# CPU-NPU COOPERATIVE LM-HEAD QUANTIZATION FOR EFFICIENT ON-DEVICE LANGUAGE GENERATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

On-device language generation is constrained by memory capacity and bandwidth. Although low-bit quantization shrinks Transformer decoders, large vocabularies leave the language-model head (LM-head) costly to store and execute on NPUs. Aggressive LM-head compression can degrade generation quality, while retrieval-based acceleration can shift substantial work to the CPU. Efficient edge inference therefore requires reducing both LM-head storage and decoding cost without restricting the vocabulary. We introduce **Q2F**, a quantized coarse-to-fine framework that combines ultra-low-bit NPU scoring with lightweight CPU refinement using existing embeddings. Q2F exploits a key observation: low-bit heads can retain promising candidates even when their top-1 predictions are inaccurate. Refining only these candidates enables both greedy and stochastic decoding at approximately *2.2 bits per LM-head weight*. Across three models with FP16 decoders, Q2F closely matches fully FP16 baselines, with at most a 0.32-percentage-point decrease in average accuracy across four benchmarks. On Snapdragon 8 Elite Gen 5, Q2F reduces LM-head weight storage by 73% and increases end-to-end generation throughput by 19.6% relative to an INT8-head baseline for Llama3.2-1B with a 4-bit decoder, while adding little CPU refinement overhead.

## 1 INTRODUCTION

Small language models (SLMs) make on-device generation increasingly practical, but memory capacity and bandwidth still constrain deployment. Heterogeneous inference systems assign operations according to processor capabilities (Xu et al., 2025). In our target CPU–NPU setting, the CPU handles embedding lookup while the NPU executes the Transformer decoder and language-model head (LM-head) as compiled *computation graphs* (Qualcomm Technologies, 2026c;d). Although tied embeddings share the same pretrained weight matrix between input embeddings and the LM-head, their access patterns differ: input embedding lookup reads a single vocabulary row per step (making it suitable for the CPU), whereas the LM-head multiplies the hidden state by the full matrix to compute logits for all vocabulary tokens (Wang et al., 2025). This dense computation and weight traffic favor NPU execution, but storing the LM-head weights alongside the decoder increases the NPU computation graph's memory footprint, limiting deployable model and vocabulary sizes.

Aggressive decoder quantization (Frantar et al., 2022; Tseng et al., 2024; Lee et al., 2025) further increases a higher-precision LM-head's share of weight storage and traffic. Quantized Llama3.2, for example, uses 4-bit Transformer weights with an 8-bit LM-head (Meta, 2024). Large vocabularies amplify this burden: Gemma3-4B has approximately 671 million LM-head parameters (Gemma Team, 2025). With a 2-bit decoder and an FP16 LM-head, the LM-head alone accounts for about 59.7% of the total NPU weight footprint (Figure 1, left). Efficient edge decoding therefore calls for compressing the LM-head's output projection while preserving generation quality.

Existing approaches leave a gap between aggressive LM-head compression and efficient CPU–NPU decoding. Vocabulary trimming reduces storage by removing tokens, sacrificing vocabulary coverage (Ushio et al., 2023). ARCHead preserves accuracy through quantized components and low-rank correction, but its reported 3.7–3.9$\times$ packed-storage compression relative to BF16 remains far from the approximately 2-bit weight budget targeted here (Kocabay et al., 2026). FlashHead reduces computation through cluster-based retrieval and rescoring (Tranheden et al., 2026), but large candidate sets create a substantial refinement workload when gathering and rescoring execute on the CPU.



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

These trade-offs motivate a design that jointly targets ultra-low-bit LM-head storage and a small CPU refinement budget, without sacrificing vocabulary coverage.

We introduce **Q2F**, a quantized coarse-to-fine decoding framework for heterogeneous CPU–NPU execution. Our key observation is that a low-bit LM-head can retain high top-$K$ candidate recall despite low top-1 agreement with a high-precision reference (Figure 2). Q2F uses the NPU to score the full vocabulary at low precision, then refines only 64–128 candidates on the CPU using existing high-precision embedding rows (Figure 1, right). This preserves the original vocabulary without adding a high-precision LM-head matrix. The refined logits support both greedy and stochastic decoding; we analyze candidate recall and retained probability mass to assess candidate-selection quality.

To preserve candidate quality with efficient NPU execution, we develop **vocabulary-grouped mixed-precision quantization (VGMP)**. Each vocabulary token corresponds to an output channel with its own quantization scale. A Fisher-based score combines output probability and quantization error to select sensitive channels for 4-bit precision, while most channels use 2 bits. An offline permutation places the two precision groups in contiguous blocks for dense per-channel NPU kernels, avoiding the overhead of fine-grained group-wise quantization (Xu et al., 2025). Low-cost LM-head fine-tuning further improves candidate quality.

Across three models with FP16 decoders, Q2F uses 2.14–2.19 bits per LM-head weight with a maximum decrease of 0.32 percentage points in four-task average accuracy relative to FP16 baselines. On Snapdragon 8 Elite Gen 5, Q2F reduces LM-head weight storage by approximately 73% relative to an INT8 LM-head (which is *7.3*$\times$ compression over FP16 LM-head) and increases end-to-end generation throughput by 19.6% for Llama3.2-1B with a 4-bit decoder and $K = 64$ (Table 4). These measurements include candidate selection, refinement, and sampling; adding CPU refinement reduces throughput by only 0.56% relative to the same VGMP LM-head without refinement. Further experiments confirm that Q2F seamlessly scales to quantized decoders, maintaining high generation accuracy while preserving end-to-end deployment efficiency on resource-constrained devices.

Our contributions are:

- **Heterogeneous CPU–NPU LM-head decoding.** Q2F exploits high candidate recall to combine low-bit NPU scoring with selective CPU refinement using existing embedding rows, supporting both greedy and stochastic decoding.
- **Vocabulary-grouped mixed-precision LM-head.** We allocate higher precision to selected vocabulary channels and organize the LM-head into two contiguous precision blocks, preserving candidate quality at approximately 2 bits per weight while supporting dense per-channel NPU execution.
- **Quality analysis and integrated mobile validation.** We analyze candidate recall and sampling fidelity, evaluate downstream accuracy, and demonstrate substantial LM-head compression and end-to-end throughput gains on a mobile CPU–NPU system with little CPU refinement overhead.

## 2 Related Work

**LLM Inference on Heterogeneous Edge Processors.** Heterogeneous edge inference systems map LLM inference operations to edge processors (e.g., CPU and NPU) according to their computational strengths (Xu et al., 2025). In CPU–NPU deployments such as Qualcomm AI Runtime (QAIRT), the CPU handles control flow, preprocessing, and postprocessing, while the NPU executes the Transformer decoder and LM-head as compiled computation graphs (Qualcomm Technologies, 2026c;d). Many edge LLMs, including Llama3.2-1B/3B and Gemma3, use *tied embeddings*: the input embedding and LM-head share the same pretrained weight matrix (Meta, 2024; Gemma Team, 2025). Their access patterns nevertheless differ. At each decoding step, embedding lookup reads a single vocabulary row, making it suitable for the CPU. The LM-head's output projection multiplies the hidden state by the full matrix, with one output channel per vocabulary token, making it suitable for NPU acceleration (Wang et al., 2025). The NPU's computation graph thus handles the computation and memory traffic of both the decoder and LM-head. Their weights and execution buffers occupy substantial memory; reducing this footprint is essential for accommodating larger models and vocabularies.

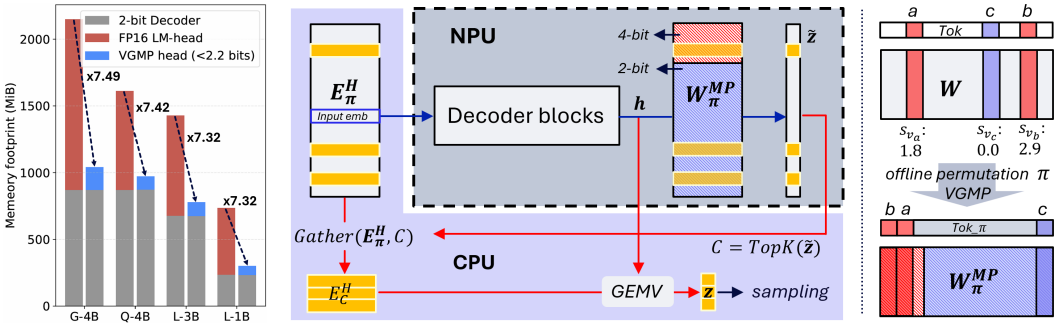


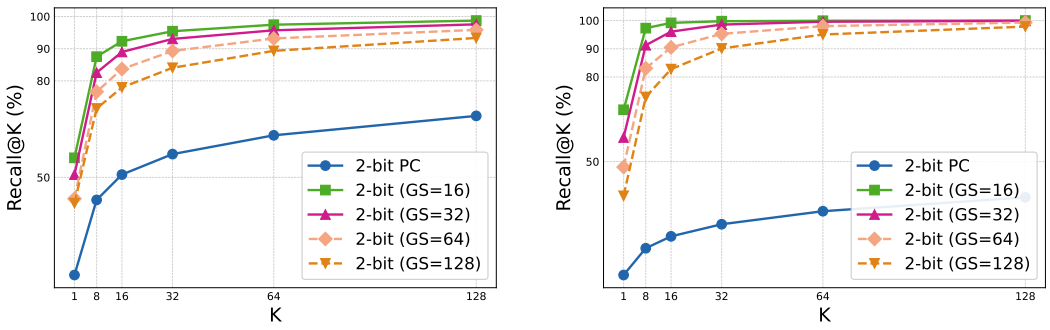

Figure 1: (Left) Estimated NPU weight footprint with a 2-bit decoder and an FP16 or VGMP LM-head; labels show head-only compression ratios. G-4B, Q-4B, L-3B, and L-1B denote Gemma3-4B, Qwen3-4B, Llama3.2-3B, and Llama3.2-1B. (Right) Q2F uses contiguous low-bit NPU projections for coarse scoring and existing CPU embedding rows for candidate refinement. $W_\pi^{\mathrm{MP}}$ and $E_\pi^H$ denote the mixed-precision head and higher-precision embedding matrices under a shared vocabulary permutation $\pi$, also applied to the tokenizer to preserve token semantics.

