Model-agnostic Measure of Generalization Difficulty

Akhilan Boopathy [1] **Kevin Liu** [1] **Jaedong Hwang** [1] **Shu Ge** [1] **Asaad Mohammedsaleh** [1] **Ila Fiete** [1]

Abstract

The measure of a machine learning algorithm is the difficulty of the tasks it can perform, and sufficiently difficult tasks are critical drivers of strong machine learning models. However, quantifying the generalization difficulty of machine learning benchmarks has remained challenging. We propose what is to our knowledge the first model-agnostic measure of the inherent generalization difficulty of tasks. Our *inductive bias complexity* measure quantifies the total information required to generalize well on a task minus the information provided by the data. It does so by measuring the fractional volume occupied by hypotheses that generalize on a task given that they fit the training data. It scales exponentially with the intrinsic dimensionality of the space over which the model must generalize but only polynomially in resolution per dimension, showing that tasks which require generalizing over many dimensions are drastically more difficult than tasks involving more detail in fewer dimensions. Our measure can be applied to compute and compare supervised learning, reinforcement learning and meta-learning generalization difficulties against each other. We show that applied empirically, it formally quantifies intuitively expected trends, e.g. that in terms of required inductive bias, MNIST $<$ CIFAR10 $<$ Imagenet and fully observable Markov decision processes (MDPs) $<$ partially observable MDPs. Further, we show that classification of complex images $<$ few-shot meta-learning with simple images. Our measure provides a quantitative metric to guide the construction of more complex tasks requiring greater inductive bias, and thereby encourages the development of more sophisticated architectures and learning algorithms with more powerful generalization capabilities.

[1]Massachusetts Institute of Technology. Correspondence to: Akhilan Boopathy <akhilan@mit.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

1. Introduction

Researchers have proposed many benchmarks to train machine learning models and test their generalization abilities, from ImageNet (Krizhevsky et al., 2012) for image recognition to Atari games (Bellemare et al., 2013) for reinforcement learning (RL). More complex benchmarks promote the development of more sophisticated learning algorithms and architectures that can generalize better.

However, we lack rigorous and quantitative measures of the generalization difficulty of these benchmarks. Generalizing on a task requires both *training data* and a model's *inductive biases*, which are *any* constraints on a model class enabling generalization. Inductive biases can be provided by a *model designer*, including the choice of architecture, learning rule or hyperparameters defining a model class. While prior work has quantified the training data needed to generalize on a task (*sample complexity*), analysis of the required inductive biases has been limited. Indeed, while the concept of inductive bias is widely used, it has not been thoroughly and quantitatively defined in general learning settings.

In this paper, we develop a novel information-theoretic framework to measure a task's *inductive bias complexity*, the information content of the inductive biases. Just as sample complexity is a property inherent to a model class (without reference to a specific training set), inductive bias complexity is a property inherent to a training set (without reference to a specific model class). To our knowledge, our measure is the first quantification of inductive bias complexity. As we will describe, our measure quantifies the fraction of the entire hypothesis space that is consistent with inductive biases for a task given that hypotheses interpolate the training data; see Figure 1 for an illustration and Definition 3.1 for a formal definition. We use this inductive bias complexity quantification to assess a task's generalization difficulty; we hope our measure can guide the development of tasks requiring greater inductive bias. As one concrete, but widely-applicable, suggestion, we find that adding Gaussian noise to the inputs of a task can dramatically increase its required inductive bias. We summarize our contributions as [1]:

[1]Our code is released at: https://github.com/FieteLab/inductive-bias-complexity

- We provide a formal, information-theoretic definition of *inductive bias complexity* that formalizes intuitive notions of inductive bias.
- We develop a novel and, to our knowledge, first quantifiable measure of inductive bias complexity, which we propose using as a measure of the generalization difficulty of a task. This measure also allows us to quantify the relative inductive biases of different model architectures applied to a task.
- We propose a practical algorithm to estimate and compare the inductive bias complexities of tasks across the domains of supervised learning, RL and few-shot meta-learning tasks. Empirically, we find that 1) partially observed RL environments require much greater inductive bias compared to fully observed ones and 2) simple few-shot meta-learning tasks can require much greater inductive bias than complex, realistic supervised learning tasks.

2. Related Work

2.1. Model Oriented Generalization Difficulty

The generalizability of machine learning models is traditionally quantified with *learning curves*, which relate the generalization error on a task to amount of training data (i.e. sample complexity) (Amari, 1993; Cortes et al., 1994; Hestness et al., 2017; Murata et al., 1992). Generalization error can be bounded by model class capacity measures, including Rademacher complexity (Koltchinskii & Panchenko, 2000) and VC dimension (Blumer et al., 1989), with simpler model classes achieving lower generalization error. While valuable, these measures often provide only loose, non-specific generalization bounds on particular tasks because they typically do not leverage a task's structure.

More recently proposed data-dependent generalization bounds yield tighter bounds on generalization error (Jiang et al., 2021; Kawaguchi et al., 2022; Lei et al., 2015; Negrea et al., 2019; Raginsky et al., 2016). These measures leverage the properties of a dataset to show that particular model classes will generalize well on the dataset. Relatedly, recent work on neural network scaling laws (Bahri et al., 2021; Hutter, 2021; Sharma & Kaplan, 2022) has modeled learning as kernel regression to show among other results that sample complexity scales *exponentially* with the intrinsic dimensionality of data. Our approach is aligned in the spirit of this line of work; however, instead of leveraging the structure of a task to bound generalization error or sample complexity, we bound the *inductive bias complexity* of a task; see Figure 1. We believe inductive bias complexity is a more appropriate measure of a task's difficulty compared to sample complexity. Notably, sample complexity assumes a specific model class; it is not model-agnostic. Thus, sample complexity only quantifies the contribution of training data towards solving a task given a particular model class; it does *not* quantify the difficulty of choosing an appropriate model class (e.g. a model architecture).

2.2. Information-theoretic Measures of Generalization Difficulty

Recent works have analyzed the properties of neural networks using information theory; for instance, information bottleneck can be used to extract well generalizing data representations (Saxe et al., 2019; Shamir et al., 2008; Shwartz-Ziv & Tishby, 2017). DIME (Zhang et al., 2020) uses information theory to estimate task difficulty, bounding the best case error rate achievable by *any* model on a dataset given the underlying data distribution. Achille et al. (2021) defines task difficulty based on the Kolmogorov complexity (Kolmogorov, 1965) required to model the relationship between inputs and outputs. Although these measures can be related to generalization, they do not directly quantify the information needed to generalize. Alternatively, generalization difficulty can be expressed as the amount of information required to perform a task in addition to any training data provided to a learning system (Chollet, 2019). This aligns with other literature assessing the *prior knowledge* contained in a learning system whether through specific abilities (Hernández-Orallo, 2016), biases (Haussler, 1988), or model architectures (Du et al., 2018; Li et al., 2021). In this work, we take a similar approach, defining generalization difficulty as the amount of information in inductive biases required on top of training data in order to solve a task within the hypothesis space.

3. Measuring Information Content of Inductive Biases

It is well known by the No Free Lunch theorem (Wolpert, 1996) that no learning algorithm can perform well on *any* task; only learning algorithms that possess appropriate inductive biases relevant to a particular task(s) can be expected to generalize well given a finite amount of data. Thus, generalizing on a task requires both training data and inductive biases (i.e. a choice of model class) (see Figure 1 (b)). In Section 3.1, we formally define *inductive bias complexity* as the amount of inductive bias required to generalize. In Section 3.2, we exactly bound this quantity in terms of the desired error rate of a task and distance between training and test distributions. In Section 3.3, we further approximate the bound to produce a practically computable estimate. Table 1 summarizes the notation used throughout this section.

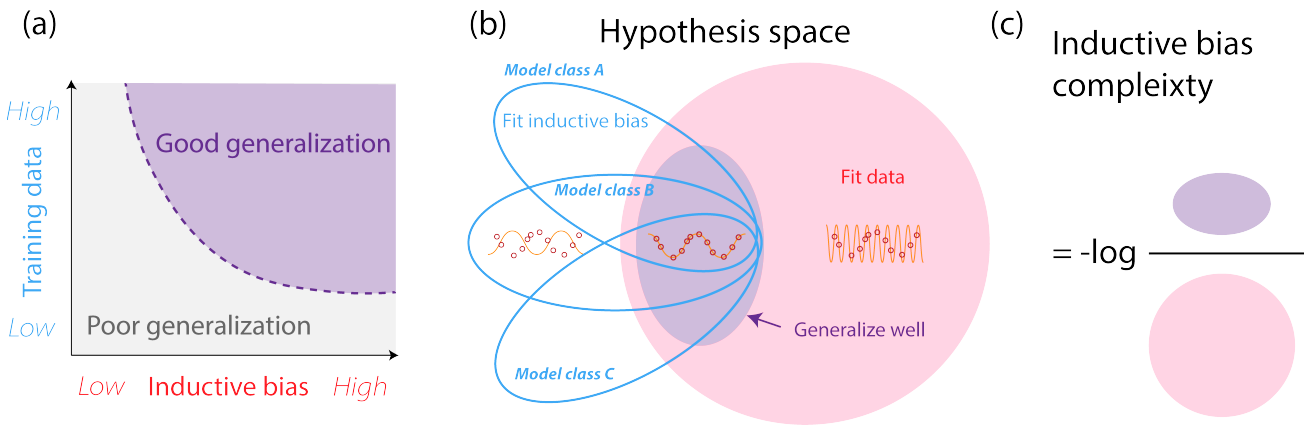


Figure 1: Dividing the hypothesis space of a task. (a) the trade-off between the amount of training data required (sample complexity) vs. the amount of inductive bias required (inductive bias complexity) to generalize well. Achieving a lower sample complexity for a fixed generalization performance target requires higher inductive bias complexity to further restrict the hypothesis space. While many papers have quantified sample complexity, we quantify inductive bias complexity. (b) categorization of hypotheses by whether or not they fit the training data (red region) or contain particular sets of inductive biases; different blue ovals correspond to different model classes (i.e. different sets of inductive biases). Together with training data, specific choices of model class can restrict hypotheses to a well-generalizing region; purple indicates well-generalizing hypotheses. Both training data and inductive biases are required to generalize well. In practice, well-generalizing hypotheses occupy a very small fraction of the region of hypothesis space fitting the training data. (c) how inductive bias complexity is computed: we quantify inductive bias complexity as negative log of the fraction of hypotheses that generalize well among the hypotheses that fit the training data.

Table 1: Table of symbols.

Symbol	Meaning
$f(\cdot;\theta)$	Function from the hypothesis space, parameterized by θ
p	Input distribution, denotes the test distribution
q	Input distribution, denotes the training distribution
$e(\theta, p)$	Error of $f(\cdot;\theta)$ on input distribution p
$\mathcal{L}$	Loss function
ε	Desired test error
$\tilde{I}$	Inductive bias complexity
Θ_q	Set of hypotheses that fit the training dataset
$L_{\mathcal{L}}$	Lipschitz constant for $\mathcal{L}$
L_f	Constant used in second-order condition on f
m	Intrinsic dimensionality of the data
k	Number of submanifolds
r	Radius of the sphere on which datapoints lie
d	Dimensionality of the model output $f(x;\theta)$
f_i	A single component of f
b	Upper bound on $\|f(x;\theta)\|$
n	Size of training dataset
M	Maximum frequency of eigenfunctions considered
δ	Spatial resolution of f; equal to $2\pi r/M$
K	Maximum integer satisfying $\sqrt{K(K+m-1)} \leq M$
E	# of eigenfunctions with frequency $\leq M$; $O(M^m)$

3.1. Formally Defining Inductive Bias Complexity

We formally relate inductive biases and generalization using the concept of a *hypothesis space*. Hypotheses may or may not satisfy a training set or different sets of inductive biases (see Figure 1 (b)); different sets of inductive biases (e.g. different model architectures) correspond to different subsets of the hypothesis space. Generalization fundamentally requires both inductive biases and training data to specify a set of well-generalizing hypotheses. Note that there is a trade-off between the amount of training data and inductive bias required to generalize (see Figure 1 (a)). We aim to quantify the minimum level of inductive bias required for generalization given a fixed amount of training data. We call this measure the *inductive bias complexity* (see Figure 1 (c)). Just as sample complexity quantifies the amount of training data required to generalize given a fixed inductive bias (i.e. a fixed model class), inductive bias complexity measures the amount of inductive bias required given a fixed amount of training data.

We begin by defining a (very broad and general) hypothesis space, consisting of functions $f(x;\theta)$ parameterized by a vector random variable θ. The hypothesis space is *not* the same as the model class (e.g. neural networks) used to solve a task. Instead, the hypothesis space is ideally a massive space that encompasses all possible model classes that might reasonably be used to solve a task. We view the

hypothesis space constructed here simply as a mathematical tool to analyze the properties of the task itself rather than relating to the actual models trained on a task. We assume that the true (target) function to be learned is expressible within this large hypothesis space, and we define it to be given by $f^*(x) = f(x; \theta^*)$. The generalization error of a specific θ and input distribution p is denoted $e(\theta, p) = \hat{e}(f(\cdot; \theta), p)$. We aim to find a hypothesis θ achieving generalization error below a threshold ε: $e(\theta, p) \leq \varepsilon$. Although this notation resembles supervised learning, our framework incorporates many different learning scenarios including unsupervised, reinforcement and meta-learning as described in Appendix A.

