Non-Vacuous Generalization Bounds for Large Language Models

Sanae Lotfi [*1] **Marc Finzi** [*2] **Yilun Kuang** [*1]
Tim G. J. Rudner [1] **Micah Goldblum** [1] **Andrew Gordon Wilson** [1]

Abstract

Modern language models can contain billions of parameters, raising the question of whether they can generalize beyond the training data or simply parrot their training corpora. We provide the first non-vacuous generalization bounds for pretrained large language models (LLMs), indicating that language models are capable of discovering regularities that generalize to unseen data. In particular, we derive a compression bound that is valid for the unbounded log-likelihood loss using prediction smoothing, and we extend the bound to handle subsampling, making bound computation 900 times faster on massive datasets. To achieve the extreme level of compression required for non-vacuous bounds, we devise SubLoRA, a simple low-dimensional nonlinear parameterization that leads to non-vacuous generalization bounds for very large models with up to 849 million parameters. Finally, we use our bounds to understand LLM generalization and find that larger models have better generalization bounds and are more compressible than smaller models.

1. Introduction

Do large language models (LLMs) merely memorize the training data, and if so, are they able to meaningfully generalize beyond their training set? This question is central to understanding LLMs as they continue to grow in capacity and are capable of memorizing and parroting training examples verbatim (Brown et al., 2020; Chowdhery et al., 2022; Carlini et al., 2020; 2023).

In this work, we address the question of generalization in LLMs by computing the first non-vacuous generalization bounds for language model pretraining on next token prediction, thereby providing a mathematical guarantee that LLMs are able to generalize beyond their training data.

Although significant progress has been made in constructing non-vacuous generalization bounds for image classification models using the PAC-Bayes framework (Catoni, 2007) in conjunction with extreme levels of model compression (Zhou et al., 2019; Lotfi et al., 2022), non-vacuous generalization bounds for large language models remain elusive.

Compared to image classification models, constructing non-trivial bounds for language models presents additional challenges: (i) LLMs are trained on autoregressive token prediction, and thus token predictions are not independent; (ii) the relevant negative log-likelihood (NLL) metric (bits per dimension) is a continuous and unbounded random variable for which previously used non-vacuous PAC-Bayes bounds are invalid; and (iii) LLMs have orders of magnitude more parameters than image classification models. To address these challenges, we derive new generalization bounds that can be applied to the unbounded bits per dimension objective. We also introduce an extension of these bounds which can be computed using only a subset of the training data, making bound computation 900 times faster on the OpenWebText dataset, which has more than 9 billion tokens.

Achieving the extreme level of compression required to obtain non-vacuous generalization bounds for LLMs is another challenge. To this end, we devise SubLoRA (Subspace-Enhanced Low-Rank Adaptation): simple nonlinear parameterization for LLMs that makes it possible to smoothly vary the level of compression while maintaining expressivity. SubLoRA combines low-rank adaptation (LoRA) (Hu et al., 2021), originally proposed for efficient *fine-tuning*, with subspace training (Li et al., 2018; Lotfi et al., 2022) to *pretrain* highly compressed LLMs from scratch.

Combining the above-described theoretical and practical contributions, we achieve *the first non-vacuous bounds for large language models*. To highlight the efficiency of our new compression technique, we compare SubLoRA to LoRA and subspace training in Figure 1 (left). We compute two metrics that we define as follows: Top-1 Error, which is the 0-1 error in predicting the next token averaged over a given document; and the bits per dimension metric, which

[*]Equal contribution [1]New York University [2]Carnegie Mellon University. Correspondence to: Sanae Lotfi <sl8160@nyu.edu>, Andrew Gordon Wilson <andrewgw@cims.nyu.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

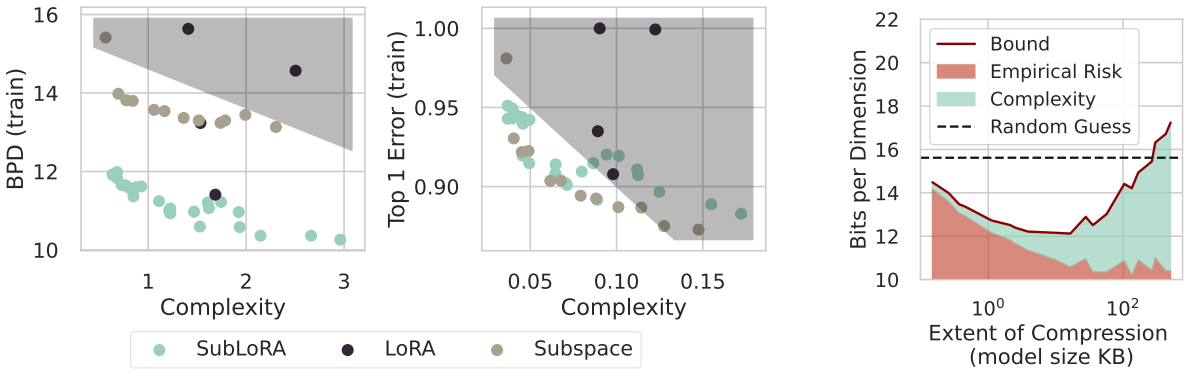

Figure 1. **Finding solutions that simultaneously achieve low training error and low complexity with SubLoRA. (Left):** The Pareto frontier of model complexity (the 2nd term in Equation 1) and the empirical risk (bits per dimension (BPD) and Top-1 Error) of language models using LoRA and subspace compression for next token prediction pretraining. The generalization bound is formed from the sum of the two axes (lower is better), with the shaded region showing where bounds are vacuous. Combining both LoRA and subspace compression in the form of SubLoRA yields the best bounds, while using LoRA alone yields vacuous bounds for top-1 error. **(Right):** SubLoRA enables a smooth tradeoff over the extent of model compression for a fixed model, finding the degree of compression that is optimal for the situation in constructing the generalization bounds. We plot the contributions of the empirical risk and the complexity term to the bound as a function of this degree of compression.

corresponds to the average negative log-likelihood per document. The shaded region highlights where bounds become vacuous, with SubLoRA achieving non-vacuous bounds for both bits per dimension and Top-1 Error. The term *vacuous* refers to the random guess performance which is $\log_2 V$ for BPD and $1 - 1/V$ for Top-1 Error, where V is the vocabulary size. In contrast, we see that only using LoRA achieves vacuous bounds for Top-1 Error and only using subspace achieves a high value of empirical BPD. Despite the simplicity of SubLoRA, it has an improved ability to trade-off model complexity with training error. In Figure 1 (right), we highlight the trade-off between model complexity and empirical risk in the generalization bounds as we vary the level of compression.

We summarize our contributions as follows:

- **Novel bounds for the unbounded negative log-likelihood objective:** we introduce novel bounds specifically tailored to account for the unbounded continuous bits-per-dimension loss, commonly used to evaluate LLMs for next-token prediction.
- **Subsampling bounds for practical bound evaluation:** To make the evaluation of the bounds practical on LLMs with massive datasets, we derive subsampling-based bounds that allow for efficient evaluation. In practice, the evaluation of the bound takes 45 minutes on a single GPU instead of 3 days on 8 GPUs in parallel for the OpenWebText dataset.
- **A simple yet powerful nonlinear subspace compression for LLMs:** as we show in Figure 1, using LoRA alone to compress the discrepancy between the random initialization and a learned model leads to vacuous bounds for the top-1 error. At the same time, linear subspace training alone does not unlock the full compression potential of LLMs compared to a nonlinear compression scheme. We show that a combination of these two approaches, while simple, yields a strong nonlinear compression of the model, which leads to the best generalization bounds for LLMs.
- **Non-vacuous generalization bounds for models with nearly a billion parameters:** our work not only introduces the first non-vacuous generalization bounds for LLMs, but it also extends these bounds to models with over 800 million parameters, demonstrating the scalability of our compression technique.
- **Improved understanding of generalization in LLMs:** as we increase the size of models, we find that they are able to find more compressed representations of the data and achieve better bounds, therefore disproving the claim that larger LLMs are simply better at regurgitating their training data.

The significance of these contributions lies in the ability to offer mathematical proof that large language models are, in fact, powerful knowledge compressors and are capable of generalization beyond their training samples, especially as their scale increases. To the best of our knowledge, our work is the first to show that generalization bounds improve with more parameters on models of practical sizes, in line with the empirical benefits of large models. We make our code available here.

