BackSlash: Rate Constrained Optimized Training of Large Language Models

Jun Wu, Jiangtao Wen*, Yuxing Han*

New York University (project lead), Tsinghua University

2025.06

1. Background

Background

Challenges in the Development of Large Language Models

Large Language Models (LLMs) have been widely adopted due to their powerful learning and generalization capabilities. Their parameter scale has experienced **rapid growth** in recent years, which poses significant **challenges** for storage, distribution, and inference.

Limitations of Model Compression

To reduce the computational and storage costs of Large Language Models (LLMs), various model compression techniques — such as quantization, pruning, distillation, and low-rank decomposition — have been proposed in recent years. However, most of these methods follow a "**train-first-then-compress**" paradigm, separating the training process from the compression process. In contrast, parameter **compression during training** has received relatively little attention.

Model name	Time	Parameter size	Growth rate
GPT-1	2018.06	117M	-
GPT-2	2019.02	1.5B	12.8x
GPT-3	2020.06	175B	116.7x
GPT-4	2024.11	1.8T	1200.0x

Table 1. Parameter scale and growth rate of GPTs as an example over recent years.

2. Methodology

Generalized Gaussion Prior

➢ Gaussion Distribution Model

Most research has assumed that model parameters follow a Gaussian distribution during the initialization phase. He initialization and Xavier initialization were proposed based on this assumption. However, relatively little attention has been given to how parameter distributions evolve during the training process.

➢ Generalized Gaussion Distribution Model

Through extensive experiments, we have found that model parameters actually conform more closely to a Generalized Gaussian Distribution (GGD) . Notably, the shape parameter ν in many models is often less than 2. For example, the shape parameters are approximately 1.36 for BERT, 1.54 for GPT, 1.26 for LLaMA, and 0.85 for DeepSeek.

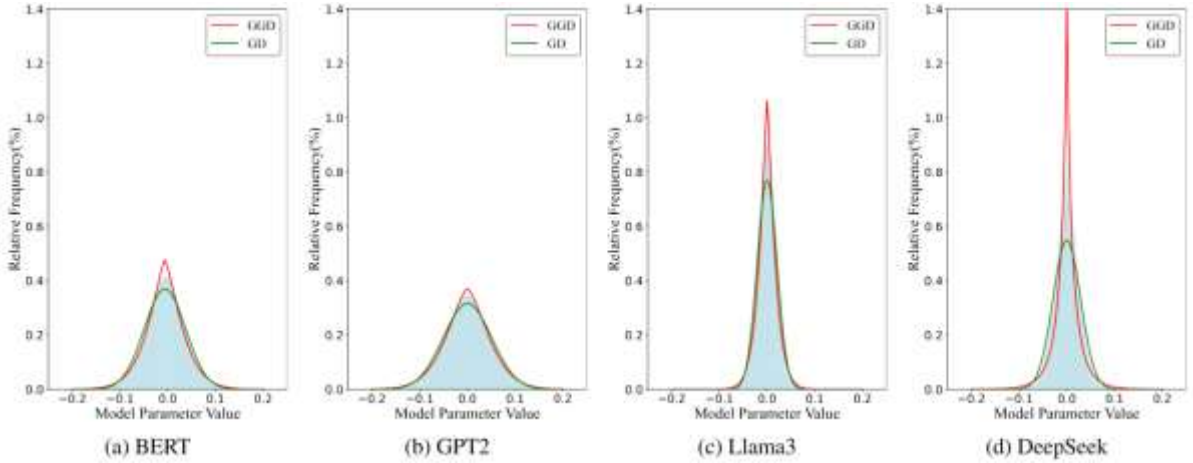

Figure 1. Parameter distributions fitting by generalized Gaussian distribution (GGD) and Gaussian distribution (GD) under different LLMs. GGD fits the boundaries of the parameter distributions better than GD does.

BackSlash Framework

Loss Funtion of BackSlash

The proposed objective function is formulated as:

$$\mathcal{J} = D + \lambda \cdot R,$$

- ✓ $D(\cdot)$ represents the task-specific loss (e.g., cross-entropy for classification),
- ✓ $R(\cdot)$ denotes the parameter information rate measured via average information content (e.g. DGGR as derived below),
- ✓ λ serves as a Lagrange multiplier controlling the intensity of rate constraint.

