Master's Project

Rethinking Scale Selection in Microscaling Formats for LLM Inference

Danila Mishin

Master Student, Computer Science

danila.mishin@epfl.ch

Supervisor: Prof. Babak Falsafi
Laboratory: PARSA — Parallel Systems Architecture Laboratory
School: School of Computer and Communication Sciences (IC)

École polytechnique fédérale de Lausanne

2026

Acknowledgements

I would like to thank Prof. Falsafi for giving me the opportunity to work on this project and for his guidance throughout its development and my Research Scholarship journey. Thanks to his mentorship, I gained invaluable experience and learned how to do rigorous research and how to approach complex and open-ended problems. I am also grateful to Dr. Khodamoradi for his expert insights and the unique industry perspective that he brought. His technical expertise and constructive feedback were invaluable throughout this project. I also thank the members of PARSA for creating a stimulating research environment and supporting me throughout my Research Scholarship. Finally, I thank my family and friends for their encouragement and support throughout my time at EPFL.

Abstract

As the size of Large Language Models (LLMs) has increased over the past few years, quantization has become a cornerstone of production LLM deployment. The numerical format is the key component in quantization algorithms, as it enables a more efficient representation of model weights and activations during inference and training. Modern block-wise numerical formats achieve accurate 4-bit quantization by sharing a scaling factor among several values, amortizing the hardware cost and compute overhead. Selecting an optimal shared scale for a block is essential for achieving high accuracy. Yet, existing scale selection strategies such as `absmax`, `max` and `midmax` choose scales suboptimally from extreme values in the block, ignoring Hessian information and undermining model performance. This project addresses this problem and studies scale selection for 4-bit microscaling datatypes. We define the mathematical scale selection objective, study its properties, and formally show why existing scale selection strategies are suboptimal. We propose our own scale selection strategy `optscale` which minimizes our objective and uses Hessian information to optimally select the scale for a given block. We empirically validate our mathematical framework and conduct an end-to-end W4A4 evaluation on the Llama-3.1-8B-Instruct model. Our method improves Wikitext-2 perplexity by up to 1.23 perplexity points over existing scale selection strategies. We also show how scale selection interacts with quantization pre-processing algorithms such as GPTQ and micro-rotations.

Résumé

Comme la taille des grands modèles de langage (LLMs) a augmenté au cours des dernières années, la quantification est devenue une pierre angulaire du déploiement des LLMs en production. Les formats numériques sont le composant clé des algorithmes de quantification, car ils permettent une représentation plus efficace des poids et des activations des modèles pendant l'inférence et l'entraînement. Les formats numériques modernes par blocs permettent une quantification 4 bits précise en partageant un facteur d'échelle entre plusieurs valeurs, ce qui amortit le coût matériel et le surcoût de calcul. Choisir une échelle partagée optimale pour un bloc est essentiel pour obtenir une grande précision. Pourtant, les stratégies existantes de sélection d'échelle telles que `absmax`, `max` et `midmax` choisissent les échelles de manière sous-optimale à partir des valeurs extrêmes du bloc, en ignorant l'information hessienne et en dégradant les performances du modèle. Ce projet traite ce problème et étudie la sélection d'échelle pour les types de données de microscaling 4 bits. Nous définissons l'objectif mathématique de sélection d'échelle, étudions ses propriétés et montrons formellement pourquoi les stratégies existantes de sélection d'échelle sont sous-optimales. Nous proposons notre propre stratégie de sélection d'échelle, `optscale`, qui minimise notre objectif et utilise l'information hessienne pour sélectionner de manière optimale l'échelle d'un bloc donné. Nous validons empiriquement notre cadre mathématique et menons une évaluation W4A4 de bout en bout sur le modèle Llama-3.1-8B-Instruct. Notre méthode améliore la perplexité Wikitext-2 jusqu'à 1.23 point de perplexité par rapport aux stratégies existantes de sélection d'échelle. Nous montrons aussi comment la sélection d'échelle interagit avec les algorithmes de pré-traitement de quantification tels que GPTQ et les micro-rotations.

Contents

1 Introduction

Over the past few years, Artificial Intelligence (AI) has become a central research topic in Computer Science. AI systems are now widely adopted in various areas such as software development, medicine, banking, personal assistance, and many other domains [1]. Large Language Models (LLMs) have been central to this adoption as they provide a natural language interface between users and the systems.

As LLMs have grown rapidly in size, their efficient deployment has become a central topic among researchers and practitioners. PaLM used 540 billion dense parameters in 2022 [2], DeepSeek-V3 reported 671 billion total parameters in 2024 [3], and DeepSeek-V4-Pro reported 1.6 trillion total parameters in 2026 [4]. This continued growth in model size makes efficiency an important topic for deploying Large Language Models in practice.

During its lifecycle, an LLM goes through two main stages: training and inference. During training, the model learns from large datasets by adjusting its parameters to predict and generate text. During inference, the trained model is deployed to process user requests and generate responses. Focusing specifically on LLM inference is important as it accounts for a large share of AI energy use in datacenters. Unlike training, inference is performed every time a user writes a request to the model. Prior work reports that inference represents 60 to 70% of AI energy use in datacenters [5, 6], which makes inference efficiency a direct target for reducing deployment cost.

Quantization is one of the most effective methods for making LLM inference more efficient. By storing weights and activations with fewer bits, quantization reduces memory and compute overhead. Recent state-of-the-art methods show that 4-bit inference can recover up to 96% of the original accuracy [7], which makes low-precision inference a practical direction for efficient LLM deployment.

Early low-precision formats mostly used scalar numerical formats, where each element is encoded independently. Floating-point formats such as BF16 store their own sign, exponent, and mantissa fields for every value [8], while integer formats such as INT8 use software-

managed scales to map integer codes to real values [9]. Although this reduces the number of bits per element, each value is still encoded separately.

In LLM tensors, many neighboring values can often use a common dynamic range, and emerging block formats exploit this structure by sharing a scale across multiple elements. For example, OCP MX formats such as MXINT4 and MXFP4 use blocks of 32 values with one shared power-of-two scale, while NVFP4 uses smaller blocks of 16 values with an FP8 scale [10, 11]. In MXFP4, storing 32 4-bit values together with one 8-bit scale gives 4.25 bits per value, a 3.76× reduction in memory consumption compared to BF16.

Once several values share a scale, the choice of that scale becomes the central design problem. The scale datatype determines which dynamic ranges the block can represent, while the scale selection algorithm chooses one of those ranges for the observed weight and activation values. Since this single choice fixes the quantization grid for every element in the block, both the scale datatype and the selection algorithm directly determine the final accuracy of the quantized model.

Existing scale selection algorithms choose the scale sub-optimally and ignore the element-wise importance information that matters for LLM quantization. Algorithms such as `absmax` [7], `max` [12], and `midmax` [12] (hereafter referred to as *max-based*) choose the scale from extreme values in the block, but they do not use Hessian information to account for how strongly each element's quantization error affects the model's output. This makes them suboptimal when the largest-magnitude element is not the element whose error matters most.

This project studies how shared scales should be selected in microscaling formats for LLM inference. It explores both the mathematical structure of the scale-selection problem and its practical effect on quantized model quality. The main contributions are:

- An optimal scale-search algorithm that reduces end-to-end WikiText-2 Llama-3.1-8B-Instruct model perplexity by up to 1.23 points compared with existing scale-selection algorithms.
- A mathematical analysis that defines the scale-selection objective, studies its properties for different shared scale datatypes, and proves the suboptimality of max-based selection in a Hessian-outlier setting.
- An empirical study that validates the analysis, characterizes the distance between optimal and max-based scales, and evaluates optimal scale selection across multiple shared scale datatypes.

The report is organized from motivation to analysis and then to empirical validation. Chapter 2 introduces the necessary background, Chapter 3 mathematically studies the scale-selection problem, Chapter 4 validates the analysis in single-layer experiments, Chapter 5 evaluates the conclusions end-to-end, and Chapter 6 summarizes the findings.

2 Background

This chapter introduces the background needed to understand shared-scale quantization for LLM inference. It starts from broad LLM compression methods, then progressively narrows the focus to post-training quantization, the problem of outliers, microscaling formats, existing shared scale datatypes and selection algorithms.

2.1 Compression Techniques

Deploying LLMs at scale is often limited by the cost of storing and processing the model. Large models require substantial storage for their parameters, high memory bandwidth to stream these parameters during decoding, enough compute throughput to sustain matrix multiplications, and enough power budget to serve requests at scale. Compression techniques address these constraints by reducing how much data is stored, how much data is moved, or how expensive each arithmetic operation is.

Compression is better viewed as a design space than as a single technique. A compression method can change numerical precision, remove unnecessary parameters, replace matrices with structured approximations, or combine several of these choices. It can be applied during training or during inference, and it can either be calibrated on a dataset or be completely data-free. It also differs in whether the compressed numerical format is directly executed by the hardware or merely stored compactly before being expanded again.

One compression technique is *quantization*, which changes the numerical format used for model tensors. Instead of storing every value in a high-precision format, quantization maps values to a lower-bit codebook and usually uses one or more scales to recover the needed dynamic range. This can reduce model size and memory traffic, and when the hardware supports the low-bit format directly, it can also reduce energy cost [9, 13, 14, 15, 16, 17, 18, 19, 20, 21, 7, 22].

Another compression technique is *pruning*, which reduces a model size by deciding which weights or structures are unnecessary and removing them. The removed elements may be

individual weights, groups of weights, channels, heads, or larger blocks, depending on the sparsity pattern and the target hardware. Pruning reduces storage immediately, but it reduces compute only when the resulting sparse structure can be exploited efficiently by kernels or accelerators [23, 24, 25, 26].

SVD-based compression replaces dense weight matrices with lower-rank approximations. The basic idea is to keep only the most important singular components, which turns one large matrix multiplication into smaller factorized operations and reduces the number of stored parameters. Recent LLM methods improve this simple truncation by weighting matrix directions by importance, scaling features before decomposition, or choosing truncation levels with loss-aware criteria [27, 28, 29, 30].

These compression techniques can be combined. A model may be pruned and then quantized, quantized after a rotation, or compressed with a structured approximation before another low-bit numerical format is chosen. However, their effects are not independent. For example, when combining pruning and quantization, the order matters because one method can change the distribution seen by the next method [31]. Therefore, the final accuracy and speed cannot always be predicted by evaluating each branch in isolation.

For the rest of this chapter, we solely focus on quantization, as it is the central compression technique in our work.

2.2 Quantization Design Space

Quantization changes the numerical format of tensor values. Instead of keeping every value in its original full-precision format, the quantizer maps it into a low-bit *codebook*. This codebook is the set of numerical values that can be represented after quantization using fewer bits, decreasing the storage requirements for the model and increasing the bandwidth.

Quantization can be applied during training or applied after training. *Quantization-aware training* adapts the model while simulated quantization noise is present, which can recover accuracy but requires training resources and data [32]. *Post-training quantization* instead starts from a fixed pre-trained model and compresses it after training, making it attractive for LLM deployment where full retraining is expensive [13, 14].

Post-training quantization can also differ in how much it depends on additional information to calibrate the method. *Data-free methods* choose quantization parameters from weights and analytical rules only, which makes the final quantized model unbiased to the data, but can be ineffective at low bit widths [33, 32]. *Calibration-based methods* use a small representative dataset to collect activations, Hessian proxies, or outlier statistics without retraining the full model to construct a better codebook or determine better parameters for quantization [13, 19].

The simplest quantization algorithm is *round-to-nearest (RTN)*. RTN maps each value independently to the nearest representable code, making it a useful baseline because it does not

use calibration data or model-specific sensitivity information. *Hessian-aware quantization* goes further by quantizing the weights sequentially and compensating for the errors in the next yet-to-be-quantized weights [13].

Before a value is mapped to a value from the codebook, the tensor, the channel or the block can be normalized by a *scale* so that the low-bit codebook is able to cover the relevant numerical range [9, 34]. This choice becomes especially important when moving from 8-bit formats to 4-bit formats, because each shared dynamic range controls a much smaller set of representable values.

Quantization can target different tensors in the inference pipeline. *Weight quantization* compresses the static model parameters, so its quantization parameters can be chosen offline before serving [13, 15]. *Activation quantization* compresses input-dependent intermediate values, so its scales must either be computed at runtime or chosen from calibration data [14, 9]. *KV-cache quantization* targets the stored keys and values used during autoregressive decoding, which becomes important for long-context inference [22].

This project focuses on post-training quantization of weights and activations with scaling. The models studied here are already trained, and the central question is how to choose shared scales for their weights and activations under hardware-native low-bit formats. The rest of the chapter therefore narrows from general quantization choices toward the tensor targets, scale granularities, and scale-selection algorithms used in this setting.

2.3 Outliers, Sensitivity, and Transformations

Outliers are one of the main reasons why LLM quantization is difficult. Outliers are unusually large values or channels that dominate the quantization range [9]. When using a shared scale, because of the outliers, the codebook is often chosen such that it can accurately represent the outlier values to avoid clipping. Because of that, the remaining values are quantized with a coarser effective resolution, increasing the quantization error for most of the tensor.

Various *outlier-aware quantization* methods either handle outliers separately or change the tensor distribution. LLM.int8() keeps extreme activation channels in higher bitwidth, avoiding the need to force every value into the same low-bit numerical format [9]. SmoothQuant instead rescales the activation and weight tensors and moves the magnitude of the outliers from activations into the weights through equivalent per-channel scaling [14]. Another approach is to apply orthogonal rotations to weights and activations, preserving the computation while spreading concentrated magnitudes across coordinates. QuaRot and SpinQuant use rotations to spread outliers before quantization, making the tensors easier to quantize with low-bit formats [20, 21].

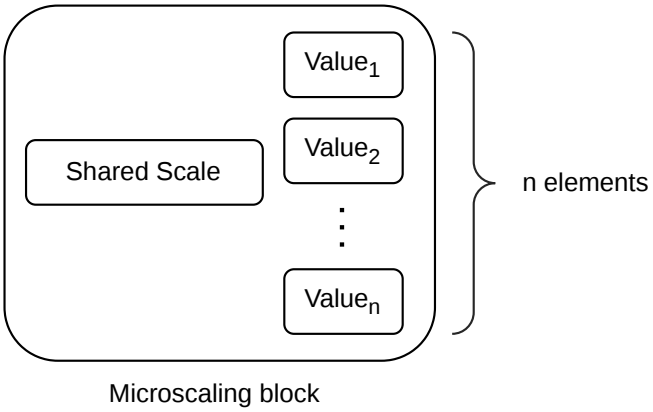


Figure 2.1: Microscaling block with one shared scale and multiple low-bit values.

2.4 Shared Scale

As was mentioned before, scaled quantization format uses a *scale* to adapt a fixed low-bit codebook to various numerical ranges. Scaling can be applied at various granularities. *Per-tensor scaling* shares one scale across an entire tensor, which globally softens the global dynamic range of the tensor with minimal cost but still does not eliminate the problem of outliers. *Per-channel scaling* works at finer granularity and scales individual channels instead of the whole tensor, which provides more control over the distribution spikes [9, 14]. *Per-block scaling* shares one scale across a small group of values, reducing outlier exposure even more.

Granularity of scaling controls the trade-off between accuracy and overhead. Fine-grained scaling improves accuracy because fewer values in the block compete for the same scale, which allows a better match of a codebook for the distribution. Coarse-grained scaling reduces additional memory overhead but also makes each scale responsible for a wider and potentially less uniform set of values.

A shared-scale format is defined by the *scale datatype* and the *scale-selection algorithm*. The scale datatype determines which scale values the numerical format supports. The scale-selection algorithm chooses one of these representable scale values for the observed tensor or block. Together, these two choices determine the quantization grid seen by every value that shares the scale, making them the primary design decisions in low-bit block formats. This will be the central topic of our project.

Table 2.1: Notation for shared-scale 4-bit formats used in this report. FP4 denotes E2M1 data values.

