. Millar, 1748.

Marcus Hutter. A theory of universal artificial intelligence based on algorithmic complexity. *arXiv preprint cs/0004001*, 2000.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pp. 876–885. Association For Uncertainty in Artificial Intelligence (AUAI), 2018.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*, 2021.

Andrei N Kolmogorov. On tables of random numbers. *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 369–376, 1963.

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Tor Lattimore and Marcus Hutter. No free lunch versus occam's razor in supervised learning. In *Algorithmic Probability and Friends. Bayesian Prediction and Artificial Intelligence*, pp. 223–235. Springer, 2013.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.

Sanae Lotfi, Pavel Izmailov, Gregory Benton, Micah Goldblum, and Andrew Gordon Wilson. Bayesian model selection, the marginal likelihood, and generalization. *Proceedings of the International Conference on Machine Learning*, 2022a.

Sanae Lotfi, Sanyam Kapoor, Marc Finzi, Andres Potapczynski, Micah Goldblum, and Andrew Gordon Wilson. PAC-Bayes compression bounds so tight that they can explain generalization. In *NeurIPS*, 2022b.

David McAllester. Simplified PAC-Bayesian margin bounds. In *Learning theory and Kernel machines*, pp. 203–215. Springer, 2003.

David McAllester. A PAC-Bayesian tutorial with a dropout bound. *arXiv preprint arXiv:1307.2118*, 2013.

David A McAllester. Some PAC-Bayesian theorems. In *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 230–234, 1998.

David A McAllester. PAC-Bayesian Model Averaging. *Proceedings of the 12th Annual Conference on Learning Theory (COLT)*, pp. 164–170, 1999.

Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pp. 165–172, 2013.

Chris Mingard, Joar Skalse, Guillermo Valle-Pérez, David Martínez-Rubio, Vladimir Mikulik, and Ard A Louis. Neural networks are a priori biased towards boolean functions with low entropy. *arXiv preprint arXiv:1909.11522*, 2019.

Andriy Mnih and Geoffrey Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pp. 641–648, 2007.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

Preetum Nakkiran. Turing-universal learners with optimal scaling laws. *arXiv preprint arXiv:2111.05321*, 2021.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5206–5210. IEEE, 2015.

Barak Pearlmutter and Ronald Rosenfeld. Chaitin-kolmogorov complexity and generalization in neural networks. *Advances in neural information processing systems*, 3, 1990.

Arthur Petrosian. Kolmogorov complexity of finite sequences and recognition of different preictal eeg patterns. In *Proceedings eighth IEEE symposium on computer-based medical systems*, pp. 212–217. IEEE, 1995.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

R Bharat Rao, Diana Gordon, and William Spears. For every generalization action, is there really an equal and opposite reaction? analysis of the conservation law for generalization performance. In *Machine Learning Proceedings 1995*, pp. 471–479. Elsevier, 1995.

Samuel Rathmanner and Marcus Hutter. A philosophical treatise of universal induction. *Entropy*, 13(6):1076–1136, 2011.

Cullen Schaffer. A conservation law for generalization performance. In *Machine Learning Proceedings 1994*, pp. 259–265. Elsevier, 1994.

Jurgen Schmidhuber. Discovering neural nets with low kolmogorov complexity and high generalization capability. *Neural networks: the official journal of the International Neural Network Society*, 10(5):857–873, 1997.

Gerhard Schurz. The meta-inductivist's winning strategy in the prediction game: A new approach to hume's problem. *Philosophy of Science*, 75(3):278–305, 2008.

Gerhard Schurz and Paul Thorn. A priori advantages of meta-induction and the no free lunch theorem: A contradiction? In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, pp. 236–248. Springer, 2017.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Ray J Solomonoff. A formal theory of inductive inference. part i. *Information and control*, 7(1): 1–22, 1964.

Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C Bayan Bruss, and Tom Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*, 2021.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Asher Trockman and J Zico Kolter. Patches are all you need? *arXiv preprint arXiv:2201.09792*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18(2):77–95, 2002.

Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *Advances in neural information processing systems*, 33:4697–4708, 2020.

Ian H Witten, Radford M Neal, and John G Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996.

David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

David H Wolpert, William G Macready, et al. No free lunch theorems for search. Technical report, Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

Eric Zelikman, Yuhuai Wu, and Noah D Goodman. Star: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*, 2022.

Hector Zenil, James AR Marshall, and Jesper Tegnér. Approximations of algorithmic and structural complexity validate cognitive-behavioural experimental results. *arXiv preprint arXiv:1509.06338*, 2015.

