PAC-Bayes Compression Bounds So Tight That They Can Explain Generalization

Anonymous Author(s)
Affiliation
Address
email

Abstract

While there has been progress in developing non-vacuous generalization bounds for deep neural networks, these bounds tend to be uninformative about why deep learning works. In this paper, we develop a compression approach based on quantizing neural network parameters in a linear subspace, profoundly improving on previous results to provide state-of-the-art generalization bounds on a variety of tasks, including transfer learning. We use these tight bounds to better understand the role of model size, equivariance, and the implicit biases of optimization, for generalization in deep learning. Notably, we find large models can be compressed to a much greater extent than previously known, encapsulating Occam's razor.

1 Introduction

There has been great effort to understand why deep learning models generalize so well on common datasets, despite the fact that these models have many more parameters than the number of training datapoints and can fit random labels [70]. These observations are not explainable through classical statistical learning theory such as VC dimension or Rademacher complexity which focus on uniform convergence over the hypothesis class [51]. The PAC-Bayes framework, in contrast, gives a convenient way of constructing bounds where the generalization gap depends on the deep learning model (or models) that are actually found by training rather than the hypothesis set as a whole. Using this framework, many different potential explanations have been proposed drawing on properties of a deep learning model that are induced by the training dataset, such as low spectral norm [55], noise stability [2], flat minima [30], derandomization [53], robustness, and compression [2, 71].

In this work, we show that neural networks are substantially more compressible than previously believed when paired with structured training datasets. Constructing tighter generalization bounds than have been previously achieved, we show that this compression *alone* is sufficient to explain much of the generalization of neural networks.

Contributions of this Paper

1. We develop a new approach for training compressed neural networks that adapts the compressed size to the difficulty of the problem. We train in selected random subspaces of the parameters and perform learned quantization. Consequently, we achieve extremely low compressed sizes for neural networks at a given accuracy level, which is a requirement for our tight bounds.
2. Using a prior encoding Occam's razor and our compression scheme, we construct the best generalization bounds to date on image datasets, both with data-dependent and data-independent priors. We also show how transfer learning improves compression and thus our generalization bounds, explaining the practical performance benefits of pre-training.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Table 1: Recent non-vacuous PAC-Bayes bounds obtained on popular image classification datasets in deep learning. $\star$ indicates bounds obtained using data-dependent priors (Section 5.2). A ✗ signifies that either the method does not support multi-class problems or that it is completely reliant on data dependent priors and therefore cannot result in data-independent bounds. Additionally, we add Binary MNIST for reference to a benchmark used in earlier works.

	Non-vacuous PAC-Bayes bounds (%)				
Reference	Binary MNIST	MNIST	FMNIST	CIFAR-10	CIFAR-100
Dziugaite and Roy [18]	16.1	✗	✗	✗	✗
Rivasplata et al. [58]	2.2	✗	✗	✗	✗
Zhou et al. [71]		46	91.6	100	100
Dziugaite et al. [19]		$11^\star$	$38^\star$	$23^\star$	✗
Pérez-Ortiz et al. [57]		$21.7/1.5^\star$	49.1	$90.0/16.7^\star$	100
Our bounds		$\mathbf{11.6/1.4}^*$	$\mathbf{32.8/10.1}^*$	$\mathbf{58.2/16.6}^*$	$\mathbf{94.6/44.4}^*$

3. Through the lens of compression and our bounds, we are able to explain why deep learning models generalize on structured datasets like CIFAR-10, but not when structure is broken such as by shuffling the pixels or shuffling the labels, the benefits of equivariant models, and whether the implicit regularization of SGD is necessary for generalization.

2 Related Work

In this section, we delineate recent techniques developed for achieving non-vacuous PAC-Bayes bounds for neural networks.

Optimizing the PAC-Bayes bound Dziugaite and Roy [18] obtain the first non-vacuous generalization bounds for deep stochastic neural networks on binary MNIST. To do so, the authors construct a relaxation of the Langford and Seeger [40] bound and optimize it with the objective of finding a posterior distribution that covers a large volume of low-loss solutions around a local minimum obtained using SGD. Similarly, Rivasplata et al. [58] extend the optimization to other bounds by developing novel relaxations of other PAC-Bayes bounds based on previous work from Blundell et al. [6].