Format	Notation	Block scale	Value	Tensor scale	Block size
MXINT4	`MXINT4`	E8M0	INT4 (E0M4)	×	32
MXFP4	`MXFP4`	E8M0	FP4 (E2M1)	×	32
NVFP4	`E4M3FP4_g_t16`	E4M3	FP4 (E2M1)	✓	16
`E5M3FP4`	`E5M3FP4`	E5M3	FP4 (E2M1)	×	32
`E4M4FP4_g`	`E4M4FP4_g`	E4M4	FP4 (E2M1)	✓	32

2.5 Microscaling Formats

Microscaling formats are hardware-native block formats with support for shared scales. As shown in Figure 2.1, a small group of values shares one scale and stores only compact data values inside the block instead of giving every low-bit value its own scale. This layout reduces storage overhead and makes the shared scale part of the numerical format, which enables efficient compute.

OCP MX formats standardize this shared-scale layout for block-32 datatypes [10, 34]. MXINT4 combines a shared power-of-two scale with INT4 data values, while MXFP4 combines a shared power-of-two scale with E2M1 FP4 data values. In both cases, the block scale is restricted to powers of two, so changing the scale moves the whole block grid by exponential steps.

The notation used in this report follows the scale datatype first and the data datatype second. We use a floating point notation for the scale datatype, which is defined by the number of exponent bits (E) and the number of mantissa bits (M) (e.g., E2M1, E4M3, E5M3). The shared scale datatype followed by the value datatype determines the overall numerical format. For example, E4M3FP4 means FP4 (E2M1) data values with E4M3 block scales. The suffix `_g` denotes an additional FP32 tensor-level scale, while `_t16` denotes a 16-value block. If block size is 32, the notation _t32 is omitted for simplicity. Table 2.1 summarizes this notation for some of the formats used in this report.

The type of the value determines the shape of the distribution inside each scaled block. INT4 forms a uniformly spaced grid, so changing the scale changes both the range and the spacing uniformly. FP4 uses the E2M1 floating-point grid, which is non-uniform.

Floating-point scales generalize the *power-of-two scale* used by MX formats. Power-of-two scales have exponent bits but no mantissa bits, so adjacent scale values differ by powers of two. Floating-point scales, such as E4M3, E5M3, and E4M4, add mantissa bits to the scale datatype, creating denser choices inside each power-of-two range.

NVFP4 is the main standardized 4-bit format with floating-point block scales. It uses E4M3 block scales, E2M1 FP4 data values, 16-value blocks, and an FP32 tensor-level scale [11]. For the rest of this report, this configuration is referred to as `E4M3FP4_g_t16` rather than NVFP4.

This keeps the notation consistent with the wider set of floating-point scale formats evaluated here, where the name explicitly records the block-scale datatype, the value datatype, the tensor-level scale, and the block size.

This report also evaluates floating-point scale variants with E4M3, E5M3, and E4M4 block scales, both with and without the tensor-level FP32 scale. These variants are not standard NVFP4 layouts, but they isolate how the block-scale datatype and the tensor-level scale affect accuracy. The tensor-level scale is useful because it can place a limited block-scale grid in a better tensor-wide range before per-block scales are selected.

2.6 Scale Selection Algorithms

Scale selection is the quantization step that chooses a representable shared scale for a tensor or block. For microscaling formats, this choice is part of the quantization algorithm rather than fully specified by the datatype. The OCP MX specification defines the format layout, but leaves the exact scale-selection policy implementation-defined [10].

Scale-selection methods differ by what information they use. *Data-free* methods choose the scale only from the values in the block. *Calibration-based* methods use representative data, activation statistics, Hessian values, or other external information to optimize a scale objective.

Max-based scale selection is a data-free method that chooses the scale from maximum values in the block. For example, the common `absmax` rule selects the smallest representable scale that keeps the block absolute maximum within the data-codebook range [7]. TensorCast `max` derives the scale exponent from the block absolute maximum after accounting for the largest data-codebook value, while TensorCast `midmax` promotes the scale when the rescaled block maximum exceeds the data-format midmax threshold, balancing clipping error against quantization error [12].

Max-based methods are attractive for hardware-native formats because they are local, deterministic, and cheap to implement. Their limitation is that they treat the block maximum as the main proxy for the best scale. This ignores activation statistics and Hessian sensitivity, so the selected scale can preserve the largest value even when a different scale would reduce the error on more important values.

There are alternative data-free approaches to the max-based methods. Instead of deriving the scale directly from the maximum value, one can evaluate candidate scales against a local quantization-error objective such as MSE on the block values. ScaleSearch follows this direction for block floating-point scales, improving the scale choice without using Hessian information or calibration data [33]. However, this method still does not account for value sensitivities or Hessian information.

Calibration-based scale selection evaluates candidate scales and minimizes a block objective that can include calibration activations or Hessian-derived sensitivity weights. This makes

the selected scale closer to the effect of quantization on the model's final output, rather than only to the value distribution inside the block. To the best of our knowledge, we are the first to research this direction.

3 Mathematical Analysis

In this chapter we introduce a formal mathematical model to study optimality of various scaling strategies. This study will lay the groundwork for optimal scaling strategy and proposed experiments in the next chapters.

This chapter is organized as follows. First, we define an objective function optimized by the shared scale. Second, we study the properties of this objective function for various numerical formats. Then, we determine the optimality of the existing scaling strategies in a hessian-outlier scenario. Consequently, we discuss how certain transformations affect our mathematical analysis and the objective. Finally, we discuss the data-free objective and its tradeoffs.

3.1 Scale Objective Function

Quantization is one of the transformations of the model's weights. For simplicity of this analysis, we consider weight-only quantization.

The following formula approximates the change in model loss caused by a quantization-induced weight update ΔW [13]:

$$\Delta L = \frac{1}{2}\Delta W H \Delta W^T, \tag{3.1}$$

where H denotes a Hessian matrix that determines the sensitivity of the weight change. Some weights are more sensitive to quantization than others, and the Hessian matrix allows us to identify the sensitive weights.

In the context of scale selection, the per-element squared quantization error $\delta_q(w,s)$ is determined by the weight itself and by the selected scale value:

$$\delta_q(w,s) = \big(w - Q_s(w)\big)^2 \tag{3.2}$$

where

$$Q_s(w) = s \cdot \text{round}_{\mathcal{C}}\big(\text{clip}_{\mathcal{C}}(w/s)\big). \tag{3.3}$$

Here $\text{clip}_{\mathcal{C}}$ clips values outside the representable range of the codebook $\mathcal{C}$, while $\text{round}_{\mathcal{C}}$ maps to the nearest codebook value.

The selection of a common scale value s affects the quantization error of W, which, therefore, affects the model loss according to Equation 3.1. This insight leads us to the formula of the scale objective.

Definition 3.1 (Scale objective)**.** Consider a block of B weights $w_1, \ldots, w_B$. Let $h_1, \ldots, h_B$ denote the corresponding diagonal Hessian entries, $h_k = \frac{1}{2}\|X_{:,k}\|^2$. For a scale value $s > 0$ and codebook $\mathcal{C}$, the block objective is

$$\mathcal{L}^{\text{obj}}(s) = \sum_{k=1}^{B} h_k \delta_q(w_k, s), \tag{3.4}$$

In this formula, apart from the quantization error value itself, we take into account the sensitivity of each weight in the block. To minimize this objective, the scale selection should preserve more sensitive weights, while allowing larger errors on less sensitive weights.

Remark 3.2. In block-wise numerical formats, the set $\mathcal{S}$ of possible values for the common scale is restricted by the underlying datatype. For example, for the power-of-two scale datatype used by MX formats, the scale set is

$$\mathcal{S}_{\text{E8M0}} = \{2^e : e \in \mathbb{Z},\ e_{\min} \le e \le e_{\max}\}, \tag{3.5}$$

with the OCP exponent range $e_{\min} = -127$ and $e_{\max} = 127$, apart from the reserved NaN encoding [10].

Using the scale objective formula, we can define an optimal scale.

Definition 3.3 (Optimal scale)**.** For a supported scale set $\mathcal{S}$, the optimal scale value for a block is

$$s^*_{\mathcal{S}} \in \operatorname*{arg\,min}_{s \in \mathcal{S}} \mathcal{L}^{\text{obj}}(s). \tag{3.6}$$

In order to develop an algorithm for selecting the optimal scale value, we need to study the properties of the objective function $\mathcal{L}^{\text{obj}}$.

3.2 Discrete Unimodality Analysis of $\mathcal{L}^{\text{obj}}$

First, we define *discrete unimodality*.

Definition 3.4. Let $\mathcal{S}$ be a finite totally ordered set. A discrete function $f : \mathcal{S} \to \mathbb{R}$ is discretely

unimodal if the set of global minimizers

$$\mathcal{M} = \arg\min_{s \in \mathcal{S}} f(s) \tag{3.7}$$

is non-empty and contiguous in the order on $\mathcal{S}$, and if f is non-increasing before $\mathcal{M}$ and non-decreasing after $\mathcal{M}$. Equivalently, the sequence has one valley.

In other words, as the scale value increases, the objective first non-increases, may stay flat at the bottom of the valley, and then non-decreases. Flat regions are allowed because exact quantization can make several adjacent scale values equally optimal.

Understanding discrete unimodality, or lack thereof, determines whether local search can reliably find the optimal scale and shapes the design of scale-selection algorithms.

3.2.1 MXFP

We study unimodality of the scale objective for MXFP. Without loss of generality, we consider the MXFP4 format. The same analysis applies to MXFP8.

We begin with a structural property of the FP4 data codebook used by the MXFP4 format.

Lemma 3.5 (MXFP4 scaled-codebook invariance under scale halving)**.** *For any scale value $s > 0$, the scaled FP4 data codebooks $s\mathcal{C}^{\text{fp}}$ and $(s/2)\mathcal{C}^{\text{fp}}$ share the values $\{0, \pm 0.5s, \pm s, \pm 1.5s, \pm 2s, \pm 3s\}$. For any w with $|w| \in [s, 3s]$, the nearest quantization point is the same under both scale values, so $\delta_q(w, s) = \delta_q(w, s/2)$.*

Proof. At scale value s, the scaled FP4 data codebook values are

$$s \cdot \{0, \pm 0.5, \pm 1, \pm 1.5, \pm 2, \pm 3, \pm 4, \pm 6\}. \tag{3.8}$$

At scale value $s/2$, they are

$$s \cdot \{0, \pm 0.25, \pm 0.5, \pm 0.75, \pm 1, \pm 1.5, \pm 2, \pm 3\}. \tag{3.9}$$

In $[s, 3s]$, both scaled FP4 data codebooks contain the same quantization points $\{s, 1.5s, 2s, 3s\}$. □

Remark 3.6. This lemma partitions the per-element objective into three phases as the power-of-two scale value $s = 2^e$ increases.

1. When the scale value is too small, the element clips and increasing s reduces the clipping error.

2. When the scale value is in the stable middle region, doubling s keeps the same relevant quantization points around the element, so the per-element squared quantization error is unchanged.

3. When the scale value is too large, the element falls into the low-magnitude part of the scaled codebook, and increasing s coarsens the available quantization points, so the error goes up.

We will use this result to prove the per-element unimodality of MXFP.

Theorem 3.7 (MXFP per-element unimodality)**.** *For any $w \neq 0$, the per-element MXFP4 objective*

$$\ell_w(e) = \delta_q(w, 2^e) = \left(w - Q_{2^e}(w)\right)^2 \tag{3.10}$$

is unimodal as a function of the power-of-two scale e.

Proof. Without loss of generality, assume $w > 0$.

For the proof, it is enough to compare two adjacent power-of-two scale values, s and $2s$. We will break our analysis into the following cases:

Case one: $w > 6s$. The scale value s is too small and clips to $6s$. Doubling the scale value does not increase the error. If $w > 12s$, the larger scale value also clips, but to $12s$, which is closer to w. If $6s < w \leq 12s$, the old clipped value $6s$ remains available at scale value $2s$ as the quantization point $3 \cdot 2s$, so nearest-neighbor rounding at scale value $2s$ has error no larger than before.

Case two: $2s \leq w \leq 6s$. The element is in the stable middle region. By Lemma 3.5 applied to scale values s and $2s$, both scaled FP4 data codebooks contain the same relevant quantization points in this interval. Therefore

$$\delta_q(w, 2s) = \delta_q(w, s). \tag{3.11}$$

Case three: $w < 2s$. The scale value is too large. In this region, the useful quantization points at scale value $2s$ are a subset of the quantization points available at scale value s:

$$\{0,\ s,\ 2s,\ 3s\} \subset \{0,\ \tfrac{1}{2}s,\ s,\ \tfrac{3}{2}s,\ 2s,\ 3s\}. \tag{3.12}$$

Points above $3s$ are farther from w than the shared point $2s$, making them irrelevant in this case. Hence doubling the scale value only removes relevant quantization points, and

$$\delta_q(w, 2s) \geq \delta_q(w, s). \tag{3.13}$$

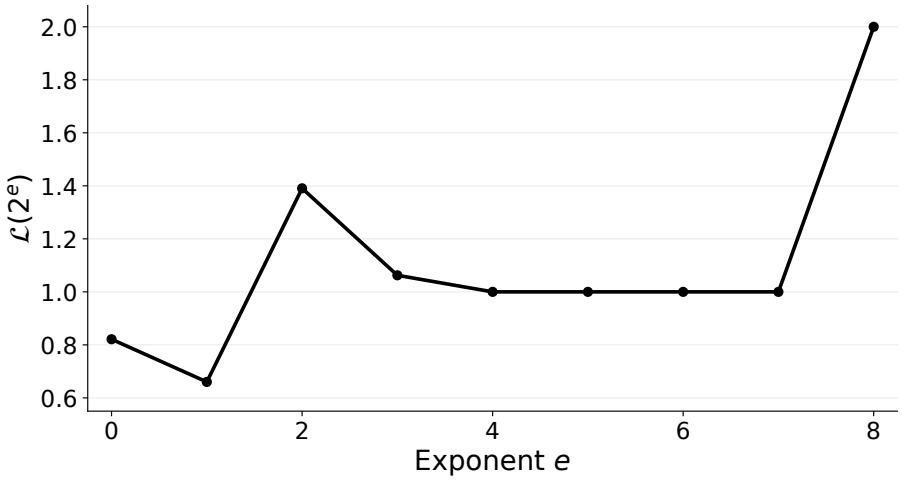


Figure 3.1: MXFP4 block objective for the two-magnitude-cluster counterexample with $w_1 = 1$, $w_2 = 64$, $h_1 = 1$, and $h_2 = 1/64^2$. The objective decreases, increases, and then decreases again, so it has two separated valleys and is not unimodal.

Thus, as the scale increases, the per-element objective first non-increases, then stays constant, and then non-decreases. This makes the sequence unimodal under the definition used in this chapter. □

However, this per-element result does not extend to a whole block. Changing the power-of-two scale only moves the whole scaled FP4 data codebook left or right on the log-magnitude axis. If a block contains two magnitude clusters, the scaled FP4 data codebook can first align with one cluster and then, several scale-exponent steps later, align with the other cluster.

Theorem 3.8 (MXFP4 block-level non-unimodality)**.** *The MXFP4 block objective $\mathcal{L}^{\text{obj}}(2^e)$ is not unimodal in general as a function of the power-of-two scale e.*

Proof. It is enough to give one counterexample. Consider two weights with the following Hessian values

$$w_1 = 1, \qquad h_1 = 1, \qquad w_2 = R, \qquad h_2 = \frac{1}{R^2}, \tag{3.14}$$

where $R = 2^m$ is large enough; for example, take $m = 6$, so $R = 64$. Figure 3.1 shows this concrete instance.

At scale $e = 0$ and $e = 1$, the first weight is represented exactly. The second weight is still clipped, but the clipped value moves from 6 to 12, so $\mathcal{L}^{\text{obj}}(2^1) < \mathcal{L}^{\text{obj}}(2^0)$. At scale $e = 2$, the first weight lies exactly between 0 and 2, so its per-element squared quantization error is 1 under either tie-breaking choice. The second weight is still clipped, now to 24. Therefore

$$\mathcal{L}^{\text{obj}}(2^2) - \mathcal{L}^{\text{obj}}(2^1) = 1 - \frac{1}{R^2}\left((R-12)^2 - (R-24)^2\right) > 0 \tag{3.15}$$

Now compare the scale $e = m-3$ and $e = m-2$. At $e = m-3$, the scale value is $R/8$, so the second weight clips to $6R/8 = 3R/4$ and makes the following weighted contribution to the objective:

$$\frac{1}{R^2}\left(R - \frac{3R}{4}\right)^2 = \frac{1}{16}. \tag{3.16}$$

The first weight is much smaller than the smallest positive quantization point and rounds to zero, contributing 1 to the objective. Hence $\mathcal{L}^{\text{obj}}(2^{m-3}) = 1 + 1/16$. At $e = m-2$, the scale value is $R/4$, so the second weight is represented exactly as $4 \cdot R/4$, while the first weight still contributes 1. Hence $\mathcal{L}^{\text{obj}}(2^{m-2}) = 1$.

