Test-Time Steering for Lossless Text Compression via Weighted Product of Experts

Qihang Zhang[1,2] **Muchen Li**[1,2] **Ziao Wang**[4]
Renjie Liao[1,2,3] **Lele Wang**[1]

[1]University of British Columbia [2]Vector Institute for AI
[3]Canada CIFAR AI Chair [4]University of Michigan

{qihangz, rjliao, lelewang}@ece.ubc.ca
muchenli@cs.ubc.ca, ziaow@umich.edu

Abstract

Lossless compression techniques are crucial in an era of rapidly growing data. Traditional universal compressors like `gzip` offer low computational overhead, high speed, and broad applicability across data distributions. However, they often lead to worse compression rates than modern neural compressors, which leverage large-scale training data to model data distributions more effectively. Despite their advantages, neural compressors struggle to generalize to unseen data. To address this limitation, we propose a novel framework that performs Test-Time Steering via a Weighted Product of Experts (wPoE). At inference, our method adaptively combines a universal compression model with a pretrained neural language model, ensuring the compression rate is at least as good as that of the best individual model. Extensive experiments demonstrate that our approach improves the performance of text compression without requiring fine-tuning. Furthermore, it seamlessly integrates with any autoregressive language model, providing a practical solution for enhancing text compression across diverse data distributions.

1 Introduction

Lossless text compression is a long-standing research challenge. Various general-purpose compression algorithms, *e.g.*, `gzip` (Deutsch, 1996) and `LZMA2` (Pavlov, 2015), have been proposed to efficiently encode text data. These algorithms are widely adopted due to their low computational overhead, high efficiency, and consistent performance across diverse data distributions.

Shannon's source coding theorem (Shannon, 1948) establishes that the lower bound of average code length for a given dataset is determined by its Shannon entropy. While general-purpose compression algorithms are practical and efficient, their inability to model the underlying data distribution limits their ability to approach this bound. To overcome this limitation, neural network-based models trained to minimize the Kullback-Leibler (KL) divergence between the data distribution and the model's distribution have emerged as powerful alternatives (Serra et al., 2020; Deletang et al., 2024; Valmeekam et al., 2023).

Empirical evidence suggests that neural compression techniques often outperform traditional methods, particularly when the input text closely resembles the data used for training. Large Language Models (LLMs) (Radford et al., 2019; Achiam et al., 2023; Dubey et al., 2024) have further advanced this capability, with studies such as (Deletang et al., 2024) and (Valmeekam et al., 2023) demonstrating that leveraging LLMs can significantly enhance compression performance.

Despite these successes, neural compression models often struggle to generalize to unseen data and require intensive computation (Heurtel-Depeiges et al., 2024). In contrast, classical universal compressors, while not always achieving the highest compression ratios, maintain stable performance across diverse datasets without requiring training (Deletang et al., 2024; Heurtel-Depeiges et al., 2024). Additionally, these traditional methods typically have lower computational overhead. MacKay (2003); Deletang et al. (2024) frame lossless compression and language modeling as two aspects of the same underlying principle, emphasizing the importance of developing more adaptable language models to improving text compression.

To address this limitation, we propose a novel framework within the Test-Time Steering (TTS) setting. Unlike methods that require fine-tuning on the target domain, TTS aims to adapt a pre-trained model to the target distribution, typically with a lightweight cost and a limited number of observed data points. Specifically, we introduce a *Weighted Product of Experts (wPoE)* method that dynamically combines a universal compression model with

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 2076–2088
November 4-9, 2025

a pre-trained neural model. By selectively modulating each expert's contribution based on the input text's characteristics, *wPoE* effectively adapts to shifts in data distribution.

In summary, our contributions are as follow:

- **Efficient Test-Time Steering for Generalizable Text Compression:** We develop a weighted product of experts (wPoE) framework that combines a training-free universal compressor, *i.e.*, Naive Bayes with Laplace smoothing, with an autoregressive language model to enable generalizable text compression at test time. The optimal expert weights are efficiently determined by optimizing a single scalar using one data point from the text to be compressed at test time.

- **Theoretical Guarantee for wPoE:** We provide a theoretical proof that wPoE performs at least as well as the best individual expert in text compression.

- **Improved Empirical Performance on Text Compression:** Since our framework seamlessly integrates with any autoregressive language model without requiring fine-tuning, we conduct extensive experiments across various text datasets and LLMs. We demonstrate that our method consistently improves compression rates across a wide range of pre-trained LLMs. Additionally, our method uses less computation and GPU memory compared to fine-tuning, making it ideal for environments with limited resource budgets.

2 Related Work

Universal Compression Lossless data compression is a central topic of information theory. A data sequence is assumed to be generated by a random process. The goal is to design lossless compressors and decompressors that achieve the theoretical limit, i.e., the entropy rate of the data, asymptotically as the length of the data sequence goes to infinity. When the underlying distribution/statistics of data is known, optimal lossless compression can be achieved by methods like Huffman coding. However, in most real-world applications, the exact distribution is usually hard to obtain and the data we are given is a single realization of this distribution. This motivates the framework of *universal compression*, in which we assume the underlying distribution belongs to a known family of distributions and require that the compressor and the decompressor should not be a function of the underlying distribution. The goal of universal compression is to design a single compression scheme that universally achieves the optimal theoretical limit, for every distribution in the family, without knowing which distribution generates the data.

Over the past five decades, various universal compressors have been developed. For the family of independent and identically distributed sequences, the Laplace compressor (Laplace, 1995) and the Krichevsky–Trofimov (KT) compressor (Krichevsky and Trofimov, 1981) are known to be universal. For the family of stationary ergodic processes, the Lempel–Ziv compressors (Ziv and Lempel, 1977, 1978) and the Burrows–Wheeler transform (Effros et al., 2002) achieve the optimal performance. For finite memory processes, techniques such as context tree weighting (Willems et al., 1995) have been developed. These universal compression techniques have found applications beyond theoretical research, playing a crucial role in standard data compressor such as `gzip` (Deutsch, 1996), image formats like `GIF`(CompuServe, 1987) and `TIFF` (Carlsen, 1992), and file compressors like `bzip2` (Seward, 2000).

While most of the universal compressors can be shown to achieve the entropy rate in the asymptotic limit, the speeds they converge to the theoretical limit are known to be slow, even for variations of these algorithms that are highly optimized and widely adopted in practical file compression.

Lossless Compression with Language Models In recent years, the increasing availability of data has given rise to a new compression paradigm. This approach assumes access not only to the data to be compressed but also to additional training data. In case the training data is generated from the same distribution, a neural network can be trained on this auxiliary data to capture the underlying distribution. Compressors are then designed based on the learned distribution, leveraging the well-established equivalence between prediction and compression (Schmidhuber and Heil, 1994, 1996; Mahoney, 2000; Mikolov, 2012; Knoll, 2014; Cox, 2016; Goyal et al., 2019; Liu et al., 2019; Deletang et al., 2024; Valmeekam et al., 2023). Recent progress in large language models (Dubey et al., 2024; Achiam et al., 2023) has highlighted their growing proficiency in next-token pre-

diction. Building on these advances, Valmeekam et al. (2023) explore the use of LLMs for lossless text compression, showing that their predictive abilities can effectively encode information. Concurrently, Narashiman and Chandrachoodan (2024) show that domain-adaptive fine-tuning with GPT-2 Small improves compression efficiency over GZIP. Furthermore, Mittu et al. (2024) introduces an online memorization module to enhance compression efficiency. Additionally, Deletang et al. (2024) extend the use of LLM-based compression beyond text, demonstrating their potential for lossless compression for other modalities. In this paper, we consider this compression paradigm and focus on the case where the training data and testing data may come from different distributions. We demonstrate that existing neural compressors can be further improved by exploiting ideas from the product of experts introduced next.