**Low-bit LLM Quantization.** Low-bit quantization reduces this memory footprint, with extensive post-training quantization (PTQ) research focused on decoder weights. Methods include activation-aware scaling in AWQ (Lin et al., 2024) and learned rounding in FlexRound (Lee et al., 2023). GPTQ uses approximate second-order information to compensate for quantization error (Frantar et al., 2022), while QuIP# combines Hadamard transforms with E8P lattice codebooks (Tseng et al., 2024). These methods' execution formats can complicate NPU deployment: fine-grained group-wise formats, including the GPTQ configurations evaluated here, require partial matrix multiplications and additional accumulation on our target backend, whereas QuIP# relies on specialized kernels for transforms and codebook decoding—operations that many mobile NPUs lack native support for (Xu et al., 2025). Furthermore, most quantization research has focused on decoder quantization, and LM-head compression has received comparatively less attention. As shown in Figure 1(Left), 2-bit decoder LLMs with a 16-bit LM-head leave the LM-head's output projection as a substantial remaining memory cost.

**Efficient LM-head Computation.** Vocabulary trimming reduces LM-head storage by removing tokens outside a target language (Ushio et al., 2023), while ARCHead preserves vocabulary coverage through quantized components and low-rank correction (Kocabay et al., 2026). Q2F targets more aggressive head quantization, achieving near-FP16 average downstream accuracy at approximately 2.2 bits per weight by reusing existing embeddings for candidate refinement. FlashHead accelerates the output projection through cluster-based retrieval, rescoring 8,192 candidates per step in its default Llama3.2-1B configuration (Tranheden et al., 2026). Q2F instead scores the full vocabulary at low precision and refines only 64–128 individual candidates. This smaller refinement budget is particularly relevant to CPU–NPU backends where dynamic gathering and rescoring execute on the CPU: our kernel-based extrapolation suggests substantial CPU cost at FlashHead's candidate budget (Appendix A.9).

## 3 Method

Q2F reduces LM-head memory and computation costs through two complementary components. First, it splits LM-head computation into two stages: a low-bit NPU projection identifies promising tokens, and the CPU refines their scores using its existing high-precision embedding table. The refined logits support both greedy and stochastic decoding. Second, selective mixed-precision quantization assigns 4 bits to sensitive vocabulary rows and 2 bits to the rest. A vocabulary permutation groups rows by precision, enabling efficient per-channel NPU kernels. Optional LM-head fine-tuning further improves candidate quality.




(a) Qwen3-4B (b) Gemma3-4B

Figure 2: Candidate recall against the FP16-head prediction on WikiText-2 test. Group-wise 2-bit quantization retains high recall despite low top-1 agreement, while uniform per-channel quantization yields lower recall. Complete results with 4 models appear in Appendix A.3.

### 3.1 CPU–NPU Quantized LM-Head Decoding

**Two-stage decoding with high candidate recall.** A low-bit LM-head can disagree with the high-precision top-1 prediction yet retain that token among its top-$K$ candidates (Figure 2). Q2F uses this high recall to separate candidate selection from accurate scoring. Let $E^H \in \mathbb{R}^{V\times d}$ denote the high-precision embedding table on the CPU and $W^L$ its low-bit quantized LM-head counterpart on the NPU, each with $V$ vocabulary rows and hidden dimension $d$. At decoding step $t$, the NPU scores the full vocabulary using the decoder's final hidden state $\mathbf{h}_t$. The CPU then selects the top-$K$ token indices and recomputes their logits using the corresponding vocabulary rows ($v$) of $E^H$ (Figure 1):

$$\begin{aligned} \widetilde{\mathbf{z}}_t &= W^L\mathbf{h}_t, \quad \mathcal{C}_t = \mathrm{TopK}(\widetilde{\mathbf{z}}_t, K), \\ z^H_{t,v} &= E^H_v\mathbf{h}_t, \qquad v \in \mathcal{C}_t. \end{aligned} \tag{1}$$

Greedy decoding selects the largest refined logit; stochastic decoding applies temperature and top-$k$/top-$p$ sampling to the refined logits within $\mathcal{C}_t$.

The NPU performs the $O(Vd)$ low-bit projection, while the CPU scans $V$ coarse logits and refines only $K \ll V$ candidates at $O(Kd)$ cost. The runtime exposes the hidden state and coarse logits to the CPU, which reuses existing embedding rows without an additional high-precision LM-head matrix. This keeps CPU overhead small: for Llama3.2-1B at $K = 128$, candidate selection and refinement take approximately 0.195 ms on the performance core, excluding sampling (Appendix A.9). Section 4.3 evaluates the complete decoding path, including sampling.

**Analysis 1: Candidate retention for greedy recovery.** Let $y^H_t = \arg\max_v E^H_v\mathbf{h}_t$ be the prediction obtained by scoring the full vocabulary with the refinement table. If this token is in $\mathcal{C}_t$, refinement recovers it under matched hidden states, scoring arithmetic, and tie-breaking:

$$y^H_t \in \mathcal{C}_t \quad\Longrightarrow\quad \widehat{y}_t = \arg\max_{v\in\mathcal{C}_t} z^H_{t,v} = y^H_t. \tag{2}$$

For a reference prediction $y^{\mathrm{ref}}_t = \arg\max_v z^{\mathrm{ref}}_{t,v}$, we measure candidate retention over $T$ matched positions as:

$$\mathrm{Recall@}\,K = \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}\left[y^{\mathrm{ref}}_t \in \mathcal{C}_t\right]. \tag{3}$$

Figure 2 shows that increasing $K$ from 1 to 128 raises recall for 2-bit GS64 LM-heads from 43.36% to 95.85% on Qwen3-4B and from 48.06% to 99.31% on Gemma3-4B. Thus, low top-1 agreement can coexist with high candidate recall, allowing CPU refinement to recover predictions that the quantized LM-head alone would miss. Uniform 2-bit per-channel quantization has lower Recall@128 (37.32%–69.11% across four models), motivating the mixed-precision design in Section 3.2.

Table 1: Gemma3-4B candidate retention on WikiText-2 test. Recall measures FP16-head top-1 inclusion; Mass is mean retained reference probability; Cov.$\geq 0.8$ is the fraction of positions retaining at least 80% mass. All three metrics are percentages. FP16 top-$K$ selection maximizes retained mass at each budget.

| LM-head configuration | Weight bits | $K = 64$ | | | $K = 128$ | | |
|---|---|---|---|---|---|---|---|
| | | Recall | Mass | Cov.$\geq 0.8$ | Recall | Mass | Cov.$\geq 0.8$ |
| FP16 baseline (full model) | 16.00 | 100.00 | 97.00 | 97.83 | 100.00 | 98.09 | 99.41 |
| RTN W2 (GS=64) | 2.25 | 98.02 | 92.59 | 87.78 | 99.31 | 95.53 | 94.34 |
| RTN W2 (PC) | 2.00 | 32.41 | 31.47 | 20.95 | 37.32 | 36.68 | 24.64 |
| VGMP + FT (PC) | 2.14 | 98.35 | 94.36 | 92.13 | 98.87 | 96.06 | 95.76 |

**Analysis 2: Sampling-distribution retention.** Stochastic decoding requires retaining probability mass beyond the most likely token. Let $p_t(v) = \mathrm{softmax}(\mathbf{z}_t^{\mathrm{ref}}/\tau)_v$ be the reference distribution at temperature $\tau > 0$, before sampling filters, and let $m_t(K) = \sum_{v \in \mathcal{C}_t} p_t(v)$. We measure the mean retained mass and the fraction of positions retaining at least a threshold $0 < \rho \leq 1$:

$$\mathrm{Mass@}\, K = \frac{1}{T} \sum_{t=1}^{T} m_t(K), \tag{4}$$

$$\mathrm{MassCoverage@}(K, \rho) = \frac{1}{T} \sum_{t=1}^{T} \mathbb{1}[m_t(K) \geq \rho]\,. \tag{5}$$