The second weight remains represented exactly up to $e = m+1$, where $R = 0.5 \cdot 2^{m+1}$, and the first weight still contributes 1 to the objective. At $e = m+2$, the second weight lies halfway between 0 and $2R$, so its weighted contribution is 1, and the objective increases again. Thus, around the large-magnitude cluster, the block objective also decreases and then increases.

The objective therefore increases after the first low-error region and decreases again before the second low-error region. This creates two separated valleys, so the block objective is not unimodal in general. □

This construction needs the magnitude clusters to be separated by more than the width of the nonzero FP4 data codebook. Since the positive scaled FP4 data codebook spans $[0.5s, 6s]$, one scale value can cover magnitudes within only about a 12× range. Once the clusters are separated by more than roughly 12× (about 3.6 scale-exponent steps), their useful scale regions become disjoint; using a 64× separation, as in the example above, makes the two low-error regions clearly separated.

Remark 3.9. This counterexample is not representative of many realistic blocks. In realistic blocks, the objective is often closer to unimodal because one Hessian-dominant element sets the main trend of $\mathcal{L}^{\text{obj}}(2^e)$, while the remaining elements contribute only smaller perturbations. By Theorem 3.7, the per-element objective of this dominant element is unimodal. Therefore, when its Hessian weight is sufficiently larger than the others, the block objective is governed by this single-valley trend and often behaves approximately as a unimodal sequence for realistic blocks. We will study the hessian-dominant scenario later in this chapter.

3.2.2 MXINT

Without loss of generality, we consider the MXINT4 format. The same analysis applies to MXINT8.

We use the MXINT4 format to denote a power-of-two scale together with a uniform signed INT4 data codebook

$$\mathcal{C}^{\text{int}} = \{-7, -6, \ldots, 0, \ldots, 7\}. \tag{3.17}$$

In contrast to MXFP4, doubling the MXINT4 scale value does not preserve a middle part

of the scaled INT4 data codebook. Instead, the grid spacing doubles and the representable interval expands from $[-7s, 7s]$ to $[-14s, 14s]$. This still gives a single-element unimodality result: as the scale value grows, the clipping error first decreases, then the element falls into the quantization grid, and finally the coarser spacing increases the rounding error.

Theorem 3.10 (MXINT4 per-element unimodality)**.** *For any $w \neq 0$, the per-element MXINT4 objective*

$$\ell_w(e) = \delta_q(w, 2^e) = \left(w - Q_{2^e}(w)\right)^2 \tag{3.18}$$

is unimodal as a function of the power-of-two scale e.

Proof. By symmetry, assume $w > 0$. Compare adjacent scale values s and $2s$. At scale value s, the positive scaled INT4 data codebook is $\{0, s, 2s, \ldots, 7s\}$. At scale value $2s$, the positive scaled INT4 data codebook is $\{0, 2s, 4s, \ldots, 14s\}$.

We break the comparison into three cases:

Case one: $w \geq 7.5s$**.** The scale value s is too small. Doubling the scale value cannot increase the error. For $7.5s \leq w \leq 9s$, the point $8s$ is at least as close as the clipped point $7s$. For larger w, the nearest even quantization point, or the clipped endpoint $14s$ when $w > 14s$, is also no farther than $7s$. Therefore

$$\delta_q(w, 2s) \leq \delta_q(w, s). \tag{3.19}$$

Case two: $7s < w < 7.5s$**.** The element is beyond the clipping threshold at scale value s. Scale value s clips to $7s$, while scale value $2s$ rounds to $8s$. Since $8s - w > w - 7s$, the larger scale value has larger error, so

$$\delta_q(w, 2s) > \delta_q(w, s). \tag{3.20}$$

Case three: $w \leq 7s$**.** The scale value s is already large enough to avoid clipping. Doubling the scale value removes the odd quantization points. The extra point $8s$ cannot be closer to w than the nearest point in $\{0, s, 2s, \ldots, 7s\}$ because $w \leq 7s$. Hence doubling the scale value cannot reduce the error, and

$$\delta_q(w, 2s) \geq \delta_q(w, s). \tag{3.21}$$

As e increases, the scale value $s = 2^e$ increases monotonically. The adjacent change is non-positive while $w \geq 7.5s$ and non-negative once $w < 7.5s$. Thus the per-element objective first non-increases and then non-decreases, which makes the sequence unimodal under the definition used in this chapter. □

As for MXFP4, the per-element result does not imply block-level unimodality. Two well-separated magnitude clusters can prefer two different scale regions, and the shared scale value moves between these regions exponentially as e changes.

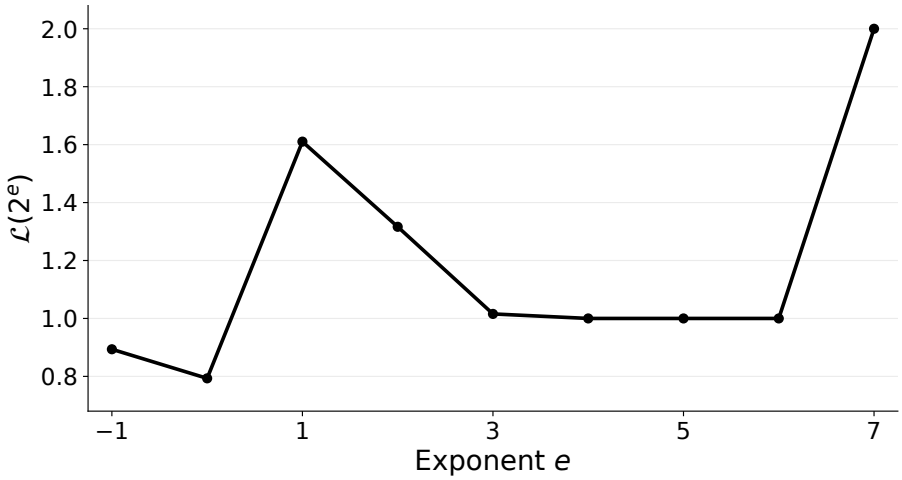


Figure 3.2: MXINT4 block objective for the two-magnitude-cluster counterexample with $w_1 = 1$, $w_2 = 64$, $h_1 = 1$, and $h_2 = 1/64^2$. The objective decreases, increases, and then decreases again, so it has two separated valleys and is not unimodal.

Theorem 3.11 (MXINT4 block-level non-unimodality)**.** *The MXINT4 block objective $\mathcal{L}^{\text{obj}}(2^e)$ is not unimodal in general as a function of the power-of-two scale e.*

Proof. It is enough to give one counterexample. Consider two positive weights with Hessian weights

$$w_1 = 1, \qquad h_1 = 1, \qquad w_2 = R, \qquad h_2 = \frac{1}{R^2}, \tag{3.22}$$

and take $R = 64$. Figure 3.2 shows this concrete instance.

First consider the small-magnitude cluster. At scale $e = -1$, the scale value is $s = 1/2$. The first weight is represented exactly, while the second weight clips to $7/2$, so

$$\mathcal{L}^{\text{obj}}(2^{-1}) = \frac{1}{64^2}\left(64 - \frac{7}{2}\right)^2. \tag{3.23}$$

At scale $e = 0$, the first weight is still represented exactly and the second weight clips to 7, giving

$$\mathcal{L}^{\text{obj}}(2^0) = \frac{1}{64^2}(64-7)^2 < \mathcal{L}^{\text{obj}}(2^{-1}). \tag{3.24}$$

At scale $e = 1$, the first weight lies halfway between 0 and 2, so its per-element squared quantization error is 1 under either tie-breaking convention. The second weight clips to 14, and therefore

$$\mathcal{L}^{\text{obj}}(2^1) - \mathcal{L}^{\text{obj}}(2^0) = 1 + \frac{(64-14)^2 - (64-7)^2}{64^2} > 0. \tag{3.25}$$

Thus, around the small-magnitude cluster, the block objective decreases and then increases.

Now consider the large-magnitude cluster. At scale $e = 3$, the scale value is $s = 8$. The first

weight rounds to zero and contributes 1 to the objective, while the second weight clips to 56, so

$$\mathcal{L}^{\text{obj}}(2^3) = 1 + \frac{(64-56)^2}{64^2} = 1 + \frac{1}{64}. \tag{3.26}$$

At scale $e = 4$, the scale value is $s = 16$, so the second weight is represented exactly as $4s$, while the first weight still contributes 1 to the objective. Hence $\mathcal{L}^{\text{obj}}(2^4) = 1$. The second weight remains represented exactly until $e = 6$. At scale $e = 7$, it lies halfway between 0 and 128, so its weighted contribution is 1, while the first weight still contributes 1. Hence $\mathcal{L}^{\text{obj}}(2^7) = 2$.

The objective therefore increases after the first low-error region and decreases again before the second low-error region. This creates two separated valleys, so the MXINT4 block objective is not unimodal in general. □

Remark 3.12. This counterexample is again intentionally adversarial and is not representative of many realistic blocks. In realistic blocks, the objective is often closer to unimodal because one Hessian-dominant element sets the main trend of $\mathcal{L}^{\text{obj}}(2^e)$, while the remaining elements contribute only smaller perturbations. By Theorem 3.10, the per-element objective of this dominant element is unimodal. Therefore, when its Hessian weight is sufficiently larger than the others, the block objective is governed by this single-valley trend and often behaves approximately as a unimodal sequence for realistic blocks.

3.2.3 Floating-Point Scale

The previous two subsections considered E8M0 scale values, where consecutive scale values differ by an exact factor of two. We now consider floating-point scale formats such as E4M3, E5M3, and E4M4. These scale sets add intermediate significand values inside each power-of-two binade. This extra density is useful for reducing quantization error, but it also destroys the simple unimodality structure that held for a single element under E8M0 scale values.

For concreteness, consider the following E4M3 scale values inside the binade $[1,2]$:

$$\mathcal{S}_{\text{E4M3}} \cap [1,2] = \left\{1, \frac{9}{8}, \frac{10}{8}, \frac{11}{8}, \frac{12}{8}, \frac{13}{8}, \frac{14}{8}, \frac{15}{8}, 2\right\}. \tag{3.27}$$

Theorem 3.13 (E4M3 single-element non-unimodality)**.** *With the E4M3 scale set, the single-element objective*

$$\ell_w(s) = \delta_q(w,s) = \big(w - Q_s(w)\big)^2 \tag{3.28}$$

is not unimodal in general as a function of the ordered scale values $s \in \mathcal{S}_{\text{E4M3}}$.

Proof. It is enough to give one counterexample. Use the FP4 data codebook

$$\mathcal{C}^{\text{fp}} = \{0, \pm 0.5, \pm 1, \pm 1.5, \pm 2, \pm 3, \pm 4, \pm 6\} \tag{3.29}$$

and a single positive weight $w = 1$.

At scale value $s = 1$, the quantization point $1 \cdot s$ represents the weight exactly, so

$$\ell_w(1) = 0. \tag{3.30}$$

At the next power-of-two scale value $s = 2$, the quantization point $0.5 \cdot s$ also represents the same weight exactly, so

$$\ell_w(2) = 0. \tag{3.31}$$

However, the intermediate E4M3 scale value $s = 11/8$ gives the two closest relevant quantization points

$$0.5 \cdot s = \frac{11}{16}, \qquad 1.0 \cdot s = \frac{11}{8}. \tag{3.32}$$

The nearest one to the weight w is $11/16$, and therefore

$$\ell_w\left(\frac{11}{8}\right) = \left(1 - \frac{11}{16}\right)^2 = \left(\frac{5}{16}\right)^2 > 0. \tag{3.33}$$

Thus the resulting objective has zero value at $s = 1$, positive value at an intermediate scale value $s = 11/8$, and zero value again at $s = 2$. Its minima are separated by a strictly worse scale value. Therefore, even the single-element objective is not unimodal for floating-point scale sets. □

The reason is specific to the interaction between a floating-point scale format and the FP4 data codebook. Increasing s inside one binade moves all quantization points, and the set of new quantization points is neither a subset nor a superset of the old quantization points. Therefore, the quantization accuracy can fluctuate as the codebook changes with the scale.

Consequently, for E4M3, E5M3, and E4M4 scale formats, unimodality cannot be assumed even before summing over a block. Block-level objectives inherit this issue and can additionally contain the non-unimodality from separated magnitude clusters shown for MXFP4 and MXINT4 above. Therefore, search for the optimal scale value over floating-point scale sets generally needs exhaustive search over the supported scale values. However, the search can be optimized, which we will discuss later in this report.

3.2.4 Discussion

The main conclusion of this section is that the block scale objective is not unimodal in general for any of the scale types considered above. For MXFP and MXINT formats with power-of-two scales, non-unimodality appears at the block level: different magnitude clusters can require different scales for precise quantization, and a single shared scale must trade them off. For floating-point scale formats, the situation is even less structured, because non-unimodality can appear for a single element before any block-level interaction is introduced.

Nevertheless, the distinction between these cases is important. For MXFP and MXINT, the

single-element objective remains unimodal as a function of the power-of-two scale. This means that, when one element dominates the scale objective, the block objective is often shaped by the smooth single-element trend of that dominant coordinate, with the remaining elements acting as smaller perturbations. This is precisely the Hessian-outlier setting studied in the next section. In that regime, the objective is still not guaranteed to be globally unimodal, but it can be smoother and closer to a single-valley landscape than the worst-case block counterexamples suggest.

Floating-point scale formats do not have the same guarantee. Since their single-element objective can already contain separated minima, a Hessian-dominant coordinate does not necessarily impose a smooth trend on the block objective. This makes local structure less reliable for search.

3.3 Optimality of Max-Element Clipping in the Hessian-Outlier Scenario

Current scale-selection strategies, such as `absmax`, `max`, and `midmax`, select the scale based on the maximum magnitude in the block. While `absmax` tries to avoid clipping the maximum element, other strategies take a smarter approach. In this analysis, we show that preserving the maximum element in range is not always optimal. We also provide a specific criterion to determine which strategy is better in which case. We perform the analysis on two numerical formats: MXINT4 and MXFP4. In both cases, we compare two power-of-two scale values:

- s_1: the minimal no-clipping power-of-two scale value, $s_1 = 2^{\lceil \log_2(\max_k |w_k|/c_{\max}) \rceil}$, where $c_{\max} = \max_{c \in \mathcal{C}} |c|$.
- $s_0 = s_1/2$: the next smaller power-of-two scale value.

To simplify our analysis, we make the following assumptions.

Assumption 3.14 (Single-element clipping)**.** Only one element, indexed $y = \arg\max_k |w_k|$, exceeds the representable range at scale s_0. All other elements satisfy $|w_k|/s_0 \le \max(\mathcal{C})$.

Assumption 3.15 (Single Hessian Outlier)**.** We distinguish a *Hessian outlier* at index $x \neq y$, whose Hessian value is much larger than the average Hessian value in the block. In other words, $h_x \gg h_m$, where

$$h_m = \frac{1}{B-2} \sum_{k \neq x,\, k \neq y} h_k. \tag{3.34}$$

Of course, the analysis can be extended to an arbitrary number of Hessian and weight outliers. However, from the empirical data we observed that the number of weight and Hessian outliers usually does not exceed 1 or 2 elements per block.

In the following subsections we again study 4-bit formats, but the analysis extends to arbitrary bitwidths.