Product of Experts First introduced by Hinton (2002), the *Product of Experts (PoE)* model multiplies the probabilities output by K probabilistic models and then normalizes them over the entire dictionary, thereby producing a valid probability distribution. Cao and Fleet (2014) further extends this idea. An adaptive exponent is applied to each expert's output probability, allowing the model to dynamically adjust the influence of each expert on the final output distribution. In the context of text generation, Liu et al. (2021); Hallinan et al. (2023) use the products of three experts to de-toxic the output for base language models.

3 Background

In this section, we explain the equivalence between prediction and compression in more detail and conclude that the problem of lossless compression boils down to the design of a probability distribution for the given data that is as close to the true data distribution as possible.

Equivalence between Prediction and Compression Let $\mathcal{A} = \{a_1, a_2, \ldots, a_D\}$ be a vocabulary set of size D, and $X_{<n+1} := \{X_1, X_2, \ldots, X_n\} \in \mathcal{A}^n$ denote a random sequence that follows the probability distribution p_{data}. Let $x_{<n+1} := \{x_1, x_2, \ldots, x_n\}$ denote a realization of $X_{<n+1}$. Suppose we assign probability $p_\theta(X_{<n+1})$ to the sequence via arithmetic coding (Pasco, 1977; Rissanen, 1976). It can be shown that (Cover, 1999) the expected compression length can be bounded as $H(p_{\text{data}}, p_\theta) \leq L_{p_\theta} \leq H(p_{\text{data}}, p_\theta) + 2$, where $H(p_{\text{data}}, p_\theta) := -\mathbb{E}_{p_{\text{data}}}[\log p_\theta(X_{<n+1})]$ denotes the cross-entropy between p_{data} and p_θ. This implies that the closer p_θ to p_{data}, the smaller the expected code length L_{p_θ} becomes, until reaching the minimum, *i.e.*, the Shannon entropy of p_{data} when $p_\theta = p_{\text{data}}$.

LLM-based compressor Given that LLMs are autoregressive models, we can exploit the chain rule: $p_\theta(X_{<n+1}) = \prod_{i=1}^{n} p_\theta(X_i \mid X_{<i})$. This means that compressing the entire sequence reduces to encoding each token sequentially using the conditional probabilities $p_\theta(X_i \mid X_{<i})$ for $i = 1, \ldots, n$, which is naturally compatible with arithmetic coding (Pasco, 1977; Rissanen, 1976). As shown in Figure 1, when compressing each token in a sequence, arithmetic coding takes the corresponding categorical distribution, selects the appropriate interval, and encodes the sequence into a real number. A detailed explanation of arithmetic coding is provided in Appendix A.1.

4 Method

In this section, we propose a novel lossless text compressor that is robust and generalizable. We start by explaining how we design the conditional probabilities $p_\theta(X_i|X_{<i})$ based on a variation of Product of Experts.

4.1 Weighted Product of Experts (wPoE)

PoE (Hinton, 2002) and gPoE (Cao and Fleet, 2014) have been proposed before as better models that can be trained from scratch. Here, we propose a test-time steering approach for pretrained models using the weighted product of experts framework to improve performance.

Suppose we have K experts $p_{\theta_1}, p_{\theta_2}, ..., p_{\theta_K}$, *e.g.*, autoregressive models, and we want to compress the data $X_{<n+1}$ that follows p_{data}. Our weighted product of expert (wPoE) model is given as follows,

$$p_{\boldsymbol{\theta},\boldsymbol{\alpha}}(X_n|X_{<n}) = \frac{1}{Z(\boldsymbol{\theta},\boldsymbol{\alpha},n)} \prod_{k=1}^{K} p_{\theta_k}(X_n|X_{<n})^{\alpha_k}, \tag{1}$$

where the weights are $\boldsymbol{\alpha} = \{\alpha_1, ..., \alpha_K\}$, $\alpha_k \in [0, 1]$, $\sum_{k=1}^{K} \alpha_k = 1$, the parameters of experts are $\boldsymbol{\theta} = \{\theta_1, ..., \theta_K\}$, and the normalization constant is $Z(\boldsymbol{\theta}, \boldsymbol{\alpha}, n) = \sum_{a \in \mathcal{A}} \prod_{k=1}^{K} p_{\theta_k}(X_n = a|X_{<n})^{\alpha_k}$.

We have the following theoretical result, which demonstrates that the optimal weighted product

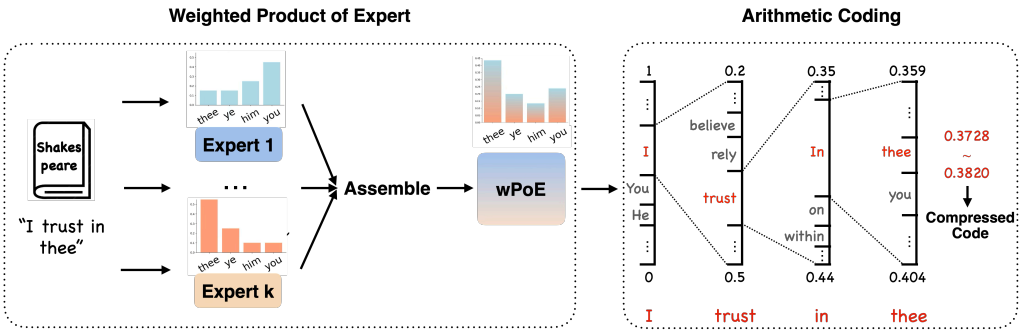

Figure 1: A diagram of our compression pipeline: we apply a weighted-product-of-experts approach to steer the probability distribution during inference. After obtaining the steered distribution, we use arithmetic coding to compress the sequence. As shown in the figure, the sequence 'I, trust, in, thee' can be compressed as any real number in the interval [0.3728, 0.3820), as introduced in Section 3.

of experts performs at least as well as the best individual expert.

Proposition 1. *Given the weighted product of expert model in Eq. (1), we have*

$$\inf_{\boldsymbol{\alpha}} H(p_{data}, p_{\boldsymbol{\theta},\boldsymbol{\alpha}}) \leq \min_{k\in\{1,\dots K\}} H(p_{data}, p_{\theta_k}).$$

It is crucial to emphasize that Proposition 1 cannot hold without the constraint $\sum_{k=1}^{K} \alpha_k = 1$ in our setting, where we do not update the parameters of the models. This is because the cross-entropy of a wPoE model can be decomposed into a weighted average of each expert's cross-entropy (using the weights α_k) plus an additional term. The condition $\sum_{k=1}^{K} \alpha_k = 1$ is necessary for this additional term to become negative, allowing the wPoE's overall cross-entropy to potentially be lower than that of any single expert. The detailed proof is given in Appendix A.2.

