BackSlash: Rate Constrained Optimized Training of Large Language Models

Jun Wu [1] **Jiangtao Wen** [2] **Yuxing Han** [1]

Abstract

The rapid advancement of large-language models (LLMs) has driven extensive research into parameter compression after training has been completed, yet compression during the training phase remains largely unexplored. In this work, we introduce Rate-Constrained Training (BackSlash), a novel training-time compression approach based on rate-distortion optimization (RDO). BackSlash enables a flexible trade-off between model accuracy and complexity, significantly reducing parameter redundancy while preserving performance. Experiments in various architectures and tasks demonstrate that BackSlash can reduce memory usage by 60% - 90% without accuracy loss and provides significant compression gain compared to compression after training. Moreover, BackSlash proves to be highly versatile: it enhances generalization with small Lagrange multipliers, improves model robustness to pruning (maintaining accuracy even at 80% pruning rates), and enables network simplification for accelerated inference on edge devices.

1. Introduction

As the foundation of modern artificial intelligence, generative large language models (LLMs) such as Llama (Touvron et al., 2023), GPT (Brown et al., 2020), and Qwen (Bai et al., 2023) exhibit remarkable self-learning and non-linear modeling capabilities. With continuous advancements in deep learning, the parameter scales of LLMs have grown at an unprecedented rate, as shown in Table 1. Looking ahead, it is expected that the parameter scale of neural networks will continue to expand rapidly, driving further progress in AI development.

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

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

The ever-increasing size of LLM parameters leads to increased computational costs, inference latency, and network distribution overhead during model deployment. To enable efficient inference of LLMs on edge devices, extensive research has focused on model compression using techniques such as parameter quantization, model pruning, and low-rank matrix decomposition. However, while there have been extensive studies on LLM redundancy at the microscopic level, such as precision and structural inefficiencies, the overall parameter distribution has received little attention. In addition, most existing compression techniques are applied after training, as opposed to being integrated into the LLM parameter training process to proactively achieve optimized trade-offs between parameter precision, model size, and model performance. Finally, existing studies assume that model parameters follow the Gaussian distribution and, therefore employ Huffman coding designed using empirical statistics for compression. Both the probabilistic model and the entropy coding technique have room for improvement.

In this paper, we introduce a rate-constrained optimized approach to LLM training. By incorporating model parameter size into the training process through a rate (R) and distortion (D) joint optimization, the proposed rate-constrained training (BackSlash) approach is capable of producing the optimal performing model for a given parameter set size, that is, producing the best-fit model given by its end application (hardware constrained parameter set size).

The main contributions of this paper are as follows:

1. Instead of the widely used Gaussian model for LLM parameter distribution, we found through extensive experiments that the generalized Gaussian (GG) distribution with the shape parameter less than 2 is a better

[1]Shenzhen International Graduate School, Tsinghua University [2]Computer Science, New York University. Correspondence to: Yuxing Han <yuxinghan@sz.tsinghua.edu.cn>, Jiangtao Wen (project lead) <jw9263@nyu.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

model.

2. We propose to use exp-Golomb (EG) codes for entropy coding of LLM parameters, whose distribution can be well-modeled by GG distributions. It has been shown (Wen & Villasenor, 1999) that for GG sources, EG codes can achieve coding efficiency very close to the entropy limit, well over 90% in many cases. We also find the optimal EG code with k=0 implementation can accommodate many applications.

3. Based on the GG distribution observation and using EG codes as entropy codes, we proposed a discretized generalized Gaussian information rate (DGGR) to measure the model information rate and an BackSlash algorithm that jointly optimizes the information rate and performance during the training phase of LLMs. Experiments with different LLMs and different deep-learning tasks show significant savings in model size as compared with both unconstrained training and unconstrained training followed by entropy coding.

2. Related Work

2.1. LLMs Compression

To achieve low-cost distribution, deployment, and inference, many compression strategies for LLMs have been proposed.

Pruning reduces computational and storage overheads by removing unimportant weights or neurons from the model. Unstructured pruning achieves compression by removing redundant connections, e.g., Han et al. (2015b) and Han et al. (2015a) proposed a pruning method based on weight paradigms. Because unstructured pruning may lead to irregular network structures, structured pruning of filters or channels was proposed. Li et al. (2016) proposed pruning based on filter, while Luo et al. (2017) proposed Thinet that minimized the reconstruction error. Hardware constraints (He et al., 2018; Wang et al., 2018a) such as energy consumption and delay were also introduced into the pruning process to optimize the model performance in resource-constrained environments. Prune continues to be an important direction for model optimization, as evidenced by recent publications such as Dynamic Structure Pruning (Park et al., 2023), LAPP (Zhai et al., 2023), and Turbo-VBI (Xia et al., 2023a).

Quantization can speed up training and inference by reducing the precision of weights and activation values. Binary weights (Courbariaux et al., 2015; Rastegari et al., 2016), triple weights (Li & Liu, 2016; Zhu et al., 2016), cluster quantization (Gong et al., 2014; Choi et al., 2016), and mixed bit-width quantization (Zhou et al., 2016; Wang et al., 2018b) were examples of quantization techniques. Long et al. (2020) used shift operation to replace the costly full-precision operation by quantizing low-bit weights and activations. Liu et al. (2021) simultaneously maintained the representational power of non-uniform quantization and the efficiency of uniform quantization.

Low-rank decomposition (Jaderberg et al., 2014; Masana et al., 2017), parameter sharing (Wang et al., 2017; Kossaifi et al., 2019), and knowledge distillation (Xu et al., 2018; Chen et al., 2017) have demonstrated significant effectiveness in applications. Additionally, with large-scale distributed deep learning training systems, communication-efficient gradient compression techniques were proposed (Lin et al., 2017). These approaches collectively enhance the efficiency of deep learning model training and deployment.

