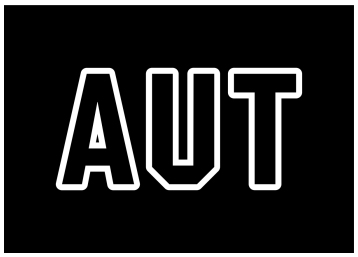


Towards Superior Quantization for Large Language Models

The Mixed and Layer-sensitive Approaches

Feng Zhang

Supervisor: Dr. Weihua Li
Dr. Yanbin Liu

School of Engineering, Computer & Mathematical Sciences
Auckland University of Technology

A thesis submitted to Auckland University of Technology
in fulfilment of the requirements for the degree of
Master of Computer and Information Sciences

25/2/2025

To my wife, for your boundless support and resilience in holding our family together across the ocean.

To my daughter, whose growth and responsibility during this journey have been my greatest pride. Your determination and love, shown through every act of maturity, were a constant source of strength.

To my sisters, for your quiet strength and devotion in caring for our mother without complaint and hesitation allowing me to pursue this dream with peace of mind.

This thesis is a testament to your sacrifices and unwavering support. For all of you, my remarkable family.

Copyright

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgments.

25/2/2025

Acknowledgements

Researching cutting-edge LLM technologies has been a challenging and overwhelming experience, characterized by a plethora of unfamiliar jargons, complex mathematical equations involving optimisation, statistics, and linear algebra, as well as the demanding software and hardware, prolonged experimentation, complicated data analysis and visualization. I am fortunate to have had a supportive team guiding me throughout this journey. Their unwavering encouragement and belief in my abilities have been a constant source of motivation, enabling me to navigate numerous setbacks and moments of frustration along the way.

I am extremely grateful to my supervisor, Dr. Weihua Li, whose encouragement and unconditional trust provided me with the motivation and confidence to pursue this work. His support has been invaluable in helping me overcome the challenges of this research. I also extend my heartfelt thanks to my co-supervisor, Dr. Yanbin Liu, whose extensive expertise in vision models has profoundly shaped my research methodology. His keen insights into experimental design and dataset preparation have been instrumental to the success of this work. I am especially appreciative of his generosity in providing datasets and research resources, which have significantly facilitated my experiments and enhanced the quality of this thesis. Furthermore, I am deeply indebted to Ms. Xiaodan Wang for her meticulous and insightful review of my work. Her ability to uncover subtle inconsistencies and propose elegant solutions has greatly enhanced the clarity of this thesis.

Finally, I would like to express my gratitude to the Department of Computer Sciences at Auckland University of Technology for supplying facilities and resources that are indepensable to the success of this thesis.

Abstract

Large Language Models (LLMs) have exhibited remarkable capabilities in tasks such as natural language comprehension, content generation, and knowledge retrieval. However, training and serving these models require substantial computational resources, posing a significant barrier to AI application development and research. To address these challenges, various model compression techniques have been explored, with quantization emerging as a key approach. Nonetheless, existing quantization methods predominantly apply uniform quantization configurations, failing to account for the varying quantization difficulty across different layers in billion-scale models. This results in a rigid memory-accuracy trade-off and leaves the potential for improving quantization accuracy through differentiated memory allocation largely unexplored. To bridge these research gaps, this thesis advances the study of LLM quantization with two key contributions. First, it introduces MXQ, a mixed-quantization method designed to provide a more flexible memory-accuracy balance. MXQ formulates a novel optimization approach to determine optimal layer-wise quantization parameters while enforcing overall memory constraints. Experimental results demonstrate that MXQ unlocks a broader spectrum of quantization configurations, simplifying the memory-accuracy trade-off while maintaining performance comparable to the baseline. Second, this thesis proposes SensiBoost and KurtBoost, two methods that enhance quantization accuracy by leveraging layer-sensitive features such as activation sensitivity and weight distribution kurtosis to identify critical layers. These approaches outperform existing baselines, achieving up to 9% lower perplexity with only a 2% increase in memory budget on Llama models.

Publications

Zhang, F.; Liu, Y.; Li, W.; Wang, X. and Bai, Q. (2025). A Mixed Quantization Approach for Data-Free Quantization of LLMs. In Proceedings of the 17th International Conference on Agents and Artificial Intelligence - Volume 2, ISBN 978-989-758-737-5, ISSN 2184-433X, pages 353-363.

Zhang, F.; Liu, Y.; Li, W.; Wang, X. and Bai, Q. Jie, L. (2025). Towards Superior Quantization Accuracy: A Layer-sensitive Approach. [to be submitted]

Table of contents

List of figures

List of tables

List of Algorithms

Chapter 1

Introduction

1.1 Background

Large vision and language models have demonstrated remarkable human-like intelligence in tasks such as natural language comprehension, problem-solving, logical reasoning, and knowledge retrieval. Training and serving these large neural network models typically requires GPUs to accelerate the neural network's forward and backward propagation processes. These specialized processors excel at executing massive parallel operations, such as large-scale matrix multiplication (Baji, 2018). To minimize latency, the model parameters, gradients, and associated optimizer states are stored in GPU memory using high-precision numeric representations like float16 or float32. However, this approach often proves inadequate in addressing the increasing computational resource demands, driven by the scaling laws that the model performance demonstrates a smooth power-law relationship with three scale factors: model parameters, size of training dataset and amount of computing resources (Kaplan et al., 2020). The size of large neural network models has been growing exponentially since the introduction of transformer (Vaswani et al., 2017) in 2017. However, the GPU memory capacity only shows moderate growth, which creates an increasing gap between model size and GPU memory as illustrated in Figure 1.1. Additionally, a recent study (Ding et al., 2023) indicates the demanding computational requirements have hindered research on large LLMs, as only very limited researches surveyed were able to conduct practical experiments due to the prohibitive costs of deploying large pre-trained language models to validate their hypotheses experimentally.

A model compression technique, known as *model quantization*, has been developed to tackle this challenge and significantly reduce storage requirements. Quantization converts the high-precision numeric representation of large pre-trained models' parameters into compact, low-bit equivalents, resulting in reduced memory consumption and inference speed

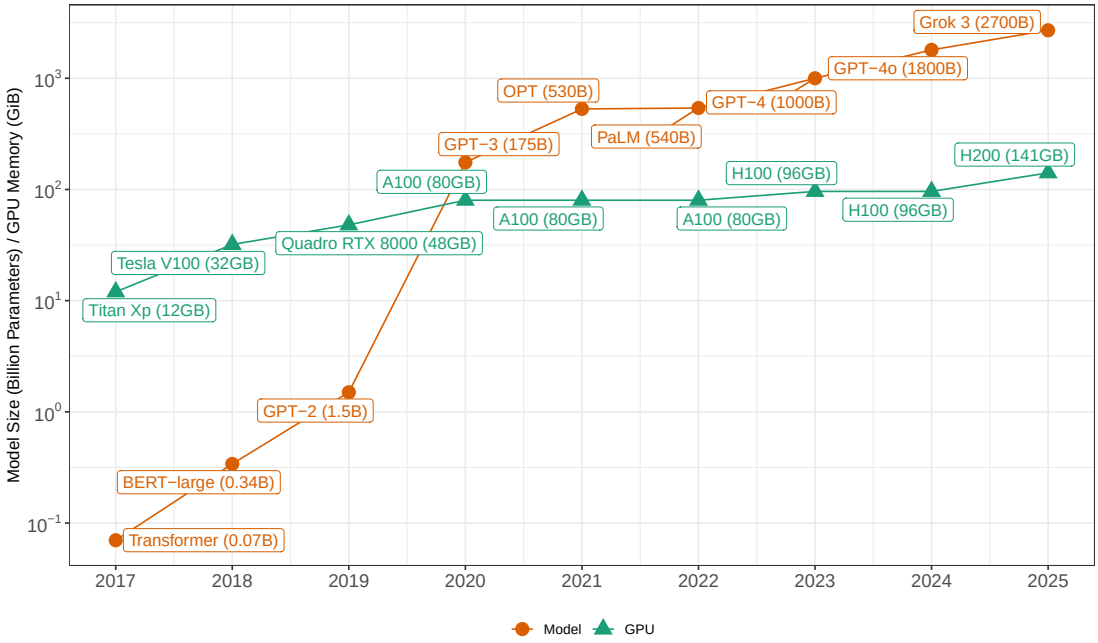


Fig. 1.1 This figure shows the growing gap between GPU memory capacity and size of large neural network models since 2017. The GPUs are the representative mean stream models of the year. The model size of Grok 3 Ultra is projected.

acceleration. As a result, by utilizing quantization techniques, it is possible to run LLMs with billion-scale parameter on consumer-grade GPUs such as the Nvidia GeForce RTX 4080 with a memory capacity under 20 GiB. Although current quantization methods strive to preserve model precision as much as possible, some loss of accuracy is inevitable. It is almost impossible for a quantization method to maintain the exact accuracy of the unquantized model. The quantization research community typically aims for near-lossless compression, commonly considered to be within 1% error relative to the uncompressed baseline, as defined in the MLCommons benchmark (Reddi et al., 2020). Quantization techniques have been extensively studied, leading to the development of numerous innovative methods applied across various use cases, including inference (Frantar et al., 2023; Lin et al., 2023), fine-tuning (Dettmers et al., 2023a; Guo et al., 2024), and optimizer state compression (Dettmers et al., 2022).

1.2 Research Motivations

The key challenge of quantization lies in the *trade-off between reducing the memory requirements and preserving the model performance*. However, existing approaches such as

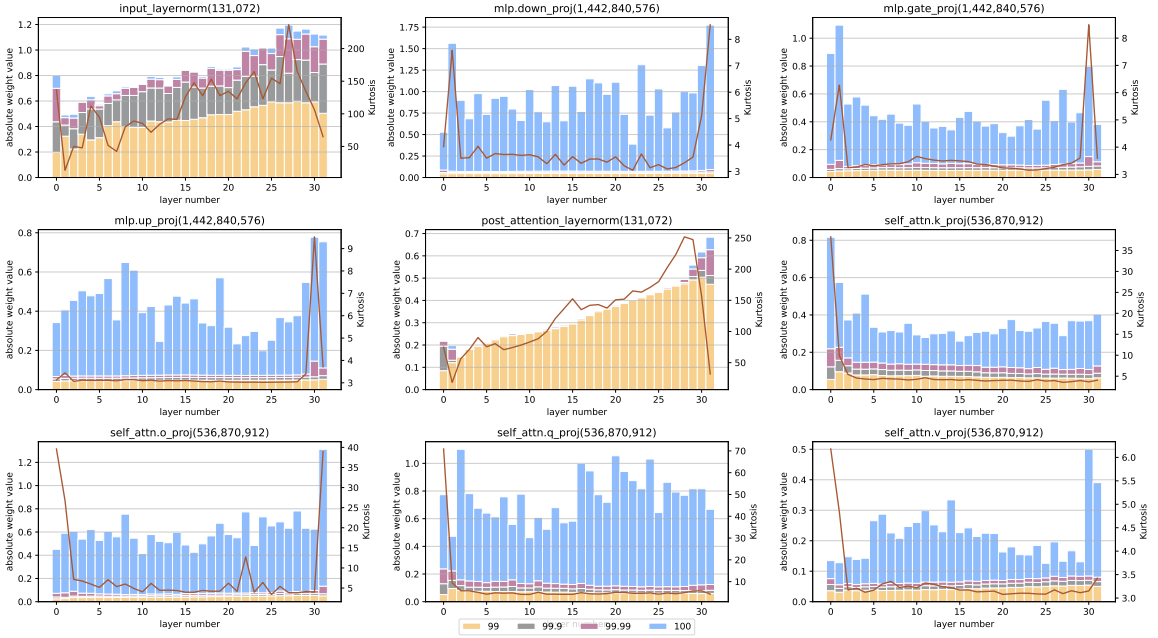


Fig. 1.2 Weight distributions and Kurtosis of the Llama-2-7B model are presented, where the four-colored bars represent the absolute weight values at the 99%, 99.9%, 99.99%, and 100% percentiles, respectively. The line graph, corresponding to the right Y-axis, illustrates the Pearson Kurtosis of each weight matrix. Notably, different layers exhibit distinct patterns in both absolute weight values and Pearson Kurtosis, which informed the development of a novel layer-wise quantization method. For optimal clarity, the figure is best viewed in color and with zoom.

AWQ (Lin et al., 2023), GPTQ (Frantar et al., 2023), BnB (Dettmers et al., 2023a) and HQQ (Badri and Shaji, 2023) often rely on uniform quantization configurations, disregarding the varying quantization difficulty across the layers of large language models (LLMs) as evidenced by Figure 1.2, where the MLP and self-attention modules exhibit significantly distinct layer-wise weight distributions and the extremely high Kurtosis values are present in the first and last layers. This uniformity can lead to sub-optimal memory allocation and reduced quantization accuracy. Furthermore, current methods lack the flexibility to make effective memory-accuracy trade-offs. For instance, state-of-the-art techniques such as AWQ, GPTQ, and HQQ do not allow users to specify arbitrary bit budgets per parameter for LLMs, limiting their adaptability to diverse hardware resource constraints.

This thesis aims to advance the existing quantization methods by introducing greater flexibility in memory-accuracy trade-off and enhancing quantization accuracy. First, it presents a novel quantization scheme, named as MXQ, that optimally allocates memory budgets across layers to maximize quantization accuracy while maintaining overall memory efficiency within a given bit budget. Specifically, the proposed method formulates the

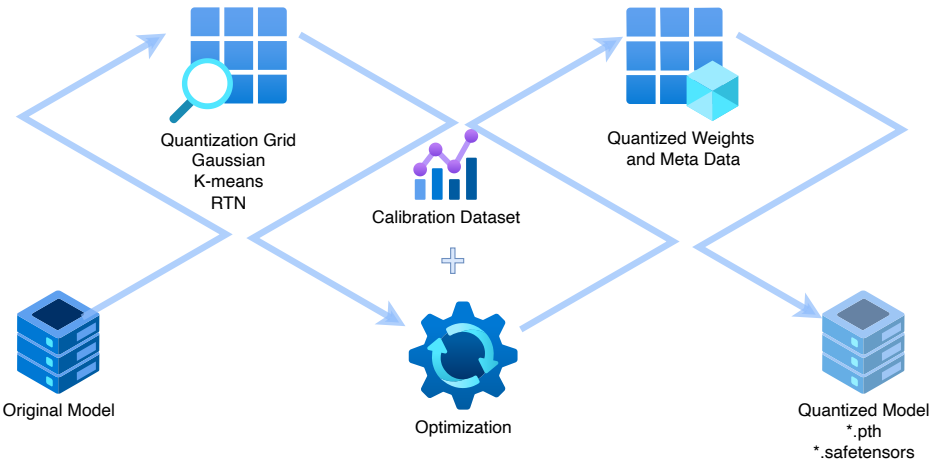


Fig. 1.3 Model quantization converts an original model to the quantized model with several steps: quantization grid, optimisation with calibration dataset, and quantized weights and metadata storage.

search for ideal quantization configurations as a linear programming problem, with memory as constraints and the sum quantization errors as the objective. Second, recognizing the varying memory requirements across layers in large language models, this work leverages layer-sensitive features, such as activation sensitivity to quantization error and Kurtosis of weight distribution, to allocate additional budget to layers critical for achieving higher overall quantization accuracy.

1.3 Quantization Essentials

Quantization represents parameters in lower precision by mapping the original high-precision numeric representation of parameters into a narrower range of values, known as quantization grids or bins, allowing them to be packed into fewer bits (Bai et al., 2022). In the case of 4-bit quantization, a parameter occupies only 4 bits, resulting in a 75% reduction in memory usage compared to the unquantized float16 counterpart. The specific process of converting high-precision representations to their lower-precision counterparts vary among methods. Nevertheless, they generally follow the procedure illustrated in Figure 1.3. The quantization process typically begins with selecting the target linear layers. Subsequently, the weight matrix is fed into the quantization grid to reduce storage size. An optimisation step is usually employed to compensate for quantization errors, which may require additional metrics, such as a calibration dataset, to help adjust the quantized weights and minimize activation errors. After optimisation, adjacent weights are packed into shared bytes to save space. Optionally, the associated metadata go through a process known as double-quantization, where the same

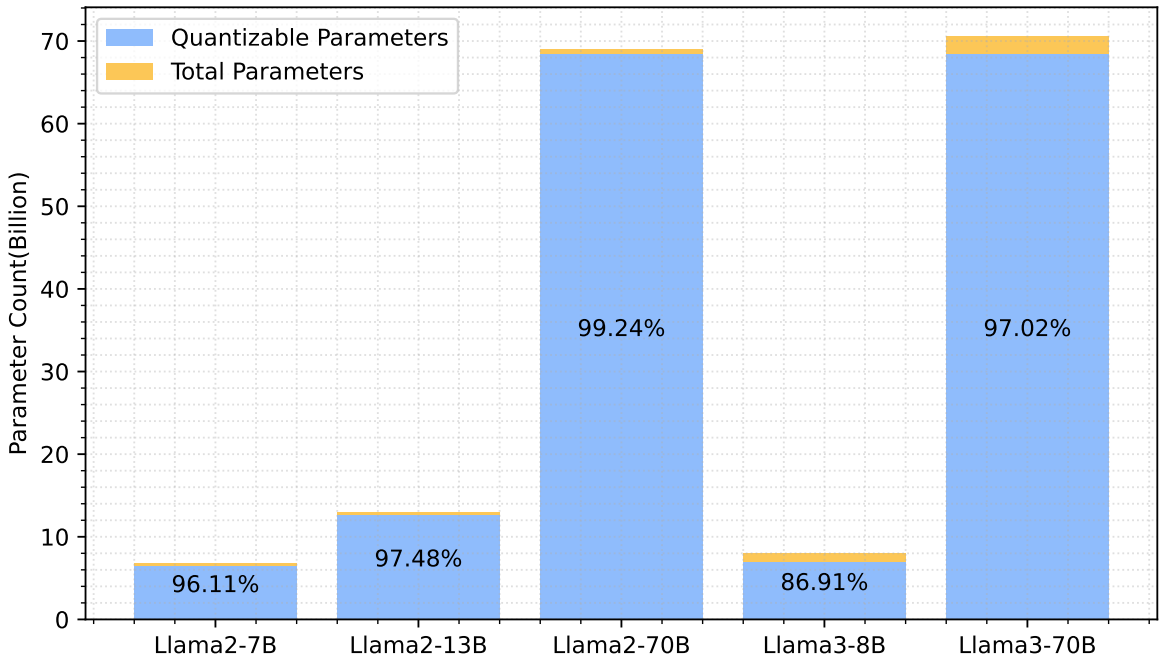


Fig. 1.4 The percentage of quantizable parameters in various Llama models. The quantizable percentage increases as the scale of the model grows. However, the Llama-3 models exhibit slight drops in quantizable percentage compared to their predecessors. Source: the data was extracted from the corresponding models hosted on HuggingFace.

process is applied to the metadata. This is especially useful to further reduce storage when the granular grouping strategy is employed. Finally, the quantized model is saved in persistent storage for distribution. The following sections explain these steps in greater detail.

1.3.1 Module Selection

The process of quantizing a large language model with parameters as large as billions typically starts with the identification of the appropriate layer or module for compression. For example, there are seven linear modules of the Llama-2 family models that are eligible for quantization, *i.e.* the four self-attention modules(*q_proj*, *k_proj*, *v_proj*, *o_proj*) and the three MLP modules(*gate_proj*, *up_proj*, *down_proj*) . However, the embedding module(*embed_tokens*), layer input norm(*input_layernorm*) and layer post attention norm(*post_attention_layernorm*) are excluded from quantization. The embedding module encodes fundamental features of the model's vocabulary which should be kept intact to to preserve the integrity of the embedding tokens. While the input and post-attention norm modules are relatively compact. There is little gain to compress these modules.

The majority parameters of transformer-based large language models are suitable for quantization. As indicated in Figure 1.4, approximately 90% of the parameters of Llama family models are quantizable. The proportion of quantizable parameters varies across Llama model versions. As model scale increases, the percentage of quantizable parameters generally rises. However, Llama-3 models exhibit a slight drop compared to their predecessors.

In short, the general rules to choose quantization targets are **1)** avoid embedding module, **2)** exclude modules with relatively small parameter number, **3)** exclude modules sensitive to weight perturbations.

1.3.2 Quantization Grid

Subsequently, the weights undergo quantization through a grid or bins, resulting in a lower-precision numeric representation, optionally accompanied by de-quantization metadata such as scales or numbers representing zero offsets.

Several techniques exist for selecting the quantization grid:

- Rounding to Nearest (RTN) (Dettmers et al., 2022), which divides the number of quantization bins evenly and maps the original weights to the bins by applying a linear transformation and rounding to the nearest value.
- Quantile quantization (Dettmers et al., 2023a), which employs a non-uniform quantization grid in accordance with statistical distribution, typically a Gaussian distribution, thereby ensuring that most parameters are closer to the grid than an evenly sized grid.
- Cluster centroids (Kim et al., 2023), which clusters weights into groups by leveraging the K-means classification algorithm and utilizes the centroids of these groups as the quantization grid.

1.3.3 Error Compensation

The aforementioned quantization transformation inevitably introduces errors, resulting in model accuracy degradation. To mitigate this problem, most quantization methods include an optimisation stage. The specific optimisation algorithm employed by various quantization approaches varies widely. Common optimisation strategies include adjusting weights to minimize activation errors with a small calibration dataset, identifying outliers by magnitude or activation, and managing outliers using sparse matrices or clipping. According to (Dettmers et al., 2023b), LLM generation is a multi-stage process where even tiny perturbations can

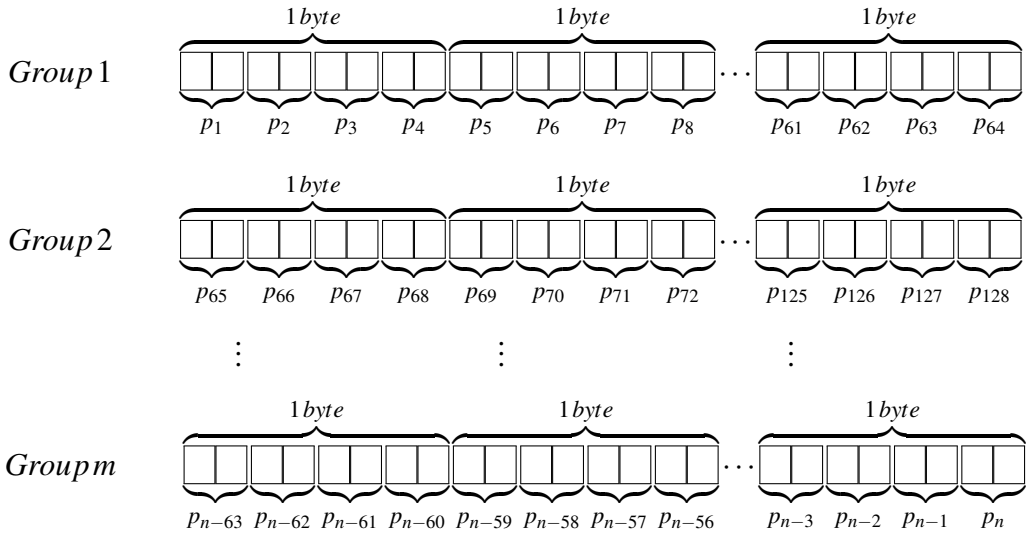


Fig. 1.5 This figure illustrates how quantized parameters are packed into bytes with a group size of 64 in 2-bit quantization (denoted as $b_1 = 2, g_1 = 64$ or simply *b2g64*). The small square boxes represent the bits used to store the quantized parameters. In this scheme, a total of n parameters are divided into $m = n/64$ groups, with each quantized parameter occupying 2 bits. To optimize memory storage, 4 successive parameters are packed into a single byte.

quickly accumulate, leading to significantly erroneous outputs. Thus, minimizing quantization errors by focusing on activation often yields better accuracy. However, this method is generally slow and depends on a carefully curated calibration dataset to perform well.

1.3.4 Bit Packing

Once the quantized parameters are optimized, it is necessary to process the associated quantization metadata and compact the lower-bit parameters to save space. In some scenarios, especially when granular groups are used to localize the weight value range, there can be substantial metadata which should be also compressed to control overall memory usage. These metadata may be quantized without optimisation, as they exhibit low variance in value distribution. When parameters and quantization metadata are converted into lower-bit representation, adjacent lower-bit weights are packed together into shared byte(s). Figure 1.5 and Figure 1.6 illustrate the how parameters are packed into shared byte(s) for 2-bit and 3-bit per parameter respectively. The 3-bit packing scheme illustrated in Figure 1.6 is employed by HQQ, which packs 10 3-bit parameters into 4 bytes, leaving first two bits unused. This results in an additional storage overhead of 0.2 bit per parameter.

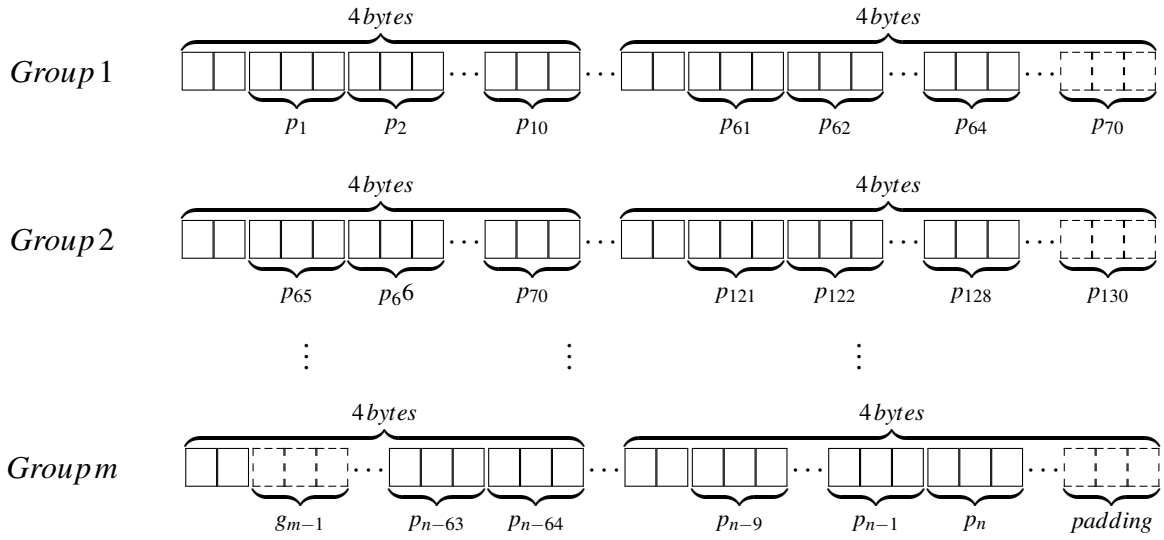


Fig. 1.6 This figure illustrates the packing scheme of parameters quantized with 3-bit and a group size of 64 (denoted as $b_1 = 3, g_1 = 64$ or simply *b3g64*) into 4 bytes. The small square boxes represent the bits used to store the quantized parameters. The square with dashed border is not part the of the group. In this scheme, a total of n parameters are divided into $m = n/64$ groups, with each quantized parameter occupying 3 bits. To optimize memory storage, 10 successive parameters are packed into 4 bytes, leaving 2 bit unused.

1.3.5 Persistence

Upon successful completion of the previous stages, the last step of the quantization procedure is to save the model's state dictionary into persistent storage for distribution, which is also known as checkpoint. There are two widely adopted formats of quantized LLMs: pickled Python object (Pilgrim, 2009) and safetensors (Huggingface, 2022). While the pickle format is flexible, it opens door to hide malicious code in the LLM, threatening the safety LLM supply chain. On the other hand, the safetensors is specifically designed to contain such security threats. Therefore, it is always recommended distribute LLMs using safetensors format. Finally, a less commonly noted benefit of quantization is the reduction in the model's persistent storage requirements. Experiments of this research indicate that with 4-bit quantization, the model file size can be reduced by approximately 65%.

1.4 Research Questions

This thesis aims to determine whether there are novel approaches that offer flexibility in choosing a balanced quantization strategy or enhance quantization accuracy with or without limited extra memory budget. Specifically, this thesis seek to answer the following questions:

1. Can memory usage be reduced by a certain percentage while maintaining accuracy loss within a given range by balancing memory-accuracy trade-off?

2. Can quantization accuracy improvement be achieved by prioritizing budget allocation for sensitive layers under strict or relaxed memory constraint?

1.5 Research Method

This study starts with a focused literature review of the post training quantization (PTQ) methods, with an emphasis on the weight-only and data-free approaches. The study is primarily empirical and quantitative as it employs data analysis and experiment to make conclusions. For example, extensive data analysis is leveraged to select features to quantify layer-sensitivity. In addition, comprehensive experiments are conducted to measure perplexity, quantization speed and memory efficiency to evaluate performance of the proposed methods. The methodologies involves developing and testing the new quantization schemes, benchmarking them on selected LLMs, and analyzing their performance compared to existing methods.

1.6 Evaluation Method

Following the common practice, this thesis validates the efficacy of the proposed method using perplexity as the primary evaluation method. The perplexity metrics are measured on WikiText-2 (Merity et al., 2016) and C4 (Raffel et al., 2020). The baselines include both calibration-based and calibration-free quantization methods. Specifically, HQQ (Badri and Shaji, 2023), BnB (Dettmers et al., 2023a), GPTQ (Frantar et al., 2023) and AWQ (Lin et al., 2023) are re-evaluated using the same models and experiment environment to ensure a fair comparison. Additionally, the quantization speed, actual memory consumption are also benchmarked to evaluate the quantization methods comprehensively. The Llama family models are selected for experiment as they are representative transformer-based LLMs which are free for research purpose.

1.7 Contributions

The contributions of this thesis are summarized as follows:

- Introduction of a novel quantization method named MXQ for precise budget control and flexible memory-accuracy balance. This method minimizes the sum of quantization errors in the objective while incorporates a storage cost function as constraint. This formulation is efficiently solved using a mixed-integer linear programming (MiLP) algorithm, enabling fast and effective quantization.
- Revealing the robustness of activation sensitivity within a family of models and their fine-tuned variants by exploring multiple transformer-based LLM families empirically.
- Design and implementation of a simple and effective outlier detection algorithm to discover sensitive layers with sensitive scores or Kurtosis metrics.
- Introduction of SensiBoost and KurtBoost methods based on the sensitivity score and Kurtosis metrics, resulting in superior performance over HQQ baseline with only 2% increment in memory budget.

1.8 Thesis Structures

The rest of the thesis is organized as follows:

Chapter 2 reviews existing literatures pertinent to LLM quantization, including calibration-based and calibration-free PTQ approaches.

Chapter 3 introduces the preliminary techniques such as transformer architectures, linear programming, which provide the theoretical foundation for the successive research.

Chapter 4 elaborates the MXQ method and explains the experiments result.

Chapter 5 presents the SensiBoost, KurtBoost, SensiMiLP and KurtMiLP methods which leverage the layer-sensitive features such as activation sensitivity and Kurtosis to optimize quantization accuracy under relaxed and strict memory constraint respectively.

Chapter 6 concludes the thesis, summarizes the limitation and proposes the future works.

Chapter 2

Literature Review

2.1 Overview

The Quantization techniques have been extensively explored, and numerous innovative methods were invented and applied in a variety of use cases, such as inference (Frantar et al., 2023; Lin et al., 2023), fine-tuning (Dettmers et al., 2023a; Guo et al., 2024) and optimizer state (Dettmers et al., 2022). These techniques can be generally divided into two categories: **(1)** Quantization Aware Training (QAT) (Nagel et al., 2021), requiring backward propagation and being tightly coupled with model training, and **(2)** Post Training Quantization (PTQ) (Nagel et al., 2019), which is a training-free methodology. PTQ has been recognised as the mainstream due to the existence of numerous pre-trained LLMs. For example, there are around 700,000 models published as of June 2024 on the HuggingFace, the largest AI model hosting site. Therefore, advancements in PTQ research can contribute to democratize AI capabilities to a wider community.

This thesis focuses on a specific class of the PTQ method, *i.e.*, weight-only quantization method, particularly those closely associated with the proposed mixed-quantization strategy, which can serve as the underlying quantization implementation. Specifically, considering whether an extra calibration dataset is adopted during quantization, weight-only method can be further divided into two categories: calibration-based method and calibration-free method. The calibration-free method lends itself to generalize to diverse model architecture and generally takes less time to quantize a model than calibration-based method. The following section surveys the works most relevant to proposed mixed-quantization method.

2.2 Calibration-based Methods

The calibration-based approaches typically leverage well-curated dataset as input to minimize the activation errors introduced by quantization based on mathematical tools such as the second derivatives of the loss function with respect to the weights, *i.e.* Hessian matrix or Fisher information. They usually produce better-quantized models. However, they tend to be computation-intensive and prone to over-fitting the calibration dataset. The representative state-of-the-art implementations of calibration-based approaches include GPTQ and AWQ.

2.2.1 GPTQ

GPTQ (Frantar et al., 2023) is a layer-wised quantization method which searches best quantized weight $\widehat{\mathrm{W}}$ with respect to the input X from the calibration dataset according to objective function defined in the Equation 2.1.

$$\underset{\widehat{\mathrm{W}}}{argmin} \|\mathrm{WX} - \widehat{\mathrm{W}}\mathrm{X}\|_2^2 \tag{2.1}$$

GPTQ leverages the Optimal Brain Quantizer (OBQ) (Frantar and Alistarh, 2022), which quantized one weight at a time while constantly updating all not-yet-quantized weights to compensate for the error incurred by quantizing a single weight as formulated in Equation 2.2.

$$\begin{aligned} w_q &= \underset{w_q}{argmin} \frac{(quant(w_q) - w_q)^2}{[\mathrm{H}_F^{-1}]_{qq}} \\ \delta_F &= -\frac{w_q - quant(w_q)}{[\mathrm{H}_F^{-1}]_{qq}} \cdot (\mathrm{H}_F^{-1})_{:,q} \end{aligned} \tag{2.2}$$

The updated inverse is given by the Equation 2.3:

$$\mathrm{H}_{-q}^{-1} = \left(\mathrm{H}^{-1} - \frac{1}{[\mathrm{H}^{-1}]_{qq}} \mathrm{H}_{:,q}^{-1} \mathrm{H}_{q,:}^{-1}\right)_{-q} \tag{2.3}$$

By applying quantizing weight column-wise, lazy update and cholesky reformulation, GPTQ reduces OBQ's algorithm complex substantially, making it scale to larger models with hundred-billion parameters. For instance, GPTQ is capable of quantizing the 170-billion scale models such as OPT-175B and BLOOM-176B to 3 or 4 bits per parameter in approximately 4 hours with GPU, enabling generative inference on 175 billion-parameter models inside a single GPU. In addition, GPTQ implements specialized GPU kernels to accelerate memory loading, resulting in 3 to 4 times inference speedup on various GPU models.

However, despite the effective improvements over previous methods, GPTQ's update of remaining full-precision weights incurs substantial computation, resulting in significantly slower quantization speed. The choice of squared error as loss function seems less effective to preserve outliers in the weights. The dependency on calibration dataset makes it difficult to generalize to heterogeneous model architectures such as multi-modal models.

2.2.2 AWQ

AWQ (Lin et al., 2023), based the observation that the importance of LLM's weights is non-uniform, proposes a quantization method to identify the tiny fraction of "salient" weights by measuring activation magnitude and pre-scaling the weights with a per-channel factor s to minimize quantization errors according to the objective function as denoted in Equation 2.4:

$$\begin{aligned} s^* &= \underset{s}{argmin}\,\mathscr{L}(s) \\ \mathscr{L}(s) &= \|Q(\mathrm{W}\cdot s)(s^{-1}\cdot \mathrm{X}) - \mathrm{WX}\| \end{aligned} \tag{2.4}$$

where the qunatization function $Q(\mathrm{W})$ is defined as:

$$Q(\mathrm{W}) = \Delta \cdot Round(\frac{\mathrm{W}}{\Delta}), \qquad \Delta = \frac{max(|\mathrm{W}|)}{2^{n-1}} \tag{2.5}$$

Since the loss function in Equation 2.4 is not differentiable, AWQ leverages a simple search space, where α is confined to the interval $[0,1]$, as defined in Equation 2.6 to find the optimal scale s.

$$s = sx^{\alpha}, \qquad \alpha^* = \underset{\alpha}{argmin}\,\mathscr{L}(sx^{\alpha}) \tag{2.6}$$

This approach unifies the treatment of the salient and non-salient weights, eliminating the need to isolate salient weights into separate storage like sparse matrix, and develop specialised mixed-precision matrix multiplication kernel for fast inference. Besides significant memory reduction, AWQ achieves approximately 3 times inference speedup compared to the FP16 implementation by Huggingface across a wide range of LLMs.

AWQ is an important baseline for recent quantization researches such as HQQ(Badri and Shaji, 2023), SqueezeLLM(Kim et al., 2023) etc. Moreover, AWQ has been widely adopted by diverse open-source LLM serving frameworks such as FastChat, vLLM, HuggingFace TGI, LMDeploy, etc. which facilitates practical industrial applications of AWQ.