3.3.1 MXFP

By Lemma 3.5 (with $s = s_1$), the scaled FP4 data codebooks at s_1 and s_0 share the same quantization points in $[s_1, 3s_1]$. This partitions the block into three regions:

$$\begin{aligned} U &= \{k : |w_k| < s_1\} && \text{(finer-scale region — lower error may be possible at } s_0\text{)}, \\ M &= \{k : s_1 \le |w_k| \le 3s_1\} && \text{(invariant region — identical quantization points)}, \\ O &= \{k : |w_k| > 3s_1\} && \text{(clipped region — clipped at } s_0\text{)}. \end{aligned} \tag{3.35}$$

Theorem 3.16 (MXFP4 exact clipping criterion)**.** *Under Assumption 3.14, let $\varepsilon_c^y = \delta_q(w_y, s_0)$ denote the squared clipping error for the maximum element y. The clipping scale value s_0 achieves a lower objective value than s_1 if and only if*

$$\underbrace{h_y\left[\varepsilon_c^y - \delta_q(w_y, s_1)\right]}_{\textit{clipping penalty}} < \underbrace{\sum_{\substack{k \neq y \\ |w_k| < s_1}} h_k\left[\delta_q(w_k, s_1) - \delta_q(w_k, s_0)\right]}_{\textit{finer-scale compensation}}. \tag{3.36}$$

Proof. By Lemma 3.5, elements with $s_1 \le |w_k| \le 3s_1$ satisfy $\delta_q(w_k, s_1) = \delta_q(w_k, s_0)$ and cancel in $\mathcal{L}^{\text{obj}}(s_0) - \mathcal{L}^{\text{obj}}(s_1)$. Under Assumption 3.14, the only clipped element at s_0 is y, so the remaining terms are the clipped maximum element and the elements with $|w_k| < s_1$:

$$\mathcal{L}^{\text{obj}}(s_0) - \mathcal{L}^{\text{obj}}(s_1) = h_y\left[\varepsilon_c^y - \delta_q(w_y, s_1)\right] + \sum_{\substack{k \neq y \\ |w_k| < s_1}} h_k\left[\delta_q(w_k, s_0) - \delta_q(w_k, s_1)\right]. \tag{3.37}$$

Setting $\mathcal{L}^{\text{obj}}(s_0) < \mathcal{L}^{\text{obj}}(s_1)$ and moving the finer-scale terms to the right-hand side gives (3.36). Both sides are nonnegative. The clipping scale value wins only when the finer-scale-region improvement strictly exceeds the maximum-element clipping penalty. □

The Hessian outlier x affects the MXFP4 criterion only if it lies in U and actually benefits from the finer spacing at s_0. If $x \in M$, its contribution cancels completely.

3.3.2 MXINT

We use the MXINT4 format to denote a power-of-two scale together with a uniformly spaced INT4 data codebook

$$\mathcal{C}^{\text{int}} = \{-7, -6, \ldots, 0, \ldots, 7\}. \tag{3.38}$$

Under scale value s, the expected per-element squared quantization error under the high-resolution approximation is

$$\mathbb{E}\big[\delta_q(w_k, s)\big] = \frac{s^2}{12} =: \delta(s). \tag{3.39}$$

Theorem 3.17 (MXINT approximate clipping criterion)**.** *Under Assumption 3.14 and the high-resolution approximation for non-clipped elements, let $\delta = s_1^2/12$, let $d_y(s_1) = \delta_q(w_y, s_1)$, and let $\varepsilon_c^y = \delta_q(w_y, s_0)$ denote the squared clipping error for element y. If $h_m > 0$, then s_0 achieves a lower expected objective value than s_1 if and only if*

$$\frac{h_x}{h_m} > \frac{4h_y}{3\delta h_m}\big(\varepsilon_c^y - d_y(s_1)\big) - (B-2). \tag{3.40}$$

If additionally $h_y \approx h_m$ and $d_y(s_1) \approx \delta$, this reduces to the simpler approximate threshold $\frac{h_x}{h_m} > \frac{4}{3}(\frac{\varepsilon_c^y}{\delta} - 1) - (B-2)$.

Proof. At the absmax scale value:

$$\mathbb{E}\big[\mathcal{L}^{\text{obj}}(s_1)\big] = h_y d_y(s_1) + \delta\big(h_x + (B-2)\,h_m\big). \tag{3.41}$$

At the clipping scale value $s_0 = s_1/2$, non-clipped elements have MSE $\delta/4$:

$$\mathbb{E}\big[\mathcal{L}^{\text{obj}}(s_0)\big] = h_y\,\varepsilon_c^y + \frac{\delta}{4}\big(h_x + (B-2)\,h_m\big). \tag{3.42}$$

Setting $\mathbb{E}[\mathcal{L}^{\text{obj}}(s_0)] < \mathbb{E}[\mathcal{L}^{\text{obj}}(s_1)]$:

$$h_y\big(\varepsilon_c^y - d_y(s_1)\big) < \frac{3\delta}{4}\big(h_x + (B-2)\,h_m\big). \tag{3.43}$$

Dividing by $h_m \cdot 3\delta/4$:

$$\frac{4h_y}{3\delta h_m}\big(\varepsilon_c^y - d_y(s_1)\big) < \frac{h_x}{h_m} + (B-2), \tag{3.44}$$

which rearranges to (3.40). □ □

Remark 3.18. For MXINT, every non-clipped element benefits from the halved scale value (MSE reduces by 4×). For MXFP4, only elements in U benefit while those in M see no change.

3.3.3 Floating-Point Scale

The same Hessian-outlier intuition also applies to floating-point scale formats, although the analysis is less clean than for power-of-two scales. Floating-point scale sets contain many intermediate values within each power-of-two binade, so the scale can be tuned more precisely to the value distribution inside a block. This additional flexibility makes it harder to obtain a simple clipping criterion like the ones above, because the optimal scale is not restricted to choosing between two adjacent powers of two.

In the presence of a strong Hessian outlier, this flexibility changes the expected behavior of the optimum. Since the objective heavily penalizes quantization error on the most sensitive weight, the optimal floating-point scale will tend to choose a value that quantizes this weight as accurately as possible, provided the resulting errors on the remaining block elements are not too large. In other words, floating-point scales can tailor the shared scale more closely to the most important coordinate, while power-of-two scales can only move in coarse steps.

3.4 Effect of the Algorithm on the Scale Objective

3.4.1 Micro-Rotations

We now analyze how a micro-rotation changes the scale objective. Let $R \in \mathbb{R}^{d\times d}$ denote the normalized Hadamard matrix used by the micro-rotation, so that $R^T R = I$. For a linear layer with activation matrix X and weight vector w, the unrotated output is Xw. The equivalent micro-rotated computation is

$$Xw = (XR)(R^T w). \tag{3.45}$$

Thus the rotated activations and weights are

$$X' = XR, \qquad w' = R^T w. \tag{3.46}$$

The proxy Hessian for weight perturbations is

$$H = \frac{1}{2} X^T X, \tag{3.47}$$

and after the micro-rotation it becomes

$$H' = \frac{1}{2}(X')^T X' = \frac{1}{2}(XR)^T (XR) = R^T \cdot \frac{1}{2} X^T X \cdot R = R^T H R. \tag{3.48}$$

Therefore the diagonal Hessian entries used in the scale objective transform as

$$h'_j = H'_{jj} = r_j^T H r_j = \frac{1}{2}\|X r_j\|_2^2, \tag{3.49}$$

where r_j is the j-th column of R.

The full second-order objective is invariant under this change of coordinates:

$$\Delta w^T H \Delta w = (\Delta w')^T H' \Delta w', \qquad \Delta w = R \Delta w'. \tag{3.50}$$

However, the diagonal approximation used by the block scale objective is not invariant. In general,

$$\sum_k h_k \Delta w_k^2 \neq \sum_k h'_k (\Delta w'_k)^2. \tag{3.51}$$

Thus a micro-rotation changes the scale-selection problem by changing both the quantized values w_k and the Hessian weights h_k.

For a normalized Hadamard matrix, each entry satisfies

$$R_{ij}^2 = \frac{1}{d}. \tag{3.52}$$

Expanding the rotated diagonal Hessian gives

$$h'_j = \frac{1}{d}\sum_{a=1}^{d} h_a + \sum_{a\neq b} R_{aj}R_{bj}H_{ab}. \tag{3.53}$$

If the off-diagonal correlations of H are small, the second term is small and all rotated diagonal Hessian entries concentrate near the mean Hessian value. With randomized Hadamard signs, this holds in expectation:

$$\mathbb{E}[h'_j] = \frac{\mathrm{tr}(H)}{d} = \frac{1}{d}\sum_{a=1}^{d} h_a. \tag{3.54}$$

Hence a micro-rotation tends to spread Hessian sensitivity across coordinates and reduce diagonal Hessian outliers.

Consequently, the rotated scale objective

$$\mathcal{L}'_{\mathrm{obj}}(s) = \sum_{k=1}^{B} h'_k \delta_q(w'_k, s) \tag{3.55}$$

is often closer to an unweighted quantization-error objective than the unrotated objective. This weakens the single-Hessian-outlier scenario analyzed earlier, because the sensitivity of one dominant coordinate is distributed across many rotated coordinates. The non-unimodality results for block scale selection still hold in general, but a micro-rotation changes the empirical distribution of weights and Hessian diagonals in a way that can make the block objective smoother and less dominated by individual coordinates.

3.4.2 GPTQ

GPTQ changes the scale objective differently from a micro-rotation. Under the layerwise approximation used in this analysis, transformations of different layers are treated as independent and the calibration input matrix for the current layer is fixed. With this assumption, GPTQ does not change the Hessian of the current layer. The Hessian depends on the calibration inputs, while GPTQ acts on the layer weights.

The main effect of GPTQ is therefore on the weights seen by the scale selector. GPTQ quantizes weights sequentially and, after each quantization decision, updates the remaining unquantized weights to compensate for the induced output error. When a block is reached, its weights may already include compensation updates caused by earlier quantization errors. Therefore,

the block objective $\mathcal{L}^{\text{obj}}(s)$ is applied to the current compensated weights, not necessarily to the original full-precision weights.

This makes the scale-selection problem path-dependent. The weights in a block depend on previous GPTQ updates, and those updates depend on previous quantization errors and scale choices. Thus, even though the local form of the objective remains the same, the values inserted into the objective are determined by the history of the GPTQ cycle.

The fixed-Hessian statement is exact only for this fixed-input layerwise proxy. For the true model loss, changing weights can slightly change later activations and, therefore, the real Hessian seen by later layers. Accounting for this effect would require repeatedly recomputing Hessian information during quantization, which is substantially more computationally expensive. We therefore keep the fixed-Hessian approximation and account for GPTQ through the compensated weights used at the time of scale selection.

3.5 Data-Free Objective

The objective in Definition 3.1 is Hessian-dependent. It is derived from a second-order approximation to the loss change caused by perturbing the weights, so each quantization error is weighted by the corresponding diagonal Hessian entry. This makes the objective task-aware, but also ties it to weight quantization and to calibration information. In particular, it does not directly give the same type of scale-selection criterion for activations.

A simpler alternative is to remove the Hessian weights and minimize the reconstruction error of the quantized values themselves. For a block of values $z_1, \dots, z_B$, where z can denote either weights or activations, the data-free objective is

$$\mathcal{L}_{\text{df}}(s; z) = \sum_{k=1}^{B} \delta_q(z_k, s) = \sum_{k=1}^{B} \big(z_k - Q_s(z_k)\big)^2. \tag{3.56}$$

The corresponding scale is selected by minimizing this unweighted reconstruction error over the available scale set:

$$s^*_{\text{df}} \in \arg\min_{s \in \mathcal{S}} \mathcal{L}_{\text{df}}(s; z). \tag{3.57}$$

This changes the scale-selection problem from minimizing a loss proxy to minimizing average reconstruction error. The advantage is that the same criterion can be used for weights and activations, and it does not require a Hessian estimate. The cost is that all elements in the block are treated as equally important: a scale that is optimal for $\mathcal{L}_{\text{df}}$ may still spend too much error on a small number of sensitive weights, which the scale objective would explicitly protect.

The data-free objective becomes more plausible when coordinate importance is less uneven. As discussed in Section 3.4.1, a micro-rotation spreads sensitivity across many dimensions and can make the rotated Hessian diagonals closer to each other. In that setting, the unweighted

reconstruction error is a better proxy for the scale objective than it would be in the presence of strong Hessian outliers.

A concurrent work, ScaleSearch, successfully uses this type of quantization-error search to choose block floating-point scales for both weights and activations [33]. However, studying this data-free objective is outside the scope of this work. The analysis in the rest of this report focuses on the scale objective in Definition 3.1.

4 Single-Layer Analysis

This chapter empirically validates the mathematical analysis on a single layer of a large language model. We first examine the shape of the scale-selection objective and show that Hessian outliers often dominate it. We then quantify the gap between the optimal scale and standard max-based scale choices. Finally, we study how the objective changes with the scale type and with the use of micro-rotations.

4.1 Experimental Setup

The single-layer analysis focuses on one layer from Llama-3.1-8B-Instruct [35]. The calibration data for Hessian extraction comes from the C4 training dataset [36], and we use 128 samples. The analyzed layer is `model.layers.10.mlp.down_proj`.

The experiments in this chapter compare two algorithms: RTN and MR-RTN (micro-rotated RTN). RTN applies blockwise quantization directly to the captured weights. MR-RTN first applies a micro-rotation, implemented as a block-diagonal Hadamard transform with the same size as the numerical format block, to both the weights and the activations, and then performs RTN quantization and applies the same scale-selection analysis in the rotated space for the weights.

The evaluated formats are the tile-32 variants MXINT4, MXFP4, E4M3FP4_g, E5M3FP4_g, E5M3FP4, E4M4FP4_g, and E4M4FP4. The scale-selection strategies are `max`, `absmax`, `midmax`, and `optimal`, where `midmax` is only applicable to MXFP4 and `max` is only applicable to MXINT4. The `optimal` strategy is computed by exhaustive search over representable shared scales, minimizing the objective for each block.

The Hessian matrix is estimated from the calibration activations as

$$H = \frac{1}{|D_{\text{calib}}|} \sum_{X \in D_{\text{calib}}} \frac{1}{2} X X^T \tag{4.1}$$

where D_{calib} is the calibration dataset and X denotes the input activation matrix for one

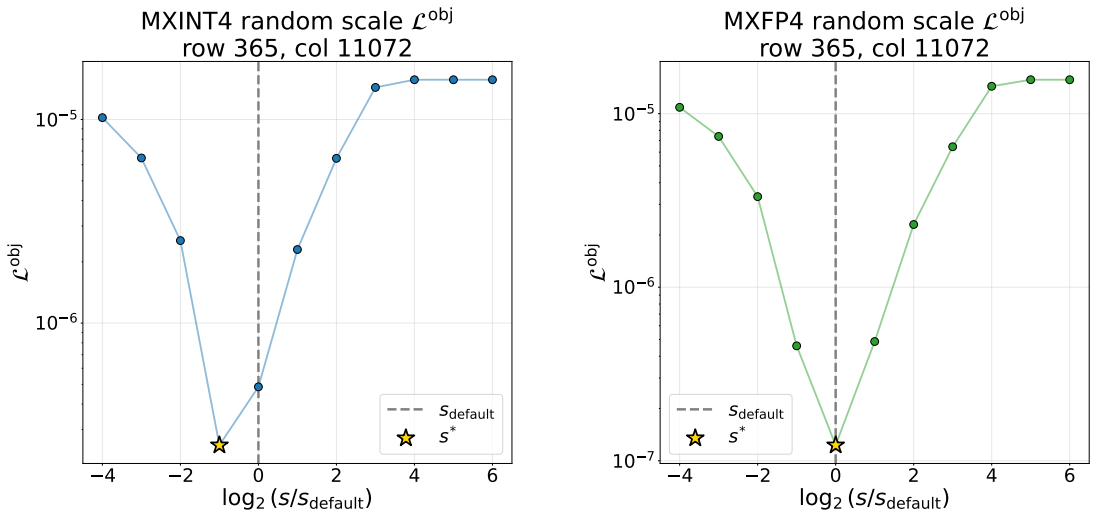


Figure 4.1: Scale objective for a randomly selected block for MXINT4 and MXFP4 under RTN without micro-rotations. Each point corresponds to one representable power-of-two shared scale.

calibration sample.

The experiments were run on AMD Instinct MI210 GPUs.

4.2 Objective Function Unimodality

In this and the following sections, we first assume that no micro-rotations are applied unless explicitly stated otherwise. We study the effect of the micro-rotations later in this chapter.