2.2. Rate Distortion Optimization

Information theory (Shannon, 1948; Cover, 1999) mathematically quantified the efficiency with which information can be transmitted, stored, and processed, where rate-distortion function defines the minimal distortion that can be achieved while entropy coding a system to a given bitrate (Davisson, 1972; Berger, 2003).

In practical applications, rate-distortion optimization (RDO) has found extensive adoption in video coding (Luttrell et al., 2000; Itu-T & Jtc, 2010; Wien, 2015; Brand et al., 2022; Chen et al., 2023; Guo et al., 2023; Chiang et al., 2023; Xia et al., 2023b; Zhang et al., 2024), etc. are also continuing to deepen the application of RDO in video and images.

In recent years, RDO has also been introduced for the compression of neural networks. For example, Gao et al. (2018) investigated the fundamental limits of model compression and proposed a compression framework for pruning, quantization, and other techniques. Isik et al. (2021) proposes a new pruning strategy based on RDO to approach the compression limits of neural networks. In both cases, RDO is applied after models have been trained to further pruning and quantization, as opposed to being integral to the training process itself.

3. Generalized Gaussian Model of LLM Parameters

Most research assumes that LLM parameters follow the Gaussian distribution in the initialization and rarely discussed the distribution after training. For example, Gaussian distribution was used by both Xavier and He for random parameter initialization (Glorot & Bengio, 2010; He et al., 2015). However, through extensive experiments, we found that the more broad generalized Gaussian (GG) distribution family with shape parameter less than 2 might be a better model for LLM models, especially considering that different regulations during training may impact parameter distribution. The distribution usually also changes during training as the model converges. In practice, the parameter

distribution tends to develop heavier tails during the training process(Fortuin et al., 2021).

Mathematically, the probability density function (pdf) of generalized Gaussian distribution is defined as

$$f(x) = C_1 e^{-C_2|x|^\nu} \tag{1}$$

where

$$C_1 = \frac{\nu\gamma}{2\Gamma(1/\nu)}, C_2 = \gamma^\nu, \tag{2}$$
$$\gamma = \frac{1}{\sigma}\sqrt{\frac{\Gamma(3/\nu)}{\Gamma(1/\nu)}},$$
$$\Gamma(\alpha) = \int_0^\infty t^{\alpha-1}e^{-t}dt,$$

and $\alpha > 0$.

It is easy to see that when $\nu = 1$, the GG distribution is the Laplacian distribution, while when $\nu = 2$, it is the Gaussian distribution. Varying the shape parameter of GG distribution allows for better match between the probabilistic model to better match LLM while using the same mathematical formulation.

The GG distribution in (1) is a continuous distribution, while in reality, LLM parameters all have fixed-length and limited precision. Therefore, we treat LLM parameters as a quantized GG distribution. Assuming the quantization step size of the parameters is δ, the probability of a parameter θ_i is

$$p(\theta_i) = \int_{\theta_i}^{\theta_i+\delta} f(x)dx. \tag{3}$$

As δ is typically small, we approximate

$$p(\theta_i) \approx \delta f(\theta_i) = \delta C_1 e^{-C_2|\theta_i|^\nu}. \tag{4}$$

The validity of this assumption could be verified with existing LLMs. For instance, BERT-base (110M) can be well-modeled by a GG with a shape parameter of 1.36, the shape parameter for GPT2 (774M) is 1.54, or 1.26 for Llama3 (1B), or 0.85 for DeepSeek (7B), and their distributions are shown in Fig. 1. These shape parameter values are, although different, all smaller than 2. The corresponding pdfs show higher peaks and longer tails than the Gaussian distribution.

Denote the size of LLM as N_p, the mean of the information content of the parameters can be calculated as $R(\theta) = -\frac{1}{N_p}\sum_{i=1}^{N_p}\log_2 p(\theta_i)$. Neglecting constant factors and terms, we define discretized generalized Gaussian rate (DGGR) as follows

$$R(\theta) = \frac{1}{N_p}\sum_{i=1}^{N_p}|\theta_i|^\nu, \tag{5}$$

and use $R(\theta)$ as a measure of the information complexity of the model.

4. Rate-Constrained Training (BackSlash) of LLMs

4.1. Overview and loss function definition

In contrast to traditional non-constrained training, the target loss function for optimization in BackSlash considers both model performance and model size is

$$\mathcal{J} = D + \lambda \cdot R, \tag{6}$$

where D denotes the distortion of the fitting data, i.e. the deviation between the model predictions and the ground truth. R denotes the rate of the model parameters, indicating the complexity of the model itself. λ is the Lagrange multiplier. The selection of distortion D varies depending on the deep-learning task. For example, we often use the categorical cross-entropy loss function in classification problems and the mean squared error loss in regression tasks. Methods like KL divergence or logarithmic loss are also utilized in some specific tasks. We can choose the most suitable empirical loss function for specific tasks, which does not affect the BackSlash results.

The rate R is expressed by the average information content of the parameters, defined using DGGR. Combining (5) and (6) we get

$$\mathcal{J} = \mathcal{L}(X, Y, f, \theta) + \lambda \cdot \frac{1}{N_p}\sum_{i=1}^{N_p}|\theta_i|^\nu, \tag{7}$$

where X and Y denote the training ground truth, f and θ denote the forward propagation function and parameter set of the neural network.

