Neural Weight Compression for Language Models

Jegwang Ryu [1] **Minkyu Kim** [1] **Seungjun Shin** [2] **Hee Min Choi** [2] **Dokwan Oh** [2] **Jaeho Lee** [1]

Abstract

Efficient compression of language model weights is increasingly critical as model scale and deployment grow. Yet, most existing methods rely on handcrafted transforms and heuristics, reflecting the limited understanding of *weights* as a data modality. To move beyond this paradigm, we formulate weight compression as neural codec learning and propose Neural Weight Compression (NWC), a framework for training neural codecs on pretrained weight datasets. NWC addresses challenges intrinsic to weight compression, including tensor heterogeneity and the mismatch between reconstruction losses and downstream performance. Experiments show that NWC achieves highly competitive accuracy–compression tradeoffs, with particularly strong results in the 4–6 bit regime, without relying on rigid handcrafted components such as the Hadamard transform. These gains extend to across diverse architectures, *e.g.,* vision encoders. Our analysis highlights the roles of entropy-constrained quantization and learned transforms in adapting compression to weight data and downstream tasks.

1. Introduction

The "weights" of neural nets constitute a new form of data, and the demand for efficient storage and transmission of this modality is rapidly increasing. This issue is particularly pressing for large language models (LLMs), whose parameter counts now reach hundreds of billions to the trillion scale (Gemini team, 2025). Beyond inference-time communication across intra- and inter-chip interconnects (Pope et al., 2023), LLM weights must also sit in cold storage on public registries, exchanged during distributed or federated training (McMahan et al., 2017), maintained as task- or user-specific updates for personalization (Hu et al., 2022), and pile up as training-checkpoint archives whose footprint now rivals that of the training data itself. Therefore, effective weight compression is essential for reducing storage and transmission costs and making LLMs easier to deploy and distribute.

Currently, the dominant paradigm in compressing language model weights is *quantization with **minimal, handcrafted transforms*** such as channel-wise scaling (Xiao et al., 2023; Lin et al., 2024) and orthogonal rotation (Ashkboos et al., 2024; Chee et al., 2023; Tseng et al., 2024b). This "minimalism" trend is especially prevalent in the accelerating LLM inference on conventional GPUs, where the latency from decoding operations can be a critical drawback. Nevertheless, the community has yet to reach a consensus on what constitutes the optimal transform, as evidenced by recent works (van Breugel et al., 2025; Liang et al., 2026). Furthermore, the manual design of transforms demands substantial human effort through trial-and-error, making it difficult to extend for covering a diverse family of LLM weights with highly heterogeneous distributions (Pierro & Abreu, 2024; Xu & Yang, 2025).

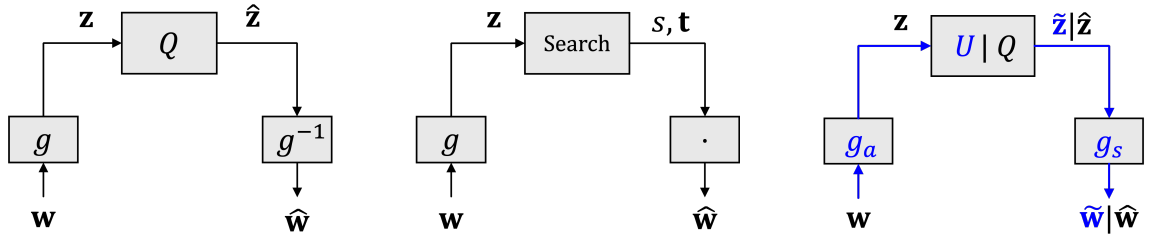
In this work, we step back from these constraints, and ask a more fundamental question:

*"What advantages can be gained by **learning to compress** language model weights?"*

Contribution. We develop Neural Weight Compression (NWC), a neural codec for the weight modality based on the transform-coding paradigm of Ballé et al. (2017). NWC consists of an analysis network, an entropy-coded latent representation, and a synthesis network trained end-to-end on pretrained weight tensors. Adapting neural codecs to weights introduces two key challenges: weight tensors are highly heterogeneous in shape and scale across layers, and the natural distortion measure—downstream task error— is intractable to directly optimize at LLM scale. NWC addresses these challenges through three components: (i) chunk-and-normalize preprocessing, (ii) an importanceaware training loss that prioritizes chunks more critical to model performance; and (iii) inference-time error compensation that propagates reconstruction errors through each layer.

Without relying on any handcrafted transforms, NWC

[1]Pohang University of Science and Technology (POSTECH) [2]Samsung Electronics Co., Ltd. Correspondence to: Jaeho Lee <jaeho.lee@postech.ac.kr>.



(a) Weight quantization *(b)* Search-based method *(c)* NWC (Ours)

Figure 1. Operational diagrams of various weight compression paradigms, and corresponding minimization objectives. Learnable paths are marked in blue. g, g^{-1} : denote the transforms and their inverse; $\cdot$: random matrix generation and multiplication; U/Q : Uniform noise / Quantization; Search : grid search of codewords and coefficients; g_a, g_s : trainable analysis & synthesis networks.

achieves competitive accuracy-compression tradeoffs. This benefit is particularly strong in the 4–6 bit regime on language models and extends to vision encoders for LLMs. Our analyses suggest that this success may be due to both the inherent limitations of the competing VQ-based baselines in handling high bitrates (Tseng et al., 2024b), and the ability of learned transform coding to capture features that are relevant to the downstream task.

Overall, our work provides a starting point for treating weight compression as a modality-specific codec problem rather than a byproduct of inference quantization.

2. Problem Statement

In this section, we formalize the general weight compression framework.

2.1. Transform coding for weight compression

Similar to the compression of natural signals, weight compression can be understood under the transform coding paradigm. Suppose a weight codec maps the original weight $\mathbf{w} \in \mathcal{W}$ to a reconstructed weight $\hat{\mathbf{w}} \in \mathcal{W}$ through a finite bitstream. Let $f : \mathcal{W} \to \mathcal{Z}$ be an *analysis transform* that maps the weight $\mathbf{w}$ to a latent representation $\mathbf{z} \in \mathcal{Z}$, and let $g : \mathcal{Z} \to \mathcal{W}$ be a *synthesis transform* that maps the latent representation back to the weight space. Then, the compression processes can be written as:

$$\text{Compression:} \quad \mathbf{c} = C(Q(f(\mathbf{w}))), \tag{1}$$

$$\text{Decompression:} \quad \hat{\mathbf{w}} = g(Q^{-1}(C^{-1}(\mathbf{c}))), \tag{2}$$

where the quantizer Q discretizes the latent representation, and the coder C encodes the quantized values into a finite bitstream $\mathbf{c}$.

2.2. Codec optimization

The performance of a codec is measured by the trade-off between rate and distortion. Let R denote the expected length of the bitstream and let $d : \mathcal{W} \times \mathcal{W} \to \mathbb{R}$ denote a task-dependent distortion measure. We consider the optimization:

$$\min_{f,g,C} \quad \mathbb{E}_{\mathbf{w}\sim\mathcal{D}_{\mathcal{W}}}[d(\mathbf{w}, \hat{\mathbf{w}})], \tag{3}$$

$$\text{subject to} \quad R \le b. \tag{4}$$

Distortion. Unlike conventional data compression, where distortion is often measured by reconstruction error such as MSE, weight compression should account for downstream model degradation. Let $\mathcal{M}_{\mathbf{w}}$ and $\mathcal{M}_{\hat{\mathbf{w}}}$ denote the models parameterized by the original and reconstructed weights, respectively. We define the distortion as $d(\mathbf{w}, \hat{\mathbf{w}}) = \mathrm{Err}_{\mathcal{T}}(\mathcal{M}_{\mathbf{w}}, \mathcal{M}_{\hat{\mathbf{w}}})$, where $\mathrm{Err}_{\mathcal{T}}$ is a task-dependent error functional measuring the discrepancy caused by replacing $\mathbf{w}$ with $\hat{\mathbf{w}}$.

2.3. From handcrafted to learned compression

Handcrafted compression. Recent existing weight compression methods can be viewed as handcrafted transform coding systems. Given a weight vector $\mathbf{w}$, they first apply an invertible linear transform T before quantization:

$$\mathbf{z} = T\mathbf{w}, \qquad \hat{\mathbf{w}} = T^{-1}\hat{\mathbf{z}}. \tag{5}$$

Two representative types of transforms have been widely explored. The first is *channel-wise scaling*, where T is a diagonal matrix. The second is *rotation-based transformation*, where T is an orthogonal matrix, so that $T^{-1} = T^{\top}$. These are computationally efficient and easy to invert.

They are also fixed-rate instances of the framework in Section 2.2. They store quantized indices using a predefined bitwidth, so the rate is fixed in advance and the objective reduces to minimizing distortion under a given rate budget.

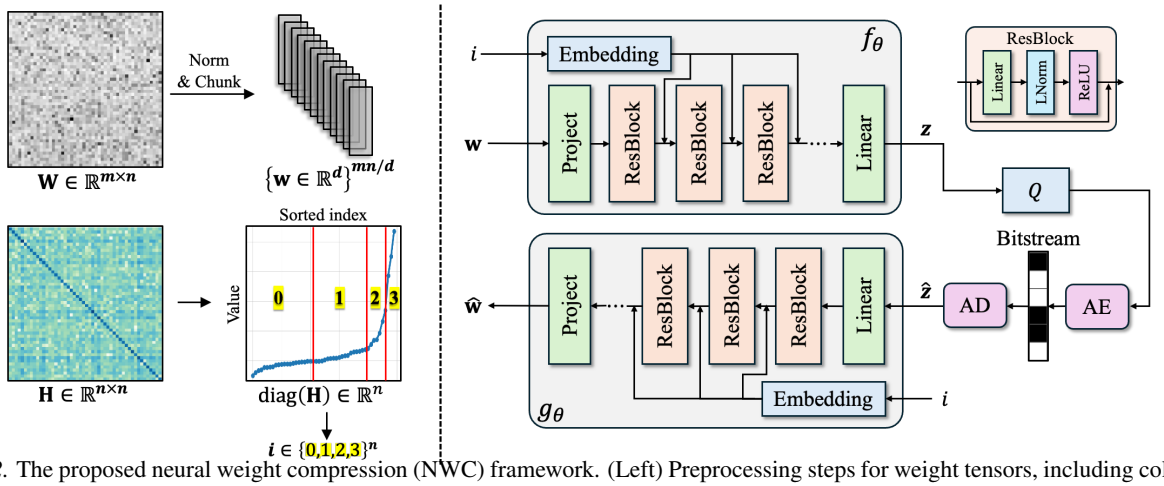


Figure 2. The proposed neural weight compression (NWC) framework. (Left) Preprocessing steps for weight tensors, including column-wise chunk-and-normalization and importance level assignment. (Right) Model architectures of the analysis and synthesis. AE/AD refer to Arithmetic Encoding/Decoding. Q denotes the rounding operator.