Discrete Generalized Gaussian Rate (DGGR)

As mentioned on the previous page, the probability of parameter θ_i can be estimated as: $p(\theta_i) = \int_{\theta_i - \frac{\delta}{2}}^{\theta_i + \frac{\delta}{2}} f(x)dx \approx \delta f(\theta_i) = \delta C_1 e^{-C_2|\theta_i|^\nu}$

Then calculate the entropy based on the information quantity of the parameters:

$$\hat{H}(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \log_2 p(\theta_i)$$

$$= \frac{1}{N}\sum_{i=1}^{N} (C_2 \cdot |\theta_i|^\nu - \ln \delta C_1)$$

δ, C_1, C_2, ν are all constants. By ignoring the constant bias and combining the constant coefficient, the entropy regularization can be simplified into the formula below:

$$\hat{H}(\theta) = \frac{\lambda}{\nu}\sum_{i=1}^{N} |\theta_i|^\nu$$

Exponential-Golomb (EG) Code

➢ Weaknesses of Huffman Code

- ✓ Huffman tables designed for different LLMs are different, while a practical implementation may often need to accommodate multiple models in the same system (e.g. on the same chip).
- ✓ Huffman table designed based on empirical distributions usually is not well-structured, leading to more complicated encoder / decoder implementation.
- ✓ We observe the Huffman code can only provide minimal efficiency gains over EG code on BackSlash-trained.

➢ Advantages of EG Code

- ✓ The performance of EG codes is robust with regard to parameter mismatch, and as a result, adaptive coding is not needed when parameters of the quantized GG source change.
- ✓ EG codes contain an infinite number of codewords, and can therefore be used for LLM of any size.
- ✓ EG codes are nicely structured, and allow for highly optimized encoder/decoder.

Parameter (k)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	···
$k=0$	1	010	011	00100	00101	00110	00111	0001000	0001001	0001010	···
$k=1$	10	11	0100	0101	0110	0111	001000	001001	001010	001011	···
$k=2$	100	101	110	111	01000	01001	01010	01011	01100	01101	···
$k=3$	1000	1001	1010	1011	1100	1101	1110	1111	010000	010001	···
$k=4$	10000	10001	10010	10011	10100	10101	10110	10111	11000	11001	···
$k=5$	100000	100001	100010	100011	100100	100101	100110	100111	101000	101001	···

Table 2. The Structure of exp-Golomb code with different parameter k which is from 0 to 5 as an example. In general, EG codes with a smaller parameter k encode better for GG sources with low shape parameters.

3. Experiment

Generalization Analysis

Model Architecture

In the top-right figure, we applied BackSlash for text classification tasks on various well-known models. As can be seen, BackSlash achieves excellent compression performance across different entropy coding schemes in all types of tasks. Specifically, Gemma achieved compression rates of 89% and 90% under EG coding and Huffman coding, respectively.

Model	Param Size	Method	FL (bits)	EG (bits)	HM (bits)	EG Compress	HM Compress	Accuracy
BERT	110M	-	10.00	7.31	5.47	27%	45%	**93.63%**
		BackSlash	10.00	2.64	2.42	**74%**	**76%**	92.59%
GPT	774M	-	11.00	7.78	5.73	29%	48%	85.92%
		BackSlash	11.00	2.46	2.25	**78%**	**80%**	**88.73%**
Llama	1B	-	10.00	5.49	4.43	45%	56%	86.09%
		BackSlash	10.00	1.72	1.66	**83%**	**83%**	**86.93%**
Gemma	2B	-	11.00	4.45	3.95	60%	64%	**86.95%**
		BackSlash	11.00	1.16	1.15	**89%**	**90%**	85.86%

Table 3. Compression performance of BackSlash with different model architectures and parameter scales.

Learning Tasks

In the bottom-right figure, we conducted classification tasks on BERT and text generation tasks on DeepSeek-7B. As can be seen, BackSlash is task-agnostic; it achieves good compression performance regardless of whether the task is classification or generation, without affecting the model's predictive performance.

Task	Dataset	Method	FL (bits)	EG (bits)	HM (bits)	EG Compress	HM Compress	Accuracy
Sentiment	IMDB	-	10.00	7.31	5.47	27%	45%	**93.63%**
		BackSlash	10.00	2.64	2.42	**74%**	**76%**	92.59%
Spam	Enron-Spam	-	10.00	7.31	5.47	27%	45%	**99.65%**
		BackSlash	10.00	2.42	2.19	**76%**	**78%**	98.96%
Topic	20 Newsgroups	-	10.00	7.31	5.47	27%	45%	**70.78%**
		BackSlash	10.00	3.61	3.18	**64%**	**68%**	69.36%
Q-A	SQuAD	-	11.00	5.95	4.70	46%	57%	99.97%
		BackSlash	11.00	2.90	2.81	**74%**	**74%**	**99.97%**
Translation	WMT-19	-	11.00	6.10	4.70	45%	57%	**99.96%**
		BackSlash	11.00	3.10	3.00	**72%**	**73%**	99.95%

Table 4. Compression performance of BackSlash under different deep learning tasks.