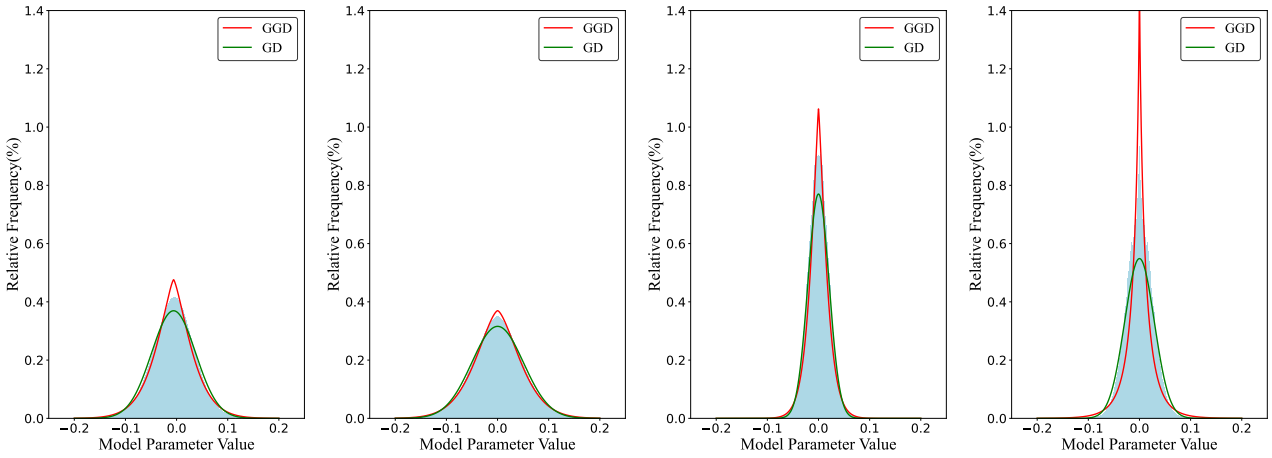
The shape parameter ν of DGGR is not a constant during training and needs to be dynamically estimated before each batch of gradient descent. A well-known method (Sharifi & Leon-Garcia, 1995) for estimating the shape parameter is by introducing a comparison function:

$$\rho(\nu) = \frac{\Gamma(1/\nu)\cdot\Gamma(3/\nu)}{\Gamma^2(2/\nu)} = \frac{\mathbb{E}[\theta^2]}{\mathbb{E}^2[|\theta|]}. \tag{8}$$

Specifically, the estimation process of ν can be organized as follows:

1. obtain the model parameters θ and compute $\mathbb{E}[\theta^2]$ and $\mathbb{E}^2[|\theta|]$,
2. estimate $\rho(\nu)$ by (8) based on $\mathbb{E}[\theta^2]$ and $\mathbb{E}^2[|\theta|]$,
3. Find ν using $\rho(\nu)$.

Additionally, notice that when $0 < \nu < 1$ and $\theta \to 0$, $\nabla R(\theta) \to \infty$. This causes severe oscillations in the parameters during gradient descent, which prevents the model



(a) BERT (b) GPT2 (c) Llama3 (d) 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.

from converging. To address this, we introduce a trick to optimize the gradient descent of $R(\theta)$ by adding a constant ϵ ($\epsilon > 0$) to control the gradient size and modify the information rate formula as $R(\theta) = \frac{1}{N_p}\sum_{i=1}^{N_p}(|\theta_i| + \epsilon)^\nu$. We refer to this method of gradient suppression as **soft gradient clipping**.

4.2. BackSlash algorithm description

The overall BackSlash algorithmic can be summarized as follows:

Algorithm 1 Rate-Constrained Training (BackSlash)

Require: Model f, learning rate η, loss function $\mathcal{L}$, Lagrange multiplier λ, and clipping coefficient ϵ.
for each epoch τ **do**
  for each batch (x_i, y_i) **do**
    Retrieve all model parameters θ.
    Estimate comparison function $\rho(\nu)$: $\rho(\nu) \leftarrow \frac{\mathbb{E}[\theta^2]}{\mathbb{E}^2[|\theta|]}$
    Find shape parameter ν using $\rho(\nu)$.
    Forward propagation and calculate RD Cost $\mathcal{J}$: $\mathcal{J} \leftarrow \mathcal{L}(x_i, y_i, f, \theta) + \lambda \cdot \frac{1}{N_p}\sum_{i=1}^{N_p}(|\theta_i| + \epsilon)^\nu$
    Backward propagation and optimize parameters: $\theta \leftarrow \theta - \eta \cdot (\frac{\partial \mathcal{L}}{\partial \theta} + \lambda \cdot \frac{\nu\theta}{N_p|\theta|}(|\theta| + \epsilon)^{\nu-1})$.
  end for
end for
Until convergence or max iterations.

It should be noted that if we set the shape parameter ν in DGGR to be $\nu = 1$ and $\nu = 2$, we find that DGGR degenerates into L_1 (Fitriani et al., 2022) regularization ($\frac{1}{N_p}\sum_{i=1}^{N_p}|\theta_i|$) and L_2 (Hoerl & Kennard, 1970) regularization ($\frac{1}{N_p}\sum_{i=1}^{N_p}|\theta_i|^2$) respectively. This means, that L_1 and L_2 regularization are special cases of DGGR when the model parameter distribution is the Laplace and the Gaussian distributions respectively.

4.3. Entropy coding of LLM using EG codes

The complexity constraint in BackSlash is defined using DGGR. In practice, the parameters will have to be entropy coded using practical entropy codes, whose rate can not exactly match the DGGR.

Due to its simplicity, Huffman codes have been used for entropy coding of LLM parameters. For example, in Han et al. (2015a) achieved 20%-30% size reduction using Huffman coding. However, using Huffman coding in BackSlash or compression of LLM in general has several drawbacks.