Wenda Zhou, Victor Veitch, Morgane Austern, Ryan P Adams, and Peter Orbanz. Non-vacuous generalization bounds at the imagenet scale: a PAC-Bayesian compression approach. In *International Conference on Learning Representations*, 2018.

A Universal Induction and Complexity in Deep Learning

Universal induction. Inspired by Kolmogorov complexity, a line of work introduces *universal induction* methods, which prefer low complexity answers (Solomonoff, 1964; Hutter, 2000; Nakkiran, 2021). Notably, Solomonoff induction (Solomonoff, 1964; Rathmanner & Hutter, 2011) places a probability distribution over bitstrings which vanishes exponentially with their Kolmogorov complexity with respect to a universal Turing machine. Then, presented with an observed sequence (e.g. training data) called a prefix and a candidate completion, we can perform inference via $p([\text{prefix}, \text{completion}] \mid \text{prefix})$. In other words, we assign probability to each completion proportional to its probability among sequences which begin with the prefix. Since low-complexity bitstrings are assigned higher probability, we will prefer completions which result in low-complexity completed sequences. Subsequent literature argues that Solomonoff induction explains why no free lunch theorems are practically irrelevant, namely weak assumptions on the structure of learning problems allow for near-optimal universal learners (Lattimore & Hutter, 2013).

Complexity in deep learning. Several works have related Kolmogorov complexity to neural networks. Pearlmutter & Rosenfeld (1990) argues that random initialization and noise in data increase the complexity of neural networks but that ensembling such models reduces complexity in expectation. Schmidhuber (1997) proposes another method for reducing Kolmogorov complexity by instead explicitly searching for simple neural networks and finds improvements in generalization on very small problems where such a search is computationally feasible. Another line of study proves that multi-layer perceptrons with boolean input features are biased towards low-entropy functions, namely ones which classify disproportionately many or few points into the same class (Mingard et al., 2019) or are insensitive to flips in the boolean features (De Palma et al., 2019).

Other notions of simplicity have also been proposed and applied to neural networks, for example involving the implicit bias of SGD (Damian et al., 2021); the connection between flat loss minima with wide-margins decision boundaries and improved generalization (Huang et al., 2020; Izmailov et al., 2018); the tendency of models to rely on simple, compressible features (Geirhos et al., 2020); or the potential for overparameterized networks to be implicitly regularized, leading to capacity control (Neyshabur et al., 2014). The marginal likelihood, or the probability that a random sample from the prior generates the data, is a natural statement of Occam's razor or the idea that simple models should be preferred (Lotfi et al., 2022a). Diffuse priors which spread mass out around the diverse high-complexity solutions are unlikely to generate observed data, so effective models admit a far higher marginal likelihood on naturally occurring labelings compared to random ones (Wilson & Izmailov, 2020).

B A Kolmogorov No Free Lunch Theorem Proof

Rewriting Equation (2), we have

$$\text{CE}(p) \geq \frac{\ln 2}{n} \left(K(Y|X) - K(p) - 2\log_2 K(p) - c \right).$$

Note that by simply counting all possible programs taking input X, there are less than 2^{k+1} labelings Y with $K(Y|X) \leq k$. Note that there are C^n distinct labelings, from which we are drawing uniformly. So that

$$\begin{aligned}
\mathbb{P}(K(Y|X) > n\log_2 C - m) &= 1 - \mathbb{P}(K(Y|X) \leq n\log_2 C - m) \\
&\geq 1 - \mathbb{P}(K(Y|X) \leq \lceil n\log_2 C \rceil - m) \\
&\geq 1 - \frac{2^{\lceil n\log_2 C\rceil - m + 1}}{C^n} \\
&\geq 1 - 2^{2-m}.
\end{aligned}$$

Alternatively, with probability at least $1-\delta$,

$$K(Y|X) > n\log_2 C - \log_2 \frac{1}{\delta} - 3.$$

Thus, with probability at least $1-\delta$, we have for every classifier p,

$$\text{CE}(p) \geq \ln C - \frac{\ln 2}{n} \left(K(p) + 2\log_2 K(p) + \log_2 \delta + c \right).$$

C PAC-BAYES COMPRESSION EXPERIMENTAL DETAILS

For the OpenML tabular classification datasets, we preprocess them first by balancing the classes, subsampling all classes down to the number of examples of the least likely class. This way, when compressing the datasets, any result achieved is nontrivial in contrast with a very class imbalanced dataset. We heavily follow the compression method of (Lotfi et al., 2022b), including the small 9 convolutional architecture which they use to generate their bounds. When cramming the tabular data into this convnet, we combine numerical features with one hot encoded categorical features and then pack these into the pixels of a 1 channel image, using however large an image as necessary to fit each of the different features inside.