We use a Bayesian perspective to quantify the amount of *information* needed to specify a well-generalizing set of hypotheses; this will be our measure of inductive bias complexity. Specifically, we assume hypotheses θ are random variables, and we assume a uniform prior probability distribution over hypotheses in the hypothesis space (which can generally be made to hold by reparameterizing). Next, we observe a set of training data that is consistent with only a set of hypotheses in the hypothesis space. Thus, the training data induces a new posterior distribution over the hypotheses. Finally, inductive bias complexity will be related to the probability that a hypothesis drawn from this posterior distribution is in the well-generalizing set of hypotheses: if generalization has a low probability under the posterior, greater inductive bias is required. Thus, our definition of inductive bias complexity can be viewed as the additional *information* required to specify the well-generalizing hypotheses *beyond any task-relevant information provided by training data.*

Formally, we define inductive bias complexity $\tilde{I}$ as follows:

Definition 1. Suppose a probability distribution is defined over a set of hypotheses, and let random variable θ correspond to a sample from this distribution. Given an error threshold ϵ which models are optimized to reach on the training set, the inductive bias complexity required to achieve error rate ε is on a task

$$\tilde{I} = -\log \mathbb{P}(e(\theta, p) \leq \varepsilon \mid e(\theta, q) \leq \epsilon) \tag{1}$$

Observe that this is exactly the information content of the probabilistic event that a hypothesis that fits the training set also generalizes well. This formalizes the intuitive notion that inductive bias corresponds to the difficulty of finding well-generalizing models among the ones that fit the training data. For ease of analysis, we define "fitting" the training set as perfectly interpolating it (setting $\epsilon = 0$). A well-generalizing hypothesis, achieving a low but non-zero error rate, might *not* interpolate the training set. Nevertheless, the regime of perfect interpolation can be relevant for training overparameterized deep networks; we leave other regimes as future work and briefly outline how to extend to these regimes in Appendix G.

Importantly, our definition of inductive bias complexity *does not specify a specific form of inductive bias*. Multiple sets of inductive biases (e.g. different model architectures) may yield the same level of generalization as suggested by Figure 1(b). Our measure quantifies the amount of information that *any* set of inductive biases must provide to generalize. Moreover, our definition of inductive bias complexity is distinct from the *expressivity of a model class* (i.e. the volume of hypotheses spanned by a model class). Inductive bias complexity measures how constrained a model class must be *as relevant to generalizing on a particular task*; by contrast, expressivity is a measure specific to a model class that is not necessarily task-dependent. Thus, a more expressive (i.e. less constrained) model class may provide more or less inductive bias on a particular task depending on the relevance of the increased expressivity to the task. For instance, we find experimentally in Section 4 that certain vision transformer architectures outperform convolutional neural networks (CNNs) on ImageNet classification (and thus more inductive bias) despite transformers arguably being a less constrained model class in the sense that their spatial invariance constraints may be weaker than for CNNs.

3.2. An Exact Upper Bound on Inductive Bias Complexity

We are able to provide an exact upper bound on the inductive bias complexity given above, under some technical assumptions. These assumptions may not hold in all settings; nevertheless, we believe they hold in many common settings (e.g. when the loss function is a square loss and the model f is twice differentiable with bounded inputs and parameters). Specifically, we assume that the input, parameter, and output spaces are equipped with standard distance metrics $d(\cdot, \cdot)$ and norms $\| \cdot \|$. We assume that the loss function is shift-invariant in the sense that $\mathcal{L}(y + f^*(x_2) - f^*(x_1), x_2) = \mathcal{L}(y, x_1)$ for all y, x_1, x_2; this assumption is satisfied by a squared loss function since $||y + f^*(x_2) - f^*(x_1) - f^*(x_2)||_2^2 = ||y - f^*(x_1)||_2^2$. We also assume a Lipschitz constant $L_{\mathcal{L}}$ on the loss function $\mathcal{L}$ which corresponds to the sensitivity of the loss function with respect to perturbations in the predicted output; we expect this sensitivity to be small for most realistic problems. Finally, we assume second-order condition on the model f: $\|f(x_1; \theta_1) - f(x_1; \theta_2) - f(x_2; \theta_1) + f(x_2; \theta_2)\| \leq L_f d(x_1, x_2) d(\theta_1, \theta_2)$. L_f intuitively corresponds to the joint sensitivity of the model with respect to input x and hypothesis parameterization θ. If slightly perturbing the model does not significantly affect the local mapping between the input and output, then L_f will be small. We also expect this to be a reasonable assumption for many problems, although there may be models (such as those with discontinuities) for

which this assumption does not hold. Appendix C further discusses the validity of these assumptions.

Finally, let p denote the test distribution, q denote the training distribution, and $W(\cdot,\cdot)$ denote the 1-Wasserstein distance. Additionally define $\Theta_q = \{\theta \mid e(\theta, q) = 0\}$ to be the set of hypotheses that perfectly interpolate the training dataset.

Theorem 2. *Suppose that the loss function satisfies $\mathcal{L}(y, x) \geq 0$, and equality holds if and only if $y = f^*(x)$. Suppose that the loss function is invariant to shifts in x in the sense that $\mathcal{L}(y + f^*(x_2) - f^*(x_1), x_2) = \mathcal{L}(y, x_1)$ for all y, x_1, x_2. Suppose $\mathcal{L}$ has a Lipschitz constant $L_{\mathcal{L}}$:*

$$\mathcal{L}(y + \delta, x) - \mathcal{L}(y, x) \leq L_{\mathcal{L}} \|\delta\| \tag{2}$$

for all y, δ, x. We also assume a second-order condition on f for some constant L_f:

$$\|f(x_1;\theta_1) - f(x_1;\theta_2) - f(x_2;\theta_1) + f(x_2;\theta_2)\| \leq L_f d(x_1, x_2) d(\theta_1, \theta_2) \tag{3}$$

for all $x_1, x_2, \theta_1, \theta_2$. Then we have the following upper bound on the inductive bias complexity:

$$\tilde{I} \leq -\log \mathbb{P}\left(d(\theta, \theta^*) \leq \frac{\varepsilon}{L_{\mathcal{L}} L_f W(p, q)} \mid \theta \in \Theta_q \right), \tag{4}$$

The bound depends on the following key quantities: desired error rate ε, interpolating hypothesis space Θ_q and 1-Wasserstein distance between the training and test distributions $W(p, q)$. See Appendix B for a proof. Observe that the proof is quite general: it does not rely on any specific assumptions on the hypothesis space, nor does it assume any relationship between the test and training distributions.

3.3. Empirical Inductive Bias Complexity Approximation

We now apply a series of approximations on the general bound of Theorem 2 to achieve a practically computable estimate. We present this approximation in the setting of supervised classification, but note that it can be applied to other settings as we show in our experiments (see Appendix D for full details on our empirical computation). As we propose the first measure of inductive bias complexity, our focus is on capturing general trends (scalings) rather than producing precise estimates. We hope future work may further refine these estimates or further relax our assumptions.

Supervised Classification Setting We assume data points x lie on d non-overlapping m-spheres of radius r, where each sphere corresponds to a class. Furthermore, we assume that the test distribution p is uniform over all the m-spheres and the training distribution q is the empirical distribution of n points drawn i.i.d. from p. This approximation captures key characteristics of data distributions in actual classification tasks: 1) bounded domain, 2) similar density over all points, 3) non-overlapping distributions for different classes. We assume the dimensionality of $f(x;\theta)$ is d, the number of classes. Let b be a constant such that $\|f(x;\theta)\| \leq b$ for all x, θ. Let δ be the "spatial resolution" of f, defined such that perturbing the input x by amounts less than δ does not significantly change the output $f(x;\theta)$ (i.e. δ is the minimum change that f is sensitive to). In our case of supervised classification, δ corresponds to the distance between classes.

Parametrization of Hypotheses In order to compute Equation (4), we must make a specific choice of parameterization for $f(x;\theta)$. Recall from Section 3.1 that we wish to select a broad and general hypothesis space encompassing all model classes practically applicable to a task. Hypotheses that we may *not* want to consider include functions which are sensitive to changes on x below the task relevant resolution θ. Thus, we consider only bandlimited functions with frequencies less than $M = 2\pi r/\delta$. Importantly, this choice of hypothesis space parameterization is *not* related to the model classes used to actually train on a task; instead, it depends only on the properties of the task.

More formally, to construct a basis of functions below a certain frequency on a sphere, we use the approach in McRae et al. (2020), which parameterises functions as a linear combination of eigenfunctions of the *Laplace-Beltrami operator* on the manifold. The Laplace-Beltrami operator is a generalization of the Laplacian to general manifolds; it takes a function defined on a manifold and outputs a measure of the function's curvature at each point on the manifold. Eigenfunctions of this operator are analogous to Fourier basis functions in Euclidean space, with the corresponding eigenvalues corresponding to the eigenfunction's local curvature (analogous to frequency). With the appropriate choice of frequency cutoff M, we may form a very large and general hypothesis space that simultaneously encompasses deep neural networks among other commonly-used model classes, while being finite-dimensional and excluding any with components that are unreasonably curved (i.e. with too high-frequency components). See Figure 6 for a visualization in the Euclidean case. We also highlight that a different choice of basis may lead to a hypothesis space with different properties: our choice explicitly excludes all high-frequency functions, but an alternative choice of basis may include functions with high-frequency components. Moreover, even with our choice of basis, the choice of frequency cutoff M has a critical effect on the size of the hypothesis space and resulting inductive bias complexity (see Appendix F.5).

Recall that we aim to construct a set of basis functions mapping from d m-spheres to $\mathbb{R}^d$. We will use a separate set of basis functions for each m-sphere and each dimen-

sion of f. For each nonnegative integer k, the Laplace-Beltrami operator on an m-sphere has $\binom{m+k}{m} - \binom{m+k-2}{m}$ eigenfunctions with frequency $\sqrt{k(k+m-1)}$ (Dai & Xu, 2013). Taking K to be the maximum integer k such that $\sqrt{k(k+m-1)} \leq M$, there are

$$E = \sum_{k=0}^{K} \binom{m+k}{m} - \binom{m+k-2}{m} = \binom{m+K-1}{m} + \binom{m+K}{m} \quad (5)$$

eigenfunctions with frequency at most M. Note that for large M, E scales as $O(M^m)$ since K scales as M. Since each coefficient has a real and imaginary part, each 1-dimensional component of the d-dimensional function f on *each of the* d *m-spheres* can be parameterized by $2E$ basis functions. The total number of basis functions parameterizing the full d-dimensional function for a single m-sphere is $2dE$. Over all d m-spheres, the total number of basis functions is $2d^2E$: this is the dimensionality of θ.

Each training point constrains the dimensionality of the possible values of θ by d because given any hypothesis, to fit any new training point with output dimensionality d, in general only d dimensions of the hypothesis need to be modified. In other words, being able to independently control d dimensions of variation in the hypothesis space is sufficient to control the d dimensional output of the hypothesis on a new point. This can be understood concretely in a linear setting: suppose hypotheses $f(x;\theta) \in \mathbb{R}^d$ are constructed as $f(x;\theta) = A\theta$ for some full-rank matrix A with dimensionality $d \times 2d^2E$. Then, in order to fit a point (x, y), θ must satisfy $y = A\theta$, which constrains θ to a $2d^2E - d$ dimensional space (specifically, θ is allowed to move in the null space of A). Therefore, the dimensionality of Θ_q is $2d^2E - nd$.

Next, we obtain a bound on $\|\theta\|$. If the parameters of a component f_k are θ_k, Parseval's identity (Stein & Shakarchi, 2011) states that $\|\theta_k\|^2$ equals the average value of $\|f_k(x;\theta_k)\|^2$, which is bounded by b^2. Since there are d^2 components, we have $\|\theta\| \leq bd$.

Next, to approximate the volume of Θ_q, we must make an assumption on the shape of Θ_q. Given that we have a bound on θ, we simply approximate Θ_q as a disk of dimensionality $2d^2E - nd$ and radius bd; note that the disk is embedded in a $2d^2E$ dimensional space. This disk approximation retains key properties of interpolating manifolds (low-dimensionality and boundedness) while being analytically tractable; see Appendix C.3 for further discussion.

The set $\left\{\theta \in \Theta_q \mid d(\theta, \theta^*) \leq \frac{\varepsilon}{L_{\mathcal{L}} L_f W(p,q)}\right\}$ is a disc with radius $\frac{\varepsilon}{L_{\mathcal{L}} L_f W(p,q)}$. Therefore,

$$\mathbb{P}\left(d(\theta, \theta^*) \leq \frac{\varepsilon}{L_{\mathcal{L}} L_f W(p,q)} \mid \theta \in \Theta_q\right) \approx \left(\frac{\frac{\varepsilon}{L_{\mathcal{L}} L_f W(p,q)}}{bd}\right)^{2d^2E - nd}. \quad (6)$$

To arrive at this approximation, we set the hypothesis space as a linear combination of eigenfunctions of the Laplace-Beltrami operator on the manifold, and then approximated volumes in the hypothesis space as balls. Since we do not make specific assumptions on the manifold of x to construct our hypothesis space, we expect the overall scaling of our result to hold for general tasks.

Approximating Wasserstein Distance We obtain a simple scaling result for the Wasserstein distance $W(p, q)$, the expected transport distance in the optimal transport between p and q. In the optimal transport plan, each training point is associated with a local region of the sphere on which it lies; the plan moves probability density between each point and the local region around each point. These regions must be equally sized, disjoint, and tile the entire region of all the spheres. Although the regions' shapes depend on the specific distribution q, if q is near uniform, we expect these regions to be similarly shaped.

We approximate these regions as hypercubes because they are the simplest shape capable of tiling m-dimensional space. In order to fill the full area $d \cdot \frac{2\pi^{(m+1)/2}}{\Gamma\left(\frac{m+1}{2}\right)} r^m$ of all spheres, the n hypercubes must have side length

$$s = \left(\frac{d}{n} \cdot \frac{2\pi^{(m+1)/2}}{\Gamma\left(\frac{m+1}{2}\right)} r^m\right)^{1/m} = rn^{-1/m} \left(\frac{2\pi^{(m+1)/2} d}{\Gamma\left(\frac{m+1}{2}\right)}\right)^{1/m}. \quad (7)$$

The expected distance between two randomly chosen points in a hypercube is at most $s\sqrt{m/6}$ (Anderssen et al., 1976). We use this distance as an approximation of the optimal transport distance between each point and its associated local hypercube. The final $W(p, q)$ can then simply be approximated as the optimal transport distance corresponding to each point:

$$W(p,q) \approx s\sqrt{\frac{m}{6}} = rn^{-1/m}\sqrt{\frac{m}{6}} \left(\frac{2\pi^{(m+1)/2} d}{\Gamma\left(\frac{m+1}{2}\right)}\right)^{1/m} \quad (8)$$

Note that this expression scales as $O(rn^{-1/m})$. This scaling is intuitively reasonable: the distance between training and test distributions scales linearly with the scale of the data manifold r. Moreover, given n randomly sampled points on an m-dimensional manifold, we may expect that roughly $n\rho^m$ of the volume of the manifold is within radius ρ of at

least one of the points; this is consistent with the $n^{-1/m}$ scaling. Indeed, Singh & Póczos (2018) finds that this scaling applies generally, regardless of manifold structure.