2. Related Work

Generalization bounds. Neural networks have seen widespread adoption because of their strong performance on new unseen test samples, known as *generalization*. Early generalization theory literature bounded the difference in training and test error, called the *generalization gap*, using complexity measures like VC-dimension (Vapnik, 1991) and Rademacher complexity (Bartlett & Mendelson, 2002). These generalization bounds were vacuous for neural networks, which are often flexible enough to fit randomly labeled training data (Zhang et al., 2021). The flexibility of neural networks and its negative impact on these classical bounds calls into question why they generalize. Neural networks are so flexible that they have parameter vectors where they fit their training data and simultaneously assign incorrect labels to testing data, and they also have parameter vectors where they fit their training data and instead assign correct labels to the testing data. Why do such flexible models actually make correct test predictions in practice?

PAC-Bayes generalization theory bridges this gap by leveraging the fact that while neural networks are highly flexible and can fit random labels, they encode a preference for the correct ones (Catoni, 2007; Dziugaite & Roy, 2017). Unlike earlier generalization bounds which measured complexity merely as a function of the hypothesis class, PAC-Bayes generalization bounds reward models which have a strong prior that places its mass on parameter vectors that align with observed data. This formulation allows one to draw a parallel between generalization and compressibility (Zhou et al., 2019; Lotfi et al., 2022). By placing disproportionate prior mass on compressible parameter vectors, achieving a tight bound simply requires finding a family of models (posterior) that well fit the training data. Such compression bounds achieve the tightest guarantees to date on modern convolutional architectures and large-scale datasets, showcasing the strong inductive bias of neural networks and indicating that they can significantly compress their training sets (Lotfi et al., 2022). While PAC-Bayes has proven a very fruitful framework for devising such bounds, the insight on using a prior to bound the complexity of a given model does not require a posterior and can actually be incorporated into simpler finite hypothesis bounds.

Recent generalization theory literature has expanded analysis to several relevant models—autoregressive time-series models and simple n-gram language models (McDonald et al., 2011; Bharadwaj & Hasegawa-Johnson, 2014; Vankadara et al., 2022). In contrast, we construct bounds for autoregressive transformer-based language models.

Existing bounds for unbounded objectives. A number of works have explored techniques for generating generalization bounds on unbounded objective functions more generally, but these approaches are not practical for application to LLMs. A well established strategy relevant for e.g. linear regression with Gaussian errors is to bound the tails of the objective as subgaussian random variables, and then generalization bounds can be constructed for subgaussians more generally (Alquier et al., 2016; Germain et al., 2016). Other kinds of known tail behavior have also been exploited (Holland, 2019; Kuzborskij & Szepesvári, 2019). For the NLL of a language model, there is no clear analogous tail behavior, so we must take a different approach.

Haddouche et al. (2021) devise an approach for general unbounded objectives by constructing a hypothesis dependent bound on the objective, even if the objective is unbounded more generally. If the risk can be bounded $\sup_x R(h, x) \leq Q(h)$ for a function $Q(h)$, then PAC-Bayes bounds can be constructed using $Q(h)$ even if $\sup_h Q(h) = \infty$. However, even though $Q(h)$ is finite for LLMs as there are only a finite number of inputs, Q grows exponentially for NLL with the number of layers in the network and is closely related with the Lipschitz constant. For large models like LLMs, this value is far too large to be useful in constructing bounds.

Language models and compression. Large language models are parameterized with as many as billions of parameters and, as a result, have a significant memory footprint, which makes pretraining, finetuning, and even evaluation challenging without access to large-scale computing infrastructure. To reduce the memory footprint of large language models, a wide array of compression schemes has been proposed to enable evaluation, fine-tuning, and pre-training with limited computational resources. Low-Rank Adaptation (Hu et al., 2021, LoRA) freezes the pre-trained model weights and inserts trainable rank decomposition matrices into each attention layer of the transformer architecture used in large language models. Doing so allows for significantly reducing the number of trainable parameters for fine-tuning on downstream tasks. For example, LoRA can reduce the number of trainable parameters in GPT-3 175B fine-tuned with Adam by a factor of 10,000 and the GPU memory requirement by a factor of 3. Building on LoRA, Q-LoRA (Dettmers et al., 2023a) quantizes a pretrained model to 4-bits, adds a small set of learnable weights parameterized using LoRA, and then tunes these weights by backpropagating gradients through the quantized model. Other compression methods for large language models use distillation (Liu et al., 2023), sub-4-bit integer quantization (Kim et al., 2023; Park et al., 2022), sparse quantized representations that identify and isolate outlier weights (Dettmers et al., 2023b), weight quantization based on approximate second-order information (Frantal et al., 2022), or tensor-train decompositions (Xu et al., 2023).

Achieving a good generalization bound has distinct requirements from the existing compression literature. Unlike existing compression schemes for language models, which aim

to accelerate inference and training or to reduce the memory footprint, we focus on specifying the trained model parameters in only few bits, even if doing so decreases neither latency nor memory requirements.

3. Background

Subspace training. Lotfi et al. (2022) train a compressible model by parameterizing a carefully constructed low-dimensional random subspace. The weights $\theta \in \mathbb{R}^D$ are then defined as the sum of a random initialization θ_0 and a projection $P \in \mathbb{R}^{D\times d}$ from a lower-dimensional subspace $w \in \mathbb{R}^d$: $\theta = \theta_0 + Pw$. P is constructed as the Kronecker product of random Gaussian matrices $P = (Q_1 \otimes Q_2)/\sqrt{D}$ for $Q_1, Q_2 \sim \mathcal{N}(0,1)^{\sqrt{D}\times\sqrt{d}}$, normalized so that $P^\top P \approx I$. The weights w can then be optimized over by backpropagating through the transformation. With a learned quantization strategy—optimizing over quantized weights and the quantization levels—Lotfi et al. (2022) use arithmetic coding to encode the weights using the empirical probabilities over quantization bins.

Low Rank Adaptation (LoRA). Similarly inspired by evidence that overparametrized models have low intrinsic dimensionality (Li et al., 2018; Aghajanyan et al., 2020), Hu et al. (2021) propose LoRA as a parameter-efficient finetuning method. Given a pretrained weight matrix $W_{\text{pretrained}} \in \mathbb{R}^{a\times b}$, LoRA decomposes its total update ΔW accumulated throughout finetuning as a product of two trainable low-rank matrices $U \in \mathbb{R}^{a\times r}, V \in \mathbb{R}^{r\times b}$ for $r \ll \min(a,b)$ while freezing $W_{\text{pretrained}}$. Thus $W_{\text{finetuned}} = W_{\text{pretrained}} + \Delta W = W_{\text{pretrained}} + UV$. In this work, we use LoRA for pretraining instead. In particular, we take randomly initialized neural network weights $W_0 \in \mathbb{R}^{a\times b}$ and represent their update during pretraining as UV, yielding $W_{\text{pretrained}} = W_0 + \Delta W = W_0 + UV$. We decrease the dimensionality further by applying subspace projection to the LoRA matrices, which we describe in detail in Section 5.

4. Methodology

In constructing non-vacuous generalization bounds for LLMs, we expand and improve upon existing techniques in three ways: (1) we construct a simple and effective nonlinear parameterization which is more effective and scalable than purely linear subspaces; (2) we construct new bounds that can handle the continuous and unbounded nature of the negative log-likelihood; (3) we make these bounds more practical to compute with LLMs by deriving a new bound which holds even when the empirical risk is evaluated only on a small subsample of the full training dataset.

4.1. Finite Hypothesis Compression Based Generalization Bounds

Given a bounded risk $R(h,x) \in [a, a+\Delta]$ and a finite hypothesis space $h \in \mathcal{H}$ for which we have a prior $P(h)$, it is straightforward to derive a generalization bound relating the empirical risk $\hat{R}(h) = \frac{1}{m}\sum_{i=1}^m R(h, X_i)$ to the expected risk $R(h) = \mathbb{E}[\hat{R}(h)]$ so long as $\{X_i\}_{i=1}^m$ are sampled independently. With probability at least $1-\delta$, we have

$$R(h) \leq \hat{R}(h) + \Delta\sqrt{\frac{\log 1/P(h) + \log 1/\delta}{2m}}. \quad (1)$$

We provide an elementary proof in Appendix A.1.