First of all, Huffman code tables are designed using explicit distributions calculated from LLM parameters. The mismatch between the distribution of the parameters and the distribution for which the Huffman code is designed may lead to severe coding efficiency loss. On the other hand, the large parameter size and non-parallelizable table building process of Huffman code may bring prohibitively high complexity to BackSlash. This is also why we used the theoretical DGGR as opposed to coded bits in the loss function.

Secondly, 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.

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.

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	···

on the same chip).

Thirdly, the Huffman table designed based on empirical distributions usually is not well-structured, leading to more complicated encoder/decoder implementation.

Fourthly, we observe the Huffman code can only provide minimal efficiency gains over EG code on BackSlash-trained models in all subsequent experiments.

As noted in Section 3, LLM parameters can be modeled well by quantized GG distributions with shape parameters less than 2. In Wen & Villasenor (1999), Wen and Villasenor studied and proposed using exp-Golomb (EG) codes to entropy coding quantized GG sources. The structure of EG code is shown in Table 2. The advantages of EG codes can be summarized as the following:

1. the efficiency of EG codes is consistently within a few percentage points of the entropy limit and almost identical to the Huffman code specifically designed for each quantized GG source,
2. 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,
3. EG codes contain an infinite number of codewords, and can therefore be used for LLM of any size,
4. EG codes are nicely structured, and allow for highly optimized encoder/decoder.

Therefore, in our experiments, we used EG codes for the actual entropy coding rate (as opposed to the theoretical rate of DGGR in BackSlash) for both entropy coding of parameters after traditional, unconstrained LLM training, and in comparison, BackSlash. In our experiment, we tested EG with different parameters on several models with BackSlash which is shown in Table 3, and the EG code that we found was optimal was EG with parameter $k = 0$.

Table 3. Average code lengths for several models with different EG parameters. With the EG parameter increasing, the average code length of the model parameters also increases and the EG code gradually converges to the fixed-length code.

Model	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$
BERT	**2.64**	3.16	3.74	4.42	5.17
GPT-2	**2.46**	3.05	3.70	4.40	5.19
Llama-3	**1.72**	2.47	3.26	4.12	5.03
Gemma-2	**1.16**	2.10	3.06	4.02	5.01

In addition, we note that after BackSlash, most model parameters are zero, while the non-zero values are extremely sparse, usually accounting for few percent of all possible values. For example, the number of code words occupied by BERT with BackSlash after quantization with a step of 2^{-8} is 2695 but only 641 are actually used. And except for a small number of quantized parameters near 0, the others are highly disordered. For example, the quantized parameter -177 of BERT is only ranked 609 by frequency but actually it takes up 142nd. Therefore, prior to entropy coding of model parameters, we first rank the number of occurrences of all parameter values and map each value that model parameters might actually take to an index. The index, instead of the parameter value, is then entropy coded. This process can be formally summarized as follows:

Algorithm 2 Parameter Entropy Encoding

Require: Model parameter set Θ, quantization step size 2^{-n}, encoding strategy Enc.
Quantize the parameters: $Q \leftarrow \text{round}(2^n \cdot \Theta)$
Sort by frequency, build the code table by quantized parameter and sorted index: $\mathcal{C} \leftarrow \{(q_i, c_i) \mid q_i \in Q_s, c_i \in C_s\}$
Map quantized parameters to codewords and encode it to bitstream: $B_{stream} \leftarrow \text{Enc}(\mathcal{C}[Q])$
Output: Bitstream B_{stream} and code table $\mathcal{C}$

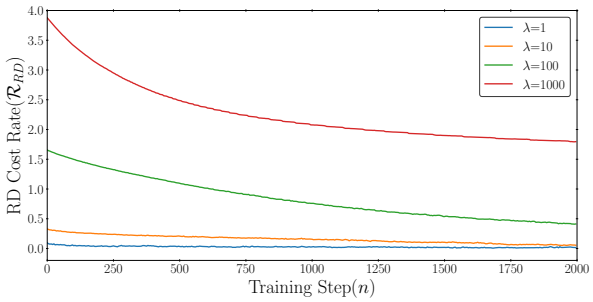


Figure 2. RD cost rate changes in training with different Lagrange multiplier (λ).

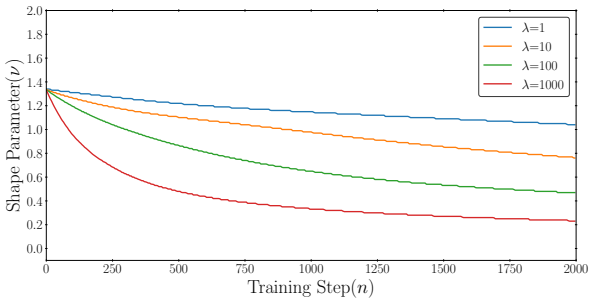


Figure 3. Shape parameter changes in training with different Lagrange multiplier (λ).

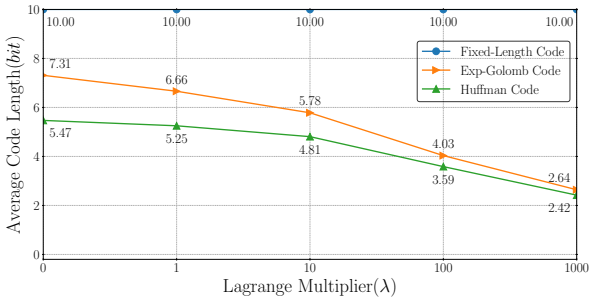


Figure 4. Impact of Lagrange multipliers on average code length of various encoding algorithms.