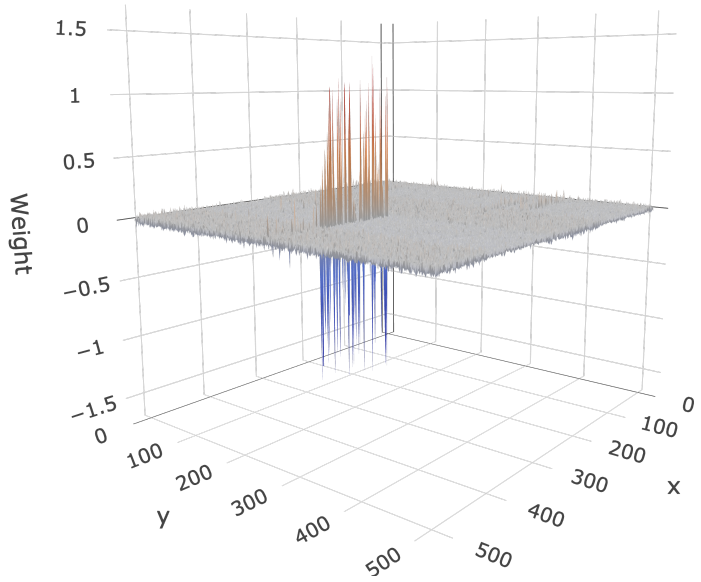


Fig. 2.1 This figure illustrates needle-like spikes piercing through the thin plane formed by the vast majority of typical parameters, indicating the presence of prominent outliers in the *self_attention.o_proj* module at the 32nd layer of the Llama-2-7B model.

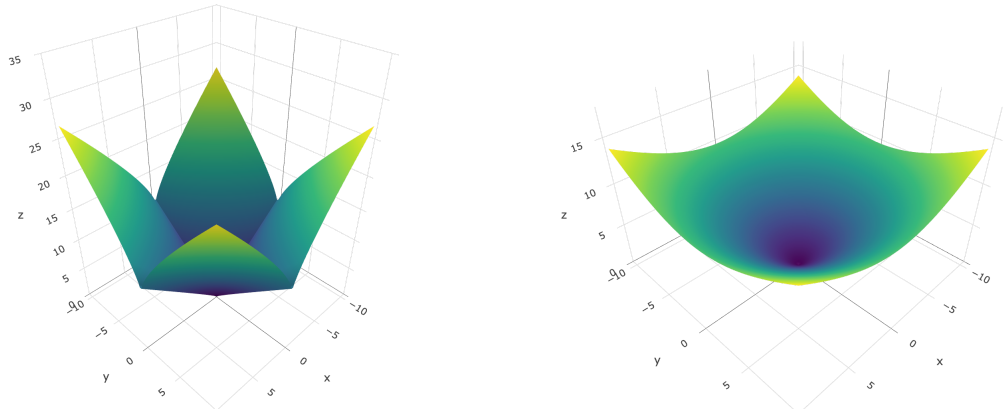
2.3 Calibration-free Methods

2.3.1 HQQ

HQQ (Badri and Shaji, 2023) approaches minimizing quantization errors by relying solely on the weight without considering the layer activation. It identifies quantization parameters zero-point z and scaling s to minimize the $L_{p<1}$-norm between the original weights W and their dequantized counterpart as denoted in Equation 2.7.

$$\underset{z,s}{argmin}\phi(\mathrm{W} - Q_{z,s}^{-1}(Q_{z,s}(\mathrm{W}))) \tag{2.7}$$

The incorporation of the $L_{p<1}$-norm in the loss function $\phi()$ enables HQQ to model outliers effectively through a hyper-Laplacian distribution, which captures the long-tailed nature of outlier more accurately than the conventional squared error. This is particularly useful to deal with salient outliers presented in Llama family model as illustrated in Figure 2.1.



(a) This figure illustrates the visualization of the two-variable $L_{p=0.7}$-norm function as a surface in 3D space. This L_p-norm is employed by HQQ to preserve outliers in weights of LLMs.

(b) This figure illustrates the visualization of the two-variable $L_{p=2}$-norm function as a surface in 3D space.

Figure 2.2a illustrates the $L_{p=0.7}$-norm, the default loss function of HQQ, as a 3D surface. Figure 2.2b shows the $L_{p=2}$-norm in 3D. The $L_{p<1}$-norm makes the loss function $\phi()$ non-convex. Therefore HQQ converts the optimisation of the non-convex loss function $\phi()$ formulated in Equation 2.7 to a new formulation denoted in Equation 2.8 so that it can leverage the Half-Quadratic solver (Geman and Reynolds, 1992).

$$\underset{z,\mathrm{W}_e}{argmin}\phi(\mathrm{W}_e)+\frac{\beta}{2}\left\|\mathrm{W}_e-(\mathrm{W}-Q_z^{-1}(Q_z(\mathrm{W})))\right\|_2^2 \tag{2.8}$$

By utilizing alternate optimisation, the Equation 2.8 is decomposed into two sub-problems as illustrated in Equation 2.9.

$$\begin{aligned}
\mathrm{W}_e^{(t+1)} &\leftarrow \underset{\mathrm{W}_e}{argmin}\phi(\mathrm{W}_e)+\frac{\beta^{(t)}}{2}\left\|\mathrm{W}_e-(\mathrm{W}-Q_z^{-1}(Q_z(\mathrm{W})))\right\|_2^2 \quad (sp_1)\\
z^{(t+1)} &\leftarrow \underset{z}{argmin}\frac{1}{2}\left\|Q_z^{-1}(Q_z(\mathrm{W}))-(\mathrm{W}-\mathrm{W}_e^{(t+1)})\right\|_2^2 \quad (sp_2)\\
\beta^{(t+1)} &\leftarrow k\beta^{(t)}
\end{aligned} \tag{2.9}$$

When $L_{p<1}$-norm is the loss function, the solution to first sub-problem(sp_1) is the generalized soft-thresholding operator(Badri and Yahia, 2016) as illustrated in Equation 2.10.

$$\begin{aligned} \mathrm{W}_e^{(t+1)} &\leftarrow shrink_{l_p}(\mathrm{W} - Q_z^{-1}(Q_z(\mathrm{W})), \beta) \\ shrink_{l_p}(x, \beta) &= sign(x) relu\Big(|x| - \frac{|x|^{p-1}}{\beta}\Big) \end{aligned} \tag{2.10}$$

The second sub-problem(sp_2) can be converted to the Equation 2.11. The solution, as presented in Equation 2.12, is the average over the axis the quantization grouping is carried out.

$$\begin{aligned} z^{(t+1)} &\leftarrow \underset{z}{argmin} \frac{1}{2} \Big\| z - (\mathrm{W}_q^{(t+1)} - \frac{\mathrm{W} - \mathrm{W}_e^{(t+1)}}{s}) \Big\|_2^2 \\ \mathrm{W}_q^{(t+1)} &= round(\mathrm{W}/s + z^{(t)}) \end{aligned} \tag{2.11}$$

$$z^{(t+1)} \leftarrow \Big\langle \mathrm{W}_q^{(t+1)} - \frac{\mathrm{W} - \mathrm{W}_e^{(t+1)}}{s} \Big\rangle \tag{2.12}$$

HQQ provides competitive performance compared to the state-of-the-art quantization methods. Additionally, HQQ exhibits extraordinary quantization speed. Experiment shows that HQQ is approximately an order of magnitude faster than the state-of-the-art calibration-based methods such as AWQ and GPTQ. In addition, HQQ offers an abundance of options to further optimize the quantization with a wide range of bits to quantize large neural network models. Available bit choices include 2, 3, 4 and 8. It also allows configurable group size, secondary bit and group size for meta data quantization. Furthermore, by adopting the calibration-free approach, HQQ avoids potential over-fitting to calibration dataset, making it model architecture-agnostic and generalize to not only diverse transformer based large language models but also multi-modal models.

2.3.2 BitsAndBytes

Weights in pre-trained neural network model exhibit a zero-centered normal distribution with standard deviation σ. In this case, dividing quantization bins evenly is sub-optimal since it increases the density of the bins close to zero, leading to larger quantization errors. The BitsAndBytes (BnB)(Dettmers et al., 2023a) employs a novel high-precision technique to solve this problem by creating quantization bins using quantiles, a method known as Quantile Quantization(Dettmers et al., 2021). This ensures each bin contains identical number of weights and the bins around zero are narrower to reduce overall quantization error. The crucial parts of BnB quantization scheme include the searching for the k-bit quantiles, known as k-bit NormalFloat, along with scaling and mapping weights into k-bit integers.

Table 2.1 This table presents the values of the NormalFloat4 data type.

Part	Integer	Float	Integer	Float
Negative	-7	-1.0	-6	-0.6961928009986877
	-5	-0.52507305145263670	-4	-0.39491748809814453
	-3	-0.28444138169288635	-2	-0.18477343022823334
	-1	-0.09105003625154495		
Zero	0	0.0		
Positive	1	0.07958029955625534	2	0.16093020141124725
	3	0.24611230194568634	4	0.33791524171829224
	5	0.44070982933044434	6	0.56261700391769410
	7	0.72295683622360230	8	1.0

BnB determines the k-bit NormalFloat, a set of pre-computed constants, by calculating the 2^k+1 quantiles of a theoretical $\mathcal{N}(0,1)$ Gaussian distribution using cumulative distribution function(shown in Equation 2.13) to obtain the k-bit quantiles. The quantiles are normalized into the $[-1,1]$ range. As a special case, the 4-bit BnB, commonly known as NormalFloat4, is a datatype representing 16 values $(q1,q2,\cdots,q16)$ in the $[-1,1]$ interval. Table 2.1 presents the 16 values and their corresponding integers.

$$\Phi(x) = P(Z \leq x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{u^2}{2}} du \tag{2.13}$$

When it comes to quantizing the weights of LLM, BnB begins by dividing the weight matrix into small groups for better quantization accuracy. It then rescales the weights into the $[-1,1]$ range using absolute maximum scaling and maps them to the nearest k-bit NormalFloat to obtain the k-bit integer representations. The scaling constants for the groups are stored as meta data. In order to minimize the storage overhead for these scaling constants incurred by the fine-grained group sizes, BnB further quantizes the high-precision scaling constants into k-bit integers. This double quantization technique is a widely adopted strategy by other state-of-the-art quantization methods.

2.3.3 Comparison of Quantization Methods

Table 2.2 presents the comparison of the four state-of-the-art quantization methods, *i.e.* BnB, GPTQ, AWQ and HQQ, reviewed in this chapter. Notably, HQQ is a balanced method that combines speed, accuracy and multiple bits support. These methods have been adopted extensively in the industry, demonstrating their practicality and efficacy. Nevertheless, the limitation of these methods is worth discussion. First, quantization methods such as GPTQ

Table 2.2 This table presents the comparison of the quatization methods reviewed in this chapter.

Item	AWQ	GPTQ	HQQ	BnB
Calibration Data	required	required	no	no
Quantization Speed	slow	very slow	fast	very fast
Quantization Accuracy	Excellent	Good	Good	Good
Supported Bits	4	2,3,4,8	2,3,4,8	4,8
Fast Inference Extensions	autoawq-kernels	exllamav2 marlin	Torchao marlin	bitsandbytes

and AWQ requires curated calibration dataset, making it challenging to generalize these method to other large neural network such as vision models which are trained on a mixture of textual and image data. Given the substantial architectural disparities and diverse choices of training datasets for these multi-modal models, curating compatible calibration dataset is definitely a maintenance headache. Second, calibration dependent approaches tend to rely on GPU to perform the quantization as a full inference pass is indispensable to measure the quantization error in term of activation. This prevents offloading the quantization task to CPU, which is more cheaper and accessible. Additionally, the quantization speed of calibration dataset dependent methods like AWQ and GPTQ are relatively slow. For instance, the GPTQ method takes approximately 4 GPU hours to quantize the OPT-175B or BLOOM-176B models (Frantar et al., 2023). Finally, the four quantization methods surveyed in this chapter employ uniform quantization configurations across entire model, which may be sub-optimal to address varying difficulty across diverse layers of billion-scale LLMs.

Therefore, research on fast and flexible quantization approaches is crucial to serving large neural network models economically and generalizing the quantization method to various model architectures and generations within a given model family. Motivated by this, the thesis aims to enhance the existing quantization methods by introducing quantization schemes that offer flexible memory-accuracy balance.

Chapter 3

Preliminary – Related Techniques

3.1 Overview

This chapter explains the foundation of linear programming, the basic methods of quantization grid and the transformer-based Llama architecture. They provide context and support the approaches introduced by this thesis, being a crucial part to understand the methodologies adopted in subsequent chapters.

Starting with Linear Programming, this chapter introduces the key enabler to the quantization approaches proposed by this thesis, provides theory framework and solvers to formulate and implement the MXQ quantization method efficiently. With Linear Programming, especially Mixed integer Linear Programming (MiLP), the quantization problem can be modelled as resource allocation optimisation problem with objective as maximizing model accuracy, memory budget as constraint.

The section that follows Linear Programming is the explanation of three major quantization grid methods, *i.e.* Rounding to Nearest, Quantile Quantization, K-means cluster centroid. This section provides essential knowledge and common terminologies that are referenced in the following chapters. The Rounding to Nearest method is adopted to calculate sensitivity score in later chapter.

The last section describes the Llama-2 architecture, a typical transformer-based large language model architecture, explaining the computation flow of the inference, which is crucial to understand the method to quantify and measure the layer-wise sensitivity to quantization error discussed in Chapter 5.

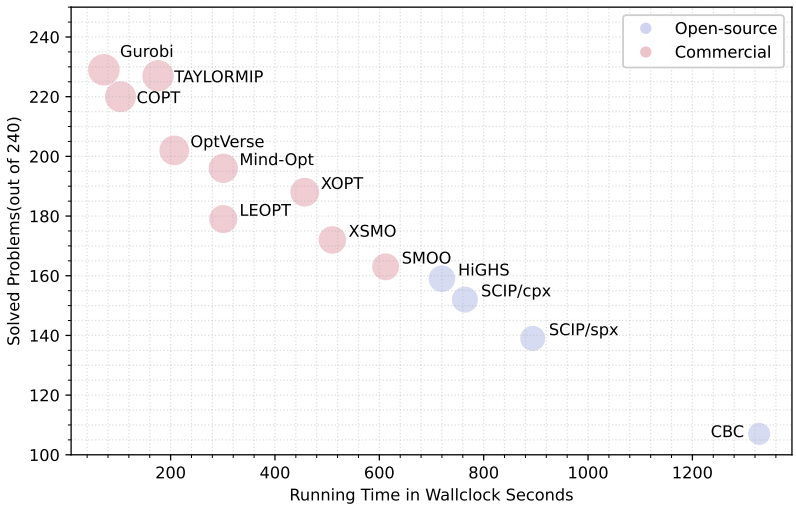


Fig. 3.1 Results of MIP2017 benchmark. The X axis denotes the wall clock time in seconds spent on the solving the problems. The Y axis shows the problems solved out of 240. The points close to upper left corner indicate superior performance. The commercial solvers lead in both capability and speed. The Gurobi is the top 1 solver. While the HiGHS performs best among the open-source solvers.

3.2 Linear Programming

3.2.1 Brief History

Linear Programming solvers have a long history of development. Driven by the needs from military operations such as logistics and resource allocation, the modern LP work, characterized by the introduction of the simplex algorithm (Dantzig, 2002), was pioneered by George Dantzig in 1947 (Bixby, 2012). It remains a crucial computational tool of LP and MIP of today. Linear Programming solvers evolve as computers grow increasingly powerful since then. In the 1970s, with the introduction of the IBM 360 series of computers, not only LP problems could be solved faster, but also new ideas could be tried which were infeasible previously. From the late 1980s to early 2000s, LP landscape saw dramatic improvements as large as over six orders of magnitude. Apart from the hardware improvements, the primary contributing factor is the enhanced algorithm which is approximately 3300 more efficient (Bixby, 2012). In late 2000s, newer LP solvers such as Gurobi, COPT, HiGHS came into existence, which further enhance the performance and offer better scalability and flexible programming interface. To evaluate the existent and emerging LP solvers objectively, benchmarks such as MIPLIB2017 (Gleixner et al., 2021) were developed. (Hans, 2024)

maintains a set of comprehensive benchmark results of LP solvers. Figure 3.1 shows the latest MIPLIB2017 (Gleixner et al., 2021) benchmarks[1] conducted on various leading commercial and open-source MIP solvers.

3.2.2 Linear Programming formulation

Optimisation involves solving complex decision or allocation problems by selecting values for a set of related variables under given constraints, with the goal of maximizing or minimizing a single objective that quantifies the quality of the decision. The optimisation problem can be generalized in mathematical formulation as in Equation 3.1 (Boyd and Vandenberghe, 2004):

$$\begin{aligned} min \quad & f_0(x) \\ s.t. \quad & f_i(x) \leq b_i, i = 1, \cdots, m \\ where: \quad & x = (x_1, \cdots, x_n), f_0 : \mathbb{R}^n \to \mathbb{R}, f_i : \mathbb{R}^n \to \mathbb{R} \end{aligned} \tag{3.1}$$

In Equation 3.1, the vector $x = (x_1, \cdots, x_n)$ is the optimisation variable. The function f_0 is the objective function, while the functions $f_i, i = 1, \cdots, m$ are constraint functions. The constraint $b_1, \cdots, b_m$ are the bounds for the constraints (Bixby, 2012). To solve Equation 3.1, an optimal vector x^* is to be identified, resulting in minimal objective value among all vectors that satisfy the constraints.

A linear programming (LP) is a type of optimisation problem in which the objective can be described as a linear function of the decision variables and the constraints are linear equalities or linear inequalities of the decision variables (Boyd and Vandenberghe, 2004; Luenberger and Ye, 2021) as formulated in Equation 3.2.

$$f_i(\alpha x + \beta y) = \alpha f_i(x) + \beta f_i(y) \tag{3.2}$$

The general formulation of LP problems can be expressed in the Equation 3.3 as follows:

$$\begin{aligned} min \quad & c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ s.t. \quad & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n && = b_1 \\ & a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n && = b_2 \\ & \vdots && \vdots \\ & a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n && = b_m \\ where: \quad & x_1 \geq 0, x_2 \geq 0, \cdots, x_n \geq 0, \end{aligned} \tag{3.3}$$

[1] Source: https://plato.asu.edu/ftp/milp.html

Or in more succinct vector notation, the LP problem can be formulated as:

$$\begin{aligned} &min && c^T x \\ &s.t. && Ax = b \quad and \quad x \geq 0 \end{aligned} \tag{3.4}$$

In Equation 3.4, the decision vector x is an n-dimensional column vector. The objective coefficient data is represented by the vector c, which is also an n-dimensional column vector. The matrix A imposes m equality constraints to the decision vector x. The inequality $x \geq 0$ mandates each element of x as non-negative.

There are other forms of LP math expression. However, they are mathematically equivalent to the standard form as formulated in Equation 3.3 and 3.4. By introducing *Slack Variables*, *Surplus Variables* or *Free Variables*, other LP formulations can be converted to the standard form (Luenberger and Ye, 2021). For example, the Equation 3.5 with inequality constraints can be converted to the standard form using extra slack variables, as formulated in the Equation 3.6.

$$\begin{aligned} max \quad & \begin{pmatrix} c_1, c_2, \cdots, c_n \end{pmatrix}_{1\times n} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}_{n\times 1} \\ s.t. \quad & \begin{pmatrix} a_{11}, a_{12}, \cdots, a_{1n} \\ a_{21}, a_{22}, \cdots, a_{2n} \\ \vdots \\ a_{m1}, a_{m2}, \cdots, a_{mn} \end{pmatrix}_{m\times n} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}_{n\times 1} \leq \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}_{m\times 1} \quad and \quad \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}_{n\times 1} \geq 0 \end{aligned} \tag{3.5}$$

$$
\begin{aligned}
\textit{min} \quad & -\left(c_1, c_2, \cdots, c_n\right)_{1\times n} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}_{n\times 1} \\
\textit{s.t.} \quad & \begin{pmatrix} a_{11}, a_{12}, \cdots, a_{1n}, \overbrace{1, 0, \cdots, 0}^{m} \\ a_{21}, a_{22}, \cdots, a_{2n}, \overbrace{0, 1, \cdots, 0}^{m} \\ \vdots \\ a_{m1}, a_{m2}, \cdots, a_{mn}, \overbrace{0, 0, \cdots, 1}^{m} \end{pmatrix}_{m\times(n+m)} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ x_{n+1} \\ \vdots \\ x_{n+m} \end{pmatrix}_{(n+m)\times 1} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}_{m\times 1} \\
\textit{where} \quad & \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}_{n\times 1} \geq 0
\end{aligned}
\tag{3.6}
$$

In the standard form Equation 3.6, the equality constraint matrix is composed of the original matrix and the $m \times m$ identity matrix. Mixed Integer Linear Programming, short for MIP or MILP, is subset of LP, where the decision variables are a mix of continuous or integer variables.

3.3 Quantization Grid

This section elaborates the common quantization grid methods such as Rounding to Nearest (RTN), Quantile Quantization and K-means. Figure 3.2 presents how the three different quantization grids divide the first 256 parameters of the self attention output projection module in the 32th layer of the Llama-2-7B model. As demonstrated by Figure 3.2, the RTN employs the uniform quantization grid which requires less computation resource. On the other hand, the grids of K-means and quantile quantization are divided non-uniformly, where the grids are more dense at the center, indicating significant amount of values are close to zero.

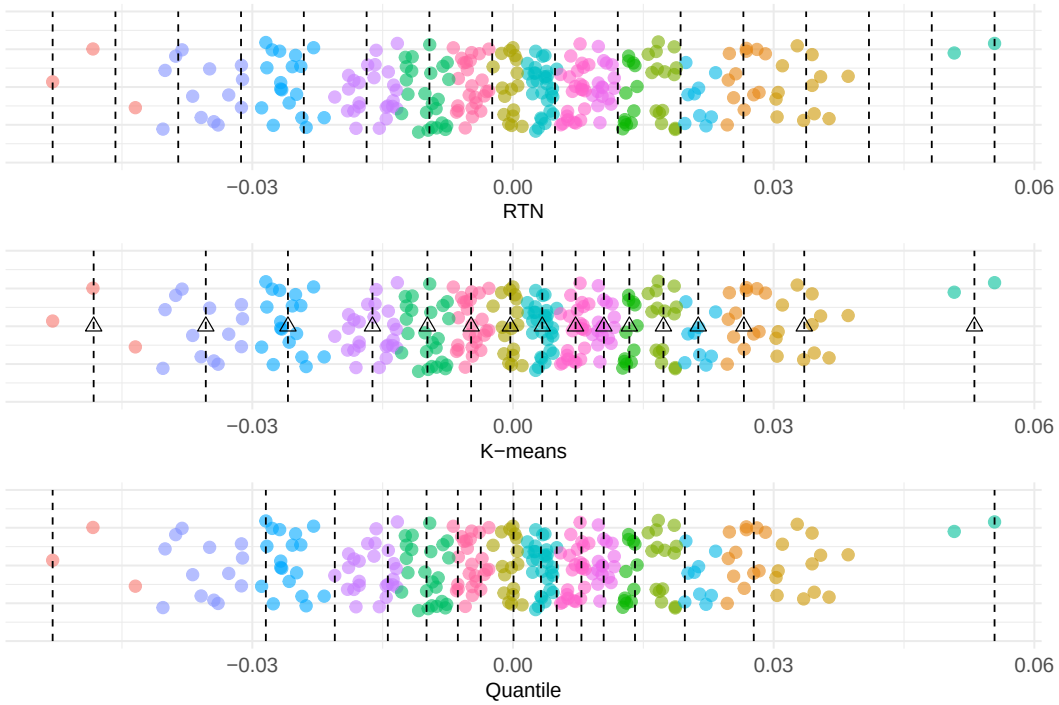


Fig. 3.2 This figure demonstrates how the three different quantization grids divide the first 256 parameters of the self attention output projection module in the 32th layer of the Llama-2-7B model. To better visualize these one-dimensional parameters, they are jittered in the y-axis to separate them apart. The hallow triangles represent the centroids of the clusters in the K-means method.

3.3.1 Rounding to Nearest

Rounding to Nearest (RTN) divides the range of values into evenly spaced intervals. It can be implemented in either a symmetric or asymmetric manner. The symmetric variant encompasses both negative and positive ranges, posing a challenge in fully utilizing all 2^b bits as it must accommodate the negative and positive halves, in addition to the zero, resulting in one bit unused on one side. Conversely, the asymmetric variant solely employs the positive range and incorporates an additional value to record the zero point, thereby facilitating the conversion of values back and forth and making the full utilization of all bits. Equation 3.7 describes the quantization function and its inverse function for the symmetric RTN method

$$\begin{aligned} Q(W) &= \left\lfloor \frac{W}{s} \right\rceil = W_q \\ Q^{-1}(W_q) &= W_q \times s \\ s &= \frac{max(|W|)}{2^{b-1}} \end{aligned} \tag{3.7}$$

While the asymmetric RTN variant, formulated in Equation 3.8, is slightly more complex as it employs the zero points.

$$\begin{aligned} Q(W) &= clip\left(\left\lfloor \frac{W}{s} + z \right\rceil, 0, 2^b - 1\right) = W_q \\ Q^{-1}(W_q) &= s \times (W_q - z) \\ s &= \frac{max(W) - min(W)}{2^b - 1} \\ z &= \frac{-min(W)}{s} \end{aligned} \tag{3.8}$$

In the equations 3.7 and 3.8, W is the weight to be quantized which can be part of the original weight matrix in group-wise quantization algorithms. W_q is the quantized weight, b is the number of bits, s is the scale factor, and z is the zero point. RTN can be implemented efficiently as it only involves basic arithmetic operations. It is hardware-friendly and demonstrates robust performance when coupled with a fine-grained grouping strategy, as evidenced by (Frantar et al., 2023).

3.3.2 Quantile Quantization

Quantile Quantization (Dettmers et al., 2022) divides the range of values into intervals based on quantiles rather than equal-width intervals. This approach is particularly useful since the weights in LLMs are typically not uniformly distributed.

Specifically, the quantile quantization maps an infinite stream of real numbers into a series of low bit integers by rounding the real numbers to the nearest mid-points of the quantile intervals. More formally, suppose b is the bit width, X is the random variable that follows a particular probability distribution P, $F(x) = P(X \leq x)$ is the cumulative distribution function (CDF). The Equation 3.9 formulates the quantiles quantization.

$$Q(W) = \underset{i \in \{0,1,\cdots,2^b\}}{argmin} \left\| W - \frac{F^{-1}(i) + F^{-1}(i+1)}{2} \right\|_1 \tag{3.9}$$

Figure 3.3 illustrates the quantiles for weights that follows the zero-centered normal distribution with 1 as standard deviation, *i.e.* $X \sim \mathcal{N}(0,1)$, for the 4-bit quantization. While Figure 3.4 presents the quantiles for the 4-bit quantization where X follows a bimodal distribution.

Quantile quantization is an information theoretically optimal scheme that ensures the quantization grids contain equal number of values. However, evaluating the quantiles of a large weight matrix is time-consuming. Fine-grained grouping is utilized to identify quantiles

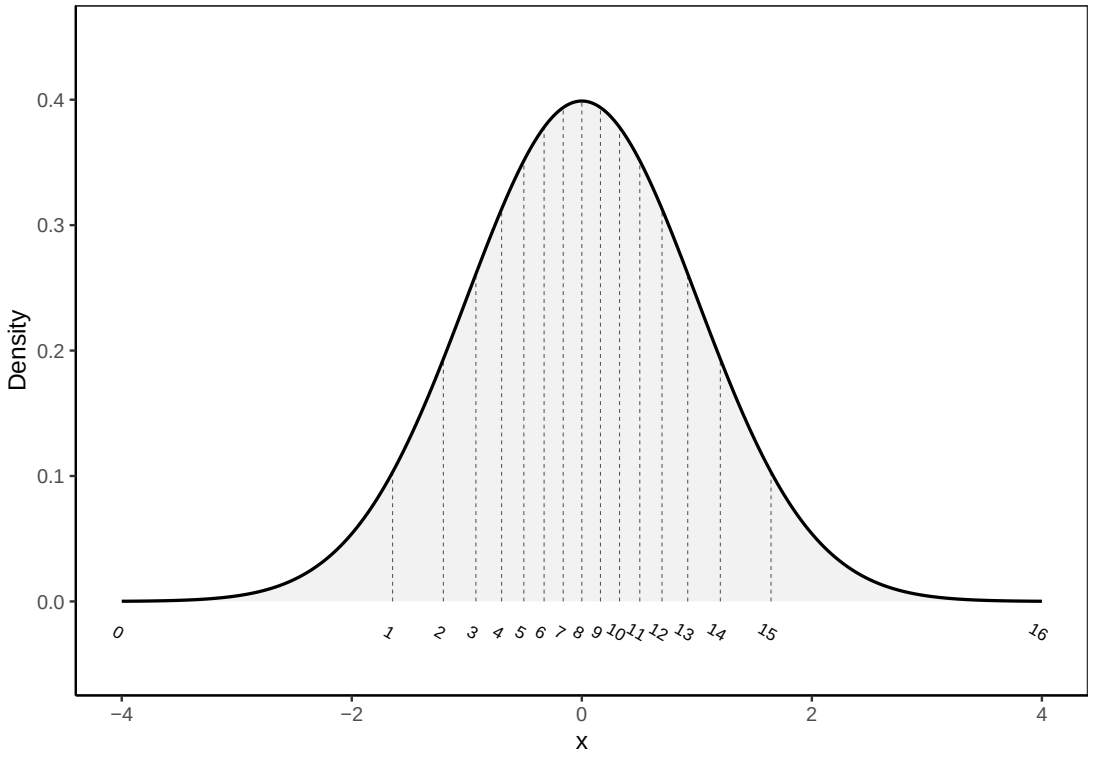


Fig. 3.3 This figure demonstrates the 16 quantization intervals of a sample under normal distribution of $\mathcal{N}(0,1)$.

faster and more accurate, however, this approach incurs additional storage overhead to keep quantiles lookup tables for each group. It is not hardware friendly as the dequantization process involves searching for the quantiles from the group-wise lookup table.

3.3.3 K-means Cluster Centroids

The K-means clustering algorithm can be leveraged to quantize a stream of real numbers as demonstrated in (Lloyd, 1982) where K-means was employed to solve pulse-code modulation (PCM) problems. In the context of LLM quantization, K-means cluster centroids are used as the best approximations for the dequantized weights, which ensures minimal quantization error due to the property of cluster centroid that minimizes the squared error as denoted in Equation 3.10. Specifically, for b bit quantization, the weights of LLM can be clustered into 2^b clusters and the centroids are the values for the dequantized weights. This approach was employed in the SqueezeLLM (Kim et al., 2023).

$$\underset{C_k}{argmin} \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|_2^2 \quad \text{where} \quad \mu_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i \tag{3.10}$$

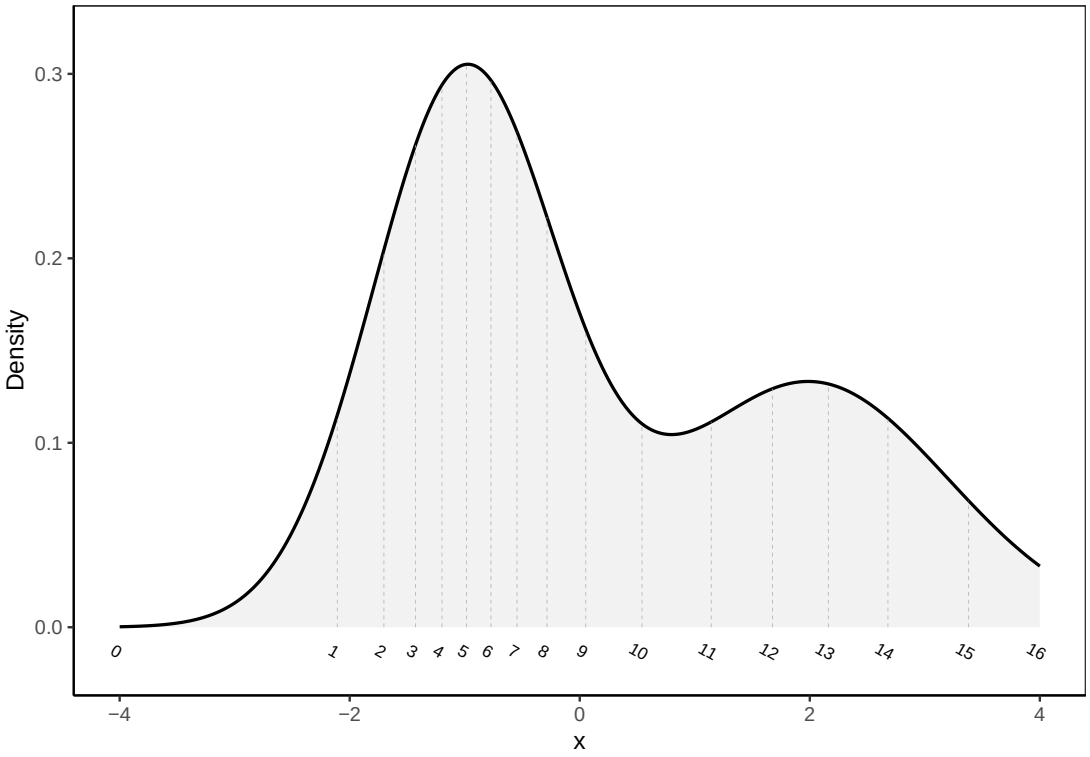


Fig. 3.4 This figure illustrates the 16 quantization intervals of a data sample conforms to a bimodal distribution with mixture of $\mathcal{N}(-1, 0.8^2)$ and $\mathcal{N}(2, 1.2^2)$.

More formally, given set $C = \{c_1, \cdots, c_{2^b}\}$ as the collection of centroids corresponding to a weight matrix or its subset W, the K-means clustering quantization approach can be formulated with the Equation 3.11:

$$Q(w) = \underset{i \in \{0,1,\cdots,2^b\}}{argmin} \|w - c_i\|_1 \tag{3.11}$$

where w denotes the individual element of the weight matrix W.

The K-means clustering algorithm is an iterative method used to partition a dataset into K clusters. The algorithm seeks to minimize the within-cluster variance by calculating the centroids of each cluster and iteratively updating them which implies that evaluating the centroids of a large weight matrix is time-consuming. Thus fine-grained groups are typically utilized to reduce errors and accelerate quantization speed. However, this approach incurs additional storage overhead since the centroid lookup tables for each group need to be stored along with the quantized model.

3.3.4 Llama-2 Architecture Overview

The Llama-2 model follows the standard transformer architecture (Touvron et al., 2023) with some notable changes to facilitate training and enhance performance. Specifically, the major customizations include layer normalization, activation function, position embedding and grouped query attention (GQA) to improve KV cache efficiency for larger models such as 70B (Touvron et al., 2023). The main architecture of a Llama-2 consists of a stack of transformer decoder layers. Each layer is composed of components such as: 1) multi-head self-attention (MHSA) or GQA, 2) Multi-Layer Perceptron or Feed-forward neural network (FFN), 3) Residual connections 4) Layer normalization. Figure 3.5 illustrates a simplified architecture diagram for the Llama-2-13B model.

The Multi-Head Self-Attention (MHSA) sub-layer employs scaled dot-product attention with multiple heads to capture diverse relationships between tokens. Queries (Q), keys (K), and values (V) are projected via learned linear transformations, and attention is computed as in Equation 3.12:

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.12}$$

The feed-forward neural network (FFN a.k.a. MLP) sub-layer includes two linear transformations with a non-linear activation function (SiLU) in between. Instead of ReLU or GELU which is popular in other transformers, Llama-2 leverages the SiLU (Sigmoid-Weighted Linear Unit) activation function which is defined as in Equation 3.13:

$$SiLU(x) = x \cdot sigmoid(x) \tag{3.13}$$

SiLU enables smooth gradient flow, reducing probability of dead neurons. Improved performance on various benchmarks.

Llama-2 employs pre-normalization which is applied at the start of self-attention and feed-forward sub-layers to make input zero-centered and having unit variance. Pre-normalization has been shown to stabilize training, especially for deep transformers.

Residual connections are used to add the input of a sub-layer to its output. The residual connections in Llama-2 are applied around the combination of layer normalization, self-attention, and feed-forward layers. This ensures smoother gradient flow during back-propagation.