If the prior likelihood $P(h)$ of the found model h can be increased (either by choosing a better prior, or by finding more likely hypotheses), then the generalization bound improves. Following Lotfi et al. (2022), we adopt the powerful but general Solomonoff prior $P(h) \leq 2^{-K(h|A)}$ (Solomonoff, 1964) where K is the prefix Kolmogorov complexity of h, with the model architecture A provided as input. The Kolmogorov complexity of hypothesis h is defined as the length of the shortest program that produces h for a fixed programming language P (Kolmogorov, 1963). While K is not computable, it is possible to compute the upper bound

$$\log 1/P(h) \leq K(h|A)\log 2 \leq C(h)\log 2 + 2\log C(h),$$

where $C(h)$ is the number of bits required to represent hypothesis h using some pre-specified coding. Therefore, if we can find hypotheses h that both have a low empirical risk *and* a small compressed size, then we can construct strong generalization bounds.

4.2. Enabling the Independence Assumption for Generalization Bounds on Text Data

Using Equation 1 requires that X_i in the sum $\hat{R}(h) = \frac{1}{m}\sum_{i=1}^m R(h, X_i)$ are drawn independently. Thus, we must be careful in the construction and interpretation of our bounds so that this constraint is satisfied. Instead of considering bounds at the level of tokens, which are correlated, we instead define X_i to be an entire document sampled from the data generating process from which the corpus was sampled. We define the risk on a given document as the negative log-likelihood of the entire document divided by its length, according to the autoregressive model.

It is also possible to choose X_i to be a *context chunk*, i.e., a sequence of length equal to the context length, as is commonly used in the training of models since a document may be larger than the maximum transformer context length. In such cases, the sequences are no longer independent samples from the data generating process. It is possible to construct valid bounds on these sequences which respect the independence assumption. However, in doing so we must shift the

interpretation of the bounds from being over the randomness in sampling from the data generating process to the randomness in sampling sequences that can be constructed from a fixed and finite dataset formed by concatenating the documents together.

We explore these alternate sequence-level bounds in Appendix B. However, we believe that the document-level bounds provide a more meaningful and significant statement about generalization.

Note (Satisfying the I.I.D assumption). *For document-level bounds, we sample the documents independently from the dataset. For sequence-level bounds, we divide the OpenWebText dataset into sequences of length equal to the context length and sample sequences independently so that a single sample from the dataset includes all of the tokens in the given sequence. Whether or not samples from a set are independent depends on how we sample from the set and not the contents of the set. We can draw independent samples from a set containing only similar elements, and the similarity of the elements does not affect the independence. Specifically, each element of our set is either an entire document or an entire sequence of tokens—not individual tokens—and we draw them independently.*

4.3. Accommodating the Unbounded NLL Objective Using Prediction Smoothing

The primary metric for pretraining of large language models, as for other autoregressive models, is the negative log-likelihood (NLL), or bits per dimension (BPD), of the generative model. The BPD loss is formally defined as the average over the negative log probabilities in logarithm base 2 as follows: $\mathrm{BPD}(h, X) = -\frac{1}{k}\sum_i^k \log_2 p_h(x_i|x_{<i})$. Unlike classification error which is a $\{0, 1\}$ valued random variable, the log-likelihood is an unbounded quantity that does not have an obvious sub-Gaussian, or other, well-understood tail behavior.

To overcome this challenge, we construct generalization bounds for BPD not of the original model but instead on a smoothed version of it that limits the worst case behavior. We define this smoothed model as a token-level mixture of the original LLM token predictions and a uniform distribution over the vocabulary of size V:

$$p_h(x_i|x_{<i}) = (1-\alpha)p_\theta(x_i|x_{<i}) + \alpha/V, \qquad (2)$$

where $p_\theta(x_i|x_{<i})$ is the base model of token probabilities, $\alpha \in (0, 1)$ is the mixing parameter, and $p_h(x_i|x_{<i})$ is the smoothed predictor.

The model on an entire document X composed of L tokens is defined autoregressively in terms of this mixture model $p_h(X) := \Pi_i^L p_h(x_i|x_{<i})$, and we find this to be a more effective way of constructing the bounds than constructing the mixture at the document level. In analogy to label smoothing where the labels of the training objective are mixed with the uniform distribution, we term this operation as prediction smoothing.

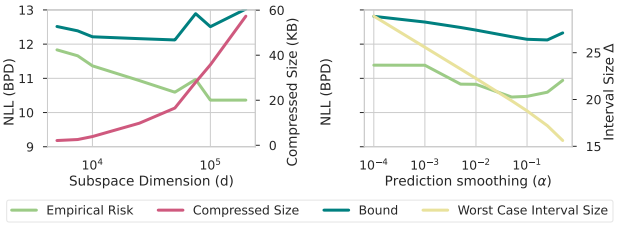

Figure 2. **Varying Parameters of the Compression Bounds. (Left):** A plot of the generalization bound as a function of the projection dimension d with LoRA. The subspace dimension gives us a way to explicitly trade off the degree of compression with the empirical risk, and we optimize d to produce the best bounds. **(Right):** A plot of the worst case range of BPD values Δ, empirical risk, and the resulting generalization bounds as a function of the prediction smoothing parameter α. For each model, a different alpha can be chosen after the models have already been trained.

As we show in Appendix A.2, the NLL of the prediction smoothed model on a document $\mathrm{BPD}(h, X) := -\log_2 p_h(X)/L$ can be bounded as follows:

$$\log_2(V/\alpha) - \Delta \leq \mathrm{BPD}(h, X) \leq \log_2(V/\alpha),$$

for $\Delta = \log_2\big(1+(1-\alpha)V/\alpha\big)$. With prediction smoothing, the risk $R(h, X) = \mathrm{BPD}(h, X)$ on a given document is bounded in an interval of size Δ, and therefore we can use Equation (1) to generate bounds for negative log-likelihood of this model. We refer to Δ as the worst-case interval size.

We explore the trade-off over different values of α in Figure 2 (right). As α gets larger, the interval size Δ representing the worst-case behavior goes down, whereas the empirical risk goes up, leading to a sweet spot in the middle. By defining the hypothesis $h = (\theta, d, r, \alpha)$ to include the model parameters, LoRA space hyperparameters d, r, and the mixture weight α, we can view α as merely one additional model parameter accounted in $\log 1/P(h)$. By doing so, we are free to optimize over α in the computation of the bound, and we can do so without retraining the model.

4.4. Using Subsampling in Bound Computation

The empirical risk requires evaluating the model on the full training dataset of m data points: $\hat{R}(h) = \frac{1}{m}\sum_{i=1} \hat{R}_i(h)$. As large language models are typically trained for only 1 epoch or less, doing so is prohibitively expensive. Instead, we propose to modify our generalization bounds to account for evaluating only a subsample of size $n \ll m$ of the training dataset when computing the empirical risk.

Denoting $\hat{\hat{R}}(h) = \sum_{i=1}^n \hat{R}_{\sigma(i)}(h)$ where $\sigma(i)$ is a random sample (with replacement) from $1, \ldots, m$. In Appendix A.3 we derive a new bound both over the randomness in $\sigma(i)$ and

Table 1. **Non-vacuous generalization bounds for GPT-2 compressed models.** Our best document-level generalization bounds achieved for the GPT-2 architecture for BPD and Top-k token prediction error, all of which are non-vacuous.

Metric	SubLoRA	LoRA Only	Subspace Only	Original Model	Random Guess
Top-1 Error Bound (%)	**96.41**	100	96.52	100	99.99
Top-10 Error Bound (%)	**77.90**	84.37	79.36	100	99.98
Top-100 Error Bound (%)	**58.34**	67.26	75.95	100	99.80
Bits per Dimension Bound	**12.12**	13.09	14.59	70.76	15.62

the randomness in X which holds with probability $\geq 1-\delta$:

$$R(h) \leq \hat{\hat{R}}(h) + \Delta\sqrt{\frac{\log\frac{1}{P(h)} + \log\frac{1}{s\delta}}{2m}} + \Delta\sqrt{\frac{\log\frac{1}{(1-s)\delta}}{2n}}, \tag{3}$$

where $s = n/(n+m)$. Using this subsampling bound, we can accelerate bound computation. For dataset sizes in the 10's of millions, we can get away with evaluating only $10,000$ data points after the model has been trained, with a negligible penalty in the bounds. In fact, we need not even train on the entirety of the training data in order to produce valid bounds as long we sample uniformly.

5. SubLoRA: A Simple and Efficient Nonlinear Parameterization of the Hypothesis Space

To find compressible solutions h that simultaneously are expressive enough to achieve low training error, we search over a carefully designed manifold of possible parameters that live within the parameter space.