Learned compression. In this paper, we consider learned compression with nonlinear transforms. We parameterize the analysis and synthesis transforms as neural networks, $f(\cdot;\theta_a)$ and $g(\cdot;\theta_s)$, and optimize them jointly with an entropy model. The resulting learning objective is the Lagrangian relaxation of the rate–distortion problem:

$$\min_{\theta_a,\theta_s,\phi} \quad \mathbb{E}_{\mathbf{w}}\left[-\log p_\phi(\hat{\mathbf{z}}) + \lambda d(\mathbf{w},\hat{\mathbf{w}})\right], \tag{6}$$

where $\hat{\mathbf{z}} = Q(f(\mathbf{w};\theta_a))$, and $\hat{\mathbf{w}} = g(\hat{\mathbf{z}};\theta_s)$. Here, the first term $\mathbb{E}[-\log p_\phi(\hat{\mathbf{z}})]$ represents the rate loss, which penalizes the expected code length under the entropy model p_ϕ over the quantized latent representation. The parameter λ controls the trade-off between this rate and the distortion $d(\mathbf{w},\hat{\mathbf{w}})$.

This formulation allows the codec to learn both the representation and the code length from weight data, rather than relying on a manually designed transform and a fixed bitwidth.

3. Neural weight compression

Now we describe the proposed neural weight compression (NWC) framework for LLM weights.

3.1. Preprocessing: Chunk and normalize

Unlike images or videos, LLM weights vary widely in shape and scale. For instance, in Llama 3-8B, the key-projection matrix has size $\mathbb{R}^{1024\times4096}$, whereas the up-projection matrix has size $\mathbb{R}^{14336\times4096}$. Their statistics also differ substantially across layers and channels; see Appendices D.3 and D.4.

To handle this heterogeneity, we preprocess each weight matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$ with a simple column-wise pipeline. We first split $\mathbf{W}$ into column vectors $\mathbf{w}_{\mathrm{col}} \in \mathbb{R}^m$, normalize each column to unit standard deviation, and then divide them into fixed-length chunks $\mathbf{w} \in \mathbb{R}^{16}$, which serve as inputs to the neural codec; see Figure 2, left.

Column-wise normalization factors are stored in FP16 and used to restore the original scale after reconstruction. This incurs only about 0.004 bits per parameter. We use column-wise chunking because it aligns naturally with our inference-time error compensation procedure, described in Section 3.3.

3.2. Importance-aware training objective

Ideally, the codec should be trained to directly minimize $\mathrm{Err}_{\mathcal{T}}(\mathcal{M}_{\mathbf{w}}, \mathcal{M}_{\hat{\mathbf{w}}})$. For LLMs, however, this is impractical: Each update would require decoding the weights, running the full model on the task, and attributing the resulting error to individual compressed chunks. We therefore adopt a local proxy based on the output distortion of each linear layer.

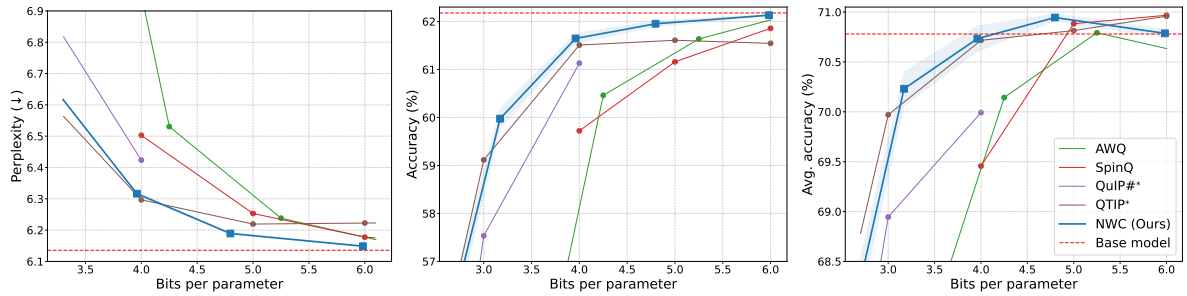
For a linear layer $\mathbf{x} \mapsto \mathbf{W}\mathbf{x}$, the effect of replacing $\mathbf{W}$ with $\hat{\mathbf{W}}$ can be measured as

$$\mathbb{E}[\|\mathbf{W}\mathbf{x} - \hat{\mathbf{W}}\mathbf{x}\|_2^2] = \mathrm{tr}\left((\mathbf{W}-\hat{\mathbf{W}})\mathbf{H}(\mathbf{W}-\hat{\mathbf{W}})^\top\right) \tag{7}$$

where the Hessian $\mathbf{H} = \mathbb{E}[\mathbf{x}\mathbf{x}^\top]$ is estimated from calibration activations. Motivated by activation outliers in LLMs (An et al., 2025; Sun et al., 2024a), we use the diagonal approximation

$$\approx \mathrm{tr}\left((\mathbf{W}-\hat{\mathbf{W}})\mathrm{diag}(\mathbf{H})(\mathbf{W}-\hat{\mathbf{W}})^\top\right), \tag{8}$$

which yields a Hessian-weighted MSE: errors in columns with larger activation (or Hessian diagonals) are penalized more heavily.



(a) WikiText-2 *(b)* MMLU *(c)* Commonsense tasks

Figure 3. Rate-accuracy tradeoffs on Llama 3-8B. We evaluate WikiText-2 perplexity with a context length of 2048, as well as zero-shot accuracies on MMLU and six common-sense tasks—ARC-Easy, ARC-Challenge, WinoGrande, PiQA, HellaSwag, BoolQ—across varying average bit-widths. NWC results are reported as an average over three random seeds, and the standard error is marked in shade.

Chunk-wise importance weights. Since NWC compresses fixed-length chunks for each column, all chunks from the same column inherit the corresponding Hessian diagonal. We discretize these values into K importance levels $i \in \{0, \dots, K-1\}$ and assign each level a weight $\lambda_I^{(i)}$. The training objective for a chunk $\mathbf{w}$ is:

$$\mathcal{L}_{\text{Imp}} = \mathbb{E}[-\log p_\phi(\hat{\mathbf{z}}) + \lambda \lambda_I^{(i)} \|\mathbf{w} - \hat{\mathbf{w}}\|_2^2]. \quad (9)$$

Here, $\lambda_I^{(i)}$ controls the reconstruction fidelity allocated to chunks of importance level i. In practice, a small number of levels, e.g., $K = 4$, is sufficient.

During training, we sample the importance level i uniformly at random and condition the codec on it. At inference, we instead use the Hessian-derived level of each column; see Figure 2 (left) and Appendix F.

Network architectures. As shown in the right panel of Figure 2, both the analysis transform f and synthesis transform g are residual MLPs. The importance index i is embedded and injected into each residual block by element-wise multiplication with the hidden states. Storing this index requires only $\lceil \log_2(K) \rceil$ bits per column, adding < 0.001 bits per parameter. We use a fully factorized entropy model with arithmetic coding (Ballé et al., 2017).

3.3. Inference-time error compensation

At inference, to further mitigate compression error, we employ two compensation mechanisms

Intra-layer error feedback. Incorporating the standard LDLQ objective (Chee et al., 2023), we update the uncompressed k-th column $\mathbf{w}_k$ to compensate for quantization errors from preceding columns:

$$\tilde{\mathbf{w}}_k = \mathbf{w}_k + (\mathbf{W}_{1:k-1} - \hat{\mathbf{W}}_{1:k-1})\mathbf{a}_k, \quad (10)$$

where $\mathbf{W}_{1:k-1}$ and $\hat{\mathbf{W}}_{1:k-1}$ denote the original and compressed preceding columns. Here, $\mathbf{a}_k$ corresponds to the k-th column of $\mathbf{L}^\top - \mathbf{I}$, where $\mathbf{L}$ is derived from the LDL decomposition of the layer's Hessian ($\mathbf{H} = \mathbf{L}^\top \mathbf{D} \mathbf{L}$). The updated $\tilde{\mathbf{w}}_k$ will then be compressed by the neural codec, and the next column will be updated subsequently. We note that our column-wise chunking strategy makes the compression pipeline compatible with this procedure.

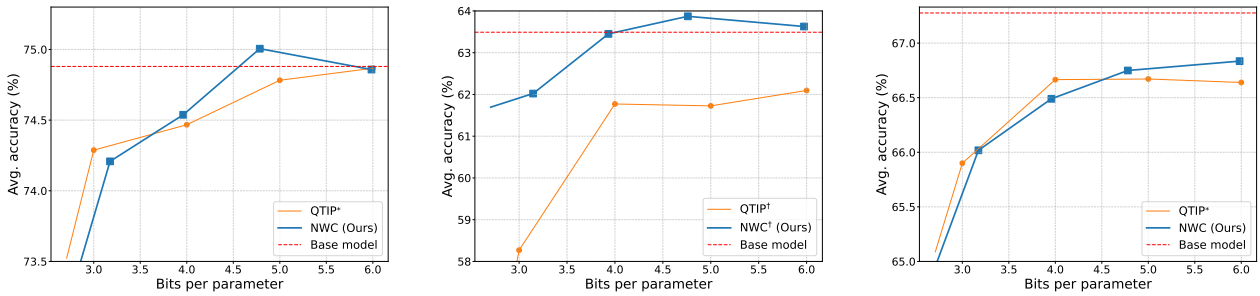
Inter-layer recovery fine-tuning. Before compressing each layer in a transformer block, we fine-tune the layer to account for the compression conducted on other layers in the same block; this is inspired by recent works in model compression (Tseng et al., 2024a; Egiazarian et al., 2024). Precisely, after compressing each layer in a k-th block, we fine-tune the uncompressed layers in the same block to minimize the MSE between the current block output and the original k-th block output of the uncompressed model. Here, as the block input, we use the features of the uncompressed model computed from calibration data enabling a parallel compression of multiple transformer blocks.

4. Experiments

4.1. Experimental setup

Baselines. We compare NWC against post-training model compression methods, including (1) Scalar quantization: AWQ (Lin et al., 2024), GPTQ (Frantar et al., 2023), and SpinQuant (Liu et al., 2025); (2) Vector quantization: QuIP# (Tseng et al., 2024a) and QTIP (Tseng et al., 2024b); (3) Pseudo-random generator: SeedLM (Shafipour et al., 2025); (4) Neural codec: ReALLM (Leconte et al., 2024). See Appendix E.1 for details.