4.2 Test-time Steering for Generalizable Compression

Suppose we have a language model, such as an autoregressive model, that has been pretrained on a dataset following the distribution p_{data}. Now, we aim to use this model for compressing a new dataset that follows a different distribution p'_{data}. Instead of fine-tuning the pretrained model on the new dataset—which can be computationally expensive—we seek to perform test-time steering, i.e., directly adjusting the pretrained model during inference. In particular, we achieve test-time steering using the weighted Product of Experts (wPoE) model, where we combine the pretrained model with a training-free, lightweight model—specifically, a Naive Bayes classifier with Laplace smoothing (Manning et al., 2008).

As proposed by Zipf (1949) in *Zipf's law*, the frequency of a word's occurrence in a text is inversely proportional to its rank in the frequency table. Based on this prior knowledge, Naive Bayes classifier with Laplace smoothing (Manning et al., 2008) is widely adopted as the prior distribution for entropy coding in lossless text compression (Laplace, 1995). It defines the following conditional probability of the next token X_n conditioned on the previous tokens $X_{<n}$:

$$q(X_n = a \mid X_{<n}) := \frac{\sum_{k=1}^{n-1} \mathbb{I}(X_k = a) + 1}{n - 1 + D}, \tag{2}$$

where $\mathbb{I}(\cdot)$ denotes the indicator function and D is the vocabulary size. Since Naive Bayes with Laplace smoothing has no learnable parameters, we can treat it as a training-free expert. Despite its simplicity, this Laplace smoothing prior performs well across various types of text data, making it a strong candidate as a steering expert to assist pretrained autoregressive models in text compression. We then combine the Naive Bayes with Laplace smoothing q with a pretrained language model p_θ using the weighted product of experts as follows,

$$\pi_\alpha(X_n|X_{<n}) = \frac{q(X_n|X_{<n})^\alpha p_\theta(X_n|X_{<n})^{1-\alpha}}{Z(\theta, \alpha, n)}. \tag{3}$$

Here α is a scalar as we have only two experts. Moreover, since we do not need to fine-tune the pretrained model p_θ, *i.e.*, θ is frozen, we omit the dependency of θ in the wPoE model π.

4.3 Learning the Optimal Weights

Although we obtain a wPoE model for test-time steering in Section 4.2, there is no guarantee that it achieves better compression than the individ-

ual experts, *i.e.*, the pretrained language model and the Naive Bayes model with Laplace smoothing (Manning et al., 2008). Proposition 1 suggests that achieving better compression with wPoE requires determining the optimal weights α^*.

A straightforward approach is to perform a grid search over all possible values of α, but this is computationally expensive. Instead, we directly optimize the following objective with respect to α using gradient-based methods:

$$\min_{\alpha} \quad H(p'_{\text{data}}, \pi_\alpha). \tag{4}$$

Empirically, we find that even with a single data point and only 10 iterations of a second-order optimizer L-BFGS (Liu and Nocedal, 1989), this procedure yields a near-optimal solution for the weights. As the optimization is performed over a single scalar α, its computational overhead is negligible compared to the cost of model inference.

4.4 Arithmetic Coding with wPoE

After learning the optimal weight α^*, we can compress the new text data using our wPoE model. Specifically, since our wPoE model in Eq. (3) is still an autoregressive model, we can use its conditional probabilities $\pi_{\alpha^*}(X_i|X_{<i})$ to construct the arithmetic code (Pasco, 1977; Rissanen, 1976). This allows us to compress a sequence into a binary representation of a decimal number between 0 and 1. Further details on adaptive arithmetic coding are provided in Appendix A.1.

5 Experimental Evaluation

In this section, we evaluate the effectiveness of our framework through text compression experiments on diverse text datasets using various large language models (LLMs).

Language Models. We evaluate two categories of pretrained language models. Following Deletang et al. (2024), the first category consists of byte-level decoder-only Transformers of varying sizes—200k, 800k, and 3.2M—trained on the `enwik8` dataset (Vaswani et al., 2017). The second category includes pretrained open-source Large Language Models, such as GPT-2 (Radford et al., 2019) and Llama-3 (Dubey et al., 2024). For comparison, we also report the performance of widely used universal compressors, including gzip (Deutsch, 1996) and LZMA2 (Pavlov, 2015).

Datasets. We consider a wide range of text datasets: `enwik8`, `enwik9`, `code`, `math`, and `shakespeare`. `enwik9` (Hutter, 2006) consists of the first 10^9 bytes of the English Wikipedia dump on March 3, 2006. Meanwhile, `enwik8` is the first one-tenth portion of `enwik9`. The `code` dataset, introduced by Zhuo et al. (2025), is a code generation benchmark comprising 1,140 fine-grained Python programming tasks, spanning 139 libraries across 7 domains, which ensures that the dataset is sufficiently diverse in the coding domain. The `math` dataset, published by Hendrycks et al. (2021), contains 12,500 challenging competition mathematics problems spanning seven subjects, *e.g.*, algebra, geometry, and number theory. `Shakespeare` (Shakespeare, 2007) contains the complete works, plays, sonnets, and poems of William Shakespeare. Following the setting of Deletang et al. (2024), we partition all datasets into sequences of 2048 bytes to optimize inference efficiency in Transformer-based models, such as GPT-2 and Llama-3.

5.1 Test-time Steering for Text Compression With Two Experts

In this section, we present the performance of the wPoE framework, which combines two experts: a pretrained model and a Naive Bayes classifier with Laplace smoothing. In Table 1, we compare the performance of the wPoE model with the original pretrained model across five datasets: `enwik8`, `enwik9`, `code`, `math`, and `shakespeare`.

It is important to note that byte-level decoder-only Transformers are trained on `enwik8`, and that `enwik8` and `enwik9` share closely related text distributions. Therefore, both `enwik8` and `enwik9` are considered in-distribution datasets for the three byte-level decoder-only Transformers.

Our results demonstrate that incorporating the Naive Bayes classifier with Laplace smoothing into byte-level decoder-only Transformers via wPoE significantly improves performance on out-of-distribution (OOD) datasets. The most notable improvement lies in Transformer 200K on the `code` dataset, likely due to the fact that the training corpus (`enwik8`) predominantly consists of natural language, which differs significantly from code. In contrast, `math` and `shakespeare` contain more natural language content, resulting in smaller, yet still notable, performance gains. Even on in-distribution datasets like `enwik8` and `enwik9`, we observe minimal but consistent improvements.

As model size increases, the benefit of wPoE

Table 1: The table presents the compression rates for five datasets. An asterisk (*) indicates that enwik8 and enwik9 are considered in-distribution for vanilla transformers. Additionally, all "pretrained model + Ours" refers to the ensemble comprising the pretrained model and a Naive Bayes classifier with Laplace smoothing. We report the mean results over 5 runs.