Combining the components discussed above, the computation flow within a single Llama-2 decoder layer can be summarized as follows:

1. Input to the decoder layer: X.

2. Pre-normalize input with RMSNorm for the self-attention sub-layer:

$$NormorlizedInput(Attention) = RMSNorm(X)$$

3. Perform self-attention and add residual connection:

$$Y = X + Attention(NormorlizedInput(Attention))$$

4. Pre-normalize output of attention sub-layer for the MLP sub-layer:

$$NormalizedInput(MLP) = RMSNorm(Y)$$

5. Perform Feed-Forward Network transformation and add residual connection:

$$Z = Y + MLP(NormalizedInput(MLP))$$

6. Output of the decoder layer: Z.

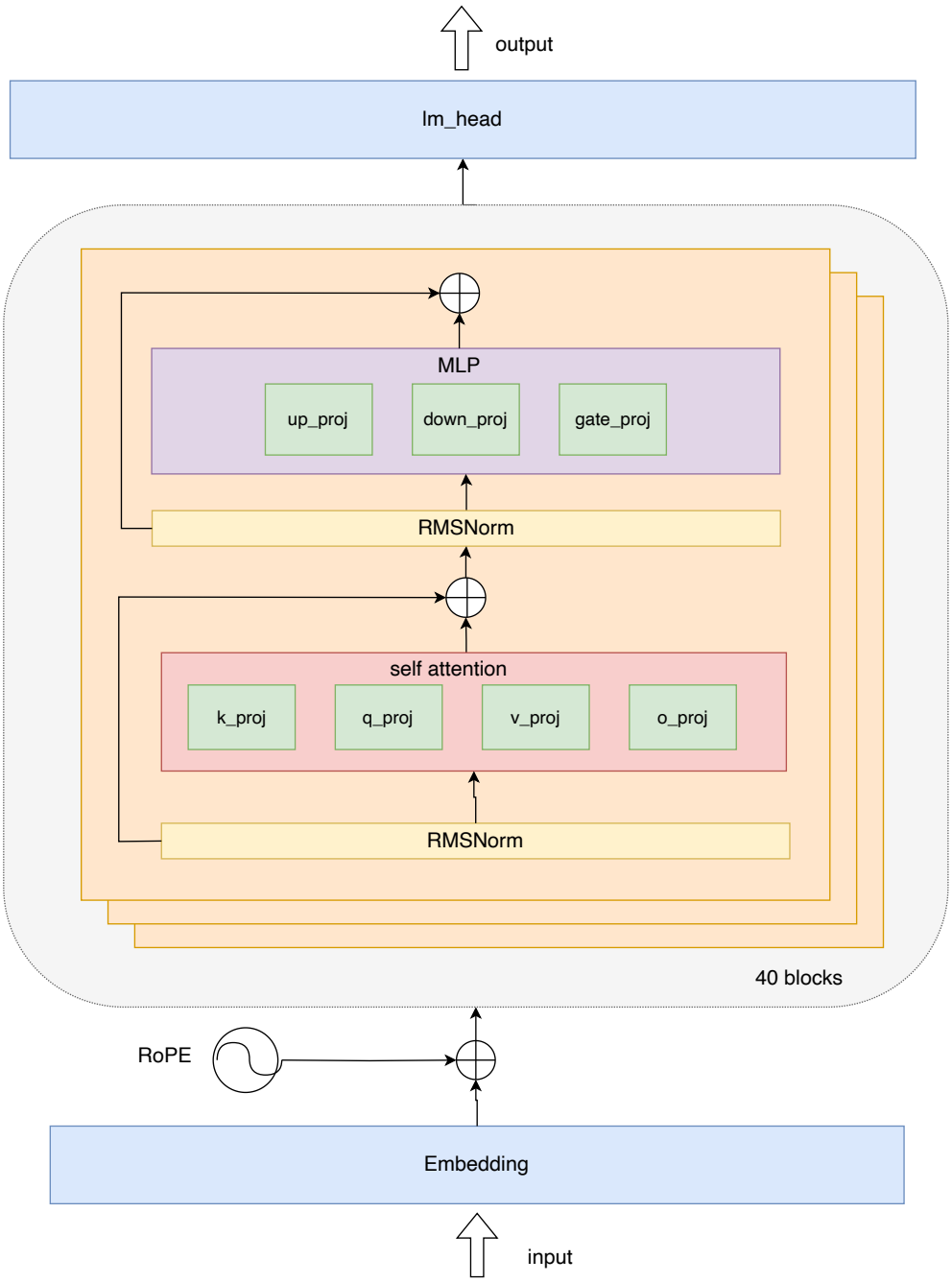


Fig. 3.5 This figure illustrates the architecture of the Llama-2-13B model by Meta. Notably, the Llama-2-13B model employs RoPE (Rotary Position Embedding), and RMSNorm (Root Mean Square Layer Normalization) in the "Pre-Norm" formulation. The model includes 40 layers and a vocabulary size of 32000.

Chapter 4

MXQ – A Mixed Quantization Approach for LLMs

4.1 Introduction

Previous studies on quantizing LLMs (Badri and Shaji, 2023; Dettmers et al., 2023a; Frantar et al., 2023; Lin et al., 2023) employed identical quantization configurations across entire model. This approach lacks flexibility in balancing the trade-off between memory consumption and model performance under various resource constraints and may be sub-optimal, especially in billion-scale LLMs as demonstrated in Figure 1.2, where some layers exhibit far taller weight bands and spikes in Kurtosis line, indicating not all layers are equally quantizable. Further experiment on the quantization errors under a diverse of quantization configurations also proves the varied difficulty across layers, as demonstrated by Figure 4.1, where the quantization errors (denoted by the FNorm) exhibit wide variations across layers and modules. This necessitates a new approach that employs non-uniform quantization settings for optimal quantization.

The contributions of this chapter are as follows: First, it introduces an effective quantization method, MXQ, designed for precise budget control and flexible memory-accuracy trade-offs. Second, it formulates MXQ using a novel optimisation framework that minimizes the sum of quantization errors in the objective while incorporating a storage cost function as constraint. This formulation is efficiently solved using a mixed-integer linear programming (MiLP) algorithm, enabling fast and effective quantization. Finally, extensive experiments are conducted on perplexity benchmarks for the Llama family of language models across a wide range of bit budgets, demonstrating MXQ's ability to achieve flexible budget control and superior memory-accuracy trade-offs compared to state-of-the-art quantization methods.

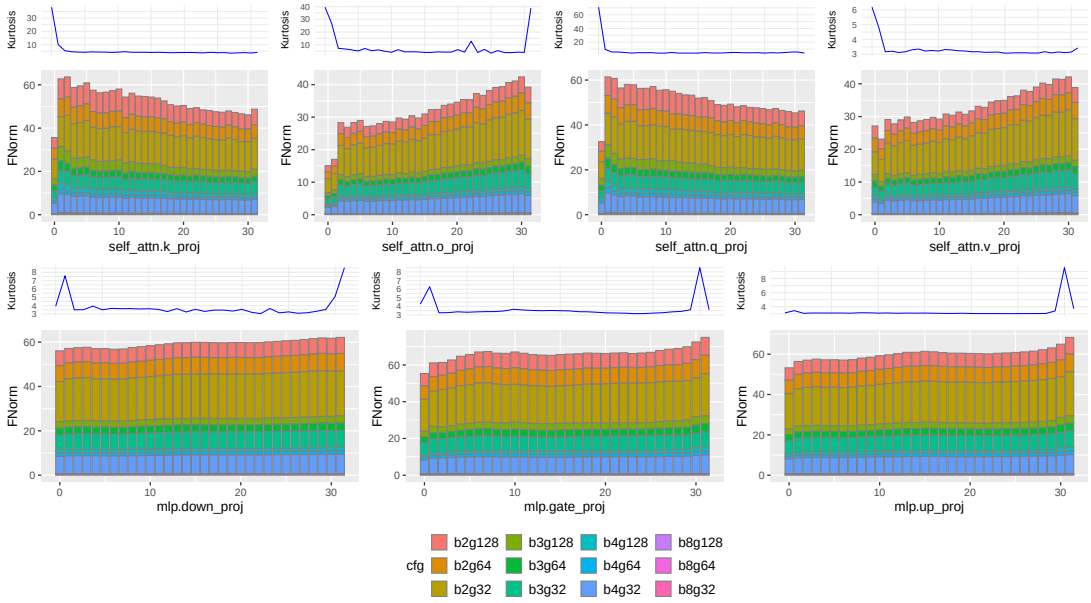


Fig. 4.1 This figure presents the Layer-wise quantization errors under 12 configurations of the Llama-2-7B model. The self-attention k and q modules exhibit decreasing quantization errors. While the other self-attention and MLP modules' quantization errors demonstrates ascending pattern. For optimal clarity, the figure is best viewed in color and with zoom.

4.2 MXQ Methodology

MXQ allocates optimal quantization configurations to each weight matrix according to a user-specified overall bit budget for each parameter to minimize the global quantization error. Specifically, MXQ identifies ideal configurations that minimize the sum of Frobenius norm of the difference between the original matrices and their quantized counterparts while confining the memory usage within the constraint of the target memory budget. Thus the problem can be formulated as a Mixed integer Linear Programming (Huangfu and Hall, 2018) problem. Let's denote $c_i = (b_1, g_1, b_2, g_2)$ as the configuration parameters used to quantize the ith matrix of the LLM, where:

- b_1 denotes the first level bit width to represent the quantized weights.
- g_1 refers to the first level group size to break the parameters into granular chunks.
- b_2 describes the second level bit width to quantize zeros and scales.
- g_2 is the second level group size for quantization of zeros and scales.

Let C be the set of all possible configurations. In this thesis, the search space is limited as listed in Table 4.1, leading to the cardinality of all possible configurations to be 12. Moreover,

Table 4.1 Valid values of the mixed quantization configuration

Parameter	b_1	g_1	b_2	g_2
Value	2, 3, 4, 8	32, 64, 128	8	128

let $\{W^{(i)}\}_{i=1}^{N}$ be the set of N matrices in the LLM to be quantized. In Equation 4.1, the quantization problem is formulated as an linear programming problem:

$$\begin{aligned} &\underset{c_1,c_2,\cdots,c_N}{\arg\min} \sum_{\substack{i\in\{1,\ldots N\}\\ c_i\in C}} \left\| W^{(i)} - \hat{W}_{c_i}^{(i)} \right\|_F \\ &\text{s.t.} \quad \sum_{\substack{i\in\{1,\ldots N\}\\ c_i\in C}} \texttt{storage}(W^{(i)}, c_i) \leq \beta, \end{aligned} \tag{4.1}$$

where parameter β denotes the overall memory budget for the quantized model in megabytes. The storage cost function `storage` is defined in Equation 4.2:

$$\texttt{storage}(W^{(i)}, c_i) = |W^{(i)}| \cdot \left(b_1 + \frac{2b_2}{g1} + \frac{32}{g_1 \cdot g_2} \right) \tag{4.2}$$

For simplicity, the same second-level bit width and group size are used for scales and zeroes, as variations in these configurations have minimal impact on overall memory consumption.

To minimize the objective function in Equation 4.1, the Frobenius norms and storage costs with respect to all possible configurations ($N \times |C|$ in total) for a given LLM need to be calculated. To accelerate the process, the Frobenius norm and storage cost can be pre-computed and stored in two matrices beforehand, denoted as $F \in \mathbb{R}^{N\times|C|}$ and $S \in \mathbb{R}^{N\times|C|}$. With these pre-computed metrics in place, the optimal quantization configurations problem is re-formulated as a mixed integer linear programming problem by introducing a series of binary decision variables x_j, where $j \in \{1,\ldots,M\}$ and $M = N \times |C|$. Specifically, the optimisation problem is as follows:

$$\begin{aligned} &\underset{X}{\arg\min} \quad F \cdot X \\ &\text{s.t.} \quad S \cdot X \leq \beta\,, \\ &\qquad A \cdot X = \begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix}^{T}_{1\times N}, \end{aligned} \tag{4.3}$$

$$
\begin{aligned}
\text{where} \qquad & X = \begin{pmatrix} x_1 & x_2 & \cdots & x_M \end{pmatrix}^T, \\
& x_j \in \{0,1\} \quad \forall j \in \{1,\ldots,M\}, \\
& F = \begin{pmatrix} f_1 & f_2 & \cdots & f_M \end{pmatrix}, \\
& S = \begin{pmatrix} s_1 & s_2 & \cdots & s_M \end{pmatrix}, \\
& A = \begin{pmatrix} \overbrace{1\cdots1}^{|C|} & \overbrace{0\cdots0}^{|C|} & \cdots & \overbrace{0\cdots0}^{|C|} \\ \overbrace{0\cdots0}^{|C|} & \overbrace{1\cdots1}^{|C|} & \cdots & \overbrace{0\cdots0}^{|C|} \\ \vdots & \vdots & \ddots & \vdots \\ \overbrace{0\cdots0}^{|C|} & \overbrace{0\cdots0}^{|C|} & \cdots & \overbrace{1\cdots1}^{|C|} \end{pmatrix}_{N\times M}.
\end{aligned}
$$

To search for one optimal configuration out of $|C|$ for a total number of N matrices, $A \in \mathbb{R}^{N\times M}$ in Equation 4.3 is initialized as a matrix of N rows by M columns with only 0's and 1's. Each row contains $|C|$ consecutive ones corresponding to positions of the weight matrices encoded in A. Equation 4.3 can be solved efficiently by off-the-shelf LP solvers such as Gurobi (Gurobi Optimization, LLC, 2023) and HiGHS (Huangfu and Hall, 2018). In the experiment, the scipy wrapper of HiGHS is adopted to solve Equation 4.3.

4.3 Experiments

Theoretically, MXQ can be extended to support quantization methods that support layer-wise configurations. However, limited by time and computation resources, the thesis focused experiments on leveraging HQQ (Badri and Shaji, 2023) as the underlying quantization implementation due to its impressive quantization speed and outstanding accuracy. The design of the experiments primarily emphasizes the 3- and 4-bit quantization as these configurations better preserve the performance of LLMs (Dettmers and Zettlemoyer, 2023). Besides the perplexity, which is a stringent measurement of accuracy and generally reflects true performance of the LLM. Additionally, quantization speed and actual GPU memory consumption are benchmarked so as to evaluate the quantization algorithms comprehensively.

The thesis evaluated the performance of the MXQ on the state-of-the-art large language models such as the Llama family models to verify the efficacy of the MXQ. The selected baselines were simultaneously tested under the same experimental settings. To facilitate

the experiment and maximize reproducibility, a harness tool named *lm-quant-toolkit*, open-sourced on GitHub, is developed to execute the experiments and collect data. The procedures to execute the experiments are elaborated in the appendix A.

4.3.1 Settings

The proposed method was applied to the Llama (Touvron et al., 2023) family models, including Llama-2-7B, Llama-2-13B, and Llama-3-8B. Perplexity is the main evaluation metrics since good results in perplexity evaluation generally reflect the true performance of the LLM (Dettmers and Zettlemoyer, 2023). It is also widely employed by previous works (Badri and Shaji, 2023; Frantar et al., 2023; Lin et al., 2023). Specifically, the Hugging Face (Face, 2024) perplexity evaluation method is employed, which is slightly different from those used in prior works and tends to yield lower values than those reported in existing literature. The perplexity metrics were evaluated on the two datasets: WikiText-2 (Merity et al., 2016) and C4 (Raffel et al., 2020) respectively. The current state-of-the-art calibration-based and calibration-free quantization methods are adopted as the baselines, including HQQ (Badri and Shaji, 2023), BnB (Dettmers et al., 2023a), GPTQ (Frantar et al., 2023) and AWQ (Lin et al., 2023). These baselines were re-evaluated with 3 or 4-bit configurations according to Table 4.1, ensuring a fair comparison.

All the experiments were conducted on a Linux workstation with Nvidia RTX 4090 GPU and partially reproduced on another Linux server with Nvidia RTX 4080 GPU. The hardware and software specifications are presented in the Table A.1. The Llama models, published by Meta, were fetched from the HuggingFace. The WikiText2 and C4 datasets were also downloaded from HuggingFace.

4.3.2 Perplexity Results

The perplexity metrics for various Llama models are presented in Table 4.2. The evaluation results for the proposed mixed quantization method are shown as the “MXQ” rows at the bottom of the table. Hundreds of bit budgets are tested , ranging from 2.13 to 8.51 as presented in Table C.1 in the appendix, some of the representative bit budgets as listed in the Table 4.2. The upper section of the table displays the results for HQQ, GPTQ, and AWQ under various bit and group size settings. The first column denotes the quantization method, while second column indicates the combination of bit width and group size.

Figure 4.2 presents the broader range of quantization options and the corresponding performance metrics within the 3-bit to 5-bit range for the Llama-2-13B models on the WikiText2 dataset on the Llama-2-13B model. The calibration-free quantization methods

Table 4.2 This table presents the perplexity metrics of the proposed MXQ method along with various baseline methods on the Llama family models. The table includes some of MXQ's representative bit budgets, ranging from 3.07 to 6.89. As the budget increases, the MXQ's perplexity approaches or even surpasses the state-of-the-art methods, demonstrating its effectiveness in prioritizing accuracy over memory.

			Llama-2-7B			**Llama-2-13B**			**Llama-3-8B**		
Method	Config	BPP[1]	↓WikiText2[2]	↓C4[3]	↓MEM[4]	↓WikiText2	↓C4	↓MEM	↓WikiText2	↓C4	↓MEM
FP16	-	16	5.18	6.95	12.55	4.63	6.45	19.21	5.81	8.98	14.96
HQQ	b4 g32	4.51	5.28	7.06	4.07	**4.69**	**6.51**	7.63	**6.07**	9.41	5.82
MXQ		4.51	5.29	7.08	**3.89**	**4.69**	**6.51**	**7.28**	6.11	9.47	**5.62**
GPTQ[5]		4.51	5.39	7.11	4.74	4.72	6.54	8.43	8.80	10.44	6.71
AWQ		4.51	**5.26**	**7.04**	3.98	**4.69**	**6.51**	7.59	**6.07**	**9.39**	5.72
HQQ	b4 g64	4.25	5.30	7.11	3.79	**4.70**	6.54	7.07	6.19	9.60	5.51
MXQ		4.25	5.31	7.13	**3.71**	4.71	6.54	**6.90**	6.29	9.76	5.49
GPTQ		4.25	5.39	7.13	4.50	4.73	6.56	7.97	11.83	11.77	6.46
AWQ		4.25	**5.29**	**7.07**	3.74	4.71	**6.53**	7.10	**6.15**	**9.52**	**5.46**
HQQ	b4 g128	4.13	5.35	7.16	3.65	4.74	6.57	6.80	6.38	9.94	5.36
MXQ		4.13	5.33	7.17	3.72	4.74	6.57	6.94	6.40	9.96	5.54
BnB[6]		4.13	5.32	7.12	**3.60**	4.72	**6.55**	**6.71**	**6.20**	**9.64**	**5.31**
GPTQ		4.13	5.39	7.18	4.39	4.74	6.57	7.74	97.03	27.77	6.33
AWQ		4.13	**5.31**	**7.10**	3.62	**4.71**	**6.55**	6.92	6.21	9.67	5.33
HQQ	b3 g32	3.51	**5.62**	**7.53**	4.07	**4.89**	**6.78**	7.63	**7.09**	**11.16**	5.82
MXQ		3.51	5.65	7.63	4.17	4.92	6.84	7.54	7.30	11.68	6.16
GPTQ		3.51	5.94	7.81	**3.20**	5.09	6.96	**5.93**	17.76	17.98	**4.88**
AWQ[7]		3.51	-	-	-	-	-	-	-	-	-
HQQ	b3 g64	3.25	**5.82**	**7.80**	3.41	**4.98**	**6.94**	6.33	**7.80**	**12.35**	5.11
MXQ		3.25	6.04	8.13	3.98	5.06	7.05	7.18	9.52	15.23	5.80
GPTQ		3.25	6.13	8.07	**2.98**	5.14	7.06	**5.49**	11.16	14.33	**4.64**
AWQ		3.25	-	-	-	-	-	-	-	-	-
HQQ	b3 g128	3.13	**6.20**	8.39	3.08	**5.15**	**7.14**	5.69	**9.31**	**14.90**	4.75
MXQ		3.13	6.36	8.60	3.85	5.24	7.34	6.92	12.23	19.49	5.67
GPTQ		3.13	6.32	**8.30**	**2.87**	5.24	7.19	**5.27**	52.78	30.04	**4.52**
AWQ		3.13	-	-	-	-	-	-	-	-	-
MXQ[8]	6.89	6.89	**5.25**	**7.01**	6.40	**4.66**	**6.48**	11.91	**6.01**	**9.30**	8.49
	5.72	5.72	5.26	7.03	5.54	4.67	6.50	10.22	**6.04**	**9.35**	7.57
	5.02	5.02	5.28	7.05	5.01	4.68	6.50	9.18	**6.05**	**9.38**	7.00
	4.21	4.21	5.32	7.14	4.43	4.72	6.55	8.04	6.34	9.84	6.46
	4.17	4.17	5.33	7.16	4.44	4.73	6.56	8.07	6.38	9.93	6.48
	4.11	4.11	5.34	7.18	4.46	4.74	6.58	8.10	6.42	9.99	6.50
	4.07	4.07	5.36	7.21	4.44	4.75	6.59	8.12	6.46	10.08	6.53
	3.95	3.95	5.43	7.29	4.44	4.79	6.65	8.06	6.59	10.36	6.45
	3.87	3.87	5.45	7.33	4.46	4.81	6.67	8.12	6.73	10.59	6.35
	3.83	3.83	5.46	7.36	4.42	4.82	6.69	8.04	6.80	10.71	6.30
	3.65	3.65	5.54	7.50	4.32	4.87	6.76	7.81	7.09	11.24	6.27
	3.19	3.19	6.15	8.30	3.91	5.11	7.12	7.05	11.20	17.89	5.73
	3.15	3.15	6.25	8.45	3.87	5.18	7.25	6.96	12.02	19.10	5.69
	3.11	3.11	6.49	8.75	3.82	5.30	7.43	6.88	12.52	20.16	5.65
	3.07	3.07	6.78	9.12	3.78	5.57	7.72	6.80	13.54	21.40	5.61

[1] Bit per parameter, i.e. the average bits a paratemer takes.

[2] Perplexity on the WikiText2 dataset, evaluation follows the HuggingFace algorithm (Face, 2024).

[3] Perplexity on a subset of the C4 dataset, which is composed of the first 1,100 entries of the en validation split.

[4] Memory is measured using PyTorch's API after loading the quantized model into GPU. This column is reported in GiB.

[5] github.com/AutoGPTQ/AutoGPTQ is used for this experiment.

[6] The 4-bit BnB employes 64 as the group size. However, the bit per parameter is 4.13 according to (Dettmers et al., 2023a).

[7] The github.com/casper-hansen/AutoAWQ used by this experiment has no 3-bit quantization implementation.

[8] Additional bit budgets unavailable to other baseline methods.

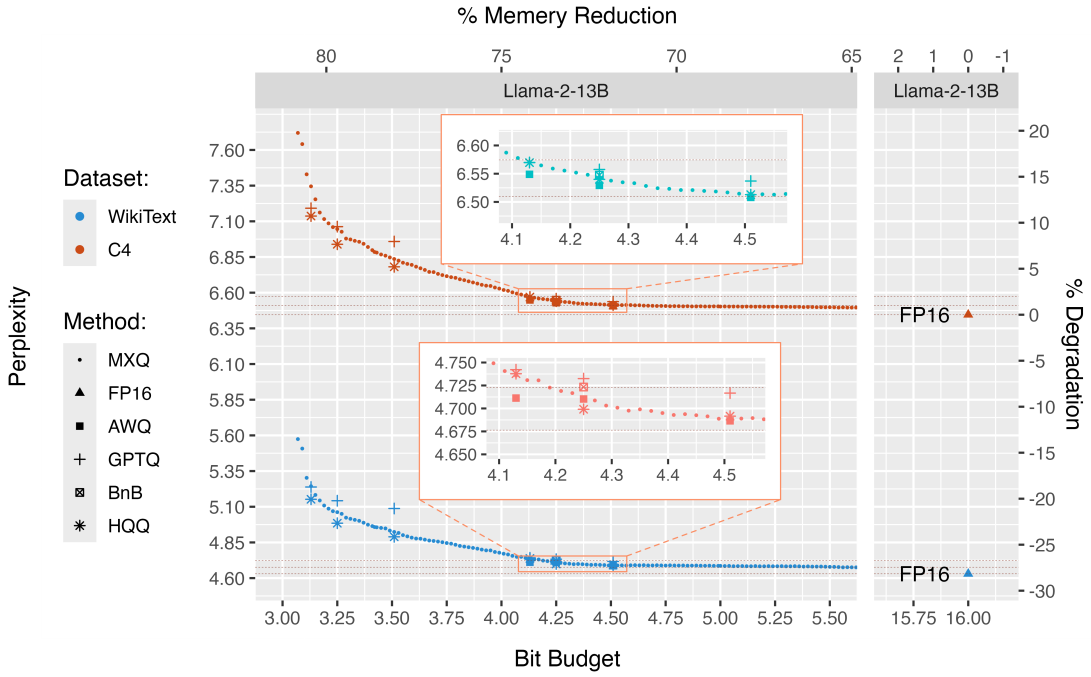


Fig. 4.2 This figure illustrates the potential trade-offs between memory and perplexity within the 3-bit to 5-bit range for the Llama-2-13B models on the WikiText2 and C4 dataset using quantization methods such as AWQ, GPTQ, HQQ and BnB. The red triangle at the lower right corner denotes the FP16 baseline. The gaps between the horizontal lines represent the 1% and 2% perplexity degradation ranges respectively. The tiny square indicates the BnB performance. As evidenced by this plot, MXQ offers far more options to balance memory and performance. It achieves the 1% performance loss goal at a bit budget around 5.6. Additionally, it enables aggressive 77% memory saving at a bit budget around 3.7 while maintaining perplexity drop within 5%.

such as HQQ and BnB are included as reference. The red triangle at the lower right corner denotes the FP16 baseline. The gaps between the three horizontal lines represent the 1% and 2% perplexity drop zones.

MXQ offers a flexible trade-off between memory consumption and accuracy by unlocking a variety of budget-perplexity options, showing good potential for real application with diverse memory constraints. On one hand, MXQ can leverage slightly larger memory budget for better accuracy, which is not an option for the existing methods. As demonstrated in Table 4.2, at the bit budget of 6.89, MXQ surpasses all SoTA methods on WikiText2 and C4 perplexity metrics for all the three Llama models. Additionally, MXQ achieves the 1% performance loss goal at a bit budget around 5.6 on the Llama-2-13B model as evidenced in the Figure 4.2. On the other hand, MXQ enables aggressive memory reduction. As illustrated

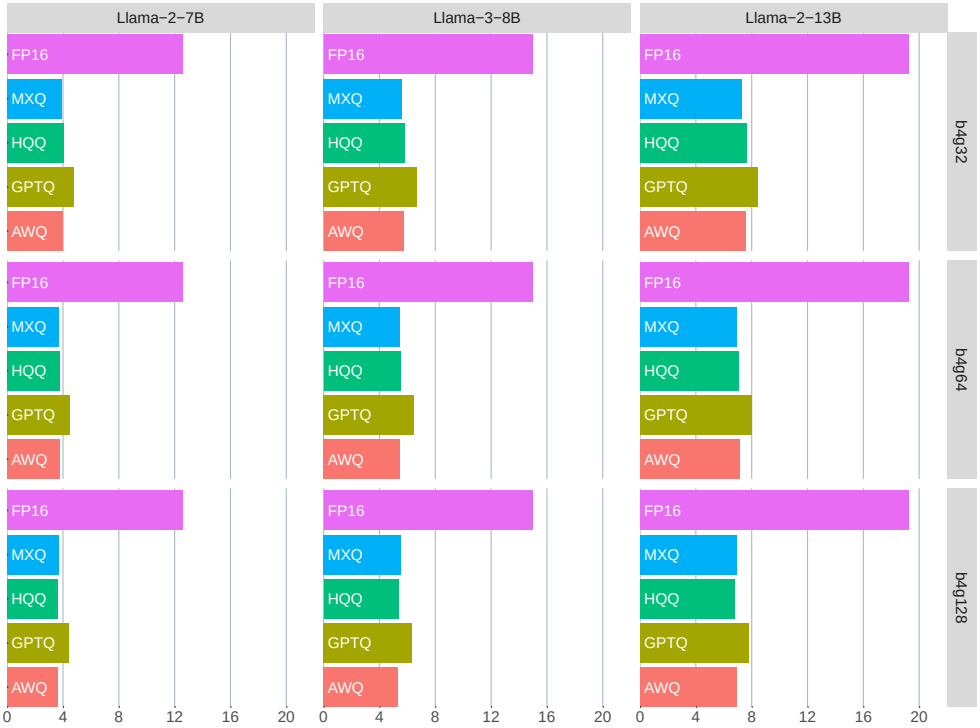


Fig. 4.3 This figure illustrates the GPU memory usage, measured in gigabytes (GiB) when loading baseline and quantized Llama models into GPU prior to executing any inference. All quantized models exhibit a substantial reduction in memory consumption when compared to the FP16 baseline. Notably, the MXQ approach offers additional memory saving comparing to HQQ.

by Figure 4.2, MXQ achieves approximately 77% memory saving at a bit budget around 3.7 while maintaining perplexity drop within 5% on the Llama-2-13B model.

4.3.3 Actual Memory Usage and Quantization Speed

In addition to evaluating perplexity, a comprehensive assessment of the actual GPU memory consumption and quantization time of MXQ and the baseline quantization methods on the Llama models were conducted. As presented in Figure 4.3, all quantization methods demonstrate significant drop in memory usage, decreasing it to approximately one-third of the memory required by the unquantized models. Notably, MXQ exhibits additional memory reduction when compared to HQQ.

With respect to the quantization time, even though MXQ incurs small overhead on the MiLP algorithm to find optimal quantization configurations, it is able to quantize the Llama-2-13B model within 1 minute, which is an order of magnitude faster than the AWQ and GPTQ as exhibited in Figure 4.4.

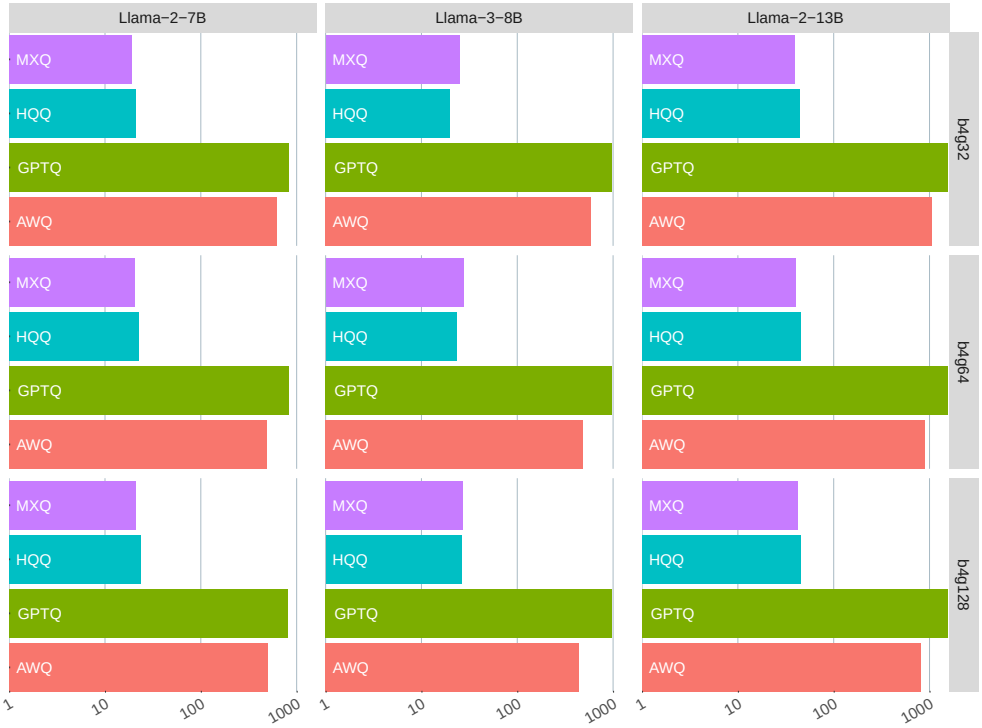


Fig. 4.4 This figure compares the quantization time of various quantization methods: MXQ, HQQ, AWQ, GPTQ. The three Llama variants: Llama-2-7B, Llama-2-13B and Llama-3-8B, are evaluated using b4g32, b4g64 and b4g128 configurations on a NIVDIA RTX 4090 GPU. The quantization time is measured in seconds and displayed in log scale (log10). Notably, GPTQ and AWQ quantization are approximately an order of magnitude slower than MXQ and HQQ.

4.4 Discussion

The MXQ method is designed to be compatible with any transformer-based models and quantization methods. However, the experiments were conducted using only HQQ on a selection of Llama family models. Evaluating the proposed method on limited underlying method and models may not ensure comparable performance on other quantization algorithm and models. Additionally, end-to-end metrics such as IFEval, BBH, MATH Level 5, GPQA, MUSR and MMLU-PRO were not assessed in the experiments, which may limit the robustness of the true performance of the proposed method on real world tasks. Furthermore, the experiments did not consider the inference speed of the quantized model, as the proposed method does not directly influence this aspect.

4.5 Conclusion

The mixed quantization(MXQ) represents an effective approach to optimizing the balance between model accuracy and memory consumption. By specifying a bit budget, this methodology offers a broader spectrum of quantization options, enabling the identification of optimal quantization configurations to enhance existing quantization methods. MXQ demonstrates the capacity to further reduce memory utilization while incurring minimal degradation in accuracy. This innovative optimisation formulation to allow us to solve layer-wise quantization parameters, incorporating memory budget constraints. This approach enables the proposed method to offer an advantageous memory-performance balance when quantizing transformer-based large models. The flexibility and effectiveness of the proposed method are substantiated through extensive experiments across multiple transformer model architectures and benchmark datasets. Incorporating the proposed MXQ into the repository of quantization utilities would constitute a valuable supplement, enriching the tools available for optimizing increasingly larger transformer-based models.

Chapter 5

Towrads Superior Quantization Accuracy: A Layer-sensitive Approach

5.1 Introduction

Large Language Models (LLMs) have significantly advanced artificial intelligence, demonstrating human-like capabilities in natural language comprehension, problem-solving, logical reasoning, and knowledge retrieval. These models power a wide range of applications, from chatbots and virtual assistants to code generation and scientific discovery. However, their deployment is hindered by demand for substantial computational resources, which necessitates efficient model quantization techniques for real-world applications.

Quantization techniques aim to reduce the memory footprint and computational requirements of LLMs while preserving their performance. Existing quantization methods, such as AWQ (Lin et al., 2023), GPTQ (Frantar et al., 2023), BnB (Dettmers et al., 2023a), and HQQ (Badri and Shaji, 2023), predominantly employ uniform quantization configurations. While effective to some extent, these approaches fail to consider the varying quantization difficulty across different layers of billion-scale models. As illustrated in Figure 5.4, the weight distributions in modules such as `mlp.up_proj` differ significantly between the final layer and earlier layers, indicating that a uniform quantization approach may not be optimal.

To address this, MXQ (Zhang et al., 2025) introduced a mixed-integer linear programming (MiLP)-based approach to assign differentiated quantization configurations while maintaining an overall memory budget. However, despite its adaptive allocation strategy, MXQ-quantized models often underperform baseline methods such as HQQ, BnB, and AWQ, suggesting that existing layer-adaptive quantization strategies may not effectively prioritize quantization accuracy over memory efficiency in the trade-off.

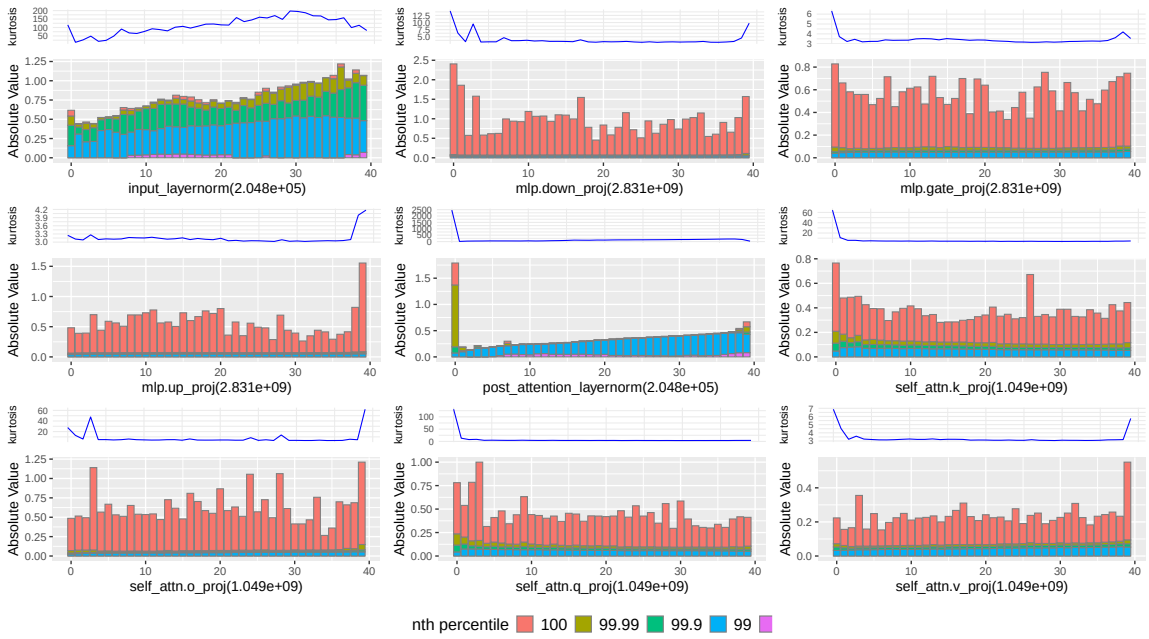


Fig. 5.1 This figure illustrates the Kurtosis and weight distributions across all layers of the Llama-2-13B model. The Kurtosis line plot shows the degree of deviation from a normal distribution, where a value of 3 indicates no deviation. The higher Kurtosis value denotes greater divergence thus greater difficulty to quantize accurately. This figure is best viewed in color and with zoom for clarity.