Evaluation. To evaluate the quality of compressed models, we measure both perplexities and zero-shot accuracies. The perplexity is measured on the WikiText-2 (Merity et al., 2016) and the C4 datasets (Raffel et al., 2020) with the context length of 2048. Zero-shot accuracies are measured across MMLU benchmark (Hendrycks et al., 2021) and 6 common-sense reasoning tasks (ARC-Easy, ARC-Challenge, WinoGrande, PiQA, HellaSwag, BoolQ).



(a) Mixtral-8x7B *(b)* GPT-OSS-20B *(c)* Llama 2-7B

Figure 4. Rate-accuracy tradeoffs on six common-sense tasks. We evaluate average zero-shot accuracies ARC-Easy, ARC-Challenge, WinoGrande, PiQA, HellaSwag, BoolQ across varying average bit-widths.

Training. We train compression network on a dataset consisting of all linear layer weight tensors from Llama 3-8B. Both encoder and decoder consist of 4-layer residual MLPs with a width of 512. See Appendix E.4 for more details.

4.2. LLM weight compression

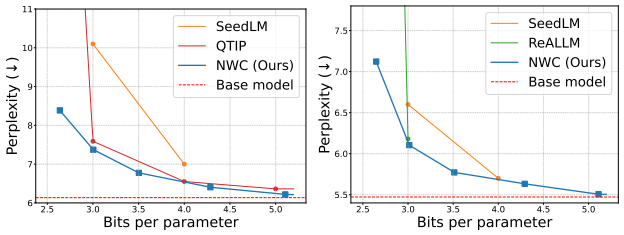
We measure the quality for compressed Llama models (Grattafiori et al., 2024; Touvron et al., 2023) at various bitrates in Figure 3 (see Figure 8 for C4 perplexity). For NWC, we report the average over three training runs of codec. NWC consistently outperforms most baselines, achieving low perplexity and high accuracy. Notably, the advantage of the neural approach becomes more pronounced at higher bitrates, i.e., over 4 bits.

In Figure 4, we validate the generalizability of our approach across diverse LLM architectures. In particular, we consider Mixtral (Jiang et al., 2024), Qwen3 (Yang et al., 2025), and GPT-OSS (Agarwal et al., 2025). We provide perplexity and MMLU accuracies in Figure 9. For MoE models, we employ a modified block-wise fine-tuning strategy—see Appendix E.6 for details. Without retraining the codec, we observe that the performance on these models is strong in the 4–6 bit regime, similar to what has been observed on the original Llama model.

4.3. Data-free scenarios

Additionally, we evaluate the performance of NWC in a data-free scenario, where the model is compressed without any calibration data. For this experiment, we compare our method against data-free baselines: SeedLM, and data-free version of ReALLM. For this experiment, we use a simplified version of NWC where all weight chunks are compressed to a uniform importance level and without applying adaptive error feedback or block-wise fine-tuning.

As shown in Figure 5, NWC consistently achieves lower perplexity compared to baselines, suggesting that it learns a more effective representation of the weights than simpler



(a) Llama 3-8B *(b)* Llama 2-7B

Figure 5. Data-free compression comparison against SeedLM and ReALLM. WikiText-2 perplexity of Llama models.

approximations based on a heuristic search. As reported in the paper, ReALLM exhibits a significant performance drop at low bitrate in the data-free setting, as it heavily relies on the tuning of low-rank components using data.

4.4. Other experiments

In Appendix B and Appendix C, we provide additional experimental results on the follows topics:

- Perplexity and MMLU result (Appendix B.1)
- Vision encoders (Appendix B.2)
- Comparision with non-PTQ methods (Appendix B.3)
- Ablation studies (Appendix C)
- Coding latency analysis (Appendix D.1)

5. Analysis

5.1. The role of entropy-constrained quantization

Why is NWC effective, especially at 4–6 bit range? Our analyses suggest that this may be due to the effectiveness of the entropy-constrained quantization—i.e., jointly optimized rate and distortion—in handling heavy tails at higher rates, where prior approach falls suboptimal.

To show this, we compare the rate-distortion curve of various compression schemes—scalar Lloyd-Max quantization (SQ), vector quantization (VQ), trellis-coded quanti-

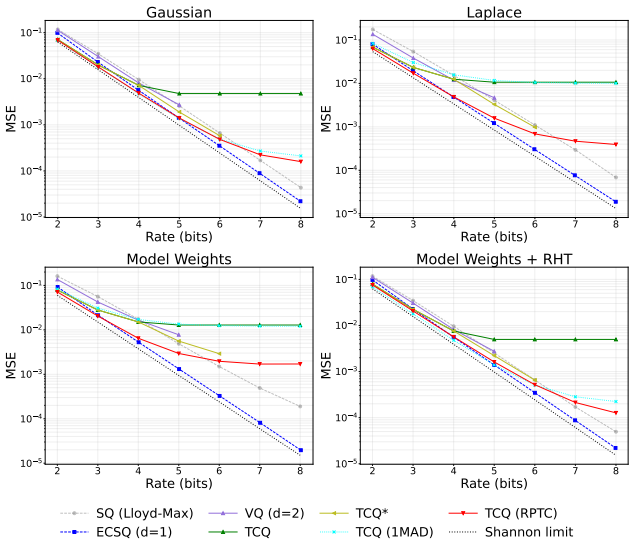
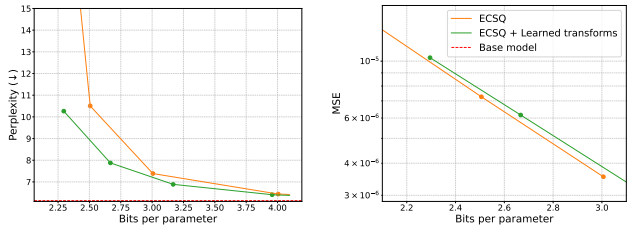


Figure 6. Rate-distortion curves for various quantization schemes and data distributions. The entropy constrained scalar quantization (ECSQ) remains close to the Shannon limit at high rates, while fixed-rate methods (SQ, VQ, TCQ, TCQ*) become suboptimal. RHT denotes the random Hadamard transform, and the model weights are from the Llama 3-8B.



(a) WikiText-2 perplexity *(b)* MSE

Figure 7. Perplexity vs. MSE analysis on Llama 3-8B. The learned compression yields superior downstream performance despite higher reconstruction error.

zation (TCQ), and entropy-constrained scalar quantization (ECSQ)—across various data distributions; For TCQ, we adopt the configuration of QTIP (Tseng et al., 2024b). We also evaluate a variant of TCQ, termed TCQ*, which uses the variable-sized codebook optimized for the rate. Furthermore, for the Laplace and model weight sources, the codebooks were initialized using a Laplace distribution instead of the standard normal distribution. Details regarding the experimental setup are provided in Appendix E.7.

In Figure 6, we observe that ECSQ stays close to the Shannon limit across all sources and rates (Gish & Pierce, 1968). In contrast, fixed-rate methods exhibit the gap that widens as the rate increases. Variants with relaxed codebook constraints, such as TCQ* and TCQ (RPTC), also fail to be near-optimal, especially in heavy-tailed distributions. While random Hadamard transform (RHT) can mitigate this, the gap persists at high rates. ECSQ yields superior and robust performance, particularly at ≥ 4 bits.

Table 1. Statistical analysis of transformed query projection weights. The neural encoder significantly reduces kurtosis and outliers compared to other transformations.

Method	Kurtosis	Max Value (σ)	Outliers ($> 3\sigma$)
Original	20.48	43.57	1.94%
DCT	0.46	10.54	0.50%
Random Rotation	5.42	16.77	1.86%
RHT	5.44	16.66	1.86%
NWC	**0.00**	**2.70**	**0.00%**

5.2. The role of learned transforms

If ECSQ alone can achieve near-Shannon-limit MSE, *what is the role of the learned transforms?* Experiments suggest that the transform helps ensure a good model quality, which is not fully guaranteed by having low MSE.

In Figure 7, we observe that while adding learned transform to ECSQ slightly worsens the MSE, it can greatly reduce the perplexity of the compressed LLM. This observation suggests that the learned transform effectively steers the compression process to retain weight components essential for model performance, rather than naïve average distortion. This intuition is also confirmed in Appendix D.2.

Finally, we note that the learned transform also has an effect of suppressing outliers and heavy tails (Table 1). Here, we observe significantly reduced kurtosis and outlier frequency for features yielded by neural encoders, rendering the representation more amenable to coding.

6. Conclusion

We introduce Neural Weight Compression, a novel framework that treats model weights as a learnable modality. Ultimately, we envision this work serving as a pioneering step toward fully automated compression pipelines in the era of large-scale models.

Limitation and future work. While our current implementation relies on tensor-level simulations due to the nontrivial nature of implementing optimized fused kernels for tile-wise arithmetic decoding and memory constraints of decoding parameters, we believe that bridging this gap through dedicated efforts will unlock meaningful inference acceleration. We leave the development of such optimized kernels as a promising direction for future research.

NWC framework opens up the possibility of leveraging recent advances from the neural data compression literature. For instance, incorporating techniques such as progressive coding (Jeon et al., 2023; Lu et al., 2021; Lee et al., 2022) and code optimization (Gao et al., 2022; Perugachi-Diaz et al., 2024) are promising directions for future work. Exploring these avenues could further enhance both the performance and flexibility of the neural weight compression.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Agarwal, S., Ahmad, L., Ai, J., Altman, S., Applebaum, A., Arbus, E., Arora, R. K., Bai, Y., Baker, B., Bao, H., et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Amini, A., Gabriel, S., Lin, S., Koncel-Kedziorski, R., Choi, Y., and Hajishirzi, H. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019.

An, Y., Zhao, X., Yu, T., Tang, M., and Wang, J. Systematic outliers in large language models. In *International Conference on Learning Representations*, 2025.

Ashkboos, S., Mohtashami, A., Croci, M. L., Li, B., Cameron, P., Jaggi, M., Alistarh, D., Hoefler, T., and Hensman, J. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *Advances in Neural Information Processing Systems*, 2024.

Ballé, J., Laparra, V., and Simoncelli, E. P. End-to-end optimized image compression. In *International Conference on Learning Representations*, 2017.

Banaei, M., Bałazy, K., Kasymov, A., Lebret, R., Tabor, J., and Aberer, K. Revisiting offline compression: Going beyond factorization-based methods for transformer language models. In *Findings of the Association for Computational Linguistics: EACL 2023*, 2023.

Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Cai, W.-P., Li, M.-Y., and Li, W.-J. Lcq: Low-rank codebook based quantization for large language models. *arXiv preprint arXiv:2405.20973*, 2024.

Chee, J., Flynn (née Renz), M., Damle, A., and De Sa, C. M. Model preserving compression for neural networks. In *Advances in Neural Information Processing Systems*, 2022.

Chee, J., Cai, Y., Kuleshov, V., and De Sa, C. M. QuIP: 2-bit quantization of large language models with guarantees. In *Advances in Neural Information Processing Systems*, 2023.

Chen, H., Gwilliam, M., Lim, S.-N., and Shrivastava, A. HNeRV: A hybrid neural representation for videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Cheng, L., Guan, P., Taherkordi, A., Liu, L., and Lan, D. Variational autoencoder-based neural network model compression. *arXiv preprint arXiv:2408.14513*, 2024.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Défossez, A., Copet, J., Synnaeve, G., and Adi, Y. High fidelity neural audio compression. *Transactions on Machine Learning Research*, 2023.

Du, D., Zhang, Y., Cao, S., Guo, J., Cao, T., Chu, X., and Xu, N. BitDistiller: Unleashing the potential of sub-4-bit LLMs via self-distillation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.

Egiazarian, V., Panferov, A., Kuznedelev, D., Frantar, E., Babenko, A., and Alistarh, D. Extreme compression of large language models via additive quantization. In *Proceedings of the International Conference on Machine Learning*, 2024.

Frantar, E. and Alistarh, D. Optimal brain compression: A framework for accurate post-training quantization and pruning. In *Advances in Neural Information Processing Systems*, 2022.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023.

Gao, C., Xu, T., He, D., Wang, Y., and Qin, H. Flexible neural image compression via code editing. In *Advances in Neural Information Processing Systems*, 2022.

Gemini team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint 2507.06261*, 2025.

Gish, H. and Pierce, J. N. Asymptotically efficient quantizing. *IEEE Trans. Inf. Theory*, 1968.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Hassibi, B., Stork, D., and Wolff, G. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, 1993.

He, D., Yang, Z., Peng, W., Ma, R., Qin, H., and Wang, Y. ELIC: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

Hsu, Y.-C., Hua, T., Chang, S., Lou, Q., Shen, Y., and Jin, H. Language model compression with weighted low-rank factorization. In *International Conference on Learning Representations*, 2022.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Iwai, S., Miyazaki, T., and Omachi, S. Controlling rate, distortion, and realism: Towards a single comprehensive neural image compression model. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024.

Jeon, S., Choi, K. P., Park, Y., and Kim, C.-S. Context-based trit-plane coding for progressive image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Jia, Z., Li, B., Li, J., Xie, W., Qi, L., Li, H., and Lu, Y. Towards practical real-time neural video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., Casas, D. d. l., Hanna, E. B., Bressand, F., et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Leconte, L., Bedin, L., Nguyen, V. M., and Moulines, E. ReALLM: A general framework for LLM compression and fine-tuning. *arXiv preprint arXiv:2405.13155*, 2024.

Lee, J.-H., Jeon, S., Choi, K. P., Park, Y., and Kim, C.-S. DPICT: Deep progressive image compression using trit-planes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Li, H., Li, S., Dai, W., Li, C., Zou, J., and Xiong, H. Frequency-aware transformer for learned image compression. In *International Conference on Learning Representations*, 2024.

Liang, Y., Chen, H., Zhang, Z., Han, S., and Liu, Z. Paroquant: Pairwise rotation quantization for efficient reasoning llm inference. In *International Conference on Learning Representations*, 2026.

Lin, C.-H., Gao, S., Smith, J. S., Patel, A., Tuli, S., Shen, Y., Jin, H., and Hsu, Y.-C. ModeGPT: Modular decomposition for large language model compression. In *International Conference on Learning Representations*, 2025.

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., Xiao, G., Dang, X., Gan, C., and Han, S. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems*, 2024.

Liu, J., Sun, H., and Katto, J. Learned image compression with mixed transformer-CNN architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023a.

Liu, Z., Oguz, B., Zhao, C., Chang, E., Stock, P., Mehdad, Y., Shi, Y., Krishnamoorthi, R., and Chandra, V. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*, 2023b.

Liu, Z., Zhao, C., Fedorov, I., Soran, B., Choudhary, D., Krishnamoorthi, R., Chandra, V., Tian, Y., and Blankevoort, T. SpinQuant: LLM quantization with learned rotations. In *International Conference on Learning Representations*, 2025.

Lu, G., Ouyang, W., Xu, D., Zhang, X., Cai, C., and Gao, Z. DVC: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2019.

Lu, Y., Zhu, Y., Yang, Y., Said, A., and Cohen, T. S. Progressive neural image compression with nested quantization and latent ordering. In *IEEE International Conference on Image Processing*, 2021.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Conference on Artificial Intelligence and Statistics*, 2017.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open

book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Nagel, M., Baalen, M. v., Blankevoort, T., and Welling, M. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1325–1334, 2019.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Perugachi-Diaz, Y., Gansekoele, A., and Bhulai, S. Robustly overfitting latents for flexible neural image compression. In *Advances in Neural Information Processing Systems*, 2024.

Pierro, A. and Abreu, S. Mamba-PTQ: Outlier channels in recurrent large language models. *arXiv preprint arXiv:2407.12397*, 2024.

Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Heek, J., Xiao, K., Agrawal, S., and Dean, J. Efficiently scaling transformer inference. In *Proceedings of Machine Learning and Systems*, 2023.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21 (140):1–67, 2020.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. ImageNet large scale visual recognition challenge. *International journal of computer vision*, 115(3): 211–252, 2015.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. WinoGrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Shafipour, R., Harrison, D., Horton, M., Marker, J., Bedayat, H., Mehta, S., Rastegari, M., Najibi, M., and Naderiparizi, S. SeedLM: Compressing llm weights into seeds of pseudo-random generators. In *International Conference on Learning Representations*, 2025.

Sharma, P., Ding, N., Goodman, S., and Soricut, R. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of ACL*, 2018.

Sun, M., Chen, X., Kolter, J. Z., and Liu, Z. Massive activations in large language models. In *Conference on Language Modeling*, 2024a.

Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, 2024b.

Tian, Y., Wang, C., Han, J., Tang, Y., and Han, K. PocketLLM: Ultimate compression of large language models via meta networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Tseng, A., Chee, J., Sun, Q., Kuleshov, V., and De Sa, C. QuIP#: Even better LLM quantization with Hadamard incoherence and lattice codebooks. In *Proceedings of the International Conference on Machine Learning*, 2024a.

Tseng, A., Sun, Q., Hou, D., and De Sa, C. M. QTIP: Quantization with trellises and incoherence processing. In *Advances in Neural Information Processing Systems*, 2024b.

van Breugel, B., Bondarenko, Y., Whatmough, P., and Nagel, M. Fptquant: Function-preserving transforms for llm quantization. *arXiv preprint arXiv:2506.04985*, 2025.

Wang, X., Alam, S., Wan, Z., Shen, H., and Zhang, M. SVD-LLM v2: Optimizing singular value truncation for large language model compression. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2025a.

Wang, X., Zheng, Y., Wan, Z., and Zhang, M. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations*, 2025b.

Weber, M., Fu, D., Anthony, Q., Oren, Y., Adams, S., Alexandrov, A., Lyu, X., Nguyen, H., Yao, X., Adams, V., et al. Redpajama: an open dataset for training large language models. *Advances in neural information processing systems*, 37:116462–116492, 2024.

Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the International Conference on Machine Learning*, 2023.

Xu, C. and Yang, D. DLLMQuant: Quantizing diffusion-based large language models. *arXiv preprint arXiv:2508.14090*, 2025.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yuan, Z., Shang, Y., Song, Y., Wu, Q., Yan, Y., and Sun, G. ASVD: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*, 2023.

Zeghidour, N., Luebs, A., Omran, A., Skoglund, J., and Tagliasacchi, M. SoundStream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2021.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Zhai, X., Mustafa, B., Kolesnikov, A., and Beyer, L. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 11975–11986, 2023.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

Appendix

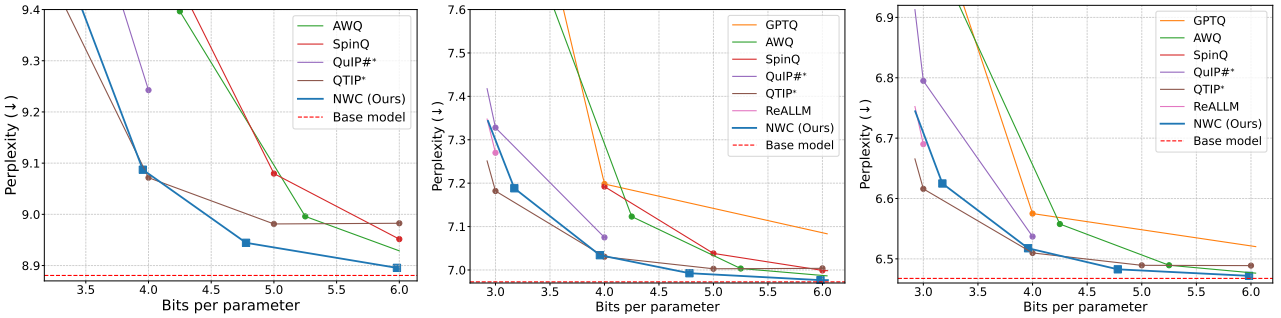
A. Related Work

Neural data compression. Unlike conventional handcrafted codecs, the neural codec framework does not require an explicit modeling of the data generation procedure, as it fits the data distribution directly. Moreover, the flexibility to optimize for any differentiable distortion measure allows the maximization of diverse quality metrics beyond simple MSE (Zhang et al., 2018). These advantages have led to state-of-the-art performance in compressing high-dimensional natural signals, such as images (Ballé et al., 2017; He et al., 2022; Li et al., 2024; Liu et al., 2023a), videos (Lu et al., 2019; Jia et al., 2025), and audio (Zeghidour et al., 2021; Défossez et al., 2023). Inspired by this progress, our work extends this paradigm to *language model weights*—data characterized by high dimensionality and complex generation procedures, where preserving model performance is paramount.