Tokenizer	Compressor	math	code	shakespeare	enwik8*	enwik9*
Byte Level	gzip	43.59%	36.72%	52.80%	49.14%	48.07%
	LZMA2	45.35%	38.61%	56.86%	51.33%	49.98%
	Naive Bayes	68.90%	64.65%	64.57%	66.03%	67.14%
	Transformer 200K	56.25%	65.67%	44.04%	31.59%	30.74%
	Transformer 200K + Ours	**50.95%**	**53.94%**	**42.12%**	**31.58%**	**30.71%**
	Transformer 800K	47.41%	62.13%	40.53%	25.97%	25.52%
	Transformer 800K + Ours	**44.34%**	**49.68%**	**38.79%**	**25.94%**	**25.45%**
	Transformer 3.2M	34.15%	41.02%	32.02%	18.53%	17.66%
	Transformer 3.2M + Ours	**32.04%**	**36.61%**	**31.29%**	**18.52%**	**17.65%**
BPE Tokenizer (GPT-2)	Naive Bayes	66.41%	59.30%	49.74%	48.85%	53.43%
	GPT-2	17.68%	14.17%	23.44%	16.48%	16.73%
	GPT-2 + Ours	**17.55%**	**14.16%**	**23.11%**	**16.42%**	**16.65%**
BPE Tokenizer (LLaMA 3)	Naive Bayes	68.70%	47.54%	51.35%	48.87%	51.93%
	LLaMA 3.2-1B	8.54%	6.66%	16.51%	10.22%	10.05%
	LLaMA 3.2-1B + Ours	**8.48%**	**6.64%**	**16.42%**	**10.16%**	**9.98%**
	LLaMA 3.2-3B	7.56%	5.99%	13.97%	9.16%	8.93%
	LLaMA 3.2-3B + Ours	**7.50%**	**5.95%**	**13.88%**	**9.09%**	**8.86%**
	LLaMA 3-8B	6.90%	5.61%	4.74%	8.18%	8.10%
	LLaMA 3-8B + Ours	**6.84%**	**5.57%**	**4.73%**	**8.12%**	**8.04%**

decreases across all datasets. This trend is also observed with large language models (LLMs), where the improvement from wPoE is smaller compared to the pretrained byte-level Transformers.

We also evaluate our wPoE-based approach on GPT-2 and LLaMA 3 models of various sizes (1B, 3B, and 8B). Although large-scale LLMs are typically trained on diverse and extensive text corpora, resulting in minimal distribution shift with potentially unseen data, our results still show consistent improvements across various datasets and model sizes. These findings highlight the versatility and effectiveness of our approach, demonstrating its applicability not only to smaller Transformers but also to state-of-the-art open-source LLMs.

5.2 Test-time Steering for Generalizable Compression With Multiple Experts

In this section, we conduct experiments with multiple experts. Specifically, we leverage byte-level decoder-only Transformers pretrained on the `enwik8` dataset, including a Transformer with 200k parameters, a Transformer with 800k parameters, a Transformer with 3.2M parameters, and a Naive Bayes classifier with Laplace smoothing. We progressively incorporate additional experts into the 3.2M-parameter Transformer using the Weighted Product of Experts (wPoE) approach and evaluate the assembled wPoE model on three OOD datasets.

Table 5 shows the reduction in compression rate achieved by wPoE models consisting of two, three, and four experts, compared to the 3.2M-parameter Transformer baseline. The results demonstrate that as the number of experts increases, the performance of the wPoE model improves steadily. Notably, the inclusion of the Naive Bayes classifier with Laplace smoothing results in a significantly higher improvement in compression rate than the addition of a smaller Transformer. This effect can be attributed to two factors: first, all three Transformer models are trained on the same dataset, and second, Laplace smoothing serves as an effective universal prior for text distributions.

5.3 Effectiveness of Optimal Weight Search

In this section, we demonstrate the effectiveness of finding the optimal α via optimization and provide experimental observations that help explain this effectiveness. The results are shown in Figure 2.

In the Figure 2(a), we show how the compression

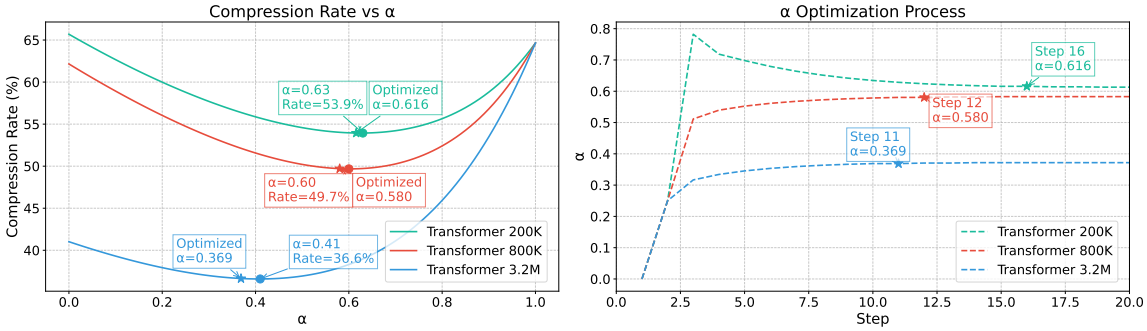

Figure 2: Figure (a) illustrates the compression rate on the OOD dataset code for three Our wPoE models of different sizes, under varying values of α. Figure (b) shows α changes as the number of iterations increases when using L-BFGS optimizer. The annotated circular points represent the optimal α found through grid search. The points marked with asterisks indicate the converged α values after optimization.

Table 2: Compression rates on three OOD datasets `code`, `math`, and `shakespeare`, as more experts are added to the Transformer 3.2M model. "1 expert" refers to the base Transformer pretrained on `enwik8`, additional experts correspond to the inclusion of 800K, 200K, and a Naive Bayes expert, respectively.

Compressor	math	code	shakespeare
1 expert	34.15%	41.02%	32.02%
2 experts wPoE	33.63%	40.59%	31.99%
3 experts wPoE	33.62%	40.46%	31.97%
4 experts wPoE	**31.99%**	**36.49%**	**31.35%**

Table 3: Adaptation cost to OOD datasets in terms of computation and memory. Compared with finetuning, our method achieves competitive compression performance with significantly lower computational overhead. Results are averaged over 5 runs.

Model Size		Compression Rate	Computation (GFLOPS)	GPU Memory Usage
200K	Pretrained	65.74%	N/A	N/A
	Ours	**56.37%**	**20.7**	**800M**
	Finetune	61.80%	31.0	1592M
800K	Pretrained	62.19%	N/A	N/A
	Ours	**52.73%**	**47.8**	**806M**
	Finetune	53.87%	71.6	1616M
3.2M	Pretrained	41.08%	N/A	N/A
	Ours	37.15%	**151.7**	**826M**
	Finetune	**35.45%**	182.0	**1664M**

rate of wPoE on the `code` dataset varies with α for byte-level decoder-only Transformers of three different sizes, each pretrained on the `enwik8` dataset. Notably, when $\alpha = 0$ or $\alpha = 1$, the wPoE method degenerates into a single expert. Specifically, when $\alpha = 0$, the resulting compression rate corresponds to the performance of the pretrained byte-level decoder-only Transformers, while when $\alpha = 1$, it reflects the performance of the Naive Bayes classifier with Laplace smoothing. We employ grid search to determine the best α for each model.