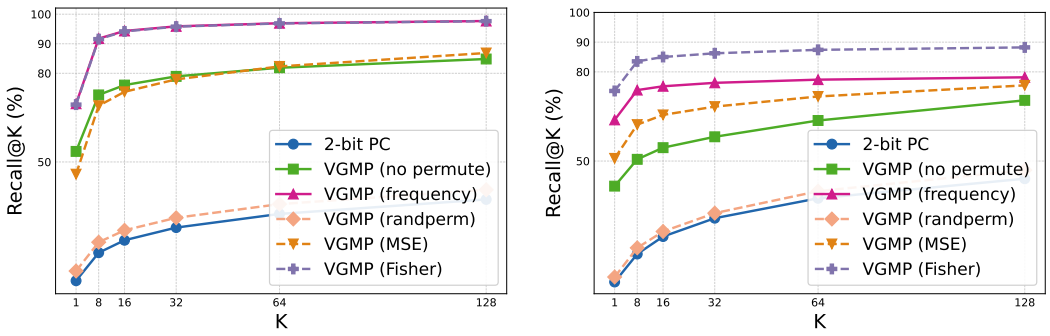
Restricting the reference distribution to the candidates and renormalizing gives a total variation distance of $1 - m_t(K)$ from the original distribution. MassCoverage therefore gives the fraction of positions with distance at most $1 - \rho$. These metrics characterize the contribution of candidate selection to sampling fidelity; refinement quantization and subsequent sampling filters can introduce additional differences. Table 1 shows that RTN W2 (GS64) preserves high candidate recall and substantial FP16 reference probability mass at $K = 128$, supporting Q2F's use of a small candidate set even with a 2-bit coarse head.

However, fine-grained group-wise scaling complicates deployment on our target NPU backend by requiring additional partial projections and accumulation (Xu et al., 2025). Simply switching to uniform 2-bit per-channel quantization reduces retained mass to 36.68% at the same candidate budget (Table 1). This trade-off motivates vocabulary-grouped mixed precision (VGMP), introduced in Section 3.2, which selectively protects sensitive vocabulary channels to preserve candidate quality while enabling dense per-channel NPU execution.

### 3.2 Vocabulary-Grouped Mixed-Precision LM-Head Quantization

**Precision allocation across vocabulary channels.** Each LM-head output channel corresponds to one vocabulary token, allowing precision to be assigned independently across tokens. We exploit this structure with *vocabulary-grouped mixed-precision quantization* (VGMP): a small set of sensitive channels uses intermediate bit-width, while the remaining majority uses low-bits. Each channel retains its own quantization scale, and channels of the same precision form a contiguous block for dense NPU execution. This preserves the efficiency of per-channel quantization while protecting channels important for candidate recall.

**Fisher-based vocabulary importance.** The choice of channels to protect determines candidate quality. Token frequency estimates how often a token appears in calibration outputs but ignores its sensitivity to quantization. Logit MSE measures quantization error but treats errors on unlikely and likely tokens equally. We propose a *Fisher-based score* that combines output probability with squared logit error, prioritizing tokens that are both likely in calibration contexts and sensitive to quantization. Let $E$ denote the original FP16 embedding matrix and $Q_b$ denote $b$-bit per-channel quantization. For $N$ calibration hidden states, define $\mathbf{p}_i = \mathrm{softmax}(E\mathbf{h}_i)$ and $\boldsymbol{\delta}_i = (E - Q_2(E))\mathbf{h}_i$. The softmax Fisher matrix $F_i = \mathrm{diag}(\mathbf{p}_i) - \mathbf{p}_i\mathbf{p}_i^\top$ gives the second-order KL approximation $\frac{1}{2}\boldsymbol{\delta}_i^\top F_i \boldsymbol{\delta}_i$. We use




(a) WikiText-2 test

(b) AIME 2024 generated responses

Figure 3: Gemma3-4B candidate recall under different VGMP row-selection rules. Scores use WikiText-2 train for calibration, and assignments remain fixed across WikiText-2 test and generated AIME 2024 responses. Each VGMP assignment promotes $M = 18$K rows to 4 bits.

its separable probability-weighted squared-error term to score each vocabulary channel:

$$s_v^{\text{Fisher}} = \frac{1}{N} \sum_{i=1}^{N} p_{i,v} \delta_{i,v}^2. \tag{6}$$

The highest-scoring eligible channels fill the unreserved positions in the $M$-row 4-bit block; reserved token IDs remain fixed (Appendix A.9). Unlike raw token counts, the probability weights also capture plausible alternative tokens, helping protect candidates beyond those observed in the calibration text.

**Contiguous implementation.** An offline permutation $\pi$ places the selected channels first, followed by the 2-bit channels. With $\pi(v)$ denoting the new zero-based index of token $v$, the precision groups are:

$$\mathcal{H}_4 = \{v : \pi(v) < M\}, \qquad \mathcal{H}_2 = \{v : \pi(v) \geq M\}. \tag{7}$$

Let $W$ denote the floating-point LM-head weights initialized from $E$ and optionally updated during fine-tuning. The deployed LM-head is:

$$(W^L)_{\pi(v)} = (W_\pi^{\text{MP}})_{\pi(v)} = \begin{cases} Q_4(W_v), & v \in \mathcal{H}_4, \\ Q_2(W_v), & v \in \mathcal{H}_2. \end{cases} \tag{8}$$

The two contiguous blocks each use a dense per-channel NPU kernel, avoiding dispatch for individual channels. Average weight precision is $2 + 2M/V$ bits, excluding metadata. Applying the same permutation to the embedding table and tokenizer mappings preserves token semantics and direct CPU candidate lookup. Fisher scoring identifies high-precision channels, enabling the required vocabulary permutation to be computed *off-line just once* for the model before deployment.

**Analysis 1: Channel-selection quality.** Figure 3 compares Fisher, token frequency, logit MSE, random selection, and original vocabulary order on Gemma3-4B with FP16 decoders and INT8 per-tensor input embeddings. On WikiText-2 test, Fisher raises Recall@128 from 37.32% for uniform 2-bit per-channel quantization and 40.58% for random selection to 97.65%, showing that selective protection recovers candidate quality with few higher-precision channels.

**Analysis 2: Generalization under distribution shift.** We keep the WikiText-calibrated assignments fixed and evaluate on generated AIME 2024 responses. Fisher and frequency perform similarly on WikiText-2 test (97.65% vs. 97.69% Recall@128), but Fisher reaches 88.23% on AIME, compared with 78.17% for frequency and 75.46% for logit MSE (Figure 3b). Fisher is the best of the tested scoring rules on this out-of-distribution corpus, exceeding frequency by 10.06 percentage points. This result supports combining output probability and quantization error to protect candidates beyond the calibration distribution. Appendix A.3 reports the complete comparisons.

Table 2: Downstream accuracy (%). The baseline is fully FP16. Quantized LM-heads use an FP16 decoder and INT8 per-tensor input embeddings. Gray rows add Q2F refinement or further VGMP on this setting. FT denotes one epoch of head-only fine-tuning; Avg. is the mean four-task accuracy. BW indicates LM-head's bitwidth. (Results with 4-bit PTQ, 2-bit QAT, and 2/4-bit QAT decoders appear in Appendices A.5 and A.6.)

| Model | LM-head quantization config. | BW | MMLU-Pro | AIME | IFEval | GSM8K | Avg. |
|---|---|---|---|---|---|---|---|
| Llama3.2-3B | FP16 baseline (full model) | 16.00 | 35.46 | 4.39 | 74.34 | 66.57 | 45.19 |
| | RTN W2 (GS64) | 2.25 | 9.75 | 0.64 | 47.72 | 6.14 | 16.06 |
| | RTN W2 (GS64) + Q2F ($K = 64$) | 2.25 | 35.65 | 4.61 | 74.22 | 66.86 | 45.34 |
| | VGMP + FT | 2.19 | 32.48 | 3.53 | 46.76 | 24.94 | 26.93 |
| | VGMP + FT + Q2F ($K = 128$) | 2.19 | 38.02 | 6.65 | 75.78 | 63.00 | 45.86 |
| Gemma3-4B | FP16 baseline (full model) | 16.00 | 41.65 | 18.33 | 83.21 | 80.67 | 55.97 |
| | RTN W2 (GS64) | 2.25 | 12.01 | 0.11 | 65.35 | 22.67 | 25.04 |
| | RTN W2 (GS64) + Q2F ($K = 64$) | 2.25 | 41.57 | 19.51 | 83.57 | 80.36 | 56.25 |
| | VGMP + FT | 2.14 | 39.20 | 14.68 | 76.37 | 78.46 | 52.18 |
| | VGMP + FT + Q2F ($K = 128$) | 2.14 | 41.80 | 19.94 | 82.97 | 79.52 | 56.06 |
| Qwen3-4B | FP16 baseline (full model) | 16.00 | 66.31 | 49.62 | 88.85 | 92.57 | 74.34 |
| | RTN W2 (GS64) | 2.25 | 38.46 | 20.04 | 66.79 | 89.61 | 53.73 |
| | RTN W2 (GS64) + Q2F ($K = 64$) | 2.25 | 66.64 | 48.77 | 88.84 | 91.66 | 73.98 |
| | VGMP + FT | 2.16 | 65.23 | 37.83 | 68.58 | 78.69 | 62.58 |
| | VGMP + FT + Q2F ($K = 128$) | 2.16 | 66.58 | 48.56 | 88.37 | 92.57 | 74.02 |