Compressing Models It is well-established that neural networks are robust to small perturbations [29, 30, 39, 38, 36, 55, 8]. The work of Arora et al. [2] leverages this connection to establish a compression-based approach that uses noise stability and the ability to reconstruct weight matrices with random projections is used to study generalization of neural networks. Following the insight on compressibility, Zhou et al. [71] develops a PAC-Bayes bound that uses the representation of a compressed model in bits, and they also add noise stability through their use of Gaussian posteriors and Gaussian mixture priors. Furthermore, they achieve the smallest model representations through pruning and quantization following ideas of Han et al. [27] and Cheng et al. [10].

Data-Dependent Priors Dziugaite et al. [19] demonstrated that for linear PAC-Bayes bounds such as Thiemann et al. [62], a tighter bound can be achieved by choosing the prior distribution to be data-dependent. That is, the prior is trained to concentrate around low loss regions on held-out data. More precisely, the authors show that the optimal data-dependent prior is the conditional expectation of the posterior given a subset of the training data. They approximate this data-dependent prior by solving a variational problem over Gaussian distributions. They evaluate the bounds for SGD-trained networks on data-dependent priors obtaining tight bounds on MNIST, Fashion MNIST, and CIFAR-10. In a similar vein, Pérez-Ortiz et al. [57] combine the data-dependent priors approach Dziugaite et al. [19] with the PAC-Bayes with Backprop (PBB) approach from Rivasplata et al. [58] to obtain *state-of-the-art* PAC-Bayes non-vacuous bounds for MNIST and CIFAR-10 with data-dependent priors. Table 1 summarizes the PAC-Bayes bounds obtained by these works alongside our bounds.

Our upper bound on the KL term closely follows Zhou et al. [71], but with key improvements: (i) we train in lower dimensional subspaces using intrinsic dimensionality and FiLM subspaces which proves to be more effective and adaptable than pruning, (ii) we consider a more effective quantization scheme with variable length codes and quantization aware training, and (iii) we take advantage of the increased compression provided by transfer learning and data dependent priors. As

a result, we obtain state-of-the-art non-vacuous bounds for MNIST, Fashion MNIST, CIFAR-10, and CIFAR-100 in both the from scratch and transfer learning setting as shown in Table 1. Motivated by the tightness of these bounds, we seek to better understand the role of equivariance, stochastic optimization, and data dependent bounds, for explaining generalization.

Ding et al. [16] is another closely related work which optimizes the PAC-Bayes bounds to predict the transferability of various upstream models to a downstream task. In contrast, we aim to explain why neural networks generalize through compression. This is different than predicting transferability as there are several factors other than compression (which we demonstrate is sufficient, but not necessary for generalization) that could make a model perform well on a downstream task.

3 PAC-Bayes Bounds

PAC-Bayes bounds are fundamentally an expression of Occam's razor: simpler descriptions of the data generalize better. As an illustration, consider the finite hypothesis generalization bound. Let $\hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(h\left(x_i\right), y_i\right)$ be the empirical risk of a hypothesis h from a finite hypothesis set $\mathcal{H}$ for the 0-1 loss ℓ, and $R(h) = \mathbb{E}[\hat{R}(h)]$ its population risk. With probability at least $1-\delta$, the population risk for a hypothesis h using n data samples satisfies

$$R(h) \leq \hat{R}(h) + \sqrt{\frac{\log|\mathcal{H}| + \log(1/\delta)}{2n}}, \tag{1}$$

bounded by the empirical risk and a complexity term $\log|\mathcal{H}|$ counting the number of bits needed to specify any $h \in \mathcal{H}$.

But what if we don't believe that each hypothesis is equally likely? If we consider that a given prior distribution P over $\mathcal{H}$ concentrates around the optimal hypothesis, then we can construct a variable length code that uses fewer bits for the more likely hypothesis. With the best possible code for P, a given hypothesis h will take $\log_2 \frac{1}{P(h)}$ bits to represent. Using this prior may allow us to pay for a smaller complexity term even if the hypothesis set is large, so long as the hypotheses that are consistent with the data are also likely under the prior.

The number of bits payed can further be reduced from $\log_2 \frac{1}{P(h)}$ to $\mathbb{KL}(Q \parallel P)$ by considering a distribution of "good" solutions Q. If we don't care which element of Q we arrive at, we can gain some bits *back* from this insensitivity (which could be used to code a separate message). The average number of bits to code a sample from Q using the prior P is the cross entropy $\mathbb{H}(Q, P)$, and one gets on average the entropy $H(Q)$ bits back from being agnostic about which sample $h \sim Q$ to use, yielding the KL-divergence between Q and P: $\mathbb{H}(Q, P) - \mathbb{H}(Q) = \mathbb{KL}(Q \parallel P)$.