Weight compression via neural codecs. Most of the prior attempts to apply neural compression on model weights—e.g., based on simple autoencoders (Banaei et al., 2023) or VAEs (Cheng et al., 2024)—largely focus on compressing small-scale models like BERT or LSTMs. Unlike these works, there have been recent attempts based on VQ-VAE to compress the weights of LLMs (Leconte et al., 2024; Tian et al., 2025). These works tend to rely on a *piecemeal strategy*—overfitting codes or tuning decoders for individual weight matrices, similar with the implicit neural representation literature (Chen et al., 2023). In contrast with these works, we propose a single, unified encoder-decoder pair that compresses the entire set of weights for an LLM. To the best of our knowledge, this is the first of such *global* neural compression framework that can be effectively scaled to handle large language models.

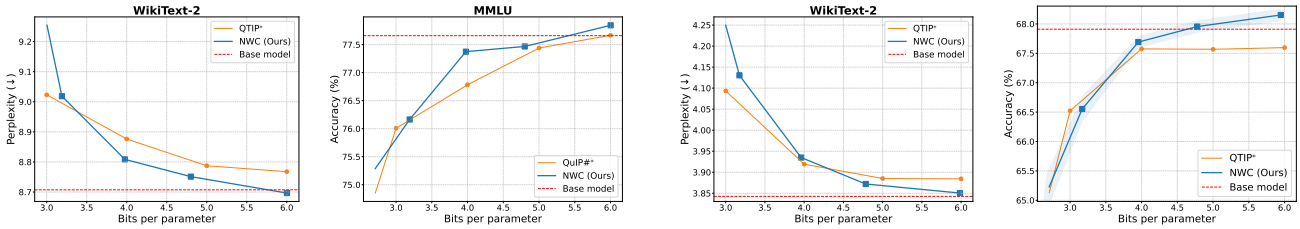
Weight compression via quantization. Quantization remains the dominant paradigm for LLM compression. Early approaches have primarily focused on scalar quantization, employing second-order error compensation (Frantar et al., 2023) or activation-aware scaling (Lin et al., 2024) to preserve salient weights. As an attempt to further enhance the robustness against outliers, subsequent works have introduced incoherent transformations, such as the random Hadamard transform, to Gaussianize weight distributions (Chee et al., 2023; Ashkboos et al., 2024). More recently, vector quantization frameworks have pushed compression to lower bitwidths (e.g., sub-3-bit) (Egiazarian et al., 2024; Tseng et al., 2024a;b). While some of these methods incorporate some data-driven optimizations, such as gradient-based codebook optimization (Egiazarian et al., 2024; Cai et al., 2024; Tseng et al., 2024b), our framework essentially differs by leveraging fully learned nonlinear transformations and entropy coding driven by learnable probability model, without using handcrafted transforms.

Other approaches. There are several other non-learned approaches for compressing neural network weights such as pruning (Hassibi et al., 1993; Chee et al., 2022; Sun et al., 2024b), and low-rank matrix approximation (Hsu et al., 2022; Yuan et al., 2023; Wang et al., 2025b; Lin et al., 2025). More recently, Shafipour et al. (2025) has explored representing weight blocks using a random seed, with corresponding coefficients as the compressed code. These works tend to put a very significant restriction on the computational footprint of the decoding procedure, using one or no matrix multiplications. In contrast, our work relaxes this restriction, considering multi-layer decoders and entropy decoding. However, as we will demonstrate, the decoding latency can be kept at a competitive level.



(a) Llama 3-8B *(b)* Llama 2-7B *(c)* Llama 2-13B

Figure 8. Rate-accuracy performance on Llama models. We evaluate C4 perplexity (context length 2048) across various average bit-widths.



(a) Qwen3-30B-A3B *(b)* Mixtral-8x7B

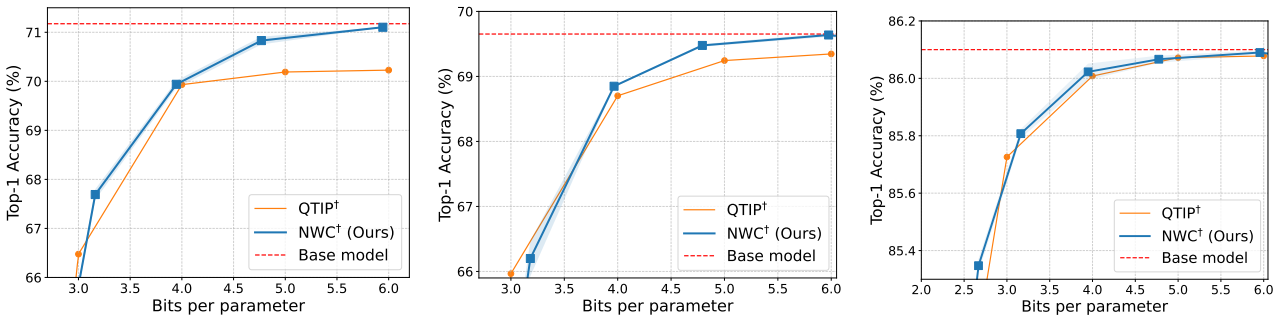
Figure 9. Rate-accuracy trade-off comparison. Each subfigure shows the Wikitext2 perplexity (left plot) and MMLU accuracy (right plot) for (a) Qwen3-30B-A3B and (b) Mixtral-8x7B.

B. Additional Results

B.1. Language models

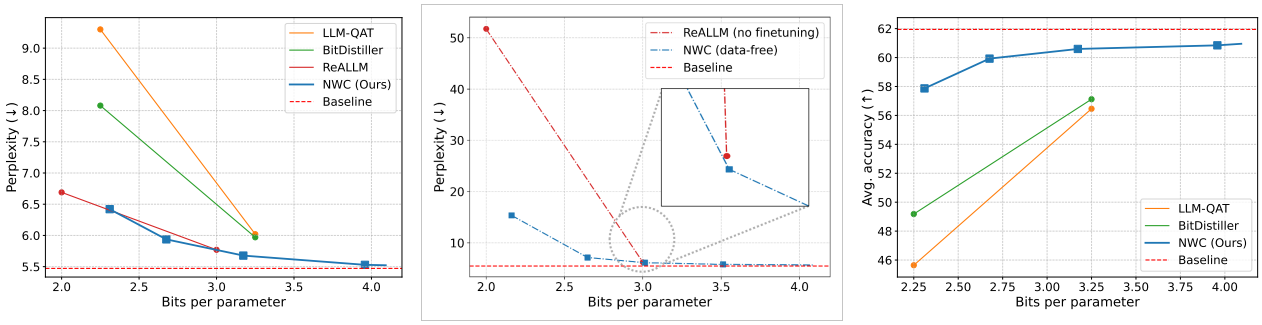
C4 perplexity of Llama models. Figure 8 presents the C4 perplexity of Llama3-8B, Llama2-7B, and Llama2-13B across various bit range.

Mixtral & Qwen3. Figure 9 illustrates the rate-distortion performance on Qwen3-30B-A3B and Mixtral-8x7B. For each model, we report the Wikitext2 perplexity (left) and MMLU accuracy (right). Consistent with the results on Llama models, our approach maintains high performance at mid to high bitrates, demonstrating its robustness across diverse architectures without requiring model-specific retraining.



(a) CLIP *(b)* SigLIP *(c)* DINOv2

Figure 10. Compression of vision encoders. We report zero-shot classification accuracy for CLIP and SigLIP, and linear probing accuracy for DINOv2 on ImageNet-1k.



(a) WikiText-2 *(b)* WikiText-2 (data-free) *(c)* Zero-shot tasks

Figure 11. Compression results of Llama 2-7B.

Table 2. Performance of Llama 3-8B compressed by SVD-based methods and NWC.

METHOD	Ratio↓	Wiki-2↓	OpenQ	ARCe	WG	HS	PQ	MathQ	Avg↑
Original	100%	6.14	0.35	0.80	0.73	0.60	0.80	0.40	0.61
SVD-LLM	80%	11.82	0.29	0.77	0.64	0.51	0.72	0.30	0.54
SVD-LLM V2	80%	8.01	0.33	0.79	0.70	0.58	0.77	0.36	0.59
NWC	25%	**6.32**	**0.34**	**0.80**	**0.75**	**0.60**	**0.79**	**0.39**	**0.61**

B.2. Vision models

To demonstrate versatility, we evaluate NWC on the prominent vision encoders. We evaluate zero-shot classification accuracy of CLIP-ViT-L/16 (Radford et al., 2021) and SigLIP-B/16 (Zhai et al., 2023), and linear probe accuracy of DINOv2-L (Oquab et al., 2023) on ImageNet (Russakovsky et al., 2015). For these experiments, the NWC codec is first pre-trained on Llama weights and then fine-tuned on the weights of each respective model. No recovery fine-tuning is applied to either method, which we mark with †. See Appendix E.5 for more details.

The results, presented in Figure 10, show that NWC achieves superior performance at mid-to-high bitrates. This is consistent with the trend observed in the LLM experiments, suggesting that the benefits of our neural approach generalize across the neural networks trained on the data of different modalities.

B.3. Comparison with other baselines

In the main experiments, we leave out comparing NWC with the QAT methods. The main reason is that QAT methods often require significantly more computation than ours. Specifically, LLM-QAT demands approximately 280 hours on an A100 80G. Furthermore, QAT relies on the full pre-training corpus or extensive synthetic data, NWC requires only the small calibration set.

QAT & ReALLM. In this section, we compare NWC against Quantization-Aware Training: LLM-QAT (Liu et al., 2023b), BitDistiller (Du et al., 2024) and other neural codecs: ReALLM (Leconte et al., 2024). The evaluation is performed on Llama2-7B, measuring perplexity on WikiText-2 and the average zero-shot accuracy across four commonsense tasks (PiQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), and ARC-Challenge (Clark et al., 2018)). As shown in Figure 11, NWC consistently demonstrates superior performance compared to QAT-based methods at equivalent bitrates. Crucially, when evaluated in a calibration data-free setting, NWC compression performance is substantially better.

SVD-based methods. Table 2 presents a comparison between NWC and SVD-based compression methods (Wang et al., 2025b;a). We evaluate both the compression ratio and the zero-shot accuracy on a range of tasks, including OpenbookQA (Mihaylov et al., 2018) and MathQA (Amini et al., 2019). Here, the compression ratio is defined ratio of total storage cost, calculated as the total number of bits in the compressed model divided by that of the original model. The results indicate that NWC achieves superior performance compared to SVD even at higher compression ratios.

PocketLLM. Table 3 presents the perplexity comparison with PocketLLM (Tian et al., 2025) on the WikiText-2 and C4 benchmarks with context length of 4096. Note that we do not include zero-shot results from the paper in this section. A

direct comparison was not feasible because their reported baseline performance for the uncompressed model and other methods diverges from ours, presumably due to differences in the evaluation setup.