### 3.3 LM-HEAD FINE-TUNING

Fine-tuning updates only the low-bit LM-head weights and quantization scales, keeping the decoder, input embedding, and precision assignment fixed. Restricting gradient updates to this single projection avoids backpropagation through the decoder and keeps the cost low: fine-tuning takes only a few minutes in our setup. Using calibration hidden states, we match the frozen FP16 LM-head's output distribution at temperature 1:

$$\mathcal{L}_{\text{head}} = \frac{1}{N} \sum_{i=1}^{N} D_{\text{KL}}\big(\text{softmax}(E_\pi \mathbf{h}_i) \,\big\|\, \text{softmax}(W_\pi^{\text{MP}} \mathbf{h}_i)\big)\,. \tag{9}$$

Because the CPU embedding remains fixed, fine-tuning changes candidate selection without changing refinement scores or the recovery condition in Equation 2. At $K = 64$, fine-tuning improves the refined 4-task average from 72.15 to 73.78 on Qwen3-4B and from 54.57 to 55.68 on Gemma3-4B (Table 14; Appendix A.8).

## 4 EXPERIMENTAL RESULTS

### 4.1 SETUP

**Downstream evaluation.** We evaluate Q2F on Qwen3-4B-Instruct-2507, Gemma3-4B-it, Llama3.2-3B-Instruct, and Llama3.2-1B-Instruct [1]. We report accuracy on MMLU-Pro (Wang et al., 2024), GSM8K (Cobbe et al., 2021), 933 AIME problems from 1983–2024 (Veeraboina, 2024), and IFEval prompt-level strict accuracy (Zhou et al., 2023). *Avg.* is the unweighted mean of these 4 scores. We use stochastic decoding (`do_sample=True`), with temperature, top-$k$, and top-$p$ from the model card fixed across head configurations within each model (e.g., 0.7, 20, and 0.8, respectively, for Qwen3-4B).

**Precision configurations.** For the main downstream comparisons and their supplementary results, the FP16 baseline uses a fully unquantized model, including its decoder, input embeddings, and LM-

[1] For brevity, we omit instruction-tuning suffixes and model release identifiers throughout the paper. Specifically, Qwen3-4B, Gemma3-4B, Llama3.2-3B, and Llama3.2-1B refer to Qwen3-4B-Instruct-2507, Gemma3-4B-it, Llama3.2-3B-Instruct, and Llama3.2-1B-Instruct, respectively. All WikiText experiments use WikiText-2 with the `wikitext-2-raw-v1` configuration.

Table 3: Downstream task average accuracy (%) and coarse-head kernel requirements for varying LM-head quantization methods. ◯: per-channel; △: group-wise; ×: Hadamard transforms and E8P decoding. Q2F adds CPU selection and refinement with $K = 64$ for RTN and $K = 128$ for VGMP (all use FP16 decoders and INT8 per-tensor input embeddings).

| LM-head quantization config. | BW | Llama3.2-3B | Gemma3-4B | Qwen3-4B | NPU |
|---|---|---|---|---|---|
| RTN W4 (PC) | 4.00 | 43.53 | 50.49 | 74.65 | ◯ |
| GPTQ W2 (GS64) | 2.25 | 19.14 | 28.67 | 68.30 | △ |
| QuIP# (E8P + FT) | 2.00 | 42.50 | 55.24 | 73.26 | × |
| RTN W2 (GS64) + Q2F | 2.25 | 45.34 | 56.25 | 73.98 | △ |
| VGMP + FT + Q2F | 2.14–2.19 | 45.86 | 56.06 | 74.02 | ◯ |

head. To isolate the impact of aggressive low-bit LM-head quantization from decoder degradation, quantized LM-head configurations share an FP16 decoder and INT8 per-tensor input embeddings unless otherwise specified. The INT8 embedding format matches our target edge deployment; Q2F reuses this embedding table for candidate refinement. On-device configurations in Table 4 instead use a 4-bit per-channel decoder, retaining INT8 input embeddings. Throughout the tables, BW denotes nominal bits per LM-head weight, with metadata accounting detailed in Appendix A.1.

**Quantization and calibration.** We compare round-to-nearest quantization (RTN), GPTQ (Frantar et al., 2022), QuIP# (Tseng et al., 2024), and vocabulary-grouped mixed precision (VGMP; Section 3.2). Unless otherwise specified, VGMP uses Fisher-based selection to promote $M = 12\text{K}$ rows to 4 bits for Qwen3-4B and both Llama3.2 models, and $M = 18\text{K}$ for Gemma3-4B; the remaining rows use 2-bit per-channel quantization. For downstream evaluation, GPTQ calibration, QuIP# Hessian estimation and fine-tuning, and VGMP row selection and fine-tuning share the same approximately five-million-token Tulu-3 SFT subset (Lambert et al., 2025), excluding downstream evaluation examples. FT denotes LM-head fine-tuning; training details appear in Appendix A.2.

**Candidate evaluation.** Recall@$K$ measures inclusion of the FP16-head top-1 token in the coarse candidate set at matched hidden states. We use teacher-forced WikiText-2 test prefixes (Merity et al., 2016) with GPTQ preprocessing (Frantar et al., 2022). For row-selection comparisons, assignments calibrated on WikiText-2 train remain fixed across WikiText-2 test and Gemma3-4B-generated responses to AIME 2024 problems; teacher-forcing the latter evaluates transfer to mathematical reasoning text. The retained-mass analysis uses Tulu-3 SFT calibration for VGMP selection and fine-tuning, matching downstream evaluation. Reference probabilities come from the FP16 head at matched hidden states, using the model's downstream temperature before top-$k$/top-$p$ filtering. Mass coverage uses $\rho = 0.8$.

### 4.2 Downstream Task Evaluation

**Q2F and VGMP recover downstream quality.** Table 2 shows that refining 64 candidates from group-wise 2-bit RTN heads recovers downstream average score within 0.36 percentage points of the fully FP16 baselines, despite poor standalone head accuracy. VGMP + FT + Q2F at $K = 128$ achieves averages within 0.67 points of these baselines at 2.14–2.19 bits per head weight. Task-level trade-offs remain: Llama3.2-3B improves from 35.46 to 38.02 on MMLU-Pro but drops from 66.57 to 63.00 on GSM8K. Appendix A.4 provides full baselines, candidate-budget sweeps, and Llama3.2-1B results; Table 14 confirms refinement benefits both with and without head fine-tuning.

**Comparison with alternative LM-head quantization.** Both Q2F configurations exceed QuIP#'s reported average accuracies across all three models (Table 3). VGMP supports contiguous per-channel kernels, whereas group-wise heads require group scaling and QuIP# requires transforms and codebook decoding. These execution requirements motivate VGMP for our target NPU. Q2F also benefits other coarse heads: with GS256 and $K = 8$, GPTQ and fine-tuned heads reach 73.88 and 74.07 on Qwen3-4B, respectively, versus the FP16 baseline of 74.34 (Table 13, Appendix A.7).

Table 4: Llama3.2-1B downstream accuracy and on-device efficiency with a 4-bit per-channel decoder and INT8 per-tensor input embeddings. NPU graph memory is in MiB; Avg. acc. is the four-task average (%). Throughput per second (TPS) reports the mean ± standard deviation over five on-device runs of 2,500 generated tokens each. VGMP promotes $M = 12\text{K}$ rows.

| LM-head quantization config. | NPU Mem | Avg. acc. (%) | TPS (Tokens/sec) |
|---|---|---|---|
| 8-bit | 743.51 | 23.56 | $59.91 \pm 0.85$ |
| 4-bit | 617.69 | 23.17 | $67.39 \pm 1.21$ |
| 2-bit | 554.71 | 6.40 | $72.41 \pm 0.67$ |
| VGMP + FT | 560.60 | 21.25 | $72.03 \pm 0.69$ |
| VGMP + FT + Q2F ($K = 32$) | 561.07 | 23.04 | $71.67 \pm 0.67$ |
| VGMP + FT + Q2F ($K = 64$) | 561.07 | 23.53 | $71.63 \pm 0.68$ |
| VGMP + FT + Q2F ($K = 128$) | 561.07 | 23.99 | $71.34 \pm 0.54$ |

### 4.3 On-Device Efficiency

We deploy Llama3.2-1B on Snapdragon 8 Elite Gen 5 (`SM8850-AC`) using QAIRT (Qualcomm Technologies, 2026b) and a custom Genie sampler (Qualcomm Technologies, 2026a). The NPU executes the 4-bit per-channel decoder and quantized LM-head. A CPU performance core selects candidates, gathers rows from the existing INT8 per-tensor embedding table, computes integer dot products, restores scales, and samples. The reference sampler uses temperature 1.0 and top-$p = 1.0$; all throughput measurements include sampling. Appendix A.9 details integration, profiling, and CPU microbenchmarks.