Based on the mathematical analysis (Remark 3.9, Remark 3.12, and Theorem 3.13), we expect the objective function to be close to unimodal for MXINT and MXFP, but non-unimodal for floating-point scales. Without micro-rotations, the activations contain outliers [9]. These outliers give some weights much larger Hessian values than the rest, so the corresponding single-coordinate terms can dominate the whole objective. Since the terms themselves are unimodal for MXINT and MXFP (Theorem 3.7 and Theorem 3.10), a block is also expected to have an approximately unimodal objective as it is dominated by a Hessian outlier.

To test this hypothesis, we calculate the objective function for all the blocks in the layer and study their unimodality on the real data.

4.2.1 MXINT and MXFP

Figure 4.1 shows the scale objective for a randomly selected block in MXINT4 and MXFP4, evaluated over all representable shared scale values. The x-axis is the logarithmic distance

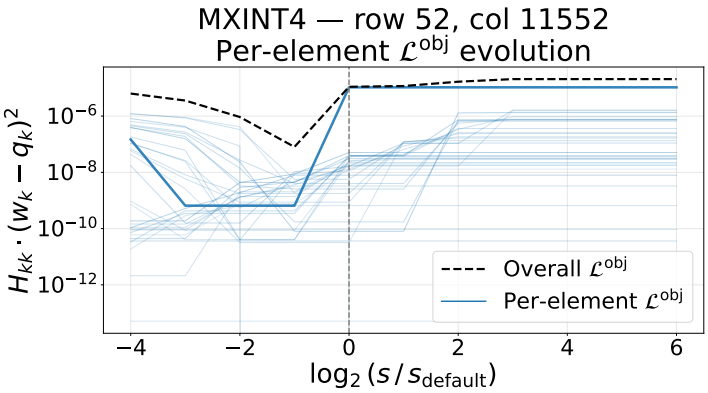


Figure 4.2: Per-element decomposition of the MXINT4 scale objective for one block under RTN without micro-rotations. The black dashed line is the full block objective, and the blue solid lines are the Hessian-weighted per-element contributions.

from the default scale s_{default}, which is selected by the `absmax` strategy. The y-axis is the value of the full block objective. The star marks the optimal scale s^*, which is the global minimum of the objective, while the vertical dashed line marks s_{default} at $x = 0$.

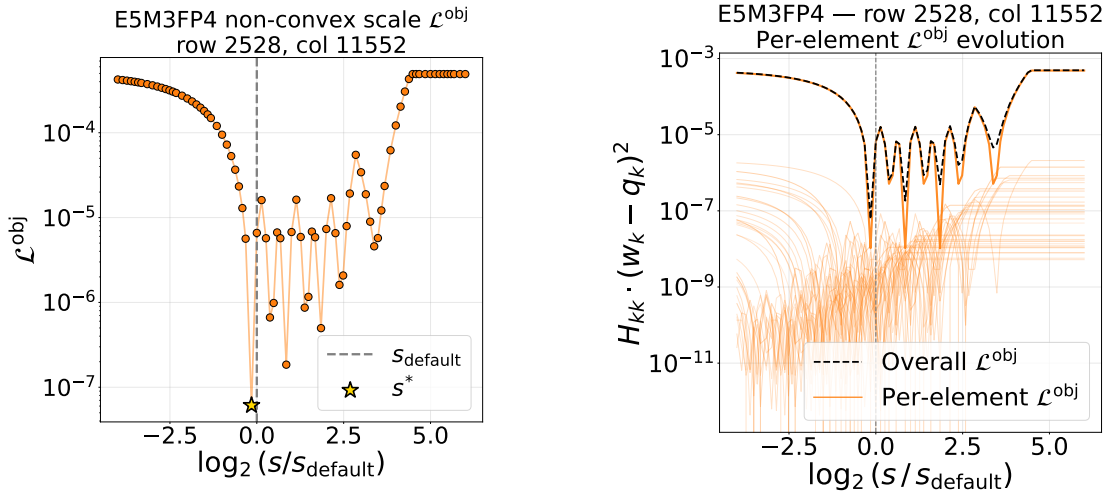
In both cases, the objective appears unimodal. We hypothesize that this behavior is caused by a Hessian-dominant element that determines the overall shape of the block objective. To test this hypothesis, we decompose the scale objective into per-element curves.

Figure 4.2 shows this decomposition for MXINT4. Each curve shows the contribution of one element in the block at scale s to the objective function. The contribution equals $H_{kk} \cdot (w_k - q_k)^2$, where k is the element index. The MXFP4 decomposition is similar, so we omit it for simplicity. The thickness and opacity of each per-element curve are proportional to the corresponding Hessian value H_{kk}, while the black curve represents the full block objective. One dominant curve is visibly thicker than the others and it closely follows the trend of the full objective. This curve corresponds to the Hessian outlier discussed previously, supporting the explanation that Hessian dominance makes the MXINT and MXFP objectives unimodal in realistic blocks.

Our experiment code did not find a case of the objective having more than one minimum for MXFP and MXINT in this particular model and layer. The objective might be non-unimodal in another layer or another model, but this result shows that the practical assumption of objective unimodality is strong.

This empirical property can be used to design a scale-selection algorithm efficiently. For example, instead of searching the whole range of scales, one could perform a local valley search. Later in this chapter we present more empirical properties of the default scale selection and the objective function that make it even easier to find an optimal scale in practice.

We can also see in Figure 4.1 that the default scale does not select the global minimum for



(a) Scale objective.

(b) Per-element decomposition.

Figure 4.3: E5M3FP4 scale objective and per-element decomposition for representative blocks under RTN without micro-rotations. The objective is non-unimodal over representable floating-point shared scales, while the decomposition shows the Hessian-weighted per-element contributions.

MXINT. This also happens for other blocks in MXFP. We analyze these scenarios and their frequency later in this chapter.

4.2.2 Floating-Point Scale

Figure 4.3a shows the scale objective for a randomly selected block in E5M3FP4, evaluated over all representable floating-point shared scale values. As predicted, the objective function is not unimodal and has several local minima. We hypothesize that this behavior comes similarly from the single-element non-unimodality (Theorem 3.13). As in the previous case of MXINT and MXFP, the Hessian outlier dominates the other elements and contributes the most to the block objective.

Figure 4.3b shows the decomposition of the scale objective for E5M3FP4. As in the MXINT and MXFP cases, there is one dominant curve that closely follows the trend of the full objective. The local minima of the full objective occur when the Hessian outlier can be quantized accurately by the numerical format.

This makes the search for the optimal scale for floating-point scales more challenging. Without unimodality, we can no longer rely on a local valley search. However, there is an important insight: the minimum is achieved when the Hessian outlier is quantized accurately by the numerical format. That means that the optimal scale should bring the Hessian outlier as close to the quantization grid as possible. This brings us to a possible optimization for scale selection. The algorithm takes all points on the quantization grid and calculates the scales

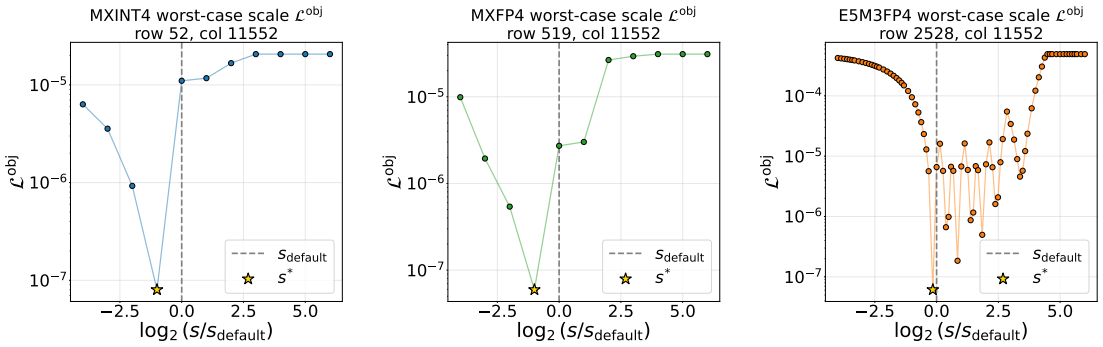


Figure 4.4: Worst-case scale objectives for MXINT4, MXFP4, and E5M3FP4 under RTN without micro-rotations. The default scale is selected with the `absmax` strategy, which avoids clipping the largest-magnitude element in the block.

that bring the Hessian outlier as close as possible to them. As a result, instead of scanning 2^8 scales, the algorithm needs to scan 2^3 scales, as the quantization grid has only 2^3 positive values when the mantissa has 4 bits.

4.3 Max-Based Scale Selection Analysis

As mentioned in the previous section, max-based scale selection does not always result in an optimal choice. In this section, we study this phenomenon more deeply by looking at worst-case scenarios for several shared-scale data types and quantifying the distance between the optimal scale and the selected max-based scale.

4.3.1 Worst-case Scenario

Figure 4.4 depicts the scale objective function for MXINT4, MXFP4 and E5M3FP4. The scale was selected using the `absmax` selection strategy as in the previous experiments.

In all three graphs, the optimal scale yields an objective value that is smaller by up to two orders of magnitude than the default scale. As studied in the mathematical chapter (Section 3.3), we hypothesize that the `absmax` strategy selects the scale very conservatively by avoiding clipping of all elements at the cost of larger quantization error for MXINT and MXFP. For E5M3FP4, it fails to optimize the scale for the Hessian outlier as it only looks at the weight magnitudes, thus yielding a suboptimal result.

Figure 4.5 shows how often `absmax` chooses a suboptimal scale, and how far away the optimal scale is from the chosen max-based scale for MXINT4, MXFP4, and E5M3FP4. As expected, MXFP and MXINT either select an optimal scale or overestimate it. The scale cannot be underestimated by the `absmax` strategy, because it is the smallest scale that avoids clipping of the maximum element, and making it larger only causes more underflow and less precision.

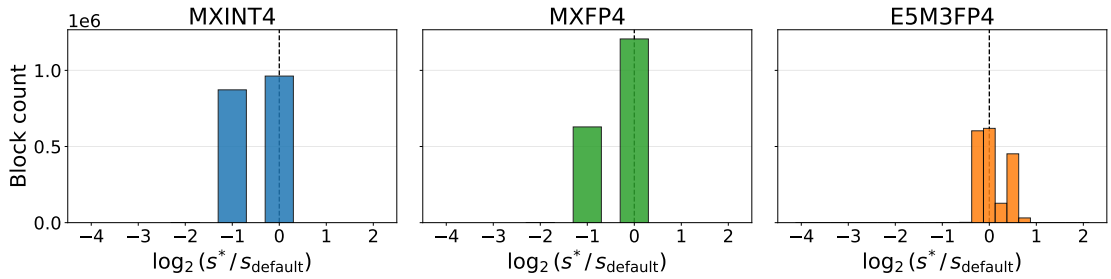


Figure 4.5: Distribution of the distance between the optimal scale and the `absmax` scale for MXINT4, MXFP4, and E5M3FP4 under RTN without micro-rotations. The mass at zero corresponds to blocks where `absmax` selects an optimal scale.

MXFP selects the optimal scale in around 66% of all blocks, while MXINT does so in around 52% of all blocks.

E5M3FP4 shows a different trend. `absmax` tends to both overestimate and underestimate the scale. The reason is that, for floating-point scales, the optimal scale focuses on the maximum accuracy of the Hessian outlier. However, `absmax` is blind to the Hessian sensitivity, and therefore chooses an optimal scale only 23% of the time.

The optimal scale for MXFP and MXINT usually sits right next to the scale selected by `absmax`. We found one rare block in the layer where the optimal scale was two steps away from the max-based scale, but in all other cases it was only one step away. This suggests another optimization for searching for optimal scales: choose between the `absmax` scale and the nearest smaller scale.

E5M3FP4 is more fine-grained, which makes the position of the optimal scale less predictable. However, the log distance is always less than 1. Therefore, a potential scale-search algorithm can use this information and look for the optimal scale only in the neighborhood of the max-based scale.

4.3.2 Midmax Threshold Analysis for MXFP4

`midmax` is an improved data-free scale selection algorithm that directly targets the main weakness of `absmax`. The previous subsection showed that `absmax` is often too conservative because it chooses the smallest power-of-two scale that avoids clipping the maximum element, even when clipping that element would reduce the scale objective. MXFP4 has another built-in rule, `midmax`, which chooses a smaller scale than `absmax` and then promotes the scale when the rescaled block maximum exceeds a certain threshold. For E2M1 data values, this threshold is 7, while the largest representable finite value is 6.

The key empirical question is whether the `midmax` threshold matches the true threshold. For

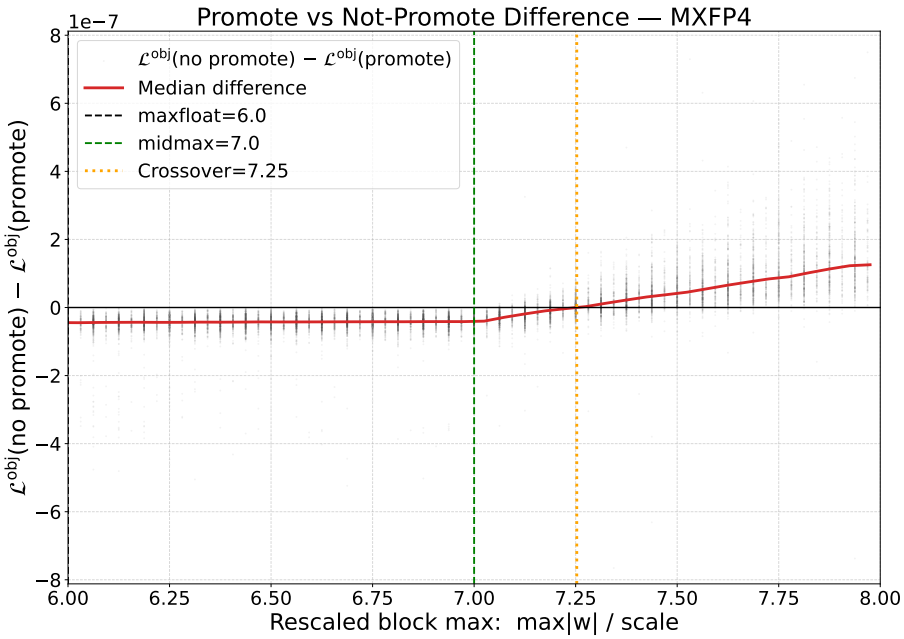


Figure 4.6: Empirical threshold comparison for MXFP4 under RTN without micro-rotations. Positive values mean that promoting the power-of-two scale, and therefore clipping the maximum element, gives a lower scale objective than keeping the smaller scale.

every block, we compare two adjacent power-of-two scales: the unpromoted scale and the promoted scale. The unpromoted scale has finer quantization points for smaller values but may clip the maximum element. The promoted scale avoids that clipping but uses coarser quantization points (equivalent to `absmax`). Figure 4.6 plots the difference between these two objectives. Positive values mean that the promoted scale is better, while negative values mean that keeping the finer unpromoted scale is better.

As we can see in the figure, the empirical optimal threshold is close to the midmax threshold, so `midmax` is much stronger than a strict no-clipping `absmax` rule for MXFP4. In the aggregate deviation experiment, `midmax` selects an optimal scale in about 93% of blocks, compared with about 66% for `absmax`. The remaining gap is small in total loss. In our experiments, replacing `midmax` with the exhaustive optimum improves the total single-layer loss by only about 1.2%.

The limitation of `midmax` is that it cannot adapt to Hessian-sensitive outliers. The rule uses only the block maximum, so it cannot account for which element has the largest Hessian value. In the worst MXFP4 block under `midmax`, the optimal objective is still about 22× smaller than the objective at the selected scale. Therefore, `midmax` solves most ordinary MXFP4 cases, but rare Hessian-outlier blocks can still require optimal objective-based scale selection.