In Appendix C.4, we empirically compute Wasserstein distances on a sphere and find that the distances follow this approximation. Although this particular approximation is valid when the empirical training distribution q is drawn i.i.d. from a uniform underlying data distribution p, our framework can be applied to non-i.i.d. settings by simply modifying our Wasserstein distance approximation to reflect the train-test distributional shift.

Final Approximation Since f is a linear combination of basis functions with coefficients θ, L_f upper bounds the norm of the gradient of the basis functions. A basis function with a single output and frequency $\sqrt{k(k+m-1)}$ has a gradient with magnitude at most k/r. Since f's output has dimensionality d, we can take $L_f = K\sqrt{d}/r$. Substituting the result of Equation (6) along with our approximation for $W(p,q)$ into Equation (4) gives an approximated inductive bias complexity of

$$\tilde{I} \approx (2d^2E - nd) \left(\log b + \frac{3}{2}\log d + \log K - \frac{1}{m}\log n - \log\frac{\varepsilon}{L_{\mathcal{L}}} + \log c\right). \tag{9}$$

where $c = \sqrt{\frac{m}{6}}\left(\frac{2\pi^{(m+1)/2}d}{\Gamma\left(\frac{m+1}{2}\right)}\right)^{1/m}$, which is roughly constant for large intrinsic dimensionality m. Recall that d is the number of classes, n is the number of training data points, ε is the target error, $L_{\mathcal{L}}$ is the Lipschitz constant of the loss function, b is a bound on the model output, and K and E are functions of m and the maximum frequency $M = 2\pi r/\delta$.

Properties of Inductive Bias Complexity The dominant term in our expression, Equation (9), for inductive bias complexity is $2d^2E - nd$. Since $E \in O(M^m)$, inductive bias complexity is *exponential* in the data's intrinsic dimensionality m. Intuitively, this is because the dimensionality of the hypothesis space scales exponentially with intrinsic dimensionality: each additional dimension of variation in the input requires hypotheses to express a new set of output values for *each* possible value in the new input dimension. Because inductive bias complexity is the difficulty of specifying a well-generalizing hypothesis, it scales with the dimensionality of the hypothesis space. This exponential dependence empirically yields large scales for our inductive bias complexity measure, with the scale largely set by the input dimensionality. See Appendix G for further intuition and discussion.

Since $M = 2\pi r/\delta$, for a fixed m, inductive bias complexity is polynomial in $1/\delta$. Intuitively, this is because changing the maximum frequency allowed in hypotheses merely linearly scales the number of hypotheses along each dimension of the hypothesis space, yielding an overall polynomial effect. Thus, δ impacts inductive bias complexity less than m, but can still be significant for large m. Inductive bias complexity increases quadratically with d due to its quadratic effect on the dimensionality of the hypothesis space, so this has less impact than m and δ. Inductive bias complexity decreases roughly linearly with the number of training points n; intuitively, this is because each training point provides a roughly equal amount of information. Finally, inductive bias complexity increases logarithmically as the desired error ε decreases; this is because ε corresponds to a radius in the hypothesis space, and inductive bias is defined based on log of a volume in the hypothesis space. Intuitively, the inductive bias complexity *must* depend on the desired performance level: achieving the performance of a random classifier on a classification task is trivial, for example.

4. Empirical Generalization Difficulty of Benchmarks

Here, we use our quantification of inductive bias complexity as a measure of the *generalization difficulty* of a task (or *task difficulty*). Inductive bias complexity corresponds to the effort that a model designer must apply to generalize on a task, while sample complexity corresponds to the contribution of the training data; thus, inductive bias complexity is an appropriate measure of the generalization difficulty of a task from a model designer's perspective. We empirically validate our measure on several supervised classification, meta-learning, and RL tasks. We also quantify the relative inductive biases of different model architectures on different tasks and empirically show task difficulty trends over parametric task variations.

Computing the difficulty $\tilde{I}$ of Equation (9) for specific tasks specifying several parameters. The target generalization error ε is fixed at a level based on the type of the task (see Appendix D Table 3). We estimate the data dimensionality m by relating the decay rate of nearest neighbor distances in the training set to intrinsic dimensionality (Pope et al., 2021). The required detail (resolution) per dimension δ is set to the inter-class margin for classification tasks and a scale at which state perturbations do not significantly affect trajectories for RL tasks. See Appendix D for full details.

4.1. Task Difficulty of Standard Benchmarks

Image classification We estimate the task difficulty of the commonly used image classification benchmarks MNIST, SVHN (Netzer et al., 2011), CIFAR10 (Krizhevsky et al., 2009), and ImageNet (Deng et al., 2009). As shown in Figure 2, models solving these tasks encode large amounts

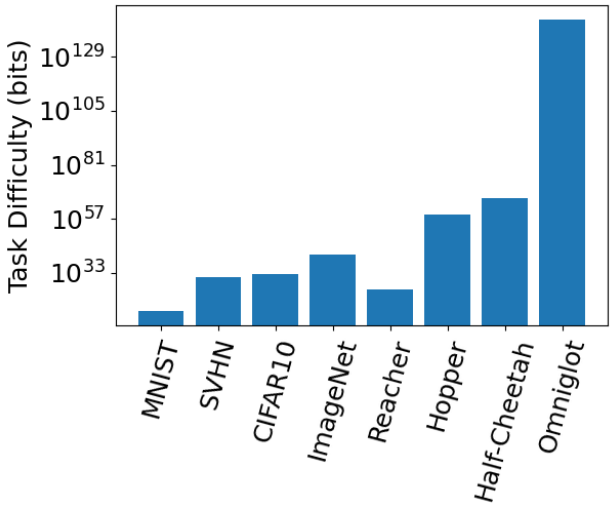


Figure 2: Task difficulties of various benchmark tasks across domains.

of inductive bias, with more complex tasks requiring models with more inductive bias.

Few-shot learning We estimate the task difficulty of 1-shot, 20-way classification on Omniglot (Lake et al., 2015). The model must learn a function f mapping a *dataset* of 20 images from an alphabet to a *classifier* g, which itself maps an image to a probability distribution over 20 classes. Both the inputs and outputs of f have much larger dimensionality than for typical image classification. This yields very large task difficulties: Omniglot has task difficulty $\approx 10^{145}$ bits (vs. 10^{41} for ImageNet); see Figure 2.

RL We find the task difficulty of the Reacher, Hopper, and Half-Cheetah control tasks in MuJoCo (Todorov et al., 2012; Duan et al., 2016). Figure 2 shows that more complex environments with higher dimensional observation and action spaces have higher task difficulties as intuitively expected. Moreover, the range of task difficulties is comparable with image classification: task difficulties are comparable across domains.

4.2. Inductive Bias Contributed by Different Models to Various Tasks

Although our task difficulty measure is model-agnostic, we can use it to compare how much inductive bias different models contribute to various tasks. To do this, we find the test set performance of a trained model from the model class (e.g. test set accuracy for classification tasks). We then use this performance level to compute ε in the task difficulty expression in Equation (9) (importantly, we do not *directly* use the test set data). This method is motivated by the idea that a model class enables generalization by providing an inductive bias, and its generalization ability can be quantified using test set performance. Thus, we quantify a model class's inductive bias as the minimum inductive bias required to achieve the performance of the model class: greater performance requires greater inductive bias. As observed in Table 2 (see Table 7 for full results), the increase in information content resulting from using a better model architecture (such as switching from ResNets (He et al., 2016) to Vision Transformers (Dosovitskiy et al., 2021)) is generally larger for more complex tasks. In other words, more complex tasks *extract* more of the inductive bias information present in a model.

Table 2: The inductive bias information contributed by different model architectures for image classification tasks.

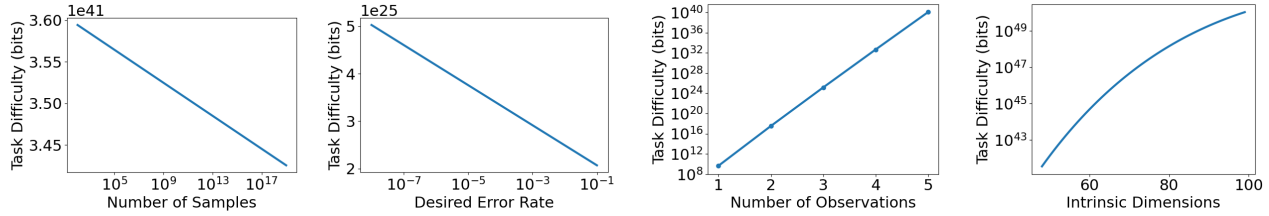
CIFAR10	
Model	Information Content ($\times 10^{32}$ bits)
Linear (Nishimoto, 2018)	2.250
AlexNet (Krizhevsky et al., 2012)	2.709
ResNet-50 (Wightman et al., 2021)	3.195
BiT-L (Kolesnikov et al., 2020)	3.453
ViT-H/14 (Dosovitskiy et al., 2021)	3.513
ImageNet	
Model	Information Content ($\times 10^{41}$ bits)
Linear (Karpathy, 2015)	2.063
AlexNet (Krizhevsky et al., 2012)	2.290
ResNet-50 (He et al., 2016)	2.362
BiT-L (Kolesnikov et al., 2020)	2.449
ViT-H/14 (Dosovitskiy et al., 2021)	2.461

4.3. Trends of Task Difficulty

In this section, we consider how task difficulty changes as we parametrically vary aspects of a task. Please see Appendix F for additional task variations.

Varying training data We vary the amount of training data provided in ImageNet and find in Figure 3a that the task difficulty decreases with more training points. Observe that even when training data is increased by many orders of magnitude, the scale of task difficulty remains large, suggesting that training data provides relatively little information to solve a task relative to inductive biases. Intuitively, this is because in the absence of strong inductive biases, training data only provides information about the approximated function in a *local* neighborhood around each training point; inductive biases, by contrast, can provide *global* constraints on the function. See Appendix G for further discussion.

Varying desired error rate In Figure 3b, we vary the desired error on the Reacher task and find that task difficulty decreases with desired error rate as expected. As with varying training data, the scale of task difficulty remains large even when dramatically reducing the desired error rate.



(a) Varying training data on ImageNet

(b) Varying desired error on Reacher

(c) Varying intrinsic dim on Cartpole

(d) Varying intrinsic dim on ImageNet

Figure 3: Task difficulties on variations of benchmark tasks.

Varying intrinsic dimension We consider a noisy version of Cartpole (Florian, 2007), an inverted pendulum control task where agents apply binary forces to control a pole. In noisy Cartpole, the agent observes a noisy version of the pole's angular position and velocity, requiring possibly multiple observations to find the optimal action. The agent must learn a function with a higher-dimensional input, making the task more difficult. As Figure 3c shows, task difficulty is exponential in the number of observations needed to find the optimal action. We also increase the intrinsic dimensionality of ImageNet classification by adding perturbations to the images outside the original image manifold; creating classifiers robust to such perturbations is known theoretically and empirically to be difficult (Madry et al., 2017; Yin et al., 2019). Figure 3d shows that task difficulty drastically increases with added intrinsic dimensions. Thus, increasing the intrinsic dimension of a task is a powerful way of increasing its difficulty.

5. Discussion

For all tasks, inductive bias complexities are much larger than practical model sizes. This is due to the volume of the hypothesis space: to be as model-free as possible, we constructed a vast space of all functions restricted only by a bandwidth constraint on frequency content Section 3.3. Well-generalizing functions occupy a small volume fraction within the space, requiring massive amounts of information in the inductive bias. See Appendix G for further discussion. A future direction is to obtain task (inductive bias) complexity measures for specific model classes (e.g. deep neural networks).

Our measure provides a quantitative guide to designing hard benchmarks. It reveals that tasks with inputs on higher dimensional manifolds require far more inductive bias relative to other variations. This yields counter-intuitive findings: meta-learning on simple Omniglot handwritten characters requires far more inductive bias than than classification of complex, realistic ImageNet images. Thus, prioritizing generalization over many degrees of variation is more important than increasing the richness of the task within each dimension of variation – even if these benchmarks, like Omniglot, may appear simple. Examples of such benchmarks include few-shot learning, adversarially robust image classification, and MDPs with very limited observations. As a specific, but broadly-applicable example of increasing the number of dimensions of variation, we suggest adding Gaussian noise to task inputs as in Figure 3d.

We also emphasize that inductive bias complexity is not intended to encompass *all* aspects of a task that may be difficult. For instance, in online RL, our definition of inductive bias complexity does not capture the policy dependence of the training data; thus, we cannot quantify the exploration-exploitation trade-off. However, our framework captures exactly the *generalization-relevant* aspects of a task: for example, in RL, generalization can be viewed in much the same way as supervised learning, except with different policies inducing different distributions during training and testing (Kakade, 2003). Importantly, in our analysis, we do not require that training and testing must be done on the same distribution (see Section 3.2); thus, we believe our measure applies to quantifying generalization in situations like online RL and beyond where the training and test distributions may be quite different.

To our knowledge, we provide the first measure of inductive bias complexity; thus, it is not designed to be quantitatively precise. We hope future work may be able to further refine our estimates. Nevertheless, we hope our measure may encourage the development of tasks requiring well-generalizing architectures and learning rules with many built-in inductive biases.

Acknowledgements

Ila Fiete is supported by the Office of Naval Research, the Howard Hughes Medical Institute (HHMI), and NIH (NIMH-MH129046).

References

Achille, A., Paolini, G., Mbeng, G., and Soatto, S. The information complexity of learning tasks, their structure and their distance. *Information and Inference: A Journal of the IMA*, 10(1):51–72, 2021.

Amari, S. A universal theorem on learning curves. *Neural Networks*, 6(2):161–166, 1993. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(93)90013-M.

An, S., Lee, M., Park, S., Yang, H., and So, J. An ensemble of simple convolutional neural network models for mnist digit recognition. *arXiv preprint arXiv:2008.10400*, 2020.

Anderssen, R. S., Brent, R. P., Daley, D. J., and Moran, P. A. P. Concerning $\int_0^1 \cdots \int_0^1 (x_1^2 + \cdots + x_k^2)^{1/2} dx_1 \cdots, dx_k$ and a Taylor series method. *SIAM Journal on Applied Mathematics*, 30(1):22–30, 1976. doi: 10.1137/0130003.