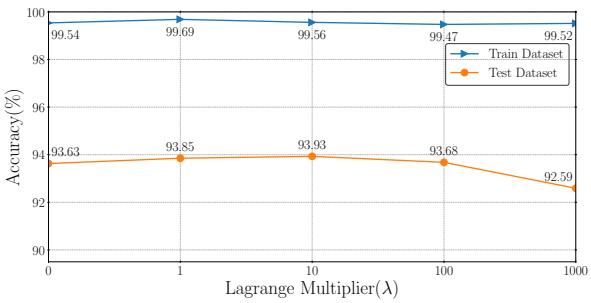


Figure 5. Impact of Lagrange multipliers on accuracy of test and train Dataset.

The distribution of indices still follows a generalized Gaussian, though the value mapping induces slight deviations compared to the original parameters. For example, the shape of parameters and index of BERT under normal training is 1.36 and 1.47, while under BackSlash they become 0.26 and 0.30, respectively. Nevertheless, these minor shape variations negligibly impact entropy coding efficiency, as EG codes maintain robustness across the entire family of generalized Gaussian sources.

The sparsity of the values model parameters is the reason that in BackSlash, we use DGGR, as opposed to EG code length directly in BackSlash - even though the same EG code may be used throughout BackSlash, the parameter value that the codeword is mapped to changes.

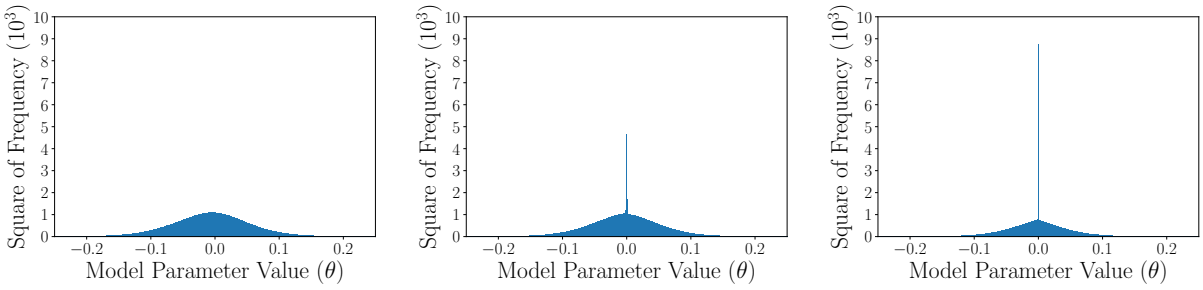
The mapping (termed "Value Mapping") between the quantized parameter set Q_s and the codeword set C_s is defined as $\mathcal{C} = \{(q_i, c_i) \mid q_i \in Q_s, c_i \in C_s\}$.

5. Experiments

We perform various classification tasks on popular LLMs including BERT, GPT, Llama, and Gemma to evaluate the performances of BackSlash by classification accuracy, and generation tasks on DeepSeek evaluated by next token accuracy. We mainly use classification tasks on BERT to analyze the effects of BackSlash when it is trained and deployed, as classification accuracy is one of the most intuitive quantitative metrics of model performance and BERT model has a better performance on classification tasks. In addition, we examined the entropy coding efficiency of EG codes as compared with Huffman (HM) coding and fixed-length (FL) coding with value mapping for all EG, HM and FL codes.

5.1. Performance

Taking the sentiment analysis task of the BERT model as an example, we tested BackSlash using different Lagrange multiplier λ settings. Fig. 2 shows how the loss changes in training under different Lagrange multipliers. As RD Costs may vary significantly with different Lagrange multipliers, for visual clarity, we used ($\mathcal{R}_{RD} = \log_{10}(\mathcal{J} - \beta\mathcal{J}_{min})$, $\beta = 0.995$) as the y-axis. As can be seen from the figure, the larger the Lagrange multiplier, the steeper the curve, reflecting the fact that the Lagrange multiplier controls the training speed of the BackSlash.



(a) Lagrange Multiplier 0 (b) Lagrange Multiplier 10 (c) Lagrange Multiplier 1000

Figure 6. Parameters distribution under different Lagrange multiplier training. With the Lagrange multipliers increasing, the parameter distributions become more concentrated and have higher peaks and lower tails.

Table 4. Compression performance of BackSlash with different model architectures and parameter scales.

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%

Fig. 3 illustrates how the shape parameter of the parameter distribution varies during training. For different λ values, the shape parameter was set to an identical initial value but converged to different values, reflecting how λ in the BackSlash led to different model distributions.

In Fig. 4, after quantizing the model parameters with the quantization step 2^{-8}, we use EG code, HM code and FL code to encode the model parameters and compute the average code length respectively. When applying EG coding and HM coding after unconstrained training, the model size was compressed to 73% and 55% of the size of FL coding, corresponding to 27% and 45% saving. Whereas BackSlash with EG coding and λ of 1000, reduced the model size to 26% and 24%. Even though Huffman coding leads to a very small gain in coding efficiency, a different Huffman table and the corresponding encoder/decoder will have to be designed and implemented for each LLM. In contrast, the same EG table could be used across models and sizes (e.g. DeepSeek 7B and 170B).

Fig. 5 demonstrates the effect of BackSlash on model accuracy. We can find that BackSlash with reasonable λ did not have a significant effect on accuracy. For the model with $\lambda = 1000$, model performance decreased by only 0.02% on the training set and 1.90% on the test set as compared with normal training (i.e. $\lambda = 0$). It was observed that model accuracy was not monotonic with regard to λ, i.e. there is an optimal λ value, the setting of which is a topic under investigation.

Fig. 6 shows the impact of λ on model parameter distribution. As can be seen clearly, as λ increases, i.e. if we give more weights to the rate in the accuracy-rate trade-off, the model trained by BackSlash would become more sparse.