Additionally, prior researches (Molchanov et al., 2019; Samragh et al., 2023; Xu et al., 2023) have shown that the importance of weights within a deep neural network is non-uniform. Motivated by these observations and the limitations of state-of-the-art quantization techniques, this chapter introduces a novel approach based on layer sensitivity analysis. This approach hypothesizes that memory allocation contributes equally to quantization accuracy across most layers in LLMs, but a subset of layers, termed sensitive layers, require additional memory to maintain optimal performance. Identifying these layers and selectively allocating extra memory resources can enhance overall quantization accuracy with minimal additional cost.

The proposed method leverages layer-wise sensitivity metrics, including activation sensitivity (hereafter referred to as "sensitivity") and weight distribution kurtosis (DeCarlo, 1997), to identify demanding layers. By selectively allocating additional memory to these layers while slightly relaxing the overall memory constraint, improved quantization accuracy is achieved without incurring significant overhead. The proposed approach consists of the following key steps:

1. Layer Sensitivity Analysis – The distribution of activation sensitivity and kurtosis across model layers are analyzed to identify quantization-sensitive regions.

2. Targeted Memory Reallocation – Instead of applying a uniform quantization scheme, additional memory is allocated to high-sensitivity layers, ensuring they retain sufficient precision for optimal performance.

3. Experimental Validation – the proposed methods are evaluated on Llama models against baseline methods such as HQQ and MXQ, as well as the its ablation test variants.

The contributions of this chapter are as follows: First, the layer-wise sensitivity to quantization error was empirically explored on multiple transformer-based LLM families, indicating the robustness of sensitivity within a family of models and their fine-tuned variants. Second, a simple outlier detection algorithm was implemented to discover sensitive layers with sensitive scores or Kurtosis metrics. Third, this chapter implemented and validated the SensiBoost and KurtBoost methods that outperform HQQ up to 9% reduction in perplexity with approximately 2% increment in memory budget, solving the research question 2 effectively. Finally, the SensiMiLP and KurtMiLP were explored to enhance quantization performance while maintaining strict memory constraint.

This chapter starts with the discussion of sensitivity, the properties observed through empirical study. Then Kurtosis, a measurement of weight distribution of various weight matrix inside large language models, is explored, followed by the algorithm to detect exceptional data points in the sensitivity or Kurtosis metrics. Subsequently, the two methods dubbed as SensiBoost and KurtBoost, are proposed to enhance HQQ performance with relaxed memory constraint. Experiments results of the two methods are discussed. Finally, another two methods named as SensiMiLP and KurtMiLP are investigated in the context of preserving strict memory constraint.

5.2 Sensitivity

5.2.1 Sensitivity Definition

Transformer-based large language models are composed of multiple layers or blocks. Each layer consists of the self-attention and Multi-Layer Perceptron (MLP, a.k.a. FFN) sub-layers. Specifically, the Llama family model's self-attention includes weights for K, Q, V and O projection, known as `k_proj`, `q_proj`, `v_proj` and `o_proj` respectively. Similarly, the MLP sub-layer is composed of weights referred to as `mlp_proj`, `mlp_down` and `mlp_gate`.

Figure 3.5 presents the architecture of the Llama-2-13B model. The weights in a large language model are not equally important as revealed by the observation from prior study (Lin et al., 2023), which claims that preserving a small portion of so-called salient weights can significantly improve the quantization accuracy. These weights corresponds to particular channels inside a weight matrix. Motivated by this conclusion, this chapter hypothesizes that there also exists sensitive layers that are more severely affected by weight perturbation than others. Protecting such layers by allocating larger bit budget will result in improvement of overall quantization accuracy.

To quantify layer-wise sensitivity, this section defines sensitivity score as formulated in Equation 5.1:

$$s_i = \frac{\left\| W_i \cdot X - Q^{-1}(Q(W_i)) \cdot X \right\|_2^2}{|W_i \cdot X|} \tag{5.1}$$

where W_i denotes the weight in the ith layer, X is the input to the model which is from a small calibration data. The $Q()$ function represents the quantization function to convert the full-precision weight into its quantized counterpart. The $Q^{-1}()$ function is the inverse of the $Q()$.

Exceptional sensitivity scores, as illustrated by the spikes in Figure 5.2, can be identified by the outlier detection algorithm and utilized as a heuristic to assign extra memory budget to improve performance.

5.2.2 Measuring Sensitivity Score

The measurement of sensitivity requires a collection of small calibration datasets which are feed into the target LLMs layer-by-layer to calculate the output under the full-precision and quantized weights respectively. Then the mean squared error (MSE) is computed to quantify the sensitivity. Specifically, three open datasets, WikiText-2, C4 and pileval are utilized to evaluate the robustness of sensitivity. Additionally, a small synthesized dataset named branch of science (BoS) is created to further validate if the sensitivity property generalize to diverse datasets. The BoS is a synthesized dataset composed of a few hundred of textual definitions for science, art and business topics such as Mathematics, Physics, Chemistry, Law, Music and Journalism etc. It is generated using the Llama-2-7B model. The details of the program to produce the BoS dataset is described in appendix B.1.5. The BoS dataset is published on Hugging Face under the name schnell18/branch-of-science.

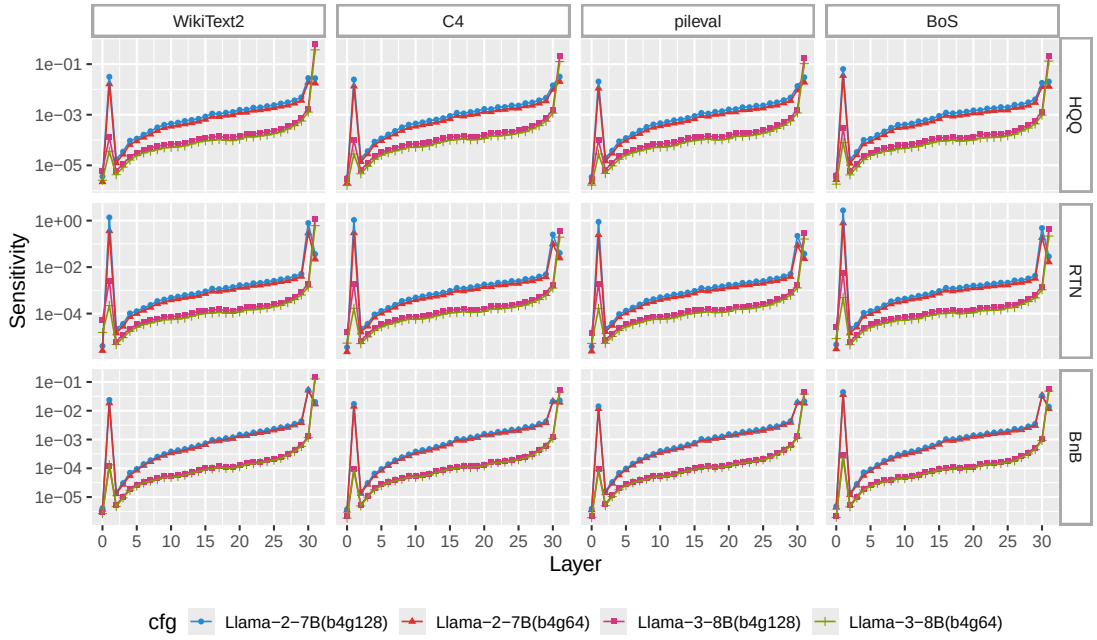


Fig. 5.2 This figure demonstrates the relationship between quantization method (HQQ, RTN, BnB), dataset (WikiText2, C4 pileval, BoS) and layer-wise sensitivity. The distinct shapes of sensitivity curves for Llama-2-7B and Llama-3-8B models indicate the sensitivity property is model dependent. While the near identical patterns across calibration datasets and quantization methods show layer-wise sensitivity to quantization error is independent of calibration datasets and quantization method. For optimal clarity, the figure is best viewed in color and with zoom.

The program to measure sensitivity is adapted from the `AutoAWQ` project on github and included in the `lm-quant-toolkit`. Instructions of using this tool are explained in appendix B.1.4.

5.2.3 Sensitivity Properties

According to diverse experiments, the layer-wise sensitivity to perturbation introduced by quantization of a particular model demonstrates considerable robustness. As evidenced by Figure 5.2, sensitivity is independent of dataset and quantization method. Moreover, the bit budget does influence the magnitude of sensitivity, however, the overall patterns of distinct bit budgets remain approximately identical as presented in Figure C.1, where the 3-bit, 4-bit and 8-bit groups share almost same spikes in layer at the beginning although they are separated by notable blank. Additionally, fine-tuned models preserve the sensitivity of base model, which is demonstrated in Figure 5.3 and Figure C.2 respectively. Experiments on the Llama family model reveals the sensitivity spikes tend to be present at the start and end layers.

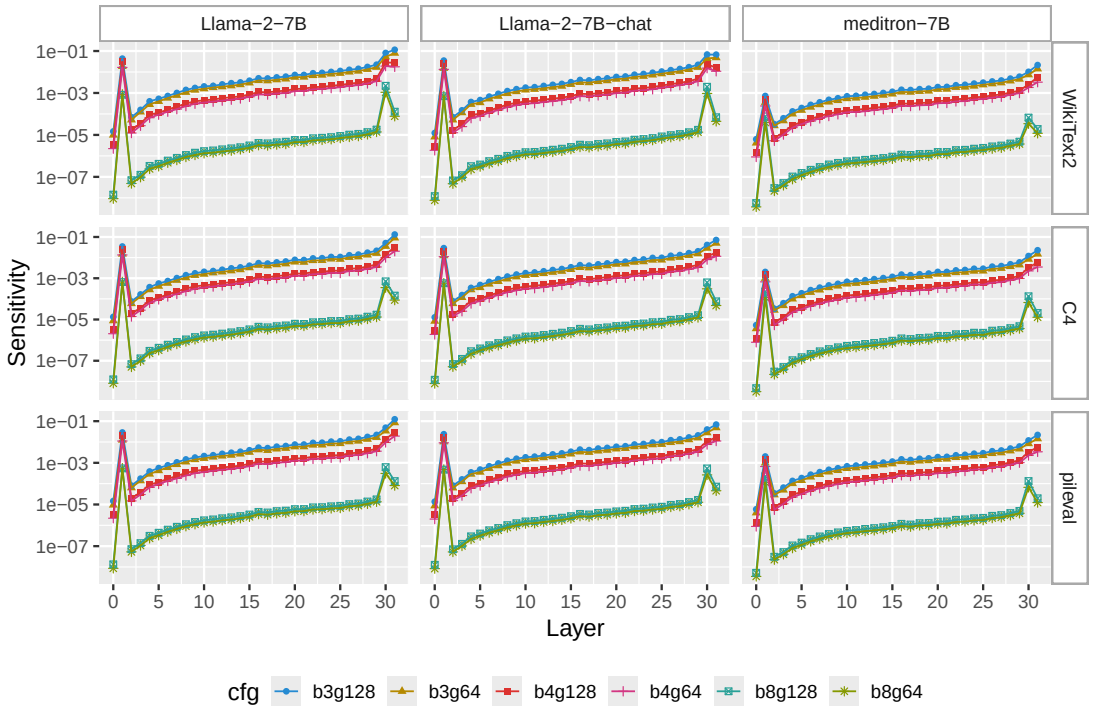


Fig. 5.3 This figure presents the sensitivity patterns among the Llama-2-7B base model and its fine-tuned mutations. Two fine-tuned models are included for comparison. The middle one is Llama-2-7B-chat. And the right is the meditron-7B which is a medical LLM fine-tuned on a carefully curated medical corpus. As indicated by the nearly identical shapes of sensitivity curve, the two fine-tuned models clearly inherit the sensitivity properties from the base model. For optimal clarity, the figure is best viewed in color and with zoom.

In summary, the sensitivity properties of large language models can be described as follows: **1)** sensitivity is independent of dataset and quantization method, **2)** sensitivity pattern is consistent among distinct bit budgets, **3)** fine-tuned models preserve the sensitivity of base model, **4)** sensitivity spikes tend to be present at the start and end layers.

5.3 Kurtosis

The Kurtosis, measuring the deviation from normal distribution in tailedness and peakedness (DeCarlo, 1997), can be formulated as the standardized fourth population moment about the mean (as denoted in Equation 5.2).

$$k = \frac{\sum_{i=1}^{n}(w_i - \bar{W})^4/n}{(\sum_{i=1}^{n}(w_i - \bar{W})^2/n)^2} \tag{5.2}$$

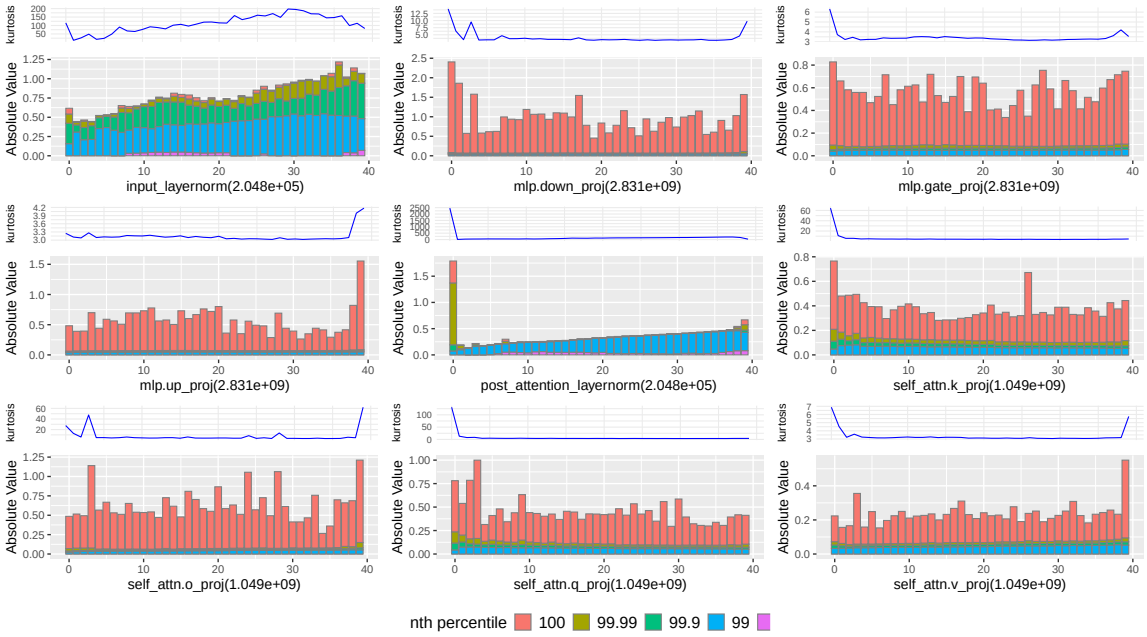


Fig. 5.4 This figure illustrates the Kurtosis and weight distributions across all layers of the Llama-2-13B model. The Kurtosis line plot shows the degree of deviation from a normal distribution, where a value of 3 indicates no deviation. The higher Kurtosis value denotes greater divergence thus greater difficulty to quantize accurately. This figure is best viewed in color and with zoom for clarity.

where w_i is the ith weight, n is the number of weights and $\bar{W}$ is the mean of the weights. Figure 5.5 presents an intuitive illustration of three distinct Kurtosis curves. A value of 3, termed mesokurtic, indicates the perfect conformance to the normal distribution. A larger Kurtosis greater than 3, known as leptokurtic, is illustrated as the blue thin bell curve with a narrow peak in Figure 5.5, whereas a lower Kurtosis less than 3, referred to as platykurtic, corresponds to a wider peak and flatter tails, as presented by the pink curve in Figure 5.5.

Presence of extremely large values of Kurtosis is a good indicator to locate layers that are challenging to quantize well with lower memory budget in large language models. Therefore, Kurtosis can be utilized to quantify the difficulty to quantize the weights across layers. Then layers with extreme Kurtosis can be isolated using the outlier detection algorithm discussed in the following section. Figure 5.4 presents the Kurtosis distribution across layers for the Llama-2-13B model.

For experiment, the Kurtosis metrics of the three Llama models are pre-calculated using the Pearson definition for each quantizable weight matrices by leveraging the `scipy.stats` library. The tool and instructions to generate Kurtosis metrics are described in appendix B.1.3.

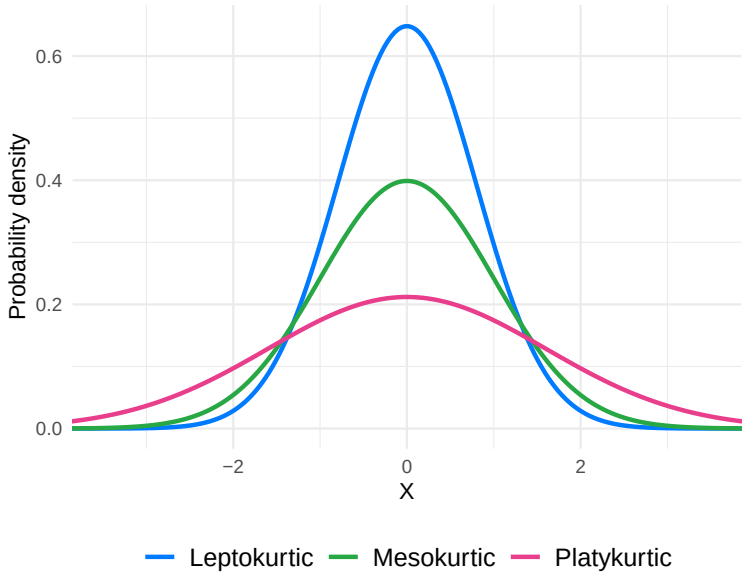


Fig. 5.5 This plot demonstrates three types of kurtosis curve. The green bell-shaped curve in the middle is mesokurtic ($k = 3$), indicating a normal distribution. The blue bell-shaped curve on the top is leptokurtic ($k > 3$), exhibiting sharper peak and flattened tails. Whereas the pink curve is platykurtic ($k < 3$).

For practical application of MXQ in production environment, the Kurtosis metrics could be calculated on-the-fly since the calculation is relatively light-weighted.

5.4 Outlier Detection Algorithm

The outlier detection algorithm is designed to single out layer with extreme sensitivity or Kurtosis values so that additional bit budget could be allocated to improve accuracy. Outliers are usually a small portion of overall dataset. Nevertheless, the proposed outliers detection algorithm is capable of prioritizing top outlier by rate of change so that only limited surplus bit budget is allocated. The layer-wise sensitivity or Kurtosis dataset is formulated in the Equation 5.3.

$$S = \{s_1, s_2, \cdots, s_n\} \tag{5.3}$$

The difference between two adjacent layers, denoted as the set D, is defined in Equation 5.4 to filter the sensitive data points.

$$D = \{s_2 - s_1, s_3 - s_2, \cdots, s_n - s_{n-1}\} \tag{5.4}$$

For dataset with an approximately ascending pattern, an alternative difference set, as denoted in Equation 5.5, is defined to use division instead of subtraction, which has the advantage of ignoring data points restore to normal range. This is beneficial to reduce false alarms and economize bit budget.

$$D = \left\{ \frac{s_2}{s_1}, \frac{s_3}{s_2}, \cdots, \frac{s_n}{s_{n-1}} \right\} \tag{5.5}$$

The set D is assumed to follow the normal distribution. Therefore, zscore can be leveraged to isolate the outliers. Constrained by memory, only top m outliers are considered. The Equation 5.6 defines the rule to identify top m sensitive points:

$$\begin{aligned} Top_m(D') &= \left\{ d \in D' \mid Rank(d, D') \leq m \right\} \\ D' &= \left\{ d \in D \mid \frac{|d - \mu|}{\sigma} > 3 \right\} \\ \sigma &= \frac{1}{n-1} \cdot \sqrt{\sum_i^n (d_i - \mu)^2} \\ \mu &= \frac{1}{n} \cdot \sum_i^n d_i \end{aligned} \tag{5.6}$$

where $Rank(d, D')$ gives the position of d when D' is sorted in descending order. To avoid the influence of extreme values, the mean and standard deviation are calculated using the trimmed approach, where the 5% smallest and largest data points are discarded. The actual implementation leverages the `scipy.stats` library. Finally, the sensitive layer can be

identified by Equation 5.7, which adds 1 to the indices returned by Equation 5.6 since $|D| = |S| - 1 = n - 1$.

$$\begin{aligned} I' &= \{i+1 \mid i \in ord(x, Top_m(D')) \, \forall x \in Top_m(D')\} \\ ord(x,S) &= \{i \mid x_i = x, x_i \in S, i \in \{1,2,\cdots,n\}\} \end{aligned} \tag{5.7}$$

Algorithm 1 presents the pseudo-code to locate the outliers, given an array of sensitivity scores or Kurtosis metrics.

Algorithm 1 The outlier detection algorithm

Input
    S Array of sensitivity scores or kurtosis metrics
    m Number of top outliers to return
    t Method to construct the difference set
Output
    I' Array of outlier indices
Require $S = \{s_1, s_2, \cdots, s_n\}$
Ensure $m \geq 1$
Ensure $t ==$'subtract' or $t ==$'divide'
if $t ==$ 'subtract' **then**
    $D \leftarrow \{s_2 - s_1, s_3 - s_2, \cdots, s_n - s_{n-1}\}$
else
    $D \leftarrow \left\{\frac{s_2}{s_1}, \frac{s_3}{s_2}, \cdots, \frac{s_n}{s_{n-1}}\right\}$ ▷ suppress data points restore to normal range
end if
$\mu \leftarrow \frac{1}{n} \cdot \sum_i^n d_i$
$\sigma \leftarrow \frac{1}{n-1} \cdot \sqrt{\sum_i^n (d_i - \mu)^2}$
$i, n \leftarrow 0, |D|$
$D' \leftarrow \{\}$
while $i \leq n$ **do**
    $z_i \leftarrow \frac{|d_i - \mu|}{\sigma}$
    if $z_i > 3$ **then**
        $D' \leftarrow D' \cup \{(d_i, ord(z_i)\}$
    end if
end while
$D'_m \leftarrow sort(D')[:m]$
$I' \leftarrow \{\}$
$i, n \leftarrow 0, |D'_m|$
while $i \leq n$ **do**
    $I' \leftarrow I' \cup \{d'_{m_i}[1] + 1\}$
end while
return I'

5.5 SensiBoost and KurtBoost Methods

5.5.1 Method Description

To explore the research question 2, this section describes how Kurtosis metrics and sensitivity score of large language model are utilized to enhance the quantization accuracy with a tiny increment in bit budget. The new approaches, referred to as KurtBoost and SensiBoost respectively, are implemented by identifying the sensitive layers using the outlier detection algorithm explained in the previous section 5.4.

The key steps of the SensiBoost and KurBoost are as follows:

1. Load the pre-calculated sensitivity score or Kurtosis metrics for the model being quantized.

2. Identify layers for additional allocation using the outlier detection algorithm according to the top-*m* setting.

3. Allocate normal budget to non-sensitive layers, assign additional budget to sensitive layers according to the boost stop setting.

4. Apply quantization using the underlying quantization method.

The amount of surplus budget for the sensitive layers, namely boost stops, can be specified by the number of stops to increase. When boost stops go beyond the maximum bit budget of the underlying quantization method, the maximum bit budget takes effect. Table 5.1 presents the range of bit budgets of HQQ.

For instance, when the base bit budget is 4.13, a setting of 2-stop will quantize the sensitive layers with a bit budget of 4.51. However, when the base bit budget is 8.25, a 2-stop increment request only results in 1 stop, i.e. a bit budget at 8.51.

The number of the layers targeted for extra allocation can be restricted based on the descending rank of sensitivity scores or Kurtosis metrics. The resulting layers, referred to as top *m* layers, enable further control over the allocation of limited extra memory budget. The actual layers eligible for additional allocation might be fewer than the specified value *m*. These layers may also vary across modules. For modules without evident outliers, no extra memory is assigned to them. Lastly, when *m* is set to 0, all layers identified by the outlier detection algorithm are considered for additional allocation.

Table 5.1 HQQ bit budgets

stop	**budget**	b_1	g_1	b_2	g_2
1	2.13	2	128	8	128
2	2.25	2	64	8	128
3	2.51	2	32	8	128
4	3.13	3	128	8	128
5	3.25	3	64	8	128
6	3.51	3	32	8	128
7	4.13	4	128	8	128
8	4.25	4	64	8	128
9	4.51	4	32	8	128
10	8.13	8	128	8	128
11	8.25	8	64	8	128
12	8.51	8	32	8	128

5.5.2 Experiments

To assess the effectiveness of the proposed SensiBoost and KurtBoost methods, models quantized using these approaches were evaluated using the WikiText-2 and C4 dataset to measure the perplexity scores. For each proposed methods, various boost stop and top-*m* configurations were benchmarked. Specifically, these experiments involved benchmarking two boost stop settings (2 and 3) and four top-*m* values (1, 2, 3, and 0) across three Llama models under six base bit budget configurations. Furthermore, ablation studies were included to validate the efficacy of the proposed methods. The ablation tests disregard the layers identified by SensiBoost or KurtBoost methods. Conversely they assign extra budgets to the layers selected randomly from a set that explicitly excluding the layers identified by SensiBoost or KurtBoost. To ensure fair comparison, the amount of extra memory and the layers are identical to SensiBoost or KurtBoost. The complete permutations of the test cases are summarized in Table C.2 where a total of 576 tests were conducted.

The comparisons of the different approaches were made among SensiBoost, KurtBoost, corresponding ablation methods, HQQ and MXQ, which are presented in Table 5.2.

To simplify the overall assessment of the diverse approaches, the win-tie-loss summary data is reported to enable qualitatively analysis. These data are aggregated across various evaluation criteria such as bit budget configuration, boost stop, evaluation dataset and top-*m*.

Table 5.2 Assessment matrix of various approaches

Method	SB	KB	SBAB	KBAB	HQQ	MXQ
SB[1]	-	X	X	-	X	X
KB[2]	-	-	-	X	X	X
SBAB[3]	-	-	-	-	-	-
KBAB[4]	-	-	-	-	-	-
HQQ	-	-	-	-	-	-
MXQ	-	-	-	-	-	-

[1] SB denotes the SensiBoost method.

[2] KB denotes the KurtBoost method.

[3] SBAB denotes the ablation test for SensiBoost method.

[4] KBAB denotes the ablation test for KurtBoost method.

5.5.3 Ablation Test

Ablation study was integrated into the experiments to validate the effectiveness of the proposed SensiBoost and KurtBoost methods, ensuring they outperform random choices of layers to allocate small portion of surplus budget. The ablation tests were carried out using random selection and explicitly avoiding choosing layers that could be probably selected by either SensiBoost or KurtBoost.

Formally, given I' as defined in Equation 5.7, I'_s is the sensitive layers identified by SensiBoost, I'_k denotes the ones for KurtBoost, the corresponding ablation test layer choices J_s and J_k are defined by Equation 5.8.

$$\begin{aligned} J_s &= \{h_1, h_2, \cdots, h_p \mid h_i \in \hat{I}, h_i \neq h_j \forall i \neq j, p = |I'_s|\} \\ J_k &= \{h_1, h_2, \cdots, h_q \mid h_i \in \hat{I}, h_i \neq h_j \forall i \neq j, q = |I'_k|\} \\ \hat{I} &= I \setminus (I'_s \cup I'_k) \\ I &= \{1, 2, \cdots, n\} \end{aligned} \tag{5.8}$$

where n is the number of layers in a particular large language model.

For example, suppose the set $\{2, 31\}$ represents the sensitive layers discovered by the SensiBoost method, whereas the set $\{1, 30, 31\}$ is identified by the KurtBoost approach, then the set $\{3, 28\}$ is a valid choice for ablation test of SensiBoost under top $m = 3$. Likewise, the $\{4, 28, 29\}$ is a legitimate ablation test configuration for KurtBoost. However, the set $\{2, 28, 29\}$ is invalid for ablation test of KurtBoost, since it contains the layer 2 which could potentially enhance quantization accuracy since it is considered as a sensitive lay by the SensiBoost method.

This chapter includes two sets of ablation tests designed to validate the efficacy of the SensiBoost and KurtBoost approaches. These tests were conducted under the same configurations as their non-ablation counterparts. Specifically, the configurations consist of two boost stop values and four top-*m* settings, evaluated across three Llama models under six base bit budget configurations. The specific parameter values are presented in Table C.2 in the appendix.

Win-tie-loss scores were used to qualitatively analyze the proposed methods. These scores were aggregated from the perplexity results benchmarked according to Table C.2 and paired based on Table 5.2. Specifically, all perplexity scores were rounded to two decimal places. The perplexity of the primary method (SensiBoost or KurtBoost) was then subtracted from that of the comparison method to determine the win-tie-loss score. A negative difference awarded the primary method 1 win, a difference of zero awarded 1 tie, and a positive difference awarded 1 loss. Finally, the win-tie-loss scores were aggregated across six quantization configurations, two stop settings, four top-*m* settings, and two evaluation datasets, providing a summarized win-tie-loss analysis for various method pairs across the three Llama models.

5.5.4 Results and Analysis

The overall results are presented in this section to qualitatively assess the effectiveness of SensiBoost and KurtBoost by leveraging win-tie-loss comparison. The win-tie-loss diagram, presented in Figure 5.6, includes the comparisons between the proposed methods (indicated by the row labels) and their ablation variants as well as baselines such as HQQ and MXQ, across three Llama models (denoted by the column labels).

Regarding the comparison to the baselines, as anticipated, both SensiBoost and KurtBoost outperform the baseline methods HQQ and MXQ due to the allocation of additional bit budgets. However, SensiBoost's relatively low win rates (53% against HQQ and 70% against MXQ) on the Llama-2-13B model suggest that achieving significant improvements in larger models with a limited extra memory budget is challenging. KurtBoost performs slightly better than SensiBoost on the Llama-2-13B, achieving 66% win rate against HQQ and 75% against MXQ.

In the context of ablation testing, both methods generally outperform their ablation variants. However, the 20% win rate and 61% tie rate on the Llama-2-13B model suggest that SensiBoost struggles to surpass its ablation counterpart when applied to larger models. In contrast, KurtBoost consistently demonstrates a slight advantage over SensiBoost, achieving higher win rates and lower loss rates across all three models. Detailed perplexity results for SensiBoost's ablation tests are provided in Table C.5 (4-bit) and Table C.6 (3-bit). Similarly,

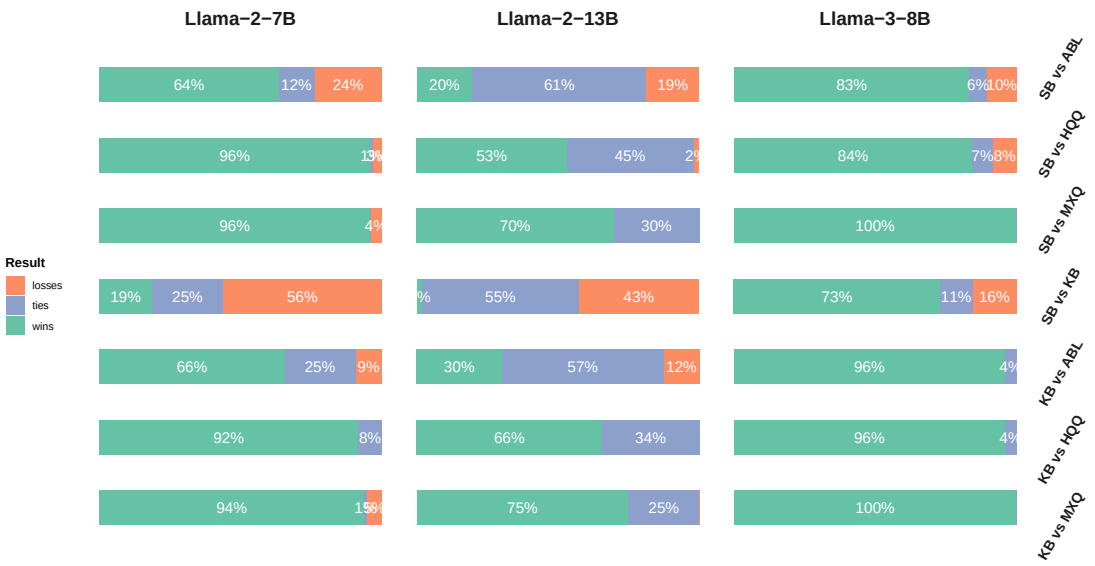


Fig. 5.6 This figure illustrates the win-tie-loss performance of the SensiBoost (denoted as "SB") and KurtBoost (denoted as "KB") methods compared to their ablation test (labeled as "ABL") as well as the baseline methods HQQ and MXQ, across three Llama models. As anticipated, SensiBoost and KurtBoost outperform the baseline methods HQQ and MXQ due to the allocation of additional bit budgets. However, their relatively low win rates (53% against HQQ and 70% against MXQ in the case of SensiBoost, 66% against HQQ and 75% against MXQ for KurtBoost) on the Llama-2-13B model suggest that achieving significant improvements in larger models with a limited extra memory budget is challenging. SensiBoost consistently outperforms its ablation test variant. However, its comparison with the KurtBoost method reveals mixed outcomes: while SensiBoost underperforms on the two Llama-2 models, it demonstrates considerable advantages on the Llama-3-8B model. For optimal clarity, the figure is best viewed in color and with zoom.

the results for KurtBoost's ablation tests are presented in Table C.7 (4-bit) and Table C.8 (3-bit). These tables are included in the appendix for reference.

5.5.5 SensiBoost and KurtBoost Comparison

The previous section provides an overall comparison of the SensiBoost and KurtBoost approaches. A detailed and direct comparison of the two methods is presented in this section to reveal the relative advantages and disadvantages of the two methods under various scenarios.

As indicated by the win-tie-loss result in Figure 5.6, SensiBoost tends to be less performant than KurtBoost. However, a deeper examination reveals that SensiBoost requires less additional memory to achieve comparable performance to KurtBoost on the Llama-2-7B

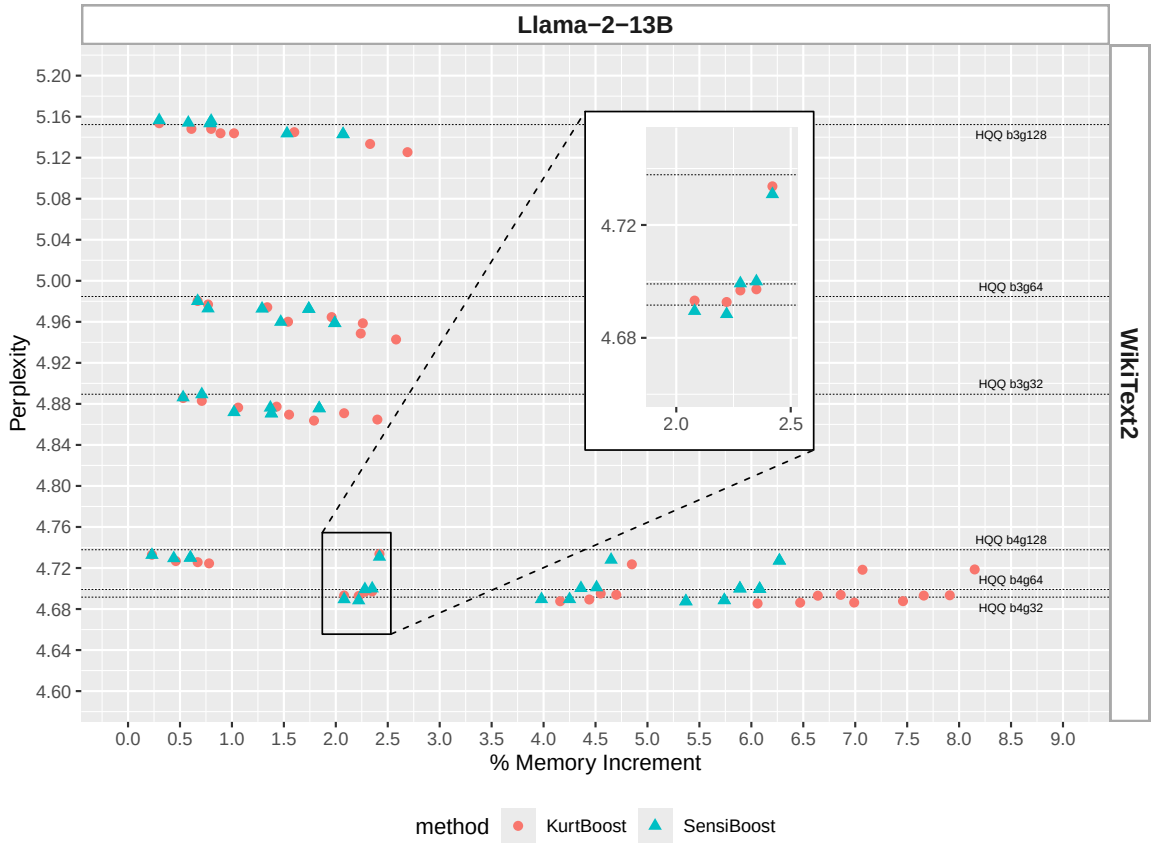


Fig. 5.7 This figure illustrates the perplexity performance of the SensiBoost and KurtBoost approaches evaluated on the Llama-2-13B model using the WikiText2 dataset. The green triangles, representing the SensiBoost method, are positioned closer to the y-axis, indicating that SensiBoost requires less additional memory to achieve comparable performance to KurtBoost. Notably, SensiBoost exhibits a slight advantage over KurtBoost, requiring approximately 2% more bit budget to attain a near-minimal perplexity score, as emphasized in the magnified sub-plot. For optimal interpretation, the figure is best viewed in color and with zoom.

and Llama-2-13B models. As demonstrated in Figure 5.7, SensiBoost exhibits a slight advantage over KurtBoost in identifying optimal quantization configuration where it requires approximately 2% more bit budget to attain a near-minimal perplexity score, as highlighted in the magnified sub-plot. This phenomenon replicates to the C4 dataset (Figure C.3) and the Llama-2-7B model (Figure C.4).