With these improvements of the finite hypothesis bound: replacing $\log|\mathcal{H}|$ with $\mathbb{KL}(Q \parallel P)$, and sampling a hypothesis $h \in \mathcal{H}$, one arrives (with minor bookkeeping) at the PAC-Bayes bound introduced in McAllester [49]. With probability at least $1-\delta$,

$$\mathop{\mathbb{E}}_{h\sim Q}\left[R\left(h\right)\right] \leq \mathop{\mathbb{E}}_{h\sim Q}[\hat{R}\left(h\right)] + \sqrt{\frac{\mathbb{KL}(Q \parallel P) + \log(n/\delta) + 2}{2n-1}}. \tag{2}$$

Many refinements of Eq. (2) have been developed [40, 48, 7, 62] but retain the same character. The lower the ratio of the $\mathbb{KL}$ to the number of data points n, the tighter the gap between empirical and expected risk. In this work, we use the tighter Catoni [7] variant of the PAC-Bayes bound (see Appendix I for details).

Universal Prior. Connecting back to Occam's razor, one can define a prior that explicitly penalizes the minimum compressed length of the hypothesis, also known as the universal prior [60]: $P(h) = 2^{-K(h)}/Z$, where K is the *prefix* Kolmogorov complexity [33] of h (the length of the shortest program that produces h and also delimits itself), and $Z \leq 1$.[1] Using a point mass posterior on a single hypothesis h^*, $\mathbb{KL}\left(Q \parallel P\right)$ from the bound reduces to

$$\mathbb{KL}\left(\mathbf{1}_{[h=h^*]} \parallel P(h)\right) = \log \frac{1}{P(h^*)} \leq K(h^*)\log 2 \leq l(h^*)\log 2 + 2\log l(h^*),$$

[1] The universal prior similar to the discrete hypothesis prior from Zhou et al. [71] but setting $m(h) = 2^{-2\log_2 l(h)}$ rather than the flat $m(h) = 2^{-72}$.

where $l(h)$ is the length of a given program that reproduces h not including the delimiter, which we can use when upper bounding the KL. For convenience, we can condition on using the same method for compression and decompression for all elements of the prior. If we can reduce the compressed length $l(h)$ of the hypotheses we find that fits the training data, then we can improve the tightness of the PAC-Bayes bound.

Model Compression. *Model compression* aims to find a nearly equivalent model that can be expressed fewer bits and is useful for deployments on mobile devices or for improving inference time on specialized hardware [9, 10]. For computing PAC-Bayes generalization bounds, however, only the model size matters. Therefore, we can employ compression methods which may otherwise be unfavorable in practice due to worse computational requirements. *Pruning* and *quantization* are among the most widely used methods for model compression. In this work, we rely on quantization (Section 4.2) to achieving tighter generalization bounds.

4 Tighter Generalization Bounds via Adaptive Subspace Compression

Training a neural network involves taking many gradient steps in a high-dimensional space $\mathbb{R}^D$. Although D may be large, the loss landscape has been, in some sense, found to be simpler than typically believed [14, 26, 23]. Analogous to the notion of intrinsic dimensionality more generally, Li et al. [44] attempt to find the lowest dimensional subspace in which the network can be trained and still fit the training data. The weights of a neural network $\theta \in \mathbb{R}^D$ are parametrized in terms of an initialization θ_0 and a projection $w \in \mathbb{R}^d$ to a lower dimensional subspace through a fixed matrix $P \in \mathbb{R}^{D \times d}$,

$$\theta = \theta_0 + Pw. \tag{3}$$

To facilitate favorable conditioning during optimization, P is chosen to be approximately orthonormal $P^\top P \approx I_{d\times d}$ and for scalability, Li et al. [44] utilize random normal matrices of the form $P \sim \mathcal{N}(0,1)^{D\times d}/\sqrt{D}$, and their sparse approximations [45, 42].

In its original form, intrinsic dimensionality (ID) is used as a scientific tool to measure the complexity of the learning task. Unlike other methods like pruning, ID has the benefit of scaling dramatically up or down in size with difficulty of the problem. While ID was not used with quantization as a full fledged model compression method, we find that it can be. We also highlight that concepts similar to the intrinsic dimensionality of a model have been explored in the context of model compression for estimating bounds as seen in Arora et al. [2]. For our work, the ability to find the intrinsic dimension $d \ll D$ has profound implications for the compressibility of the models, and therefore our ability to construct generalization bounds. As demonstrated by Zhou et al. [71], the compressibility of neural network models has a direct connection to generalization and immediately allows us to compute non-vacuous PAC-Bayes bounds for transfer learning. Building upon ID, we describe our approach next.