**Throughput and memory.** At $K = 64$, VGMP + FT + Q2F obtains 23.53% average accuracy, 0.03 percentage points below the 8-bit-head reference, and generates 71.63 tokens/s. Relative to this baseline with the same 4-bit decoder, throughput increases by 19.6% and NPU graph memory falls from 743.51 to 561.07 MiB (24.5%). Adding Q2F to VGMP + FT reduces mean throughput by only 0.56%; the reduction remains below 1% at $K = 128$. This net overhead includes changes in sampling cost (Appendix A.9).

**Comparison with prior LM-head optimization.** Our exploratory selective low-rank variant reduced NPU memory but achieved lower throughput than row promotion (Appendix A.10). FlashHead's default Llama3.2-1B configuration rescores 8,192 candidates (Tranheden et al., 2026). On our backend, dynamic gathering and rescoring run on the CPU; extrapolating our gather-and-GEMV measurements to this budget gives approximately 4.1 ms, motivating Q2F's $K = 32$–$128$ budgets. This estimate covers candidate processing with our kernels, not end-to-end FlashHead execution, which also depends on centroid scoring, cluster layout, and sampling (Appendix A.9).

## 5 Limitations and Future Work

Q2F's embedding reuse relies on pretrained tied input and output weights and compatible inference representations. Transformations such as final-normalization fusion in SpinQuant (Liu et al., 2025) can alter this correspondence, requiring adaptations to the refinement path. Our current implementation targets models that preserve this compatibility. We evaluate standard autoregressive decoding; integration with speculative decoding (Leviathan et al., 2023) and its effects on generation quality and throughput remain directions for future work.

## 6 Conclusion

We presented Q2F, which combines ultra-low-bit NPU candidate scoring with selective CPU refinement using existing tied embeddings. Fisher-weighted row promotion and offline permutation support candidate retention and contiguous per-channel execution. Across evaluated SLMs, Q2F maintains near-reference downstream accuracy with approximately 2-bit heads. On-device evaluation demonstrates faster decoding with little refinement overhead, highlighting the value of designing LM-head quantization around candidate preservation and heterogeneous execution.

## Disclosure of AI Use

We used generative AI to assist with manuscript drafting and editing, literature discovery, and the interpretation and presentation of results. We also used generative AI to help prepare portions of the supplementary code from author-written research implementations. The research ideas, method development, and experimental design, implementation, and execution were carried out by the human authors. The authors reviewed the AI-assisted content and take full responsibility for the final manuscript, including its methods, results, and claims.

## References

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers, 2022. URL https://arxiv.org/abs/2210.17323.

Gemma Team. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Şuayp Talha Kocabay, Talha Rüzgar Akkuş, and Kamer Ali Yuksel. ARCHead: Activation-metric residual correction for large language model output heads, 2026. URL https://arxiv.org/abs/2608.02703.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Proceedings of the Conference on Language Modeling (COLM)*, 2025.

Jung Hyun Lee, Jeonghoon Kim, Se Jung Kwon, and Dongsoo Lee. Flexround: learnable rounding based on element-wise division for post-training quantization. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Jung Hyun Lee, Seungjae Shin, Vinnam Kim, Jaeseong You, and An Chen. Unifying block-wise ptq and distillation-based qat for progressive quantization toward 2-bit instruction-tuned llms, 2025. URL https://arxiv.org/abs/2506.09104.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Junxian Guo, Xiuyu Li, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. SVDQuant: Absorbing outliers by low-rank component for 4-bit diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=vWR3KuiQur.

Yixiao Li, Yifan Yu, Chen Liang, Nikos Karampatziakis, Pengcheng He, Weizhu Chen, and Tuo Zhao. Loftq: LoRA-fine-tuning-aware quantization for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=LzPWWPAdY4.

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 llm compression and acceleration. In *MLSys*, 2024.

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*, 2025. URL https://openreview.net/forum?id=ogO6DGE6FZ.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Meta. Llama 3.2 model card. Official model card, 2024. URL https://github.com/meta-llama/llama-models/blob/main/models/llama3_2/MODEL_CARD.md. Model Information and Quantization sections. Accessed September 22, 2026.

Qualcomm Technologies. Qualcomm Gen AI Inference Extensions. Official deployment documentation, 2026a. URL https://www.qualcomm.com/developer/software/gen-ai-inference-extensions.

Qualcomm Technologies. Qualcomm AI Runtime SDK. Official deployment documentation, 2026b. URL https://www.qualcomm.com/developer/software/qualcomm-ai-runtime-sdk-qairt.

Qualcomm Technologies. GenieX QAIRT Plugin: CPU-side embedding input provider. Official runtime source code, 2026c. URL https://github.com/qualcomm/geniex-qairt-plugin/blob/main/core/include/llm/input_provider.h. Accessed September 22, 2026.

Qualcomm Technologies. Qualcomm AI Hub Models: LLM onboarding. Official deployment documentation, 2026d. URL https://github.com/qualcomm/ai-hub-models/blob/main/tutorials/llm/onboarding.md. Parts Splitting and Compile and Link sections. Accessed September 22, 2026.

Rajarshi Saha, Naomi Sagan, Varun Srivastava, Andrea Goldsmith, and Mert Pilanci. Compressing large language models using low rank and low precision decomposition. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=lkx3OpcqSZ.

Wilhelm Tranheden, Shahnawaz Ahmed, Devdatt Dubhashi, Jonna Matthiesen, and Hannes von Essen. FlashHead: Efficient drop-in replacement for the classification head in language model inference, 2026. URL https://arxiv.org/abs/2603.14591.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better LLM quantization with hadamard incoherence and lattice codebooks, 2024. URL https://arxiv.org/abs/2402.04396.

Asahi Ushio, Yi Zhou, and Jose Camacho-Collados. Efficient multilingual language model compression through vocabulary trimming. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 14725–14739, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.981. URL https://aclanthology.org/2023.findings-emnlp.981/.

Hemish Veeraboina. Aime problem set 1983-2024, 2024. URL https://www.kaggle.com/datasets/hemishveeraboina/aime-problem-set-1983-2024.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 95266–95290. Curran Associates, Inc., 2024. doi: 10.52202/079017-3018. URL https://arxiv.org/pdf/2406.01574.

Zhaode Wang, Jingbang Yang, Xinyu Qian, Shiwen Xing, Xiaotang Jiang, Chengfei Lv, and Shengyu Zhang. MNN-LLM: A generic inference engine for fast large language model deployment on mobile devices, 2025. URL https://arxiv.org/abs/2506.10443.

Daliang Xu, Hao Zhang, Liming Yang, Ruiqi Liu, Gang Huang, Mengwei Xu, and Xuanzhe Liu. Fast on-device LLM inference with NPUs. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2025. doi: 10.1145/3669940.3707239. URL https://arxiv.org/abs/2407.05858.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Table 5: FP16-target candidate recall (%) on WikiText-2 test. PC denotes per-channel quantization and GS denotes group size. R@$K$ is Recall@$K$ from Equation 3.

| | Qwen3-4B | | | Llama3.2-3B | | |
|---|---|---|---|---|---|---|
| Method | R@1 | R@32 | R@128 | R@1 | R@32 | R@128 |
| 4-bit PC | 86.34 | 100.00 | 100.00 | 88.14 | 99.98 | 100.00 |
| 2-bit PC | 19.66 | 57.23 | 69.11 | 13.25 | 36.15 | 44.31 |
| 2-bit (GS32) | 50.87 | 93.03 | 97.56 | 65.41 | 99.72 | 99.97 |
| 2-bit (GS64) | 43.36 | 89.28 | 95.85 | 60.05 | 98.65 | 99.90 |
| 2-bit (GS128) | 41.97 | 84.06 | 93.33 | 51.05 | 95.52 | 98.60 |
| | Gemma3-4B | | | Llama3.2-1B | | |
| Method | R@1 | R@32 | R@128 | R@1 | R@32 | R@128 |
| 4-bit PC | 73.72 | 99.20 | 99.96 | 88.64 | 99.99 | 100.00 |
| 2-bit PC | 9.80 | 27.79 | 37.32 | 15.14 | 43.05 | 55.09 |
| 2-bit (GS32) | 58.55 | 98.57 | 99.90 | 56.45 | 99.09 | 99.91 |
| 2-bit (GS64) | 48.06 | 95.28 | 99.31 | 49.68 | 96.59 | 99.45 |
| 2-bit (GS128) | 37.86 | 90.11 | 97.86 | 43.50 | 91.51 | 95.89 |

# A Appendix

## A.1 Bitwidth accounting.

The BW column reports nominal bits per LM-head weight. For group-wise scalar quantization, BW includes group-scale overhead: a 2-bit GS64 head with one 16-bit scale per group uses $2 + 16/64 = 2.25$ bits per weight. For VGMP, BW counts weight values only, giving $2 + 2M/V$ bits on average. Per-channel scale overhead depends on the hidden dimension $d$: one 16-bit scale per vocabulary row adds $16/d$ bits per weight, or approximately 0.008 bits at $d = 2048$. We omit this small overhead from nominal BW. QuIP# is reported at its nominal 2-bit E8P code precision. Codebook metadata and input embeddings are also excluded.

