MODEL-AGNOSTIC MEASURE OF GENERALIZATION DIFFICULTY

Anonymous authors
Paper under double-blind review

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 task 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 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* including those provided by a model designer, such as 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 (*the sample complexity*), analysis of the required inductive biases has been limited.

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. In this paper, we make the following contributions [1]:

[1]An anonymized version of our code can be found at: `https://anonymous.4open.science/r/task-difficulty-CF26`

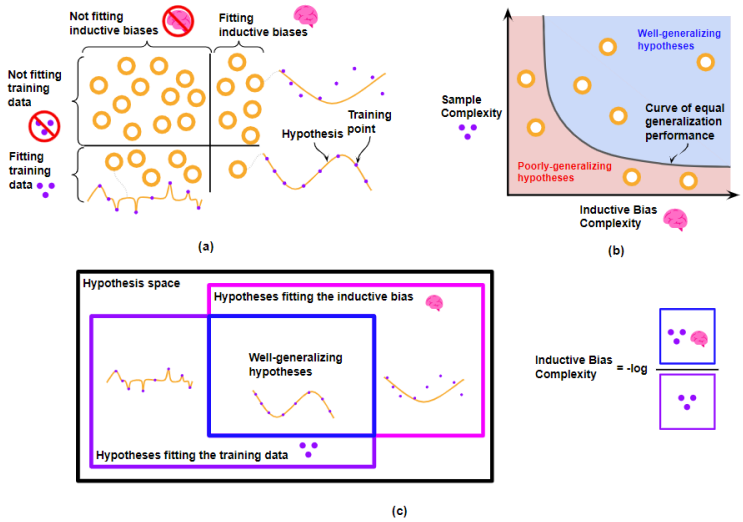


Figure 1: Dividing the hypothesis space of a task. Orange circles are hypotheses and purple dots are training points. (a) categorizes hypotheses by whether or not they fit the training data and whether or not they contain particular inductive biases. Both training data and inductive biases are required to generalize well. (b) plots a trade-off curve 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 target of generalization performance requires higher inductive bias complexity to further restrict the hypothesis space. While many papers have quantified sample complexity, we quantify inductive bias complexity. (c) shows how volumes in the hypothesis space yield a measure of inductive bias complexity: we quantify inductive bias complexity as negative log of the fraction of hypotheses that generalize well among the hypotheses that fit the training data.

- We propose a feature-based task framework to analyze generalization difficulty in various learning problems, including supervised learning, RL and meta-learning.
- We develop a novel and to our knowledge first information-theoretic measure of the inductive bias complexity of a task that quantifies task difficulty as the information content of the inductive biases required for generalization. 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 difficulty of tasks across the domains of supervised learning, RL and few-shot meta-learning tasks. Applied empirically to tasks, we find that 1) task difficulties of partially observed RL environments may be much higher than of fully observed environments and 2) task difficulties of simple few-shot meta-learning tasks can be much larger than complex, realistic supervised learning tasks.

2 Related Work

2.1 Model Oriented Task 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) (Cortes et al., 1994; Murata et al., 1992; Amari, 1993; Hestness et al., 2017). 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 (Negrea et al., 2019; Raginsky et al., 2016; Kawaguchi et al., 2022; Lei et al., 2015; Jiang et al., 2021). These measure leverage the properties of a dataset to show that particular model classes will

generalize well on the dataset. Our approach is aligned in spirit to this line of work; however, instead of leveraging the structure of a task to bound generalization error or sample complexity, we instead 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 Task 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 (Shwartz-Ziv & Tishby, 2017; Saxe et al., 2019; Shamir et al., 2008). 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. However, DIME does not consider the difficulty of *generalizing* from a finite number of samples. 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 Feature-based Task Framework

Here, we develop a single framework that allows us to analyze a number of generalization problems, including supervised classification, meta-learning and RL. The framework explicitly reveals the dimensions along which a learning system needs to generalize, allowing us to analyze the generalization difficulty of many learning problems. Table 1 summarizes our notation.