Bahri, Y., Dyer, E., Kaplan, J., Lee, J., and Sharma, U. Explaining neural scaling laws. *arXiv preprint arXiv:2102.06701*, 2021.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *JAIR*, 47:253–279, Jun 2013. doi: 10.1613/jair.3912.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. Learnability and the vapnik-chervonenkis dimension. *J. ACM*, 36(4):929–965, oct 1989. doi: 10.1145/76359.76371.

Chollet, F. On the measure of intelligence. *arXiv preprint*, 2019.

Ciregan, D., Meier, U., and Schmidhuber, J. Multi-column deep neural networks for image classification. In *CVPR*, 2012.

Cortes, C., Jackel, L. D., Solla, S., Vapnik, V., and Denker, J. Learning curves: Asymptotic values and rate of convergence. In *NeurIPS*, 1994.

Dai, F. and Xu, Y. *Approximation theory and harmonic analysis on spheres and balls*, volume 23. Springer, 2013.

Dekkers, A. L. M., Einmahl, J. H. J., and Haan, L. D. A moment estimator for the index of an extreme-value distribution. *The Annals of Statistics*, 1989.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

Du, S. S., Wang, Y., Zhai, X., Balakrishnan, S., Salakhutdinov, R., and Singh, A. How many samples are needed to estimate a convolutional or recurrent neural network? *NeurIPS*, 2018.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In *ICML*, 2016.

Florian, R. V. Correct equations for the dynamics of the cart-pole system. *Center for Cognitive and Neural Studies (Coneural), Romania*, 2007.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*, 2013.

Haussler, D. Quantifying inductive bias: Ai learning algorithms and valiant's learning framework. *Artificial Intelligence*, 36(2):177–221, 1988.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Hernández-Orallo, J. Evaluation in artificial intelligence: from task-oriented to ability-oriented measurement. *Artificial Intelligence Review*, Aug 2016.

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M. M. A., Yang, Y., and Zhou, Y. Deep learning scaling is predictable, empirically. *arXiv preprint*, 2017.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *CVPR*, 2017.

Hutter, M. Learning curve theory. *arXiv preprint arXiv:2102.04074*, 2021.

Jiang, Y., Natekar, P., Sharma, M., Aithal, S. K., Kashyap, D., Subramanyam, N., Lassance, C., Roy, D. M., Dziugaite, G. K., Gunasekar, S., et al. Methods and analysis of the first competition in predicting generalization of deep learning. In *NeurIPS 2020 Competition and Demonstration Track*, 2021.

Kakade, S. M. *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.

Karpathy, A. Breaking linear classifiers on imagenet. *Andrej Karpathy blog*, 2015.

Kawaguchi, K., Deng, Z., Luh, K., and Huang, J. Robustness implies generalization via data-dependent generalization bounds. In *ICML*, 2022.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *ICLR*, 2017.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. Big transfer (BiT): General visual representation learning. In *ECCV*, 2020.

Kolmogorov, A. N. Three approaches to the quantitative definition ofinformation'. *Problems of information transmission*, 1965.

Koltchinskii, V. and Panchenko, D. Rademacher processes and bounding the risk of function learning. *High Dimensional Probability II*, pp. 443–457, 2000. doi: 10.1007/978-1-4612-1358-1_29.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 and cifar-100 datasets. 2009. URL https://www.cs.toronto.edu/kriz/cifar.html.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

Lake, B., Salakhutdinov, R., and Tenenbaum, J. Human-level concept learning through probabilistic program induction. *Science*, 350:1332 – 1338, 2015.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Lee, C.-Y., Xie, S., Gallagher, P., Zhang, Z., and Tu, Z. Deeply-supervised nets. In *AISTATS*, 2015.

Lei, Y., Dogan, U., Binder, A., and Kloft, M. Multi-class svms: From tighter data-dependent generalization bounds to novel algorithms. *NeurIPS*, 2015.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. *NeurIPS*, 2018.

Li, Z., Zhang, Y., and Arora, S. Why are convolutional nets more sample-efficient than fully-connected nets? *ICLR*, 2021.

Lin, Z., Memisevic, R., and Konda, K. How far can we go without convolution: Improving fully-connected networks. *arXiv preprint*, 2015.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Mauch, L. and Yang, B. A novel layerwise pruning method for model reduction of fully connected deep neural networks. In *ICASSP*, 2017.

McRae, A., Romberg, J., and Davenport, M. Sample complexity and effective dimension for regression on manifolds. *NeurIPS*, 2020.

Mhaskar, H., Liao, Q., and Poggio, T. When and why are deep networks better than shallow ones? In *AAAI*, 2017.

mrgrhn. Alexnet with tensorflow, 2021. URL https://medium.com/swlh/alexnet-with-tensorflow-46f366559ce8.

Murata, N., Yoshizawa, S., and Amari, S.-i. Learning curves, model selection and complexity of neural networks. In *NeurIPS*, 1992.

Negrea, J., Haghifam, M., Dziugaite, G. K., Khisti, A., and Roy, D. M. Information-theoretic generalization bounds for sgld via data-dependent estimates. *NeurIPS*, 2019.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Reading digits in natural images with unsupervised feature learning. *NeurIPS*, 2011.

Nishimoto, B. E. Linear classifier, 2018. URL https://github.com/brunonishimoto/linear-classifier.

Poggio, T., Kawaguchi, K., Liao, Q., Miranda, B., Rosasco, L., Boix, X., Hidary, J., and Mhaskar, H. Theory of deep learning iii: explaining the non-overfitting puzzle. *arXiv preprint arXiv:1801.00173*, 2017.

Pope, P., Zhu, C., Abdelkader, A., Goldblum, M., and Goldstein, T. The intrinsic dimension of images and its impact on learning. *ICLR*, 2021.

Raginsky, M., Rakhlin, A., Tsao, M., Wu, Y., and Xu, A. Information-theoretic analysis of stability and bias of learning algorithms. In *IEEE Information Theory Workshop (ITW)*, 2016.

Sánchez, J. and Perronnin, F. High-dimensional signature compression for large-scale image classification. In *CVPR*, 2011.

Saxe, A. M., Bansal, Y., Dapello, J., Advani, M., Kolchinsky, A., Tracey, B. D., and Cox, D. D. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020, 2019.

Shamir, O., Sabato, S., and Tishby, N. Learning and generalization with the information bottleneck. In *ALT*, 2008.

Sharma, U. and Kaplan, J. Scaling laws from the data manifold dimension. *JMLR*, 23:9–1, 2022.

Shwartz-Ziv, R. and Tishby, N. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

Singh, S. and Póczos, B. Minimax distribution estimation in wasserstein distance. *arXiv preprint*, 2018.

Stein, E. and Shakarchi, R. *Fourier Analysis: An Introduction*. Princeton lectures in analysis. Princeton University Press, 2011. ISBN 9781400831234.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *IROS*, 2012.

Treves, F. *Topological Vector Spaces, Distributions and Kernels: Pure and Applied Mathematics, Vol. 25*, volume 25. Elsevier, 2016.

Veeramacheneni, L., Wolter, M., Klein, R., and Garcke, J. Canonical convolutional neural networks. *arXiv preprint arXiv:2206.01509*, 2022.

Wightman, R., Touvron, H., and Jégou, H. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021.

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

Yin, D., Kannan, R., and Bartlett, P. Rademacher complexity for adversarially robust generalization. In *ICML*, 2019.

Yu, T., Quillen, D., He, Z., Julian, R., Narayan, A., Shively, H., Bellathur, A., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *CoRL*, 2019.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zhang, P., Wang, H., Naik, N., Xiong, C., and Socher, R. DIME: An information-theoretic difficulty measure for AI datasets. *NeurIPS Workshop DL-IG*, 2020.

A. Mapping Between Our Framework and Common Learning Settings

In this section, we provide detailed descriptions of the mapping of various learning settings to our framework. We first introduce a unified set of notation we use to describe all these settings, and then individually discuss each setting.

We define a *task* as consisting of *instances* x and *actions* y; a learning system aims to learn a mapping from instances to actions. We assume that each instance x is described by a set of *features*: x has K_c continuous features $c_1, c_2, ...c_{K_c}$ and K_d discrete features $d_1, d_2, ...d_{K_d}$. Finally, we assume that the features uniquely describe all possible points x: there exists an invertible function A such that:

$$x = A(c_1, c_2, ...c_{K_c}, d_1, d_2, ...d_{K_d}) \tag{10}$$

for all possible values of x and the features on their respective manifolds. Note that such a decomposition of x always exists by using a single feature $c_1 = x$ or $d_1 = x$; tasks with a more complex structure may have instances with a further decomposition reflecting the task structure.

We assume that for each instance x, there is a unique desired action $y \in \mathcal{Y}$ among the set of possible actions $\mathcal{Y}$; the optimal mapping is denoted by $y = f^*(x)$. In general, learning the exact mapping f^* may be intractable. Instead, it may be feasible to closely match the function f^* within some error. Given a loss function $\mathcal{L}$, we define the error of a function f under a particular distribution p over x as:

$$\hat{e}(f, p) = \mathbb{E}_p[\mathcal{L}(f(x), x)] \tag{11}$$

We summarize the notational mapping between this framework and various learning scenarios in Table 6.

Supervised classification In a supervised classification setting, instances can be viewed as consisting of two features: a discrete feature d_1 corresponding to a class and a continuous feature c_1 specifying a particular example within the class:

$$x = A(c_1, d_1). \tag{12}$$

The optimal action function is to output the class $f^*(x) = d_1$. Observe that the continuous feature c_1 is irrelevant to the task; the goal of this task is to learn from training data how to extract the feature d_1 while maintaining invariance to c_1.

As a concrete example, consider the MNIST (Lecun et al., 1998) classification dataset; here labels correspond to d_1, and c_1 corresponds to all other information necessary to specify each image (e.g. line width, rotation, overall size). In general, c_1 may not be directly extractable from images; one method of estimating a representation of c_1 is to train a class-conditional autoencoder on MNIST images, and extract the latent representation of each image as c_1.

Meta-learning for few-shot supervised classification In a few-shot meta-learning setting, instances can be viewed as consisting of two features: a continuous feature c_1 corresponding to a set of related classes, and a continuous feature c_2 corresponding to a set of samples from those classes.

$$x = A(c_1, c_2) \tag{13}$$

The objective is to output a function g such that g maps inputs in set of classes to corresponding labels: $f^*(x) = g$. Note that g here depends only on c_1; the desired function f^* is invariant to c_2. Typically, a meta-learning system will receive instances from a subset of c_1 values, and will be asked to generalize to a different subset within the same distribution. To allow the meta-learning system to associate instances with classes on the unseen c_1 values, a few inputs with the unseen c_1 are provided for each class.

An example of such a meta-learning problem is few-shot classification on the Omniglot (Lake et al., 2015) dataset. Here, c_1 corresponds to an alphabet, and c_2 corresponds to a set of sample images within that alphabet.

Reinforcement learning In a reinforcement learning setting, instances can be viewed as consisting of a single feature specifying all the information in an instance. In the language of reinforcement learning, instances may typically correspond to the state of an environment for a fully-observed Markov decision process (MDP), or a sequence of observations for a partially-observed MDP. The single feature describing the instance may in general be discrete or continuous; for notational purposes, we default to continuous features when features may be either continuous or discrete.

$$x = A(c_1) \tag{14}$$

Here, $f^*(x)$ denotes the optimal action to take given state or observation sequence c_1. The goal of the learning system is to generalize over unseen states or observation sequences. However, note that the agent is not asked to generalize over parameters of the underlying MDP; the MDP may be fixed.

Note that our framework does not capture certain important aspects of RL settings; for instance, for online RL, it does not capture the fact that instances observed during may depend on the previous actions of the agent. This dependence is important to analyze the exploration-exploitation tradeoff of an RL task, for instance. Nevertheless, our framework is sufficient to assess the difficulty of generalizing in RL. The fact that the set of states or observation sequences over which is trained depends on the behavior of the agent has limited relevance to the generalization difficulty of the task.

As an example, consider the Cartpole (Florian, 2007) continuous control task implemented in the MuJoCo suite (Todorov et al., 2012). Here, c_1 corresponds to the observed state of the environment from which the agent is trained to extract the optimal action.

Unsupervised autoencoding In unsupervised autoencoding, the optimal action function $f^*(x)$ is simply the identity function $f^*(x) = x$. The goal is to generalize on a test set after observing a set of training instances. Usually, this is non-trivial as the learning system is constrained such that learning an identity function is infeasible. However, we highlight that our measure is model-agnostic; in this case the task difficulty can be viewed as a measure of complexity of the instance set. A specific example of such a problem may be unsupervised autoencoding of MNIST digits, each of which has a single continuous valued feature.

Meta-reinforcement learning In a meta-reinforcement learning setting, instances correspond to trajectories through *environments*, which consist of two features: a continuous feature c_1 parameterizing the environment and a continuous feature c_2 corresponding to the specific trajectory within the environment. Both features may in general be discrete or continuous; we use continuous features for our notation:

$$x = A(c_1, c_2) \tag{15}$$

The objective is to output a *policy* g such that g that maps states in an environment to actions: $f^*(x) = g$. Note that g depends only on c_1; the desired function f^* is invariant to c_2. The learning system is required to learn from the set of c_1 in the training set and generalize to an unseen set of c_1.

As an example, consider the Meta-World (Yu et al., 2019) meta-reinforcement learning benchmark in which an agent controlling a robotic arm must generalize from a set of training skills to an unseen set of test skills. The skills include simple manipulations such as pushing, turning, placing etc. In our formulation, c_1 corresponds to the skill and c_2 corresponds to specific trajectories of the arm performing the skill.