4.1 Finding Better Random Subspaces to Train In

To further improve upon the scalability and effectiveness of the projections P used by Li et al. [44], we introduce three new methods.

Kronecker Sum Projector: Using the Kronecker product $\otimes$, we construct the matrix $P_\oplus = (\mathbf{1} \otimes R_1 + R_2 \otimes \mathbf{1})/\sqrt{2D}$ where $R_1, R_2 \sim \mathcal{N}(0,1)^{\sqrt{D}\times d}$ and $\mathbf{1}$ is the vector of all ones in $\mathbb{R}^{\sqrt{D}}$. From the fact that $R_1 \perp\!\!\!\perp R_2$ and the entries are standard normal, one can verify that $P_\oplus^\top P_\oplus = I_{d\times d} + \mathcal{O}(1/\sqrt{D})$.

Kronecker Product Projector: Alternatively, we can form the matrix $P_\otimes = Q_1 \otimes Q_2/\sqrt{D}$ with the smaller $Q_1, Q_2 \sim \mathcal{N}(0,1)^{\sqrt{D}\times\sqrt{d}}$, and again this matrix is approximately orthogonal: $P_\otimes^\top P_\otimes = (Q_1^\top Q_1/\sqrt{D}) \otimes (Q_2^\top Q_2/\sqrt{D}) = I \otimes I + O(D^{-1/4}) = I_{d\times d} + O(D^{-1/4})$. [2]

[2] As neither D nor d is typically a perfect square, we concatenate a dense random matrix to pad out the difference between D, d, and a perfect square. As $(\sqrt{D}+1)^2 = D + 2\sqrt{D} + 1$, we have that the size of

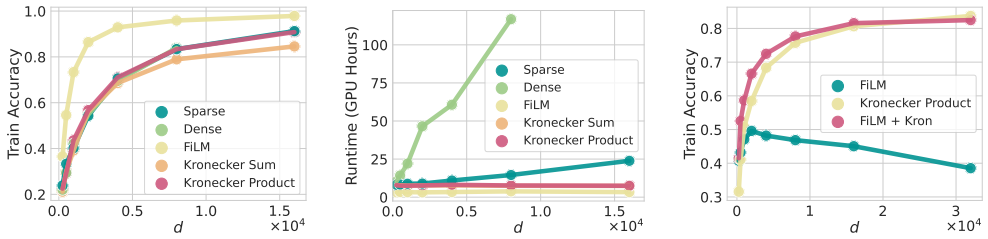

Figure 1: **(Left)** Different projection operators P (Section 4.1) used for transfer learning from Imagenet to CIFAR-10 on a ResNet-34 across different subspace dimensions d. Kronecker product, Sparse, and Dense perform almost identically **(Center)** Kronecker product runs with substantially reduced the runtime cost compared to the Sparse or Fastfood matrices used by Li et al. [44]. **(Right)** Training from scratch on CIFAR-10. The FiLM projector alone is unable to fit the data when training from scratch, and instead a sum of FiLM and Kronecker Product projectors perform the best.

The matrix vector multiply $w \mapsto Pw$ for both the above projectors can be performed in time $O(d\sqrt{D})$ and $O(\sqrt{dD})$ respectively, rather than the $O(dD)$ that is required by the dense random matrix. Figure 1 demonstrates the runtime speedup and training performance improvement in comparison to the methods using by Li et al. [44]. Notably, the Kronecker-structured projections retain the fidelity of the dense random matrix while being orders of magnitude faster than the alternative operators when scaling to larger values of d.

FiLM projector: It is well known that the BatchNorm parameters have an outsized effect on the downstream task performance relative to their size, and this fact has been used in Featurewise independent Linear Modulation (FiLM) [56, 17] for efficient control of neural networks in many different settings. Several authors have explored performing finetuning for transfer learning solely on these parameters and the final linear layer [35]. Drawing on these observations, we construct a projection matrix P_{FiLM} where only columns corresponding with BatchNorm or head parameters are non-zero and sampled from $\mathcal{N}(0,1)^d/\sqrt{D}$, which we also show in Figure 1. While the FiLM projector is highly effective for transfer learning (shown in Figure 1 left), the performance saturates quickly when training from scratch. For this reason, when training from scratch we employ the sum $P_{\text{FiLM}+\otimes} = (P_{\text{FiLM}} + P_{\otimes})/\sqrt{2}$, which outperforms the two projectors individually as shown in Figure 1 right.