On the other hand, however, the situation is reversed on the Llama-3-8B model, where KurtBoost is more effective in discovering optimal quantization configuration which is both memory-efficient and yielding better accuracy, as illustrated in Figure C.5. The Kurtosis metrics are generated individually for distinct modules in each layer. In contrast, the sensi-

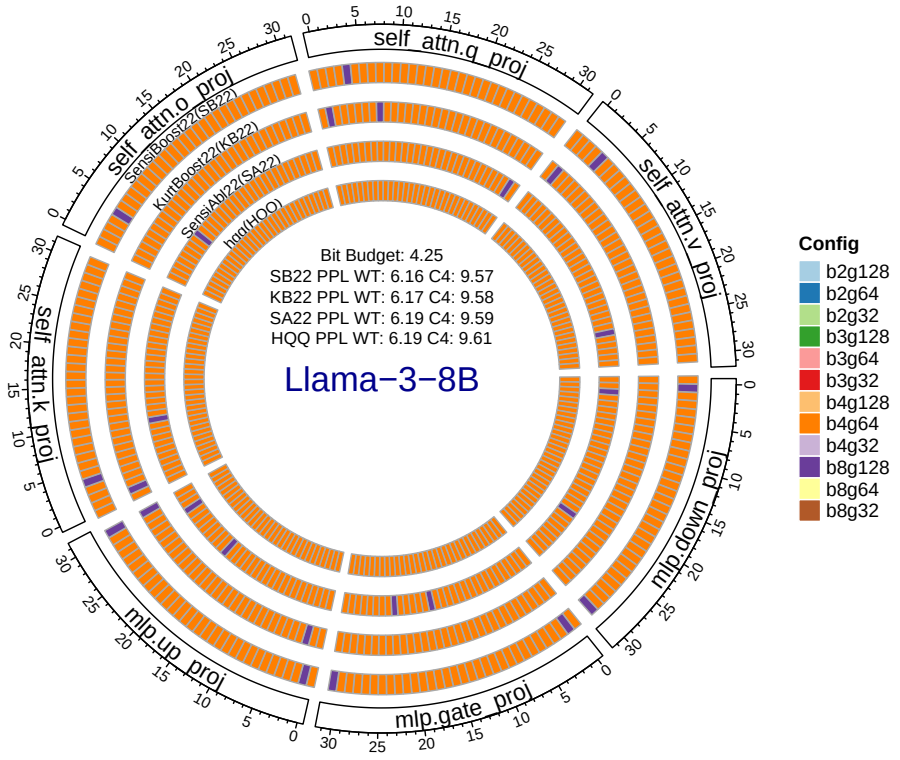


Fig. 5.8 This figure presents a comparison of quantization configuration allocations as determined by SensiBoost, KurtBoost, HQQ and MXQ for the Llama-3-8B model, under a bit budget of 4.25. The colored rings represent the assigned quantization configurations denoted as b2g128 through b8g32, while the text in the center indicates the perplexity scores. As demonstrated by this figure the SensiBoost with a boost stop value of 2 and top-m 1, as denoted by SB22, yields a perplexity score of 6.16 on WikiText2, 9.57 on C4, outperforming the HQQ baseline, its ablation variant (SA22) and KurtBoost (KB22). For optimal clarity, the figure is best viewed in color and with zoom.

tivity scores are measured by following the neural-network computation sequence, where some modules share the same sensitivity score as they are not standalone computation unit. Therefore, the KurtBoost approach may yield more nuanced quantization configurations that are more memory-efficient, as demonstrated in Figure 5.8, where the `self_attn.o_proj`, `mlp.down_proj` and `mlp.gate_proj` modules are not assigned extra budget. The data plotted in Figure 5.7 and Figure C.3 are presented in Table C.3 (4-bit) and Table C.4 (3-bit) in the appendix respectively.

In conclusion, both SensiBoost and KurtBoost demonstrate advantages over the baselines and their ablation variants, indicating the effectiveness of the two approaches. Both methods enable model accuracy enhancement by using approximately 2% additional memory budget.

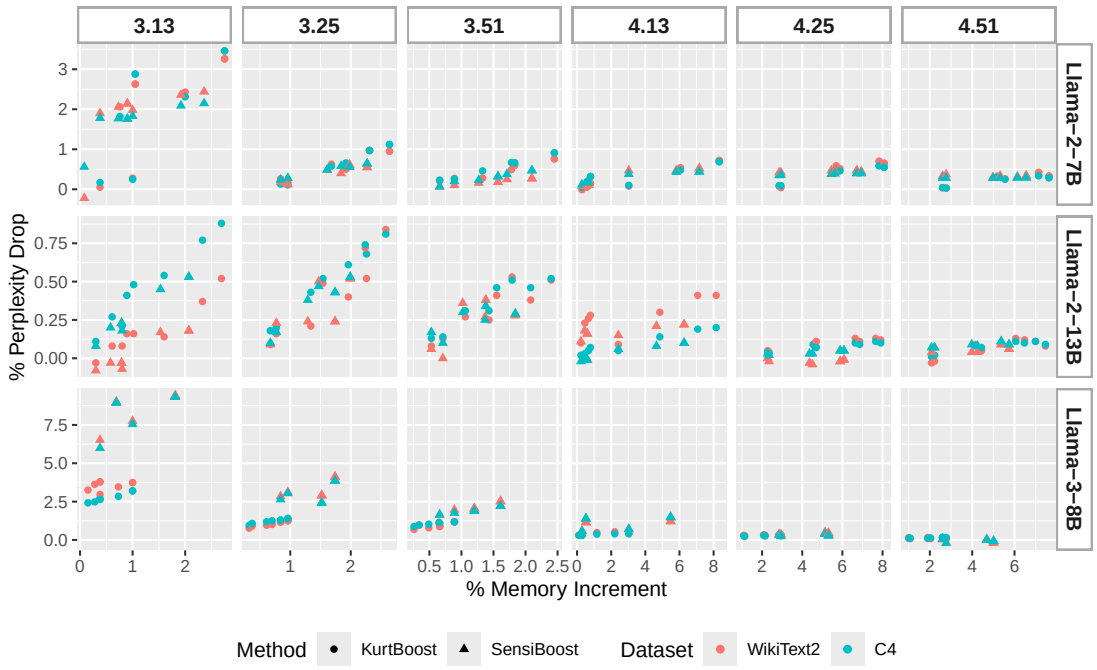


Fig. 5.9 This figure compares the SensiBoost and KurtBoost methods' performance on quantization accuracy enhancement with additional budget. The X axis denotes the percentage of additional budget assigned. The Y axis represents the percentage of perplexity drop. The improvement is more pronounced around the 3-bit range on the two smaller models Llama-2-7B and Llama-3-8B respectively. Notably, SensiBoost, denoted as triangles, exhibits more aggressive improvement with perplexity drop up to 9% on Llama-3-8B under a budget of 3.13. This figure also demonstrates the challenge of achieving substantial quantization accuracy elevation at higher bit budgets. For optimal clarity, the figure is best viewed in color and with zoom.

Specifically, the improvement is more pronounced in the 3-bit range with perplexity drop up to 9% on Llama-3-8B archived by the SensiBoost as illustrated in Figure5.9. This figure also demonstrates the challenge of achieving substantial quantization accuracy improvements at higher bit budgets, as evidenced by the notably flat pattern in the sub-plots for 4.25 and 4.51 configuration.

5.6 SensiMiLP and KurtMiLP Methods

5.6.1 Method Description

This section explores the potential enhancement by leveraging layer-wise sensitivity and Kurtosis with strict memory constraint. In contrast to the MXQ approach which employs

sum of Frobenius norm as objective, the sensitivity-based approach leverages the sum of quant score as defined in Equation 5.9. It assigns lower quant score to higher bit width and smaller group size, indicating better quantization accuracy can be achieved with larger memory budget. On the other hand, it penalizes lower bit width and larger group size with higher quant score. Specifically, it leverages the inverse of the storage function defined in Equation 4.2 as the basis. Additionally the sensitivity scores of 2-bit and 3-bit are scaled with a factor of 4 and 2 respectively to enlarge the difference between bit ranges. Subsequently, the quant score is further scaled with the percentage of parameters over the total parameters. This is necessary since each parameter should contribute to overall model equally. Finally, to allocate sensitive modules with extra bit budget, the corresponding cost scores are multiplied by 2. Equation 5.9 formulates the quant score calculation scheme:

$$
\begin{aligned}
quantScore(i,j) &= \frac{sensiScale(i,j)\cdot bitScale(b1)}{b1+2\cdot\frac{b2}{g1}+\frac{32}{g1\cdot g2}}\cdot\frac{100\cdot params_{i,j}}{total\_params} \\
sensiScale(i,j) &= \begin{cases} 2 & : \quad i\in I'_j \\ 1 & : \quad otherwise \end{cases} \\
bitScale(b) &= \begin{cases} 1 & : \quad b_1>3 \\ 2 & : \quad b_1=3 \\ 4 & : \quad b_1<3 \end{cases}
\end{aligned}
\tag{5.9}
$$

where $i\in[0,n]$ is the layer number, j represents the module within a given layer and I' is defined by Equation 5.7.

With sensitivity-based objective formulation in place, the F term in the Equation 4.3 becomes C which is a vector representation of $quantScore(i,j)$. The updated MiLP formulation is denoted in the Equation 5.10.

$$
\begin{aligned}
\arg\min_{X} \quad & C\cdot X \\
\text{s.t.} \quad & S\cdot X\leq\beta\,, \\
& A\cdot X=\begin{pmatrix}1 & 1 & \cdots & 1\end{pmatrix}^{T}_{1\times N}\,,
\end{aligned}
\tag{5.10}
$$

$$
\begin{aligned}
\text{where} \quad & X = \begin{pmatrix} x_1 & x_2 & \cdots & x_M \end{pmatrix}^T, \\
& x_j \in \{0,1\} \quad \forall j \in \{1,\ldots,M\}, \\
& C = \begin{pmatrix} c_1 & c_2 & \cdots & c_M \end{pmatrix}, \\
& S = \begin{pmatrix} s_1 & s_2 & \cdots & s_M \end{pmatrix}, \\
& A = \begin{pmatrix}
\overbrace{1\cdots1}^{|C|} & \overbrace{0\cdots0}^{|C|} & \cdots & \overbrace{0\cdots0}^{|C|} \\
\overbrace{0\cdots0}^{|C|} & \overbrace{1\cdots1}^{|C|} & \cdots & \overbrace{0\cdots0}^{|C|} \\
\vdots & \vdots & \ddots & \vdots \\
\overbrace{0\cdots0}^{|C|} & \overbrace{0\cdots0}^{|C|} & \cdots & \overbrace{1\cdots1}^{|C|}
\end{pmatrix}_{N\times M}.
\end{aligned}
$$

5.6.2 Experiments

To evaluate the effectiveness of the proposed SensiMiLP and KurtMiLP methods, perplexity on the WikiText-2 and C4 dataset was measured on models quantized using these approaches. For each proposed methods, various top-m configurations were explored. Specifically, these experiments involved benchmarking three top-m values (1, 2, 3) across three Llama models under 21 bit budget configurations. Furthermore, ablation studies were included to validate the efficacy of the proposed methods. The ablation tests simply ignore the layers identified by SensiMiLP and KurtMiLP methods, assigning normal budgets to them instead. The complete permutations of the test cases are summarized in Table C.9 where a total of 504 tests were conducted.

Table C.10 presents the comparison matrix to assess the performance of SensiMiLP and KurtMiLP, their corresponding ablation variants, HQQ and MXQ, which results in a total seven performance comparisons. To facilitate overall qualitative analysis of the proposed approaches, the win-tie-loss plot is reported, which aggregates the experiment results across evaluation criteria such as bit budget configuration, evaluation dataset and top-m.

5.6.3 Results and Analysis

The overall results are presented in this section to qualitatively assess the effectiveness of SensiMiLP and KurtMiLP by leveraging win-tie-loss comparison. The win-tie-loss diagram, presented in Figure 5.10, includes the comparisons between the proposed methods (indicated

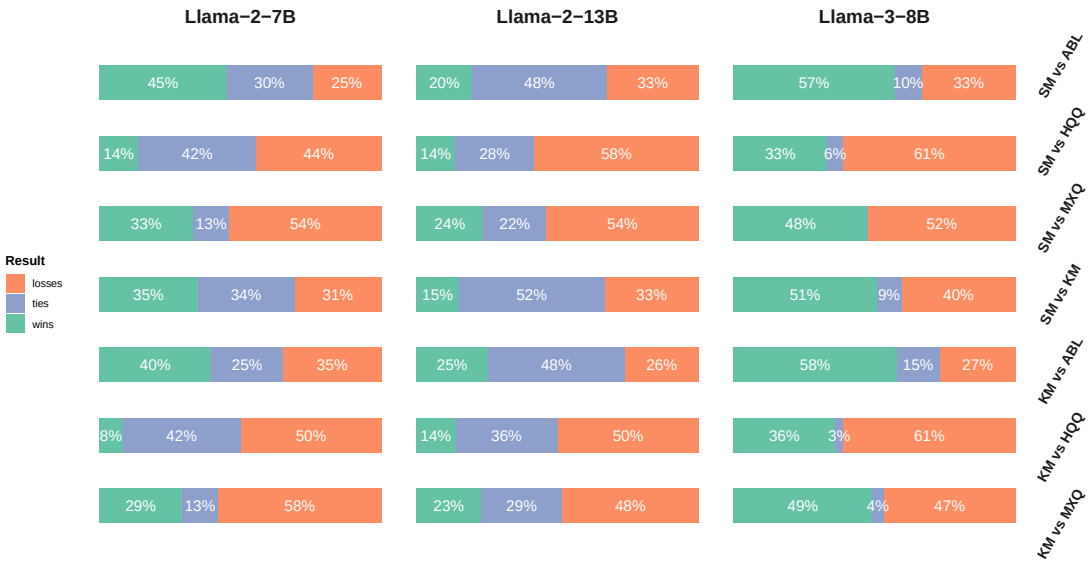


Fig. 5.10 This figure illustrates the win-tie-loss performance of the SensiMiLP (denoted as "SM") and KurtMiLP (denoted as "KM") methods compared to their ablation test (labeled as "ABL") as well as the baseline methods HQQ and MXQ, across three Llama models. For optimal clarity, the figure is best viewed in color and with zoom.

by the row labels) and their ablation variants as well as baselines such as HQQ and MXQ, across three Llama models (denoted by the column labels).

Compared to the baselines, both SensiMiLP and KurtMiLP under-perform the baseline methods HQQ and MXQ. With respect to ablation test, both methods slightly surpass their ablation variants with exception on the Llama-2-13B model, where SensiMiLP achieves a win rate of 20%, a loss rate of 33%. On the other hand, KurtMiLP demonstrates slight better result on Llama-2-13B with a win rate of 25% and a loss rate of 26%. The detailed perplexity results for SensiMiLP and KurtMiLP are presented in Table C.12 (3-bit), Table C.11 (4-bit) and Table C.13 (other bits) in the appendix for reference. Complete perplexity results of the SensiMiLP ablation tests are presented in Table C.15 (3 and 4-bit) and Table C.16 (other bits) in the appendix. Similarly, Table C.17 (3 and 4-bit) and Table C.18 (other bits) present the ablation test results of the KurtMiLP approach. These tables are included for reference purpose.

5.7 Discussion

Sensitivity-based methods, such as SensiBoost and SensiMiLP, have two notable limitations. First, the calculation of sensitivity scores is highly dependent on the model architecture,

making it challenging to generalize these methods to large language models (LLMs) with significantly different architectures. Second, the process of measuring sensitivity scores is both computationally and memory intensive, as it requires a layer-by-layer forward pass using both full-precision and quantized weights. In contrast, Kurtosis-based methods, such as KurtBoost and KurtMiLP, are not tightly coupled with the underlying model architecture and are comparatively efficient to compute. Despite their simplicity, Kurtosis-based methods achieve slightly better performance to sensitivity-based approaches, making them a viable alternative to SensiBoost and SensiMiLP.

Chapter 6

Conclusion

This thesis addresses the lack of flexibility in state-of-the-art quantization methods for optimally allocating memory budgets across layers to maximize quantization accuracy while maintaining overall memory efficiency. The first part introduces MXQ, a novel scheme designed to balance the trade-off between memory and accuracy in the quantization of large language models. MXQ expands the range of memory-accuracy trade-offs, exploring a total of 259 bit budgets ranging from 2.13 to 8.51 bits. This broader selection enables more aggressive memory reduction while minimizing accuracy degradation. Experimental results demonstrate that MXQ achieves a 77% reduction in memory usage with less than 5% accuracy loss, effectively addressing the first research question. In the second part, the thesis proposes various enhancements to MXQ under both strict and relaxed memory constraints. Experimental evaluations show that the SensiBoost and KurtBoost methods outperform the HQQ baseline under relaxed constraints. Notably, KurtBoost, which is computationally efficient and independent of the underlying model architecture, achieves superior performance compared to sensitivity-based approaches. These findings effectively address the second research question.

The mixed quantization (MXQ) represents an effective approach to optimizing the balance between model accuracy and memory consumption. This method unlocks a broad spectrum of quantization options, enabling the creation of compressed models that maintain performance comparable to the baseline with a simple specification of bit budget. By leveraging latent features such as sensitivity and Kurtosis, MXQ achieves further improvement in quantization accuracy, reducing perplexity up to 9% with approximately 2% additional budget. Integrating the proposed MXQ methods into the repertoire of quantization utilities would provide a valuable supplement, enhancing the state-of-the-art quantization methods in an orthogonal way and enriching the tools available for optimizing increasingly larger neural network models.

6.1 Contributions

The contributions of this thesis are as follows: First, it introduces an effective quantization method, MXQ, designed for precise budget control and flexible memory-accuracy trade-offs. It proposes a novel optimisation framework that minimizes the sum of quantization errors or sensitivity scores in the objective while incorporating a storage cost function as constraint. This formulation is efficiently solved using a mixed-integer linear programming (MiLP) algorithm, enabling fast and effective quantization. Second, the layer-wise sensitivity to quantization error is empirically explored on multiple transformer-based LLM families, revealing the robustness of sensitivity within a family of models and their fine-tuned variants. Third, a simple outlier detection algorithm is implemented to discover sensitive layers with sensitive scores or Kurtosis metrics. Finally, the SensiBoost and KurtBoost methods are implemented and outperform HQQ with only 2% increment in memory budget.

6.2 Limitations

Extensive experiments empirically reveal that allocating memory with significant budget disparities across layers tends to result in inferior quantization accuracy. Conversely, a more even allocation with smaller budget discrepancies generally achieves better quantization accuracy. This observation should be further explored theoretically and leveraged to improve budget assignment in both SensiMiLP and the MiLP formulation introduced in chapter 4.

Sensitivity-based methods, such as SensiBoost and SensiMiLP, have two notable limitations. First, the calculation of sensitivity scores is highly dependent on the model architecture, making it challenging to generalize these methods to large language models (LLMs) with significantly different architectures. Second, the process of measuring sensitivity scores is both computation and memory intensive, as it requires a layer-by-layer forward pass using both full-precision and quantized weights.

The MXQ method is designed to be compatible with any transformer-based models and quantization methods. However, the experiments were conducted using only HQQ on a selection of Llama family models. Evaluating MXQ method on limited method and models may not ensure comparable performance on other quantization algorithm and models. Additionally, end-to-end metrics such as IFEval, BBH, MATH Level 5, GPQA, MUSR and MMLU-PRO were not assessed in the experiments, which may limit the robustness of the true performance of the proposed method on real world tasks. Furthermore, the experiments did not consider the inference speed of the quantized model, as MXQ is built on-top of state-of-the-art method. Therefore it does not directly influence this aspect.

6.3 Future Work

Future work will focus on enhancing the accuracy of SensiMiLP and KurtMiLP by exploring alternative objective functions that minimize the disparity in quantization configurations across layers. Equally important is extending the generalizability of MXQ to diverse transformer-based language models and multi-modal models, as well as enabling its integration with other state-of-the-art quantization methods such as BnB which allows on-the-fly quantization. Additionally, incorporating end-to-end benchmarks, such as IFEval, BBH, GPQA, and MMLU-PRO, into the evaluation process would ensure the robustness of the proposed method's performance in real-world tasks.

References

Badri, H. and Shaji, A. (2023). Half-quadratic quantization of large machine learning models.

Badri, H. and Yahia, H. (2016). A non-local low-rank approach to enforce integrability. *IEEE Transactions on Image Processing*, 25(8):3562–3571.

Bai, H., Hou, L., Shang, L., Jiang, X., King, I., and Lyu, M. R. (2022). Towards efficient post-training quantization of pre-trained language models. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 1405–1418. Curran Associates, Inc.

Baji, T. (2018). Evolution of the GPU Device widely used in AI and Massive Parallel Processing. In *2018 IEEE 2nd Electron Devices Technology and Manufacturing Conference (EDTM)*, pages 7–9.

Bixby, R. E. (2012). A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, 2012:107–121.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Dantzig, G. B. (2002). Linear Programming. *Operations Research*, 50(1):42–47. Publisher: INFORMS.

DeCarlo, L. T. (1997). On the meaning and use of kurtosis. *Psychological Methods*, 2(3):292–307. Place: US Publisher: American Psychological Association.

Dettmers, T., Lewis, M., Shleifer, S., and Zettlemoyer, L. (2021). 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*.

Dettmers, T., Lewis, M., Shleifer, S., and Zettlemoyer, L. (2022). 8-bit Optimizers via Block-wise Quantization. arXiv:2110.02861 [cs].

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023a). QLoRA: Efficient Finetuning of Quantized LLMs. arXiv:2305.14314 [cs].

Dettmers, T., Svirschevski, R., Egiazarian, V., Kuznedelev, D., Frantar, E., Ashkboos, S., Borzunov, A., Hoefler, T., and Alistarh, D. (2023b). SpQR: A Sparse-Quantized Representation for Near- Lossless LLM Weight Compression. *arXiv preprint arXiv:2306.03078*.

Dettmers, T. and Zettlemoyer, L. (2023). The case for 4-bit precision: k-bit Inference Scaling Laws. In *Proceedings of the 40th International Conference on Machine Learning*, pages 7750–7774. PMLR. ISSN: 2640-3498.

Ding, N., Qin, Y., Yang, G., Wei, F., Yang, Z., Su, Y., Hu, S., Chen, Y., Chan, C.-M., Chen, W., Yi, J., Zhao, W., Wang, X., Liu, Z., Zheng, H.-T., Chen, J., Liu, Y., Tang, J., Li, J., and Sun, M. (2023). Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235. Publisher: Nature Publishing Group.

Face, H. (2024). Perplexity of fixed-length models.

Frantar, E. and Alistarh, D. (2022). Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. (2023). GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. arXiv:2210.17323 [cs].

Geman, D. and Reynolds, G. (1992). Constrained restoration and the recovery of discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(3):367–383.

Gleixner, A., Hendel, G., Gamrath, G., Achterberg, T., Bastubbe, M., Berthold, T., Christophel, P., Jarck, K., Koch, T., Linderoth, J., et al. (2021). Miplib 2017: data-driven compilation of the 6th mixed-integer programming library. *Mathematical Programming Computation*, 13(3):443–490.

Gu, Z., Gu, L., Eils, R., Schlesner, M., and Brors, B. (2014). circlize implements and enhances circular visualization in r. *Bioinformatics*, 30(19):2811–2812.

Guo, H., Greengard, P., Xing, E. P., and Kim, Y. (2024). LQ-LoRA: Low-rank Plus Quantized Matrix Decomposition for Efficient Language Model Finetuning. arXiv:2311.12023 [cs].

Gurobi Optimization, LLC (2023). Gurobi Optimizer Reference Manual.

Hans, D. M. (2024). Benchmarks for Optimization Software.

Huangfu, Q. and Hall, J. A. J. (2018). Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142.

Huggingface (2022). safetensors.

Hugh-Jones, D. (2024). *ggmagnify: Create a Magnified Inset of Part of a "Ggplot" Object*. R package version 0.4.1.9000, https://hughjonesd.github.io/ggmagnify/.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling Laws for Neural Language Models. arXiv:2001.08361 [cs, stat].

Kim, S., Hooper, C., Gholami, A., Dong, Z., Li, X., Shen, S., Mahoney, M. W., and Keutzer, K. (2023). SqueezeLLM: Dense-and-Sparse Quantization. *arXiv preprint arXiv:2306.07629*.

Lin, J., Tang, J., Tang, H., Yang, S., Dang, X., and Han, S. (2023). AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. *arXiv preprint arXiv:2306.00978*.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137.

Luenberger, D. G. and Ye, Y. (2021). *Linear and Nonlinear Programming*, volume 228 of *International Series in Operations Research & Management Science*. Springer International Publishing, Cham.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2016). Pointer Sentinel Mixture Models. arXiv:1609.07843 [cs].

Molchanov, P., Mallya, A., Tyree, S., Frosio, I., and Kautz, J. (2019). Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11264–11272.

Nagel, M., Baalen, M. v., Blankevoort, T., and Welling, M. (2019). Data-Free Quantization Through Weight Equalization and Bias Correction. pages 1325–1334.

Nagel, M., Fournarakis, M., Amjad, R. A., Bondarenko, Y., van Baalen, M., and Blankevoort, T. (2021). A White Paper on Neural Network Quantization. arXiv:2106.08295 [cs].

Pilgrim, M. (2009). *Serializing Python Objects*, pages 205–223. Apress, Berkeley, CA.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Reddi, V. J., Cheng, C., Kanter, D., and Mattson, P. a. e. (2020). MLPerf Inference Benchmark. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 446–459.

Samragh, M., Farajtabar, M., Mehta, S., Vemulapalli, R., Faghri, F., Naik, D., Tuzel, O., and Rastegari, M. (2023). Weight subcloning: direct initialization of transformers using larger pretrained ones. *arXiv preprint arXiv:2312.09299*.

Shuangbin Xu, Chen, M., Feng, T., Zhan, L., Zhou, L., and Yu, G. (2021). Use ggbreak to effectively utilize plotting space to deal with large datasets and outliers. *Frontiers in Genetics*, 12:774846.

Touvron, H., Martin, L., and Stone, K. e. a. (2023). Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs].

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Xu, Z., Chen, Y., Vishniakov, K., Yin, Y., Shen, Z., Darrell, T., Liu, L., and Liu, Z. (2023). Initializing models with larger ones. In *The Twelfth International Conference on Learning Representations*.

Zhang, F., Liu, Y., Li, W., Wang, X., and Bai, Q. (2025). A mixed quantization approach for data-free quantization of llms.

Appendix A

Experiment Procedures

A.1 Language Model Experiments

The results presented in the previous sections were produced by conducting experiments on the Llama family models. The Llama model experiments include quantization and perplexity evaluation.

The specialized harness tool named *lm-quant-toolkit* facilitates the tasks to quantize, evaluate language and vision transformer models using the proposed MXQ method as well as the baselines such as HQQ, AWQ, GPTQ. This harness is designed to perform long-running experiments. It tracks the experiment status and automatically resumes interrupted experiment from last failed point. It collects experiment duration, GPU memory consumption and key metrics such as perplexity and Open LLM LeaderBoard scores. It consolidates outputs from sub-tasks into experiment dataset in .csv format for further analysis and reporting.

The quantization and evaluation of LLMs require substantial computation and storage resources. The experiment environment's hardware and software configurations are presented in Table A.1.

The patched software need to be cloned from github, as noted under Table A.1, then be installed using the PIP's editable install method as they haven't been integrated in the upstream yet. Additional Python libraries are automatically installed when the lm-quant-toolkit is installed. For complete dependencies, please review the 'pyproject.toml' file of the lm-quant-toolkit project.

The three Llama models, published on Hugging Face by Meta, employed in this experiment are the Llama-2-7B(meta-llama/Llama-2-7b-hf), Llama-2-13B(meta-llama/Llama-2-13b-hf) and Llama-3-8B(meta-llama/Meta-Llama-3-8B). The main procedure to reproduce the language model experiment results consists of:

Table A.1 Experiment hardware/software configuration

Item	Configuration
CPU	Intel i9-14900KF
Memory	192 GiB
Disk	HP SSD FX900 Pro 2TB x 2
GPU	NVIDIA GeForce RTX 4090
OS	Ubuntu 24.04
Python	3.11.9
CUDA	12.5
PyTorch	2.4.1
lm-quant-toolkit	master[1]
AutoAWQ	0.2.5
AutoGPTQ	ea829c7 with the CUDA 12.5 patch[2]
HQQ	aad6868 with MXQ patch[3]

[1] The lm-quant-toolkit is published on github: https://github.com/ghAcctOfAuthor1/lm-quant-toolkit.

[2] The CUDA 12.5 patch can be fetched from: https://github.com/ghAcctOfAuthor1/AutoGPTQ.

[3] The MXQ patch can be fetched from: https://github.com/ghAcctOfAuthor1/hqq.

1. evaluate the FP16 baseline's perplexity metrics on WikiText2 and C4 dataset for original Llama models
2. quantize the selected Llama models using AWQ, GPTQ, HQQ with configurations b4g32, b4g64, b4g128, b3g32, b3g64 and b3g128
3. evaluate perplexity metrics on WikiText2 and C4 dataset for the AWQ, GPTQ and HQQ quantized models
4. quantize the selected Llama models using MXQ with a wide range of bit budgets ranging from 3.07 to 8.51
5. evaluate perplexity metrics on WikiText2 and C4 dataset for the MXQ quantized models
6. aggregate, analyze and visualize experiment results using R

The lm-quant-toolkit includes bash scripts to automate the aforementioned evaluation tasks. The results of the evaluation tasks, formatted as .csv files, are stored in the experiment directories specified in the corresponding task script. The quantized models are stored under the snapshot directory specified in the quantization scripts.

Table A.2 This table presents various bash scripts to conduct experiments and benchmarks of the proposed MXQ and baseline methods.

Script	Evaluation Task
quant-llm-hqq.sh	Apply HHQ algorithm to quantize selected Llama models using 12 different bit-group configurations.
quant-llm-awq.sh	Apply AWQ algorithm to quantize selected Llama models using 3 different bit-group configurations.
quant-llm-gptq.sh	Apply GPTQ algorithm to quantize selected Llama models using 6 different bit-group configurations.
quant-llm-mxq.sh	Apply MXQ algorithm to quantize selected Llama models using various bit budgets.
eval-llm-storage.sh	Load FP16 pristine and quantized models, measure GPU consumption and model file size for all models produced in quantization stage.
eval-llm-ppl-fp16.sh	Evaluate perplexity on WikiText and C4 dataset for the FP16 baseline on selected Llama models.
eval-llm-ppl-awq.sh	Evaluate perplexity on WikiText and C4 dataset for selected AWQ quantized Llama models.
eval-llm-ppl-gptq.sh	Evaluate perplexity on WikiText and C4 dataset for selected GPTQ quantized Llama models.
eval-llm-ppl-hqq.sh	Evaluate perplexity on WikiText and C4 dataset for selected HQQ quantized Llama models.
eval-llm-ppl-mxq.sh	Evaluate perplexity on WikiText and C4 dataset for selected MXQ quantized Llama models.

Appendix B

Experiment Utilities

B.1 The lm-quant-toolkit overview

The `lm-quant-toolkit` is a suite of tools to facilitate large neural network quantization research. It includes a quantization harness tool to drive quantization experiments on large language models and vision models, to collect and summarize experiment data for further analysis. It also includes tool to prepare experiment meta data and visualization tools to interpret experiment results. Specifically, `lm-quant-toolkit` consists of:

- LLM quantization harness tool
- ViT quantization harness tool
- FNorm Metadata Preparation Tool
- Kurtosis Metrics Measuring Tool
- Sensitivity Score Measuring Tool
- Calibration Dataset Generation Tool
- Visualization Tools

Most tools are implemented in Python and are extensively tested under the Python 3.11.9. The visualization tools are implemented in R. The usages of these tools are elaborated in the following sections. This section describes how to setup the `lm-quant-toolkit` and the companion visualization tools.

The Python tools depend on Python libraries such as transformers, datasets, numpy, PyTorch etc. A few Python libraries are patched to support MXQ. Specifically, required

patched dependencies include AutoGPTQ (for CUDA 12.5 compatibility), HQQ (support MXQ extension), lm_eval (for end-to-end LLM performance evaluation), clip_benchmark (for vision model evaluation). These dependencies are installed automatically as part of setup process. To setup the Python tools, follow this procedure:

1. Ensure Python and miniconda are installed
2. Create a Python virtual enivonrment using Python 3.11.9 and activate this enivonrment
3. Clone the `lm-quant-toolkit` project from https://github.com/schnell18/lm-quant-toolkit.git
4. Run the script `setup-harness.sh` under the root directory of the `lm-quant-toolkit` project

The visualization tools are R scripts to transform, aggregate and visualize experiment results. They are wrapped in bash scripts to automate the whole experiment loop, which consists of model quantization, perplex evaluation, memory consumption test and experiment report generation. The R visualization scripts can also be used separately. To setup the visualization tools, please follow this procedure:

1. Ensure a recent version of R, for instance R 4.4.1, is installed.
2. Optionally, RStudio could be installed to extend and trouble shoot the visualization tools in an intuitive enivonrment.
3. Install the third-party packages required by the visualization tools by running the script `setup-visualization.sh` under the root directory of the `lm-quant-toolkit` project.

B.1.1 LLM Quantization Harness Tool

This tool executes various quantization tasks and runs diverse evaluation benchmarks such as perplexity, GPU memory usage, quantized model storage. It also supports end-to-end LLM performance evaluation through the integration with the `lm-eval` tool. This harness tool works with various state-of-the-art quantization methods such as GPTQ, AWQ, BitsAndBytes and HQQ, which enables a fair comparison between the proposed methods and the state-of-the-art baselines. Furthermore, it facilitates the complex and time-consuming benchmarking tasks by offering resumption from failed subtasks, aggregate subtask's evaluation results. Lastly, this tool provides declarative CLI interface to ease complex experiment automation through shell scripting.

B.1.2 FNorm Metadata Preparation Tool

This tool calculates the Frobenius norms, a.k.a FNorm, of the quantization errors of all weight matrices inside a particular large language model. The FNorm meta-data are crucial to the MXQ quantization scheme as it guides MXQ to allocate optimal quantization configurations. This tool accepts a list of Hugging Face-compliant model identifiers. The output of this tool is a series of .csv files under specified directory. Each file contains the Frobenius norms for the 12 quantization configurations as defined in Table 4.1 for corresponding models.

The tool is implemented in Python and provides a convenient CLI interface to enable shell scripting. It is located separately in the `dump.py` file under the `src` folder in the `lm-quant-toolkit` project, which helps to reduce unnecessary dependencies. A typical usage is demonstrated in the code snippet as follows:

```
#!/bin/bash

TOOLKIT_DIR="$HOME/work/lm-quant-toolkit"
MODELS="meta-llama/Llama-2-7b-hf meta-llama/Llama-2-13b-hf meta-llama/Llama-3.1-8B"
mkdir -p /tmp/fnorm-dump
python $TOOLKIT_DIR/src/dump.py fnorm \
    --model $MODELS \
    --output-dir /tmp/fnorm-dump
```

B.1.3 Kurtosis Metrics Measuring Tool

This tool calculates the Kurtosis metrics of weight matrices layer-by-layer inside a particular large language model. The Kurtosis metrcis are crucial to identify sensitive layers to improve the accuracy of MXQ quantization. This tool accepts a list of Hugging Face-compliant model identifiers. The output of this tool is a series of .csv files under specified directory. Each file contains the Kurtosis metrics for corresponding models.

The tool is implemented in Python and provides a convenient CLI interface to enable shell scripting. It is included in the `dump.py` file under the `src` folder in the `lm-quant-toolkit` project. A typical usage is demonstrated in the code snippet as follows:

```
#!/bin/bash

MODELS="meta-llama/Llama-2-7b-hf meta-llama/Llama-2-13b-hf
↪  meta-llama/Meta-Llama-3-8B"
mkdir -p /tmp/kurtosis-dump
python ../src/dump.py kurtosis \
    --model $MODELS \
    --output-dir /tmp/kurtosis-dump
```

This code snippet demonstrates dumping the kurtosis metrics for the three Llama models into the `/tmp/kurtosis-dump` directory.

B.1.4 Sensitivity Score Measuring Tool

This tool calculates the sensitivity score of each layer of a particular large language model. The sensitivity score are crucial to identify sensitive layers to improve the accuracy of MXQ quantization. This tool accepts a list of Hugging Face-compliant model identifiers. The output of this tool is a series of .csv files, each contains the sensitivity score for corresponding model. These files are crucial inputs to guide the SensiBoost and Sensitivity-based MiLP.