## A.2 Fine-Tuning Details

Configurations marked FT optimize the coarse-head weights and quantization scales using Equation 9, while keeping the input embedding and row-promotion assignment frozen. For experiments with a full-precision decoder, both the decoder and input embedding remain full precision and frozen throughout fine-tuning. For experiments with a quantized decoder, the decoder is quantized during fine-tuning but remains frozen, such that only the coarse-head weights and quantization scales are updated. We use the Adam optimizer with learning rates of $10^{-5}$ for the head weights and $10^{-4}$ for the quantization scales. Fine-tuning runs for one epoch over 1,200 batches with a batch size of 1 and a sequence length of 4,096, total approximately 4.92 million token positions from the Tulu-3 SFT dataset. Training takes 5–10 minutes on a single NVIDIA A100 GPU.

## A.3 Complete Candidate-Recall Results

Table 5 reports FP16-target Recall@$K$ on WikiText-2 test for all four models. These measurements use teacher-forced prefixes and compare each quantized head with the FP16 head at the same hidden states. At 2 bits, finer groups substantially improve candidate recall: GS64 retains the FP16 top-1 token within 128 candidates at 95.85%–99.90% of positions, compared with 37.32%–69.11% for per-channel quantization. The gap between Recall@1 and Recall@128 motivates refinement: the reference prediction can remain recoverable even when the coarse head's top-1 prediction changes.

**Row selection under distribution shift.** Table 6 compares mixed-precision assignments on WikiText-2 test and a fixed corpus of responses generated by FP16 Gemma3-4B on AIME 2024. The latter is a recall-evaluation corpus, subset of the AIME benchmark (1983-2024) used for downstream task accuracy. All row-selection statistics are calibrated on WikiText-2 train, and the resulting assign-

Table 6: FP16-target candidate recall (%) for mixed-precision row assignments calibrated on WikiText-2 train. WikiText-2 test is the in-distribution corpus; AIME uses generated responses to AIME 2024 problems. VGMP promotes $M = 18$K rows for Gemma3-4B and $M = 12$K for Llama3.2-1B.

| | WikiText-2 test | | | AIME responses | | |
|---|---|---|---|---|---|---|
| Method | Recall@1 | Recall@32 | Recall@128 | Recall@1 | Recall@32 | Recall@128 |
| **Gemma3-4B** | | | | | | |
| 2-bit per-channel | 9.80 | 27.79 | 37.32 | 9.31 | 30.77 | 44.05 |
| VGMP (random) | 13.09 | 31.10 | 40.58 | 10.99 | 32.47 | 46.95 |
| VGMP (frequency) | 69.69 | 95.89 | 97.69 | 63.84 | 76.31 | 78.17 |
| VGMP (no permutation) | 53.57 | 78.96 | 84.84 | 41.56 | 58.16 | 70.42 |
| VGMP (logit MSE) | 45.87 | 77.95 | 86.87 | 50.90 | 68.31 | 75.46 |
| VGMP (Fisher) | 69.40 | 95.79 | 97.65 | 73.58 | 86.25 | 88.23 |
| **Llama3.2-1B** | | | | | | |
| 2-bit per-channel | 15.14 | 43.05 | 55.09 | 32.44 | 53.98 | 59.36 |
| VGMP (random) | 17.31 | 43.66 | 56.43 | 25.16 | 65.00 | 72.85 |
| VGMP (frequency) | 81.44 | 93.45 | 95.01 | 61.38 | 83.99 | 87.89 |
| VGMP (no permutation) | 73.15 | 86.19 | 90.49 | 81.55 | 89.85 | 93.48 |
| VGMP (logit MSE) | 58.28 | 78.97 | 88.06 | 52.32 | 85.12 | 93.20 |
| VGMP (Fisher) | 80.76 | 93.50 | 95.54 | 68.93 | 90.03 | 94.48 |

ments are held fixed on both evaluation corpora. Promotion budgets are $M = 18$K for Gemma3-4B and $M = 12$K for Llama3.2-1B. Frequency, logit MSE, and Fisher select ordinary-token rows according to their respective scores; random allocation selects those rows randomly. Special-token IDs remain fixed under every assignment. The original-order baseline, labeled *No permutation*, assigns 4 bits to the first $M$ token IDs and 2 bits to the remainder. It therefore changes the allocation of precision relative to score-based selection, rather than isolating the effect of layout with the promoted rows held constant.

For Gemma3-4B, Fisher and frequency selection achieve similar WikiText-2 Recall@128: 97.65% and 97.69%, respectively. Fisher achieves higher Recall@128 on the AIME response corpus, at 88.23% versus 78.17%. On Llama3.2-1B, Fisher achieves the highest AIME Recall@128 among the compared assignments, at 94.48%, although the original-order baseline has higher Recall@1 (81.55% versus 68.93%). Thus, the criterion that best preserves the coarse top-1 prediction need not give the best candidate recall at the refinement budget.

### A.4 Additional Downstream Results

Tables 7–9 provide additional quantization baselines and candidate-size sweeps for the three models in Table 2. Quantized LM-head configurations use an FP16 decoder and INT8 input embeddings, while the reference baseline is fully FP16, including the decoder, input embeddings, and LM-head. Generation settings are fixed across configurations within each model. Scores are MMLU-Pro accuracy, accuracy on the 933 AIME problems from 1983–2024, IFEval prompt-level strict accuracy, and GSM8K accuracy. Avg. is their unweighted mean. VGMP uses Fisher row selection and one epoch of head-only fine-tuning for these three tables.

**Candidate-set size.** For Gemma3-4B, the average score increases from 55.28 at $K = 32$ to 55.68 at $K = 64$ and 56.06 at $K = 128$, compared with 55.97 for the FP16-head reference. Qwen3-4B remains close to its 74.34 reference score across the reported budgets, with averages of 74.20, 73.78, and 74.02. Llama3.2-3B is more sensitive to the budget: its average rises from 42.62 at $K = 64$ to 45.86 at $K = 128$ and 46.48 at $K = 256$, compared with 45.19 for the reference. Increasing $K$ expands the candidate set at a given hidden state, but downstream scores need not increase monotonically because sampling and the generated trajectories also change.

**Llama3.2-1B.** Table 10 reports simulated downstream accuracy for Llama3.2-1B using the same promotion budget $M = 12$K as the device study. The decoder is FP16 and the input embedding table is INT8; these are not on-device accuracy measurements. The VGMP head scores 24.40 without refinement, while Q2F reaches 29.09, 28.96, and 28.77 at $K = 32$, 64, and 128, respectively,

Table 7: Additional baselines and candidate-size sweep for Gemma3-4B-it. All scores are percentages, with an FP16 decoder and INT8 input embeddings. VGMP promotes $M = 18$K rows and uses head-only fine-tuning. PC denotes per-channel quantization; GS64 denotes group size 64. BW follows Appendix A.1.

| LM-head configuration | BW | MMLU-Pro | AIME | IFEval | GSM8K | Avg. |
|---|---|---|---|---|---|---|
| FP16 baseline (full model) | 16.00 | 41.65 | 18.33 | 83.21 | 80.67 | 55.97 |
| RTN W4 (PC) | 4.00 | 36.65 | 9.32 | 81.53 | 74.45 | 50.49 |
| RTN W2 (PC) | 2.00 | 2.14 | 0.00 | 56.35 | 0.00 | 14.62 |
| RTN W2 (GS64) | 2.25 | 12.01 | 0.11 | 65.35 | 22.67 | 25.04 |
| GPTQ W2 (GS64) | 2.25 | 19.28 | 0.00 | 66.19 | 29.19 | 28.67 |
| QuIP# (E8P + FT) | 2.00 | 41.19 | 17.36 | 82.13 | 80.28 | 55.24 |
| RTN W2 (GS64) + Q2F ($K = 64$) | 2.25 | 41.57 | 19.51 | 83.57 | 80.36 | 56.25 |
| VGMP + FT | 2.14 | 39.20 | 14.68 | 76.37 | 78.46 | 52.18 |
| VGMP + FT + Q2F ($K = 32$) | 2.14 | 41.70 | 18.01 | 81.41 | 79.98 | 55.28 |
| VGMP + FT + Q2F ($K = 64$) | 2.14 | 41.73 | 18.44 | 82.85 | 79.68 | 55.68 |
| VGMP + FT + Q2F ($K = 128$) | 2.14 | 41.80 | 19.94 | 82.97 | 79.52 | 56.06 |

Table 8: Additional baselines and candidate-size sweep for Qwen3-4B.