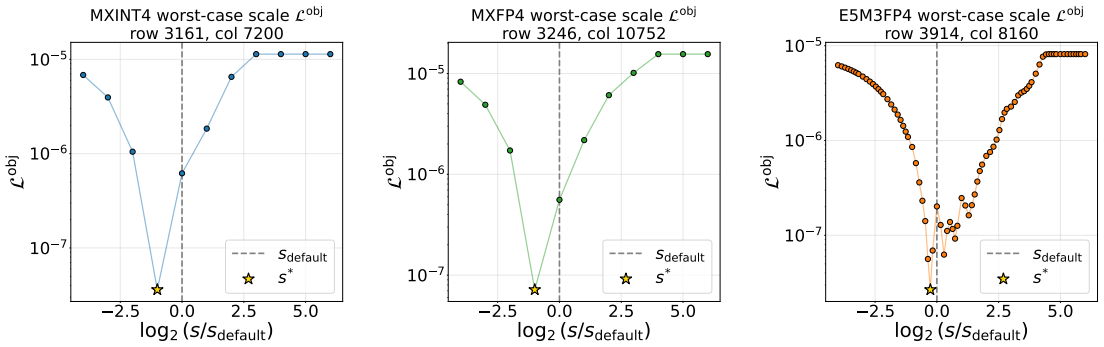


Figure 4.7: Worst-case scale objectives for MXINT4, MXFP4, and E5M3FP4 under MR-RTN with `absmax` scale selection.

4.4 Effect of Micro-Rotations

Now, we want to examine the effect of micro-rotations on the accuracy of scale selection algorithms.

Micro-rotations reduce the worst-case scale-selection gap, but they still do not make the scale-selection problem trivial. Figure 4.7 shows the worst-case objective plot after applying MR-RTN. As we can see, the objective curves still contain visible dips and a well-defined global minimum, so the qualitative features of the objective function remain.

However, the worst-case gaps become smaller after the micro-rotation. The worst-case objective ratio drops from about 138× to 17× for MXINT4, from about 46× to 8× for MXFP4, and from about 108× to 8× for E5M3FP4. This supports the analysis in Section 3.4.1: the micro-rotation spreads Hessian sensitivity across coordinates, which weakens single-coordinate outliers and makes bad scale choices less catastrophic.

The MR-RTN decomposition shows that the remaining dips are less dominated by a single Hessian outlier. In the RTN decomposition from Figure 4.2, one very thick per-element curve closely follows the full objective. In contrast, Figure 4.8 shows that the MR-RTN per-element curves have more similar thickness and opacity. The Hessian values are therefore more evenly spread, although they are not exactly identical.

The remaining dip is mainly an aggregate quantization-grid effect. Even if the Hessian values were uniform, decreasing the scale can improve many non-maximum elements by giving them finer quantization points, while only a few large elements are clipped. The optimal scale is therefore no longer determined by one dominant Hessian outlier. It is determined by a tradeoff between clipping a few large values and improving many smaller values.

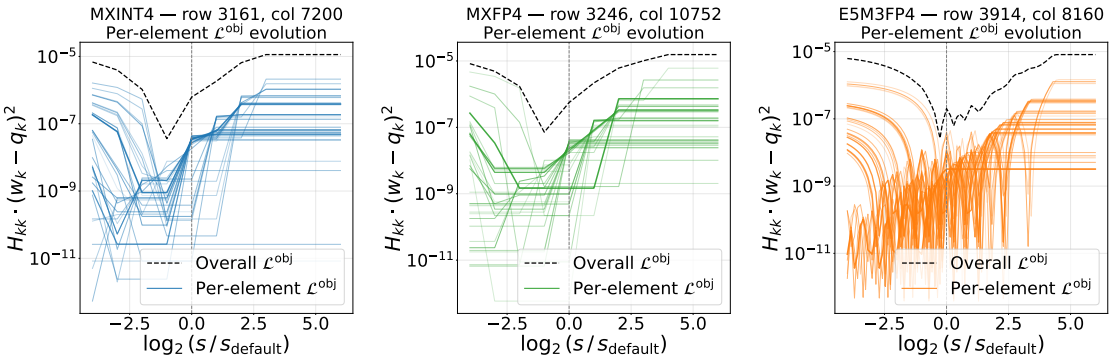


Figure 4.8: Per-element decomposition of the MR-RTN worst-case scale objectives for MXINT4, MXFP4, and E5M3FP4 under `absmax` scale selection. The line thickness and opacity are proportional to the corresponding Hessian diagonal value.

4.5 Effect of the Global Scale

The tensor-level global scale aligns the representable scale grid with the tensor range. In the _g variants, the block scale is encoded as a block-level scale multiplied by one FP32 tensor-level factor. This global factor shifts the whole grid so that the available block scales are better matched to the overall magnitude range of the tensor.

The global scale has little effect when the floating-point scale already has enough exponent range. In the example of E5M3FP4, the `absmax` scale is optimal in a similar fraction of blocks, around 23% without the global scale and 22% with it. Figure 4.9 shows the scale-deviation distributions, which are nearly identical for E5M3FP4 and E5M3FP4_g. The E4M3FP4 row shows a stronger effect because E4M3 has less exponent range than E5M3. The tensor-level factor shifts the representable scale grid, so the deviation distribution changes more visibly than in the E5M3FP4 pair.

However, global scale does not remove the per-block scale-selection problem. In Figure 4.9, both E4M3FP4 variants still have visible mass away from $\log_2(s^* / s_{\text{default}}) = 0$. The E4M3FP4_g distribution is shifted relative to E4M3FP4, which shows that aligning the tensor-level range and selecting the best block scale are separate problems.

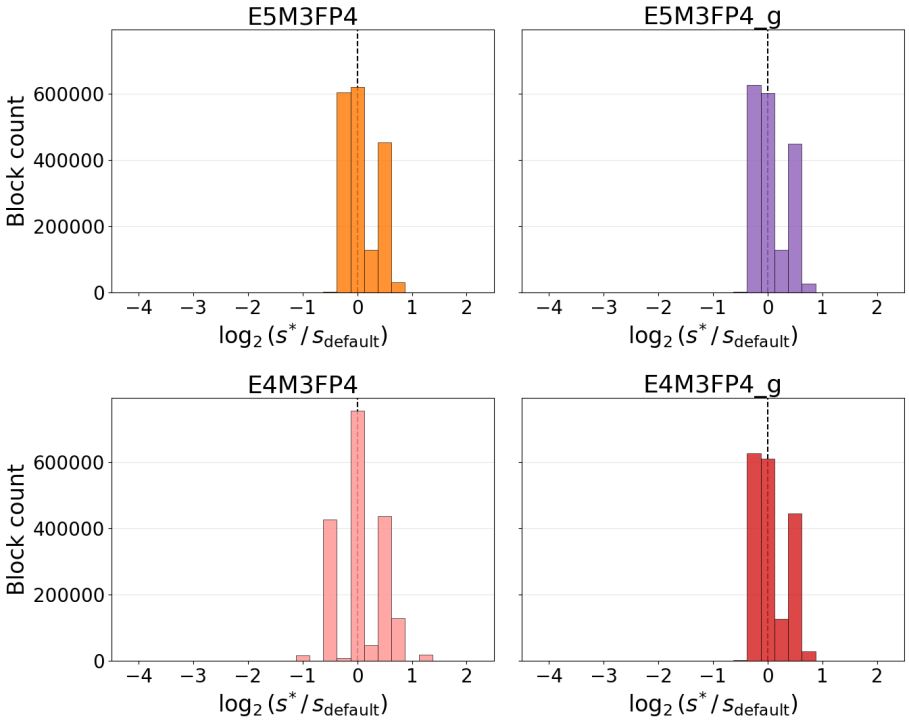


Figure 4.9: Scale-deviation distributions for floating-point scale formats with and without a tensor-level FP32 global scale under RTN with `absmax` scale selection.

5 End-to-End Evaluation

The previous chapters analyzed scale selection through the mathematical analysis and the single-layer experiments. This chapter conducts a full end-to-end evaluation of the optimal scale selection strategy in various quantization configurations to confirm the positive impact on model quality. This analysis also allows a quantitative comparison of different datatypes for the shared scale.

5.1 Experimental Setup

We evaluate the effect of the scale datatype and the scale selection in an end-to-end W4A4 setting. The model is Llama-3.1-8B-Instruct. Both weights and activations are quantized to 4-bit MX formats, and the BF16 model is used as the baseline.

Weight quantization is performed with either RTN or GPTQ. All GPTQ-based runs, including GPTQ and MR-GPTQ, use the FineWeb [37] calibration with 1024 samples. Both weights and activations are quantized to the same datatype. When a format uses two-level scaling, the activation and the weight tensor each receive an FP32 scale before casting the elements to the underlying MX format.

The evaluation spans across four design axes: the shared scale datatype, the scale selection policy, the quantization algorithm and the block size. Table 5.1 lists the tested formats.

For each format, we compare the implemented scale-selection policy with `optscale`. The baseline policies are `absmax` for all formats, with two other power-of-two scale policies included where they are supported: `max` for `mxint4` and `midmax` for `mxfp4`. `optscale` searches the discrete scale set to minimize the scale objective for each block, as in Chapter 4.

We report Wikitext-2 [38] perplexity as the primary evaluation metric because it is sensitive to full-model degradation and stable across nearby configurations. We also evaluate accuracy on WinoGrande 5-shot [39], HellaSwag 10-shot [40], ARC-Challenge 25-shot [41], and PIQA 0-shot [42]. The complete results for these datasets can be found in Tables A.2, A.3, A.4, and A.5.

Table 5.1: Formats tested in the end-to-end W4A4 evaluation.

Format	Weight grid	Block scale	Tensor scale	Block size
`mxint4`	INT4	E8M0	×	32
`mxfp4`	FP4	E8M0	×	32
`e4m3fp4`	FP4	E4M3	×	32
`e4m4fp4`	FP4	E4M4	×	32
`e5m3fp4`	FP4	E5M3	×	32
`e4m3fp4_g`	FP4	E4M3	✓	32
`e4m4fp4_g`	FP4	E4M4	✓	32
`e5m3fp4_g`	FP4	E5M3	✓	32
`e5m3fp4_t16`	FP4	E5M3	×	16
`e4m3fp4_g_t16`	FP4	E4M3	✓	16
`e4m4fp4_g_t16`	FP4	E4M4	✓	16
`e5m3fp4_g_t16`	FP4	E5M3	✓	16

The complete grid compares RTN, GPTQ, MR-RTN, and MR-GPTQ. The MR prefix denotes the micro-rotated variant: MR-RTN is RTN after a micro-rotation, implemented as a block-diagonal Hadamard transform, and MR-GPTQ is GPTQ after the same micro-rotation. In both cases, the micro-rotation is applied to weights and activations before quantization.

In GPTQ and MR-GPTQ, the columns are quantized from left to right in groups of the same size as the MX block. The optimal scale is selected for the active group after the previous groups are handled by the algorithm.

5.2 Shared Scale Datatype

5.2.1 Power-of-Two vs. Floating-Point Scales

The first question is whether the shared scale should remain a power of two, as in the standard MX formats, or whether a floating-point scale is preferable. To answer this question, Table 5.2 compares the block-32, non-rotated RTN and GPTQ results. The table reports the results for both `absmax` and `optscale`.

Floating-point scales consistently give better accuracy than power-of-two scale formats in this comparison. With `optscale`, the best power-of-two result is `mxfp4`, reaching 9.12 perplexity under RTN and 8.94 under GPTQ. The floating-point scale formats lie between 8.19 and 8.43 under RTN and between 8.17 and 8.33 under GPTQ. The gap is larger for `mxint4`, which reaches only 9.87 under RTN and 9.59 under GPTQ even after `optscale`. This suggests that the power-of-two scale is a major limitation for INT4 blocks. The `mxfp4` format is stronger because its FP4 value grid is non-uniform, which fits the real distributions better, but its power-of-two scale still leaves a visible gap to the floating-point scale formats.

Table 5.2: Tile-32 Wikitext-2 perplexity without micro-rotations. Lower is better.

Format	Scale type	RTN	RTN + `optscale`	GPTQ	GPTQ + `optscale`
`mxint4`	E8M0	11.09	9.87	10.26	9.59
`mxfp4`	E8M0	9.32	9.12	8.96	8.94
`e4m3fp4`	E4M3	8.96	8.43	8.47	8.33
`e4m4fp4`	E4M4	8.69	8.32	8.37	8.21
`e5m3fp4`	E5M3	8.66	8.30	8.39	8.27
`e4m3fp4_g`	FP32 + E4M3	8.68	8.29	8.37	8.20
`e4m4fp4_g`	FP32 + E4M4	8.45	8.19	8.27	8.17
`e5m3fp4_g`	FP32 + E5M3	8.71	8.29	8.31	8.27

5.2.2 Floating-Point Scale Comparison

Among the block-32 floating-point scale formats, E4M4 and E5M3 give the best accuracy. With `optscale`, `e4m4fp4_g` reaches 8.19 PPL under RTN and 8.17 under GPTQ. E5M3 is close, with `e5m3fp4` at 8.30 under RTN and 8.27 under GPTQ, and `e5m3fp4_g` at 8.29 under RTN and 8.27 under GPTQ. E4M3 benefits more from the tensor-level scale: without it, `e4m3fp4` reaches 8.43 under RTN and 8.33 under GPTQ, while `e4m3fp4_g` improves to 8.29 and 8.20.

The small degradation of E5M3_g relative to E4M3_g likely comes from the tensor-level scale selection. In two-level formats, the tensor-level FP32 scale is computed from the product of the FP4 value-grid maximum and the maximum representable block scale. Since E5M3 has a much larger maximum scale than E4M3, the same tensor is normalized with a much larger scale-range target.

The impact of the tensor-level scale is format-dependent. For E5M3, the variant without the tensor-level scale shows similar accuracy to the two-level variant under both RTN and GPTQ, especially after `optscale`. This indicates that the wider exponent range of E5M3 is enough to cover the relevant tensor scales in this setting, making the additional tensor-level FP32 scale mostly redundant. For E4M4, the two-level variant brings more improvement, reducing the `optscale` result from 8.32 to 8.19 under RTN and from 8.21 to 8.17 under GPTQ. For E4M3, the tensor-level scale is more important, reducing the `optscale` result from 8.43 to 8.29 under RTN and from 8.33 to 8.20 under GPTQ.

These results show that moving from power-of-two scales to floating-point scales gives a significant accuracy improvement. Moreover, introducing a global scale can further improve the end-to-end accuracy for certain numerical formats with a limited scale dynamic range.

5.3 Scale Selection Policy

The previous chapters showed that max-based scale selection is not generally optimal for the scale objective. To confirm this claim, Table 5.3 compares each implemented scale-selection policy with `optscale` for block-32 quantization without micro-rotations.

Table 5.3: Effect of `optscale` on tile-32 Wikitext-2 perplexity without micro-rotations. Lower is better.

Format	Policy	RTN	RTN + `optscale`	GPTQ	GPTQ + `optscale`
`mxint4`	`absmax`	11.09	9.87 (-1.23)	10.26	9.59 (-0.67)
`mxint4`	`max`	10.15	9.55 (-0.60)	9.50	9.33 (-0.17)
`mxfp4`	`absmax`	9.32	9.12 (-0.20)	8.96	8.94 (-0.03)
`mxfp4`	`midmax`	9.12	9.00 (-0.12)	8.84	8.83 (-0.01)
`e4m3fp4`	`absmax`	8.96	8.43 (-0.53)	8.47	8.33 (-0.13)
`e4m4fp4`	`absmax`	8.69	8.32 (-0.37)	8.37	8.21 (-0.16)
`e5m3fp4`	`absmax`	8.66	8.30 (-0.36)	8.39	8.27 (-0.12)
`e4m3fp4_g`	`absmax`	8.68	8.29 (-0.40)	8.37	8.20 (-0.17)
`e4m4fp4_g`	`absmax`	8.45	8.19 (-0.26)	8.27	8.17 (-0.10)
`e5m3fp4_g`	`absmax`	8.71	8.29 (-0.42)	8.31	8.27 (-0.04)

The strongest effect appears for `mxint4`. With the default `absmax` policy, `optscale` improves RTN perplexity from 11.09 to 9.87 and GPTQ perplexity from 10.26 to 9.59. Even when the stronger `max` policy is used, `optscale` still improves RTN from 10.15 to 9.55 and GPTQ from 9.50 to 9.33. This matches the single-layer analysis: the INT4 value grid is uniform, so a poorly selected shared scale shifts the whole grid away from the Hessian-important weights.