The tool is implemented in Python and provides a convenient CLI interface to enable shell scripting. It is compatible with any transformer-based LLMs with an implementation of the popular Hugging Face transformers library. It is located separately in the `dump.py` file under the `src` folder in the `lm-quant-toolkit` project, which helps to reduce unnecessary dependencies. A typical usage is demonstrated in the code snippet as follows:

```
#!/bin/bash

TOOLKIT_DIR="$HOME/work/lm-quant-toolkit"
RESULT_BASE_DIR="/data/llm/mxq/results"
CALIB_DATASETS="bos pileval wikitext c4"
CONFIGS="b2g128 b2g64 b2g32 b3g128 b3g64 b3g32 b4g128 b4g64 b4g32 b8g128 b8g64
↪ b8g32"
MODELS="Qwen/Qwen2.5-7B Qwen/Qwen2.5-Coder-7B Qwen/Qwen2.5-Coder-7B-Instruct
↪ Qwen/Qwen2.5-Math-7B"

EXP_NAME=sensi_qwen25
RESULT_DIR=$RESULT_BASE_DIR/$EXP_NAME
mkdir -p $RESULT_DIR/data

for DS in $CALIB_DATASETS; do
    for CFG in $CONFIGS; do
        for MODEL in $MODELS; do
            SHORT_ID=$(echo $MODEL | cut -d/ -f2)
            OUT_FILE="${RESULT_DIR}/data/qwen25-sensi-${SHORT_ID}-${CFG}-${DS}.csv"
            python $TOOLKIT_DIR/src/dump.py sensi \
                --model $MODEL \
                --config $CFG \
                --calib-dataset $DS \
                --output-file $OUT_FILE
        done
    done
```

```
done
```

The code snippet demonstrates how to calculate the sensitivity scores for a series of Qwen2.5 models using 4 calibration datasets under 12 bit budgets.

B.1.5 Calibration Dataset Generation Tool

This tool generates a small synthensized dataset named branch of science (denoted as BoS, published on Hugging Face), which includes a few hundred of textual definitionsdefintions for science, art and business topics such as Mathematics, Physics, Chemstry, Law, Music and Journalism etc. The dataset is intended to validate if the sensitivity property generalize to diverse datasets.

The tool generates an initial dataset in .csv format which requires further processing. The output of this tool is random due to the generative nature of LLM. This tool requires a Llama-2-7B model being served with an OpenAI compatible RESTful API endpoint. User can either use a hosted API endpoint or deploy a local instance by following the instruction at the end of this section.

Once the API endpoint is secured, run the following script to generate the BoS dataset:

```
#!/bin/bash

TOOLKIT_DIR="$HOME/work/lm-quant-toolkit"

$TOOLKIT_DIR/utils/generate.py \
  --model="meta-llama/Llama-2-7b-chat-hf" \
  --variant="vLLM" \
  --topic-file=topics-l1.txt \
  --trace
```

Lastly, find the result in the csv files under current directory.

Local API endpoint

To deploy a local API endpoint using vLLM, create a virtual environment using `conda` as follows:

```
conda create -n vllm python=3.11 -y
conda activate vllm
pip install vllm==0.6.4.post1
```

Then configure and launch the API server

```
#!/bin/bash

vllm serve meta-llama/Llama-2-7b-chat-hf --dtype auto --api-key token-abc123
```

Watch the output vLLm to make sure it starts up successfully.

B.2 Visualization Tools

The visualization tools facilitate visualizing the experiment results and the weight distribution, and generating insights of the latent features to quantize LLMs more efficiently. Most visualization tools are implemented in R and leverages the open-source plot libraries such as ggplot2 (Wickham, 2016), circlize (Gu et al., 2014), ggbreak (Shuangbin Xu et al., 2021), ggmagnify (Hugh-Jones, 2024). They provide CLI interface to simplify integaration with the quantization harness tool.

These CLI tools support diverse options to allow user specify input dataset, select particular model or approach to plot. To get help on these specific CLI options, type `./plot_xxx.R --help` on command line prompt. For instance, to get help on the MXQ allocation visualization tool, you may run command as follows:

```
./plot-mxq-allocation.R --help
Usage: ./plot-mxq-allocation.R [options]

Options:
        -h, --help
                Show this help message and exit

        -m CHARACTER, --model=CHARACTER
                Model ID

        -b DOUBLE, --budget=DOUBLE
                Bit Budget

        -d CHARACTER, --baseline_data_dir=CHARACTER
                Data directory of baseline results

        -q CHARACTER, --quant_cfg_allot_file=CHARACTER
                The combined quant config allocation csv file

        --attempt1=CHARACTER
                The first attempt to plot

        --attempt2=CHARACTER
```

```
24                  The second attempt to plot
25
26          --fnorm
27                  Display FNorm value in the bar chart
```

B.2.1 Weight Distribution Visualization Tool

This tool enables visualizing layer-wised weight distribution of large language models. It is implemented as an R script, which provides a convenient CLI interface to enable shell scripting. Figure 5.4 is an example generated by this tool. Given a weight distribution metrics csv file, it produces a pdf file under the `pdfs` with 3x3 sub-plots of column digrams for the 9 modules in the Llama family models.

The tool is named `plot-wdist-llm.R` and located under the `data-vis` folder in the `lm-quant-toolkit` project. A typical usage is demonstrated in the code snippet as follows:

```
1  #!/bin/bash
2
3  TOOLKIT_DIR="$HOME/work/lm-quant-toolkit"
4
5  $TOOLKIT_DIR/data-vis/plot-wdist-llm.R -m Llama-2-7b-hf
```

B.2.2 Perplexity vs Bit Budget Visualization Tool

This tool enables visualizing the relationship between perplexity and bit budget for diverse MXQ experiments against their baselines. The generated diagram shows how memory reduction affects perplexity, which facilitates memory-accuracy trade-off. Figure 4.2 is an example produced by this tool.

The tool is named `plot-ppl-mem.R` and located under the `data-vis` folder in the `lm-quant-toolkit` project. It accepts a csv file containing the perplexity metrics of MXQ and its baselines. The output are series of PDF files corresponding to the models defined in the input file, which are placed under the `pdfs` subfolder. A typical usage is demonstrated in the code snippet as follows:

```
1  #!/bin/bash
2
3  TOOLKIT_DIR="$HOME/work/lm-quant-toolkit"
4
5  $TOOLKIT_DIR/data-vis/plot-ppl-mem.R -d data/combined.csv
```

B.2.3 Quantization Speed Comparison Visualization Tool

This tool generates column digrams to explore the quantization speed among various approaches. Figure 4.4 is an example generated by this tool. The tool is also implemented as an R script, which provides a convenient CLI interface to enable shell scripting. The tool is named `plot-quant-speed.R` and located under the `data-vis` folder in the `lm-quant-toolkit` project. Given a combined perplexity metrics csv file, it produces a column digrams with x-axis in log-scale. Similar to other tools, the PDF file is placed under the `pdfs` subfolder. A typical usage is demonstrated in the code snippet as follows:

```
#!/bin/bash

TOOLKIT_DIR="$HOME/work/lm-quant-toolkit"

$TOOLKIT_DIR/data-vis/plot-quant-speed.R -d data/combined.csv
```

B.2.4 GPU Memory Usage Visualization Tool

This tool generates column digrams to present the actual GPU memory consumption of LLMs quantized by diverse methods. Figure 4.3 is an example generated by this tool. The tool is also implemented as an R script, which provides a convenient CLI interface to enable shell scripting. The tool is named `plot-mem-consumption.R` and located under the `data-vis` folder in the `lm-quant-toolkit` project. Given a combined perplexity metrics csv file, it produces a column digrams of GPU memory usage in Giga-byte. Similar to other tools, the PDF file is placed under the `pdfs` subfolder. A typical usage is demonstrated in the code snippet as follows:

```
#!/bin/bash

TOOLKIT_DIR="$HOME/work/lm-quant-toolkit"

$TOOLKIT_DIR/data-vis/plot-mem-consumption.R -d data/combined.csv
```

B.2.5 Quantization Configuration Allocation Visualization Tool

This tool offers insights into the way MXQ and its variants allocate bit budget to modules and layers. The variants, a.k.a. attempt, to include in the plot are configurable. A maximum of 4 variants can be plotted in a circular layout thanks to plot library circlize (Gu et al., 2014). Figure **??** is an example generated by this tool. The first input expected by the tool is a combined quantization configuration allocation csv file which should include experiment

outcome for diverse methods such as HQQ and MXQ. The second parameter is the directory where Frobenius norms csv files are located. The third parameter is the perplexity score csv file. The tool produces circos digram in PDF format.

The tool is named `plot-circos-allot.R` and located under the `data-vis` folder in the `lm-quant-toolkit` project. A typical usage is demonstrated in the code snippet as follows:

```
#!/bin/bash

TOOLKIT_DIR="../../.."

MODELS="
Llama-3-7b-hf
Llama-3-13b-hf
Meta-Llama-3-8B
"
BUDGETS="4.25 3.51"

STOP=2
TOPM=2
for MODEL in $MODELS; do
    for BUDGET in $BUDGETS; do
        $TOOLKIT_DIR/data-vis/plot-circos-allot.R \
          --model $MODEL \
          --budget $BUDGET \
          --fnorm_data_dir $TOOLKIT_DIR/src/data/ \
          --ppl_csv_file data/combined.csv \
          --quant_cfg_allot_file data/quant-cfg-allocation.csv \
          --attempt1 sensi-boost-${STOP}-${TOPM} \
          --attempt2 kurt-boost-${STOP}-${TOPM} \
          --attempt3 hqq\
          --attempt4 mxq1
    done
done
```

This code snippet demonstrates how to generate a quant config allocation comparison diagram to examine the nuanced difference between the SensiBoost and kurtBoost approaches, with a stop of 2 and top-m 2, as well as the HQQ and MXQ baselines.

B.2.6 SensiBoost/KurtBoost Win-Tie-Loss Visualization Tool

This tool enables qualitative analysis of effectiveness of the proposed SensiBoost and KurtBoost methods. It is implemented as an R script, which provides a conventional CLI interface

to ease automation. Figure 5.6 is an example generated by this tool. Given a combined perplexity metrics csv file, it produces a series of column digrams in PDF format. The csv file should include experiment outcome for SensiBoost, KurtBoost, the ablation tests or baseline such as HQQ and MXQ. The name experiment, a.k.a. attempt, should follow the pattern `<method>-<stop>-<top m>`.

This tool is included in the `lm-quant-toolkit` under `data-vis` folder. A typical usage is demonstrated in the code snippet as follows:

```
#!/bin/bash

TOOLKIT_DIR="$HOME/work/lm-quant-toolkit"

$TOOLKIT_DIR/data-vis/plot-win-tie-loss.R -f data/combined.csv
```

Appendix C

Supplementary figures and tables

C.1 MXQ Experiment Settings

Table C.1 The table shows the dense bit budget settings for the MXQ experiment, where a total of 259 bit budgets are included, ranging from 2.13 to 8.51. these budgets are presented in 7 groups, *i.e.* 2-bit through 8-bit for clarity.

Group	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
8-bit	8.51	8.25	8.13									
7-bit	7.76	7.74	7.72	7.70	7.68	7.66	7.63	7.62	7.60	7.57	7.56	7.54
	7.53	7.50	7.48	7.47	7.44	7.42	7.41	7.38	7.36	7.34	7.32	7.30
	7.29	7.26	7.24	7.22	7.20	7.19	7.16	7.14	7.13	7.10	7.08	7.06
	7.04	7.02	7.01									
6-bit	6.98	6.96	6.95	6.92	6.89	6.87	6.86	6.83	6.81	6.78	6.77	6.75
	6.72	6.71	6.69	6.68	6.65	6.63	6.61	6.59	6.57	6.55	6.53	6.51
	6.49	6.47	6.45	6.43	6.41	6.39	6.37	6.35	6.33	6.31	6.29	6.27
	6.25	6.23	6.21	6.19	6.17	6.15	6.13	6.11	6.09	6.07	6.06	6.05
	6.04	6.03	6.02	6.00								
5-bit	5.98	5.96	5.94	5.92	5.90	5.88	5.86	5.84	5.82	5.80	5.78	5.76
	5.74	5.72	5.70	5.68	5.66	5.64	5.62	5.60	5.58	5.56	5.54	5.52
	5.50	5.48	5.46	5.44	5.42	5.40	5.38	5.36	5.34	5.32	5.30	5.28
	5.26	5.24	5.22	5.20	5.18	5.16	5.14	5.12	5.10	5.08	5.06	5.04
	5.02	5.00										
4-bit	4.99	4.97	4.95	4.93	4.91	4.89	4.87	4.85	4.83	4.81	4.79	4.77
	4.75	4.73	4.71	4.69	4.67	4.65	4.63	4.61	4.59	4.57	4.55	4.53
	4.51	4.49	4.47	4.45	4.43	4.41	4.39	4.37	4.35	4.33	4.31	4.29
	4.27	4.25	4.23	4.21	4.19	4.17	4.15	4.13	4.11	4.09	4.07	4.05
	4.03	4.01										
3-bit	3.99	3.97	3.95	3.93	3.91	3.89	3.87	3.85	3.83	3.81	3.79	3.77
	3.75	3.73	3.71	3.69	3.67	3.65	3.63	3.61	3.59	3.57	3.55	3.53
	3.51	3.49	3.47	3.45	3.43	3.42	3.41	3.39	3.37	3.35	3.33	3.31
	3.29	3.27	3.25	3.23	3.21	3.19	3.17	3.15	3.13	3.11	3.09	3.07
2-bit	2.57	2.55	2.53	2.51	2.49	2.47	2.45	2.31	2.29	2.27	2.25	2.23
	2.21	2.19	2.17	2.15	2.13							

C.2 Sensitivity Properties

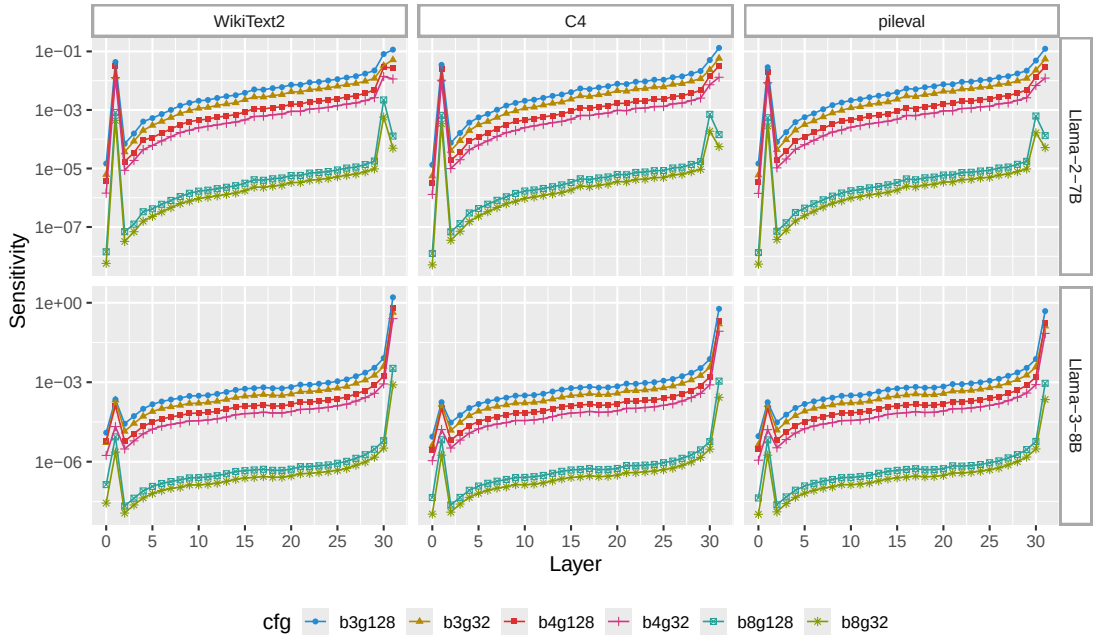


Fig. C.1 This figure illustrates how bit budget influnces layer-wise sensitivity. The magnitude of sensitivity varies among the 3-bit, 4-bit and 8-bit groups. The 4-bit and 8-bit groups show larger difference as indicated by the wider blank. However, the overall patterns of the three bit groups demonstrates close resemblance. For optimal clarity, the figure is best viewed in color and with zoom.

C.3 SensiBoost and KurtBoost Experiment Settings

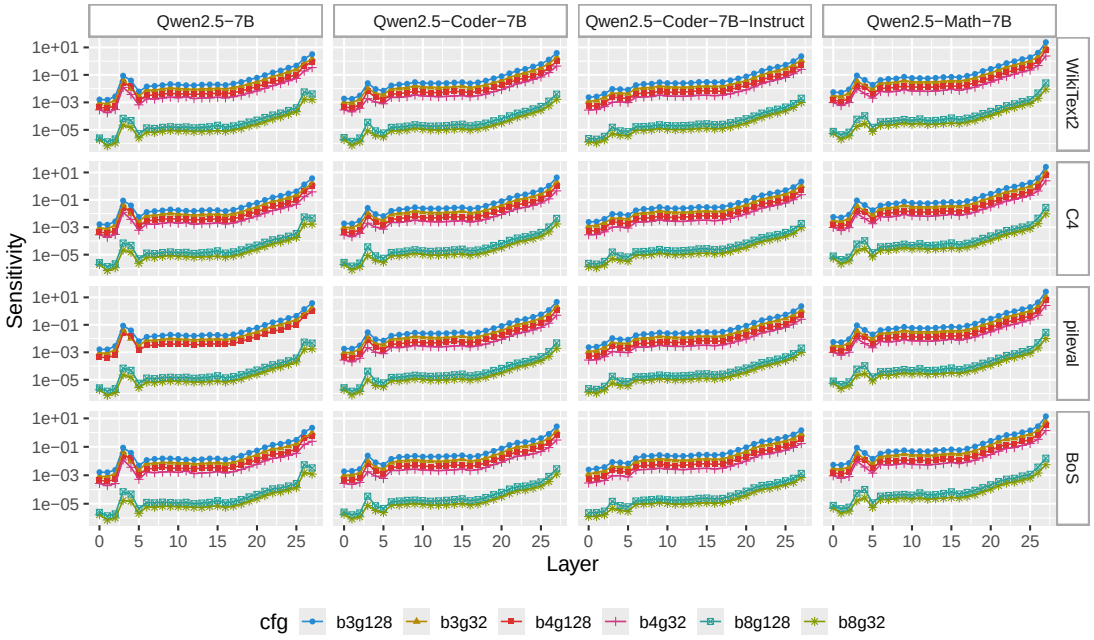


Fig. C.2 This figure presents the sensitivity patterns among the Qwen2.5-7B base model and its fine-tuned mutations. Three fine-tuned models are included for comparison. Except for the *Qwen2.5-Coder-7B-Instruct*, the sensitivity curves exhibit almost identical shapes, which is clear indication of the inheritance of the sensitivity properties btween the base model and its fine-tuned variants. For optimal clarity, the figure is best viewed in color and with zoom.

C.4 SensiBoost and KurtBoost Experiment Results

Table C.2 This table presents the test cases benchmarking the SensiBoost and KurtBoost methods and their ablation tests. Each approach involves two boost stop settings (2 and 3) and four top-m values (1, 2, 3, and 0) across three Llama models under six base bit budget configurations.

			Llama-2-7B		Llama-2-13B		Llama-3-8B	
Method	Config	BPP	Stop	Top m	Stop	Top m	Stop	Top m
SensiBoost			2	1	2	1	2	1
	b4g32	4.51	2	2	2	2	2	2
	b4g64	4.25	2	3	2	3	2	3
	b4g128	4.13	2	0	2	0	2	0
	b3g32	3.51	3	1	3	1	3	1
	b3g64	3.25	3	2	3	2	3	2
	b3g128	3.13	3	3	3	3	3	3
			3	0	3	0	3	0
SensiBoost Ablation			2	1	2	1	2	1
	b4g32	4.51	2	2	2	2	2	2
	b4g64	4.25	2	3	2	3	2	3
	b4g128	4.13	2	0	2	0	2	0
	b3g32	3.51	3	1	3	1	3	1
	b3g64	3.25	3	2	3	2	3	2
	b3g128	3.13	3	3	3	3	3	3
			3	0	3	0	3	0
KurtBoost			2	1	2	1	2	1
	b4g32	4.51	2	2	2	2	2	2
	b4g64	4.25	2	3	2	3	2	3
	b4g128	4.13	2	0	2	0	2	0
	b3g32	3.51	3	1	3	1	3	1
	b3g64	3.25	3	2	3	2	3	2
	b3g128	3.13	3	3	3	3	3	3
			3	0	3	0	3	0
KurtBoost Ablation			2	1	2	1	2	1
	b4g32	4.51	2	2	2	2	2	2
	b4g64	4.25	2	3	2	3	2	3
	b4g128	4.13	2	0	2	0	2	0
	b3g32	3.51	3	1	3	1	3	1
	b3g64	3.25	3	2	3	2	3	2
	b3g128	3.13	3	3	3	3	3	3
			3	0	3	0	3	0

C.5 SensiMiLP and KurtMiLP Experiment Settings

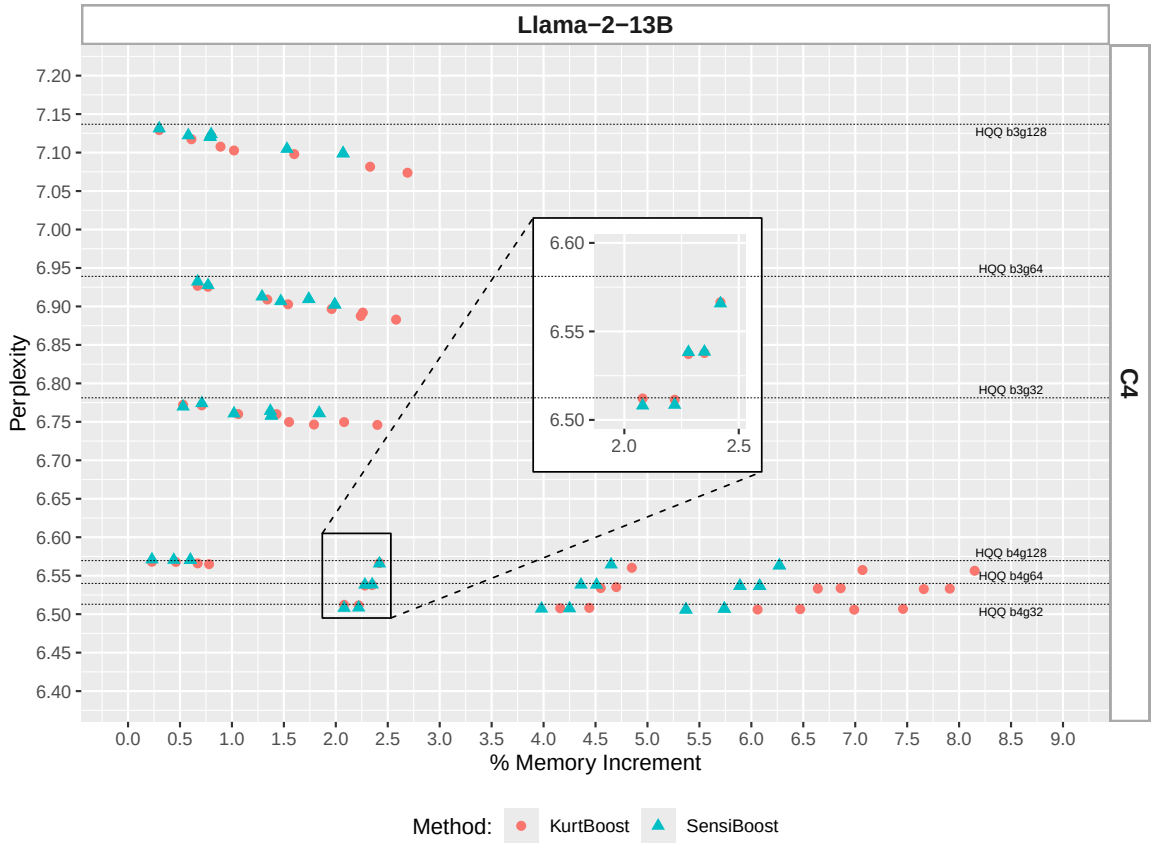


Fig. C.3 This figure illustrates the perplexity performance of the SensiBoost and KurtBoost methods evaluated on the Llama-2-13B model with the C4 dataset. The green triangles, representing the SensiBoost method, are located closer to the y-axis, showing that the SensiBoost approach tends to use less extra memory to achieve comparable performance as the KurtBoost. SensiBoost demonstrates a slight advantage over KurtBoost with approximately 2% additional bugdet to achieve near lowest perplexity score, as highlighted by the maginified sub-plot. For optimal clarity, the figure is best viewed in color and with zoom.

C.6 SensiMiLP and KurtMiLP Experiment Results

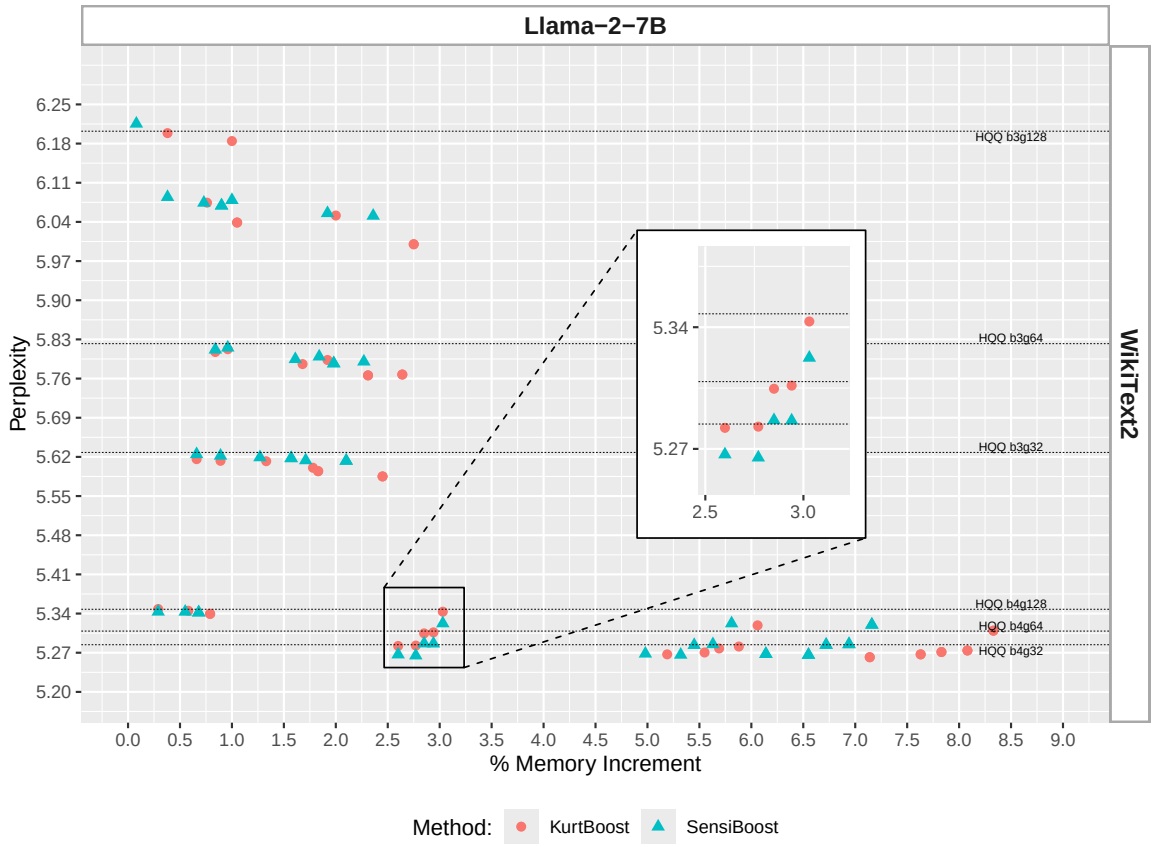


Fig. C.4 This figure illustrates the perplexity performance of the SensiBoost and KurtBoost approaches evaluated on the Llama-2-7B model using the WikiText2 dataset. The green triangles, representing the SensiBoost method, are positioned closer to the y-axis, indicating that SensiBoost requires less additional memory to achieve comparable performance to KurtBoost. Notably, SensiBoost exhibits a slight advantage over KurtBoost, requiring approximately 2% more bit budget to attain a near-minimal perplexity score, as emphasized in the magnified sub-plot. For optimal interpretation, the figure is best viewed in color and with zoom.

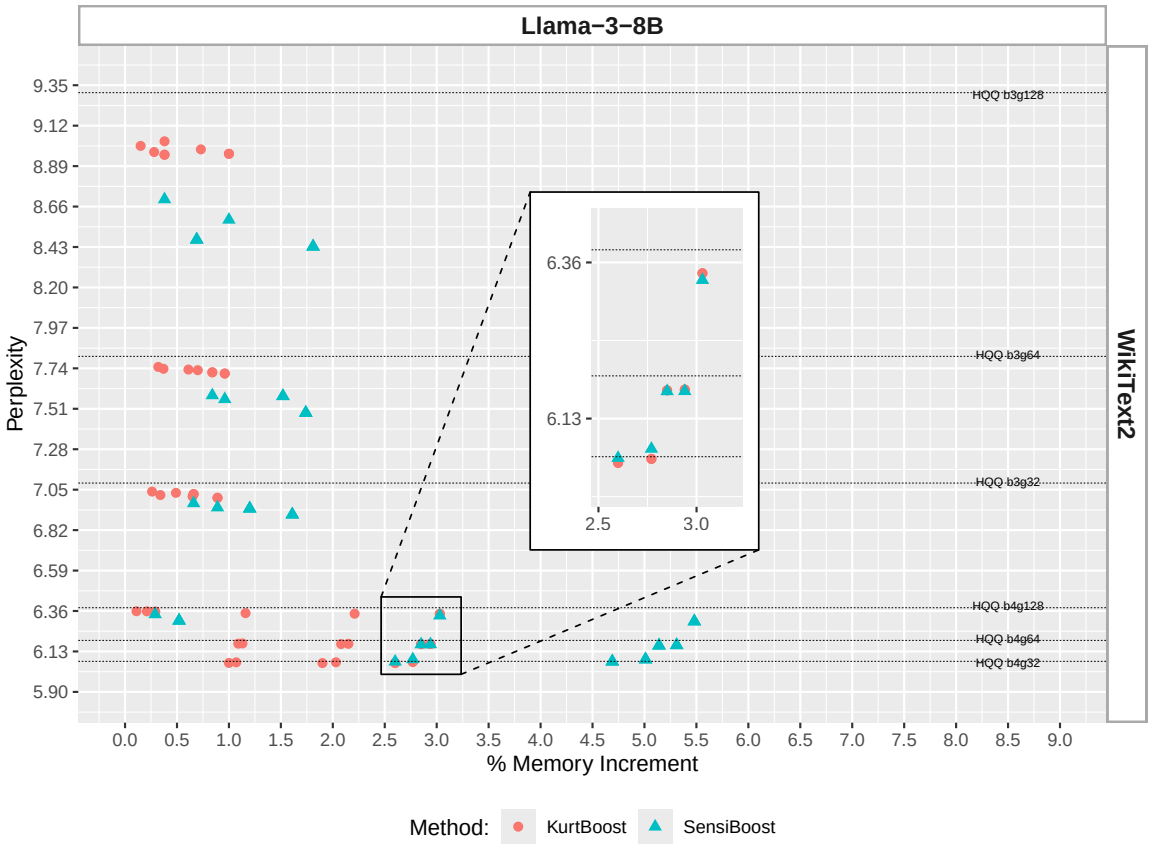


Fig. C.5 This figure illustrates the perplexity performance of the SensiBoost and KurtBoost approaches evaluated on the Llama-2-13B model using the WikiText2 dataset. The green triangles, representing the SensiBoost method, are positioned closer to the y-axis, indicating that SensiBoost requires less additional memory to achieve comparable performance to KurtBoost. Notably, SensiBoost exhibits a slight advantage over KurtBoost, requiring approximately 2% more bit budget to attain a near-minimal perplexity score, as emphasized in the magnified sub-plot. For optimal interpretation, the figure is best viewed in color and with zoom.