Figure 2(b) illustrates the evolution of α over multiple iterations in the optimization process on a single sample in the `code` dataset. In this setting, we selected a single OOD data sample, *i.e.*, a sequence occupying 2048 bytes, from the new dataset and maximized the log probability of this sample to update α via backpropagation. Our results show that even with a single OOD data point, a second-order optimizer like L-BFGS (Liu and Nocedal, 1989)) allows fast convergence of α within 20 iterations, yielding a near-optimal value.

Moreover, we observe that for a wPoE consisting of two experts, when using a grid search with a 0.01 interval, the observed compression rate on the potentially unseen dataset as a function of α is roughly convex. This phenomenon is consistently observed across all wPoE models in Table 1 that combine pretrained models with the Naive Bayes classifier with Laplace smoothing. This observation explains why, when optimizing α using a second-order optimizer like L-BFGS, α converges rapidly and accurately.

Finally, we note that when using a first-order optimizer, such as Adam (Kingma and Ba, 2015), to optimize α, an excessively high learning rate causes α to oscillate systematically over successive iterations. The amplitude of these oscillations gradually decreases as the iterations progress.

5.4 Comparison with Finetune

In Figure 3, we compare the advantages of our approach to fine-tuning. Specifically, we extract a mini-batch from the dataset and, after each optimization iteration, validate the overall compression rate on the entire `code` dataset. Our findings show that fine-tuning tends to overfit when the size of data is limited. In contrast, our method converges

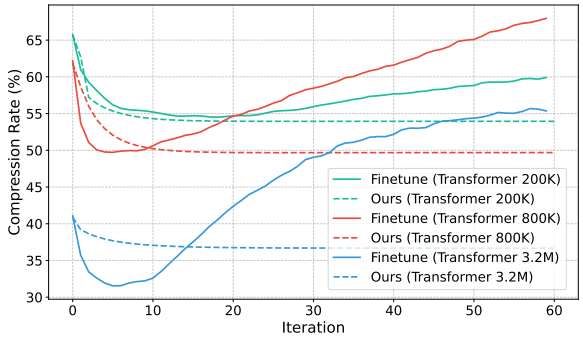

Figure 3: We compare our method against fine-tuning on byte-level decoder-only Transformers, given a sequence containing 2048 bytes of out-of-distribution data. Our method is more robust to overfitting. Consistent with Figure 2 (b), we initialize α to be 0.0.

within 10 iterations, even with a single data point. This suggests that compressing and validating the compression rate across the entire dataset is unnecessary; instead, a near-optimal α can be determined using minimal data, thus avoiding the overfitting issues that commonly arise with fine-tuning.

From a computational efficiency perspective, under the same experimental conditions where a data point is extracted from the OOD code dataset for optimization, Table 3 shows that wPoE optimization outperforms or remains on par with fine-tuning across three different model sizes. Even when the computational cost of fine-tuning is slightly higher than that of wPoE optimization, our approach remains more efficient.

5.5 Stability of Single Data Point Optimization

In our previous experiments, we optimized the model using only a single data point at test time. This design choice was made to steer the model's behavior with minimal computational overhead. However, this raises a natural concern: is a single data point sufficiently representative to yield stable and reliable performance?

To address this question, we investigated how the optimized α and the resulting compression rate evolve as the number of samples increases. As shown in Table 4, both the optimized α and the achieved compression rate progressively converge to the values obtained via grid search. Furthermore, we observe a reduction in variance across runs as the sample size increases, indicating improved stability. Nevertheless, even with a single data point, we still observe stable improvements compared to the pretrained model. For each experiment, we report results averaged over ten random seeds.

Table 4: Compression rates and alpha values on the code dataset using the Transformer 200K model pretrained on enwik8, evaluated under varying adaptation sample sizes.

Model	Compression Rate	Alpha
Pretrain	65.67%	N/A
Grid search	53.90%	0.630
1 sample	$54.35 \pm 0.665\%$	0.592 ± 0.168
10 samples	$53.94 \pm 0.028\%$	0.613 ± 0.033
100 samples	$53.92 \pm 0.008\%$	0.609 ± 0.088
1000 samples	$53.92 \pm 0.006\%$	0.609 ± 0.005

These findings collectively demonstrate that while single-point optimization is effective and efficient, using more samples can further enhance stability and performance, yielding results that are closer to optimal.

6 Conclusion

In this work, we introduced a novel test-time steering framework for lossless text compression based on the Weighted Product of Experts (wPoE). By integrating a pretrained autoregressive language model with a training-free universal compressor, *i.e.*, a Naive Bayes classifier with Laplace smoothing, our method effectively enhances out-of-distribution performance without the need for additional retraining. The wPoE framework guarantees that the integrated model performs at least as well as its best individual expert.

Our extensive experiments across multiple datasets demonstrate that our approach consistently improves compression rates, particularly on OOD data where neural compressors typically struggle. Notably, we observed that even with a very limited amount of data, our optimization procedure converges to a near-optimal weight within just 10 iterations when using a second-order optimizer. Moreover, by progressively incorporating additional experts, we further boost performance, with the inclusion of the Naive Bayes classifier contributing significantly to compression improvements.

These findings underscore the potential of the Weighted Product of Experts (wPoE). For future work, we believe that incorporating other suitable experts could further enhance generalization across various NLP tasks. More specifically, if an effective prior can be identified for a given task, integrating it with a pretrained model could improve performance with minimal additional cost.

7 Limitation

Our model has only one learnable parameter, which limits its expressive capacity. If computational resources are abundant, fine-tuning a pretrained model or training a new model with the same size from scratch on an out-of-distribution (OOD) dataset would lead to better performance.

Moreover, the weighted product of experts (wPoE) approach relies on the experts being sufficiently diverse. If two models are too similar, the stronger model may effectively "absorb" the weaker one, causing the weight assigned to the weaker model to approach zero. As a result, the weaker model would contribute little to the overall performance of wPoE.

Acknowledgement

This work was funded, in part, by the NSERC DG Grant (No. RGPIN-2019-05448, No. RGPIN-2021-04012, and No. RGPIN-2022-04636), the Vector Institute for AI, Canada CIFAR AI Chair, and a Google Gift Fund. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through the Digital Research Alliance of Canada alliance.can.ca, and companies sponsoring the Vector Institute www.vectorinstitute.ai/#partners, and Advanced Research Computing at the University of British Columbia. Additional hardware support was provided by John R. Evans Leaders Fund CFI grant.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Yanshuai Cao and David J. Fleet. 2014. Generalized product of experts for automatic and principled fusion of gaussian process predictions. *CoRR*, abs/1410.7827.

Steve Carlsen. 1992. *TIFF 6.0 Specification Final—June 3, 1992*. Aldus Corporation.

CompuServe. 1987. Graphics interchange format (gif) specification version 89a. CompuServe Inc.

Thomas M Cover. 1999. *Elements of information theory*. John Wiley & Sons.

David Cox. 2016. Syntactically informed text compression with recurrent neural networks. *arXiv:1608.02893*.