| LM-head configuration | BW | MMLU-Pro | AIME | IFEval | GSM8K | Avg. |
|---|---|---|---|---|---|---|
| FP16 baseline (full model) | 16.00 | 66.31 | 49.62 | 88.85 | 92.57 | 74.34 |
| RTN W4 (PC) | 4.00 | 66.47 | 50.48 | 89.45 | 92.19 | 74.65 |
| RTN W2 (PC) | 2.00 | 0.00 | 0.00 | 33.33 | 0.76 | 8.52 |
| RTN W2 (GS64) | 2.25 | 38.46 | 20.04 | 66.79 | 89.61 | 53.73 |
| GPTQ W2 (GS64) | 2.25 | 59.87 | 38.80 | 82.25 | 92.26 | 68.30 |
| QuIP# (E8P + FT) | 2.00 | 65.91 | 46.51 | 88.73 | 91.89 | 73.26 |
| RTN W2 (GS64) + Q2F ($K = 64$) | 2.25 | 66.64 | 48.77 | 88.84 | 91.66 | 73.98 |
| VGMP + FT | 2.16 | 65.23 | 37.83 | 68.58 | 78.69 | 62.58 |
| VGMP + FT + Q2F ($K = 32$) | 2.16 | 66.66 | 49.41 | 88.37 | 92.34 | 74.20 |
| VGMP + FT + Q2F ($K = 64$) | 2.16 | 66.85 | 48.23 | 88.01 | 92.04 | 73.78 |
| VGMP + FT + Q2F ($K = 128$) | 2.16 | 66.58 | 48.56 | 88.37 | 92.57 | 74.02 |

compared with 27.31 for the FP16 head. These results establish the effect of head refinement under the simulation protocol. Table 4 separately measures storage and throughput with the 4-bit device decoder.

### A.5 Downstream Evaluation with a 4-Bit Decoder

We additionally evaluate Q2F with a 4-bit per-channel decoder following (Lee et al., 2023) on Qwen3-4B and Llama3.2-1B. Input embeddings use 8-bit per-tensor quantization, matching the decoder and embedding precision formats used for on-device profiling in Section 4.3. For each model, the quantized decoder and input embeddings remain fixed across LM-head configurations; the FP16 row uses an FP16 LM-head as the reference. The tasks and sampling settings follow Section 4.1. VGMP promotes $M = 12$K rows for Qwen3-4B and Llama3.2-1B, with one epoch of LM-head fine-tuning following Appendix A.2. Q2F reuses the INT8 embedding table for candidate refinement.

**Quality recovery with a quantized decoder.** Table 11 shows that Q2F remains effective when the decoder is also quantized. At $K = 64$, refinement raises the VGMP + FT average from 70.33 to 74.21 on Qwen3-4B and from 21.25 to 23.53 on Llama3.2-1B, compared with FP16-head references of 74.49 and 23.50, respectively. The refined averages are within 0.28 percentage points of the corresponding references on both models. At $K = 128$, VGMP + FT + Q2F reaches 74.14 and 23.99, respectively. These results complement the on-device profiling by demonstrating near-reference average accuracy with both the decoder and coarse LM-head operating at low precision.

**Effect of the coarse-head configuration.** Refinement also substantially improves group-wise RTN: at $K = 64$, its average increases from 39.32 to 68.39 on Qwen3-4B and from 12.32 to 23.44 on

Table 9: Additional baselines and candidate-size sweep for Llama3.2-3B.

| LM-head configuration | BW | MMLU-Pro | AIME | IFEval | GSM8K | Avg. |
|---|---|---|---|---|---|---|
| FP16 baseline (full model) | 16.00 | 35.46 | 4.39 | 74.34 | 66.57 | 45.19 |
| RTN W4 (PC) | 4.00 | 32.94 | 3.64 | 73.86 | 63.68 | 43.53 |
| RTN W2 (GS64) | 2.25 | 9.75 | 0.64 | 47.72 | 6.14 | 16.06 |
| GPTQ W2 (GS64) | 2.25 | 16.17 | 0.21 | 47.36 | 12.81 | 19.14 |
| QuIP# (E8P + FT) | 2.00 | 32.70 | 3.00 | 70.38 | 63.91 | 42.50 |
| RTN W2 (GS64) + Q2F ($K = 64$) | 2.25 | 35.65 | 4.61 | 74.22 | 66.86 | 45.34 |
| VGMP + FT | 2.19 | 32.48 | 3.53 | 46.76 | 24.94 | 26.93 |
| VGMP + FT + Q2F ($K = 64$) | 2.19 | 35.00 | 6.43 | 73.02 | 56.03 | 42.62 |
| VGMP + FT + Q2F ($K = 128$) | 2.19 | 38.02 | 6.65 | 75.78 | 63.00 | 45.86 |
| VGMP + FT + Q2F ($K = 256$) | 2.19 | 36.84 | 5.89 | 77.22 | 65.96 | 46.48 |

Table 10: Simulated downstream accuracy (%) for Llama3.2-1B

| LM-head configuration | BW | MMLU-Pro | AIME | IFEval | GSM8K | Avg. |
|---|---|---|---|---|---|---|
| FP16 baseline (full model) | 16.00 | 16.56 | 2.89 | 58.63 | 31.15 | 27.31 |
| 8-bit per-channel RTN | 8.00 | 17.23 | 2.25 | 58.15 | 31.31 | 27.24 |
| 2-bit per-channel RTN | 2.00 | 0.08 | 0.00 | 23.50 | 1.44 | 6.26 |
| VGMP | 2.19 | 17.83 | 2.57 | 50.12 | 27.06 | 24.40 |
| VGMP + Q2F ($K = 32$) | 2.19 | 16.81 | 2.25 | 61.75 | 35.55 | 29.09 |
| VGMP + Q2F ($K = 64$) | 2.19 | 16.98 | 1.71 | 61.15 | 36.01 | 28.96 |
| VGMP + Q2F ($K = 128$) | 2.19 | 16.63 | 2.36 | 60.91 | 35.18 | 28.77 |

Llama3.2-1B. On Qwen3-4B, VGMP + FT + Q2F achieves a 5.82-point higher average than RTN (GS64) + Q2F at the same candidate budget. This comparison highlights the value of combining candidate refinement with a carefully constructed coarse head in the 4-bit-decoder setting.

### A.6 Downstream Evaluation with QAT Decoders

We further evaluate Q2F on Qwen3-4B with a 2-bit decoder and a 2/4-bit mixed-precision decoder (average 3-bit), both obtained through quantization-aware training (QAT) following UPQ (Lee et al., 2025). Within each decoder configuration, the decoder weights and INT8 per-tensor input embeddings remain fixed across LM-head variants. The FP16-head reference measures the quality of the corresponding quantized decoder with an FP16 output projection and FP16 input embeddings. VGMP promotes $M = 12$K vocabulary rows to 4 bits and uses 2-bit per-channel quantization for the remaining rows. All VGMP configurations include one epoch of LM-head fine-tuning, and Q2F reuses the INT8 embedding table for candidate refinement. The tasks, sampling settings, calibration data, and head fine-tuning protocol follow Section 4.1 and Appendix A.2.

**Refinement with a 2-bit decoder.** With the 2-bit QAT decoder, replacing the FP16 head with VGMP + FT reduces the average score from 53.39 to 47.59 (Table 12). Adding Q2F raises the average to 52.77 at $K = 64$ and 53.61 at $K = 128$, corresponding to gains of 5.18 and 6.02 percentage points over VGMP + FT alone. At $K = 128$, IFEval increases from 67.03 to 77.70 and GSM8K from 70.28 to 80.36, showing that refinement recovers much of the instruction-following and reasoning quality lost under direct low-bit output projection.

**Refinement with a mixed-precision decoder.** With the 2/4-bit QAT decoder, VGMP + FT achieves an average of 60.27, compared with 62.77 for the FP16-head reference. Q2F raises this average to 62.30 at $K = 64$ and 62.67 at $K = 128$. At $K = 128$, the refined averages are within 0.22 percentage points of their respective FP16-head references across both decoder configurations. These results extend the quality recovery observed with the 4-bit decoder to more aggressively quantized QAT decoders, supporting Q2F as a complementary approach to decoder quantization.