4.2 Quantization Scheme and Training

Through quantization, the average number of bits used per parameter can be substantially reduced. When optimizing purely for model size rather than efficiency on specialized hardware, one can choose non-linearly spaced quantization levels which are learned, and use variable length coding schemes as shown in Han et al. [27]. On the other side, the straight through estimator has been central to learning weights in binary neural networks [31]. We combine these ideas to simultaneously optimize the quantized weights and the quantization levels for maximum compression.

Given the full precision weights $w = [w_1, \ldots, w_d] \in \mathbb{R}^d$ and a vector $c = [c_1, \ldots c_L] \in \mathbb{R}^L$ of L quantization levels, we construct the quantized vector $\hat{w} = [\hat{w}_1, \ldots, \hat{w}_d]$ such that $\hat{w}_i = c_{q(i)}$ where $q(i) = \arg\min_k |w_i - c_k|$. The quantization levels c are learned alongside w, where the gradients are defined using the straight through estimator [3, 68]:

$$\frac{\partial \hat{w}_i}{\partial w_j} = \delta_{ij} \quad \text{and} \quad \frac{\partial \hat{w}_i}{\partial c_k} = 1_{[q(i)=k]} \tag{4}$$

We initialize c with uniform spacing between the minimum and maximum values in parameter vector w, which performs better than k-means initialization, as also observed in Choi et al. [11]. To further compress the network, we use a variable length code in the form of arithmetic coding [47], which takes advantage of the fact that certain quantization levels are more likely than others. Given probabilities p_k (empirical fractions) for cluster c_k, arithmetic coding of w takes at most

this padding is at most $\sqrt{D} \times \sqrt{d}$, so it does not increase the asymptotic cost of performing the matrix vector multiplies.

Algorithm 1 Compute PAC-Bayes Bound.

Inputs: Neural network f_θ, Training dataset $\{x_i, y_i\}_{i=1}^n$, Clusters L, Intrinsic dimension d, Confidence $1-\delta$, and Prior distribution P.
function COMPUTE_BOUND($f_\theta, L, d, (x_i, y_i)_{i=1}^n, \delta, P$)
    $w \leftarrow$ TRAIN_ID(f_θ, d, $(x_i, y_i)_{i=1}^n$) ▷ (Section 4.1)
    $\hat{w} \leftarrow$ TRAIN_QUANTIZE(w, L, $(x_i, y_i)_{i=1}^n$)
    Compute quantized train error $\hat{R}(\hat{w})$.
    $\mathbb{KL}(Q, P) \leftarrow$ GET_KL($\hat{w}, P$) ▷ (Section 3)
    return GET_CATONI_BOUND($\hat{R}(\hat{w})$, $\mathbb{KL}(Q, P)$, δ, n) ▷ (Section 3)
end function
function TRAIN_QUANTIZE($w, L, (x_i, y_i)_{i=1}^n$) ▷ (Section 4.2)
    Initialize $c \leftarrow$ GET_CLUSTERS(w, L)
    for $i = 1$ to quant_epochs **do**
        $c \leftarrow c - \rho \nabla_c \mathcal{L}(w, c)$ and $w \leftarrow w - \rho \nabla_w \mathcal{L}(w, c)$
    end for
    return $\hat{w}$
end function
function GET_KL($(\hat{w}, P)$)
    c, count $\leftarrow$ GET_UNIQUE_VALS_COUNTS($\hat{w}$)
    message_size $\leftarrow$ DO_ARITHMETIC_ENCODING($\hat{w}$, c, count)
    message_size $\leftarrow$ message_size + hyperparam_search ▷ (Section 4.3)
    return message_size $+ 2 \times \log$(message_size)
end function

$\lceil d \times \mathbb{H}(p) \rceil + 2$ bits, where $\mathbb{H}(p)$ is the entropy $\mathbb{H}(p) = -\sum_k p_k \log_2 p_k$. For a small number of quantization levels, we find it makes a small improvement in the compression over more common Huffman coding.

In summary, we use $\lceil d \times \mathbb{H}(p) \rceil + 2$ bits for coding the quantized weights $\hat{w}$, $16L$ bits for the codebook c (represented in half precision), and additional $L \times \lceil \log_2 d \rceil$ bits for representing the probabilities p_k, arriving at $l(w) \leq \lceil d \times \mathbb{H}(p) \rceil + L \times (16 + \lceil \log_2 d \rceil) + 2$.