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
$\mathcal{T}$	Event that the hypothesis fits the training dataset
$\mathcal{K}$	Event that the hypothesis fits inductive biases
ε	Desired test error
$\tilde{I}$	Task difficulty
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
Z	Number of submanifolds
R	Radius of the sphere on which datapoints lie
D	Dimensionality of the model output $f(x;\theta)$
f_k	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 Setup and Notation

We define a task as consisting of *instances* x and *actions* y; a learning system's goal is 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:

$$x = A(c_1, c_2, ...c_{K_c}, d_1, d_2, ...d_{K_d}) \tag{1}$$

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:

$$\hat{e}(f, p) = \mathbb{E}_p[\mathcal{L}(f(x), f^*(x))] \tag{2}$$

3.2 MAPPING THE FRAMEWORK TO DIVERSE LEARNING SETTINGS

Although the task framework above used notation that resembled supervised learning, our framework incorporates many different learning scenarios including unsupervised, reinforcement and meta learning. We summarize the notational mapping between our framework and various learning scenarios in Table 2. Further details of this mapping are given in Appendix A.

Our framework reveals the dimensions along which a task requires generalization. We use this in Section 4 where we quantify the difficulty of *generalizing* on a task. However, our framework may conceal aspects of a task relevant to other notions of difficulty. For instance, in online RL, we do 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: in RL, for instance, 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 4.1); thus, we believe our framework applies to studying generalization in situations like online RL and beyond where the training and test distributions may be quite different.

Table 2: Mapping notation in the feature-based task framework to different learning settings.

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

4 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).

To relate inductive biases and generalization, we use the concept of a *hypothesis space*. Hypotheses may or may not satisfy a training set or a set of inductive biases (see Figure 1 (a)). Generalization fundamentally requires a model to possess inductive biases such that the biases and training data together specify a set of well-generalizing hypotheses. Note that there is a trade-off between sample complexity and inductive bias complexity (see Figure 1 (b)). We aim to quantify the 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 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 large 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 use a Bayesian framework, assuming that f is drawn from some distribution and that the prior distribution of θ is uniform (which can generally be made to hold by reparameterizing). 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 $y(x) = f(x;\theta^*)$.

The generalization error of a specific θ and input distribution p is denoted $e(\theta, p) = \hat{e}(f(\cdot;\theta), p)$. To achieve a desired generalization error ε, we consider constraining the hypothesis space: let $\mathcal{T}$

be the event that our hypothesis fits the training dataset, and $\mathcal{K}$ be the event that our hypothesis fits some inductive biases (see Figure 1c). For ease of analysis, we define "fitting" the training set as perfectly interpolating it. 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. Given the hypotheses fitting the training set, the inductive biases represented by $\mathcal{K}$ guarantee an error ε almost surely if $\mathbb{P}(e(\theta, p) \leq \varepsilon \mid \mathcal{K}, \mathcal{T}) = 1$. These inductive biases have information content $I(\mathcal{K} \mid \mathcal{T}) = -\log \mathbb{P}(\mathcal{K} \mid \mathcal{T})$. Task difficulty $\tilde{I}$ is the minimum information needed to achieve error ε:

$$\tilde{I} = \min_{\mathcal{K},\, \mathbb{P}(e(\theta,p) \leq \varepsilon \mid \mathcal{K}, \mathcal{T}) = 1} I(\mathcal{K} \mid \mathcal{T}). \tag{3}$$

Here, we define task difficulty as the minimum information content of $\mathcal{K}$ such that $\mathcal{K}$ and $\mathcal{T}$ *together* ensure error $\leq \varepsilon$. This aligns with the intuition that more difficult tasks require more inductive bias.

To simplify this expression, note that $\mathbb{P}(e(\theta, p) \leq \varepsilon \mid \mathcal{K}, \mathcal{T}) = 1$ implies that $\mathbb{P}(\mathcal{K} \mid \mathcal{T}) \leq \mathbb{P}(e(\theta, p) \leq \varepsilon \mid \mathcal{T})$, since conditioned on $\mathcal{T}$, $\mathcal{K}$ implies $e(\theta, p) \leq \varepsilon$. Additionally, equality is achieved by setting $\mathcal{K}$ to the event $e(\theta, p) \leq \varepsilon$. Therefore, the task difficulty is

$$\begin{aligned}
\tilde{I} &= \min_{\mathcal{K},\, \mathbb{P}(e(\theta,p) \leq \varepsilon \mid \mathcal{K}, \mathcal{T}) = 1} I(\mathcal{K} \mid \mathcal{T}) && (4)\\
&= -\log \max_{\mathcal{K},\, \mathbb{P}(e(\theta,p) \leq \varepsilon \mid \mathcal{K}, \mathcal{T}) = 1} \mathbb{P}(\mathcal{K} \mid \mathcal{T}) && (5)\\
&= -\log \mathbb{P}(e(\theta, p) \leq \varepsilon \mid \mathcal{T}). && (6)
\end{aligned}$$