Table 3. Perplexity comparison of Llama 2-7B on WikiText-2 and C4 benchmarks (context length 4096).

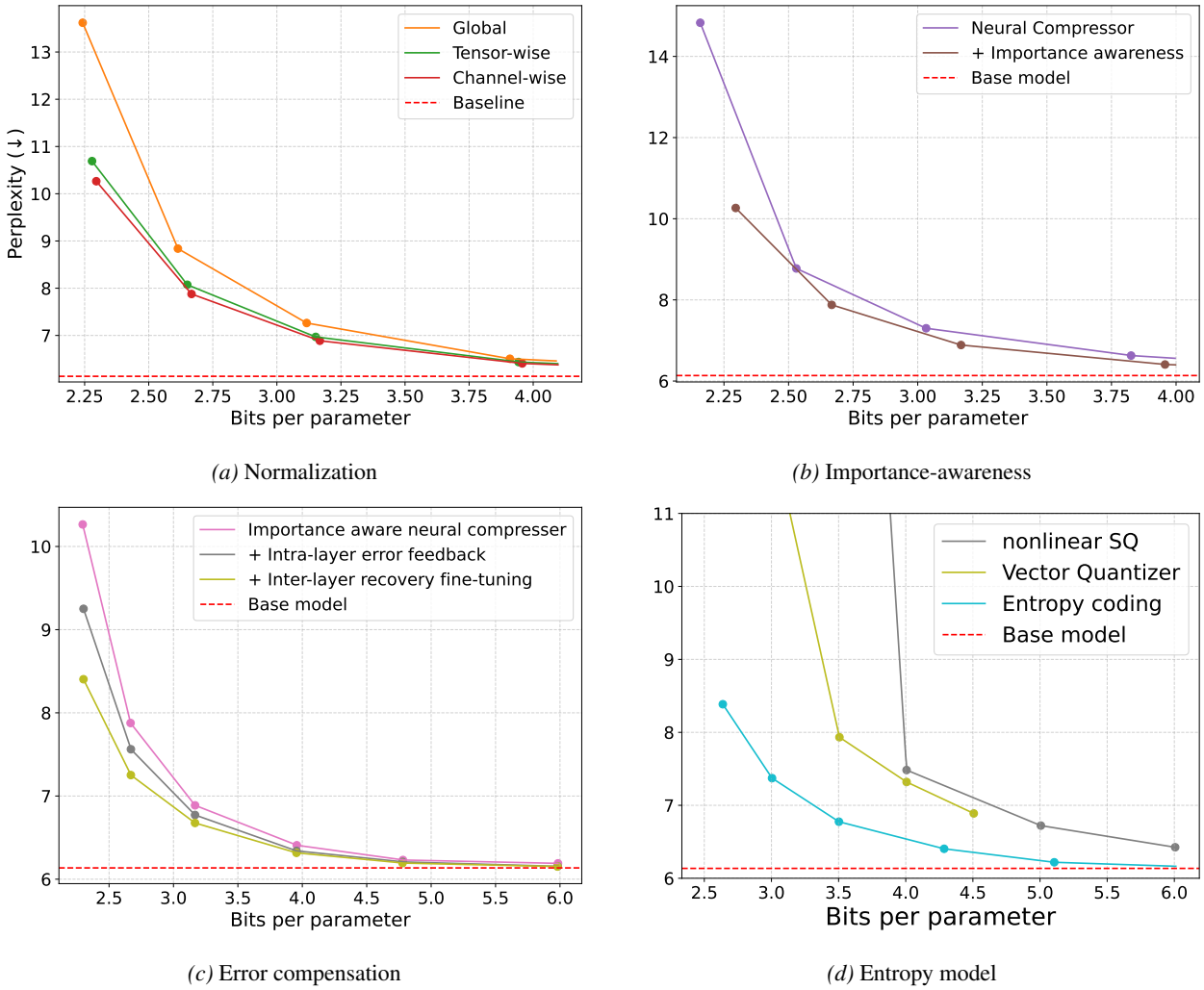
Method	**Bits**	**WikiText-2**	**C4**
FP16	16	5.12	6.63
PocketLLM	3.98	5.27	6.86
NWC (Ours)	3.96	**5.17**	**6.70**

B.4. Standard deviations across multiple seeds

Table 4 summarizes the standard deviations observed across three independent seeds.

Table 4. Standard deviations of Llama 3-8B performance metrics computed over three seeds.

	Bits per parameter		
Metric	**3.96 bits**	**4.80 bits**	**5.98 bits**
WikiText-2 (PPL)	0.009	0.008	0.005
C4 (PPL)	0.004	0.003	0.005
Common-sense (%)	0.23	0.07	0.08
MMLU (%)	0.20	0.12	0.04



(a) Normalization

(b) Importance-awareness

(c) Error compensation

(d) Entropy model

Figure 12. Ablation studies on NWC components. We report WikiText-2 perplexity (context length 2048) on Llama 3-8B.

C. Ablation studies

We systematically evaluate the efficacy of each technical component of the proposed framework. First, we observe that applying normalization at a finer granularity (i.e., channel-wise) yields the most favorable rate-distortion trade-off compared to coarser alternatives (Figure 12a). Second, the analysis confirms that leveraging importance-awareness is crucial for minimizing perplexity, particularly in low bitrate regimes (Figure 12b). Furthermore, we validate the necessity of the inference-time error mitigation strategies; the results demonstrate that both intra- and inter-layer compensation mechanisms are beneficial for effectively recovering model performance during compression (Figure 12c). Finally, comparisons against fixed-rate baselines reveal the clear superiority of our learned entropy model over standard scalar and vector quantization approaches, as shown in Figure 12d.

C.1. Encoder and decoder networks

To quantify the benefit of the learnable g_a and g_s networks, we compare the model against an entropy-only baseline. This baseline replaces g_a and g_s with identity functions and is trained using only the auxiliary and rate losses; without a distortion loss, it is incapable of controlling the bitrate or adapting reconstruction quality. For this ablation, both importance awareness and error compensation were disabled. The results in Table 5 demonstrate that the learnable networks offer a large reduction in perplexity compared to the entropy-only model.

Table 5. Wikitext-2 and C4 perplexity for Llama-3-8B.

En/Decoder	Bits	Wikitext-2↓	C4↓
$\mathrm{Identity}(\cdot)$	2.12	27.8	33.2
g_a/g_s	2.13	14.5	20.7

C.2. Use of entropy model

In Figure 12d, to validate the effectiveness of density estimation-based entropy coding within our framework, we conducted an ablation study comparing our method against baselines where the entropy model is replaced by non-linear fixed-length quantizers. We trained and compared compression models utilizing a non-linear scalar quantizer, a vector quantizer, and the entropy model, respectively. For all three models, the encoder and decoder architectures were identical, and no input importance was employed. The two fixed-length quantizers were trained using the Straight-Through Estimator (STE), with the dimension of the vector quantizer set to 2.

C.3. Importance aware compression

Figure 12b presents a comparison of compression performance between our importance-guided quality allocation and a baseline using a uniform quality assignment. The results demonstrate that leveraging importance consistently leads to lower perplexity. This performance gain is particularly significant in the low bitrate regime.

C.4. Channel-wise normalization

Figure 12a presents an ablation study on the granularity of weight normalization. We compare the performance when normalization is applied at different levels: globally (across the entire model weights), tensor-wise (per-layer), and channel-wise. The results, which account for the bit overhead required to store the normalization factors, show a clear trend: a finer granularity (i.e., channel-wise) consistently leads to lower perplexity.

C.5. Compensation for compression error

The effectiveness of our inference-time compression error compensation is demonstrated in Figure 12c. The results show that applying each of the intra- and inter-layer error compensation mechanisms leads to a lower perplexity, confirming the positive contribution of both components.

Table 6. Encoding and decoding latency for 4096×4096 tensor, measured on a single NVIDIA RTX 6000 Ada.

Operation	NWC (Ours)	GPTQ	QTIP	QuIP#
Encoding ($\times 10^3$ ms)	1.64 $\pm$0.30	0.69 $\pm$0.02	19.84 $\pm$0.30	21.71 $\pm$0.82
Decoding (ms)	1.17 $\pm$0.07	0.08 $\pm$0.04	0.52 $\pm$0.02	1.33 $\pm$0.33
Synthesis, g	1.07 $\pm$0.07	-	-	-
Entropy Decoding	0.10 $\pm$0.01	-	-	-

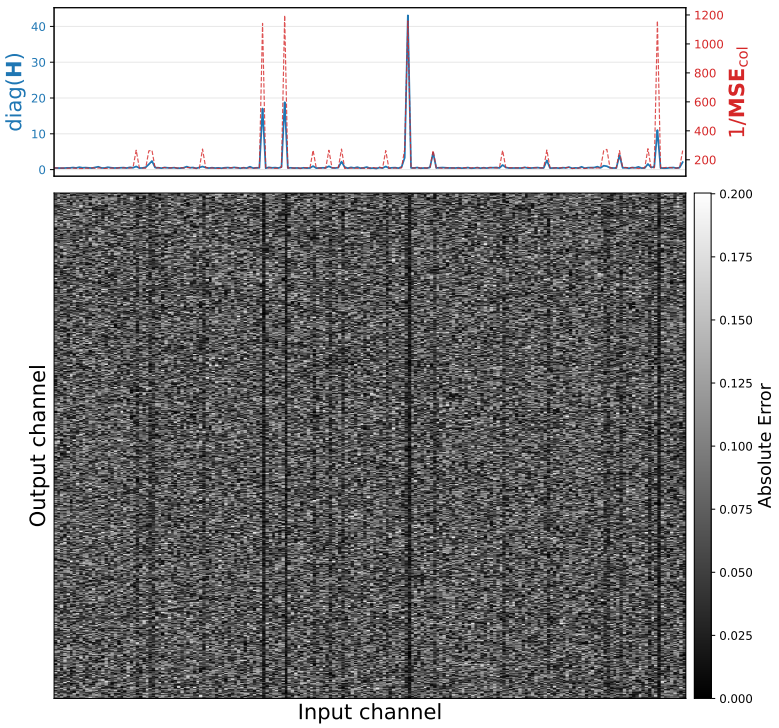


Figure 13. Compression error for the 10th layer Q-projection in Llama 3-8B. Weights were normalized by their column-wise standard deviation prior to compression. (Top) Comparison between the Hessian diagonals and the inverse column-wise MSE. (Bottom) Heatmap visualizing the absolute error magnitude.

D. Additional Analyses

D.1. Latency analysis

In Table 6, we assess the computational efficiency of NWC by measuring the wall clock latency for compressing and decompressing a 4096×4096 weight tensor. We have evaluated under at the target rate of 4 bits per parameter, on a single NVIDIA RTX A6000 GPU. Detailed experimental settings are provided in Appendix E.8.