4.3 Subspace Dimension Optimization and Hyperparameters in the Universal Prior

The smaller the chosen intrinsic dimension d, the more similar θ is to the initialization θ_0 in Eq. (3). Consequently, that value of θ is more likely under the universal prior given the shorter description length. Note that in this prior, we condition on the random seed used to generate θ_0 and P. As we optimize over different parameters such as the subspace dimension $d = 1, .., D$, and possibly other hyperparameters such as the learning rate, or number of quantization levels L, we must encode these into our prior and thus pay a penalty for optimizing over them. We can accomodate this very simply by considering the hypothesis h as not just specifying the weights, but also specifying these hyperparameters: $h = (\theta, d, L, \mathrm{lr})$, and therefore using the universal prior $P(h) = 2^{-K(h)}/Z$ we pay additional bits for each of these quantities: $K(h) \leq K(\theta|d, L) + K(d) + K(L) + K(\mathrm{lr})$. If we optimize over a fixed number H of distinct values known in advance for a given hyperparameter such as L, then we can code L using this information in only $\log_2(H)$ bits. In general, we can also bound the dimensionalities searched over by the maximum D so that $K(d) \leq \lceil \log_2 D \rceil$ in any case.

For transfer learning, we replace θ_0 with a learned initialization $\theta_\mathcal{D}$ that is found using the pretraining task and data $\mathcal{D}$. With the ID compression, the universal prior $P(h|\mathcal{D})2^{-K(h|\mathcal{D})}/Z$ will place higher likelihood on solutions θ that are close to the pretraining solution $\theta_\mathcal{D}$.

5 Empirical Non-Vacuous Bounds

5.1 Non-Vacuous PAC-Bayes Bounds

To compute our bounds on benchmark datasets, we use the joint approach of projected subspace training, parameter quantization, and optimizing the subspace dimension with respect to the bounds

Table 2: Our PAC-Bayesian Subspace Compression Bounds compared to *state-of-the-art* (SoTA) bounds. All results are obtained with 95% confidence, i.e. $\delta = .05$. All data-independent SOTA numbers are achieved by [57].

Dataset	Data-independent priors		Data-dependent priors	
	Err. Bound (%)	SoTA (%)	Err. Bound (%)	SoTA (%)
MNIST	**11.6**	21.7	**1.4**	1.5 [57]
+ SVHN Transfer	**9.0**	16.1		
FashionMNIST	**32.8**	46.5	**10.1**	38 [19]
+ CIFAR-10 Transfer	**28.2**	30.1		
CIFAR-10	**58.2**	89.9	**16.6**	16.7 [57]
+ ImageNet Transfer	**35.1**	54.2		
CIFAR-100	**94.6**	100	**44.4**	–
+ ImageNet Transfer	**81.3**	98.1		

as summarized in Algorithm 1. We also optimize other bound hyperparameters such as the learning rate of the quantization-aware training. We offer more details about bound optimization in Appendix D. Our results for *data-independent* priors, i.e., priors that do are not pre-trained on a subset of the training data, are shown in Table 2. We derive the first non-vacuous bounds on FashionMNIST, CIFAR-10, and CIFAR-100 without data-dependent priors. These results are of significance as we argue in Section 5.2 that using data-dependent priors are not explanatory about the learning process. In particular, we improve over the compression bound results obtained by Zhou et al. [71] on MNIST from $< 46\%$ to 11.55%. We also dramatically improve their compression results as we reduce the compressed size for the best bound by 94% bringing it down from 6.23 KB to 0.38KB for MNIST with LeNet5. The tightness of our SOTA subspace compression bounds allows us to improve the understanding of several deep learning phenomena as discussed in Section 6. See Appendix A for additional results.

5.2 Data-Dependent Priors

So far, we demonstrated the strength of our bounds on *data-independent* priors, where we considerably improve on the state-of-the-art. However, a number of recent papers have considered data-dependent priors as a way of achieving tighter bounds [57, 19]. In this setup, the training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ is partitioned into two parts, $\mathcal{D}_a$ and $\mathcal{D}_b$, with length $n - m$ and m. The first dataset is used to construct a data dependent prior $P(h|\mathcal{D}_a)$, and then the bound is formed over the remaining part of the process: the adaptation of the prior $P(h|\mathcal{D}_a)$ to the posterior $Q(h)$ using the data $\mathcal{D}_b$. The empirical risk is computed over $\mathcal{D}_b$ only. Intuitively, using dataset $\mathcal{D}_a$ it is possible to construct a much tighter prior over the possible neural network solutions.