This relates the difficulty of a task to the fraction of hypotheses fitting the training set that achieve a test error rate at or below ε. In Section 4.1, we exactly bound this quantity by the fraction of the interpolating hypothesis space that is within a certain radius of the true hypothesis. In Section 4.2, we further approximate this quantity to produce a practically computable estimate.

4.1 An Exact Upper Bound on Task Difficulty

We are able to provide an exact upper bound on the task difficulty given above, under some technical assumptions. Assume that the input, parameter, and output spaces are equipped with standard distance metrics $d(\cdot, \cdot)$ and norms $\|\cdot\|$. 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 1. *Suppose that the loss function satisfies $\mathcal{L}(y_1, y_2) \geq 0$, and equality holds if and only if $y_1 = y_2$. Suppose that the loss function is invariant to shifts $\mathcal{L}(y_1 + y, y_2 + y) = \mathcal{L}(y_1, y_2)$ for all y_1, y_2, y. We further assume the following Lipschitz condition on $\mathcal{L}$: for some constant $L_{\mathcal{L}}$,*

$$\mathcal{L}(y_1 + y, y_2) - \mathcal{L}(y_1, y_2) \leq L_{\mathcal{L}} \|y\| \tag{7}$$

for all y_1, y_2, y. We also assume a second-order condition on f for some constant L_f:

$$\|f(x_1; \theta_1) - f(x_1; \theta_2) - f(x_2; \theta_1) + f(x_2; \theta_2)\| \leq L_f d(x_1, x_2) d(\theta_1, \theta_2) \tag{8}$$

for all $x_1, x_2, \theta_1, \theta_2$. Then we have the following upper bound on the task difficulty:

$$\tilde{I} \leq -\log \frac{\mathrm{vol}\left(\left\{\theta \in \Theta_q \mid d(\theta, \theta^*) \leq \frac{\varepsilon}{L_{\mathcal{L}} L_f W(p,q)}\right\}\right)}{\mathrm{vol}(\Theta_q)}, \tag{9}$$

where vol *denotes volume.*

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.

4.2 Empirical Task Difficulty Approximation

We now apply a series of approximations on the general bound given in Theorem 1 to achieve a practically computable estimate. 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.

The data points x lie on Z submanifolds of dimension m, where Z is the total number of possible combinations of discrete features. We approximate these manifolds as m-spheres with radius R. We define D to be the dimensionality of $f(x;\theta)$, and denote its components by f_k. We let B be a constant such that $\|f(x;\theta)\| \leq B$ for all x, θ. We also denote the size of the training dataset by n.

Approximating Wasserstein Distance Assuming the test distribution p is uniform on the sphere and that the training distribution q is the empirical distribution of n points drawn i.i.d. from p, we can obtain a simple scaling result for the Wasserstein distance $W(p, q)$. The Wasserstein distance is 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; 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 sphere. Although in general, 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. These hypercubes must have side length

$$s = \left(\frac{1}{n} \cdot \frac{2\pi^{(m+1)/2}}{\Gamma\left(\frac{m+1}{2}\right)} R^m\right)^{1/m} = Rn^{-1/m}\left(\frac{2\pi^{(m+1)/2}}{\Gamma\left(\frac{m+1}{2}\right)}\right)^{1/m}. \tag{10}$$

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:

$$W(p, q) \approx s\sqrt{\frac{m}{6}} = Rn^{-1/m}\sqrt{\frac{m}{6}}\left(\frac{2\pi^{(m+1)/2}}{\Gamma\left(\frac{m+1}{2}\right)}\right)^{1/m} \in O(Rn^{-1/m}) \tag{11}$$

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. Also, we may expect this approximation to hold for large n to achieve near uniform q. 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.