From the table, we observe that the NWC achieves the latencies that are competitive to prior quantization methods, despite the prevalent belief that entropy-coding-based methods are slow. Indeed, it outperforms vector quantization baselines in terms of the encoding speed, and performs similarly in terms of the decoding speed. This is largely due to the availability of GPU-accelerated entropy decoding library—namely, the "nvCOMP" library by NVIDIA.[1] These results highlight the proposed neural codec may have some potential to be further developed into a practical tool in the near future.

D.2. The effect of Hessian-based importance-aware compression

The Hessian-based importance-aware transform effectively steers the compression process to retain weight components essential for model performance, rather than naïve average distortion. This intuition is confirmed by Figure 13, where we visualize the loss Hessian and channel-wise reconstruction error. As expected, channels with larger Hessian diagonals demonstrate lower MSE.

[1] https://developer.nvidia.com/nvcomp

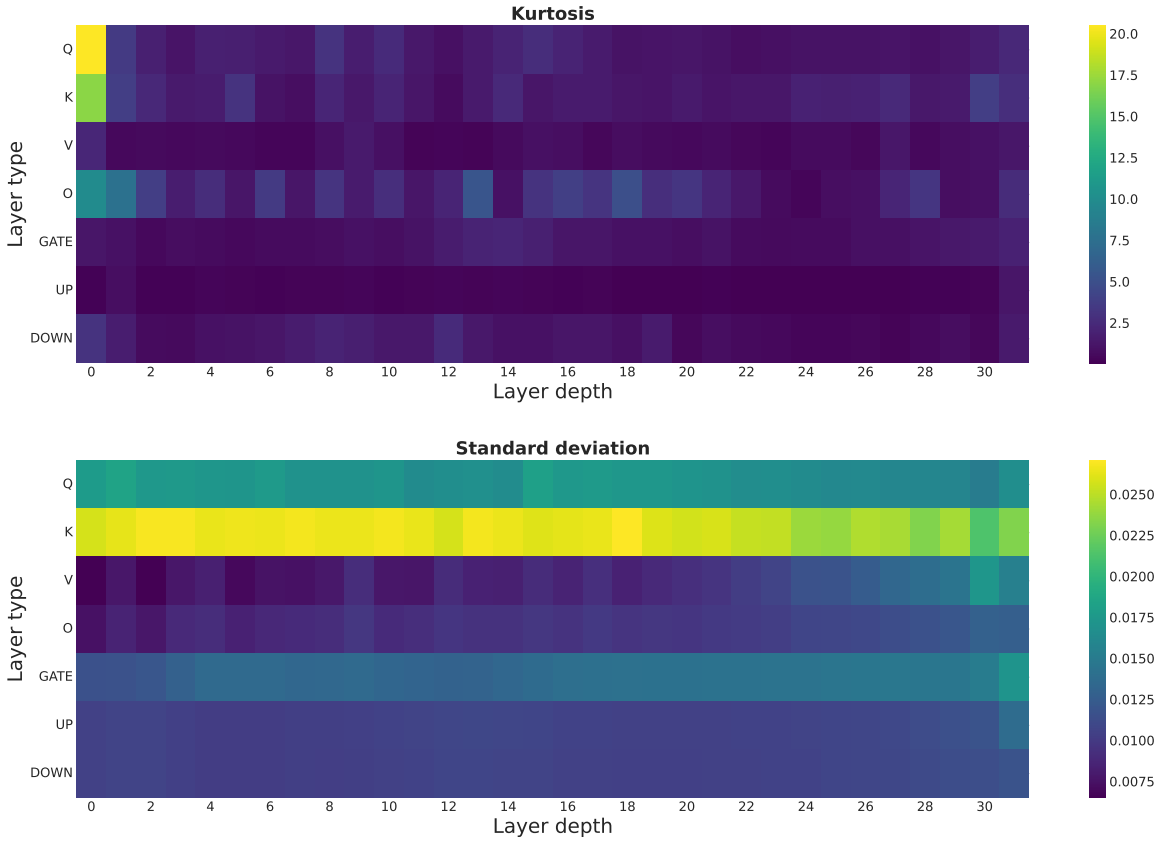


Figure 14. Layer-wise statistic of Llama 3-8B. (Top) Kurtosis. (Bottom) Standard deviation.

D.3. Per-layer statistic of Large language model

In Figure 14, we visualize the kurtosis and standard deviation across different layer depths and types. From these results, we observe three key characteristics:

- The majority of layers exhibit kurtosis values higher than that of a Gaussian distribution (i.e., > 0), indicating heavy-tailed distribution.
- Certain layers exhibit extremely high kurtosis. This is particularly pronounced in the Q, K and O projections of the first block.
- The standard deviation varies depending on the layer type.

D.4. Channel-wise scale

Similar to the observation in CNNs (Nagel et al., 2019), the transformer-based models exhibit scaling differences across channels. We visualize the channel-wise scale for the Q matrix of Llama 3-8B in Figure 15

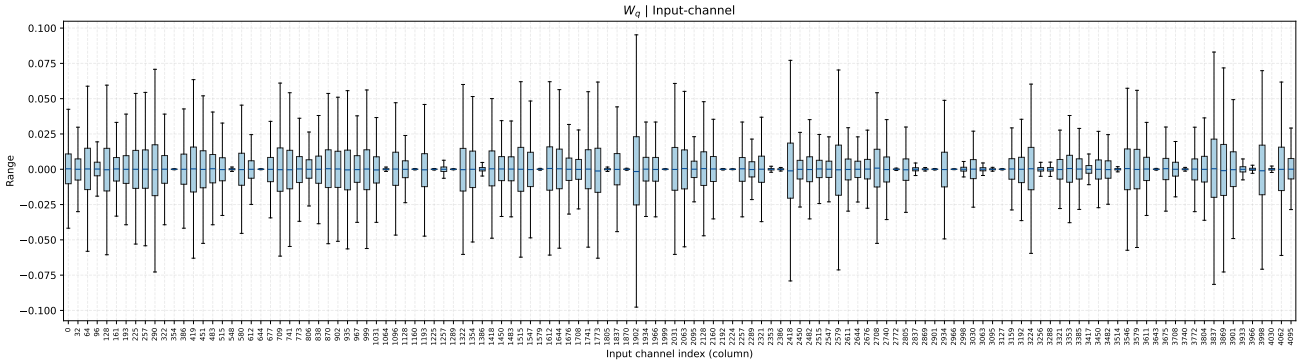


Figure 15. Channel-wise range of Q projection of Llama 3-8B.

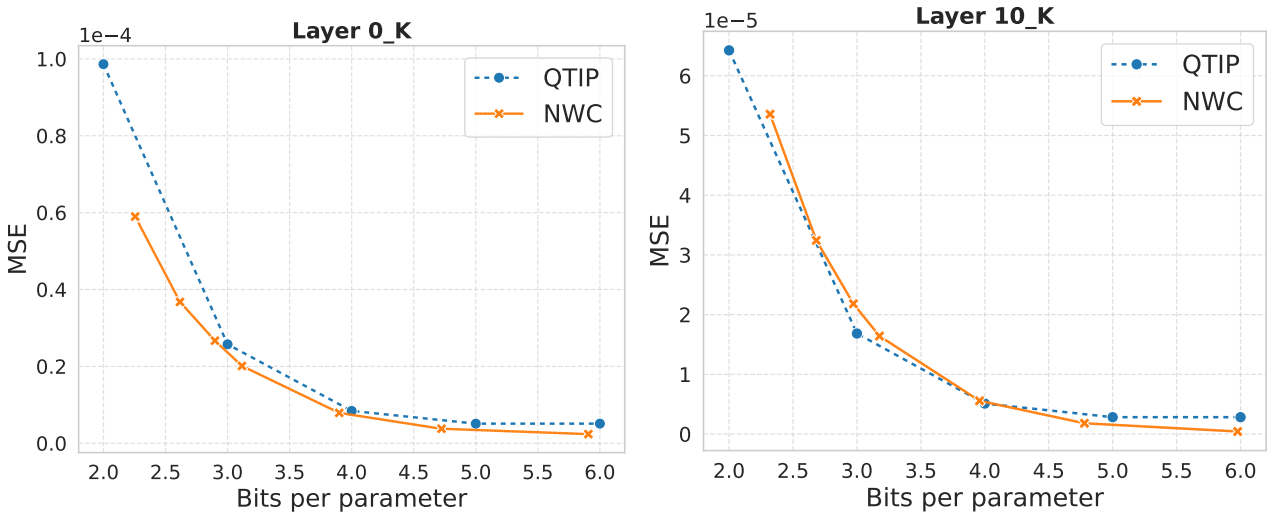


Figure 16. Per-layer rate-distortion curve of K projection layer in different blocks.

D.5. Robustness to heavy-tailed distributions

Figure 16 presents the single-layer rate-distortion curves for QTIP and NWC. Our method outperforms QTIP in terms of MSE, particularly in layers characterized by high kurtosis (i.e., heavy-tailed distributions). This observation aligns with Figure 6 and Table 1, demonstrating that learned codecs are more robust to extreme distributions and surpass handcrafted approaches in handling heavy-tailed weights.

E. Implementation Details

E.1. Experimental setup details

For a thorough comparison, we reproduce the results of the baselines over the bitwidths missing in the original paper. For closed-source models, we compare with the figures in the original papers. Furthermore, to evaluate the efficacy of the compression scheme in isolation, we compare with the baselines without end-to-end fine-tuning. To clarify this point, we mark the modified baselines—QuIP# and QTIP—with an asterisk (*). For all experiments, we report the effective bits per parameter, which explicitly accounts for the overhead of metadata, including normalization factors and importance level indices.

E.2. Reproduction of baselines over various bitwidths

For QTIP, AWQ, and SpinQuant, we reproduce the results of the baselines over the bitwidths missing in original papers.

- QTIP: We change k in hybrid-computed codes
- SpinQuant: We optimize the WbA16KV16 rotation scheme for each target bitwidth b, then apply PTQ using this rotation in b-bits weight quantization setting.
- AWQ: We apply b-bit weight quantization using group size of 128, which induces roughly 0.25 extra bits per parameter. This overhead is included in the *bits per parameter* reported on the X-axis of the comparison plots. (e.g. Figure 3, Figure 8)

In contrast, we were unable to reproduce results for QuIP# at arbitrary bit-widths. This is because it relies on fixed codebooks for each specific bitrate targeted in the original paper, which does not allow for flexible bitrate adjustment.