For `mxfp4`, the gains are smaller. Starting from `absmax`, `optscale` improves RTN from 9.32 to 9.12 and GPTQ from 8.96 to 8.94. Starting from `midmax`, the gap nearly disappears: RTN improves from 9.12 to 9.00, while GPTQ improves from 8.84 to 8.83. This suggests that `midmax` is already a strong heuristic for `mxfp4` in this setting.

Floating-point scale formats also benefit from `optscale`, but the gains are more moderate than for `mxint4`. Under RTN, the improvement is consistently visible, ranging from 0.26 PPL for `e4m4fp4_g` to 0.53 PPL for `e4m3fp4`. Under GPTQ, the gains shrink to roughly 0.04 to 0.17 PPL. This indicates that GPTQ already compensates for part of the quantization error, but does not make scale selection irrelevant.

Overall, the end-to-end results confirm that the optimal scale selection strategy improves the end-to-end accuracy. The importance of `optscale` depends on the format, as it is largest for `mxint4`, still useful for floating-point scales, and smallest for `mxfp4` with `midmax`. Optimal scale search matters most when the value grid is coarse or when the baseline scale policy is tied too strongly to the maximum element.

5.4 Interaction with GPTQ

The interaction between GPTQ and `optscale` is format-dependent. Without `optscale`, GPTQ consistently improves over RTN. For example, it reduces perplexity from 11.09 to 10.26 for `mxint4` with `absmax`, from 10.15 to 9.50 for `mxint4` with `max`, and from 9.32 to 8.96 for `mxfp4` with `absmax`. This is the expected behavior: GPTQ diffuses quantization error using Hessian

Table 5.4: Effect of micro-rotations on tile-32 RTN Wikitext-2 perplexity. Lower is better.

Format	Policy	RTN	MR-RTN	RTN + optscale	MR-RTN + optscale
mxint4	absmax	11.09	10.53 (-0.56)	9.87	9.56 (-0.30)
mxint4	max	10.15	9.62 (-0.53)	9.55	9.29 (-0.26)
mxfp4	absmax	9.32	9.24 (-0.08)	9.12	9.03 (-0.10)
mxfp4	midmax	9.12	9.04 (-0.08)	9.00	8.99 (-0.02)
e4m3fp4	absmax	8.96	9.06 (+0.11)	8.43	8.92 (+0.49)
e4m4fp4	absmax	8.69	9.09 (+0.41)	8.32	8.68 (+0.36)
e5m3fp4	absmax	8.66	8.95 (+0.29)	8.30	8.83 (+0.53)
e4m3fp4_g	absmax	8.68	9.04 (+0.35)	8.29	8.77 (+0.48)
e4m4fp4_g	absmax	8.45	8.86 (+0.41)	8.19	8.59 (+0.40)
e5m3fp4_g	absmax	8.71	9.03 (+0.32)	8.29	8.71 (+0.41)

information, while RTN quantizes each value independently.

Once optscale is used, the gap between RTN and GPTQ becomes smaller. For mxint4, GPTQ still improves over RTN by 0.22 to 0.28 PPL after optscale. For mxfp4, the remaining improvement is around 0.17 to 0.18 PPL. For floating-point scale formats, however, RTN with optscale is often close to GPTQ with optscale. For example, e4m3fp4_g gives 8.29 with RTN and 8.20 with GPTQ, while e5m3fp4 gives 8.30 with RTN and 8.27 with GPTQ.

These results suggest that GPTQ and optscale are partly complementary and partly redundant. GPTQ remains useful for power-of-two scale formats, where the scale grid is coarse and is less forgiving. For floating-point scale formats, better scale placement already removes a substantial part of the error that GPTQ would otherwise correct.

5.5 Micro-Rotations

The mathematical analysis in Section 3.4.1 predicts that micro-rotations change the scale-selection problem by spreading Hessian sensitivity across coordinates. The single-layer experiments in Chapter 4 confirmed this effect, as micro-rotations reduced the worst-case scale-selection gap for MXINT4, MXFP4 and E5M3FP4, but they did not remove the gap completely. To confirm the effect of micro-rotations on end-to-end accuracy, Table 5.4 compares RTN with and without micro-rotations for block-32 formats.

The clearest benefit from micro-rotations appears for mxint4. With absmax, micro-rotations reduce perplexity from 11.09 to 10.53, and with absmax plus optscale they reduce it from 9.87 to 9.56. The same pattern holds for max, where micro-rotations improve perplexity from 10.15 to 9.62 without optscale and from 9.55 to 9.29 with optscale. This validates the single-layer analysis.

For mxfp4, the same behavior is present but the gain is much smaller. Without optscale, MR-RTN improves RTN from 9.32 to 9.24 under absmax and from 9.12 to 9.04 under midmax.

With optscale, micro-rotations remain mildly beneficial, improving perplexity from 9.12 to 9.03 under absmax and from 9.00 to 8.99 under midmax. This also matches the single-layer picture. Micro-rotations reduce the worst-case objective gap for MXFP4, but midmax already solves most ordinary MXFP4 blocks and the FP4 value grid is non-uniform. As a result, fewer harmful scale-selection errors remain for the micro-rotation to fix. The best mxfp4 result still uses a micro-rotation, optscale and midmax, but the improvement is modest.

The most important difference from the single-layer analysis appears for floating-point scale formats. In the single-layer study, micro-rotations reduced the worst-case objective gap for E5M3FP4. However, they consistently hurt perplexity under both absmax and optscale in the end-to-end results. Under absmax, the degradation ranges from 0.11 to 0.41 PPL. With optscale, the degradation ranges from 0.36 to 0.53 PPL across all tested block-32 floating-point scale formats.

This counter-intuitive result does not contradict the earlier analysis. The single-layer result says that micro-rotations make the scale-selection objective less dominated by Hessian outliers. It does not say that the micro-rotated tensor values are always easier to quantize with a given numerical format. Floating-point scale formats already have a dense set of block-scale values, so optscale can directly adapt the scale to Hessian-sensitive coordinates without needing to first smooth the block by micro-rotation. After the micro-rotation, the original outlier problem is weaker, but the quantizer sees a different distribution of values and scale requirements. For formats that already adapt well locally, this distribution shift can outweigh the reduction in Hessian-outlier dominance.

The overall conclusion is therefore more nuanced. Micro-rotations do reduce the outliers identified in the mathematical and single-layer analyses, but they are only beneficial when the outlier value cannot be accurately represented. That is the case for mxint4, partially the case for mxfp4, and the opposite for the floating-point scale formats.

5.6 Block Size

Block size controls how many values share one local scale. Smaller blocks allow finer control over the shared scales, but they also increase cost and may be less attractive for hardware. Table 5.5 compares RTN against RTN with optscale without micro-rotations for both block-32 and block-16 formats.

Block-16 consistently improves Wikitext-2 perplexity over block-32. Without optscale, the block-size improvement ranges from 0.12 to 0.17 PPL across the matched RTN rows. With optscale enabled, the improvement remains similar, ranging from 0.14 to 0.18 PPL. This confirms that smaller blocks reduce the blast radius of Hessian outliers, but the size of the gain is modest compared with the gain from optimizing the scale.

However, block size does not remove the benefit of optscale. At block 16, optscale improves

Table 5.5: Effect of `optscale` on floating-point scale formats at two block sizes. Wikitext-2 perplexity, lower is better.

Format	Method	Block 32	Block 16
`e5m3fp4`	RTN	8.66	8.54
`e5m3fp4`	RTN + `optscale`	8.30	8.12
`e4m3fp4_g`	RTN	8.68	8.52
`e4m3fp4_g`	RTN + `optscale`	8.29	8.13
`e4m4fp4_g`	RTN	8.45	8.30
`e4m4fp4_g`	RTN + `optscale`	8.19	8.05
`e5m3fp4_g`	RTN	8.71	8.54
`e5m3fp4_g`	RTN + `optscale`	8.29	8.13

`e5m3fp4_t16` from 8.54 to 8.12, `e4m3fp4_g_t16` from 8.52 to 8.13, `e4m4fp4_g_t16` from 8.30 to 8.05, and `e5m3fp4_g_t16` from 8.54 to 8.13. These gains range from 0.25 to 0.42 PPL, which is larger than the gain from reducing the block size alone. This indicates that reducing the block size and optimizing the scale solve distinct problems: `optscale` chooses the best representable scale for the block while smaller blocks allow values to be scaled more locally.

5.7 Practical Recommendations

The end-to-end results support four practical recommendations. First, prefer floating-point block scales over power-of-two scales when the hardware format allows it. The best tested configuration is `e4m4fp4_g_t16` with GPTQ and `optscale`, which reaches 8.04 Wikitext-2 perplexity. The next best configurations are also floating-point scale formats: `e4m4fp4_g_t16` reaches 8.05 with RTN and `optscale`, `e5m3fp4_t16` reaches 8.09 with GPTQ and `optscale`, and `e4m3fp4_g_t16` reaches 8.10 with GPTQ and `optscale`. By contrast, the best power-of-two scale results are 8.83 for `mxfp4` and 9.18 for `mxint4`, even after combining micro-rotations, GPTQ and `optscale`.

Second, use `optscale` whenever the implementation can afford the search. Its benefit is largest for `mxint4`, where it reduces block-32 GPTQ perplexity by 0.67 PPL from the `absmax` baseline and RTN perplexity by 1.23 PPL. It remains useful for floating-point scale formats, especially under RTN, and it still improves many GPTQ rows. The only near-exception is `mxfp4` with `midmax`, where the heuristic is already close to optimal in this grid.

Third, micro-rotations are format-dependent. They are useful for `mxint4` and mildly useful for the best `mxfp4` configuration, but they are not a general improvement. For floating-point scale formats, micro-rotations consistently worsen Wikitext-2 perplexity in the tested block-32 GPTQ configurations. This makes micro-rotations attractive when the scale grid is coarse and outliers dominate the scale choice, but not when the block scale can already adapt finely.

Fourth, prefer block-16 over block-32 when the hardware cost is acceptable. Block-16 gives the best perplexity results in this evaluation and improves every matched floating-point scale

row with `optscale`. However, the improvement is smaller than the jump from power-of-two scales to floating-point scales. If block-16 is too expensive, block-32 with a floating-point scale and `optscale` remains a strong configuration.

Overall, the most robust recommendation is to combine an FP4 value grid with a floating-point shared scale, `optscale` and the smallest practical block size. GPTQ is still useful, but its benefit becomes less uniform once scale selection is optimized. For power-of-two scale formats, use the strongest available heuristic before `optscale`: `max` for `mxint4` and `midmax` for `mxfp4`. If micro-rotations are available, apply them to power-of-two scale formats.

6 Conclusion

This chapter concludes the report by bringing together the mathematical analysis, the single-layer experiments, and the end-to-end evaluation. We first summarize the main findings about shared-scale datatypes, scale-selection policies, micro-rotations, block size, and global scaling. We then discuss directions for future work.

6.1 Summary of Findings

This report studied shared-scale selection as a formal optimization problem. We formulated the scale-selection objective as a Hessian-weighted sum of individual quantization errors, connected it with the model's loss and used it to define `optscale` as the best representable shared scale for a block according to the objective. This framework allowed us to compare the scale datatypes. We showed that power-of-two scales have per-element unimodality, while floating-point scales do not. The framework also clarified why max-based rules can be suboptimal. The largest element does not necessarily contribute to the objective the most, and the optimal scaling strategy can prefer clipping that element when a smaller scale improves the quantization resolution of more important values with smaller magnitude.

We then tested the objective on a real model layer. We discovered that for power-of-two scales the Hessian-dominant coordinate shapes the scaling objective, so it is close to unimodal. For the examined floating-point scale case, we observed the non-unimodality predicted by the mathematical analysis. Its local minima appeared when the sensitive coordinate was represented accurately, which is a useful insight for an efficient algorithm for optimal scale search. We also showed that `absmax` often selects a suboptimal scale for all scale datatypes, while `midmax` is a strong heuristic for ordinary `mxfp4` blocks.

Consequently, we evaluated various configurations end-to-end. The experiments showed that floating-point shared scales are the strongest format-level improvement over power-of-two scales. We also showed that `optscale` improves most tested configurations, especially when the scale is a power-of-two or when the baseline policy is `absmax`. We found that GPTQ and

`optscale` complement each other in several numerical formats. We also found that micro-rotations are useful only when the outliers cannot be represented accurately. This matches the mathematical and single-layer analyses, where micro-rotations spread Hessian sensitivity and reduce the worst failures of max-based scale selection. However, micro-rotations are not a default choice for floating-point shared scales, because these formats can already adapt the local scale more finely and the micro-rotated values may become harder to quantize.

Finally, we studied block size and tensor-level global scaling. Smaller blocks reduce the number of values that share one scale, so they reduce the pressure on each local scale-selection decision. Tensor-level global scaling solves a different problem, as it shifts the block-scale grid to match the tensor magnitude range. However, neither mechanism eliminates the local scale-selection problem.

Overall, the strongest design direction is to combine an FP4 value grid, a floating-point shared scale, `optscale`, and the smallest practical block size, while using GPTQ and micro-rotations only when they match the format. Therefore, the scale datatype, the scale-selection policy, the weight quantization algorithm, the micro-rotation policy, and the block size should be treated as one coupled design problem.

6.2 Future Work

This work leaves several directions for future work. The first direction is activation scale selection. Activations depend on the input, so their scales cannot be optimized before inference. The Hessian-weighted objective used in this report is naturally tied to weight quantization. For activations, future work may need data-free objectives, online approximations, or hybrid policies that adapt to the input without adding excessive runtime cost.

The second direction is cheaper objective-based search. In the tested power-of-two scale weight blocks, the optimal scale often stayed close to the max-based scale. This suggests that a local search around the baseline scale may be enough in many realistic blocks. Floating-point shared scales are less regular, but the single-layer analysis suggests that candidate scales can be built from sensitive coordinates and nearby quantization points. These shortcuts would make `optscale` more practical when exhaustive search is too expensive.

Future work should also study scale selection together with tensor-level global scaling, which has shown promising results in recent work [7], and with learned rotations [21]. Both mechanisms change the values seen by the block quantizer, so they should be evaluated together with the shared scale datatype and the scale-selection policy rather than in isolation. Finally, the experiments in this report used one model family and a limited evaluation setting. Additional model families, calibration datasets, and downstream tasks are needed to test how broadly the conclusions transfer.

A Additional End-to-End Results

Tables A.1, A.2, A.3, A.4, and A.5 report the complete FineWeb-1024 W4A4 evaluation grid together with the BF16 baseline. The main text uses selected views of the end-to-end evaluation to isolate individual effects, while this appendix gives the Wikitext-2 perplexity and task accuracy results for every completed configuration.

Table A.1: Full W4A4 Wikitext-2 perplexity results for the FineWeb-1024 evaluation grid with the BF16 baseline. Lower is better.

Format	Policy	RTN	RTN + optscale	GPTQ	GPTQ + optscale	MR-RTN	MR-RTN + optscale	MR-GPTQ	MR-GPTQ + optscale
bf16	Baseline	7.55							
mxint4	absmax	11.09	9.87	10.26	9.59	10.53	9.56	10.01	9.44
mxint4	max	10.15	9.55	9.50	9.33	9.62	9.29	9.25	9.18
mxfp4	absmax	9.32	9.12	8.96	8.94	9.24	9.03	9.16	8.93
mxfp4	midmax	9.12	9.00	8.84	8.83	9.04	8.99	8.99	8.90
e4m3fp4	absmax	8.96	8.43	8.47	8.33	9.06	8.92	8.94	8.88
e4m4fp4	absmax	8.69	8.32	8.37	8.21	9.09	8.68	8.96	8.64
e5m3fp4	absmax	8.66	8.30	8.39	8.27	8.95	8.83	8.87	8.67
e4m3fp4_g	absmax	8.68	8.29	8.37	8.20	9.04	8.77	8.84	8.68
e4m4fp4_g	absmax	8.45	8.19	8.27	8.17	8.86	8.59	8.78	8.60
e5m3fp4_g	absmax	8.71	8.29	8.31	8.27	9.03	8.71	8.92	8.69
e5m3fp4_t16	absmax	8.54	8.12	8.27	8.09	8.96	8.54	8.73	8.47
e4m3fp4_g_t16	absmax	8.52	8.13	8.29	8.10	8.84	8.58	8.76	8.48
e4m4fp4_g_t16	absmax	8.30	8.05	8.15	8.04	8.73	8.49	8.66	8.38
e5m3fp4_g_t16	absmax	8.54	8.13	8.34	8.10	8.89	8.59	8.72	8.51

Table A.2: Full W4A4 WinoGrande 5-shot accuracy results for the FineWeb-1024 evaluation grid with the BF16 baseline, reported in percent using the `acc` metric. Higher is better.