Parametrization of Hypotheses 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). For a classification problem, δ may correspond to the distance between classes. We parameterize $f(x;\theta)$ using the approach in McRae et al. (2020), which uses eigenfunctions of the Laplace-Beltrami operator on the manifold as basis functions in a linear parametrization of hypotheses. The maximum frequency of these functions is $M = 2\pi R/\delta$.

For each nonnegative integer k, the Laplace-Beltrami operator on the 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

$$E = \sum_{k=0}^{K} \binom{m+k}{m} - \binom{m+k-2}{m} = \binom{m+K-1}{m} + \binom{m+K}{m} \tag{12}$$

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, the dimensionality of the parameterization of a single component f_k is $2E$. There are D components on each of Z submanifolds, so the dimensionality of the full parameterization θ is $2ZDE$.

Each training point constrains the dimensionality of the possible values of θ by D. Therefore, the dimensionality of Θ_q is $2ZDE - nD$. To approximate the volume of Θ_q, 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 ZD components, we have $\|\theta\| \leq B\sqrt{ZD}$. Thus, we approximate Θ_q as a ball with radius $B\sqrt{ZD}$.

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 ball with radius $\frac{\varepsilon}{L_{\mathcal{L}} L_f W(p,q)}$. Therefore,

$$\frac{\mathrm{vol}\left(\left\{\theta \in \Theta_q \mid d(\theta, \theta^*) \leq \frac{\varepsilon}{L_{\mathcal{L}} L_f W(p,q)}\right\}\right)}{\mathrm{vol}(\Theta_q)} \approx \left(\frac{\frac{\varepsilon}{L_{\mathcal{L}} L_f W(p,q)}}{B\sqrt{ZD}}\right)^{2ZDE-nD}. \tag{13}$$

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.

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 13 along with our approximation for $W(p,q)$ into Equation 9 gives an approximated task difficulty of

$$\tilde{I} \approx (2ZDE - nD)\left(\log B + \frac{1}{2}\log Z + \log D + \log K - \frac{1}{m}\log n - \log\frac{\varepsilon}{L_{\mathcal{L}}} + \log c\right). \tag{14}$$

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 Z is the number of submanifolds on which data points lie, D is the output dimensionality of f, 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 $M = 2\pi R/\delta$.

Properties of Task Difficulty The dominant term in our expression, Equation 14, for task difficulty is $2ZDE - nD$. Since $E \in O(M^m)$, task difficulty is *exponential* in the data's intrinsic dimensionality m. Additionally, $M = 2\pi R/\delta$, so for a fixed m, task difficulty is polynomial in $1/\delta$. Thus, δ impacts task difficulty less than m, but can still be significant for large m. Task difficulty increases roughly linearly in Z and D, so these values have less impact than m and δ. Task difficulty 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, task difficulty increases logarithmically as the desired error ε decreases. Intuitively, the task difficulty depends on the desired performance level: achieving the performance of a random classifier on a classification task is trivial, for example.

5 Empirical Task Difficulty of Benchmarks

Here, we empirically validate our measure of task difficulty 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 14 for specific tasks requires selecting the target generalization error ε and estimating the dimensionality of the data points m and the required detail (resolution) per dimension δ among other parameters. The target error ε is set at a fixed level based on the type of the task (see Appendix C Tab 4). Appendix C describes how other quantities are estimated.

5.1 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 Fig 2, models solving these tasks encode large amounts of inductive bias, with more complex tasks requiring models with more inductive bias.

Few-shot learning Here, 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 Fig 2.

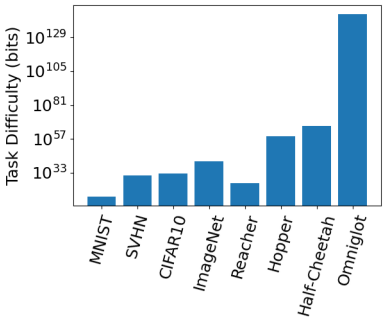


Figure 2: Task difficulties of various benchmark tasks across domains.

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). Fig 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.