B. Proof of Theorem 2

Proof. Note that for any x_1, x_2, θ, we have

$$\begin{aligned}
\mathcal{L}(f(x_1;\theta), x_1) &= \mathcal{L}\Big(f(x_2;\theta) + (f(x_1;\theta^*) - f(x_2;\theta^*)) \\
&\qquad + (f(x_1;\theta) - f(x_1;\theta^*) - f(x_2;\theta) + f(x_2;\theta^*)), x_1\Big) && (16)\\
&\leq \mathcal{L}\Big(f(x_2;\theta) + (f(x_1;\theta^*) - f(x_2;\theta^*)), x_1\Big) \\
&\qquad + L_{\mathcal{L}}||f(x_1;\theta) - f(x_1;\theta^*) - f(x_2;\theta) + f(x_2;\theta^*)|| && (17)\\
&\leq \mathcal{L}(f(x_2;\theta), x_2) \\
&\quad + L_{\mathcal{L}}\|f(x_1;\theta) - f(x_1;\theta^*) - f(x_2;\theta) + f(x_2;\theta^*)\| && (18)\\
&\leq \mathcal{L}(f(x_2;\theta), x_2) + L_{\mathcal{L}} L_f d(x_1, x_2) d(\theta, \theta^*), && (19)
\end{aligned}$$

where (17) is a result of the Lipschitz condition on $\mathcal{L}$, (18) is a result of the shift-invariance of $\mathcal{L}$ and (19) is a result of our condition on f. This allows us to bound the difference

$$e(\theta, p) - e(\theta, q) = \mathbb{E}_p[\mathcal{L}(f(x;\theta), x)] - \mathbb{E}_q[\mathcal{L}(f(x;\theta), x)] \tag{20}$$

as follows: let γ be a distribution on (x_1, x_2) with marginal distributions p and q, respectively. Then

$$e(\theta, p) - e(\theta, q) = \mathbb{E}_\gamma[\mathcal{L}(f(x_1;\theta), x_1) - \mathcal{L}(f(x_2;\theta), x_2)] \tag{21}$$
$$\leq \mathbb{E}_\gamma[L_{\mathcal{L}} L_f d(x_1, x_2) d(\theta, \theta^*)]. \tag{22}$$

By the definition of Wasserstein distance,

$$e(\theta, p) - e(\theta, q) \leq L_{\mathcal{L}} L_f d(\theta, \theta^*) \inf_\gamma \mathbb{E}_\gamma[d(x_1, x_2)] = L_{\mathcal{L}} L_f d(\theta, \theta^*) W(p, q), \tag{23}$$

where the infimum is taken over all distributions γ with marginal distributions p, q. Therefore, for all $\theta \in \Theta_q$ with $d(\theta, \theta^*) \leq \frac{\varepsilon}{L_{\mathcal{L}} L_f W(p,q)}$, we have $e(\theta, p) \leq e(\theta, q) + \varepsilon = \varepsilon$ (since $e(\theta, q) = 0$). Recall that we assume a uniform prior on θ. Note that this can typically be made to hold with an appropriate choice of reparameterization: if our original density on θ is $P(\theta)$, then we may choose new parameters $\tilde{\theta} = g(\theta)$ satisfying $|det(\frac{\partial g(\theta)}{\partial \theta})| = P(\theta)$ to yield a uniform density over $\tilde{\theta}$.

Using the uniform prior assumption on θ, this gives

$$\tilde{I} = -\log \mathbb{P}(e(\theta, p) \leq \varepsilon \mid e(\theta, q) = 0) \tag{24}$$
$$\leq -\log \mathbb{P}\left(d(\theta, \theta^*) \leq \frac{\varepsilon}{L_{\mathcal{L}} L_f W(p, q)} \mid \theta \in \Theta_q\right), \tag{25}$$

as desired. □

C. Validity of Theoretical Assumptions

To arrive at our final expression for task difficulty in Equation (9), we make a number of assumptions. In this section, we further discuss the validity of each of our assumptions.

C.1. Shift-invariance of Loss Function

In many supervised classification and regression settings, the shift invariance assumption is satisfied by a squared error loss function. For clarity, in supervised regression, we denote inputs as x and outputs as y. A squared error loss function is constructed as $\mathcal{L}(y, x) = ||y - f^*(x)||_2^2$ which is shift invariant since

$$\mathcal{L}(y + f^*(x_1) - f^*(x_2), x_1) = ||y + f^*(x_1) - f^*(x_2) - f^*(x_1)||_2^2 = ||y - f^*(x_2)||_2^2 = \mathcal{L}(y, x_2). \tag{26}$$

Next, in a few-shot meta-learning setting, there are reasonably broad conditions under which shift-invariance holds: Instances correspond to datasets and actions correspond to functions mapping from inputs to the outputs of the inner loop task. For clarity, we define the inner task inputs and outputs as i and o; the goal is to learn a mapping from datasets x to functions y that map $o = y(i)$. Furthermore, suppose a loss function $L(o, i)$ is defined for the inner loop task. Then, suppose the meta-loss function is defined as $\mathcal{L}(y, x) = \mathbb{E}_{i \sim x}[L(y(i), i)]$. We assume that functions y are additive in the following sense: for any y_1, y_2, x, $(y_1 + y_2)(x) = y_1(x) + y_2(x)$. Finally, we assume a generalized version of shift-invariance for the inner loss function L:

$$\mathbb{E}_{i \sim x_2}[L(y(i), i))] = \mathbb{E}_{i \sim x_1}[L(y(i) + f^*(x_2)(i) - f^*(x_1)(i), i)] \tag{27}$$

This then implies shift-invariance for the meta-loss $\mathcal{L}$:

$$\mathcal{L}(y, x_2) = \mathcal{L}(y + f^*(x_2) - f^*(x_1), x_1) \tag{28}$$

The shift invariance assumption implies addition is defined over the action space. We believe that such additivity is reasonable for most learning problems. For instance, in classification settings, action spaces typically correspond to a vector of logits over all classes; these action spaces are Euclidean and actions can be added. Similarly, in reinforcement learning, both continuous and discrete action spaces are often additive, with the logits of a probability distribution over possible actions outputted in the discrete case. Moreover, shift-invariance is a property of the commonly used squared error loss, and indeed any loss function that has the form $\mathcal{L}(y, x) = G(y - f^*(x))$ for some function G.

At the same time, we note that many loss functions may not satisfy shift invariance. For example the error rate (computed as the fraction of incorrectly classified points in supervised classification), is neither differentiable nor shift-invariant. Thus, during training, differentiable proxy loss functions such as squared error or cross-entropy loss are often used, to similar effect. In other words, shift-invariant proxies can exist for non-shift-invariant losses. In reinforcement learning, loss functions may correspond to a Q function computing the cumulative negative reward over an episode conditioned on taking a particular action. In this case, loss functions often cannot be written in closed form, making it difficult to show shift invariance for these functions even if they are actually shift invariant.

For these reasons, we believe shift-invariance is a useful approximation that allows us to theoretically prove properties of task difficulty. We further note that the result of Theorem 1, which we used shift-invariance to prove, may hold even for loss functions without without shift invariance:

$$\mathcal{L}(f(x_1;\theta), x_1) - \mathcal{L}(f(x_2;\theta), x_2) \leq L_{\mathcal{L}} L_f d(x_1, x_2) d(\theta, \theta^*) \tag{29}$$

Intuitively, it states that when parameters are near their optimal values, small changes in inputs x yield only small changes in the loss, and this feels like a very simple and general requirement.

C.2. Second-order Condition on Model

The second order condition in Equation (3) ($\|f(x_1;\theta_1) - f(x_1;\theta_2) - f(x_2;\theta_1) + f(x_2;\theta_2)\| \leq L_f d(x_1, x_2) d(\theta_1, \theta_2)$) assumes that changes in the model f can be bounded when the inputs and parameters of f change a small amount. First, observe that if f is twice differentiable, then the condition always holds for some choice of L_f assuming x and θ are defined on a bounded domain. As a concrete example, consider the case of image classification with a Tanh activated neural network where each input pixel is bounded in range $[0, 1]$ and network parameters are bounded by a large radius (the bounded parameter assumption is reasonable since practically speaking, trained neural network parameters can be bounded by some radius depending on the amount of training).

As an example of a function that does not satisfy the condition, note that neural networks with the ReLU activation function may not satisfy Equation (3). However, it has been observed that ReLU networks behave like neural networks with smooth, polynomial activation functions (Poggio et al., 2017); thus, it may be reasonable to apply our assumption even to ReLU networks.

C.3. Shape of Interpolating Hypothesis Space

Regarding our approximation of the interpolating hypothesis space, we first note that the dimensionality of the interpolating hypothesis is *lower* than the full dimensionality of the hypothesis space. Taking an interpolating hypothesis and slightly perturbing it in an *arbitrary* direction will likely break interpolation. By contrast, we expect that there exists some subspace in which perturbations along the subspace maintain interpolation (i.e. there is a continuous manifold of interpolating hypotheses). This is analogous to how in neural networks, local minima can often be in *flat* regions in which perturbing parameters along those directions does not significantly affect the loss (Keskar et al., 2017).

We approximate the interpolating hypothesis space as a disk in the overall hypothesis space (for clarity, we have revised our terminology from "ball" to "disk"); this disk has lower dimensionality than the overall hypothesis space but is embedded in a higher dimensional space. The disk assumption preserves key properties that we may reasonably expect of interpolating hypothesis spaces, namely 1) low-dimensionality and 2) boundedness, while being analytically tractable.

Finally, we emphasize that the disk approximation is made to approximate the volume of the interpolating hypothesis space. The detailed shape of the interpolating hypothesis space is not itself central to our result, but rather the scaling of its volume is more important. We believe the disk assumption captures the key scalings: the dependence of the volume on b and d.

C.4. Empirically Validating Wasserstein Distance Approximation

In this section, we empirically validate our approximation of Wasserstein distance made in Section D. Recall that We approximate the Wasserstein distance between a uniform distribution p on a sphere of radius 4 and an empirical distribution q of n randomly sampled points on the sphere as:

$$W(p, q) \in O(rn^{-1/m}) \tag{30}$$

Note that this expression grows linearly with r and decays sub-linearly with n, with a slower decay with larger m for which more points may be required for all regions to be within a fixed radius of any point. Theoretically, we may expect this approximation to hold for large n to achieve near uniform q. If this scaling holds, then we would expect $\log W(p, q)$ to decrease linearly with $\log n$ (with slope $-1/m$). Similarly, we would expect $\log W(p, q)$ to decrease linearly with $1/m$ (with slope $-\log n$).

We conduct experiments on a unit sphere ($r = 1$). We approximate a uniform distribution by sampling 10000 points on a unit sphere and calculate the exact Wasserstein distance between the empirical distribution of 10000 points and the empirical distribution of $n < 10000$ points (we vary n from 10 to 500). Distances are computed for varying choices of m varied from 3 to 19.

As illustrated in Figure 4, we find that the relationship between $\log W(p, q)$ and $\log n$ is linear with a steeper slope for small m, as expected. We also observe that the relationship between $\log W(p, q)$ and $\log m$ is nonlinear, with distances for large m being larger than expected by a linear trend-line. Nevertheless, a linear trend-line still produces a strong fit to the empirical data.

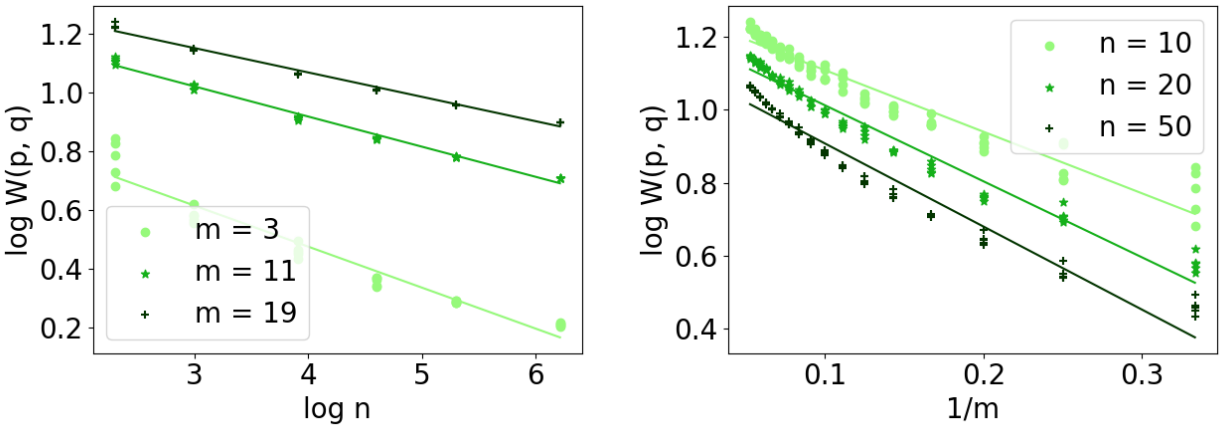


Figure 4: Wasserstein distance estimates between a uniform and empirically sampled uniform distribution on a unit sphere as a function of the dimensionality of the sphere m and the number of sampled points n. Results are shown for five trials in each setting, and linear trend-lines are fitted to the results.

D. Empirically Approximating Task Difficulty

In Section 3.3, we presented our inductive bias complexity approximation for supervised classification. Here, we provide more details on how to apply and compute our approximation to other settings.

First, we write a slightly modified version of our task difficulty approximation

$$\tilde{I} \approx (2dzE - nd)\left(\log b + \frac{1}{2}\log z + \log d + \log K - \frac{1}{m}\log n - \log\frac{\varepsilon}{L_{\mathcal{L}}} + \log c\right). \tag{31}$$

where $c = \sqrt{\frac{m}{6}}\left(\frac{2\pi^{(m+1)/2}z}{\Gamma\left(\frac{m+1}{2}\right)}\right)^{1/m}$, and K and E are functions of m and the maximum frequency $M = 2\pi r/\delta$. In supervised classification settings, $z = d$.

In general settings, the parameters controlling task difficulty $(m, n, \varepsilon, L_{\mathcal{L}}, r, b, d, z)$ can be interpreted as follows: m is the intrinsic dimensionality of the manifold on which instances lie. n is the number of training points. ε is the desired error rate on the task. $L_{\mathcal{L}}$ is the Lipschitz constant of the loss function; for tasks without an explicit, differentiable loss function (such as in reinforcement learning), it can be interpreted as the overall sensitivity of the loss with respect to the model output. r and b is are bound on the size of the model input and output respectively. d corresponds to the dimensionality of the model output. z corresponds to the number of submanifolds on which instances lie (with each combination of discrete features of

instances corresponding to a submanifold). In supervised classification settings, the number of submanifolds is simply the number of classes, which is why z is set equal to d in these settings.