Table C.3 Perplexity results of SensiBoost vs KurtBoost (4-bit)

			Llama-2-7B			Llama-2-13B			Llama-3-8B		
Method	Config	BPP	WikiText2	C4	MEM	WikiText2	C4	MEM	WikiText2	C4	MEM
MXQ-SB20	b4g32	4.51	5.27	7.05	4.15	4.69	6.51	7.67	6.07	9.41	5.82
MXQ-KB20		4.51	5.26	7.05	4.18	4.69	6.51	7.78	6.06	9.39	5.74
MXQ-SB21		4.51	5.27	7.05	4.02	4.69	6.51	7.44	6.07	9.40	5.74
MXQ-KB21		4.51	5.28	7.07	4.02	4.69	6.51	7.44	6.06	9.40	5.68
MXQ-SB22		4.51	5.27	7.05	4.11	4.69	6.51	7.58	6.07	9.41	5.82
MXQ-KB22		4.51	5.27	7.05	4.11	4.69	6.51	7.59	6.06	9.40	5.72
MXQ-SB23		4.51	5.27	7.05	4.15	4.69	6.51	7.67	6.07	9.41	5.82
MXQ-KB23		4.51	5.26	7.05	4.18	4.69	6.51	7.72	6.06	9.39	5.74
MXQ-SB30		4.51	5.27	7.05	4.16	4.69	6.51	7.69	6.08	9.42	5.83
MXQ-KB30		4.51	5.27	7.05	4.20	4.69	6.51	7.81	6.07	9.39	5.74
MXQ-SB31		4.51	5.26	7.05	4.03	4.69	6.51	7.45	6.09	9.43	5.75
MXQ-KB31		4.51	5.28	7.07	4.03	4.69	6.51	7.45	6.07	9.40	5.69
MXQ-SB32		4.51	5.27	7.05	4.12	4.69	6.51	7.59	6.08	9.42	5.83
MXQ-KB32		4.51	5.27	7.05	4.13	4.69	6.51	7.60	6.07	9.40	5.72
MXQ-SB33		4.51	5.27	7.05	4.16	4.69	6.51	7.69	6.08	9.42	5.83
MXQ-KB33		4.51	5.27	7.05	4.20	4.69	6.51	7.74	6.07	9.39	5.74
HQQ		4.51	5.28	7.07	3.93	4.69	6.51	7.30	6.07	9.41	5.65
AWQ		4.51	5.26	7.04	3.98	4.69	6.51	7.59	6.07	9.39	5.72
GPTQ		4.51	5.39	7.11	4.74	4.72	6.54	8.43	8.80	10.44	6.71
MXQ-SB20	b4g64	4.25	5.28	7.09	3.96	4.70	6.54	7.37	6.16	9.57	5.59
MXQ-KB20		4.25	5.27	7.08	4.00	4.69	6.53	7.48	6.17	9.57	5.51
MXQ-SB21		4.25	5.29	7.09	3.84	4.70	6.54	7.14	6.17	9.57	5.51
MXQ-KB21		4.25	5.30	7.11	3.84	4.70	6.54	7.14	6.17	9.58	5.45
MXQ-SB22		4.25	5.28	7.09	3.92	4.70	6.54	7.27	6.16	9.57	5.59
MXQ-KB22		4.25	5.28	7.08	3.93	4.70	6.53	7.28	6.17	9.58	5.49
MXQ-SB23		4.25	5.28	7.09	3.96	4.70	6.54	7.37	6.16	9.57	5.59
MXQ-KB23		4.25	5.27	7.08	4.00	4.69	6.53	7.41	6.17	9.57	5.51
MXQ-SB30		4.25	5.29	7.09	3.97	4.70	6.54	7.38	6.17	9.58	5.60
MXQ-KB30		4.25	5.27	7.08	4.01	4.69	6.53	7.49	6.17	9.58	5.52
MXQ-SB31		4.25	5.29	7.09	3.84	4.70	6.54	7.14	6.17	9.58	5.52
MXQ-KB31		4.25	5.31	7.11	3.84	4.70	6.54	7.14	6.18	9.58	5.45
MXQ-SB32		4.25	5.29	7.09	3.93	4.70	6.54	7.28	6.17	9.58	5.60
MXQ-KB32		4.25	5.28	7.09	3.93	4.69	6.54	7.29	6.17	9.58	5.49
MXQ-SB33		4.25	5.29	7.09	3.97	4.70	6.54	7.38	6.17	9.58	5.60
MXQ-KB33		4.25	5.27	7.08	4.01	4.69	6.53	7.43	6.17	9.58	5.52
HQQ		4.25	5.31	7.12	3.74	4.70	6.54	6.99	6.19	9.61	5.41
AWQ		4.25	5.29	7.07	3.74	4.71	6.53	7.10	6.15	9.52	5.46
GPTQ		4.25	5.39	7.13	4.50	4.73	6.56	7.97	11.83	11.77	6.46
BnB		4.25	5.32	7.12	3.60	4.72	6.55	6.71	6.20	9.64	5.31
MXQ-SB20	b4g128	4.13	5.34	7.15	3.67	4.73	6.57	6.77	6.31	9.80	5.33
MXQ-KB20		4.13	5.34	7.14	3.67	4.72	6.57	6.78	6.36	9.91	5.32
MXQ-SB21		4.13	5.34	7.15	3.66	4.73	6.57	6.75	6.34	9.88	5.32
MXQ-KB21		4.13	5.35	7.16	3.66	4.73	6.57	6.75	6.36	9.91	5.31
MXQ-SB22		4.13	5.34	7.15	3.66	4.73	6.57	6.76	6.31	9.80	5.33
MXQ-KB22		4.13	5.34	7.15	3.67	4.73	6.57	6.76	6.36	9.91	5.32
MXQ-SB23		4.13	5.34	7.15	3.67	4.73	6.57	6.77	6.31	9.80	5.33
MXQ-KB23		4.13	5.34	7.14	3.67	4.73	6.57	6.78	6.36	9.91	5.32
MXQ-SB30		4.13	5.32	7.13	3.87	4.73	6.56	7.13	6.30	9.79	5.49
MXQ-KB30		4.13	5.31	7.11	3.91	4.72	6.56	7.25	6.34	9.90	5.41
MXQ-SB31		4.13	5.32	7.13	3.74	4.73	6.57	6.89	6.33	9.86	5.41
MXQ-KB31		4.13	5.34	7.15	3.74	4.73	6.57	6.89	6.35	9.90	5.35
MXQ-SB32		4.13	5.32	7.13	3.83	4.73	6.56	7.03	6.30	9.79	5.49
MXQ-KB32		4.13	5.32	7.13	3.84	4.72	6.56	7.04	6.34	9.90	5.39
MXQ-SB33		4.13	5.32	7.13	3.87	4.73	6.56	7.13	6.30	9.79	5.49
MXQ-KB33		4.13	5.31	7.11	3.91	4.72	6.56	7.18	6.34	9.90	5.41
HQQ		4.13	5.35	7.16	3.65	4.74	6.57	6.74	6.38	9.94	5.31
AWQ		4.13	5.31	7.10	3.62	4.71	6.55	6.92	6.21	9.67	5.33
GPTQ		4.13	5.39	7.18	4.39	4.74	6.57	7.74	97.03	27.77	6.33

Table C.4 Perplexity results of SensiBoost vs KurtBoost (3-bit)

			Llama-2-7B			Llama-2-13B			Llama-3-8B		
Method	Config	BPP	WikiText2	C4	MEM	WikiText2	C4	MEM	WikiText2	C4	MEM
MXQ-SB20	b3g32	3.51	5.62	7.53	3.92	4.87	6.76	7.28	6.94	10.95	5.64
MXQ-KB20		3.51	5.59	7.50	3.92	4.86	6.75	7.27	7.02	11.04	5.64
MXQ-SB21		3.51	5.62	7.54	3.93	4.89	6.77	7.29	6.97	10.97	5.64
MXQ-KB21		3.51	5.62	7.53	3.93	4.89	6.77	7.29	7.04	11.06	5.65
MXQ-SB22		3.51	5.62	7.53	3.92	4.87	6.76	7.29	6.94	10.95	5.64
MXQ-KB22		3.51	5.61	7.51	3.92	4.88	6.76	7.28	7.03	11.04	5.64
MXQ-SB23		3.51	5.62	7.53	3.92	4.87	6.76	7.28	6.94	10.95	-
MXQ-KB23		3.51	5.59	7.50	3.92	4.87	6.75	7.28	7.02	11.04	5.64
MXQ-SB30		3.51	5.61	7.51	3.93	4.88	6.76	7.30	6.91	10.91	5.65
MXQ-KB30		3.51	5.59	7.48	3.93	4.86	6.75	7.30	7.00	11.03	5.65
MXQ-SB31		3.51	5.62	7.53	3.93	4.89	6.77	7.30	6.95	10.96	5.65
MXQ-KB31		3.51	5.61	7.53	3.93	4.88	6.77	7.30	7.02	11.05	5.65
MXQ-SB32		3.51	5.61	7.52	3.93	4.88	6.76	7.30	6.91	10.91	5.65
MXQ-KB32		3.51	5.60	7.50	3.93	4.88	6.76	7.30	7.01	11.03	5.65
MXQ-SB33		3.51	5.61	7.51	3.93	4.88	6.76	7.30	6.91	10.91	5.65
MXQ-KB33		3.51	5.59	7.48	3.93	4.87	6.75	7.30	7.00	11.03	5.65
HQQ		3.51	5.63	7.55	3.93	4.89	6.78	7.30	7.09	11.16	5.65
GPTQ		3.51	5.94	7.81	3.20	5.09	6.96	5.93	17.76	17.98	4.88
MXQ-SB20	b3g64	3.25	5.79	7.77	3.34	4.97	6.91	6.25	7.58	12.06	5.04
MXQ-KB20		3.25	5.77	7.73	3.35	4.96	6.89	6.27	7.72	12.20	5.03
MXQ-SB21		3.25	5.81	7.79	3.33	4.98	6.93	6.23	7.59	12.04	5.03
MXQ-KB21		3.25	5.81	7.80	3.33	4.98	6.93	6.23	7.75	12.24	5.03
MXQ-SB22		3.25	5.79	7.77	3.34	4.97	6.91	6.25	7.58	12.06	5.04
MXQ-KB22		3.25	5.79	7.76	3.34	4.97	6.91	6.25	7.73	12.21	5.03
MXQ-SB23		3.25	5.79	7.77	3.34	4.97	6.91	6.25	7.58	12.06	5.04
MXQ-KB23		3.25	5.77	7.73	3.35	4.96	6.90	6.26	7.72	12.20	5.03
MXQ-SB30		3.25	5.79	7.76	3.35	4.96	6.90	6.27	7.49	11.88	5.05
MXQ-KB30		3.25	5.77	7.72	3.36	4.94	6.88	6.28	7.71	12.19	5.03
MXQ-SB31		3.25	5.82	7.79	3.33	4.97	6.93	6.24	7.56	11.99	5.04
MXQ-KB31		3.25	5.81	7.80	3.33	4.98	6.93	6.24	7.74	12.23	5.03
MXQ-SB32		3.25	5.80	7.76	3.34	4.96	6.91	6.26	7.49	11.88	5.05
MXQ-KB32		3.25	5.79	7.76	3.35	4.96	6.90	6.26	7.73	12.21	5.03
MXQ-SB33		3.25	5.96	8.03	3.28	4.96	6.90	6.27	7.49	11.88	5.05
MXQ-KB33		3.25	5.77	7.72	3.36	4.95	6.89	6.27	7.71	12.19	5.03
HQQ		3.25	5.82	7.81	3.32	4.98	6.94	6.22	7.81	12.36	5.02
GPTQ		3.25	6.13	8.07	2.98	5.14	7.06	5.49	11.16	14.33	4.64
MXQ-SB20	b3g128	3.13	6.07	8.25	3.17	5.15	7.12	5.77	8.47	13.56	4.77
MXQ-KB20		3.13	6.04	8.16	3.18	5.14	7.10	5.80	8.95	14.49	4.74
MXQ-SB21		3.13	6.08	8.25	3.13	5.16	7.13	5.71	8.70	14.00	4.74
MXQ-KB21		3.13	6.20	8.38	3.13	5.15	7.13	5.71	9.00	14.53	4.72
MXQ-SB22		3.13	6.07	8.25	3.15	5.15	7.12	5.75	8.47	13.56	4.77
MXQ-KB22		3.13	6.07	8.24	3.16	5.15	7.12	5.75	8.97	14.52	4.73
MXQ-SB23		3.13	6.07	8.25	3.17	5.15	7.12	5.77	8.47	13.56	4.77
MXQ-KB23		3.13	6.04	8.16	3.18	5.14	7.11	5.79	8.95	14.49	4.74
MXQ-SB30		3.13	6.05	8.22	3.15	5.14	7.10	5.74	8.43	13.50	4.75
MXQ-KB30		3.13	6.00	8.11	3.15	5.13	7.07	5.76	8.96	14.42	4.73
MXQ-SB31		3.13	6.08	8.24	3.12	5.16	7.12	5.69	8.59	13.77	4.73
MXQ-KB31		3.13	6.18	8.38	3.12	5.15	7.12	5.69	9.03	14.50	4.72
MXQ-SB32		3.13	6.06	8.22	3.14	5.14	7.10	5.72	8.43	13.50	4.75
MXQ-KB32		3.13	6.05	8.20	3.14	5.14	7.10	5.72	8.99	14.47	4.73
MXQ-SB33		3.13	6.22	8.35	3.15	5.14	7.10	5.74	8.43	13.50	4.75
MXQ-KB33		3.13	6.00	8.11	3.15	5.13	7.08	5.75	8.96	14.42	4.73
HQQ		3.13	6.20	8.40	3.11	5.15	7.14	5.67	9.31	14.89	4.71
GPTQ		3.13	6.32	8.30	2.87	5.24	7.19	5.27	52.78	30.04	4.52

Table C.5 Perplexity results of SensiBoost vs Ablation (4-bit)

			Llama-2-7B			Llama-2-13B			Llama-3-8B		
Method	Config	BPP	WikiText2	C4	MEM	WikiText2	C4	MEM	WikiText2	C4	MEM
MXQ-SB20	b4g32	4.51	5.27	7.05	4.15	4.69	6.51	7.67	6.07	9.41	5.82
MXQ-SBAB20		4.51	5.28	7.06	4.15	4.69	6.51	7.67	6.07	9.40	5.82
MXQ-SB21		4.51	5.27	7.05	4.02	4.69	6.51	7.44	6.07	9.40	5.74
MXQ-SBAB21		4.51	5.29	7.07	4.02	4.69	6.51	7.44	6.07	9.41	5.74
MXQ-SB22		4.51	5.27	7.05	4.11	4.69	6.51	7.58	6.07	9.41	5.82
MXQ-SBAB22		4.51	5.28	7.07	4.11	4.69	6.51	7.57	6.07	9.40	5.82
MXQ-SB23		4.51	5.27	7.05	4.15	4.69	6.51	7.67	6.07	9.41	5.82
MXQ-SBAB23		4.51	5.28	7.06	4.15	4.69	6.51	7.67	6.07	9.41	5.82
MXQ-SB30		4.51	5.27	7.05	4.16	4.69	6.51	7.69	6.08	9.42	5.83
MXQ-SBAB30		4.51	5.28	7.07	4.16	4.69	6.51	7.69	6.07	9.40	5.83
MXQ-SB31		4.51	5.26	7.05	4.03	4.69	6.51	7.45	6.09	9.43	5.75
MXQ-SBAB31		4.51	5.28	7.07	4.03	4.69	6.51	7.45	6.08	9.41	5.75
MXQ-SB32		4.51	5.27	7.05	4.12	4.69	6.51	7.59	6.08	9.42	5.83
MXQ-SBAB32		4.51	5.28	7.07	4.12	4.69	6.51	7.59	6.07	9.41	5.83
MXQ-SB33		4.51	5.27	7.05	4.16	4.69	6.51	7.69	6.08	9.42	5.83
MXQ-SBAB33		4.51	5.28	7.07	4.16	4.69	6.51	7.69	6.07	9.41	5.83
MXQ-SB20	b4g64	4.25	5.28	7.09	3.96	4.70	6.54	7.37	6.16	9.57	5.59
MXQ-SBAB20		4.25	5.30	7.11	3.96	4.70	6.53	7.37	6.17	9.58	5.59
MXQ-SB21		4.25	5.29	7.09	3.84	4.70	6.54	7.14	6.17	9.57	5.51
MXQ-SBAB21		4.25	5.30	7.11	3.84	4.70	6.54	7.14	6.19	9.59	5.51
MXQ-SB22		4.25	5.28	7.09	3.92	4.70	6.54	7.27	6.16	9.57	5.59
MXQ-SBAB22		4.25	5.30	7.11	3.92	4.70	6.54	7.27	6.19	9.59	5.59
MXQ-SB23		4.25	5.28	7.09	3.96	4.70	6.54	7.37	6.16	9.57	5.59
MXQ-SBAB23		4.25	5.30	7.11	3.96	4.69	6.53	7.37	6.18	9.59	5.59
MXQ-SB30		4.25	5.29	7.09	3.97	4.70	6.54	7.38	6.17	9.58	5.60
MXQ-SBAB30		4.25	5.30	7.11	3.97	4.70	6.54	7.38	6.18	9.58	5.60
MXQ-SB31		4.25	5.29	7.09	3.84	4.70	6.54	7.14	6.17	9.58	5.52
MXQ-SBAB31		4.25	5.31	7.12	3.84	4.70	6.54	7.14	6.19	9.60	5.52
MXQ-SB32		4.25	5.29	7.09	3.93	4.70	6.54	7.28	6.17	9.58	5.60
MXQ-SBAB32		4.25	5.31	7.11	3.93	4.70	6.54	7.28	6.19	9.60	5.60
MXQ-SB33		4.25	5.29	7.09	3.97	4.70	6.54	7.38	6.17	9.58	5.60
MXQ-SBAB33		4.25	5.30	7.11	3.97	4.70	6.53	7.38	6.18	9.59	5.60
MXQ-SB20	b4g128	4.13	5.34	7.15	3.67	4.73	6.57	6.77	6.31	9.80	5.33
MXQ-SBAB20		4.13	5.34	7.16	3.67	4.73	6.57	6.77	6.37	9.92	5.33
MXQ-SB21		4.13	5.34	7.15	3.66	4.73	6.57	6.75	6.34	9.88	5.32
MXQ-SBAB21		4.13	5.35	7.16	3.66	4.74	6.57	6.75	6.38	9.92	5.32
MXQ-SB22		4.13	5.34	7.15	3.66	4.73	6.57	6.76	6.31	9.80	5.33
MXQ-SBAB22		4.13	5.34	7.15	3.66	4.74	6.57	6.76	6.37	9.91	5.33
MXQ-SB23		4.13	5.34	7.15	3.67	4.73	6.57	6.77	6.31	9.80	5.33
MXQ-SBAB23		4.13	5.34	7.16	3.67	4.74	6.57	6.77	6.36	9.90	5.33
MXQ-SB30		4.13	5.32	7.13	3.87	4.73	6.56	7.13	6.30	9.79	5.49
MXQ-SBAB30		4.13	5.34	7.15	3.87	4.73	6.56	7.13	6.36	9.91	5.49
MXQ-SB31		4.13	5.32	7.13	3.74	4.73	6.57	6.89	6.33	9.86	5.41
MXQ-SBAB31		4.13	5.35	7.16	3.74	4.74	6.57	6.89	6.38	9.93	5.41
MXQ-SB32		4.13	5.32	7.13	3.83	4.73	6.56	7.03	6.30	9.79	5.49
MXQ-SBAB32		4.13	5.34	7.15	3.83	4.73	6.56	7.03	6.36	9.91	5.49
MXQ-SB33		4.13	5.32	7.13	3.87	4.73	6.56	7.13	6.30	9.79	5.49
MXQ-SBAB33		4.13	5.34	7.15	3.87	4.73	6.56	7.13	6.34	9.88	5.49

Table C.6 Perplexity results of SensiBoost vs Ablation (3-bit)

			Llama-2-7B			Llama-2-13B			Llama-3-8B		
Method	Config	BPP	WikiText2	C4	MEM	WikiText2	C4	MEM	WikiText2	C4	MEM
MXQ-SB20	b3g32	3.51	5.62	7.53	3.92	4.87	6.76	7.28	6.94	10.95	5.64
MXQ-SBAB20		3.51	5.59	7.51	3.92	4.88	6.76	7.28	7.05	11.10	5.64
MXQ-SB21		3.51	5.62	7.54	3.93	4.89	6.77	7.29	6.97	10.97	5.64
MXQ-SBAB21		3.51	5.61	7.53	3.93	4.88	6.78	-	7.06	11.10	5.64
MXQ-SB22		3.51	5.62	7.53	3.92	4.87	6.76	7.29	6.94	10.95	5.64
MXQ-SBAB22		3.51	5.61	7.53	3.92	4.88	6.77	7.28	7.05	11.08	5.63
MXQ-SB23		3.51	5.62	7.53	3.92	4.87	6.76	7.28	6.94	10.95	-
MXQ-SBAB23		3.51	5.61	7.51	3.92	4.88	6.76	7.28	7.04	11.08	5.64
MXQ-SB30		3.51	5.61	7.51	3.93	4.88	6.76	7.30	6.91	10.91	5.65
MXQ-SBAB30		3.51	5.60	7.52	3.93	4.88	6.76	7.30	7.05	11.07	5.65
MXQ-SB31		3.51	5.62	7.53	3.93	4.89	6.77	7.30	6.95	10.96	5.65
MXQ-SBAB31		3.51	5.62	7.53	3.93	4.88	6.77	7.30	7.05	11.07	5.65
MXQ-SB32		3.51	5.61	7.52	3.93	4.88	6.76	7.30	6.91	10.91	5.65
MXQ-SBAB32		3.51	5.61	7.52	3.93	4.88	6.77	7.30	7.04	11.05	5.65
MXQ-SB33		3.51	5.61	7.51	3.93	4.88	6.76	7.30	6.91	10.91	5.65
MXQ-SBAB33		3.51	5.61	7.52	3.93	4.87	6.76	7.30	7.05	11.05	5.65
MXQ-SB20	b3g64	3.25	5.79	7.77	3.34	4.97	6.91	6.25	7.58	12.06	5.04
MXQ-SBAB20		3.25	5.78	7.75	3.34	4.97	6.91	6.25	7.74	12.26	5.04
MXQ-SB21		3.25	5.81	7.79	3.33	4.98	6.93	6.23	7.59	12.04	5.03
MXQ-SBAB21		3.25	5.80	7.78	3.33	4.98	6.93	6.23	7.77	12.28	5.03
MXQ-SB22		3.25	5.79	7.77	3.34	4.97	6.91	6.25	7.58	12.06	5.04
MXQ-SBAB22		3.25	5.78	7.77	3.34	4.98	6.92	6.25	7.73	12.22	5.04
MXQ-SB23		3.25	5.79	7.77	3.34	4.97	6.91	6.25	7.58	12.06	5.04
MXQ-SBAB23		3.25	5.80	7.76	3.34	4.96	6.91	6.25	7.73	12.23	-
MXQ-SB30		3.25	5.79	7.76	3.35	4.96	6.90	6.27	7.49	11.88	5.05
MXQ-SBAB30		3.25	5.78	7.75	3.35	4.97	6.91	6.27	7.70	12.20	5.04
MXQ-SB31		3.25	5.82	7.79	3.33	4.97	6.93	6.24	7.56	11.99	5.04
MXQ-SBAB31		3.25	5.81	7.78	3.33	4.98	6.93	6.24	7.77	12.31	5.03
MXQ-SB32		3.25	5.80	7.76	3.34	4.96	6.91	6.26	7.49	11.88	5.05
MXQ-SBAB32		3.25	5.79	7.76	3.34	4.97	6.92	6.26	7.72	12.21	5.04
MXQ-SB33		3.25	5.96	8.03	3.28	4.96	6.90	6.27	7.49	11.88	5.05
MXQ-SBAB33		3.25	5.78	7.75	3.35	4.96	6.91	6.27	7.72	12.22	5.04
MXQ-SB20	b3g128	3.13	6.07	8.25	3.17	5.15	7.12	5.77	8.47	13.56	4.77
MXQ-SBAB20		3.13	6.16	8.35	3.17	5.13	7.12	5.77	9.18	14.69	4.76
MXQ-SB21		3.13	6.08	8.25	3.13	5.16	7.13	5.71	8.70	14.00	4.74
MXQ-SBAB21		3.13	6.20	8.37	3.13	5.15	7.13	5.71	9.21	14.77	4.74
MXQ-SB22		3.13	6.07	8.25	3.15	5.15	7.12	5.75	8.47	13.56	4.77
MXQ-SBAB22		3.13	6.15	8.35	3.15	5.14	7.12	5.75	9.18	14.65	4.76
MXQ-SB23		3.13	6.07	8.25	3.17	5.15	7.12	5.77	8.47	13.56	4.77
MXQ-SBAB23		3.13	6.16	8.33	3.17	5.13	7.11	5.77	9.23	14.78	4.76
MXQ-SB30		3.13	6.05	8.22	3.15	5.14	7.10	5.74	8.43	13.50	4.75
MXQ-SBAB30		3.13	6.13	8.29	3.14	5.12	7.10	5.74	9.16	14.65	4.75
MXQ-SB31		3.13	6.08	8.24	3.12	5.16	7.12	5.69	8.59	13.77	4.73
MXQ-SBAB31		3.13	6.18	8.37	3.12	5.14	7.12	5.69	9.24	14.74	4.73
MXQ-SB32		3.13	6.06	8.22	3.14	5.14	7.10	5.72	8.43	13.50	4.75
MXQ-SBAB32		3.13	6.14	8.31	3.14	5.13	7.11	5.72	9.16	14.62	4.75
MXQ-SB33		3.13	6.22	8.35	3.15	5.14	7.10	5.74	8.43	13.50	4.75
MXQ-SBAB33		3.13	6.14	8.32	3.14	5.12	7.09	5.74	9.16	14.68	4.75

Table C.7 Perplexity results of KurtBoost vs Ablation (4-bit)

			Llama-2-7B			Llama-2-13B			Llama-3-8B		
Method	Config	BPP	WikiText2	C4	MEM	WikiText2	C4	MEM	WikiText2	C4	MEM
MXQ-KB20	b4g32	4.51	5.26	7.05	4.18	4.69	6.51	7.78	6.06	9.39	5.74
MXQ-KBAB20		4.51	5.28	7.06	4.18	4.69	6.51	7.78	6.07	9.41	5.74
MXQ-KB21		4.51	5.28	7.07	4.02	4.69	6.51	7.44	6.06	9.40	5.68
MXQ-KBAB21		4.51	5.28	7.07	4.02	4.69	6.51	7.44	6.07	9.41	5.68
MXQ-KB22		4.51	5.27	7.05	4.11	4.69	6.51	7.59	6.06	9.40	5.72
MXQ-KBAB22		4.51	5.28	7.07	4.11	4.69	6.51	7.59	6.07	9.41	5.72
MXQ-KB23		4.51	5.26	7.05	4.18	4.69	6.51	7.72	6.06	9.39	5.74
MXQ-KBAB23		4.51	5.28	7.06	4.18	4.69	6.51	7.72	6.07	9.41	5.74
MXQ-KB30		4.51	5.27	7.05	4.20	4.69	6.51	7.81	6.07	9.39	5.74
MXQ-KBAB30		4.51	5.28	7.07	4.20	4.69	6.51	7.81	6.07	9.41	5.75
MXQ-KB31		4.51	5.28	7.07	4.03	4.69	6.51	7.45	6.07	9.40	5.69
MXQ-KBAB31		4.51	5.28	7.07	4.03	4.69	6.51	7.45	6.07	9.41	5.69
MXQ-KB32		4.51	5.27	7.05	4.13	4.69	6.51	7.60	6.07	9.40	5.72
MXQ-KBAB32		4.51	5.28	7.07	4.13	4.69	6.51	7.60	6.07	9.41	5.72
MXQ-KB33		4.51	5.27	7.05	4.20	4.69	6.51	7.74	6.07	9.39	5.74
MXQ-KBAB33		4.51	5.28	7.07	4.20	4.69	6.51	7.74	6.07	9.41	5.75
MXQ-KB20	b4g64	4.25	5.27	7.08	4.00	4.69	6.53	7.48	6.17	9.57	5.51
MXQ-KBAB20		4.25	5.30	7.11	4.00	4.69	6.53	7.48	6.19	9.59	5.51
MXQ-KB21		4.25	5.30	7.11	3.84	4.70	6.54	7.14	6.17	9.58	5.45
MXQ-KBAB21		4.25	5.31	7.11	3.84	4.70	6.54	7.14	6.19	9.60	5.45
MXQ-KB22		4.25	5.28	7.08	3.93	4.70	6.53	7.28	6.17	9.58	5.49
MXQ-KBAB22		4.25	5.30	7.11	3.93	4.70	6.54	7.28	6.19	9.60	5.49
MXQ-KB23		4.25	5.27	7.08	4.00	4.69	6.53	7.41	6.17	9.57	5.51
MXQ-KBAB23		4.25	5.30	7.11	4.00	4.69	6.53	7.42	6.19	9.59	5.51
MXQ-KB30		4.25	5.27	7.08	4.01	4.69	6.53	7.49	6.17	9.58	5.52
MXQ-KBAB30		4.25	5.30	7.11	4.01	4.69	6.53	7.49	6.19	9.60	5.52
MXQ-KB31		4.25	5.31	7.11	3.84	4.70	6.54	7.14	6.18	9.58	5.45
MXQ-KBAB31		4.25	5.31	7.11	3.84	4.70	6.54	7.14	6.19	9.60	5.45
MXQ-KB32		4.25	5.28	7.09	3.93	4.69	6.54	7.29	6.17	9.58	5.49
MXQ-KBAB32		4.25	5.31	7.11	3.93	4.70	6.54	7.29	6.19	9.60	5.49
MXQ-KB33		4.25	5.27	7.08	4.01	4.69	6.53	7.43	6.17	9.58	5.52
MXQ-KBAB33		4.25	5.30	7.11	4.01	4.70	6.53	7.43	6.19	9.60	5.52
MXQ-KB20	b4g128	4.13	5.34	7.14	3.67	4.72	6.57	6.78	6.36	9.91	5.32
MXQ-KBAB20		4.13	5.34	7.15	3.67	4.74	6.56	6.78	6.37	9.93	5.32
MXQ-KB21		4.13	5.35	7.16	3.66	4.73	6.57	6.75	6.36	9.91	5.31
MXQ-KBAB21		4.13	5.34	7.16	3.66	4.74	6.57	6.75	6.38	9.93	5.31
MXQ-KB22		4.13	5.34	7.15	3.67	4.73	6.57	6.76	6.36	9.91	5.32
MXQ-KBAB22		4.13	5.34	7.16	3.67	4.73	6.57	6.76	6.37	9.93	5.32
MXQ-KB23		4.13	5.34	7.14	3.67	4.73	6.57	6.78	6.36	9.91	5.32
MXQ-KBAB23		4.13	5.34	7.16	3.67	4.73	6.56	6.78	6.37	9.92	5.32
MXQ-KB30		4.13	5.31	7.11	3.91	4.72	6.56	7.25	6.34	9.90	5.41
MXQ-KBAB30		4.13	5.33	7.15	3.91	4.73	6.56	7.25	6.37	9.92	5.41
MXQ-KB31		4.13	5.34	7.15	3.74	4.73	6.57	6.89	6.35	9.90	5.35
MXQ-KBAB31		4.13	5.34	7.16	3.74	4.73	6.57	6.89	6.38	9.93	5.35
MXQ-KB32		4.13	5.32	7.13	3.84	4.72	6.56	7.04	6.34	9.90	5.39
MXQ-KBAB32		4.13	5.34	7.15	3.84	4.73	6.56	7.04	6.38	9.92	5.39
MXQ-KB33		4.13	5.31	7.11	3.91	4.72	6.56	7.18	6.34	9.90	5.41
MXQ-KBAB33		4.13	5.34	7.15	3.91	4.73	6.56	7.18	6.37	9.91	5.41

Table C.8 Perplexity results of KurtBoost vs Ablation (3-bit)

			Llama-2-7B			Llama-2-13B			Llama-3-8B		
Method	Config	BPP	WikiText2	C4	MEM	WikiText2	C4	MEM	WikiText2	C4	MEM
MXQ-KB20	b3g32	3.51	5.59	7.50	3.92	4.86	6.75	7.27	7.02	11.04	5.64
MXQ-KBAB20		3.51	5.60	7.51	3.92	4.88	6.76	7.27	7.06	11.11	5.64
MXQ-KB21		3.51	5.62	7.53	3.93	4.89	6.77	7.29	7.04	11.06	5.65
MXQ-KBAB21		3.51	5.62	7.53	3.93	4.88	6.77	7.29	7.08	11.14	5.65
MXQ-KB22		3.51	5.61	7.51	3.92	4.88	6.76	7.28	7.03	11.04	5.64
MXQ-KBAB22		3.51	5.60	7.52	3.92	4.88	6.77	7.28	7.07	11.13	5.64
MXQ-KB23		3.51	5.59	7.50	3.92	4.87	6.75	7.28	7.02	11.04	5.64
MXQ-KBAB23		3.51	5.60	7.50	3.92	4.87	6.76	7.27	7.06	11.10	5.64
MXQ-KB30		3.51	5.59	7.48	3.93	4.86	6.75	7.30	7.00	11.03	5.65
MXQ-KBAB30		3.51	5.60	7.51	3.93	4.88	6.76	7.30	7.05	11.09	5.65
MXQ-KB31		3.51	5.61	7.53	3.93	4.88	6.77	7.30	7.02	11.05	5.65
MXQ-KBAB31		3.51	5.61	7.53	3.93	4.88	6.78	7.30	7.07	11.14	5.65
MXQ-KB32		3.51	5.60	7.50	3.93	4.88	6.76	7.30	7.01	11.03	5.65
MXQ-KBAB32		3.51	5.60	7.50	3.93	4.88	6.77	-	7.07	11.13	5.65
MXQ-KB33		3.51	5.59	7.48	3.93	4.87	6.75	7.30	7.00	11.03	5.65
MXQ-KBAB33		3.51	5.59	7.51	3.93	4.87	6.76	7.30	7.06	11.11	5.65
MXQ-KB20	b3g64	3.25	5.77	7.73	3.35	4.96	6.89	6.27	7.72	12.20	5.03
MXQ-KBAB20		3.25	5.79	7.74	3.35	4.96	6.90	6.27	7.78	12.28	5.03
MXQ-KB21		3.25	5.81	7.80	3.33	4.98	6.93	6.23	7.75	12.24	5.03
MXQ-KBAB21		3.25	5.81	7.79	3.33	4.98	6.93	6.23	7.79	12.31	5.03
MXQ-KB22		3.25	5.79	7.76	3.34	4.97	6.91	6.25	7.73	12.21	5.03
MXQ-KBAB22		3.25	5.78	7.77	3.34	4.97	6.92	6.25	7.77	12.30	5.03
MXQ-KB23		3.25	5.77	7.73	3.35	4.96	6.90	6.26	7.72	12.20	5.03
MXQ-KBAB23		3.25	5.79	7.75	3.35	4.97	6.91	6.26	7.75	12.29	5.03
MXQ-KB30		3.25	5.77	7.72	3.36	4.94	6.88	6.28	7.71	12.19	5.03
MXQ-KBAB30		3.25	5.78	7.75	3.36	4.96	6.90	6.28	7.75	12.32	5.03
MXQ-KB31		3.25	5.81	7.80	3.33	4.98	6.93	6.24	7.74	12.23	5.03
MXQ-KBAB31		3.25	5.80	7.78	3.33	4.98	6.93	6.24	7.77	12.31	5.03
MXQ-KB32		3.25	5.79	7.76	3.35	4.96	6.90	6.26	7.73	12.21	5.03
MXQ-KBAB32		3.25	5.79	7.76	3.35	4.97	6.91	6.26	7.78	12.31	5.03
MXQ-KB33		3.25	5.77	7.72	3.36	4.95	6.89	6.27	7.71	12.19	5.03
MXQ-KBAB33		3.25	5.78	7.74	3.36	4.97	6.91	6.28	7.74	12.27	5.03
MXQ-KB20	b3g128	3.13	6.04	8.16	3.18	5.14	7.10	5.80	8.95	14.49	4.74
MXQ-KBAB20		3.13	6.18	8.35	3.18	5.13	7.10	5.80	9.23	14.80	4.74
MXQ-KB21		3.13	6.20	8.38	3.13	5.15	7.13	5.71	9.00	14.53	4.72
MXQ-KBAB21		3.13	6.18	8.37	3.13	5.14	7.12	5.71	9.27	14.85	4.73
MXQ-KB22		3.13	6.07	8.24	3.16	5.15	7.12	5.75	8.97	14.52	4.73
MXQ-KBAB22		3.13	6.17	8.31	3.16	5.14	7.12	5.75	9.28	14.89	4.73
MXQ-KB23		3.13	6.04	8.16	3.18	5.14	7.11	5.79	8.95	14.49	4.74
MXQ-KBAB23		3.13	6.17	8.34	3.18	5.13	7.11	5.79	9.27	14.83	4.74
MXQ-KB30		3.13	6.00	8.11	3.15	5.13	7.07	5.76	8.96	14.42	4.73
MXQ-KBAB30		3.13	6.14	8.31	3.15	5.11	7.08	5.76	9.23	14.77	4.73
MXQ-KB31		3.13	6.18	8.38	3.12	5.15	7.12	5.69	9.03	14.50	4.72
MXQ-KBAB31		3.13	6.18	8.36	3.12	5.14	7.12	5.69	9.25	14.80	4.72
MXQ-KB32		3.13	6.05	8.20	3.14	5.14	7.10	5.72	8.99	14.47	4.73
MXQ-KBAB32		3.13	6.15	8.33	3.14	5.12	7.11	-	9.23	14.78	4.73
MXQ-KB33		3.13	6.00	8.11	3.15	5.13	7.08	5.75	8.96	14.42	4.73
MXQ-KBAB33		3.13	6.13	8.30	3.15	5.11	7.08	5.75	9.24	14.72	4.73

Table C.9 This table presents the test configurations benchmarking the SensiMiLP and KurtMiLP methods. Each approach involves three top-m values (1, 2, 3) across three Llama models under 21 bit budget configurations. In the ablation tests, all layers are assumed to contribute to quantization accuracy equally, therefore top-*m* is no longer applicable.

		Llama-2-7B	Llama-2-13B	Llama-3-8B
Method	BPP	Top m	Top m	Top m
SensiMiLP	6.89 5.72 5.02 4.51 4.25 4.21 4.17	1	1	1
	4.13 4.11 4.07 3.95 3.87 3.83 3.65	2	2	2
	3.51 3.25 3.19 3.15 3.13 3.11 3.07	3	3	3
SensiMiLP Ablation	6.89 5.72 5.02 4.51 4.25 4.21 4.17	-	-	-
	4.13 4.11 4.07 3.95 3.87 3.83 3.65	-	-	-
	3.51 3.25 3.19 3.15 3.13 3.11 3.07	-	-	-
KurtMiLP	6.89 5.72 5.02 4.51 4.25 4.21 4.17	1	1	1
	4.13 4.11 4.07 3.95 3.87 3.83 3.65	2	2	2
	3.51 3.25 3.19 3.15 3.13 3.11 3.07	3	3	3
KurtMiLP Ablation	6.89 5.72 5.02 4.51 4.25 4.21 4.17	-	-	-
	4.13 4.11 4.07 3.95 3.87 3.83 3.65	-	-	-
	3.51 3.25 3.19 3.15 3.13 3.11 3.07	-	-	-

Table C.10 Assessment matrix of SensiMiLP/KurtMiLP approaches

Method	SM	KM	SMAB	KMAB	HQQ	MXQ
SM[1]	-	X	X	-	X	X
KM[2]	-	-	-	X	X	X
SMAB[3]	-	-	-	-	-	-
KMAB[4]	-	-	-	-	-	-
HQQ	-	-	-	-	-	-
MXQ	-	-	-	-	-	-

[1] SM denotes the SensiMiLP method.

[2] KM denotes the KurtMiLP method.

[3] SMAB denotes the ablation test for SensiMiLP method.

[4] KMAB denotes the ablation test for KurtMiLP method.