5.2 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 evaluate the task difficulty using Equation 14. 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 Tab 3, 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 3: 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

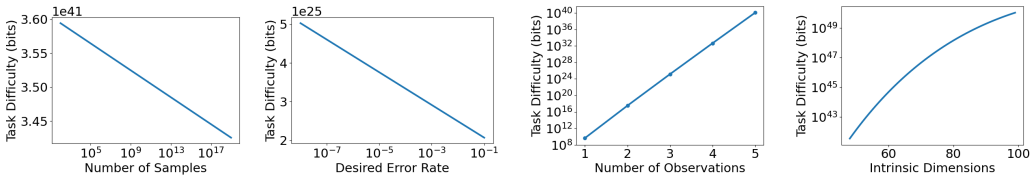
ImageNet

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

5.3 Trends of Task Difficulty

In this section, we consider how task difficulty changes as we parametrically vary aspects of a task.

Varying training data We vary the amount of training data provided in ImageNet and find in Fig 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.



(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 parametric variations of benchmark tasks.

Varying desired error rate In Fig 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.

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, thus requiring possibly multiple observations to find the optimal action. Here, the agent must learn a function with a higher-dimensional input, making the task more difficult. As Fig 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). Fig 3d shows that task difficulty drastically increases with additional intrinsic dimensions. The results reveal that increasing the intrinsic dimension of a task is a powerful way of increasing its difficulty relative to other parametric variations.

6 Discussion

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 4.2: 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.

Our measure reveals that tasks with inputs on higher dimensional manifolds require far more inductive bias relative to other variations of tasks. 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. More generally, in both RL and supervised learning, adding noise to observations/inputs can be a useful way to increase the intrinsic dimensionality of a task and thereby increase its difficulty. These findings suggest that to construct tasks requiring high inductive bias, prioritizing generalization over many degrees of variation is more important than increasing the richness of the task within each dimension of variation. Good benchmarks would require generalization over high dimensional spaces, but, like Omniglot, may appear simple. Examples of such benchmarks include few-shot learning, adversarially robust image classification, and MDPs with very limited observations. Our measure provides a quantitative guide to designing such benchmarks.

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.

REFERENCES

Shunichi Amari. 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.

Sanghyeon An, Minjun Lee, Sanglee Park, Heerin Yang, and Jungmin So. An ensemble of simple convolutional neural network models for mnist digit recognition. *arXiv preprint arXiv:2008.10400*, 2020.

R. S. Anderssen, R. P. Brent, D. J. Daley, and P. A. P. Moran. 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.

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

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

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

Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *CVPR*, 2012.

Corinna Cortes, L. D. Jackel, Sara Solla, Vladimir Vapnik, and John Denker. Learning curves: Asymptotic values and rate of convergence. In *NeurIPS*, 1994.

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

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

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

Simon S. Du, Yining Wang, Xiyu Zhai, Sivaraman Balakrishnan, Ruslan Salakhutdinov, and Aarti Singh. How many samples are needed to estimate a convolutional or recurrent neural network? *NeurIPS*, 2018.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *ICML*, 2016.

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

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

Ian J Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*, 2013.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

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

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint*, 2017.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

Yiding Jiang, Parth Natekar, Manik Sharma, Sumukh K Aithal, Dhruva Kashyap, Natarajan Subramanyam, Carlos Lassance, Daniel M Roy, Gintare Karolina Dziugaite, Suriya Gunasekar, et al. Methods and analysis of the first competition in predicting generalization of deep learning. In *NeurIPS 2020 Competition and Demonstration Track*, 2021.

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

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

Kenji Kawaguchi, Zhun Deng, Kyle Luh, and Jiaoyang Huang. Robustness implies generalization via data-dependent generalization bounds. In *ICML*, 2022.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (BiT): General visual representation learning. In *ECCV*, 2020.

Vladimir Koltchinskii and Dmitriy Panchenko. 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.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 and cifar-100 datasets. *URl: https://www.cs. toronto. edu/kriz/cifar. html*, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

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

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

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *AISTATS*, 2015.

Yunwen Lei, Urun Dogan, Alexander Binder, and Marius Kloft. Multi-class svms: From tighter data-dependent generalization bounds to novel algorithms. *NeurIPS*, 2015.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *NeurIPS*, 2018.

Zhiyuan Li, Yi Zhang, and Sanjeev Arora. Why are convolutional nets more sample-efficient than fully-connected nets? *ICLR*, 2021.

Zhouhan Lin, Roland Memisevic, and Kishore Konda. How far can we go without convolution: Improving fully-connected networks. *arXiv preprint*, 2015.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

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

Andrew McRae, Justin Romberg, and Mark Davenport. Sample complexity and effective dimension for regression on manifolds. *NeurIPS*, 2020.