Format	Policy	RTN	RTN + `optscale`	GPTQ	GPTQ + `optscale`	MR-RTN	MR-RTN + `optscale`	MR-GPTQ	MR-GPTQ + `optscale`
`bf16`	Baseline	78.4							
`mxint4`	`absmax`	67.9	69.6	69.0	71.7	69.9	71.9	69.2	73.4
`mxint4`	`max`	71.2	72.3	73.0	71.3	72.4	74.1	71.0	72.0
`mxfp4`	`absmax`	71.3	74.2	74.2	73.2	75.1	74.3	75.4	73.8
`mxfp4`	`midmax`	73.1	73.3	74.3	75.3	74.2	73.5	73.2	74.0
`e4m3fp4`	`absmax`	74.1	73.9	75.8	73.6	75.1	73.3	73.8	73.9
`e4m4fp4`	`absmax`	75.9	75.2	75.3	75.1	72.8	74.6	73.5	75.3
`e5m3fp4`	`absmax`	74.0	75.6	75.3	75.2	74.3	75.3	74.0	73.6
`e4m3fp4_g`	`absmax`	73.9	75.8	74.4	75.5	73.6	74.3	73.2	75.0
`e4m4fp4_g`	`absmax`	74.3	74.9	75.5	76.1	74.1	75.2	74.2	74.0
`e5m3fp4_g`	`absmax`	74.2	74.5	75.1	74.2	73.7	75.8	73.3	75.0
`e5m3fp4_t16`	`absmax`	75.6	75.4	75.8	75.9	75.2	76.5	74.0	74.5
`e4m3fp4_g_t16`	`absmax`	75.0	74.5	74.0	75.4	75.1	74.3	74.7	75.1
`e4m4fp4_g_t16`	`absmax`	74.8	76.6	75.1	76.8	74.6	75.4	75.9	74.3
`e5m3fp4_g_t16`	`absmax`	74.4	77.0	75.2	75.0	75.5	75.0	75.6	74.6

Table A.3: Full W4A4 HellaSwag 10-shot accuracy results for the FineWeb-1024 evaluation grid with the BF16 baseline, reported in percent using the `acc` metric. Higher is better.

Format	Policy	RTN	RTN + `optscale`	GPTQ	GPTQ + `optscale`	MR-RTN	MR-RTN + `optscale`	MR-GPTQ	MR-GPTQ + `optscale`
`bf16`	Baseline	59.7							
`mxint4`	`absmax`	52.0	54.8	54.0	54.3	53.4	55.1	54.2	55.1
`mxint4`	`max`	53.8	54.8	55.6	54.9	54.8	55.6	55.5	55.8
`mxfp4`	`absmax`	55.4	55.8	55.9	55.9	55.5	55.5	55.7	56.5
`mxfp4`	`midmax`	56.0	55.9	56.0	56.1	55.8	55.8	56.9	56.5
`e4m3fp4`	`absmax`	55.8	57.3	57.3	57.6	56.7	57.2	56.8	56.8
`e4m4fp4`	`absmax`	56.8	57.6	57.8	57.6	55.8	57.5	56.7	57.4
`e5m3fp4`	`absmax`	56.8	57.5	57.4	57.6	57.0	57.4	56.9	57.3
`e4m3fp4_g`	`absmax`	56.0	57.6	57.5	57.6	56.5	57.3	57.1	56.7
`e4m4fp4_g`	`absmax`	57.1	57.9	57.8	57.7	56.7	57.6	56.9	57.2
`e5m3fp4_g`	`absmax`	56.8	57.7	57.8	57.7	56.8	57.7	57.1	57.5
`e5m3fp4_t16`	`absmax`	57.2	58.1	58.5	57.9	57.3	57.7	57.3	58.1
`e4m3fp4_g_t16`	`absmax`	57.1	57.9	58.0	58.2	57.5	57.0	57.6	57.5
`e4m4fp4_g_t16`	`absmax`	58.2	58.3	58.3	58.2	58.1	57.5	57.5	57.7
`e5m3fp4_g_t16`	`absmax`	57.0	58.0	58.4	57.9	57.3	57.4	57.4	57.6

Table A.4: Full W4A4 ARC-Challenge 25-shot accuracy results for the FineWeb-1024 evaluation grid with the BF16 baseline, reported in percent using the `acc` metric. Higher is better.

Format	Policy	RTN	RTN + optscale	GPTQ	GPTQ + optscale	MR-RTN	MR-RTN + optscale	MR-GPTQ	MR-GPTQ + optscale
bf16	Baseline	57.5							
mxint4	absmax	47.1	48.8	49.7	50.9	49.7	51.2	49.0	50.3
mxint4	max	47.2	49.8	51.6	51.0	52.1	51.9	49.0	51.0
mxfp4	absmax	50.8	50.6	52.4	52.6	52.6	52.0	51.8	52.6
mxfp4	midmax	51.3	52.0	53.5	53.1	52.3	52.8	51.6	53.1
e4m3fp4	absmax	55.0	52.7	54.4	54.7	51.8	52.3	53.0	52.7
e4m4fp4	absmax	54.3	54.5	53.3	54.9	51.1	54.3	51.3	54.5
e5m3fp4	absmax	54.3	54.4	54.8	54.9	52.6	53.7	52.1	52.2
e4m3fp4_g	absmax	53.3	54.9	53.8	54.1	51.7	54.9	52.5	54.4
e4m4fp4_g	absmax	56.1	54.3	56.0	54.1	51.5	54.2	53.3	53.2
e5m3fp4_g	absmax	53.2	54.0	53.2	55.5	52.6	55.3	51.0	53.8
e5m3fp4_t16	absmax	56.1	54.3	55.5	54.7	53.3	53.4	53.2	54.7
e4m3fp4_g_t16	absmax	55.2	55.3	55.5	55.0	53.2	55.5	54.6	50.7
e4m4fp4_g_t16	absmax	55.8	54.9	53.8	55.0	52.6	55.8	54.8	53.3
e5m3fp4_g_t16	absmax	54.1	54.7	54.9	55.1	54.2	55.3	52.4	54.3

Table A.5: Full W4A4 PIQA 0-shot accuracy results for the FineWeb-1024 evaluation grid with the BF16 baseline, reported in percent using the `acc` metric. Higher is better.

Format	Policy	RTN	RTN + optscale	GPTQ	GPTQ + optscale	MR-RTN	MR-RTN + optscale	MR-GPTQ	MR-GPTQ + optscale
bf16	Baseline	80.2							
mxint4	absmax	75.6	77.6	76.7	76.7	76.0	76.8	77.1	78.2
mxint4	max	76.3	77.6	78.2	77.6	77.1	77.5	76.9	77.5
mxfp4	absmax	77.8	77.8	78.1	77.3	78.0	77.5	77.7	77.8
mxfp4	midmax	77.1	78.6	79.1	78.2	78.0	77.6	79.1	78.7
e4m3fp4	absmax	78.0	79.2	78.5	78.8	76.8	78.3	78.0	78.3
e4m4fp4	absmax	79.7	79.7	78.5	79.5	77.8	78.3	78.1	78.6
e5m3fp4	absmax	78.7	78.7	78.5	79.5	77.2	78.1	77.5	78.5
e4m3fp4_g	absmax	79.1	78.8	78.4	78.5	78.0	77.6	77.5	77.5
e4m4fp4_g	absmax	79.2	78.7	78.1	78.8	77.2	78.2	78.0	78.2
e5m3fp4_g	absmax	78.2	79.0	79.1	78.2	77.3	78.0	77.5	77.1
e5m3fp4_t16	absmax	79.8	80.0	78.9	79.8	78.2	78.7	78.6	78.5
e4m3fp4_g_t16	absmax	78.5	78.7	78.6	78.9	77.5	77.8	78.3	78.8
e4m4fp4_g_t16	absmax	79.2	79.4	78.2	78.2	78.3	77.8	78.6	78.2
e5m3fp4_g_t16	absmax	78.8	79.2	78.8	78.6	77.7	78.1	77.6	77.7

Bibliography

[1] Stanford Institute for Human-Centered Artificial Intelligence. Artificial Intelligence Index Report 2026, 2026. URL https://aiindex.stanford.edu/report/.

[2] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sashank Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24(1), January 2023. ISSN 1532-4435.

[3] DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024. doi: 10.48550/arXiv.2412.19437. URL https://arxiv.org/abs/2412.19437.

[4] DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. Hugging Face model card, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro.

[5] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of machine learning and systems*, 4:795–813, 2022.

[6] David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R So, Maud Texier, and Jeff Dean. The carbon footprint of machine learning training will plateau, then shrink. *Computer*, 55(7):18–28, 2022.

[7] Vage Egiazarian, Roberto L. Castro, Denis Kuznedelev, Andrei Panferov, Eldar Kurtic, Shubhra Pandit, Alexandre Noll Marques, Mark Kurtz, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Bridging the gap between promise and performance for microscaling FP4 quantization. *CoRR*, abs/2509.23202, 2025. doi: 10.48550/ARXIV.2509.23202. URL https://doi.org/10.48550/arXiv.2509.23202.

[8] Shibo Wang and Pankaj Kanwar. BFloat16: The secret to high performance on cloud tpus. Google Cloud Blog, 2019. URL https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus.

[9] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *CoRR*, abs/2208.07339, 2022. doi: 10.48550/ARXIV.2208.07339. URL https://doi.org/10.48550/arXiv.2208.07339.

[10] Open Compute Project. Ocp microscaling formats (mx) specification, 2023. URL https://www.opencompute.org/documents/ocp-microscaling-formats-mx-v1-0-spec-final-pdf.

[11] Eduardo Alvarez, Omri Almog, Eric Chung, Simon Layton, Dusan Stosic, Ronny Krashinsky, and Kyle Aubrey. Introducing NVFP4 for efficient and accurate low-precision inference. NVIDIA Technical Blog, 2025. URL https://developer.nvidia.com/blog/introducing-nvfp4-for-efficient-and-accurate-low-precision-inference/.

[12] ROCm. TensorCast: Tensor casting and quantization library. GitHub repository, 2026. URL https://github.com/ROCm/tensorcast.

[13] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: accurate post-training quantization for generative pre-trained transformers. *CoRR*, abs/2210.17323, 2022. doi: 10.48550/ARXIV.2210.17323. URL https://doi.org/10.48550/arXiv.2210.17323.

[14] Guangxuan Xiao, Ji Lin, Mickaël Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 38087–38099. PMLR, 2023. URL https://proceedings.mlr.press/v202/xiao23c.html.

[15] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: activation-aware weight quantization for on-device LLM compression and acceleration. In Phillip B. Gibbons, Gennady Pekhimenko, and Christopher De Sa, editors, *Proceedings of the Seventh Annual Conference on Machine Learning and Systems, MLSys 2024, Santa Clara, CA, USA, May 13-16, 2024*. mlsys.org, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html.

[16] Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless LLM weight compression. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Q1u25ahSuy.

[17] Shih-Yang Liu, Zechun Liu, Xijie Huang, Pingcheng Dong, and Kwang-Ting Cheng. LLM-FP4: 4-bit floating-point quantized transformers. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 592–605. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.39. URL https://doi.org/10.18653/v1/2023.emnlp-main.39.

[18] Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. Quip: 2-bit quantization of large language models with guarantees. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/0df38cd13520747e1e64e5b123a78ef8-Abstract-Conference.html.

[19] Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=8Wuvhh0LYW.

[20] Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/b5b939436789f76f08b9d0da5e81af7c-Abstract-Conference.html.

[21] Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: LLM quantization with learned rotations. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=ogO6DGE6FZ.

[22] Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. Turboquant: Online vector quantization with near-optimal distortion rate. *CoRR*, abs/2504.19874, 2025. doi: 10.48550/ARXIV.2504.19874. URL https://doi.org/10.48550/arXiv.2504.19874.

[23] Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pages 598–605. Morgan Kaufmann, 1989.

[24] Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In Stephen Jose Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems 5, [NIPS Conference, Denver, Colorado, USA, November 30 - December 3, 1992]*, pages 164–171. Morgan Kaufmann, 1992. URL http://papers.nips.cc/paper/647-second-order-derivatives-for-network-pruning-optimal-brain-surgeon.

[25] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR, 2023. URL https://proceedings.mlr.press/v202/frantar23a.html.

[26] Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=PxoFut3dWW.

[27] Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization. *CoRR*, abs/2207.00112, 2022. doi: 10.48550/ARXIV.2207.00112. URL https://doi.org/10.48550/arXiv.2207.00112.

[28] Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. ASVD: activation-aware singular value decomposition for compressing large language models. *CoRR*, abs/2312.05821, 2023. doi: 10.48550/ARXIV.2312.05821. URL https://doi.org/10.48550/arXiv.2312.05821.

[29] Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: truncation-aware singular value decomposition for large language model compression. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=LNYIUouhdt.

[30] Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. SVD-LLM V2: optimizing singular value truncation for large language model compression. *CoRR*, abs/2503.12340, 2025. doi: 10.48550/ARXIV.2503.12340. URL https://doi.org/10.48550/arXiv.2503.12340.

[31] Simla Burcu Harma, Ayan Chakraborty, Elizaveta Kostenok, Danila Mishin, Dongho Ha, Babak Falsafi, Martin Jaggi, Ming Liu, Yunho Oh, Suvinay Subramanian, and Amir Yazdanbakhsh. Effective interplay between sparsity and quantization: From theory to practice. In *The Thirteenth International Conference on Learning Representations, ICLR*

2025, Singapore, April 24-28, 2025. OpenReview.net, 2025. URL https://openreview.net/forum?id=wJv4AIt4sK.

[32] Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. LLM-QAT: data-free quantization aware training for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of *Findings of ACL*, pages 467–484. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.26. URL https://doi.org/10.18653/v1/2024.findings-acl.26.

[33] Tanmaey Gupta, Hayden Prairie, Xiaoxia Wu, Reyna Abhyankar, Qingyang Wu, Austin Silveria, Pragaash Ponnusamy, Jue Wang, Ben Athiwaratkun, Leon Song, Tri Dao, Daniel Y. Fu, and Chris De Sa. Search your block floating point scales! *arXiv preprint arXiv:2605.12464*, 2026. doi: 10.48550/arXiv.2605.12464. URL https://arxiv.org/abs/2605.12464.

[34] Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, Dusan Stosic, Venmugil Elango, Maximilian Golub, Alexander Heinecke, Phil James-Roxby, Dharmesh Jani, Gaurav Kolhe, Martin Langhammer, Ada Li, Levi Melnick, Maral Mesmakhosroshahi, Andres Rodriguez, Michael Schulte, Rasoul Shafipour, Lei Shao, Michael Y. Siu, Pradeep Dubey, Paulius Micikevicius, Maxim Naumov, Colin Verilli, Ralph Wittig, Doug Burger, and Eric S. Chung. Microscaling data formats for deep learning. *CoRR*, abs/2310.10537, 2023. doi: 10.48550/ARXIV.2310.10537. URL https://doi.org/10.48550/arXiv.2310.10537.

[35] Llama Team. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783. URL https://doi.org/10.48550/arXiv.2407.21783.

[36] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020. URL https://jmlr.org/papers/v21/20-074.html.

[37] Guilherme Penedo, Hynek Kydlícek, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin A. Raffel, Leandro von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/370df50ccfdf8bde18f8f9c2d9151bda-Abstract-Datasets_and_Benchmarks_Track.html.

[38] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017,*

Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, 2017. URL https://openreview.net/forum?id=Byj72udxe.

[39] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8732–8740. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6399. URL https://doi.org/10.1609/aaai.v34i05.6399.

[40] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1472. URL https://doi.org/10.18653/v1/p19-1472.

[41] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457, 2018. URL http://arxiv.org/abs/1803.05457.

[42] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6239. URL https://doi.org/10.1609/aaai.v34i05.6239.