Table C.11 Perplexity results of SensiMiLP vs KurtMiLP (4-bit)

			Llama-2-7B			Llama-2-13B			Llama-3-8B		
Method	Config	BPP	WikiText2	C4	MEM	WikiText2	C4	MEM	WikiText2	C4	MEM
MXQ-SM1	b4g32	4.51	5.28	7.07	3.94	4.69	6.52	7.31	6.13	9.49	5.64
MXQ-KM1		4.51	5.30	7.08	3.94	4.69	6.53	7.31	6.10	9.45	5.65
MXQ-SM2		4.51	5.28	7.08	3.94	4.69	6.53	7.35	6.14	9.52	5.62
MXQ-KM2		4.51	5.28	7.08	3.94	4.69	6.52	7.37	6.10	9.45	5.64
MXQ-SM3		4.51	5.29	7.09	3.94	4.70	6.53	7.37	6.14	9.52	5.62
MXQ-KM3		4.51	5.28	7.08	3.94	4.69	6.53	7.37	6.12	9.48	5.64
HQQ		4.51	5.28	7.07	3.93	4.69	6.51	7.30	6.07	9.41	5.65
AWQ		4.51	5.26	7.04	3.98	4.69	6.51	7.59	6.07	9.39	5.72
GPTQ		4.51	5.39	7.11	4.74	4.72	6.54	8.43	8.80	10.44	6.71
MXQ		4.51	5.29	7.08	3.89	4.69	6.51	7.28	6.11	9.47	5.62
MXQ-SM1	b4g64	4.25	5.31	7.11	3.74	4.70	6.55	6.99	6.19	9.61	5.41
MXQ-KM1		4.25	5.31	7.12	3.74	4.70	6.54	6.99	6.21	9.62	5.41
MXQ-SM2		4.25	5.31	7.12	3.74	4.71	6.55	6.99	6.22	9.65	5.41
MXQ-KM2		4.25	5.31	7.12	3.74	4.70	6.54	6.99	6.20	9.60	5.41
MXQ-SM3		4.25	5.31	7.12	3.74	4.70	6.55	6.99	6.22	9.65	5.41
MXQ-KM3		4.25	5.31	7.11	3.74	4.70	6.55	6.98	6.20	9.61	5.41
HQQ		4.25	5.31	7.12	3.74	4.70	6.54	6.99	6.19	9.61	5.41
AWQ		4.25	5.29	7.07	3.74	4.71	6.53	7.10	6.15	9.52	5.46
GPTQ		4.25	5.39	7.13	4.50	4.73	6.56	7.97	11.83	11.77	6.46
BnB		4.25	5.32	7.12	3.60	4.72	6.55	6.71	6.20	9.64	5.31
MXQ		4.25	5.31	7.13	3.71	4.71	6.54	6.90	6.29	9.76	5.49
MXQ-SM1	b4g128	4.13	5.35	7.16	3.65	4.73	6.57	6.74	6.36	9.91	5.31
MXQ-KM1		4.13	5.35	7.16	3.65	4.74	6.57	6.74	6.37	9.91	5.31
MXQ-SM2		4.13	5.35	7.16	3.65	4.73	6.57	6.74	6.37	9.92	5.31
MXQ-KM2		4.13	5.35	7.16	3.65	4.73	6.57	6.74	6.37	9.93	5.31
MXQ-SM3		4.13	5.35	7.16	-	4.73	6.57	6.74	6.37	9.92	5.31
MXQ-KM3		4.13	5.35	7.16	3.65	4.73	6.57	6.74	6.37	9.93	5.31
HQQ		4.13	5.35	7.16	3.65	4.74	6.57	6.74	6.38	9.94	5.31
AWQ		4.13	5.31	7.10	3.62	4.71	6.55	6.92	6.21	9.67	5.33
GPTQ		4.13	5.39	7.18	4.39	4.74	6.57	7.74	97.03	27.77	6.33
MXQ		4.13	5.33	7.17	3.72	4.74	6.57	6.94	6.40	9.96	5.54

Table C.12 Perplexity results of SensiMiLP vs KurtMiLP (3-bit)

			Llama-2-7B			Llama-2-13B			Llama-3-8B		
Method	Config	BPP	WikiText2	C4	MEM	WikiText2	C4	MEM	WikiText2	C4	MEM
MXQ-SM1		3.51	5.73	7.72	3.30	4.99	6.91	6.08	7.33	11.80	4.94
MXQ-KM1		3.51	5.80	7.80	3.30	5.00	6.91	6.07	7.68	12.28	4.94
MXQ-SM2		3.51	5.72	7.72	3.30	4.98	6.91	6.08	7.64	12.10	4.94
MXQ-KM2		3.51	5.71	7.71	3.31	5.00	6.89	6.08	8.02	12.81	4.95
MXQ-SM3	b3g32	3.51	5.71	7.73	3.31	5.01	6.89	6.08	7.64	12.10	4.94
MXQ-KM3		3.51	5.69	7.65	3.31	4.99	6.90	6.08	7.90	12.66	4.94
HQQ		3.51	5.63	7.55	3.93	4.89	6.78	7.30	7.09	11.16	5.65
GPTQ		3.51	5.94	7.81	3.20	5.09	6.96	5.93	17.76	17.98	4.88
MXQ		3.51	5.65	7.61	3.43	4.89	6.82	6.49	7.24	11.49	5.21
MXQ-SM1		3.25	5.93	8.03	3.17	5.13	7.07	5.80	8.19	13.05	4.79
MXQ-KM1		3.25	6.08	8.24	3.17	5.12	7.08	5.80	8.35	13.44	4.79
MXQ-SM2		3.25	6.04	8.17	3.17	5.13	7.07	5.80	8.22	13.20	4.79
MXQ-KM2		3.25	5.98	8.11	3.17	5.11	7.06	5.80	8.23	13.19	4.78
MXQ-SM3	b3g64	3.25	5.97	8.04	3.17	5.12	7.07	5.80	8.22	13.20	4.79
MXQ-KM3		3.25	5.94	8.04	3.17	5.12	7.06	5.80	8.46	13.64	4.79
HQQ		3.25	5.82	7.81	3.32	4.98	6.94	6.22	7.81	12.36	5.02
GPTQ		3.25	6.13	8.07	2.98	5.14	7.06	5.49	11.16	14.33	4.64
MXQ		3.25	5.96	8.03	3.28	5.01	6.98	6.08	9.67	15.42	4.87
MXQ-SM1		3.13	6.18	8.38	3.11	5.15	7.13	5.67	9.27	14.82	4.72
MXQ-KM1		3.13	6.21	8.43	3.11	5.16	7.13	5.67	9.22	14.80	4.72
MXQ-SM2		3.13	6.19	8.38	3.11	5.15	7.14	5.67	9.27	14.84	4.72
MXQ-KM2		3.13	6.20	8.40	3.11	5.15	7.13	5.67	9.24	14.85	4.72
MXQ-SM3	b3g128	3.13	6.20	8.41	3.11	5.16	7.13	5.67	9.27	14.84	4.72
MXQ-KM3		3.13	6.19	8.38	3.11	5.16	7.13	5.67	9.24	14.85	4.72
HQQ		3.13	6.20	8.40	3.11	5.15	7.14	5.67	9.31	14.89	4.71
GPTQ		3.13	6.32	8.30	2.87	5.24	7.19	5.27	52.78	30.04	4.52
MXQ		3.13	6.22	8.35	3.15	5.17	7.18	5.82	13.40	20.90	4.74

Table C.13 Perplexity results of SensiMiLP vs KurtMiLP (other bits part1)

			Llama-2-7B			Llama-2-13B			Llama-3-8B		
Method	Config	BPP	WikiText2	C4	MEM	WikiText2	C4	MEM	WikiText2	C4	MEM
FP16	base	16.00	5.18	6.95	12.55	4.63	6.45	19.21	5.81	8.98	14.96
MXQ-SM1		6.89	5.23	7.00	5.77	4.66	6.48	10.87	6.03	9.36	7.56
MXQ-SM1		5.72	5.25	7.03	4.86	4.67	6.49	9.19	6.06	9.41	6.62
MXQ-SM1		5.02	5.26	7.04	4.33	4.68	6.51	8.08	6.08	9.40	6.05
MXQ-SM1		4.21	5.32	7.12	3.71	4.70	6.55	6.90	6.24	9.69	5.38
MXQ-SM1		4.17	5.34	7.14	3.68	4.72	6.56	6.82	6.29	9.77	5.35
MXQ-SM1		4.11	5.36	7.17	3.64	4.74	6.58	6.72	6.44	10.04	5.30
MXQ-SM1		4.07	5.37	7.21	3.62	4.76	6.60	6.68	6.64	10.47	5.30
MXQ-SM1		3.95	5.47	7.35	3.55	4.80	6.66	6.55	6.69	10.47	5.21
MXQ-SM1		3.87	5.53	7.44	3.51	4.83	6.71	6.47	7.07	11.20	5.17
MXQ-SM1		3.83	5.54	7.46	3.49	4.83	6.71	6.42	7.22	11.38	5.14
MXQ-SM1		3.65	5.65	7.64	3.39	4.91	6.83	6.23	7.08	11.23	5.02
MXQ-SM1		3.19	6.04	8.20	3.14	5.14	7.09	5.74	8.49	13.51	4.75
MXQ-SM1		3.15	6.08	8.25	3.12	5.16	7.13	5.69	8.68	13.92	4.73
MXQ-SM1		3.11	6.25	8.45	3.09	5.22	7.22	5.64	12.93	20.68	4.70
MXQ-SM1		3.07	6.87	9.31	3.04	5.33	7.40	5.57	11.19	17.86	4.65
MXQ-KM1		6.89	5.24	7.00	5.77	4.66	6.48	10.90	6.02	9.34	7.56
MXQ-KM1		5.72	5.25	7.04	4.86	4.67	6.50	9.24	6.05	9.39	6.62
MXQ-KM1		5.02	5.28	7.06	4.33	4.68	6.51	8.08	6.06	9.38	6.06
MXQ-KM1		4.21	5.32	7.13	3.71	4.72	6.56	6.92	6.24	9.69	5.38
MXQ-KM1		4.17	5.34	7.14	3.68	4.72	6.56	6.82	6.30	9.78	5.35
MXQ-KM1		4.11	5.36	7.18	3.64	4.75	6.57	6.72	6.42	10.04	5.30
MXQ-KM1		4.07	5.37	7.21	3.62	4.76	6.60	6.68	6.89	10.85	5.29
MXQ-KM1		3.95	5.47	7.34	3.55	4.80	6.64	6.54	6.89	10.80	5.21
MXQ-KM1		3.87	5.51	7.41	3.51	4.83	6.71	6.47	7.17	11.28	5.16
MXQ-KM1		3.83	5.54	7.47	3.48	4.83	6.69	6.42	7.00	11.06	5.14
MXQ-KM1		3.65	5.69	7.68	3.38	4.95	6.83	6.22	7.43	11.90	5.02
MXQ-KM1		3.19	6.12	8.30	3.14	5.12	7.09	5.74	8.65	13.82	4.75
MXQ-KM1		3.15	6.19	8.39	3.12	5.15	7.12	5.69	8.90	14.09	4.73
MXQ-KM1		3.11	6.30	8.57	3.09	5.23	7.25	5.64	10.55	16.96	4.69
MXQ-KM1		3.07	6.89	9.45	3.04	5.31	7.38	5.56	10.73	17.08	4.65
MXQ-SM2		6.89	5.22	7.00	5.75	4.66	6.48	10.98	6.04	9.37	7.56
MXQ-SM2		5.72	5.25	7.02	4.85	4.67	6.49	9.15	6.05	9.38	6.61
MXQ-SM2		5.02	5.26	7.05	4.32	4.68	6.50	8.07	6.08	9.40	6.05
MXQ-SM2		4.21	5.32	7.12	3.71	4.72	6.56	6.91	6.25	9.70	5.38
MXQ-SM2		4.17	5.34	7.14	3.68	4.72	6.57	6.82	6.29	9.76	5.35
MXQ-SM2		4.11	5.35	7.17	3.64	4.75	6.58	6.72	6.43	10.03	5.30
MXQ-SM2		4.07	5.39	7.21	3.62	4.76	6.61	6.68	6.49	10.17	5.28
MXQ-SM2		3.95	5.46	7.34	3.55	4.80	6.66	6.55	6.97	10.97	5.21
MXQ-SM2		3.87	5.50	7.41	3.51	4.83	6.71	6.47	6.85	10.75	5.16
MXQ-SM2		3.83	5.53	7.45	3.48	4.84	6.72	6.42	7.11	11.36	5.14
MXQ-SM2		3.65	5.64	7.64	3.39	4.95	6.84	6.22	7.38	11.74	5.03
MXQ-SM2		3.19	6.06	8.22	3.14	5.14	7.09	5.74	8.37	13.43	4.75
MXQ-SM2		3.15	6.08	8.25	3.12	5.15	7.12	5.69	9.08	14.52	4.73
MXQ-SM2		3.11	6.34	8.58	3.09	5.21	7.20	5.64	11.24	17.72	4.69

Table C.14 Perplexity results of SensiMiLP vs KurtMiLP (other bits part2)

			Llama-2-7B			Llama-2-13B			Llama-3-8B		
Method	Config	BPP	WikiText2	C4	MEM	WikiText2	C4	MEM	WikiText2	C4	MEM
FP16	base	16.00	5.18	6.95	12.55	4.63	6.45	19.21	5.81	8.98	14.96
MXQ-SM2		3.07	6.86	9.28	3.05	5.36	7.40	5.56	13.97	21.63	4.65
MXQ-KM2		6.89	5.23	7.00	5.76	4.66	6.48	10.86	6.02	9.32	7.56
MXQ-KM2		5.72	5.25	7.02	4.86	4.67	6.49	9.18	6.04	9.39	6.62
MXQ-KM2		5.02	5.26	7.05	4.33	4.68	6.50	8.08	6.06	9.38	6.06
MXQ-KM2		4.21	5.32	7.12	3.71	4.70	6.55	6.89	6.25	9.69	5.38
MXQ-KM2		4.17	5.34	7.14	3.68	4.72	6.56	6.82	6.30	9.78	5.35
MXQ-KM2		4.11	5.38	7.23	3.64	4.74	6.58	6.72	6.42	10.04	5.30
MXQ-KM2		4.07	5.41	7.25	3.62	4.76	6.60	6.68	6.50	10.22	5.28
MXQ-KM2		3.95	5.46	7.33	3.55	4.80	6.66	6.55	6.84	10.70	5.22
MXQ-KM2		3.87	5.53	7.44	3.50	4.83	6.70	6.47	7.13	11.23	5.16
MXQ-KM2		3.83	5.53	7.45	3.49	4.84	6.73	6.42	7.44	11.67	5.14
MXQ-KM2		3.65	5.63	7.61	3.39	4.95	6.83	6.22	7.60	12.08	5.03
MXQ-KM2		3.19	6.04	8.17	3.14	5.13	7.08	5.74	8.59	13.64	4.76
MXQ-KM2		3.15	6.09	8.27	3.12	5.14	7.12	5.69	8.98	14.44	4.73
MXQ-KM2		3.11	7.42	10.13	3.09	5.20	7.21	5.65	10.37	16.44	4.69
MXQ-KM2		3.07	6.75	9.20	3.04	5.36	7.39	5.56	17.41	27.29	4.65
MXQ-SM3		6.89	5.23	7.00	5.76	4.66	6.48	10.93	6.04	9.37	7.56
MXQ-SM3		5.72	5.25	7.03	4.87	4.67	6.49	9.22	6.05	9.38	6.61
MXQ-SM3		5.02	5.26	7.05	4.33	4.68	6.50	8.07	6.08	9.40	6.05
MXQ-SM3		4.21	5.33	7.12	3.71	4.72	6.56	6.90	6.25	9.70	5.38
MXQ-SM3		4.17	5.33	7.14	-	4.72	6.57	6.81	6.29	9.76	5.35
MXQ-SM3		4.11	5.36	7.17	-	4.75	6.58	6.72	6.43	10.03	5.30
MXQ-SM3		4.07	5.37	7.19	3.62	4.76	6.60	6.68	6.49	10.17	5.28
MXQ-SM3		3.95	5.46	7.34	3.55	4.80	6.66	6.55	6.97	10.97	5.21
MXQ-SM3		3.87	5.50	7.41	3.51	4.83	6.71	6.47	6.85	10.75	5.16
MXQ-SM3		3.83	5.56	7.49	3.49	4.84	6.72	6.42	7.11	11.36	5.14
MXQ-SM3		3.65	5.64	7.64	3.39	4.95	6.85	6.22	7.38	11.74	5.03
MXQ-SM3		3.19	6.15	8.34	3.14	5.14	7.10	5.74	8.37	13.43	4.75
MXQ-SM3		3.15	6.08	8.25	3.12	5.15	7.12	5.69	9.08	14.52	4.73
MXQ-SM3		3.11	6.40	8.60	3.09	5.23	7.22	5.63	11.24	17.72	4.69
MXQ-SM3		3.07	14.62	24.89	3.04	5.36	7.39	5.56	13.97	21.63	4.65
MXQ-KM3		6.89	5.23	7.00	5.76	4.66	6.48	10.88	6.02	9.33	7.56
MXQ-KM3		5.72	5.24	7.03	4.86	4.67	6.49	9.22	6.05	9.38	6.62
MXQ-KM3		5.02	5.26	7.04	4.33	4.68	6.51	8.09	6.06	9.39	6.05
MXQ-KM3		4.21	5.33	7.12	3.71	4.72	6.56	6.90	6.25	9.70	5.38
MXQ-KM3		4.17	5.34	7.13	3.68	4.72	6.56	6.80	6.30	9.80	5.35
MXQ-KM3		4.11	5.36	7.17	3.64	4.75	6.58	6.72	6.46	10.09	5.31
MXQ-KM3		4.07	5.39	7.22	3.62	4.75	6.59	6.68	6.79	10.64	5.28
MXQ-KM3		3.95	5.43	7.27	3.55	4.81	6.66	6.55	6.95	10.96	5.22
MXQ-KM3		3.87	5.48	7.36	3.51	4.83	6.70	6.47	7.46	11.80	5.16
MXQ-KM3		3.83	5.51	7.39	3.49	4.87	6.72	6.42	7.41	11.59	5.14
MXQ-KM3		3.65	5.61	7.55	3.39	4.92	6.84	6.23	7.60	12.08	5.03
MXQ-KM3		3.19	6.02	8.12	3.14	5.14	7.10	5.73	8.76	14.03	4.75
MXQ-KM3		3.15	6.09	8.27	3.12	5.14	7.12	5.69	9.13	14.66	4.73
MXQ-KM3		3.11	6.33	8.58	3.09	5.21	7.22	5.64	10.05	15.65	4.70
MXQ-KM3		3.07	7.10	9.51	3.04	5.34	7.37	5.56	10.45	16.67	4.65

Table C.15 Perplexity results of SensiMiLP vs Ablation (3 and 4-bit)

			Llama-2-7B			Llama-2-13B			Llama-3-8B		
Method	Config	BPP	WikiText2	C4	MEM	WikiText2	C4	MEM	WikiText2	C4	MEM
MXQ-SM1	b4g32	4.51	5.28	7.07	3.94	4.69	6.52	7.31	6.13	9.49	5.64
MXQ-SM2		4.51	5.28	7.08	3.94	4.69	6.53	7.35	6.14	9.52	5.62
MXQ-SM3		4.51	5.29	7.09	3.94	4.70	6.53	7.37	6.14	9.52	5.62
MXQ-SMAB		4.51	5.28	7.07	3.93	4.69	6.51	7.33	6.07	9.41	5.65
MXQ-SM1	b4g64	4.25	5.31	7.11	3.74	4.70	6.55	6.99	6.19	9.61	5.41
MXQ-SM2		4.25	5.31	7.12	3.74	4.71	6.55	6.99	6.22	9.65	5.41
MXQ-SM3		4.25	5.31	7.12	3.74	4.70	6.55	6.99	6.22	9.65	5.41
MXQ-SMAB		4.25	5.31	7.12	3.74	4.70	6.54	6.98	6.20	9.61	5.41
MXQ-SM1	b4g128	4.13	5.35	7.16	3.65	4.73	6.57	6.74	6.36	9.91	5.31
MXQ-SM2		4.13	5.35	7.16	3.65	4.73	6.57	6.74	6.37	9.92	5.31
MXQ-SM3		4.13	5.35	7.16	-	4.73	6.57	6.74	6.37	9.92	5.31
MXQ-SMAB		4.13	5.35	7.16	3.65	4.73	6.57	6.74	6.37	9.92	5.31
MXQ-SM1	b3g32	3.51	5.73	7.72	3.30	4.99	6.91	6.08	7.33	11.80	4.94
MXQ-SM2		3.51	5.72	7.72	3.30	4.98	6.91	6.08	7.64	12.10	4.94
MXQ-SM3		3.51	5.71	7.73	3.31	5.01	6.89	6.08	7.64	12.10	4.94
MXQ-SMAB		3.51	5.87	7.93	3.32	4.94	6.85	6.08	8.03	12.87	4.94
MXQ-SM1	b3g64	3.25	5.93	8.03	3.17	5.13	7.07	5.80	8.19	13.05	4.79
MXQ-SM2		3.25	6.04	8.17	3.17	5.13	7.07	5.80	8.22	13.20	4.79
MXQ-SM3		3.25	5.97	8.04	3.17	5.12	7.07	5.80	8.22	13.20	4.79
MXQ-SMAB		3.25	5.93	8.04	3.17	5.13	7.08	5.80	8.72	13.89	4.78
MXQ-SM1	b3g128	3.13	6.18	8.38	3.11	5.15	7.13	5.67	9.27	14.82	4.72
MXQ-SM2		3.13	6.19	8.38	3.11	5.15	7.14	5.67	9.27	14.84	4.72
MXQ-SM3		3.13	6.20	8.41	3.11	5.16	7.13	5.67	9.27	14.84	4.72
MXQ-SMAB		3.13	6.18	8.37	3.11	5.15	7.13	5.67	9.28	14.83	4.71

Table C.16 Perplexity results of SensiMiLP vs Ablation (other bits)

			Llama-2-7B			Llama-2-13B			Llama-3-8B		
Method	Config	BPP	WikiText2	C4	MEM	WikiText2	C4	MEM	WikiText2	C4	MEM
MXQ-SM1		6.89	5.23	7.00	5.77	4.66	6.48	10.87	6.03	9.36	7.56
MXQ-SM1		5.72	5.25	7.03	4.86	4.67	6.49	9.19	6.06	9.41	6.62
MXQ-SM1		5.02	5.26	7.04	4.33	4.68	6.51	8.08	6.08	9.40	6.05
MXQ-SM1		4.21	5.32	7.12	3.71	4.70	6.55	6.90	6.24	9.69	5.38
MXQ-SM1		4.17	5.34	7.14	3.68	4.72	6.56	6.82	6.29	9.77	5.35
MXQ-SM1		4.11	5.36	7.17	3.64	4.74	6.58	6.72	6.44	10.04	5.30
MXQ-SM1		4.07	5.37	7.21	3.62	4.76	6.60	6.68	6.64	10.47	5.30
MXQ-SM1		3.95	5.47	7.35	3.55	4.80	6.66	6.55	6.69	10.47	5.21
MXQ-SM1		3.87	5.53	7.44	3.51	4.83	6.71	6.47	7.07	11.20	5.17
MXQ-SM1		3.83	5.54	7.46	3.49	4.83	6.71	6.42	7.22	11.38	5.14
MXQ-SM1		3.65	5.65	7.64	3.39	4.91	6.83	6.23	7.08	11.23	5.02
MXQ-SM1		3.19	6.04	8.20	3.14	5.14	7.09	5.74	8.49	13.51	4.75
MXQ-SM1		3.15	6.08	8.25	3.12	5.16	7.13	5.69	8.68	13.92	4.73
MXQ-SM1		3.11	6.25	8.45	3.09	5.22	7.22	5.64	12.93	20.68	4.70
MXQ-SM1		3.07	6.87	9.31	3.04	5.33	7.40	5.57	11.19	17.86	4.65
MXQ-SM2		6.89	5.22	7.00	5.75	4.66	6.48	10.98	6.04	9.37	7.56
MXQ-SM2		5.72	5.25	7.02	4.85	4.67	6.49	9.15	6.05	9.38	6.61
MXQ-SM2		5.02	5.26	7.05	4.32	4.68	6.50	8.07	6.08	9.40	6.05
MXQ-SM2		4.21	5.32	7.12	3.71	4.72	6.56	6.91	6.25	9.70	5.38
MXQ-SM2		4.17	5.34	7.14	3.68	4.72	6.57	6.82	6.29	9.76	5.35
MXQ-SM2		4.11	5.35	7.17	3.64	4.75	6.58	6.72	6.43	10.03	5.30
MXQ-SM2		4.07	5.39	7.21	3.62	4.76	6.61	6.68	6.49	10.17	5.28
MXQ-SM2		3.95	5.46	7.34	3.55	4.80	6.66	6.55	6.97	10.97	5.21
MXQ-SM2		3.87	5.50	7.41	3.51	4.83	6.71	6.47	6.85	10.75	5.16
MXQ-SM2		3.83	5.53	7.45	3.48	4.84	6.72	6.42	7.11	11.36	5.14
MXQ-SM2		3.65	5.64	7.64	3.39	4.95	6.84	6.22	7.38	11.74	5.03
MXQ-SM2		3.19	6.06	8.22	3.14	5.14	7.09	5.74	8.37	13.43	4.75
MXQ-SM2		3.15	6.08	8.25	3.12	5.15	7.12	5.69	9.08	14.52	4.73
MXQ-SM2		3.11	6.34	8.58	3.09	5.21	7.20	5.64	11.24	17.72	4.69
MXQ-SM2		3.07	6.86	9.28	3.05	5.36	7.40	5.56	13.97	21.63	4.65
MXQ-SM3		6.89	5.23	7.00	5.76	4.66	6.48	10.93	6.04	9.37	7.56
MXQ-SM3		5.72	5.25	7.03	4.87	4.67	6.49	9.22	6.05	9.38	6.61
MXQ-SM3		5.02	5.26	7.05	4.33	4.68	6.50	8.07	6.08	9.40	6.05
MXQ-SM3		4.21	5.33	7.12	3.71	4.72	6.56	6.90	6.25	9.70	5.38
MXQ-SM3		4.17	5.33	7.14	-	4.72	6.57	6.81	6.29	9.76	5.35
MXQ-SM3		4.11	5.36	7.17	-	4.75	6.58	6.72	6.43	10.03	5.30
MXQ-SM3		4.07	5.37	7.19	3.62	4.76	6.60	6.68	6.49	10.17	5.28
MXQ-SM3		3.95	5.46	7.34	3.55	4.80	6.66	6.55	6.97	10.97	5.21
MXQ-SM3		3.87	5.50	7.41	3.51	4.83	6.71	6.47	6.85	10.75	5.16
MXQ-SM3		3.83	5.56	7.49	3.49	4.84	6.72	6.42	7.11	11.36	5.14
MXQ-SM3		3.65	5.64	7.64	3.39	4.95	6.85	6.22	7.38	11.74	5.03
MXQ-SM3		3.19	6.15	8.34	3.14	5.14	7.10	5.74	8.37	13.43	4.75
MXQ-SM3		3.15	6.08	8.25	3.12	5.15	7.12	5.69	9.08	14.52	4.73
MXQ-SM3		3.11	6.40	8.60	3.09	5.23	7.22	5.63	11.24	17.72	4.69
MXQ-SM3		3.07	14.62	24.89	3.04	5.36	7.39	5.56	13.97	21.63	4.65
MXQ-SMAB		6.89	5.23	7.02	5.77	4.66	6.48	10.95	6.03	9.35	7.56
MXQ-SMAB		5.72	5.25	7.02	4.85	4.67	6.49	9.22	6.05	9.37	6.63
MXQ-SMAB		5.02	5.27	7.07	4.33	4.68	6.50	8.12	6.06	9.41	6.06
MXQ-SMAB		4.21	5.32	7.13	3.71	4.72	6.56	6.92	6.25	9.71	5.38
MXQ-SMAB		4.17	5.34	7.15	3.68	4.73	6.56	6.83	6.30	9.80	5.35
MXQ-SMAB		4.11	5.36	7.18	3.64	4.75	6.58	6.72	6.44	10.04	5.30
MXQ-SMAB		4.07	5.38	7.21	3.62	4.76	6.60	6.68	6.90	10.78	5.28
MXQ-SMAB		3.95	5.47	7.35	3.55	4.80	6.66	6.55	6.93	10.82	5.21
MXQ-SMAB		3.87	5.50	7.38	3.50	4.86	6.70	6.45	7.30	11.41	5.16
MXQ-SMAB		3.83	5.58	7.48	3.49	4.85	6.71	6.42	7.59	11.91	5.14
MXQ-SMAB		3.65	5.66	7.64	3.39	4.91	6.83	6.23	7.32	11.53	5.03
MXQ-SMAB		3.19	6.16	8.32	3.14	5.13	7.10	5.74	9.04	14.36	4.75
MXQ-SMAB		3.15	6.16	8.35	3.12	5.14	7.13	5.69	9.18	14.68	4.73
MXQ-SMAB		3.11	6.30	8.53	3.09	5.23	7.24	5.63	10.13	16.12	4.69
MXQ-SMAB		3.07	6.75	9.17	3.04	5.32	7.35	5.56	11.87	18.96	4.65

Table C.17 Perplexity results of KurtMiLP vs Ablation (3 and 4-bit)

			Llama-2-7B			Llama-2-13B			Llama-3-8B		
Method	Config	BPP	WikiText2	C4	MEM	WikiText2	C4	MEM	WikiText2	C4	MEM
MXQ-KM1	b4g32	4.51	5.30	7.08	3.94	4.69	6.53	7.31	6.10	9.45	5.65
MXQ-KM2		4.51	5.28	7.08	3.94	4.69	6.52	7.37	6.10	9.45	5.64
MXQ-KM3		4.51	5.28	7.08	3.94	4.69	6.53	7.37	6.12	9.48	5.64
MXQ-KMAB		4.51	5.28	7.07	3.93	4.69	6.51	7.33	6.07	9.41	5.65
MXQ-KM1	b4g64	4.25	5.31	7.12	3.74	4.70	6.54	6.99	6.21	9.62	5.41
MXQ-KM2		4.25	5.31	7.12	3.74	4.70	6.54	6.99	6.20	9.60	5.41
MXQ-KM3		4.25	5.31	7.11	3.74	4.70	6.55	6.98	6.20	9.61	5.41
MXQ-KMAB		4.25	5.31	7.12	3.74	4.70	6.54	6.98	6.20	9.61	5.41
MXQ-KM1	b4g128	4.13	5.35	7.16	3.65	4.74	6.57	6.74	6.37	9.91	5.31
MXQ-KM2		4.13	5.35	7.16	3.65	4.73	6.57	6.74	6.37	9.93	5.31
MXQ-KM3		4.13	5.35	7.16	3.65	4.73	6.57	6.74	6.37	9.93	5.31
MXQ-KMAB		4.13	5.35	7.16	3.65	4.73	6.57	6.74	6.37	9.92	5.31
MXQ-KM1	b3g32	3.51	5.80	7.80	3.30	5.00	6.91	6.07	7.68	12.28	4.94
MXQ-KM2		3.51	5.71	7.71	3.31	5.00	6.89	6.08	8.02	12.81	4.95
MXQ-KM3		3.51	5.69	7.65	3.31	4.99	6.90	6.08	7.90	12.66	4.94
MXQ-KMAB		3.51	5.87	7.93	3.32	4.94	6.85	6.08	8.03	12.87	4.94
MXQ-KM1	b3g64	3.25	6.08	8.24	3.17	5.12	7.08	5.80	8.35	13.44	4.79
MXQ-KM2		3.25	5.98	8.11	3.17	5.11	7.06	5.80	8.23	13.19	4.78
MXQ-KM3		3.25	5.94	8.04	3.17	5.12	7.06	5.80	8.46	13.64	4.79
MXQ-KMAB		3.25	5.93	8.04	3.17	5.13	7.08	5.80	8.72	13.89	4.78
MXQ-KM1	b3g128	3.13	6.21	8.43	3.11	5.16	7.13	5.67	9.22	14.80	4.72
MXQ-KM2		3.13	6.20	8.40	3.11	5.15	7.13	5.67	9.24	14.85	4.72
MXQ-KM3		3.13	6.19	8.38	3.11	5.16	7.13	5.67	9.24	14.85	4.72
MXQ-KMAB		3.13	6.18	8.37	3.11	5.15	7.13	5.67	9.28	14.83	4.71

Table C.18 Perplexity results of KurtMiLP vs Ablation (other bits)

			Llama-2-7B			Llama-2-13B			Llama-3-8B		
Method	Config	BPP	WikiText2	C4	MEM	WikiText2	C4	MEM	WikiText2	C4	MEM
MXQ-KM1		6.89	5.24	7.00	5.77	4.66	6.48	10.90	6.02	9.34	7.56
MXQ-KM1		5.72	5.25	7.04	4.86	4.67	6.50	9.24	6.05	9.39	6.62
MXQ-KM1		5.02	5.28	7.06	4.33	4.68	6.51	8.08	6.06	9.38	6.06
MXQ-KM1		4.21	5.32	7.13	3.71	4.72	6.56	6.92	6.24	9.69	5.38
MXQ-KM1		4.17	5.34	7.14	3.68	4.72	6.56	6.82	6.30	9.78	5.35
MXQ-KM1		4.11	5.36	7.18	3.64	4.75	6.57	6.72	6.42	10.04	5.30
MXQ-KM1		4.07	5.37	7.21	3.62	4.76	6.60	6.68	6.89	10.85	5.29
MXQ-KM1		3.95	5.47	7.34	3.55	4.80	6.64	6.54	6.89	10.80	5.21
MXQ-KM1		3.87	5.51	7.41	3.51	4.83	6.71	6.47	7.17	11.28	5.16
MXQ-KM1		3.83	5.54	7.47	3.48	4.83	6.69	6.42	7.00	11.06	5.14
MXQ-KM1		3.65	5.69	7.68	3.38	4.95	6.83	6.22	7.43	11.90	5.02
MXQ-KM1		3.19	6.12	8.30	3.14	5.12	7.09	5.74	8.65	13.82	4.75
MXQ-KM1		3.15	6.19	8.39	3.12	5.15	7.12	5.69	8.90	14.09	4.73
MXQ-KM1		3.11	6.30	8.57	3.09	5.23	7.25	5.64	10.55	16.96	4.69
MXQ-KM1		3.07	6.89	9.45	3.04	5.31	7.38	5.56	10.73	17.08	4.65
MXQ-KM2		6.89	5.23	7.00	5.76	4.66	6.48	10.86	6.02	9.32	7.56
MXQ-KM2		5.72	5.25	7.02	4.86	4.67	6.49	9.18	6.04	9.39	6.62
MXQ-KM2		5.02	5.26	7.05	4.33	4.68	6.50	8.08	6.06	9.38	6.06
MXQ-KM2		4.21	5.32	7.12	3.71	4.70	6.55	6.89	6.25	9.69	5.38
MXQ-KM2		4.17	5.34	7.14	3.68	4.72	6.56	6.82	6.30	9.78	5.35
MXQ-KM2		4.11	5.38	7.23	3.64	4.74	6.58	6.72	6.42	10.04	5.30
MXQ-KM2		4.07	5.41	7.25	3.62	4.76	6.60	6.68	6.50	10.22	5.28
MXQ-KM2		3.95	5.46	7.33	3.55	4.80	6.66	6.55	6.84	10.70	5.22
MXQ-KM2		3.87	5.53	7.44	3.50	4.83	6.70	6.47	7.13	11.23	5.16
MXQ-KM2		3.83	5.53	7.45	3.49	4.84	6.73	6.42	7.44	11.67	5.14
MXQ-KM2		3.65	5.63	7.61	3.39	4.95	6.83	6.22	7.60	12.08	5.03
MXQ-KM2		3.19	6.04	8.17	3.14	5.13	7.08	5.74	8.59	13.64	4.76
MXQ-KM2		3.15	6.09	8.27	3.12	5.14	7.12	5.69	8.98	14.44	4.73
MXQ-KM2		3.11	7.42	10.13	3.09	5.20	7.21	5.65	10.37	16.44	4.69
MXQ-KM2		3.07	6.75	9.20	3.04	5.36	7.39	5.56	17.41	27.29	4.65
MXQ-KM3		6.89	5.23	7.00	5.76	4.66	6.48	10.88	6.02	9.33	7.56
MXQ-KM3		5.72	5.24	7.03	4.86	4.67	6.49	9.22	6.05	9.38	6.62
MXQ-KM3		5.02	5.26	7.04	4.33	4.68	6.51	8.09	6.06	9.39	6.05
MXQ-KM3		4.21	5.33	7.12	3.71	4.72	6.56	6.90	6.25	9.70	5.38
MXQ-KM3		4.17	5.34	7.13	3.68	4.72	6.56	6.80	6.30	9.80	5.35
MXQ-KM3		4.11	5.36	7.17	3.64	4.75	6.58	6.72	6.46	10.09	5.31
MXQ-KM3		4.07	5.39	7.22	3.62	4.75	6.59	6.68	6.79	10.64	5.28
MXQ-KM3		3.95	5.43	7.27	3.55	4.81	6.66	6.55	6.95	10.96	5.22
MXQ-KM3		3.87	5.48	7.36	3.51	4.83	6.70	6.47	7.46	11.80	5.16
MXQ-KM3		3.83	5.51	7.39	3.49	4.87	6.72	6.42	7.41	11.59	5.14
MXQ-KM3		3.65	5.61	7.55	3.39	4.92	6.84	6.23	7.60	12.08	5.03
MXQ-KM3		3.19	6.02	8.12	3.14	5.14	7.10	5.73	8.76	14.03	4.75
MXQ-KM3		3.15	6.09	8.27	3.12	5.14	7.12	5.69	9.13	14.66	4.73
MXQ-KM3		3.11	6.33	8.58	3.09	5.21	7.22	5.64	10.05	15.65	4.70
MXQ-KM3		3.07	7.10	9.51	3.04	5.34	7.37	5.56	10.45	16.67	4.65
MXQ-KMAB		6.89	5.23	7.02	5.77	4.66	6.48	10.95	6.03	9.35	7.56
MXQ-KMAB		5.72	5.25	7.02	4.85	4.67	6.49	9.22	6.05	9.37	6.63
MXQ-KMAB		5.02	5.27	7.07	4.33	4.68	6.50	8.12	6.06	9.41	6.06
MXQ-KMAB		4.21	5.32	7.13	3.71	4.72	6.56	6.92	6.25	9.71	5.38
MXQ-KMAB		4.17	5.34	7.15	3.68	4.73	6.56	6.83	6.30	9.80	5.35
MXQ-KMAB		4.11	5.36	7.18	3.64	4.75	6.58	6.72	6.44	10.04	5.30
MXQ-KMAB		4.07	5.38	7.21	3.62	4.76	6.60	6.68	6.90	10.78	5.28
MXQ-KMAB		3.95	5.47	7.35	3.55	4.80	6.66	6.55	6.93	10.82	5.21
MXQ-KMAB		3.87	5.50	7.38	3.50	4.86	6.70	6.45	7.30	11.41	5.16
MXQ-KMAB		3.83	5.58	7.48	3.49	4.85	6.71	6.42	7.59	11.91	5.14
MXQ-KMAB		3.65	5.66	7.64	3.39	4.91	6.83	6.23	7.32	11.53	5.03
MXQ-KMAB		3.19	6.16	8.32	3.14	5.13	7.10	5.74	9.04	14.36	4.75
MXQ-KMAB		3.15	6.16	8.35	3.12	5.14	7.13	5.69	9.18	14.68	4.73
MXQ-KMAB		3.11	6.30	8.53	3.09	5.23	7.24	5.63	10.13	16.12	4.69
MXQ-KMAB		3.07	6.75	9.17	3.04	5.32	7.35	5.56	11.87	18.96	4.65