Hrushikesh Mhaskar, Qianli Liao, and Tomaso Poggio. 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`.

Noboru Murata, Shuji Yoshizawa, and Shun-ichi Amari. Learning curves, model selection and complexity of neural networks. In *NeurIPS*, 1992.

Jeffrey Negrea, Mahdi Haghifam, Gintare Karolina Dziugaite, Ashish Khisti, and Daniel M Roy. Information-theoretic generalization bounds for sgld via data-dependent estimates. *NeurIPS*, 2019.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Ng. Reading digits in natural images with unsupervised feature learning. *NeurIPS*, 2011.

Bruno Eidi Nishimoto. Linear classifier, 2018. URL `https://github.com/brunonishimoto/linear-classifier`.

Phillip Pope, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein. The intrinsic dimension of images and its impact on learning. *ICLR*, 2021.

Maxim Raginsky, Alexander Rakhlin, Matthew Tsao, Yihong Wu, and Aolin Xu. Information-theoretic analysis of stability and bias of learning algorithms. In *IEEE Information Theory Workshop (ITW)*, 2016.

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

Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020, 2019.

Ohad Shamir, Sivan Sabato, and Naftali Tishby. Learning and generalization with the information bottleneck. In *ALT*, 2008.

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

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

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite. *arXiv preprint*, 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, 2012.

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

Lokesh Veeramacheneni, Moritz Wolter, Reinhard Klein, and Jochen Garcke. Canonical convolutional neural networks. *arXiv preprint arXiv:2206.01509*, 2022.

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

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

Dong Yin, Ramchandran Kannan, and Peter Bartlett. Rademacher complexity for adversarially robust generalization. In *ICML*, 2019.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Avnish Narayan, Hayden Shively, Adithya Bellathur, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *CoRL*, 2019.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Peiliang Zhang, Huan Wang, Nikhil Naik, Caiming Xiong, and Richard Socher. DIME: An information-theoretic difficulty measure for AI datasets. *NeurIPS Workshop DL-IG*, 2020.

A SETTINGS OF THE FEATURE-BASED TASK FRAMEWORK

In this section, we provide detailed descriptions of the mapping of various learning settings to our framework.

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:

$$x = A(c_1, d_1). \tag{15}$$

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 (Deng, 2012) 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 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.

$$x = A(c_1, c_2) \tag{16}$$

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.

$$x = A(c_1) \tag{17}$$

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 continuous control task from the MuJoCo suite Tassa et al. (2018). 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:

$$x = A(c_1, c_2) \tag{18}$$

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 1

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

$$\begin{aligned}
\mathcal{L}(f(x_1;\theta), f(x_1;\theta^*)) = \mathcal{L}\Big(& f(x_2;\theta) + (f(x_1;\theta^*) - f(x_2;\theta^*)) \\
& + (f(x_1;\theta) - f(x_1;\theta^*) - f(x_2;\theta) + f(x_2;\theta^*)), \\
& f(x_2;\theta^*) + (f(x_1;\theta^*) - f(x_2;\theta^*))\Big) && (19)\\
\leq \mathcal{L}(& f(x_2;\theta), f(x_2;\theta^*)) \\
& + L_{\mathcal{L}} \| f(x_1;\theta) - f(x_1;\theta^*) - f(x_2;\theta) + f(x_2;\theta^*) \| && (20)\\
\leq \mathcal{L}(& f(x_2;\theta), f(x_2;\theta^*)) + L_{\mathcal{L}} L_f d(x_1, x_2) d(\theta, \theta^*), && (21)
\end{aligned}$$

where equation 20 is a result of the shift-invariance and Lipschitz condition on $\mathcal{L}$, and equation 21 is a result of our condition on f. This allows us to bound the difference

$$e(\theta, p) - e(\theta, q) = \mathbb{E}_p[\mathcal{L}(f(x;\theta), f(x;\theta^*))] - \mathbb{E}_q[\mathcal{L}(f(x;\theta), f(x;\theta^*))] \tag{22}$$