E.3. Hessian generation

We generate Hessian matrices following the protocol of QuIP# (Tseng et al., 2024a), using the exact same matrices for all baseline comparisons (NWC, QuIP#, QTIP). We use the RedPajama dataset (Weber et al., 2024) for all language models:

6,144 sequences (length 2,048) for the Llama 2/3 families, and 2,048 sequences for Mixtral-7x8B, Qwen3-30B-A3B, GPT-OSS-20B, and Gemma3-4B. For vision models, Hessians are estimated using 512 samples from the Conceptual Captions Dataset (Sharma et al., 2018).

E.4. Network design and training

Hyperparmeter. Details regarding the architecture and training of our compression network are provided in Table 7. We employ an auxiliary loss to update the quantile parameters of the entropy model, which is trained separately from the main rate-distortion objective.

Table 7. Hyperparameters for network design and training

Hyperparameter	Value
Chunk size	16
g_a network width	512
g_a number of residual blocks	4
g_s network width	512
g_s number of residual blocks	4
Entropy model channel size	16
Learning rate	1×10^{-4}
Learning rate (auxiliary loss)	1×10^{-3}
Optimizer	Adam
λ	$\{30, 50, 100, 300, 1000, 10000\}$
λ_I	$\{0.29,\ 0.83,\ 10,\ 20\}$

Training cost. We train the codec with the following compute resources and data:

- Compute cost: The codec is trained for 60 epochs, requiring 11.45 hours on a single NVIDIA A6000 Ada GPU.
- Dataset: The codec training does not require any external datasets or text corpora, utilizing only the model weights. For computational efficiency during random sampling, we aggregated 64 weight chunks into a single training sample. Consequently, for LLaMA-3-8B, the training set consists of approximately 6.8 million examples, with a validation set of 1,000 samples.

Fine-tuning on specific model. For vision models, fine-tuning on model weights required maximum half an hour for 6 epochs using a single NVIDIA RTX A6000 GPU. For laguage models, our experimental results demonstrate that the learned codec generalizes effectively to diverse architectures even without fine-tuning step. We observed that when attempting to fine-tune on new model weights, the initial loss was already near convergence, only minimal adaptation is required for most current transformer-based architectures.

E.5. Experiment details for vision models

We applied our compression technique to the vision encoder models as follows:

- CLIP and SigLIP, compression was applied to all projection layers within both the text encoder and vision encoder blocks.
- DINOv2, compression was applied to all layers within the main encoder blocks.
- For all models, the collection of weights targeted for compression was used as a dataset to fine-tune the NWC codec specifically for that model.

E.6. Block-wise inter-layer recovery fine-tuning for MoE models

Recent Mixture-of-Experts (MoE) architectures, such as Mixtral (Jiang et al., 2024), and Qwen (Yang et al., 2025), contain a significantly larger number of linear layers within a single Transformer block due to the multiplicity of experts. Consequently, performing block-wise recovery fine-tuning after compressing every single layer incurs prohibitive computational overhead. To address this we adopted a modified strategy that applies fine-tuning to groups of layers in the experiment of Figure 4 and Figure 9 for NWC and QTIP. Specifically, we execute the recovery fine-tuning following these groups of

layers:

- Compress the self-attention weights ($\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_o$) and the router weights, followed by block fine-tuning.
- Compress the up-projection weights of *all* experts, followed by block fine-tuning.
- Compress the gate-projection weights of *all* experts, followed by block fine-tuning.
- Compress the down-projection weights of *all* experts, followed by block fine-tuning.

E.7. Experiment details of Figure 6

To substantiate the efficacy of entropy coding and optimization via the joint rate-distortion objective, we compare the rate-distortion performance of various compression schemes—scalar Lloyd-Max quantization (SQ), vector quantization (VQ), trellis coded quantization (TCQ), and entropy-constrained scalar quantization (ECSQ)—across i.i.d. Gaussian and Laplacian sources, and pre-trained language model weights. For all evaluations, we fix a sample size of 2^{20}

For TCQ baselines, we evaluate several configurations to ensure a rigorous comparison:

- **Standard TCQ:** We adopt the default configuration from QTIP (Tseng et al., 2024b) with constraint length $L = 16$, trellis states $V = 2$, and a fixed codebook size $Q = 9$.
- **TCQ* (Scaled codebook):** To address performance saturation at higher bitrates (specifically ≥ 5 bits), we employ a variant where the codebook size scales dynamically with the target rate ($Q = 2 \times \text{rate} + 1$). Furthermore, for Laplace and model weight sources, codebooks are initialized using a Laplace distribution rather than the standard normal distribution.
- **TCQ variants:** We also include the Random Permutation Trellis Coder (RPTC) as implemented in QTIP, and the Lookup-Free Computed Codebook (1MAD) variant to benchmark against diverse TCQ optimizations.

E.8. Details of latency analysis

In this section, we provide details of the latency analysis presented in Appendix D.1. The reported values represent the average wall-clock time and standard deviation for processing a 4096×4096 weight tensor at a target rate of 4 bits per parameter on NVIDIA RTX A6000.

- **Encoding Time:** This measures the total duration to transform the original tensor into its compressed format. Crucially, this includes the computational overhead of the Hessian-based intra-layer error feedback mechanism (e.g., LDLQ). Note that we exclude the runtime for inter-layer recovery fine-tuning for NWC, QuIP#, and QTIP. This is because this process operates at the block level—making tensor-wise attribution ambiguous—and incurs a comparable computational cost across all methods.
- **Decoding Time:** This corresponds to the latency involved in fully reconstructing the full precision tensor in VRAM from the compressed bitstream. For NWC, the total decoding time is calculated as the sum of the neural synthesis latency and the entropy decoding time. Specifically, the entropy decoding component is benchmarked separately using the Asymmetric Numeral Systems (ANS) codec from the NVIDIA nvCOMP library[1].

F. Details of Determination and Assignment of Importance Levels

Assigning importance levels during inference. We determine the importance level for each column based on the magnitude of its corresponding Hessian diagonal elements. Specifically, we employ a quantile-based stratification strategy with thresholds that are fixed across all weight tensors. Columns in the top 0.1% of Hessian values are assigned importance level 3, those in the top 1% receive level 2, and those in the top 10% are set to level 1. All remaining columns are assigned level 0. This ensures that bit allocation is prioritized for the most sensitive parameters.

Importance level training coefficient. During training, an importance level coefficient is sampled from a discrete set for each weight chunk. These values were empirically selected to span a wide spectrum of operating points, ensuring the model learns to handle diverse rate-distortion trade-offs. Following the literature on variable-rate neural image compression (Iwai et al., 2024), we formulate the loss by applying the reciprocal of the importance coefficient λ_I to the rate term (i.e., weighting the rate by $1/\lambda_I$). We observe that this formulation enhances the model's ability to learn diverse bitrate controls compared to distortion weighting schemes.

F.1. Importance-aware strategy in AWQ and GPTQ

In this section, we explicitly compare the proposed NWC with existing post-training quantization methods like AWQ and GPTQ. Our method differs in two critical aspects:

- **Discrete importance:** For compact storage, we discretize the scaling factors into a small set of levels (e.g., $K = 4$). These levels can be stored using only trivial overhead (e.g., 2 bits per channel). In contrast, the sensitivity metrics in GPTQ and AWQ are continuous.
- **Importance-augmentation during training:** To address the imbalance in samples across importance levels, we conduct importance-augmented training where each vector is paired with a randomly selected scaling factor.

F.2. Activation-aware scaling in AWQ

AWQ's activation-aware scaling is defined as

$$\mathbf{s} = {\mathbf{s}_\mathbf{X}}^{\alpha^*}, \quad \alpha^* = \arg\min_{\alpha \in [0,1]} \mathcal{L}(\mathbf{s}_\mathbf{X}^\alpha) \tag{11}$$

where $(\mathbf{s}_\mathbf{X})_c = \frac{1}{N}\sum_{i=1}^{N} |\mathbf{X}_{c,i}|$, and $\mathcal{L}(\mathbf{s})$ is quantization objective of Equation (4) in Lin et al. (2024).

This scaling mechanism relies on the empirical assumption that the quantization step size $\Delta = \max(|\mathbf{w}|)$, remains stable even after scaling (i.e., $\Delta' = \max(|\mathbf{s}^\top \mathbf{w}|) \approx \Delta$). As the scaling factor s increases, this assumption breaks down, forcing AWQ to use a heuristic search to find a safe α.

As a fully learnable framework, NWC can explicitly incorporate the scale into its loss function, allowing the optimizer to find the most effective compression directly with respect to the importance.

F.3. Sensitivity metrics in GPTQ

Drawing from OBQ (Frantar & Alistarh, 2022), GPTQ leverages arbitrary ordering and Cholesky reformulation to define the error sensitivity metrics of each weight column as:

$$\mathbf{s} = \mathrm{diag}(\mathbf{L}^\top)^{-1}, \quad \mathbf{L} = \mathrm{Cholesky}(\mathbf{H}^{-1}). \tag{12}$$

Adopting this as an importance metric, however, is computationally more intensive than simply using the Hessian diagonal $\mathbf{\Lambda}$. It requires one additional matrix inversion and one Cholesky decomposition for each Hessian matrix. Empirically, we find that using the diagonal of the Hessian directly is not only sufficient but often yields superior results for training neural compression models.

G. Justification for Diagonal Hessian Approximation

In Section 3.2, we employ a diagonal approximation of the Hessian matrix to measure parameter sensitivity. While the full Hessian captures cross-parameter correlations, we justify the adequacy of the diagonal approximation based on the distinct activation characteristics–outlier activations– in LLMs.

LLMs are known to have massive outlier activations, where specific feature dimensions possess magnitudes significantly larger than others (Sun et al., 2024a; An et al., 2025). Let $\mathbf{x} \in \mathbb{R}^{d_{in}}$ denote the input activation. The Hessian is typically approximated using the expected outer product of the inputs, $H \approx \mathbb{E}[\mathbf{x}\mathbf{x}^\top]$.

Consider a feature dimension d that corresponds to an outlier feature. Mathematically, if the magnitude of this outlier feature is significantly larger than other dimensions (i.e., $|x_d| \gg |x_j|$ for $j \neq d$), the diagonal term H_{dd} dominates the off-diagonal terms:

$$H_{dd} \approx \mathbb{E}[x_d^2] \gg \mathbb{E}[x_d x_j] \approx H_{dj} \tag{13}$$

The Hessian matrix becomes effectively diagonally dominant in the presence of strong outliers. Therefore, the diagonal approximation serves as accurate and computationally efficient proxy for parameter importance.