In the current study, the setting of λ was still through trials-and-errors. For example, when we set up a set of values for Λ and train BERT model using BackSlash until convergence, we found that the model trained with $\lambda = 2000$ achieved the best overall trade-off with 2.52% loss in accuracy and only 13% of the size. Moreover, in our extensive experiments, it consistently achieves similar and remarkable effectiveness across various models and tasks.

5.2. Generalization Analysis

Model architectures and training tasks are of great significance to both the process and the performance of the model and tend to affect the final model obtained from training

Table 5. Compression performance of BackSlash under different deep learning tasks.

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%

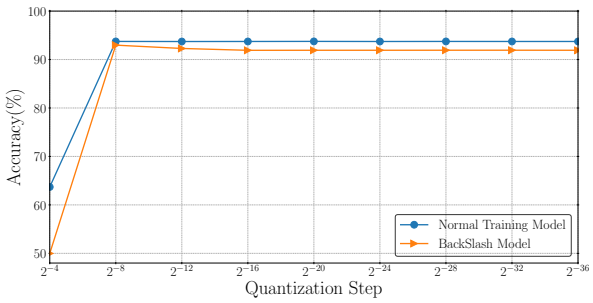


Figure 7. Quantization using different quantization steps for BackSlash model and normal training model.

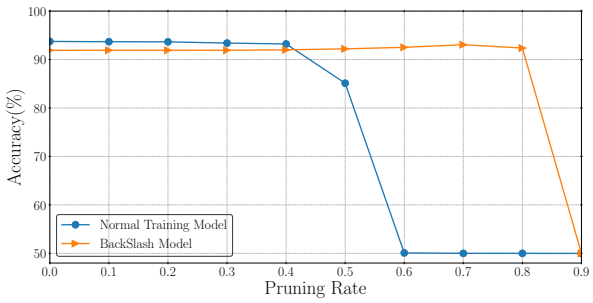


Figure 8. Pruning using different pruning rates for BackSlash model and normal training model.

heavily. It is worth discussing whether BackSlash has the same effects in other models and tasks besides the sentiment analysis of BERT.

In Table 4, we perform the sentiment analysis task on BERT, GPT, Llama, and Gemma under normal training and BackSlash, respectively. These models are chosen because the differences in structure and parameter size among them are large enough to reflect the wide utility of BackSlash. Although different model structures introduce some variability in the results, BackSlash performs similarly for parameter compression. For all the models, BackSlash compresses them by more than 75%, with the highest being 90% for Gemma. Such similar performance comes from the insensitivity of BackSlash to network structure and parameter size. In addition, it can be seen that in GPT and Llama, the accuracy of using BackSlash is instead slightly higher than that of normal training, which we analyze as originating from the regularization effect attached to BackSlash.

In Table 5, we perform more classification tasks on the BERT model and generation tasks on DeepSeek model under normal training and BackSlash. The "Sentiment" and "Spam" are both binary-classification tasks and the "Topic" is a 20-class-classification tasks, which are evaluated by classification accuracy. The "Q-A" and "Translation" are both text generation tasks, which are evaluated by next token accuracy. These tasks can achieve satisfactory compression performance without compromising model accuracy. BackSlash achieves approximately 70% compression rate compared to the original size in both classification and generation tasks, demonstrating its strong generalization capability across different task types.

5.3. Deployment

Deployment and inference for edge devices are always the central problem and primary purpose of model compression, and quantization and pruning are the main means to deploy the fine-tuned LLMs in edge devices. Therefore, it is necessary to discuss whether the generalization ability of BackSlash models can be maintained in quantization and pruning.

Table 6. Compression performance of BackSlash under different regularization terms.

DGGG	Convergence Shape	Accuracy	FL (bits)	EG (bits)	HM (bits)	EG Compression	HM Compression
DGGR	0.13	91.18%	10.00	1.37	1.32	86%	87%
$L_{0.5}$	0.22	91.88%	10.00	2.90	2.01	71%	80%
L_1	0.15	90.65%	10.00	1.52	1.46	85%	85%
L_2	0.10	88.29%	10.00	1.16	1.14	88%	89%

Fig. 7 illustrates how the accuracy of the BERT model varies with the quantization steps under normal training and BackSlash. When the quantization step is taken 2^{-4}, the generalization ability of both normal training and BackSlash models is completely destroyed. When the quantization step is not less than 2^{-8}, the accuracy of both models changes very smoothly. Both models show the same trend in quantization. This is because quantization uniformly destroys the accuracy of the parameters, so whether or not to use BackSlash does not have an additional negative impact on the quantization results. Furthermore, We performed the same experiments in GPT, Llama, and Gemma, and they all showed identical results to BERT.

Fig. 8 illustrates how the accuracy of the BERT model varies with the pruning rates under normal training and BackSlash. We can see that the predictive ability of the conventionally trained model has begun to degrade when the pruning ratio reaches 50% and has completely lost its predictive ability when it reaches 60%. Instead, the BackSlash model continually maintains its generalization accuracy when the pruning ratio reaches 80%. This is because BackSlash makes the model's parameter distribution more sparse, which increases the space for pruning. So BackSlash's model is also easier to deploy on edge-end devices through pruning and performs more efficient inference. Furthermore, we also performed pruning on GPT, Llama, and Gemma under BackSlash, and the maximum pruning rates for them to maintain accuracy are all 90% while the normal training models start to lose their effectiveness at pruning rates less than 60%, which is similar to the BERT model.

5.4. Ablation

As discussed in Section 4.2, L_1 and L_2 regularizations are special cases of DGGR assuming model parameters follow a Laplace and Gaussian Distribution, respectively. So whether such shape-specific L_p regularization terms can effectively substitute DGGR in the BackSlash framework warrants further investigation.