In contrast to Lotfi et al. (2022), we consider a nonlinear parameterization of the model weights $\theta = f(\theta_0, w)$ given by the composition of LoRA (Hu et al., 2021) (a nonlinear parameterization) and the subspace compression matrices. Given a vector of model parameters θ, we break down its constituent components into the different weight matrices W_i and associated biases b_i: $\text{unflatten}(\theta) = \{(W_i, b_i)\}_{i\in I}$. We define a nonlinear parameterization of the hypothesis space as

$$\theta = \theta_0 + \text{LoRA}(Pw), \tag{4}$$

where LoRA is defined by the implementation of the low-rank products for the weight matrices, leaving the biases unchanged. As Pw and θ are the flattened parameter vectors, $\text{LoRA}(\cdot)$ is defined as the operation that unflattens the vector, applies the low-rank product, and then flattens the result. Here, θ_0 is merely a random initialization of the model parameters, and $P \in \mathbb{R}^{D\times d}$ is a Kronecker product projector $P = Q_1 \otimes Q_2$ for Q_1, Q_2 constructed by orthogonalizing Gaussian random matrices by QR factorization: $P_1, P_2 \sim \mathcal{N}(0, 1/\sqrt{D})^{\sqrt{D}\times\sqrt{d}}$ with $Q_1 R_1 = P_1$ and similarly for Q_2. We apply LoRA only over the self-attention layer and the last linear layer weight matrices, meaning that other model parameters do not differ from their initialized values. In order to compress the model, we need only to represent the vector w since θ_0 and P are chosen ahead of time and specified in the architecture via random initialization.

Note (Selecting the LoRA layers). *While LoRA was developed for finetuning LLMs, we find that even when pretraining using LoRA, we can achieve non-trivial performance. Our initial exploration of LoRA for pretraining involved applying LoRA not only to attention layers but to all other linear layers as well. We found that for pretraining, it is more efficient to use LoRA for both the attention layers and the last linear layer, while including other layers provides insignificant returns.*

In Figure 1 (left), we show the Pareto frontier of empirical risk and the complexity penalty in the relevant generalization bound with LoRA, subspace training, and SubLoRA. Rather than being competing methods for compression, LoRA and subspace training are complementary and exploit different structures in the parameter space to provide a family of models in the original hypothesis space that are both expressive and compressible. SubLoRA achieves a strict improvement over LoRA and subspace training, often being the deciding factor whether the bounds are vacuous or non-vacuous. In Figure 2 (left), we explore how the compressed size of the model and the empirical risk vary as a function of the subspace dimension d.

6. Non-Vacuous Generalization Bounds for Large Language Models

We outline the pretraining and bound computation pipeline and then present our empirical results.

6.1. End-to-end Pipeline

Assembling the components described in Section 4, we train variants of a GPT-style architecture through the nonlinear compressed parameterization in Equation (4). We use several values for the subspace dimension d and two values for the rank of the LoRA matrices r. Nearing the end of training, we train for additional steps using quantization-aware

Table 2. **Validation performance vs. bounds.** Validation performance of the models achieving the best bits per dimension bound for each setting. For models trained using different compression techniques, SubLoRA performs best in terms of validation loss. Although the original model achieves the best validation BPD and NLL, it leads to vacuous bounds given its large number of parameters.

Metric	SubLoRA	LoRA Only	Subspace Only	Original Model	Random Guess
Bits per Dimension Bound	12.12	13.09	14.59	70.76	15.62
Validation BPD	10.53	11.21	13.85	4.35	–
Validation NLL	7.29	7.77	9.60	3.01	–

training with a small number of quantization levels (with additional details listed in Appendix E). We express w in this quantization and encode it using arithmetic coding to determine the compressed size of the model. Added to the size of the model are the bits needed to encode the choice of d, r, α, the learning rate, and the quantization levels.

We evaluate the empirical log probabilities and token predictions for each token in the document on a small subset of the training data $n = 10,000$ documents. We use this sub-sampling size to evaluate all the bounds reported in the paper. With these predictions, we can compute the generalization bound in Equation (3) as a function of α, and we optimize over this parameter for each model. Finally, we can tune the extent of compression through the different choices of d and choose the subspace dimension that produces the best bound. We provide additional details about bound computation in Appendix C.

6.2. Non-Vacuous Bounds for GPT-2 Small

We consider the GPT-2 small architecture with 124M parameters and compute our next token prediction document-level bounds by pretraining these models on the OpenWebText dataset using SubLoRA. We report the results in Table 1. We consider the token level error averaged over a document as the empirical risk. For instance, the Top-1 Error Bound refers to the upper bound on the expected Top-1 error per token averaged over the document $R(h, X_k) = \frac{1}{L}\sum_{i=1}^{L} \mathbf{1}[\operatorname{argmax} p(x_i|x_{<i} = x^k_{<i}) = x^k_i]$, where the upper index k denotes the document index and the lower index denotes the position within the document. The argmax operator operates over the discrete conditional probability distribution across the vocabulary. Thus, $R(h, X_k)$ represents the proportion of tokens accurately predicted within a given document. Random guess performance is $\log_2 V$ for BPD and $1 - k/V$ for Top-k Error.

The best bounds are indeed obtained using our simple compression technique, which combines the strengths of both low-rank adaptation and subspace training. When we solely apply quantization and arithmetic coding without implementing LoRA or linear subspace compression during the training phase, we obtain vacuous bounds.

6.3. Performance

In Table 2, we report the validation performance of the models that achieve the best bounds in Table 1. In particular, we report the bits per dimension (BPD) and negative log-likelihood (NLL) losses alongside the generalization bounds. For compressed models, we see that the bounds have the same trend as the validation performance, as all these models contain a smaller number of parameters compared to the initial parameter space. Although the original model with the full set of parameters achieves the best validation performance, it leads to vacuous bounds given its large number of parameters.

It is important to note that we report the validation BPD loss of the model achieving the best bounds, and not the best validation BPD loss achieved by the different compression schemes, which can be lower depending on the subspace parameters. Figure 2 (left) reflects this trade-off between the compressed size of the model and the empirical risk.

6.4. Extending Our Bounds to Larger Models

We use SubLoRA to obtain generalization bounds for much larger variants of GPT-2 of sizes 354M (GPT-2 medium), 458M, 773M (GPT-2 large), and 849M parameters. Table 3 shows that our simple compression approach yields non-vacuous bounds for models with nearly a billion parameters. Moreover, we see that the smallest model, where we previously performed experiments and tuned our hyperparameters, actually achieves the worst bound on bits per dimension as we scale the models up. In conclusion, our approach extends naturally to much larger language models and proves that it is possible to achieve tighter bounds as we increase the size of the model.

Note (Limitations). *Note that due to computational constraints, we pre-train the larger GPT-2 variants with SubLoRA only for a limited number of hyperparameter settings in contrast to the* 124*M model for which we did a thorough hyperparameter sweep. It is likely that the tightest empirically achievable bounds are much stronger for the new large models than what we report in Table 3.*

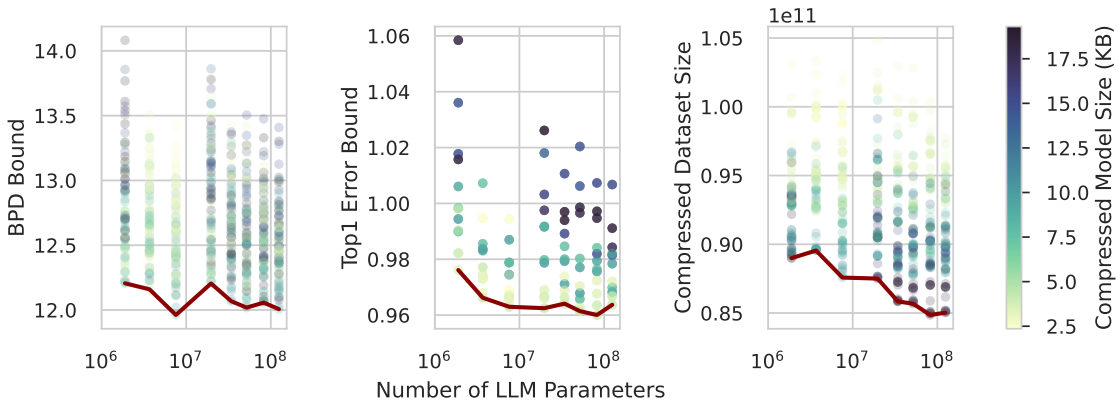

Figure 3. **Larger models achieve stronger generalization bounds.** As we scale up the size of the model via the model parameters (holding the training set fixed), we find that our generalization bounds get *better* rather than worse. Dots show models trained with differing degrees of compression, indicated by their color. On the right we show the number of bits required to express the training dataset using the model and including the model weights in the compression. Classification error bounds consistently favor smaller models, while data compression favors much larger models, and BPD bounds are in between.