With respect to the sizes of the random subspaces that we train the compressed models in, we consider 250,500,1000, and 2000. For tabular label compression, we employ a 2 hidden layer MLP with hidden dimension $k = 192$, and we consider the same 250,500,1000, and 2000 values for subspace dimension. We train for 80 epochs with 20 epochs of quantization at a batch size of 512 using Adam at lr= 3×10^{-4}. For image classification label compression, we use the 9-layer convnet with subspace dimensions 2000, 3000, 5000, and we train for 80 epochs using SGD at learning rate 0.1 and quantize for the remaining 20 epochs, at a batch size of 50. For calculating the length of the code for model architecture and decompressor, we need only the implementation of the model, the arithmetic decoder, and the loading of the quantized values. Removing wasted bits, we minified the python file, leading to a size of approximately 2.5KB.

D GPT-3 EXPERIMENTAL DETAILS

To feed sequences into a model, we split up sequence elements into individual byte-pair encoding tokens corresponding to their decimal digits, and we place comma tokens between sequence elements as delimiters, also beginning every input with an `<|endoftext|>` token. We choose to use the byte-pair encoding of decimal digits with a space inserted before the digit, e.g. ' 0' as this is known to enhance the ability of language models to perform arithmetic (Zelikman et al., 2022). For example, the sequence $10, 11$ will be split up into ['`<|endoftext|>`', ' 1', ' 0', ',', ' 1', ' 1'], and each element of the list is tokenized individually. Then, the log-probability of a sequence is given by the sum of the log-probabilities corresponding to the correct decimal digits in their respective slots of the model's output. Note that various sequences will contain different numbers of decimal digits, and the sequence's log-probability will decrease with every token. Therefore, in order for fair comparison, we limit all sequences to 30 decimal digit tokens and truncate there.

E SEQUENCE GENERATION AND COMPLETION WITH RANDOMLY INITIALIZED LANGUAGE MODELS

For these experiments, we use Huggingface[2] GPT-2 architectures and pre-trained checkpoints. In order to estimate the probabilities assigned by randomly initialized language models to each bitstring, we generate one million random sequences, ensuring that many instances of each bitstring are generated as there are only $2^{10} = 1024$ bistrings of length 10.

We also include plots with other sizes of GPT-2 architectures in Figure 7 and Figure 8.

We also include the hypothesis test referenced in Section 4.3. For this experiment, we generate 100,000 length-100 sequences from randomly intialized GPT-2 variants, pre-trained GPT-2 variants, and a uniform distribution. We then perform a one-sided t-test on the null hypothesis that $\mu(K(S_{\text{GPT}})) \geq \mu(K(S_{\mathcal{U}}))$, for both initialized and pre-trained models. Table 1 contains the resulting sample means, t-statistics and p-values. In all cases, we reject the null hypothesis with very low p-values, indicating that language models do prefer to generate low-complexity sequences. Notably, pre-trained language models exhibit an increased simplicity bias, and bigger and better language models even moreso.

[2] https://huggingface.co/

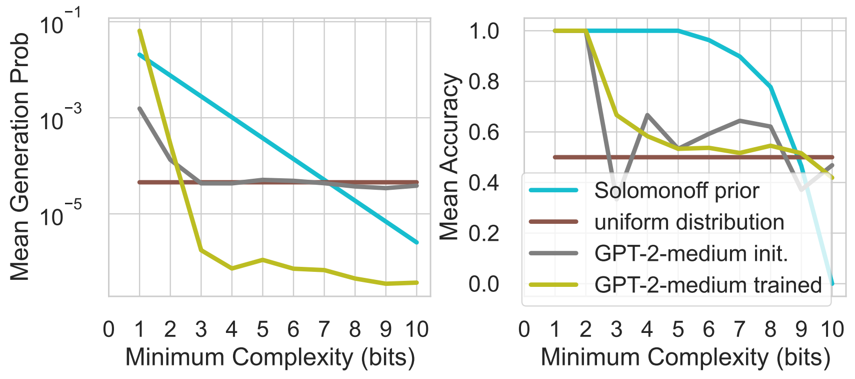


Figure 7: Randomly initialized GPT-2 Medium prefers low-complexity sequences generated by bitstring repetition. Left: Average log-probability of sequences by complexity. **Right:** Average accuracy.