In order to compute the difficulty $\tilde{I}$ of a task, we must determine the values of these parameters. For all tasks, we set $\varepsilon/L_{\mathcal{L}}$ as the desired performance level, which is the distance in the output space corresponding to an error of ε. For each task, we set a fixed desired performance level corresponding to the category of the task (i.e. a fixed error rate of 1.0% for image classification tasks and a fixed error rate of 0.001 for RL tasks); see Table 3 However, because typical well-performing models in the literature trained on MNIST/ImageNet achieve error significantly better/worse performance than 1.0%, we adjust the desired error rate to 0.1% and 10.0% for MNIST and ImageNet respectively. This is done to properly quantify the difficulty of these tasks in the error rate regimes in which these tasks are typically used. However, our task difficulty numbers are relatively insensitive to the precise choice of desired performance level: using a fixed error rate of 1.0% for all tasks, we find that the scale of task difficulties do not significantly change. Specifically, task difficulty falls incrementally for MNIST from 1.09×10^{16} bits to 9.03×10^{15} bits, and rises incrementally for ImageNet from 2.48×10^{43} bits to 2.81×10^{43} bits.

Table 3: Desired error rates for different tasks used to compute a measure of inductive bias complexity. For each task, desired error rate is set as the performance of typical well-performing models proposed in the literature trained on the task. For classification tasks, error rate corresponds to test set accuracy. For Cartpole, error rate corresponds to the probability of making an incorrect action at any given time step. For MuJoCo continuous control tasks with continuous action spaces, error rate corresponds to desired distance in from the optimal action.

Quantity	MNIST	SVHN	CIFAR-10	ImageNet	Omniglot	Cartpole	Reacher	Hopper	Half-cheetah
Error rate	0.1 %	1.0 %	1.0 %	10.0 %	1.0 %	0.001	0.001	0.001	0.001

While intrinsic dimensionality m is known for some tasks (e.g. MuJoCo (Du et al., 2018) tasks), it must be estimated for others; we use the nearest neighbors estimation approach described in Pope et al. (2021). The spatial resolution δ is set based on the nature of each task. For classification tasks, it is natural to set the spatial resolution as the margin between classes. For RL tasks, it is set as a scale on which trajectories with perturbations of size δ are not likely to be meaningfully different. For all tasks, we approximate r to be the maximum norm of the training data points.

D.1. Empirically Computing Classification Task Difficulty

For an image classification task, the model output is a probability distribution on the classes so b to be 1. We set δ to be the minimum distance between points of different classes, as this is the minimum distance between inputs that f must be able to distinguish. For large datasets where computing δ exactly is impractical, we estimate it using statistical methods. This procedure is explained in Appendix E.

We estimate m using the technique described by Pope et al. (2021), which gives an MLE estimate based on distances to nearest neighbors. We use $k = 5$ nearest neighbors for our estimation. For large datasets where computing nearest neighbors for all points is impractical, we use the anchor approximation from Pope et al. (2021), selecting a random sample of points to use as anchors and finding their nearest neighbors in the entire dataset. To estimate m for ImageNet, we randomly select 2000 points to be our anchors. See Table 4 for our estimates of m and δ for image classification benchmarks.

Table 4: Estimated m and δ for various image classification benchmarks. These estimates are used to estimate the amount of inductive bias required to solve each task.

Quantity	MNIST	SVHN	CIFAR-10	ImageNet
m	14	19	27	48
δ	2.4	1.6	2.8	65

D.2. Empirically Computing Meta-Learning Task Difficulty

The Omniglot one-shot classification task is to identify the class of an image given an example image of each of 20 letters. Thus, the meta-learning task is to learn a function f that maps a set of 20 images to a function g that can map a single image to a probability distribution over 20 classes. In the following discussion, for quantities such as z, d, and E, a subscript f will

be used to denote that we are considering these values for the function f, and a subscript g will be used to denote that we are considering these values for the function g.

In our framework, the input to f has no discrete features, so $z_f = 1$. Since g is the output of f, d_f equals the dimensionality of the parameterization of g, which is $2z_g d_g E_g$. The function g classifies an image among 20 classes, so $z_g = 20$ and $d_g = 19$. We can compute E_g as a function of $M_g = 2\pi r_g/\delta_g$ and m_g. As with image classification tasks, we set r_g to be the maximum norm of an image in the dataset and δ_g to be the minimum distance between images of different classes. We set m_g to be the dimensionality m_0 of an image drawn from a single alphabet. This is done by estimating the dimensionality of each alphabet in the dataset and averaging the results.

Since the input to f is 20 images, $r_f = r_g\sqrt{20}$. To compute m_f, let $m_1 > m_0$ be the dimensionality of an image drawn from the entire dataset. Then the dimensionality of the alphabet underlying the input to f is $m_1 - m_0$, and the dimensionality of the images themselves is $20m_0$. Therefore, $m_f = m_1 + 19m_0$. We want f to be sensitive to a change in only one of the 20 images, so we set $\delta_f = \delta_g$.

The training dataset consists of several alphabets. We write ℓ_a for the number of letters in alphabet a. We set n to be the number of sets of 20 images that can be drawn from the training dataset that represent 20 different letters in the same alphabet. Since the training dataset contains 20 images representing each letter,

$$n = \sum_a \binom{\max\{\ell_a, 20\}}{20} \cdot 20^{20}, \tag{32}$$

where the sum is taken over all training alphabets a.

Finally, b_f is the maximum norm of f's output, which contains the parameterization of g. We found that this can be bounded by $b_g\sqrt{z_g d_g}$. We have already computed z_g and d_g, and since g outputs a probability distribution, $b_g = 1$. We then use the values z_f, d_f, m_f, δ_f, n, b_f, and the state-of-the-art error rate $\varepsilon/L_{\mathcal{L}}$ to compute the task difficulty.

D.3. Empirically Computing Reinforcement Learning Task Difficulty

For evaluating task difficulty for reinforcement learning tasks, we use some different approximations than for the supervised classification that are more suited to our particular setting; in particular, we use a different assumption for the manifold on which instances lie and a different approximation for Wasserstein distance $W(p, q)$.

We first write our general version of the task difficulty expression without applying the Wasserstein distance approximation in Section 3.3:

$$\tilde{I} \approx (2dzE - nd)\left(\log b + \frac{1}{2}\log z + \log d + \log K - \log r + \log W(p,q) - \log\frac{\varepsilon}{L_{\mathcal{L}}}\right). \tag{33}$$

In the reinforcement learning tasks we evaluate, $z = 1$, so we can write:

$$\tilde{I} \approx (2dE - nd)\left(\log b + \log d + \log K - \log r + \log W(p,q) - \log\frac{\varepsilon}{L_{\mathcal{L}}}\right). \tag{34}$$

The input to f is an m-dimensional observation, which can be considered as a point in the hypercube $[-\pi, \pi]^m$ after scaling. Parameterizing f as a sum of eigenfunctions of the Laplace-Beltrami operator,

$$f(x;\theta) = \sum_{p\in\mathbb{Z}^m} \theta_p e^{ip^\top x} \tag{35}$$

(Treves, 2016). The wavelength of the component corresponding to p is $2\pi/\|p\|$, so we restrict our parameterization to components satisfying $\|p\| \le 2\pi/\delta$. Thus, we have $L_f = 2\pi/\delta$. The number of eigenfunctions E is the number of vectors $p \in \mathbb{Z}^m$ satisfying $\|p\| \le 2\pi/\delta$, which we approximate as the volume of an m-dimensional ball with radius $2\pi/\delta$, giving

$$E \approx \left(\frac{2\pi}{\delta}\right)^m V_m, \tag{36}$$

where $V_m = \frac{\pi^{m/2}}{\Gamma\left(\frac{m}{2}+1\right)}$ is the volume of an m-dimensional ball with radius 1. We now find the Wasserstein distance $W(p, q)$. In the optimal transport, each training data point is associated with an equally-sized region of the hypercube. We approximate

these regions as hypercubes with side length $s = 2\pi n^{-1/m}$. The expected distance between two randomly chosen points in one of these hypercubes is at most $s\sqrt{m/6}$ (Anderssen et al., 1976), so

$$W(p, q) \approx \sqrt{\frac{m}{6}} \cdot 2\pi n^{-1/m}. \tag{37}$$

Finally, we note that f's output is a force in $\{-1, +1\}$ in the discrete case and an element of $[-1, 1]^d$ in the continuous case. Therefore, we can set $b = \sqrt{d}$ in all cases. Using these values, we find

$$\tilde{I} \approx d\left(2\left(\frac{2\pi}{\delta}\right)^m V_m - n\right)\left(\log\frac{4\pi^2}{\sqrt{6}} + \log d + \frac{1}{2}\log m - \log\delta - \frac{1}{m}\log n - \log\frac{\varepsilon}{L_{\mathcal{L}}}\right). \tag{38}$$

Recall that d is the dimensionality of the output space, and m is the dimensionality of the observation space. In the noisy Cartpole task, $T > 1$ observations may be needed to determine the optimal action. Each observation is a two-dimensional state (containing the pole's angular position and velocity), so we set $m = 2T$. For fully observed tasks, note that $T = 1$.

It remains to choose values for the parameters n, δ, and $\varepsilon/L_{\mathcal{L}}$. We choose the relatively small value of 0.001 for δ and $\varepsilon/L_{\mathcal{L}}$ for all tasks. We choose a small value for δ because small changes in initial conditions can lead to large changes over time, and we choose a small value for $\varepsilon/L_{\mathcal{L}}$ because agents should be able to perform near-optimal actions to achieve the task's goal. In the noisy Cartpole task, we set $n = 10000$, corresponding to observing 100 episodes with 100 timesteps each. In the MuJoCo environments, we set $n = 1000000$, corresponding to observing 1000 episodes with 1000 timesteps each. We emphasize that the most important factor contributing to task difficulty is m, which is determined by the task specification.

E. Estimating δ for Classification Tasks

If we cannot compute δ exactly, we estimate it using extreme value theory (Dekkers et al., 1989).

We wish to find the maximum value of a distribution, in this case the distribution of the reciprocal of the distance between a random pair of points from different classes. The extreme value distribution is characterized by the extreme value index γ. Let $X_1, \ldots, X_n$ be values drawn from the distribution, and let $X_{1,n} \leq \cdots \leq X_{n,n}$ be these values in sorted order. For a fixed $k < n$, define

$$M_n^{(r)} = \frac{1}{k}\sum_{i=0}^{k-1}(\log X_{n-i,n} - \log X_{n-k,n})^r. \tag{39}$$

Then we can estimate γ as

$$\hat{\gamma}_n = M_n^{(1)} + 1 - \frac{1}{2}\left(1 - \frac{(M_n^{(1)})^2}{M_n^{(2)}}\right)^{-1}. \tag{40}$$

Empirically, we typically find $\hat{\gamma}_n > 0$. In this case, we estimate δ by estimating the quantile corresponding to the top $1/P$ of the distribution, where P is the number of pairs of training data points from different classes. By assuming that training data points are evenly distributed among the classes, we approximate P as $n^2(1 - 1/C)$, where C is the number of classes. If $a_n = kP/n$, this gives an estimate of

$$\frac{a_n^{\hat{\gamma}_n} - 1}{\hat{\gamma}_n} \cdot X_{n-k,n} M_n^{(1)} + X_{n-k,n} \tag{41}$$

for $1/\delta$.

The theoretical results (Dekkers et al., 1989) require that k and n/k go to infinity as $n \to \infty$. For ImageNet, we choose $n = 40000$ and $k = 200$. To reduce the noise in our estimation, we compute 10 estimates for $1/\delta$ and average the results to obtain our final estimate for $1/\delta$.

F. Additional Task Variations

In this section, in order to provide more intuition for our empirical results on inductive bias complexity, we compute the inductive bias complexity of additional variations of tasks.

F.1. Task Combinations

Setup We first consider inductive bias complexities of task combinations. We assume we are given two tasks, corresponding to the mapping between instances x_1 to $y_1 = \bar{f}_1^*(x_1)$ and x_2 to $y_2 = \bar{f}_2^*(x_2)$ respectively. Furthermore, we assume that training and test distributions q_1, p_1 and p_2, q_2 are provided. We then construct a combination of the two tasks as the mapping from (x_1, x_2) to $f^*(x_1, x_2) = (\bar{f}_1^*(x_1), \bar{f}_2^*(x_2))$. The test distribution of instances for the new task is constructed as:

$$p(x_1, x_2) = p_1(x_1)p_2(x_2) \tag{42}$$

And the training distribution constructed as:

$$q(x_1, x_2) = q_1(x_1)q_2(x_2) \tag{43}$$

Note that if q_1 and q_2 each correspond to training sets of size n_1 and n_2, q corresponds to a training set of size $n_1 n_2$. We assume that the two tasks have loss functions $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively. For the purposes of our analysis in this section, we assume that the loss functions satisfy $\mathcal{L}_i(y, x) \geq 0$, and equality holds if and only if $y = f_i^*(x)$ for $i = 1, 2$. We construct the loss function $\mathcal{L}$ for the combined task as:

$$\mathcal{L}((y_1, y_2), (x_1, x_2)) = \alpha\mathcal{L}_1(y_1, x_1) + (1 - \alpha)\mathcal{L}_2(y_2, x_2) \tag{44}$$

for a parameter α in $(0, 1)$. We denote a hypothesis as θ parameterizing a function that inputs (x_1, x_2) and outputs the predictions for each task: $f(x_1, x_2; \theta) = (f_1(x_1, x_2; \theta), f_2(x_1, x_2; \theta))$. Then, the generalization error of a particular hypothesis θ under distribution p (and analogous for q) is:

$$\begin{aligned} e(\theta, p) = \hat{e}(f(\cdot; \theta), p) &= \mathbb{E}_p[\mathcal{L}(f(x_1, x_2; \theta), (x_1, x_2))] \\ &= \alpha\mathbb{E}_p[\mathcal{L}_1(f_1(x_1, x_2), x_1)] + (1 - \alpha)\mathbb{E}_p[\mathcal{L}_2(f_2(x_1, x_2), x_2)] \end{aligned} \tag{45}$$

For notational convenience, we will also define $e_i(\theta, p)$ as (and analogous for q):

$$e_i(\theta, p) = \mathbb{E}_p[\mathcal{L}_i(f_i(x_1, x_2; \theta), x_i)] \tag{46}$$

for $i = 1, 2$. Thus, we may express $e(\theta, p)$ as:

$$e(\theta, p) = \alpha e_1(\theta, p) + (1 - \alpha)e_2(\theta, p) \tag{47}$$

Note that $e_1(\theta, p)$ and $e_2(\theta, p)$ *do not* correspond to errors of hypothesis of θ on tasks 1 and 2 directly because θ parameterizes a function which takes both x_1 and x_2 as input. Instead, $e_1(\theta, p)$ (and analogously for $e_2(\theta, p)$) corresponds to the error of a variant of task 1 where instances x_1 are augmented with a *distractor* input x_2 which is irrelevant to the task (producing instances (x_1, x_2)), but the task output y_1 is constructed the same way as $y_1 = \bar{f}_1^*(x_1)$. For clarify, we define this mapping as $y_1 = f_1^*(x_1, x_2)$ (and analogously for the variant of task 2). Importantly, note that this variant of task 1 has an input with a larger intrinsic dimensionality.