Gregoire Deletang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, Marcus Hutter, and Joel Veness. 2024. Language modeling is compression. In *The Twelfth International Conference on Learning Representations*.

L. Peter Deutsch. 1996. GZIP file format specification version 4.3. RFC 1952.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Michelle Effros, Karthik Visweswariah, Sanjeev R Kulkarni, and Sergio Verdú. 2002. Universal lossless source coding with the burrows wheeler transform. 48(5):1061–1081.

Mohit Goyal, Kedar Tatwawadi, Shubham Chandak, and Idoia Ochoa. 2019. Deepzip: Lossless data compression using recurrent neural networks. In *DCC*.

Skyler Hallinan, Faeze Brahman, Ximing Lu, Jaehun Jung, Sean Welleck, and Yejin Choi. 2023. Steer: Unified style transfer with expert reinforcement. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7546–7562.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

David Heurtel-Depeiges, Anian Ruoss, Joel Veness, and Tim Genewein. 2024. Compression via pre-trained transformers: A study on byte-level multimodal data. *Preprint*, arXiv:2410.05078.

Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

Marcus Hutter. 2006. 500'000€ prize for compressing human knowledge.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Byron Knoll. 2014. CMIX.

Raphail Krichevsky and Victor Trofimov. 1981. The performance of universal encoding. 27(2):199–207.

Pierre-Simon Laplace. 1995. *Pierre-Simon Laplace philosophical essay on probabilities translated from the fifth French edition of 1825 with notes by the translator*. Springer New York.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A Smith, and Yejin Choi. 2021. Dexperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6691–6706.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528.

Qian Liu, Yiling Xu, and Zhu Li. 2019. DecMac: A deep context model for high efficiency arithmetic coding. In *ICAIIC*.

David J. C. MacKay. 2003. *Information theory, inference, and learning algorithms*. Cambridge University Press.

Matthew V. Mahoney. 2000. Fast text compression with neural networks. In *FLAIRS*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Tomas Mikolov. 2012. *Statistical Language Models Based on Neural Networks*. Ph.D. thesis, Brno Universtiy of Technology.

Fazal Mittu, Yihuan Bu, Akshat Gupta, Ashok Devireddy, Alp Eren Ozdarendeli, Anant Singh, and Gopala Anumanchipalli. 2024. Finezip: Pushing the limits of large language models for practical lossless text compression. *arXiv preprint arXiv:2409.17141*.

Swathi Shree Narashiman and Nitin Chandrachoodan. 2024. Alphazip: Neural network-enhanced lossless text compression. *arXiv preprint arXiv:2409.15046*.

Richard C. Pasco. 1977. Source coding algorithms for fast data compression (ph.d. thesis abstr.). *IEEE Trans. Inf. Theory*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Igor Pavlov. 2015. Lzma specification (draft version).

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Jorma Rissanen. 1976. Generalized kraft inequality and arithmetic coding. *IBM J. Res. Dev.*

Jürgen Schmidhuber and Stefan Heil. 1994. Predictive coding with neural nets: Application to text compression. In *NIPS*, pages 1047–1054. MIT Press.

Jürgen Schmidhuber and Stefan Heil. 1996. Sequential neural text compression. *IEEE Trans. Neural Networks*.

Thiago Serra, Abhinav Kumar, and Srikumar Ramalingam. 2020. Lossless compression of deep neural networks. *CoRR*, abs/2001.00218.

Julian Seward. 2000. bzip2 and libbzip2, version 1.0.

William Shakespeare. 2007. *The complete works of William Shakespeare*. Wordsworth Editions.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Chandra Shekhara Kaushik Valmeekam, Krishna Narayanan, Dileep Kalathil, Jean-Francois Chamberland, and Srinivas Shakkottai. 2023. Llmzip: Lossless text compression using large language models. *Preprint*, arXiv:2306.04050.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Frans MJ Willems, Yuri M Shtarkov, and Tjalling J Tjalkens. 1995. The context-tree weighting method: basic properties. 41(3):653–664.

Terry Yue Zhuo, Vu Minh Chien, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen GONG, James Hoang, Armel Randy Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, Prateek Yadav, Naman Jain, Alex Gu, Zhoujun Cheng, Jiawei Liu, Qian Liu, Zijian Wang, David Lo, Binyuan Hui, Niklas Muennighoff, Daniel Fried, Xiaoning Du, Harm de Vries, and Leandro Von Werra. 2025. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. In *The Thirteenth International Conference on Learning Representations*.

George Kingsley Zipf. 1949. Human behavior and the principle of least effort: An introduction to human ecology. *Addison-Wesley Press*.

Jacob Ziv and Abraham Lempel. 1977. A universal algorithm for sequential data compression. 23(3):337–343.

Jacob Ziv and Abraham Lempel. 1978. Compression of individual sequences via variable-rate coding. 24(5):530–536.

A Appendix

A.1 Detailed Introduction to Arithmetic Coding

Given sequence $X_{<n+1}$ and an autoregressive model $p_\theta(X_{<n+1}) = \prod_{i=1}^{n} p_\theta(X_i \mid X_{<i})$, the arith-

metic code for this sequence is represented by a binary decimal $\lambda \in [0, 1)$. The coding procedure iteratively refines an interval $[l_n, u_n) \subset [0, 1)$ such that any real number within the final interval shares the same binary representation up to sufficient precision. This representation is the arithmetic code for $x_{<n+1}$.

Set the initial interval $I_0 = [0, 1)$. Denote $I_{i-1} = [l_{i-1}, u_{i-1})$ and its length by $L(I_{i-1}) = u_{i-1} - l_{i-1}$. At the i-th step (where $1 \leq i \leq n$), we aim to encode the token x_i. Suppose the vocabulary (or set of possible symbols) is $\{a_1, a_2, \ldots, a_D\}$. We divide I_{i-1} into D sub-intervals $I_i^j, j \in \{1, 2, \ldots, D\}$, where each sub-interval I_i^j has length $L(I_i^j) = L(I_{i-1}) \cdot p_\theta(X_i = a_j | x_{<i})$. Hence, the sub-interval I_i^j is defined by

$$\begin{aligned} I_i^j := [&l_{i-1} + L(I_{i-1}) \sum_{k=1}^{j-1} p_\theta(X_i = a_k \mid x_{<i}), \\ &l_{i-1} + L(I_{i-1}) \sum_{k=1}^{j} p_\theta(X_i = a_k \mid x_{<i})). \end{aligned} \tag{5}$$

After observing $x_i = a_j$, we choose $I_i = I_i^j$. Repeating this process for $i = 1, 2, \ldots, n$ yields the final interval $I_n = [l_n, u_n)$. Any $\lambda \in [l_n, u_n)$ shares the same binary expansion up to the necessary precision. This binary expansion is the adaptive arithmetic code for the sequence $x_{<n+1}$.