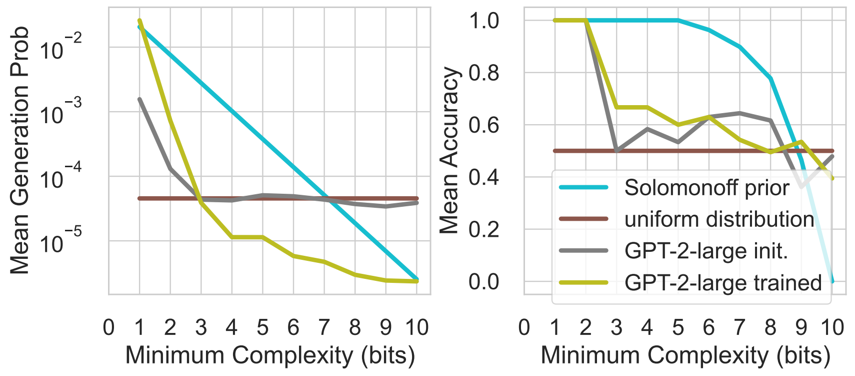


Figure 8: Randomly initialized GPT-2 Large prefers low-complexity sequences generated by bitstring repetition. Left: Average log-probability of sequences by complexity. **Right:** Average accuracy.

Table 1: Hypothesis test for language model simplicity bias. t-tests are one-sided, and p-values are rounded to 4 digits. We also report the mean Kolomogorov complexity of sequences generated by each language model and a uniform distribution.

Model	$\overline{K(S_{\text{GPT}})}$	t-statistic	p-value
Uniform Distribution	98.36	-	-
GPT-2 Base Initialized	98.00	-39.95	0.0000
GPT-2 Medium Initialized	97.99	-40.91	0.0000
GPT-2 Large Initialized	98.00	-40.11	0.0000
GPT-2 Base Trained	60.81	-255.17	0.0000
GPT-2 Medium Trained	48.41	-325.16	0.0000
GPT-2 Large Trained	46.34	-342.80	0.0000

F Big and Small Training Sets

Polynomial regression. We choose three example target functions on which to perform regression: $\cos(\frac{3\pi}{2}x)$, x^2, and $-36x + 49x^5 - 14x^7 + x^{10}$. Training data is randomly drawn from a uniform distribution over the unit interval, and we add noise to training labels from $\mathcal{N}(0, 0.1)$. In each case, for each dataset size, we average the mean squared error over 100 randomly sampled training sets. For Tikhonov regularized polynomial regression on the cosine and degree 2 polynomial target functions, we use $\alpha = 0.01$, and we use $\alpha = 0.001$ for regression on the degree 10 polynomial target function.

Image classification with neural networks. For ImageNet trained models, we employ publicly available checkpoints from `torchvision`[3]. We train models on CIFAR-10 and CIFAR-100 for 200 epochs with initial learning rate 0.1 and cosine annealing along with horizontal flip and random crop augmentations. We use SGD with momentum 0.9 and batches of size 128. All CIFAR images are rescaled to 224×224 so that we can use an identical model for ImageNet and CIFAR data. In order to learn the parameter c controlling the convex combination of models, we perform 10 epochs of training, where the models' parameters are frozen, and we apply weight decay with coefficient 10^{-5}. We learn the parameter c using SGD with momentum 0.9 and batch size 128, initial learning rate 0.1, and cosine annealing.

Table 2: Combinations of large and small architectures form single models that achieve high test accuracy on all dataset sizes. "GoogLeNet + ViT" denotes a model formed as a convex combination of the logits of the two constituent models with weight decay on the parameter c controlling the convex combination which multiplies the logits of the larger model, ensuring that the small model is prefered as long as it fits the data.

Model	CIFAR-10	CIFAR-100	ImageNet
GoogLeNet	93.840 %	75.160 %	69.778 %
ViT-B/16	72.020 %	48.140 %	81.072 %
Swin-B	74.710 %	64.200 %	83.582 %
GoogLeNet + ViT	93.860 %	71.990 %	81.090 %
GoogLeNet + Swin	93.760 %	75.360 %	83.150 %

[3] `https://pytorch.org/vision/stable/index.html`