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

$$\begin{aligned}
e(\theta, p) - e(\theta, q) &= \mathbb{E}_\gamma[\mathcal{L}(f(x_1;\theta), f(x_1;\theta^*)) - \mathcal{L}(f(x_2;\theta), f(x_2;\theta^*))] && (23)\\
&\leq \mathbb{E}_\gamma[L_{\mathcal{L}} L_f d(x_1, x_2) d(\theta, \theta^*)]. && (24)
\end{aligned}$$

By the definition of Wasserstein distance,

$$e(\theta, p) - e(\theta, q) \leq L_{\mathcal{L}} L_f d(\theta, \theta^*) \inf_\gamma \mathbb{E}_\gamma[d(x_1, x_2)] = L_{\mathcal{L}} L_f d(\theta, \theta^*) W(p, q), \tag{25}$$

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

$$\tilde{I} = -\log \mathbb{P}(e(\theta, p) \leq \varepsilon \mid \mathcal{T}) \tag{26}$$

$$\leq -\log \mathbb{P}\left(d(\theta, \theta^*) \leq \frac{\varepsilon}{L_{\mathcal{L}} L_f W(p,q)} \mid \theta \in \Theta_q\right) \tag{27}$$

$$= -\log \frac{\mathrm{vol}\left(\left\{\theta \in \Theta_q \mid d(\theta, \theta^*) \leq \frac{\varepsilon}{L_{\mathcal{L}} L_f W(p,q)}\right\}\right)}{\mathrm{vol}(\Theta_q)}, \tag{28}$$

as desired. □

C Empirically Approximating Task Difficulty

In order to compute the difficulty $\tilde{I}$ of a task, we must determine the values of several parameters. These parameters are the number of submanifolds Z, the output dimensionality D of the function we aim to estimate, the dimensionality m of the data points, the desired spatial resolution δ, the number of training data points n, the maximum norm B of the function's output, the target error ε, and the Lipschitz constant $L_{\mathcal{L}}$ of the loss function.

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 4 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.

Table 4: 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 set R to be the maximum norm of the training data points.

C.1 Empirically Computing Classification Task Difficulty

For an image classification task, we consider the class of a data point to be a discrete feature, and set Z to be the number of classes. The model output is a probability distribution on the classes, so we set D to be one less than the number of classes and 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 D.

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 5 for our estimates of m and δ for image classification benchmarks.

Table 5: 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

C.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 $2Z_gD_gE_g$. 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,

$$n = \sum_a \binom{\max\{\ell_a, 20\}}{20} \cdot 20^{20}, \tag{29}$$

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 $B_g\sqrt{Z_gD_g}$. 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.

C.3 Empirically Computing Reinforcement Learning Task Difficulty

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

$$\tilde{I} \approx (2DE - nD)\left(\log B + \frac{1}{2}\log D + \log L_f + \log W(p,q) - \log\frac{\varepsilon}{L_{\mathcal{L}}}\right). \tag{30}$$

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,

$$f(x;\theta) = \sum_{p\in\mathbb{Z}^m} \theta_p e^{ip^\top x} \tag{31}$$

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

$$E \approx \left(\frac{2\pi}{\delta}\right)^m V_m, \tag{32}$$

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

$$W(p, q) \approx \sqrt{\frac{m}{6}} \cdot 2\pi n^{-1/m}. \tag{33}$$

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

$$\tilde{I} \approx D\left(2\left(\frac{2\pi}{\delta}\right)^m V_m - n\right)\left(\log\frac{4\pi^2}{\sqrt{6}} + \log D + \frac{1}{2}\log m - \log\delta - \frac{1}{m}\log n - \log\frac{\varepsilon}{L_{\mathcal{L}}}\right). \tag{34}$$

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.

D 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

$$M_n^{(r)} = \frac{1}{k}\sum_{i=0}^{k-1}(\log X_{n-i,n} - \log X_{n-k,n})^r. \tag{35}$$

Then we can estimate γ as

$$\hat{\gamma}_n = M_n^{(1)} + 1 - \frac{1}{2}\left(1 - \frac{(M_n^{(1)})^2}{M_n^{(2)}}\right)^{-1}. \tag{36}$$

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

$$\frac{a_n^{\hat{\gamma}_n} - 1}{\hat{\gamma}_n} \cdot X_{n-k,n} M_n^{(1)} + X_{n-k,n} \tag{37}$$

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 $1/\delta$.