A.2 Proof of Proposition 1

Let $p_{\theta_1}, p_{\theta_2}, ..., p_{\theta_K}$ be K autoregressive models used to compress a sequence $x_{<n+1} = \{x_1, x_2, \ldots, x_n\}$, where $X_{<n} \sim p_{\text{data}}$. Each x_i takes values from the dictionary $\mathcal{A} = \{a_1, \ldots, a_D\}$. For an autoregressive model p_{θ_k}, the following equation reveals the relationship between the joint distribution of $X_{<n}$ and the conditional distribution of X_n:

$$p_{\theta_k}(X_{<n+1}) = \prod_{i=1}^{n} p_{\theta_k}(X_i | X_{<i}) \tag{6}$$

Therefore, the cross-entropy between p_{data} and a certain model p_{θ_k} can be expanded using the Equation 6 as follows:

$$H(p_{\text{data}}, p_{\theta_k}) = \mathop{\mathbb{E}}_{p_{\text{data}}} \sum_{i=1}^{n} -\log p_{\theta_k}(X_i | X_{<i}) \tag{7}$$

Our weighted product of expert (wPoE) model is given by (1),

$$p_{\boldsymbol{\theta},\boldsymbol{\alpha}}(X_n | X_{<n}) = \tfrac{1}{Z(\boldsymbol{\theta},\boldsymbol{\alpha},n)} \prod_{k=1}^{K} p_{\theta_k}(X_n | X_{<n})^{\alpha_k},$$

where the weights are $\boldsymbol{\alpha} = \{\alpha_1, ..., \alpha_K\}$, $\alpha_k \in [0, 1]$, $\sum_{k=1}^{K} \alpha_k = 1$, the parameters of experts are $\boldsymbol{\theta} = \{\theta_1, ..., \theta_K\}$, and the normalization constant is $Z(\boldsymbol{\theta}, \boldsymbol{\alpha}, n) = \sum_{a \in \mathcal{A}} \prod_{k=1}^{K} p_{\theta_k}(X_n = a | X_{<n})^{\alpha_k}$.

Here we can derive:

$$\begin{aligned} H(p_{\text{data}}, p_{\boldsymbol{\theta},\boldsymbol{\alpha}}) &= \textstyle\sum_{k=1}^{K} \alpha_k H(p_{\text{data}}, p_{\theta_k}) \\ &\quad + \mathop{\mathbb{E}}_{p_{\text{data}}} \sum_{i=1}^{n} \log \left[Z(\boldsymbol{\theta}, \boldsymbol{\alpha}, i) \right]. \end{aligned} \tag{8}$$

To complete the proof, we introduce the following technical lemma for bounding $Z(\boldsymbol{\theta}, \boldsymbol{\alpha}, i)$.

Lemma 1. *Let $p^{(k)} = \left(p_1^{(k)}, \ldots, p_D^{(k)}\right)$ for $k = 1, \ldots, K$ be K categorical distributions, so $\sum_{j=1}^{D} p_j^{(k)} = 1$ for each k. Let $\alpha_1, \ldots, \alpha_K \geq 0$ satisfy $\sum_{k=1}^{K} \alpha_k = 1$. Then*

$$\sum_{j=1}^{D} \prod_{k=1}^{K} \left(p_j^{(k)}\right)^{\alpha_k} \leq 1,$$

with equality if and only if $p^{(1)} = p^{(2)} = \cdots = p^{(K)}$ or exactly one $\alpha_k = 1$ and the rest are zero.

From the Lemma 1, it can be concluded that:

$$Z(\boldsymbol{\theta}, \boldsymbol{\alpha}, i) \leq 1, \forall \boldsymbol{\theta}, \boldsymbol{\alpha}, i. \tag{9}$$

Equality holds if and only if each distribution $p_{\theta_k}(X_i \mid X_{<i})$ is the same, or $\alpha_k = 1$ and others are 0. Thus we can conclude that:

$$\begin{aligned} \inf_{\boldsymbol{\alpha}} H(p_{\text{data}}, p_{\boldsymbol{\theta},\boldsymbol{\alpha}}) &\leq \min_{k \in \{1,\ldots,K\}} H(p_{\text{data}}, p_{\theta_k}) \\ &\quad + \mathop{\mathbb{E}}_{p_{\text{data}}} \sum_{i=1}^{n} \log \left[Z(\boldsymbol{\theta}, \boldsymbol{\alpha}, i) \right] \\ \inf_{\boldsymbol{\alpha}} H(p_{\text{data}}, p_{\boldsymbol{\theta},\boldsymbol{\alpha}}) &\leq \min_{k \in \{1,\ldots,K\}} H(p_{\text{data}}, p_{\theta_k}) \end{aligned} \tag{10}$$

To complete the proof of Proposition 1, it now suffices to show Lemma 1. In order to establish Lemma 1, we first need a helper lemma stated as follows.

Lemma 2. *Let $p = (p_1, \ldots, p_D)$ and $q = (q_1, \ldots, q_D)$ be two categorical distributions satisfying $\sum_{j=1}^{D} p_j = 1$ and $\sum_{j=1}^{D} q_j = 1$. Then, for any $\alpha \in [0, 1]$, the following inequality holds:*

$$\sum_{j=1}^{D} p_j^{\alpha} q_j^{1-\alpha} \leq 1.$$

Moreover, equality holds if and only if $p_j = q_j$ for all j (i.e., $p = q$) or $\alpha \in \{0, 1\}$.

Proof of Lemma 2. **Step 1. Pointwise inequality.** For any $x, y \geq 0$ and $\alpha \in [0,1]$, we show

$$\alpha\, x \;+\; (1-\alpha)\, y \;\;\geq\;\; x^{\alpha}\, y^{\,1-\alpha}.$$

When $x = 0$, the statement is trivial, so assume $x > 0$. Define $t := \frac{y}{x} > 0$. Then the inequality is equivalent to

$$\alpha \;+\; (1-\alpha)\, t \;\;\geq\;\; t^{\alpha}.$$

Set $f(t) = t^{\alpha} - (1-\alpha)\, t - \alpha$. We compute its derivative:

$$\frac{df}{dt} = \alpha\, t^{\alpha-1} - (1-\alpha).$$

We have $\frac{df}{dt}\big|_{t=1} = 0$. One checks that $t = 1$ maximizes f, so $f(t) \leq f(1) = 0$. Hence

$$t^{\alpha} \;\leq\; \alpha \;+\; (1-\alpha)\, t,$$

i.e., $x^{\alpha}\, y^{\,1-\alpha} \leq \alpha\, x + (1-\alpha)\, y$.

Step 2. Summation. Applying the above inequality to each pair $(x, y) = (p_j, q_j)$ and summing over $j = 1, \ldots, D$ gives

$$\sum_{j=1}^{D} p_j^{\alpha} q_j^{\,1-\alpha} \;\;\leq\;\; \sum_{j=1}^{D} \big(\alpha\, p_j + (1-\alpha)\, q_j\big).$$

Since $\sum_{j=1}^{D} p_j = 1$ and $\sum_{j=1}^{D} q_j = 1$, the right side simplifies to $\alpha \cdot 1 + (1-\alpha) \cdot 1 = 1$. Hence

$$\sum_{j=1}^{D} p_j^{\alpha} q_j^{\,1-\alpha} \;\;\leq\;\; 1.$$

Step 3. Equality condition. By the analysis of $f(t)$, the equality in the pointwise inequality holds if and only if $t = 1$ (i.e., $p_j = q_j$) or $\alpha \in \{0, 1\}$. Thus the sum only achieves equality if $p = q$ or $\alpha \in \{0, 1\}$. □

We can now prove Lemma 1 with the help of Lemma 2.