Quantization and Pruning

Quantization

In the top-right figure, we quantized both the BackSlash model and the conventionally trained model using different quantization step sizes. As can be seen, quantization does not introduce additional negative effects on the BackSlash model. This is because quantization equally degrades the precision of parameters, so the curves for both the BackSlash model and the conventionally trained model exhibit nearly overlapping trends.

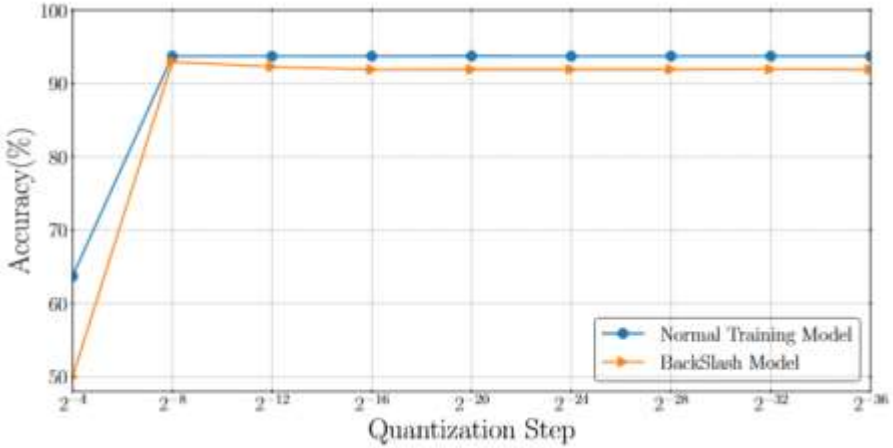

Figure 2. Quantization using different quantization steps for BackSlash model and normal training model.

Pruning

In the bottom-right figure, we pruned both the BackSlash model and the conventionally trained model using different pruning rates. As can be seen, the pruning performance of the BackSlash model is better. The conventionally trained model begins to experience a decline in accuracy when the pruning rate reaches 50%, while the BackSlash model maintains its original predictive capability even at an 80% pruning rate. This indicates that BackSlash works well in conjunction with pruning.

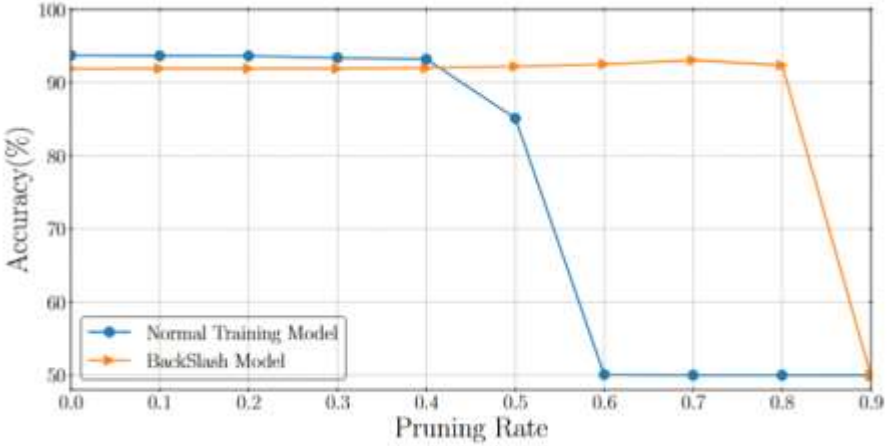

Figure 3. Pruning using different pruning rates for BackSlash model and normal training model.

4. Conclusion

Conclusion

Summary

- ✓ **Generalized Gaussian Prior:** Through extensive analysis of parameter distributions in current large language models, we observed that they are better described by the family of generalized Gaussian distributions.
- ✓ **Training-Compression Integration:** BackSlash merges model optimization and compression techniques during training to produce compact, high-performance models ready for deployment.
- ✓ **Efficient and Hardware-Friendly:** BackSlash produces smaller models that are more efficient to compute and transmit, and better compatible with hardware deployment via pruning.

Impact

Instead of using standard backpropagation to train a large model and compressing it afterward, our BackSlash framework integrates efficiency directly into the training process to produce small and easy-to-deploy models. This framework can significantly influence how the next-generation foundation models are trained and deployed, both in software and hardware.

Thank you for Listening!

Jun Wu, Jiangtao Wen*, Yuxing Han*

New York University (lead), Tsinghua University

2025.06