Now, we consider the inductive bias complexity of the combined task corresponding to f^*. Recall that we define inductive bias complexity as:

$$\tilde{I} = -\log \mathbb{P}(e(\theta, p) \leq \varepsilon \mid e(\theta, q) = 0) \tag{48}$$

We relate this inductive bias complexity to the inductive bias complexities of tasks corresponding to f_1^* and f_2^*:

$$\tilde{I}_i = -\log \mathbb{P}(e_i(\theta, p) \leq \varepsilon \mid e_i(\theta, q) = 0) \tag{49}$$

where $\tilde{I}_i$ corresponds to the inductive bias complexity for the task corresponding to f_i^*. Under certain conditional independence assumptions, we are able to relate $\tilde{I}$ and the $\tilde{I}_i$. Specifically, we assume that if θ interpolates task 1, then additionally interpolating task 2 does not change the probability that θ will generalize on task 1 (and vice versa). We also assume that if θ interpolates both task 1 and task 2, then generalizing on task 1 does not affect the probability that θ generalizes on task 2 (and vice versa). This is reasonable if we consider task 1 and task 2 to be constructed independently in the sense that inputs x_2 do not provide information on $f_1^*(x_1, x_2)$ and vice versa; being able to interpolate or generalize on one task does not affect the ease of generalization on the other. Thus, we are able to make the following statement:

Theorem 3. *Assume the following conditional independencies:*

$$e_1(\theta, p) \leq \varepsilon \perp\!\!\!\perp e_2(\theta, q) = 0 | e_1(\theta, q) = 0 \tag{50}$$

$$e_2(\theta, p) \leq \varepsilon \perp\!\!\!\perp e_1(\theta, q) = 0 | e_2(\theta, q) = 0 \tag{51}$$

$$e_1(\theta, p) \leq \varepsilon \perp\!\!\!\perp e_2(\theta, p) \leq \varepsilon | e_1(\theta, q) = 0, e_2(\theta, q) = 0 \tag{52}$$

Then,

$$-\log(e^{-\tilde{I}_1} + e^{-\tilde{I}_2}) \leq \tilde{I} \leq \tilde{I}_1 + \tilde{I}_2 \tag{53}$$

Proof. First, that by conditional independence:

$$\mathbb{P}(e_1(\theta, p) \leq \varepsilon \mid e(\theta, q) = 0)\mathbb{P}(e_2(\theta, p) \leq \varepsilon \mid e(\theta, q) = 0) \\ = \mathbb{P}(e_1(\theta, p) \leq \varepsilon \wedge e_2(\theta, p) \leq \varepsilon \mid e(\theta, q) = 0) \tag{54}$$

Note that $e(\theta, p) \leq \varepsilon \impliedby e_1(\theta, p) \leq \varepsilon \wedge e_2(\theta, p) \leq \varepsilon$. This implies:

$$\mathbb{P}(e(\theta, p) \leq \varepsilon \mid e(\theta, q) = 0) \geq \mathbb{P}(e_1(\theta, p) \leq \varepsilon \wedge e_2(\theta, p) \leq \varepsilon \mid e(\theta, q) = 0) \\ = \mathbb{P}(e_1(\theta, p) \leq \varepsilon \mid e(\theta, q) = 0)\mathbb{P}(e_2(\theta, p) \leq \varepsilon \mid e(\theta, q) = 0) \tag{55}$$

Also, observe that $e(\theta, p) \leq \varepsilon \implies e_1(\theta, p) \leq \varepsilon \vee e_2(\theta, p) \leq \varepsilon$. This implies:

$$\mathbb{P}(e_1(\theta, p) \leq \varepsilon \mid e(\theta, q) = 0) + \mathbb{P}(e_2(\theta, p) \leq \varepsilon \mid e(\theta, q) = 0) \geq \mathbb{P}(e(\theta, p) \leq \varepsilon \mid e(\theta, q) = 0) \tag{56}$$

Next, using our first two conditional independence assumptions:

$$\mathbb{P}(e_1(\theta, p) \leq \varepsilon \mid e_1(\theta, q) = 0)\mathbb{P}(e_2(\theta, p) \leq \varepsilon \mid e_2(\theta, q) = 0) \leq \mathbb{P}(e(\theta, p) \leq \varepsilon \mid e(\theta, q) = 0) \tag{57}$$

$$\mathbb{P}(e_1(\theta, p) \leq \varepsilon \mid e_1(\theta, q) = 0) + \mathbb{P}(e_2(\theta, p) \leq \varepsilon \mid e_2(\theta, q) = 0) \geq \mathbb{P}(e(\theta, p) \leq \varepsilon \mid e(\theta, q) = 0) \tag{58}$$

Finally, taking the negative log of both sides:

$$-\log(e^{-\tilde{I}_1} + e^{-\tilde{I}_2}) \leq \tilde{I} \leq \tilde{I}_1 + \tilde{I}_2 \tag{59}$$

as desired. □

Note that $-\log(e^{-\tilde{I}_1} + e^{-\tilde{I}_2}) \approx \min\{\tilde{I}_1, \tilde{I}_2\}$. Thus, the statement intuitively says that the combined task requires inductive bias up to the total inductive bias of the two tasks treated individually, and requires *at least* the inductive bias of the easier task. We emphasize again that $\tilde{I}_i$ does *not* correspond to the inductive bias required to solve task i, but rather the inductive bias required to solve a variant of task i with a distractor added to instances provided from the other task.

Experiments Next, we empirically compute inductive bias complexities for combinations of image classification tasks. To do this, we compute the task difficulty of a version of each task with a task-irrelevant distractor from the *other* task appended to each instance ($\tilde{I}_1$ or $\tilde{I}_2$ as described above). We then report the upper bound $\tilde{I}_1 + \tilde{I}_2$ and lower bound $-\log(e^{-\tilde{I}_1} + e^{-\tilde{I}_2})$ of $\tilde{I}$.

In order to compute task difficulties for tasks with distractor-appended instances, we first revisit our generalized approximation of inductive bias complexity from Equation (31):

$$\tilde{I} \approx (2dzE - nd)\left(\log b + \frac{1}{2}\log z + \log d + \log K - \frac{1}{m}\log n - \log\frac{\varepsilon}{L_{\mathcal{L}}} + \log c\right). \tag{60}$$

where $c = \sqrt{\frac{m}{6}}\left(\frac{2\pi^{(m+1)/2}}{\Gamma\left(\frac{m+1}{2}\right)}\right)^{1/m}$, which is roughly constant for large intrinsic dimensionality m. Recall that d is the output dimensionality, z is the number of manifolds that the input data lie on, n is the number of training data points, ε is the target error, $L_{\mathcal{L}}$ is the Lipschitz constant of the loss function, and K and E are functions of m and the maximum frequency $M = 2\pi r/\delta$. Typically for classification problems, $z = d$ is set as the number of classes.

When we append task-irrelevant distractors to the instances of a task, five key parameters change: the number of training points n, the intrinsic dimensionality m, the number of manifolds z and the bound on model output b and bound on model input r. Importantly, all other parameters stay fixed (namely, δ, d, ε and $L_{\mathcal{L}}$). The number of training points scales with the number of different distractor instances; if task 1 with n_1 training points is augmented with instances from task 2 with n_2 training points, the new task has $n_1 n_2$ training points. Also, the new intrinsic dimensionality of the task is the *sum* of the intrinsic dimensionalities of the individual tasks since the combined manifold of (x_1, x_2) is the *product* of the manifolds of x_1 and x_2 individually. Thus, if task 1 has intrinsic dimensionality m_1 and task 2 has intrinsic dimensionality m_2, the intrinsic dimensionality of the augmented task is $m_1 + m_2$. The number of manifolds of the input of the combined task is the number of manifolds corresponding to (x_1, x_2), which is simply $z_1 z_2$ if task 1 and task 2 have z_1 and z_2 classes respectively. If task 1 and task 2 have model outputs bounded by b_1 and b_2 respectively, then the combined model is bounded by $\sqrt{b_1^2 + b_2^2}$. Similarly, the new value of r is $\sqrt{r_1^2 + r_2^2}$. We may then compute the task difficulty of distractor-appended task using the formula above.

We compute task difficulties for pairwise combinations of MNIST, SVHN and CIFAR-10 (which have individual task difficulties (in bits) of: 1×10^{16}, 1×10^{31}, 3×10^{32}). As found in Table 5, combined task difficulties are significantly greater than the difficulties of individual tasks. As a very rough rule of thumb, the combined task difficulty is approximately the product of the individual task difficulties. This makes sense since the task difficulty scales exponentially with the intrinsic dimension of the data and combining together two tasks *adds* together the intrinsic dimensionality of the two tasks (and thus multiplies their task difficulties).

Table 5: Combined task difficulty bounds (in bits) for combinations of image classification tasks. Difficulties are reported as "lower bound / upper bound." Individual task difficulties of MNIST, SVHN and CIFAR-10 are (in bits): 1×10^{16}, 1×10^{31}, 3×10^{32} respectively.

	MNIST	SVHN	CIFAR-10
MNIST	1×10^{26} / 3×10^{26}	6×10^{39} / 5×10^{45}	3×10^{42} / 8×10^{44}
SVHN	-	4×10^{54} / 8×10^{54}	4×10^{50} / 4×10^{61}
CIFAR-10	-	-	2×10^{55} / 3×10^{55}

Why does combining together two simple tasks like MNIST result in a much more difficult task? This is because separating the two parts of the combined task requires a lot of inductive bias. Without this inductive bias, solving a combination of two tasks looks like solving a single task with a higher dimensional input, which correspondingly requires much greater inductive bias. This also suggests how practical model classes like neural networks may be able to provide such vast amounts of inductive bias to a task: namely, by breaking down the input into lower-dimensional components.

F.2. Predicting Zero-output

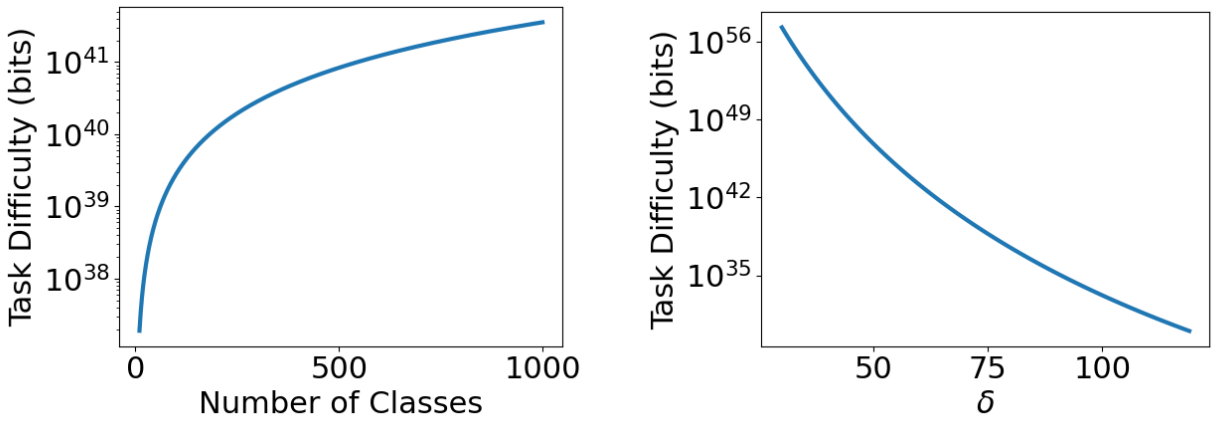
To gain intuition for the properties of task difficulty, we compute task difficulties for a task in which the target function always has output 0: $f^*(x)$. To compute the task difficulty, we return to the definition of inductive bias complexity from Definition 3.1:

$$\tilde{I} = -\log \mathbb{P}(e(\theta, p) \leq \varepsilon \mid e(\theta, q) \leq \epsilon) \tag{61}$$

Keeping $\epsilon = 0$ as we have done throughout the paper, observe that just as with the general case, the inductive bias complexity is based on the fraction of interpolating hypotheses that generalize well. In the case of zero output, generalizing well means that $f(\cdot; \theta)$ should be sufficiently close to 0 on the distribution p. Note that interpolating hypotheses may not necessarily have near-zero output over distribution p; thus, solving the a zero-output task may require significant levels of inductive bias.

The key to determining the difficulty of a zero-output task is setting the hypothesis space. Recall that in Section 3.3, we constructed the hypothesis space to be a linear combinations of a set of basis functions sensitive to a *task-relevant* resolution (or larger). In the case of zero-output, all resolutions are *task-irrelevant*: the true output $f^*(x)$ is not sensitive to any changes in the input. Thus, if we were to use the approach of Section 3.3 to construct hypotheses, we would find that there is only a single hypothesis in the hypothesis space, namely $f*$. Since f^* interpolates the training data and perfectly generalizes, the probability of generalizing given interpolation is 1. Thus, the task difficulty is 0.

However, other choices of the base hypothesis space may lead to very large task difficulties. Consider a version of ImageNet



(a) Varying number of classes on ImageNet

(b) Varying spatial resolution on ImageNet

Figure 5: Task difficulties on parametric variations of benchmark tasks.

in which all inputs are mapped to a 1000-dimensional zero vector. If we construct the base hypothesis space in the same way as for the original ImageNet task, assuming we wish to achieve the same generalization error target as we set for the original ImageNet task, the task difficulty approximation for the zero-output ImageNet would be the *same* as for the original ImageNet: 3.55×10^{41} bits.