Table 3. **Non-vacuous generalization bounds for models with up to 849M parameters.** Non-vacuous bounds achieved for GPT-2 architectures with different sizes, ranging from 124 to 849 million parameters. We report below the bounds on the bits-per-dimension (BPD), Top-1 Error, and Top-100 Error. All of the BPD bounds are non-vacuous and tighter than the GPT-2 small bounds.

Model Size	BPD	Top-1 Error	Top-100 Error
124M	12.12	96.41	58.34
354M	11.96	95.99	58.4
458M	11.95	96.69	58.49
773M	12.10	96.17	59.25
849M	12.01	96.51	58.89

7. Understanding the Generalization of LLMs

As language models grow in size, it is clear that they gain an increasing capacity to fit their training data. On the one hand, this increasing capacity might mean that, as LLMs become capable of learning increasingly complex functions, they become increasingly likely to merely memorize their training samples and not perform any meaningful generalization beyond their training corpora. On the other hand, large language models have proven to be surprisingly capable of generalizing, often extending to tasks that seem quite different from the training objective.

We investigate the tension between these two narratives along several fronts: We assess how generalization bounds change with the size of the model, whether language models can form a compression of the training data even when accounting for their large size, and how structure in the training data affects the generalization of the learned model. In Appendix D, we use our bounds to quantify of the benefits of pre-training in LLMs.

7.1. Larger Models Are More Compressible and Generalize Better

Empirically, it has been found that LLMs generalize better as the number of parameters is increased, with a fixed size of dataset (Kaplan et al., 2020; Brown et al., 2020), and this fact is of great importance leading to the creation of ever larger and more powerful models. From a generalization theory perspective, this trend is counterintuitive because of the growing hypothesis class, and a naive analysis would suggest that larger models should generalize worse. To date, we are not aware of any convincing demonstration that generalization bounds improve with more parameters on models of practical sizes.

We evaluate our bounds on a collection of LLMs with different numbers of parameters, choosing the appropriate scaling for the width, depth, number of attention heads, etc. Surprisingly, we find that our generalization bounds in fact *improve* with model size, even as the training dataset is held fixed. With our SubLoRA compression, larger models find even simpler representations of the data given a fixed training set. These results are shown in Figure 3. While some explanations for why larger models should generalize better have been put forward in the literature (Nakkiran et al., 2021; Gunasekar et al., 2017), the mechanism by which larger models become more compressible is not clear, and we believe this result is noteworthy and requires further investigation.

In addition to constructing generalization bounds, we can use our compressed models to form a compression of the training dataset itself. In Figure 3, we count the number of bits needed to encode the model $C(h)$ and the number of bits to encode the data using the model $C(\{X\}_{i=1}^m|h)$, which is the negative log-likelihood of the entire dataset according to the model. Adding these two up, we have a

compression of the training dataset using the model, and one which is closely related to our generalization bounds.

7.2. How Does Generalization of Large Language Models Depend on Structure in Text?

Neural networks that fit a training dataset of random noise will not be able to generalize, and the ability of over-parametrized networks to fit noise implies that uniform convergence is impossible across the general hypothesis class (Nagarajan & Kolter, 2019). This is a clear demonstration that the structure of the dataset influences generalization. However, the impact of more subtle structures is less understood theoretically. Here, we use our bounds to investigate how the temporal order structure relates to generalization.

We train models that explicitly break the temporal structure of the text data by applying random permutations to each sequence during training. Consequently, the model can only make use of the input information as if it were a bag of words. We find that this broken order structure indeed leads to less favorable generalization bounds. Figure 4 shows the best error bounds when the original and perturbed data are used to train the model and evaluate the bounds for the bits per dimension, top-1 error, and top-100 error losses. While the top-1 error bound becomes vacuous as we break the text structure, the top-100 error and bits per dimensions bounds remain non-vacuous. This might be due to the fact that as we perturb the sequence, predicting the next token accurately becomes an extremely difficult task for LLMs, while predicting a token that fits generally into the context, without necessarily being the correct token, is an easier task.

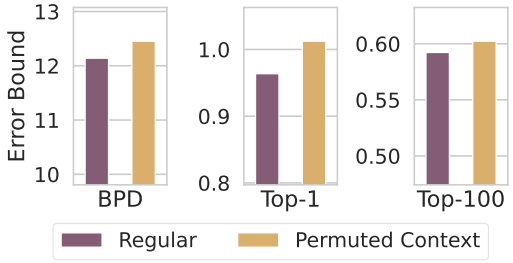

Figure 4. **Breaking text structure with permutations.** We compute bounds for LLMs that were trained with the order of the tokens shuffled within each sequence.

8. Discussion

In this work, we demonstrated that large language models can themselves form highly compressed representations of distributions over text. Using these highly compressed LLMs, we compute the first non-vacuous generalization bounds for LLM pretraining. Our findings suggest that the development of tighter compression bounds presents a fruitful avenue for understanding how and why language models generalize.

We discuss below the limitations of our work and their implications for future work.

Non I.I.D. token level bounds. In our work, we split up the training data into I.I.D. documents that form the basis of our bounds. However, the loss for each of these documents also decomposes as a sum over non I.I.D. tokens, and it is likely that this additional structure could also be exploited in the bound construction to significantly increase the effective number of training samples.

Efficient bound computation on pretrained models. Our procedure for computing generalization bounds requires training LLMs from scratch through our SubLoRA parametrization. It may be possible to devise a fast method of computing bounds on a model that has already been trained, but still constraining its generalization error.

Nonlinear parameterizations. Unlike previous state-of-the-art bounds from Lotfi et al. (2022), we employ a nonlinear parameterization via LoRA, significantly improving the bounds. This observation opens up an avenue for rich nonlinear parameterizations that simultaneously reduce the number of parameters while also including diverse functions which are likely to fit the training data.

Bounds for models that generate high quality text. In Table 6 and Table 7, we show samples of generated text using both a GPT-2 style model pretrained in the standard fashion and a GPT-2 style model pretrained using SubLoRA. While the vanilla GPT-2 style model produces reasonable sentences, the SubLoRA pretrained model often outputs ungrammatical text.

Alternative approaches to learning with LLMs. Modern language models make possible new inference techniques such as in-context learning and prompt-tuning. These modes are already seeing widespread deployment and warrant analogous theories of generalization.

Generalization beyond the training distribution. Recent work showed that language models prefer low-complexity numerical sequences on which they were not trained, even at random initialization (Goldblum et al., 2023), and generalization theory may be useful for explaining why LLMs can generalize far outside of their training distribution, and even outside of the text modality, for example to tabular data (Hegselmann et al., 2023) or images (Delétang et al., 2023).

We hope that future work will address these limitations and that our approach can help lay the groundwork for computing LLM generalization bounds for even larger models.

Acknowledgements

We acknowledge anonymous reviewers for helpful feedback. This work is supported by NSF CAREER IIS-2145492, NSF CDS&E-MSS 2134216, NSF HDR-2118310, BigHat Biosciences, Capital One, and an Amazon Research Award.

Impact Statement

The goal of this work is to advance our understanding of large language models and improve their trustworthiness. However, we note that there are many facets of large language models—such as biases—that may not be captured by generalization bounds.

References

Aghajanyan, A., Zettlemoyer, L., and Gupta, S. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*, 2020.

Alquier, P., Ridgway, J., and Chopin, N. On the properties of variational approximations of gibbs posteriors. *The Journal of Machine Learning Research*, 17(1):8374–8414, 2016.

Bartlett, P. L. and Mendelson, S. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

Bharadwaj, S. and Hasegawa-Johnson, M. A PAC-Bayesian approach to minimum perplexity language modeling. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pp. 130–140, Dublin, Ireland, 2014.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T. B., Song, D., Erlingsson, U., Oprea, A., and Raffel, C. Extracting training data from large language models. *arXiv preprint arXiv:2012.07805*, 2020.