We perform the sentiment analysis task on BERT with BackSlash using DGGR, $L_{0.5}$, L_1, L_2 respectively, to evaluate their impacts on model performance and code rate. As shown in Table 6. Shape-specific L_p regularization terms present notable theoretical and practical limitations.

From a theoretical perspective, the $L_{0.5}$, L_1, and L_2 terms implicitly assume that model parameters follow generalized Gaussian distributions with fixed shape parameters of 0.5, 1, and 2, respectively. However, the actual shape parameters of the model converge to 0.22, 0.15, and 0.10, contradicting the fixed-shape hypothesis. In contrast, DGGR's dynamic shape parameter adaptation naturally accommodates the evolving weight distribution throughout the optimization process.

From an effectiveness perspective, L_2 achieves marginally better compression than DGGR but incurs significant accuracy degradation, which indicates its detrimental impact on model performance. L_1 is inferior to DGGR in both code length and accuracy. $L_{0.5}$ demonstrates slightly better accuracy but its code length is more than twice that of DGGR, which shows its weakness in parameters compression. These findings suggest that DGGR's adaptive shape parameter adjustment puts performance and code rate in a better balance.

6. Conclusion and Future Work

We propose BackSlash, a training framework for LLMs that jointly optimizes model size and performance. We found that LLM parameters can be well modeled with quantized GG sources of shape parameters less than 2, and can be entropy coded with extremely high efficiency and robustness using EG codes. Experiments with popular LLMs show that BackSlash was capable of reducing model size by up to 80% with virtually no loss in performance.

Currently, we are conducting more experiments with more LLMs and tasks. The optimal setting of λ is also under investigation, as well as efficient hardware architecture that can take advantage of the increased sparseness of the model in more efficient training and inference.

Impact Statement

This paper introduces a fundamentally new approach to training large models. 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.

References

Bai, J., Bai, S., Yang, S., Wang, S., Tan, S., Wang, P., Lin, J., Zhou, C., and Zhou, J. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. 2023. URL https://api.semanticscholar.org/CorpusID:261101015.

Berger, T. Rate-distortion theory. *Wiley Encyclopedia of Telecommunications*, 2003.

Brand, F., Fischer, K., Kopte, A., Windsheimer, M., and Kaup, A. Rdonet: Rate-distortion optimized learned image compression with variable depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1759–1763, 2022.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., teusz Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020. URL https://api.semanticscholar.org/CorpusID:218971783.

Chen, Y., Wang, N., and Zhang, Z. Darkrank: Accelerating deep metric learning via cross sample similarities transfer. *ArXiv*, abs/1707.01220, 2017. URL https://api.semanticscholar.org/CorpusID:19207026.

Chen, Y., Wang, S., Ip, H., and Kwong, S. Rate distortion optimization with adaptive content modeling for random-access versatile video coding. *Information Sciences*, 645: 119325, 2023.

Chiang, J.-C., Shang, H.-Y., and Qiu, J.-J. Multi-exposure image compression considering rate-distortion optimization in rendered high dynamic range image. *IEEE Open Journal of Signal Processing*, 4:132–147, 2023.

Choi, Y., El-Khamy, M., and Lee, J. Towards the limit of network quantization. *ArXiv*, abs/1612.01543, 2016. URL https://api.semanticscholar.org/CorpusID:17299045.

Courbariaux, M., Bengio, Y., and David, J.-P. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Neural Information Processing Systems*, 2015. URL https://api.semanticscholar.org/CorpusID:1518846.

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

Davisson, L. Rate distortion theory: A mathematical basis for data compression. *IEEE Transactions on Communications*, 20(6):1202–1202, 1972.

Fitriani, S. A., Astuti, Y., and Wulandari, I. R. Least absolute shrinkage and selection operator (lasso) and k-nearest neighbors (k-nn) algorithm analysis based on feature selection for diamond price prediction. In *2021 International Seminar on Machine Learning, Optimization, and Data Science (ISMODE)*, pp. 135–139, 2022. doi: 10.1109/ISMODE53584.2022.9742936.

Fortuin, V., Garriga-Alonso, A., Wenzel, F., Rätsch, G., Turner, R. E., van der Wilk, M., and Aitchison, L. Bayesian neural network priors revisited. *ArXiv*, abs/2102.06571, 2021. URL https://api.semanticscholar.org/CorpusID:231918454.

Gao, W., Wang, C., and Oh, S. Rate distortion for model compression: From theory to practice. In *International Conference on Machine Learning*, 2018. URL https://api.semanticscholar.org/CorpusID:53111003.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Gong, Y., Liu, L., Yang, M., and Bourdev, L. D. Compressing deep convolutional networks using vector quantization. *ArXiv*, abs/1412.6115, 2014. URL https://api.semanticscholar.org/CorpusID:6251653.

Guo, H., Zhu, C., Ye, M., Luo, L., and Yang, X. Pre-encoding based temporal dependent rate–distortion optimization for hevc. *Signal Processing: Image Communication*, 115:116957, 2023.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *arXiv: Computer Vision*

and Pattern Recognition, 2015a. URL https://api.semanticscholar.org/CorpusID:2134321.

Han, S., Pool, J., Tran, J., and Dally, W. J. Learning both weights and connections for efficient neural network. In *Neural Information Processing Systems*, 2015b. URL https://api.semanticscholar.org/CorpusID:2238772.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015. URL https://api.semanticscholar.org/CorpusID:13740328.

He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., and Han, S. Amc: Automl for model compression and acceleration on mobile devices. In *European Conference on Computer Vision*, 2018. URL https://api.semanticscholar.org/CorpusID:52048008.

Hoerl, A. E. and Kennard, R. W. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Isik, B., No, A., and Weissman, T. Successive pruning for model compression via rate distortion theory. *ArXiv*, abs/2102.08329, 2021. URL https://api.semanticscholar.org/CorpusID:231933836.