In our setting, similar to transfer learning, we use the prior $P_{\mathcal{D}_a}(\theta) = 2^{-K(\theta|\theta_{\mathcal{D}_a})}/Z$ where for compression we use $\theta = \theta_{\mathcal{D}_a} + Pw$, and $\theta_{\mathcal{D}_a}$ is the solution found by training the model (without random projections) on the data $\mathcal{D}_a$ rather than initializing randomly. With these data-dependent priors, we achieve the best bounds in Table 2.

However, our adaptive approach exposes a significant downside of data-dependent priors. To the extent that PAC-Bayes bounds can be used for explanation, in this case they only yield insights into the procedure used to adapt $P_{\mathcal{D}_a}(\theta)$ to Q using $\mathcal{D}_b$, any learning that is done in finding $P_{\mathcal{D}_a}(\theta)$ is not constrained or explained by the bound (we have no information on the compressibility of $\theta_{\mathcal{D}_a}$). Given the ability to adapt the size of the KL to the difficulty of the problem, it is possible to squeeze all of the learning into $P_{\mathcal{D}_a}(\theta)$ and none in this adaption to Q. This happens as the $KL \to 0$, which we find happens empirically (or very nearly so) for many splits of the data where $n - m$ is large. Setting $Q(\theta) = \mathbf{1}_{[\theta=\theta_{\mathcal{D}_a}]}$, the KL has only the contribution from the optimization over d: $\mathbb{KL}(Q||P_{\mathcal{D}_a}) \leq \log D$ we find that the bound is nothing more than a variant of the simple Hoeffding bound where $\mathcal{D}_b$ is the validation set $R\left(\theta_{\mathcal{D}_a}\right) \leq \hat{R}_{\mathcal{D}_b}\left(\theta_{\mathcal{D}_a}\right) + \sqrt{\frac{\log(Dm/\delta)+2}{2m-1}}$.

This phenomenon can be seen in Appendix C, where optimization over $d = 1, ..., D$ leads to very small values of the KL when $n - m$ is a large fraction of n, and hence highly simple solutions approaching the Hoeffding bound or confidence interval. This demonstrates that data-dependent priors based PAC-Bayes bounds are not particularly meaningful, and attempts to optimize these

bounds will lead inevitably to the statistical bounds formed from the validation set such as a simple Hoeffding bound or confidence interval. While these bounds are correct, they are only informative about the difference in performance of the pretrained model on validation data and test data.

Note that this is in contrast with the data-independent bounds that constrain and explain generalization for the entirety of the learning process. Similarly, our transfer learning bounds meaningfully constrain what happens in the finetuning on the downstream task, but they do not constrain the prior determined from the upstream task.

5.3 Non-Vacuous PAC-Bayes Bounds for Transfer Learning

By directly interpreting PAC-Bayes bounds through the lens of compression, we immediately see the benefits of employing an upstream dataset for transfer learning. In fact, transfer learning allows us to constrain the prior around parameters we know are consistent with the upstream task, and hopefully the downstream task as well, reducing the KL-divergence between the prior and the posterior and leading to even tighter bounds as we show in Table 2. We obtain unequivocally tighter bounds using transfer learning for data-independent priors, providing a theoretical certification that transfer learning can improve generalization. Our PAC-Bayes transfer learning approach also indicates that transfer learning can boost generalization whenever codings optimized on a pre-training task are more efficient for encoding a downstream posterior than an a priori guess made before seeing data. In contrast, downstream tasks which differ greatly from the upstream task may only be consistent with models that are not compressible under the learned prior, a scenario which may describe negative transfer. As pre-training becomes the default in deep learning, our bounds open the door to providing learning guarantees for state-of-the-art techniques. See Appendix B for more details.

6 Understanding Generalization through PAC-Bayes Bounds

Unlike the classical statistical learning theory viewpoint of uniform convergence, where the focus is on the properties of the hypothesis class as a whole, the ability of a given network to be compressed is highly dependent on the actual dataset that the model was trained on. In this respect, the PAC-Bayes perspective shows us that the ability to compress and therefore generalize is not a function of the hypothesis space but rather of individual functions: there are many networks in that space that fit corrupted data and provably cannot be compressed (see Appendix A.1). But these models are not the ones that we find when training on real world datasets, as real world datasets have a tremendous amount of structure. If they did not, it would be pointless trying to learn from them as famously argued by Hume [32] and the No Free Lunch theorem [65, 25]. When training on these structured datasets that have low Kolmogorov complexity, the compressibility of the dataset is reflected in the compressiblity of the model. Breaking the structure in the dataset also breaks the structure in the model and leads to worse generalization bounds for shuffled labels, shuffled pixels, and equivariance more generally.