Carlini, N., Ippolito, D., Jagielski, M., Lee, K., Tramer, F., and Zhang, C. Quantifying memorization across neural language models. *Proceedings of the 37th International Conference on Learning Representations (ICLR 2023)*, 2023.

Catoni, O. Pac-bayesian supervised classification: the thermodynamics of statistical learning. *arXiv preprint arXiv:0712.0248*, 2007.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. Palm: Scaling language modeling with pathways, 2022.

Delétang, G., Ruoss, A., Duquenne, P.-A., Catt, E., Genewein, T., Mattern, C., Grau-Moya, J., Wenliang, L. K., Aitchison, M., Orseau, L., et al. Language modeling is compression. *arXiv preprint arXiv:2309.10668*, 2023.

Dettmers, T., Shmitchell, S., Roberts, A., Lee, K., Brown, T. B., Song, D., and Raffel, C. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023a.

Dettmers, T., Shmitchell, S., Roberts, A., Lee, K., Brown, T. B., Song, D., and Raffel, C. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2308.07234*, 2023b.

Dziugaite, G. K. and Roy, D. M. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

Frantal, Z., Gruslys, A., and Kiela, D. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Germain, P., Bach, F., Lacoste, A., and Lacoste-Julien, S. Pac-bayesian theory meets bayesian inference. *Advances in Neural Information Processing Systems*, 29, 2016.

Goldblum, M., Finzi, M., Rowan, K., and Wilson, A. G. The no free lunch theorem, kolmogorov complexity, and the role of inductive biases in machine learning. *arXiv preprint arXiv:2304.05366*, 2023.

Gunasekar, S., Woodworth, B. E., Bhojanapalli, S., Neyshabur, B., and Srebro, N. Implicit regularization in matrix factorization. *Advances in neural information processing systems*, 30, 2017.

Haddouche, M., Guedj, B., Rivasplata, O., and Shawe-Taylor, J. Pac-bayes unleashed: Generalisation bounds with unbounded losses. *Entropy*, 23(10):1330, 2021.

Hegselmann, S., Buendia, A., Lang, H., Agrawal, M., Jiang, X., and Sontag, D. Tabllm: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics*, pp. 5549–5581. PMLR, 2023.

Hoeffding, W. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pp. 409–426, 1994.

Holland, M. Pac-bayes under potentially heavy tails. *Advances in Neural Information Processing Systems*, 32, 2019.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kim, J., Lee, J. H., Kim, S., Park, J., Yoo, K. M., Kwon, S. J., and Lee, D. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. *arXiv preprint arXiv:2305.14152*, 2023.

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

Kuzborskij, I. and Szepesvári, C. Efron-stein pac-bayesian inequalities. *arXiv preprint arXiv:1909.01931*, 2019.

Langdon, G. G. An introduction to arithmetic coding. *IBM Journal of Research and Development*, 28(2):135–149, 1984.

Li, C., Farkhoor, H., Liu, R., and Yosinski, J. Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838*, 2018.

Liu, Y., Xu, Q., Xu, W., and Zhu, J. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*, 2023.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Lotfi, S., Finzi, M., Kapoor, S., Potapczynski, A., Goldblum, M., and Wilson, A. G. Pac-bayes compression bounds so tight that they can explain generalization. *Advances in Neural Information Processing Systems*, 35:31459–31473, 2022.

McDonald, D. J., Shalizi, C. R., and Schervish, M. Generalization error bounds for stationary autoregressive models. *arXiv preprint arXiv:1103.0942*, 2011.

Nagarajan, V. and Kolter, J. Z. Uniform convergence may be unable to explain generalization in deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

Park, G., Kim, J., Kim, J., Choi, E., Kim, S., Kim, S., Lee, M., Shin, H., and Lee, J. Lut-gemm: Quantized matrix multiplication based on luts for efficient inference in large-scale generative language model. *arXiv preprint arXiv:2206.09557*, 2022.

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

Vankadara, L. C., Faller, P. M., Hardt, M., Minorics, L., Ghoshdastidar, D., and Janzing, D. Causal forecasting: generalization bounds for autoregressive models. In *Uncertainty in Artificial Intelligence*, pp. 2002–2012. PMLR, 2022.

Vapnik, V. Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 4, 1991.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2019.

Xu, Q., Xu, W., and Zhu, J. Tensorgpt: Efficient compression of the embedding layer in llms based on the tensor-train decomposition. *arXiv preprint arXiv:2307.00526*, 2023.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Zhou, W., Veitch, V., Austern, M., Adams, R. P., and Orbanz, P. Non-vacuous generalization bounds at the imagenet scale: a pac-bayesian compression approach. In *International Conference on Learning Representations*, 2019.

A. Derivations and Generalization Bounds

A.1. Finite Hypothesis Bound

Theorem A.1. *Consider a bounded risk $R(h, x_i) \in [a, a + \Delta]$ and a finite hypothesis space $h \in \mathcal{H}$ for which we have a prior $P(h)$ that does not depend on $\{x_i\}$. Let the empirical risk $\hat{R}(h) = \frac{1}{m}\sum_{i=1}^{m} R(h, x_i)$ be a sum over independent random variables $R(h, x_i)$ for a fixed hypothesis h. Let $R(h) = \mathbb{E}[\hat{R}(h)]$ be the expected risk.*

With probability at least $1 - \delta$:

$$R(h) \leq \hat{R}(h) + \Delta\sqrt{\frac{\log 1/P(h) + \log 1/\delta}{2m}}, \tag{5}$$

Proof. As $m\hat{R}(h)$ is the sum of independent and bounded random variables, we can apply Hoeffding's inequality (Hoeffding, 1994) for a given choice of h . For any $t > 0$

$$P(R(h) \geq \hat{R}(h) + t) = P(mR(h) \geq m\hat{R}(h) + mt)$$
$$P(R(h) \geq \hat{R}(h) + t) \leq \exp\left(-2mt^2/\Delta^2\right).$$

We will choose $t(h)$ differently for each hypothesis h according to

$$\exp\left(-2mt(h)^2/\Delta^2\right) = P(h)\delta.$$

Solving for $t(h)$, we have

$$t(h) = \Delta\sqrt{\frac{\log 1/P(h) + \log 1/\delta}{2m}} \tag{6}$$

This bound holds for a fixed hypothesis h. However h was constructed using the training data, so for $h^*(\{x\})$, the random variable,

$$\hat{R}(h^*) = \frac{1}{m}\sum_{i=1}^{m} R(h^*(\{x\}), x_i),$$

cannot be decomposed as a sum of independent random variables. Since $h^* \in \mathcal{H}$, if we can bound the probability that $R(h) \geq \hat{R}(h) + t(h)$ for *any* h, then the bound also holds for h^*.

Applying a union over the events $\bigcup_{h\in\mathcal{H}}\left[R(h) \geq \hat{R}(h) + t(h)\right]$, we have

$$\begin{aligned} P(R(h^*) \geq \hat{R}(h^*) + t(h^*)) &\leq P\Big(\bigcup_{h\in\mathcal{H}}\left[R(h) \geq \hat{R}(h) + t(h)\right]\Big) \\ &\leq \sum_{h\in\mathcal{H}} P\big(R(h) \geq \hat{R}(h) + t(h)\big) \\ &\leq \sum_{h\in\mathcal{H}} P(h)\delta = \delta. \end{aligned}$$

Therefore we conclude that for any h (dependent on x or not), with probability at least $1 - \delta$,

$$R(h) \leq \hat{R}(h) + \Delta\sqrt{\frac{\log 1/P(h) + \log 1/\delta}{2m}}.$$

□

Note (Satisfying the finite hypothesis space assumption). *We consider neural networks as programs which run on a computer, and therefore they must have a finite size. For instance, without using any compression, the weights of a neural network are typically represented using floating point numbers, and therefore the weights of real models can only take on a finite number of values determined by the precision. If we consider a compression of the hypothesis, then we can express it in a smaller number of bits, which is again finite and not real valued. From this perspective, all neural networks that we train and deploy in practice belong to a finite hypothesis space given a fixed architecture.*

A.2. Bounding Log-Likelihood

Theorem A.2. *Given $\alpha \in (0,1)$, an α prediction smoothed autoregressive language model h over a token vocabulary of size V for a given sequence X will have a* $\mathrm{BPD}(h, X)$ *that lies in the interval*

$$\mathrm{BPD}(h, X) \in \big(\log_2(V/\alpha) - \log_2\big(1 + (1-\alpha)V/\alpha\big), \log_2(V/\alpha)\big), \tag{7}$$

and the size of the interval is $\Delta = \log_2\big(1 + (1-\alpha)V/\alpha\big)$.