Itu-T and Jtc, I. I. Advanced video coding for generic audiovisual services. 2010. URL https://api.semanticscholar.org/CorpusID:60356047.

Jaderberg, M., Vedaldi, A., and Zisserman, A. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

Kossaifi, J., Bulat, A., Tzimiropoulos, G., and Pantic, M. T-net: Parametrizing fully convolutional nets with a single high-order tensor. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7814–7823, 2019. URL https://api.semanticscholar.org/CorpusID:102353394.

Li, F. and Liu, B. Ternary weight networks. *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2016. URL https://api.semanticscholar.org/CorpusID:13556195.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. *ArXiv*, abs/1608.08710, 2016. URL https://api.semanticscholar.org/CorpusID:14089312.

Lin, Y., Han, S., Mao, H., Wang, Y., and Dally, W. J. Deep gradient compression: Reducing the communication bandwidth for distributed training. *ArXiv*, abs/1712.01887, 2017. URL https://api.semanticscholar.org/CorpusID:38796293.

Liu, Z., Cheng, K.-T., Huang, D., Xing, E. P., and Shen, Z. Nonuniform-to-uniform quantization: Towards accurate quantization via generalized straight-through estimation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4932–4942, 2021. URL https://api.semanticscholar.org/CorpusID:244715141.

Long, X., Zeng, X., Ben, Z., Zhou, D., and Zhang, M. A novel low-bit quantization strategy for compressing deep neural networks. *Computational Intelligence and Neuroscience*, 2020(1):7839064, 2020.

Luo, J.-H., Wu, J., and Lin, W. Thinet: A filter level pruning method for deep neural network compression. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5068–5076, 2017. URL https://api.semanticscholar.org/CorpusID:11169209.

Luttrell, M., Wen, J., and Villasenor, J. D. Trellis-based rd optimal quantization in h. 263+. In *Proceedings 2000 International Conference on Image Processing (Cat. No. 00CH37101)*, volume 2, pp. 852–854. IEEE, 2000.

Masana, M., van de Weijer, J., Herranz, L., Bagdanov, A. D., and Álvarez, J. M. Domain-adaptive deep network compression. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 4299–4307, 2017. URL https://api.semanticscholar.org/CorpusID:11067299.

Park, J.-H., Kim, Y., Kim, J., Choi, J.-Y., and Lee, S. Dynamic structure pruning for compressing cnns. *ArXiv*, abs/2303.09736, 2023. URL https://api.semanticscholar.org/CorpusID:257622926.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. Xnor-net: Imagenet classification using binary convolutional neural networks. *ArXiv*, abs/1603.05279, 2016. URL https://api.semanticscholar.org/CorpusID:14925907.

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

Sharifi, K. and Leon-Garcia, A. Estimation of shape parameter for generalized gaussian distributions in subband decompositions of video. *IEEE Trans. Circuits Syst. Video Technol.*, 5:52–56, 1995. URL https://api.semanticscholar.org/CorpusID:41130607.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971, 2023. URL https://api.semanticscholar.org/CorpusID:257219404.

Wang, K., Liu, Z., Lin, Y., Lin, J., and Han, S. Haq: Hardware-aware automated quantization with mixed precision. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8604–8612, 2018a. URL https://api.semanticscholar.org/CorpusID:102350477.

Wang, P., Hu, Q., Zhang, Y., Zhang, C., Liu, Y., and Cheng, J. Two-step quantization for low-bit neural networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4376–4384, 2018b. doi: 10.1109/CVPR.2018.00460.

Wang, Y., Xu, C., Xu, C., and Tao, D. Beyond filters: Compact feature map for portable deep model. In *International Conference on Machine Learning*, 2017. URL https://api.semanticscholar.org/CorpusID:29145201.

Wen, J. and Villasenor, J. Structured prefix codes for quantized low-shape-parameter generalized gaussian sources. *IEEE Transactions on Information Theory*, 45(4):1307–1314, 1999. doi: 10.1109/18.761289.

Wien, M. High efficiency video coding. *Coding Tools and specification*, 24:1, 2015.

Xia, C.-G., Tsang, D. H.-K., and Lau, V. K. N. Structured bayesian compression for deep neural networks based on the turbo-vbi approach. *IEEE Transactions on Signal Processing*, 71:670–685, 2023a. URL https://api.semanticscholar.org/CorpusID:257050720.

Xia, F., Jin, J., Meng, L., Ding, F., and Zhang, H. Gan-based image compression with improved rdo process. In *International Conference on Image and Graphics*, pp. 361–372. Springer, 2023b.

Xu, D., Ouyang, W., Wang, X., and Sebe, N. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 675–684, 2018. URL https://api.semanticscholar.org/CorpusID:21670200.

Zhai, P., Guo, K., Liu, F., Xing, X., and Xu, X. Lapp: Layer adaptive progressive pruning for compressing cnns from scratch. *ArXiv*, abs/2309.14157, 2023. URL https://api.semanticscholar.org/CorpusID:262459258.

Zhang, Z., Lu, G., Liang, H., Tang, A., Hu, Q., and Song, L. Efficient dynamic-nerf based volumetric video coding with rate distortion optimization. *arXiv preprint arXiv:2402.01380*, 2024.

Zhou, S., Ni, Z., Zhou, X., Wen, H., Wu, Y., and Zou, Y. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *ArXiv*, abs/1606.06160, 2016. URL https://api.semanticscholar.org/CorpusID:14395129.

Zhu, C., Han, S., Mao, H., and Dally, W. J. Trained ternary quantization. *ArXiv*, abs/1612.01064, 2016. URL https://api.semanticscholar.org/CorpusID:224893.