How can the inductive bias required to generalize on ImageNet stay the same even when the target function is made much simpler (assuming the hypothesis space is kept the same)? This is because inductive bias complexity corresponds to the difficulty of specifying a generalizing set of hypotheses from the interpolating hypotheses. Due to the large size of the base hypothesis space, the size of the interpolating hypothesis space can be expected to be the same for both the true ImageNet target function and the zero target function: we may expect a the same fraction of hypothesis to satisfy the zero target training samples as the true ImageNet training samples. The base hypothesis space does not over-represent hypotheses near the zero target function relative to other hypotheses. Then, given similarly sized interpolating hypothesis spaces, the difficulty of specifying the true hypothesis is similar for the two tasks.

F.3. Predicting Random Targets

Next, we consider a variation of ImageNet where the ImageNet target function is replaced with a random function selected from the base hypothesis space used for the original ImageNet. In this case, the training set would appear to be a version of ImageNet with randomized outputs for each image. Importantly, note that this is different than independently selecting a random output for each point in the training set: we would training points that are sufficiently close (with distances around the spatial resolution δ or smaller) to have similar outputs. We consider this formulation of randomizing the outputs of ImageNet instead of selecting independent random targets for each ImageNet input since constructing independently chosen random targets for all possible input of ImageNet may not correspond to a well-defined target function.

Assume we wish to achieve the same generalization error rate as for the original ImageNet task. The parameters governing task difficulty would remain the same as in Equation 9: namely, the parameters specific to the input distribution (m, n, d, r), model output and loss function $(b, L_{\mathcal{L}})$ and error rate (ε) would remain the same. Thus, the task difficulty would be the *same* as for the original ImageNet: 3.55×10^{41} bits.

How can solving a randomized version of ImageNet require no more inductive bias than the original task despite having a less structured target function? Intuitively, this is because the base hypothesis space considers all hypothesis equally: it does not over-represent structured hypotheses (such as the target function of ImageNet) relative to other ones. Thus, the difficulty of specifying a well generalizing region in the hypothesis space is the same for both the randomized and original versions of ImageNet.

F.4. Varying Number of Classes

We vary the number of classes in ImageNet (while keeping all other task parameters fixed including the number of training points n) and find in Figure 5a that task difficulty grows with the number of classes. As the number of classes increased from 10 to 1000, the task difficulty increases by roughly 5 orders of magnitude. This is primarily due to the increased size of the input space when more classes are added. The results indicate that adding more classes to a classification task can be a moderately powerful way of increasing the difficulty of a task, although not as powerful as increasing the intrinsic dimensionality of a task.

F.5. Varying Spatial Resolution

We vary the spatial resolution used to construct the base hypothesis space of ImageNet (while keeping all other task parameters fixed) and find in Figure 5b that task difficulty drastically shrinks with the spatial resolution. This makes sense: as the spatial resolution used to construct the hypothesis space grows, the hypothesis space shrinks, and it becomes much more difficult to specify regions in the hypothesis space. These results emphasize the critical effect of the selection of base hypothesis space on task difficulty.

G. Additional Discussion

G.1. How to extend the inductive bias complexity measure to the non-interpolating case?

One limitation of our work is that we only consider interpolating hypotheses (in other words, we only consider training error $\epsilon = 0$ in Definition 3.1). Although we will leave a formal extension of our results to non-interpolating hypotheses as a future work, we briefly outline how are results might be extended:

First, we may provide an analogous result to Theorem 1 in the non-interpolating case by arguing that non-interpolating hypotheses may generalize well as long as they are within a smaller radius of the true hypotheses (relative to the radius for interpolating hypotheses). Specifically, we may expect the more general expression for the radius to be $\frac{\varepsilon-\epsilon}{L_{\mathcal{L}} L_f W(p,q)}$.

Next, in order to quantify probabilities in the hypothesis space and find a practical estimate for inductive bias complexity, we must quantify the size of the hypothesis space that fits the training data up to error ϵ. Intuitively, we may expect it to be a region around the interpolating hypothesis space with dimensionality equal to that of the base hypothesis space. We may expect the size of this region to scale polynomially with ϵ. As with the interpolating case, we can then estimate the fraction of the near-interpolating hypothesis space that is close enough to the true hypothesis to find a generalized, practically-computable expression for task difficulty.

G.2. Why does inductive bias exponentially depend on data dimension?

Intuitively, inductive bias scales exponentially with data dimension because inductive bias scales with the dimensionality of the hypothesis space, and the dimensionality of the hypothesis space scales exponentially with data dimension.

First, we consider the intuition for why inductive bias scales with the dimensionality of the hypothesis space: recall that generalizing requires specifying a specific set of well-generalizing hypotheses in the hypothesis space. The amount of information needed to specify a point in an m dimensional space scales with m since it is simply the number of coordinates of the point. Thus, assuming well-generalizing hypotheses are concentrated in a region around a point, the amount of information required to specify the region also scales with the dimensionality of the hypothesis space.

Next, we consider the intuition for why the dimensionality of the hypothesis space scales with data dimensionality: consider a very simple "grid-based" method of constructing hypotheses in which the input manifold is divided into equally sized hypercubes that tile the entire manifold. A hypothesis consists of a mapping between hypercubes and outputs. Note that the dimensionality of each hypothesis scales with the number of hypercubes since each hypothesis can be specified by its output at each hypercube. The number of hypercubes scales exponentially with the dimensionality of the input manifold; thus the hypothesis space dimensionality also scales exponentially with the dimensionality of the input manifold.

We view this exponential dependence as fundamental: generalizing over more dimensions of variation significantly expands the hypothesis space regardless of how we parameterize hypotheses, and thus dramatically increases inductive bias complexity.

G.3. Why do training points provide so little inductive bias?

Experimentally, we observe that changing the amount of training data, even by many orders of magnitude, does not significantly affect the amount of inductive bias required to generalize. We can interpret this as training data providing relatively little information on which to generalize. Why do training points provide so little information? Intuitively, it is because in the absence of strong constraints on the hypothesis space, training points only provide information about the function we aim to approximate in a *local* neighborhood around each training point. By contrast, inductive biases can provide more *global* constraints on the hypothesis space than can dramatically reduce the size of the hypothesis space and allow for generalization. Without strong inductive biases, training points need to cover large regions of the input manifold to allow for generalization. With high dimensional manifolds, this can require very large numbers of training points.

Through simple scaling arguments, we can estimate the number of training samples we need to generalize on ImageNet in the absence of strong inductive biases. We estimate the intrinsic dimensionality of ImageNet to be 48 and the task-relevant resolution δ of ImageNet to be 65. We can then estimate the number of training points needed to generalize as the number such that any region on the manifold is within radius δ of a training point. With n training points, we can expect to cover a volume of about $n65^{48}$ on the manifold. Given a manifold of volume about $(255 \times 224 \times \sqrt{3})^{48}$ (corresponding to the $[0, 255]$ pixel range and $224 \times 224 \times 3$ extrinsic dimensionality of ImageNet images), we would then require about $(255 \times 224 \times \sqrt{3}/65)^{48} \approx 10^{153}$ points to generalize. Indeed, following the linear trend of Figure 3, we may expect to approximately halve the required inductive bias with this number of training points. However, with only 10^{17} points, strong inductive biases would be necessary to generalize as indicated in Figure 3.

G.4. Why is the scale of inductive bias complexity so large?

Across all settings, our measure yields very large inductive information content, many orders of magnitude larger than parameter dimensionalities of practical models. These large numbers can be attributed to the vast size of the hypothesis space constructed in Section 3.3: it includes *any* bandlimited function on the data manifold below a certain frequency threshold. Typical function classes may already represent only a very small subspace of the hypothesis space: for instance, neural networks have biases toward compositionality and smoothness that may significantly reduce the hypothesis space (see Li et al. (2018); Mhaskar et al. (2017)), with standard initializations and optimizers further shrinking the subspace. Moreover, functions that can be practically implemented by reasonably sized programs on our computer hardware and software may themselves occupy a small fraction of the full hypothesis space. Future work may use a base hypothesis space that already includes some of these constraints, which could reduce the scale of our measured task difficulty.

H. Additional Tables and Figures

Table 6: Mapping different learning settings under a common set of notation.

Setting	x	**Features**	$f^*(x)$
Supervised classification	input	d_1 : class c_1 : instance within class	class
Reinforcement learning	state or observation sequence	c_1 : state or observation sequence	desired action
Meta-learning for few-shot classification	samples from a set of related classes	c_1: a set of related classes c_2: choice of samples from c_1	classifier mapping inputs to their class
Unsupervised autoencoding	input	c_1: input	x
Meta-reinforcement learning	trajectories through environments	c_1: environment c_2: trajectory within environment	policy mapping states to desired actions

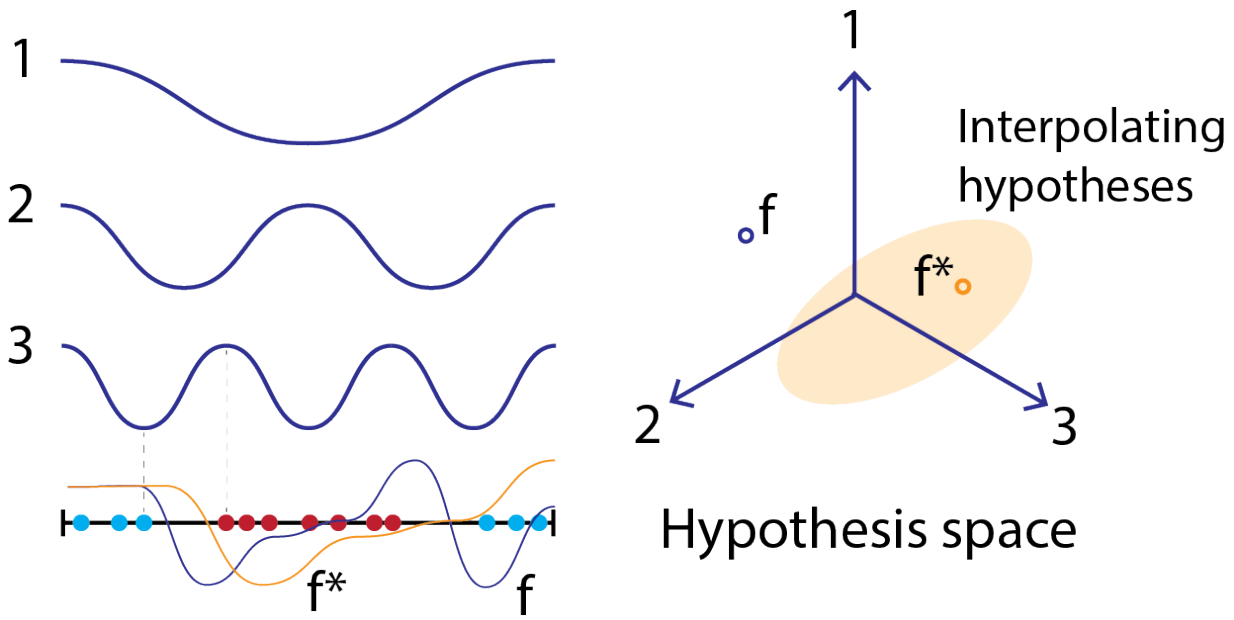


Figure 6: An example of constructing a hypothesis space for a binary-classification task with a 1 dimensional input. The black line indicates the data manifold and the blue and red dots on the line indicate training points. The hypothesis space is constructed using basis functions of three different frequencies; observe that the highest frequency is chosen to have scale corresponding to the minimum distance between classes. Two specific hypothesis are illustrated in the hypothesis space, the true hypothesis (in orange), f^* and another hypothesis f (in purple). Both can be expressed as linear combinations of the basis functions, and thus correspond to points in the hypothesis space as illustrated. The f^* hypothesis fits the training data, and thus is part of the interpolating hypothesis set indicated by the light orange oval.

Table 7: The inductive bias information contributed by different model architectures for image classification tasks. (FC-N refers to a fully connected network with N layers.)

MNIST	
Model	Information Content ($\times 10^{16}$ bits)
Linear (Lecun et al., 1998)	0.705
FC-3 (Lecun et al., 1998)	0.817
AlexNet (mrgrhn, 2021)	0.888
LeNet-5 (CNN) (Lecun et al., 1998)	0.907
DSN (Lee et al., 2015)	0.976
MCDNN (Ciregan et al., 2012)	1.020
Ensembled CNN (An et al., 2020)	1.094

SVHN	
Model	Information Content ($\times 10^{31}$ bits)
FC-6 (Mauch & Yang, 2017)	0.870
AlexNet (Veeramacheneni et al., 2022)	0.985
Deep CNN (Goodfellow et al., 2013)	1.049
DSN (Lee et al., 2015)	1.060
DenseNet (Huang et al., 2017)	1.075
WRN-16-8 (Zagoruyko & Komodakis, 2016)	1.077
WRN-28-10 (Foret et al., 2020)	1.114

CIFAR10	
Model	Information Content ($\times 10^{32}$ bits)
Linear (Nishimoto, 2018)	2.250
FC-4 (Lin et al., 2015)	2.354
MCDNN (Ciregan et al., 2012)	2.704
AlexNet (Krizhevsky et al., 2012)	2.709
DSN (Lee et al., 2015)	2.786
DenseNet (Huang et al., 2017)	3.010
ResNet-50 (Wightman et al., 2021)	3.195
BiT-L (Kolesnikov et al., 2020)	3.453
ViT-H/14 (Dosovitskiy et al., 2021)	3.513

ImageNet	
Model	Information Content ($\times 10^{41}$ bits)
Linear (Karpathy, 2015)	2.063
SIFT + FVs (Sánchez & Perronnin, 2011)	2.261
AlexNet (Krizhevsky et al., 2012)	2.290
DenseNet-121 (Huang et al., 2017)	2.348
ResNet-50 (He et al., 2016)	2.362
DenseNet-201 (Huang et al., 2017)	2.363
WRN-50-2-bottleneck (Zagoruyko & Komodakis, 2016)	2.368
BiT-L (Kolesnikov et al., 2020)	2.449
ViT-H/14 (Dosovitskiy et al., 2021)	2.461