Proof. The BPD decomposes as the average over the negative log probabilities,

$$\mathrm{BPD}(h, X) = -\frac{1}{k}\sum_{i}^{k} \log_2 p_h(x_i|x_{<i}).$$

Since $p_\theta(x_i|x_{<i}) \in (0,1)$, we can conclude that

$$\begin{aligned} -\log_2 p_h(x_i|x_{<i}) &= -\log_2\big((1-\alpha)p_\theta(x_i|x_{<i}) + \alpha/V\big) \\ -\log_2 p_h(x_i|x_{<i}) &< \log_2(V/\alpha) \end{aligned}$$

and

$$\begin{aligned} -\log_2 p_h(x_i|x_{<i}) &= -\log_2\big((1-\alpha)p_\theta(x_i|x_{<i}) + \alpha/V\big) > -\log_2\big((1-\alpha) + \alpha/V\big) \\ -\log_2 p_h(x_i|x_{<i}) &> -\log_2\left(\tfrac{\alpha}{V}\big(1 + (1-\alpha)V/\alpha\big)\right) \\ -\log_2 p_h(x_i|x_{<i}) &> \log_2(V/\alpha) - \log_2\big(1 + (1-\alpha)V/\alpha\big). \end{aligned}$$

Since each element $-\log_2 p_h(x_i|x_{<i})$ of the average is in the interval $\big(\log_2(V/\alpha) - \Delta, \log_2(V/\alpha)\big)$, so is $\mathrm{BPD}(h, X)$.

□

A.3. Subsample Bounds

Denoting $\hat{\hat{R}}(h) = \frac{1}{n}\sum_{i=1}^{n} \hat{R}_{\sigma(i)}(h)$ where $\sigma(i)$ is a random sample (with or without replacement) from $1, \ldots, m$, we can construct a simple Hoeffding bound over the randomness in $\sigma(i)$, considering X fixed. Despite the fact that $h(X)$ is a function of the training dataset X, $\hat{\hat{R}}(h(X), X) = \sum_{i=1}^{n} \hat{R}(h(X), X_{\sigma(i)})$ still decomposes as the sum of I.I.D. random variables (or I.I.D. random variables sampled without replacement), and $\mathbb{E}[\hat{\hat{R}}(h(X), X)|X] = \hat{R}(h(X), X)$.

Applying the Hoeffding bound (Hoeffding, 1994), with probabiliiy $1 - \delta_2$: $\hat{R} \leq \hat{\hat{R}}(h) + \sqrt{\frac{\log 1/\delta_2}{2n}}$. Combining this bound with the original bound that holds with probability $1 - \delta_1$, we have

$$R(h) \leq \hat{\hat{R}}(h) + \Delta\sqrt{\frac{\log 1/P(h) + \log 1/\delta_1}{2m}} + \Delta\sqrt{\frac{\log 1/\delta_2}{2n}}.$$

Combining the two failure probabilities into one: $\delta = \delta_1 + \delta_2$, we can choose δ_1 and δ_2 so that optimize the bound keeping their sum fixed. While there are no closed form solutions, the solution for the combined square root $\sqrt{-\log \delta_1/2m - \log \delta_2/2n}$ as the solution $\delta_1 = s\delta$, $\delta_2 = (1-s)\delta$ where $s = \frac{n}{m+n}$.

Plugging these values into the bound, we have

$$R(h) \leq \hat{\hat{R}}(h) + \Delta\sqrt{\frac{\log \frac{1}{P(h)} + \log \frac{1}{s\delta}}{2m}} + \Delta\sqrt{\frac{\log \frac{1}{(1-s)\delta}}{2n}}. \tag{8}$$

Algorithm 1 Compute Finite Hypothesis Bound.

Inputs: Neural network f_θ, Training dataset of m documents $\{X_k\}_{k=1}^m$, subsampled set of n documents $\{X_{\sigma(i)}\}_{i=1}^n$, quantization levels C, Intrinsic dimension d, LoRA rank r, prediction smoothing probability α, Confidence $1-\delta$.
function COMPUTE_BOUND($f_\theta, L, d, r, \alpha, \{X_k\}_{k=1}^m, \{X_{\sigma(i)}\}_{i=1}^n, \delta,$)
  $w \leftarrow$ TRAIN_SUBLORA(f_θ, d, r, $\{X_k\}_{k=1}^m$) ▷ (Section 5)
  $\hat{w} \leftarrow$ TRAIN_QUANTIZE(w, C, $\{X_k\}_{k=1}^m$)
  Compute quantized train error $\hat{\hat{R}}(\hat{w})$ with prediction smoothing probability α and subsampled dataset $\{X_{\sigma(i)}\}_{i=1}^n$.
  $\log 1/P(h) \leftarrow$ GET_COMPRESSED_SIZE($\hat{w}$) ▷ (Section 4.1)
  return GET_FINITE_HYPOTHESIS_BOUND($\hat{\hat{R}}(\hat{w})$, $\log 1/P(h)$, m, n) ▷ (Section 4.4)
end function
function TRAIN_QUANTIZE(w, C, $\{X_k\}_{k=1}^m$) ▷ (Appendix E)
  Initialize $c \leftarrow$ GET_CLUSTERS(w, C)
  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_COMPRESSED_SIZE($\hat{w}$)
  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 ▷ (Appendix E)
  return message_size $+ 2 \times \log$(message_size)
end function

B. Sequence-level Bounds

We construct sequence level bounds on chunks of size 1024 (equal to the context length) which are sampled from the non-overlapping chunkings of the OpenWebText dataset.

We report our sequence-level bounds in Table 4. Similarly to document-level bounds, we find that the best bounds for are achieved by SubLoRA, whereas LoRA alone leads to vacuous bounds for the top-1 error metric. We find that despite the differing interpretation, the bounds are very similar in values to the document level bounds that we report in Table 1, with differences arising from the empirical risk evaluation and having a slightly larger m due to the presence of some long documents.

Table 4. Our best sequence-level generalization bounds achieved for the GPT-2 architecture for BPD and Top-k token prediction error, all of which are non-vacuous.

Metric	SubLoRA	LoRA Only	Subspace Only	Original Model	Random Guess
Top-1 Error (%)	**96.17**	100	97.40	100	99.99
Top-10 Error (%)	**78.18**	85.85	80.15	100	99.98
Top-100 Error (%)	**58.72**	65.19	76.11	100	99.80
Bits per Dimension	**12.09**	12.90	14.68	65.37	15.62

C. Bound Computation

We provide pseudo-code for bound computation in Algorithm 1.

We first train a compressed neural network in the SubLoRA subspace, determined by the LoRA rank r and the intrinsic dimensionality d of the linear subspace projection. To further compress the model, we use aggressive quantization with quantization-aware training as proposed by Lotfi et al. (2022) in order to map the weights of SubLoRA into C quantization levels, and compute the number of bits required to represent our model using arithmetic coding, a form of variable length

encoding that leverages the fact that certain quantization levels are more frequent than others (Langdon, 1984).

Finally, we evaluate the empirical risk using a subset of the training data and a prediction smoothing probability α, where the subsampling size $n = 10,000$ documents. Having both the empirical risk and the compressed size of the model, we compute our bound as described in Equation (3). In practice, we compute the bounds for different values of r, d, L and α while paying additional bits for each of these settings, and report the lowest bound in Table 1 and Table 3.

D. The Importance of Pretraining LLMs

To demonstrate the benefits of pretraining for LLM, we fine-tune both a randomly initialized and a pretrained GPT-2 model with SubLoRA on the QQP and CoLA binary classification datasets from GLUE (Wang et al., 2019). For both models, we fine-tune using SubLoRA with rank 8 and intrinsic dimension equal to 30000 for 5 epochs with a learning rate of 2×10^{-5}. We quantize the checkpoints of finetuned models and obtain the following non-vacuous classification accuracy bounds in Table 5.

Table 5. Pretrained GPT-2 models finetuned with SubLoRA leads to significantly better bounds compared to randomly initialized GPT-2 models finetuned with SubLoRA.