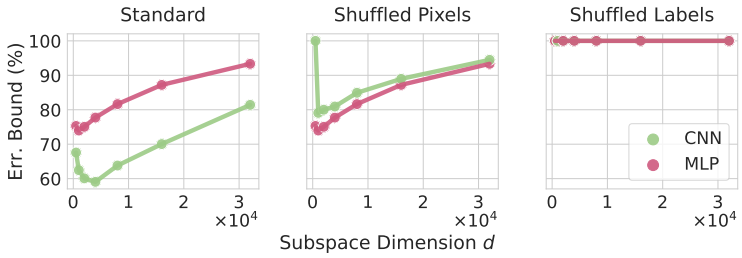

Figure 2: Our PAC-Bayes Bound vs subspace dimension for a fixed size convolutional network and MLP both with 500K parameters. With training on **(Left)** Standard CIFAR-10 **(Center)** CIFAR-10 with shuffled pixels **(Right)** CIFAR-10 with shuffled labels. Structure in the dataset induces structure in the model. As structure is removed from the dataset, the models become much less compressible hence generalize worse. When the image structure is destroyed with pixel shuffling, we find that the CNN generalizes worse than an MLP.

MLPs vs CNNs: It is well known that convolutional neural networks (CNNs) generalize much better than standard multilayer perceptrons (MLPs) even when controlling for the number of parameters. We find that this fact is reflected in the improved compressibility of CNNs and our generalization bounds, shown in Figure 2 (left) on the image dataset CIFAR-10 using the same number of parameters for the two models. The CNN is able find a more structured and lower description length explanation of the training data.

Shuffled Pixels: However, when this image structure is broken by shuffling the pixels we find that CNNs are no better at generalizing than MLPs, and this is reflected in the compressed sizes. The CNNs become substantially less compressible when trained on this dataset, and hence our bounds show them generalizing worse than MLPs, see Figure 2 (center). Not using this image structure (and invariant to a global permutation of the features), MLPs do not suffer when this structure is broken since they never used it in the first place.

Shuffled Labels: When the structure of the dataset is entirely broken by shuffling the labels, the compressibility of the models (both for CNNs and MLPs) are lost. Regardless of the subspace dimension used, our generalization bounds are all at 100% error as shown in Figure 2 (right). Using low subspace dimensions it is not possible to fit the training data, and when using a large enough dimension to fit the data the compressed size of the model is larger than the training data and hence the generalization bounds are vacuous.

Equivariance: Designing models which are *equivariant* to certain symmetry transformations has been a guiding principle for the development of data efficient neural networks in many domains [12, 13, 63, 64, 22, 34]. While intuitively it is clear that respecting dataset symmetries severely improves generalization, relatively little has been proven for neural networks [21, 72, 4, 20]. We compress and evaluate rotationally equivariant and non-equivariant Wide ResNets [64, 69] trained on MNIST and a rotated version of MNIST. As shown in Appendix E, the rotationally equivariant models are more compressible and provably generalize better than their non equivariant counterparts when paired with a dataset that also has the rotational symmetry.

7 Discussion

In this work, we have constructed a new method for compressing deep learning models which is highly adaptive to the level of structure in the model and the training dataset. According to the Occam's prior which considers shorter compressed length models to be more likely, we construct state-of-the-art generalization bounds across a variety of settings. Through our bounds and the viewpoint of Occam's razor and compressibility, we show how generalization relates to the structure in the dataset and the structure in the model, and we are able to explain aspects of neural network generalization for natural image datasets, shuffled pixels, shuffled labels, equivariant models, and for stochastic training.

Limitations We note that despite the power of this compression scheme and our bounds, there are certain aspects of deep learning generalization that remain unexplained. Our compression bounds still show a preference towards models with a smaller number of base parameters as shown in Appendix G, despite the fact that larger models tend to generalize better. While we have achieved better model compression than previous works, it is unlikely that we are close to theoretical limits. We hope that with a further improved compression scheme (possibly using nonlinear parameter projections), we might find that larger deep learning model solutions are compressible to smaller sizes than smaller models. There is also a question of whether models not trained in a special way can be related and bounded by the performance of the compressed models, perhaps leveraging ideas from Nagarajan and Kolter [50] and others investigating this from a theoretical lens, and this would be helpful for applying our findings more broadly. Additionally, while our bounds show that the compressibility of our models implies generalization, we have no statements in the reverse direction. We believe however that the compression approach and Occam's razor have yet untapped explanatory power in deep learning.