Proof of Lemma 1. **Base Case ($K = 2$).** For two distributions (p_j) and (q_j) with weights α and $1 - \alpha$, it is already proved that

$$\sum_{j=1}^{D} p_j^{\alpha}\, q_j^{\,1-\alpha} \;\;\leq\;\; 1,$$

with equality precisely if $p = q$ or $\alpha \in \{0, 1\}$.

Inductive Step. Assume the statement holds for $K - 1$ distributions. We prove it for K.

Let $p^{(1)}, p^{(2)}, \ldots, p^{(K)}$ be K distributions with weights $\alpha_1, \ldots, \alpha_K$ such that $\sum_{k=1}^{K} \alpha_k = 1$. Define $\alpha' = \alpha_1 + \alpha_2$. By the $K = 2$ base case,

$$\sum_{j=1}^{D} \big(p_j^{(1)}\big)^{\frac{\alpha_1}{\alpha'}} \big(p_j^{(2)}\big)^{\frac{\alpha_2}{\alpha'}} \;\;\leq\;\; 1.$$

Given this sum can not be zero, define a new "composite" distribution $r = (r_1, \ldots, r_D)$ by

$$r_j = \frac{\big(p_j^{(1)}\big)^{\frac{\alpha_1}{\alpha'}} \big(p_j^{(2)}\big)^{\frac{\alpha_2}{\alpha'}}}{\sum_{i=1}^{D} \big(p_i^{(1)}\big)^{\frac{\alpha_1}{\alpha'}} \big(p_i^{(2)}\big)^{\frac{\alpha_2}{\alpha'}}}, \quad j = 1, \ldots, D.$$

Clearly, $\sum_{j=1}^{D} r_j = 1$. Now we have $K - 1$ distributions: $r, p^{(3)}, \ldots, p^{(K)}$, with weights $\alpha', \alpha_3, \ldots, \alpha_K$ summing to 1.

By the inductive hypothesis,

$$\sum_{j=1}^{D} r_j^{\alpha'} \big(p_j^{(3)}\big)^{\alpha_3} \cdots \big(p_j^{(K)}\big)^{\alpha_K} \;\;\leq\;\; 1.$$

Substituting the definition of r_j into the left-hand side shows

$$\sum_{j=1}^{D} (p_j^{(1)})^{\alpha_1} (p_j^{(2)})^{\alpha_2} \prod_{k=3}^{K} (p_j^{(k)})^{\alpha_k} \leq \left(\sum_{i=1}^{D} (p_i^{(1)})^{\alpha_1} (p_i^{(2)})^{\alpha_2} \right)^{\alpha'}. \quad (11)$$

Since the base case says $\sum_{i=1}^{D} (p_i^{(1)})^{\alpha_1} (p_i^{(2)})^{\alpha_2} \leq 1$, raising it to the power α' also yields a value at most 1. Thus

$$\sum_{j=1}^{D} \prod_{k=1}^{K} \big(p_j^{(k)}\big)^{\alpha_k} \;\;\leq\;\; 1,$$

which completes the inductive step.

Equality Condition. In base case, equality demands $p = q$ or $\alpha = 0$ or $\alpha = 1$. Propagating this requirement through each step of the recursion shows all distributions must be identical or all but one weight are zero.

By induction, the proof is complete. □

A.3 Implementation details

During pretraining, we employ the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 5e-4, a batch size of 128, and 30,000 training iterations. For optimization with LBFGS (Liu and Nocedal,

1989), we use a tolerance of 1e-5 for both the gradient and parameter changes, setting the initial learning rate to 0.5. All experiments are conducted on a single A100 GPU with 80GB. For all our implementations, we use PyTorch(Paszke et al., 2019) which is released under BSD 3-Clause license.

A.4 Additional Discussions

A.4.1 Naive Bayes with Lidstone Smoothing

We consider the whole family of Naive Bayes with Lidstone Smoothing, where the token probability is

$$\hat{p}(x) = \frac{c(x) + \alpha}{N + \alpha|V|}, \quad \alpha > 0,$$

with $c(x)$ the token count, N the total count, and $|V|$ the vocabulary size. This estimator can be written as an interpolation between the empirical Maximum Likelihood Estimation and a uniform distribution:

$$\hat{p}(x) = \frac{N}{N + \alpha|V|} \frac{c(x)}{N} + \frac{\alpha|V|}{N + \alpha|V|} \frac{1}{|V|}.$$

When $\alpha = 0$, it recovers the unsmoothed Maximum Likelihood Estimation. When $\alpha = \frac{1}{2}$, it corresponds to the Krichevsky–Trofimov (KT) estimator. When $\alpha = 1$, it represents Naive Bayes with Laplace smoothing.

We sweep α and report compression on Enwik8. KT (α=0.5) is slightly better than Laplace (α=1), but smaller α is not uniformly better: as α decreases, the estimator adheres more closely to training counts and can become less robust under sparsity or domain shift. Since α behaves like a dataset-dependent hyperparameter and our goal is simplicity, we fix α=1 in the main experiments.

Table 5: Compression rates on datasets `enwik8`, as α are changed from 1.0 to 0.01 and context length changed from 1 to 2048.

Context length	$\alpha = 1.0$ **(Laplace)**	$\alpha = 0.5$ **(KT)**	$\alpha = 0.1$	$\alpha = 0.01$
1	99.38%	99.12%	98.38%	101.50%
2	99.12%	98.62%	97.88%	103.75%
4	97.88%	96.88%	94.75%	101.88%
8	95.88%	93.75%	89.50%	96.62%
16	92.12%	88.75%	82.12%	87.25%
32	87.12%	82.62%	74.88%	78.00%
64	81.25%	76.12%	69.00%	70.75%
128	75.50%	70.75%	65.50%	66.50%
256	70.75%	67.12%	63.62%	64.50%
512	67.75%	65.25%	63.00%	63.62%
1 024	66.38%	64.38%	62.75%	63.38%
2 048	66.00%	64.25%	62.75%	63.25%

A.4.2 Beyond a Single Global α

A natural question is whether a single global weight α is too restrictive when the data distribution shifts in complex, context-dependent ways. We explored a dynamic alternative by predicting a per-token α with a small MLP placed on top of the frozen backbone's latent representation (the base model and expert remain frozen; only the MLP is trained). This delivered consistent and modest gains over a single global α.

Table 6: Compression performance comparison between fixed and dynamic weight approaches.

Model	**math**	**code**	**shakespeare**	**enwik8***
Transformer 200K	56.25%	65.67%	44.04%	31.59%
Transformer 200K + single α	50.95%	53.94%	42.12%	31.58%
Transformer 200K + per-token α	**49.23%**	**51.39%**	**41.56%**	**31.54%**

Although a context-aware α could offer finer control, our choice of a single scalar reflects a deliberate trade-off: while it has lower capacity than full finetuning, it is appealing in resource-constrained and real-time settings (e.g., limited-memory GPUs or quantized models that cannot be finetuned). Moreover, much of the expressive power in our approach comes from the additional expert itself, which offsets the simplicity of a global α.