Dataset	Error Bound for pretrained LLM (%) + SubLoRA Finetuning	Error Bound for randomly initialized LLM (%) + SubLoRA Finetuning	Random Guess (%)
QQP (%)	**35.27**	71.72	50
CoLA (%)	**38.89**	53.42	50

As Table 5 shows, pretrained LLMs lead to tighter, non-vacuous bounds compared to randomly initialized LLM when finetuned on the same set of downstream tasks. Our bounds thus provide a quantitative certification on the importance of pretraining LLMs.

E. Experimental Details

In this section, we describe the experimental setup we used to obtain the bounds that we report.

We follow the pretraining setup described in nanoGPT[1] as a backbone for our experiments The model architecture in use is a 124 million parameter GPT-2-style model with 12 layers, 12 heads in multi-headed attention, and an embedding dimension of 768, and we pretrain this model on the training split of the OpenWebText dataset[2] using SubLoRA, LoRA, Subspace training. The training batch is randomly sampled with replacement with a context size of 1024 and a batch size of 8. For optimization, we use a PyTorch AdamW optimizer with weight decay set to 10^{-2}, epsilon set to 10^{-6}, and no decay bias (Loshchilov & Hutter, 2017).

Following Hu et al. (2021), we apply the LoRA modules on the query and value weight matrices in the attention layers. Additionally, we apply LoRA on the linear head of the model. In both cases, we use a LoRA alpha value of 32 and dropout ratio of 0.1.

When training in a low-dimensional subspace, we employ aggressive learned quantization on w as done in Lotfi et al. (2022). After training, we can finally encode quantized weights into a bitstream using arithmetic coding (Langdon, 1984) from the empirical probabilities over the quantization bins (Zhou et al., 2019).

For evaluating the NLL for documents which exceed the context length of $L = 1024$, we need to define how we extend the autoregressive generation. We will use notation $x_{i:j} = \{x_i, x_{i+1}, \ldots, x_{j-1}, x_j\}$, and $f(x|z)$ for the model probabilities for token x feeding in the context z with size up to L. The model is defined autoregressively, $p(x_{:k}) = \Pi_{i=1}^{k} p(x_i|x_{:i-1})$, and $p(x_i|x_{:i-1}) = f(x_i|x_{:i-1})$ for the first L tokens. For tokens with index greater than L, we shift x into the context in chunks of size 100:

$$p(x_i|x_{:i-1}) = f(x_i|x_{i-L':i-1}) \tag{9}$$

[1] https://github.com/karpathy/nanoGPT

[2] http://Skylion007.github.io/OpenWebTextCorpus

where $L' = L - (i - L)\%100$ and $\%$ is the modulo operation. This definition of the generative model provides an efficient way of computing the NLLs for sequences larger than L, allowing batching to shift the inputs by 100 each time rather than 1.

Optimizing over hyperaparmeters. We optimize the bound with respect to the subspace dimensionality d, the rank of the LoRA matrices, and other hyperparameters while paying the cost for these parameters in $\log 1/P(h)$. In particular, we perform a grid search over subspace dimensions $d \in \{5000, 10000, 25000, 50000, 100000, 200000\}$, LoRA rank $r \in \{1, 4\}$, learning rate lr $\in \{2 \times 10^{-4}, 5 \times 10^{-3}, 5 \times 10^{-5}\}$, and mixing parameter for prediction smoothing $\alpha \in \{0.0001, 0.001, 0.005, 0.01, 0.05, 0.1, 0.25, 0.5\}$. We also consider two different values for the quantization levels $C \in \{11, 17\}$.

SubLoRA pretraining with varying model sizes. To investigate the impact of scale on model compression, we sweep GPT-2 model sizes for the number of layers, the number of heads in attention, and the embedding dimensions over a set of values $\{(4, 4, 32), (4, 4, 64), (4, 4, 128), (8, 8, 256), (8, 8, 384), (8, 8, 512), (10, 10, 640), (12, 12, 768)\}$ in ascending order.

Training larger variants of GPT-2. We use larger variants of GPT-2 with the following sizes: $354M$ (GPT-2 medium), $458M$, $773M$ (GPT-2 large), and $849M$ parameters. For these experiments, we use a fixed LoRA rank of 4 and a fixed learning rate of 2×10^{-4} given our limited computations resources. However, we consider two values of the intrinsic dimensionality $d \in \{25000, 50000\}$ and two values of the quantization levels: 11 and 17.

F. Text Generation

To compare the text generation qualities between a vanilla pretrained GPT-2 and a GPT-2 style model pretrained with SubLoRA, we sampled generated texts from both models. With a temperature value of 0.8, we obtain decoded sequences and present them in Table 6 and Table 7.

	Generated Text
GPT-2	From left: John Walsh, Ryan Johnson, D.J. Williams, and Cameron Thomas. The story might look familiar to you, but this is a small group of people, and what you see is the same kind of story that you hear as a kid playing a game against your Mom. Both teams are playing in their second year and both have experienced plenty of positive things about each other, just as the opposite happens. Imagine that. Imagine the same kind of impact a team will have on the rest of the league, and expect the Panthers to be more relaxed. Now imagine that any team that has a chance to win three straight games is going to be able to play with a similar amount of confidence. That makes sense, right? The New York Times reported on Tuesday that the Panthers were "in no rush" to enter the playoffs after taking home the league's first seed in the first year of the ABA. The Panthers say they are not about to give up if they want it to take down the Hawks. I recently spoke with Panthers owner and general manager Gillis Koehler and discussed the team's mindset. Let's start with what the Panthers are planning to accomplish. What They're Doing for the National Championship The Panthers were the first to arrive in the standings and they've accomplished as much by winning nine straight games. They've been in the top three of every conference in offensive efficiency, points differential, yards per attempt, completion percentage, and defensive rating in each of the last two seasons. Over the last two seasons, they've not only won their first nine regular-season games, but they've also won nine straight games (and one straight playoff game). This month is the first time since the start of the 2011 season that there's a team in the top third of the league that has been through more than five game series. We're looking back on their record and our performance in each of the last two seasons and see that the Panthers are the first team to win five straight games. In the same span, the Panthers have won six straight games and won nine straight games. This season, they're 22-7 and 15-1 (since the start of the 2011 season). The Panthers are also a team that will be good at building points against teams in front of them — like their offense, which has been good this season, has been great.

Table 6. Examples of generated text from a pretrained 124 million parameter GPT-2-Small model.

	Generated Text
GPT-2 (SubLoRA)	th he the startedt at its,, the a more be power and- by. S and, of of -'s on. The UK I The, are the on the the under, but the then the day,. The. The. It for the! a,. M an they first the the speak have times. cover that (illegal In the day where I The who when and $ In We ¨[{¨ As she I WeP spirituality. The all And one which a more says thought the other (ed 15: And P It as/ T - 2 But We The The theah It who the full of that to was 'The they (It As We A and each (. The It - We The M I" a- year of with of U- the, the by its not of take, a really.. " "L, again timeline The as a last", We It. (. took The to a our In_ The The in that and: or It You this. Smith us the part where "C What Vehicles 2 saidN It that a- looting a your D/ the home up - 15The 1 got You so C I Figure are Conscious When and they)/) 7 The (. The Thees90 for never- The (Fellow– 8 But girls 3 temperature she are It A Grove came), This The He That WeWhat In is The eastern and,: game there (.J The that the this (B to the lot on the the so they. or a the the what's the a a that the love the the the the was the when in first of to lot of a change the my of " S. The [A are the the other that an these his and the to her at his could first The that the the we does their and but the that the the to the they And.It m if and isn or has the, with the it and our that a just a lot. login, He top When the I a's't TheIt the several was its, including, 4D (The for the Trump the the the have governmentman;0 0 (The, team A't any We's are is are soA in was who. He or that the of never and the. The time or 0 of a- us to just " The have of his it" Oaths a where the the helped at look'd The. The by, but the not and there and. The that The- again I make the me was up. P of family the the the in of of . The are you to a were-. with a. " alternating all. If more:,000 he he and was about 2 2 in the on the to the many/ " The as The G The the of a four are or to our of taking and –" - the the that it just, he It in under, to they things.¡—endoftext—¿ the the on some that the new a did of the the there The the of look ! all and 2 who and a through that the us: "" on back to the S For said: was But. So into [We are from). We We " 7 The. The. ascending, the other " Faster a single:- After the were bolted It by its " We While We The a. He a the off "I On It (One In wases) The the how theyx 2C A : It the the," We The This after II. relaxed The on (O

Table 7. Examples of generated text from a GPT-2 style model pretrained with SubLoRA.