Table 11: Downstream task accuracy (%) with a 4-bit per-channel decoder and INT8 per-tensor input embeddings. Row labels specify the LM-head; gray rows add Q2F refinement. PC denotes per-channel quantization, GS64 denotes group size 64, and FT denotes one epoch of LM-head fine-tuning. Avg. is the unweighted mean of the four task scores.

| LM-head configuration | BW | MMLU-Pro | AIME | IFEval | GSM8K | Avg. |
|---|---|---|---|---|---|---|
| **Qwen3-4B (4-bit PTQ decoder)** | | | | | | |
| FP16 baseline (4-bit decoder) | 16.00 | 66.92 | 49.62 | 89.09 | 92.34 | 74.49 |
| RTN W4 (PC) | 4.00 | 61.88 | 36.76 | 87.53 | 92.04 | 69.55 |
| RTN W2 (PC) | 2.00 | 0.00 | 0.00 | 31.18 | 0.00 | 7.80 |
| RTN W2 (GS64) | 2.25 | 22.06 | 7.71 | 60.55 | 66.94 | 39.32 |
| RTN W2 (GS64) + Q2F ($K = 64$) | 2.25 | 61.98 | 35.16 | 85.73 | 90.67 | 68.39 |
| VGMP + FT | 2.16 | 65.64 | 44.91 | 81.53 | 89.23 | 70.33 |
| VGMP + FT + Q2F ($K = 32$) | 2.16 | 66.61 | 48.98 | 89.45 | 87.41 | 73.11 |
| VGMP + FT + Q2F ($K = 64$) | 2.16 | 66.62 | 49.20 | 88.97 | 92.04 | 74.21 |
| VGMP + FT + Q2F ($K = 128$) | 2.16 | 66.67 | 48.87 | 88.73 | 92.27 | 74.14 |
| **Llama3.2-1B (4-bit PTQ decoder)** | | | | | | |
| FP16 baseline (4-bit decoder) | 16.00 | 15.13 | 0.32 | 55.04 | 23.50 | 23.50 |
| RTN W4 (PC) | 4.00 | 15.06 | 0.32 | 54.56 | 22.74 | 23.17 |
| RTN W2 (PC) | 2.00 | 0.00 | 0.00 | 24.22 | 1.36 | 6.40 |
| RTN W2 (GS64) | 2.25 | 6.83 | 0.00 | 40.77 | 1.67 | 12.32 |
| RTN W2 (GS64) + Q2F ($K = 64$) | 2.25 | 15.03 | 0.32 | 55.51 | 22.90 | 23.44 |
| VGMP + FT | 2.19 | 17.33 | 0.43 | 47.96 | 19.26 | 21.25 |
| VGMP + FT + Q2F ($K = 32$) | 2.19 | 15.87 | 0.21 | 52.88 | 23.21 | 23.04 |
| VGMP + FT + Q2F ($K = 64$) | 2.19 | 15.67 | 0.43 | 54.20 | 23.81 | 23.53 |
| VGMP + FT + Q2F ($K = 128$) | 2.19 | 15.70 | 0.32 | 55.52 | 24.41 | 23.99 |

Table 12: Downstream task accuracy (%) for Qwen3-4B with QAT decoders and INT8 per-tensor input embeddings. Each block fixes the decoder and varies the LM-head. Gray rows add Q2F refinement; FT denotes one epoch of LM-head fine-tuning. VGMP promotes $M = 12$K rows. Avg. is the unweighted mean of the four task scores.

| LM-head configuration | BW | MMLU-Pro | AIME | IFEval | GSM8K | Avg. |
|---|---|---|---|---|---|---|
| **2-bit QAT decoder** | | | | | | |
| FP16 baseline (2-bit decoder) | 16.00 | 49.75 | 5.04 | 78.54 | 80.21 | 53.39 |
| VGMP + FT | 2.16 | 48.65 | 4.39 | 67.03 | 70.28 | 47.59 |
| VGMP + FT + Q2F ($K = 64$) | 2.16 | 50.06 | 4.82 | 76.50 | 79.68 | 52.77 |
| VGMP + FT + Q2F ($K = 128$) | 2.16 | 50.57 | 5.79 | 77.70 | 80.36 | 53.61 |
| **2/4-bit mixed-precision QAT decoder** | | | | | | |
| FP16 baseline (2/4-bit decoder) | 16.00 | 58.60 | 19.19 | 85.49 | 87.79 | 62.77 |
| VGMP + FT | 2.16 | 57.93 | 19.19 | 78.30 | 85.67 | 60.27 |
| VGMP + FT + Q2F ($K = 64$) | 2.16 | 58.75 | 18.58 | 85.13 | 86.73 | 62.30 |
| VGMP + FT + Q2F ($K = 128$) | 2.16 | 58.73 | 18.86 | 85.73 | 87.34 | 62.67 |

### A.7 Compatibility with Different Coarse-Head Quantizers

Table 13 evaluates Q2F with RTN, GPTQ, and fine-tuned 2-bit heads using group size 256 on Qwen3-4B-Instruct-2507. The decoder is FP16 and the input embeddings are INT8, following Section 4.1. The fine-tuned head uses the head-only KL objective of Equation 9. At $K = 8$, refinement increases the average scores from 37.07 to 71.89 for RTN, from 52.94 to 73.88 for GPTQ, and from 69.78 to 74.07 for the fine-tuned head; the FP16-head reference scores 74.34. These results show that candidate refinement benefits multiple coarse-head quantizers, including configurations with coarser groups.

Table 13: Q2F consistently improves downstream accuracy across diverse LM-head quantization approaches on Qwen3-4B with 2-bit GS256 heads, an FP16 decoder, and INT8 input embeddings. Results are shown for RTN, GPTQ, and head-only fine-tuning, demonstrating that Q2F is orthogonal to the underlying quantization method. Avg. denotes the unweighted mean of four downstream task scores; the FP16 reference baseline achieves 74.34.

| Configuration | RTN | GPTQ | Fine-tuning |
|---|---|---|---|
| Without Q2F | 37.07 | 52.94 | 69.78 |
| Q2F ($K = 4$) | 69.12 | 71.49 | 73.31 |
| Q2F ($K = 8$) | 71.89 | 73.88 | 74.07 |

Table 14: Head-only fine-tuning ablation with an FP16 decoder and INT8 input embeddings. All Q2F rows use $K = 64$. VGMP promotes $M = 12$K rows for Qwen3-4B and $M = 18$K for Gemma3-4B. Rows labeled FT use one epoch of head fine-tuning; the other VGMP rows omit this step.

| LM-head configuration | BW | MMLU-Pro | AIME | IFEval | GSM8K | Avg. |
|---|---|---|---|---|---|---|
| **Qwen3-4B** | | | | | | |
| FP16 | 16.00 | 66.31 | 49.62 | 88.85 | 92.57 | 74.34 |
| VGMP | 2.16 | 60.70 | 41.69 | 57.31 | 76.87 | 59.14 |
| VGMP + Q2F | 2.16 | 66.15 | 48.34 | 83.81 | 90.29 | 72.15 |
| VGMP + FT | 2.16 | 65.23 | 37.83 | 68.58 | 78.69 | 62.58 |
| VGMP + FT + Q2F | 2.16 | 66.85 | 48.23 | 88.01 | 92.04 | 73.78 |
| **Gemma3-4B** | | | | | | |
| FP16 | 16.00 | 41.65 | 18.33 | 83.21 | 80.67 | 55.97 |
| VGMP | 2.14 | 32.75 | 10.18 | 70.74 | 70.28 | 45.99 |
| VGMP + Q2F | 2.14 | 40.97 | 19.93 | 78.90 | 78.47 | 54.57 |
| VGMP + FT | 2.14 | 39.20 | 14.68 | 76.37 | 78.46 | 52.18 |
| VGMP + FT + Q2F | 2.14 | 41.73 | 18.44 | 82.85 | 79.68 | 55.68 |

### A.8 Effect of Head-Only Fine-Tuning

Table 14 disentangles the contributions of head-only fine-tuning (FT) and candidate refinement. While FT alone provides modest benefits, its primary value is in closing the remaining accuracy gap after Q2F is applied. With Q2F enabled ($K = 64$), FT improves the average score from 72.15 to 73.78 on Qwen3-4B and from 54.57 to 55.68 on Gemma3-4B, bringing performance to within 0.56 and 0.29 points of the FP16-head baselines, respectively.

The largest gains from FT in the refined setting occur on IFEval, which increases from 83.81 to 88.01 on Qwen3-4B and from 78.90 to 82.85 on Gemma3-4B. By contrast, applying FT without refinement improves the average score by only 3.44 points on Qwen3-4B (59.14 $\rightarrow$ 62.58) and 6.19 points on Gemma3-4B (45.99 $\rightarrow$ 52.18), leaving a substantial gap to the FP16 reference.

These results suggest that Q2F is responsible for recovering most of the accuracy lost to aggressive LM-head quantization, while FT serves as a complementary mechanism that improves candidate selection and closes the final residual gap. Notably, the decoder and embedding table used for refinement remain frozen throughout FT.

### A.9 On-Device Q2F: Integration, Protocol, and Microbenchmark

**Vocabulary permutation and special-token compatibility.** To maintain compatibility with inference backends and application launchers that rely on fixed special-token IDs, we preserve the reserved-ID set $\mathcal{S}$: $\pi(s) = s$ for every $s \in \mathcal{S}$. The remaining rows are sorted by decreasing Fisher score and assigned to unreserved positions in ascending index order. The permutation is applied consistently to the LM-head, input embedding table, and tokenizer mappings.

Precision follows physical row position: the first $M$ rows use 4 bits and the remainder use 2 bits, as defined in Equation 7. Reserved rows inside the prefix use 4 bits and count toward $M$; those outside use 2 bits. The highest-scoring ordinary tokens fill the available positions in the 4-bit block. This construction maintains contiguous precision blocks while preserving the IDs expected by the runtime. The no-permutation baseline retains the original vocabulary order and assigns 4 bits to the first $M$ IDs.

**End-to-end profiling protocol.** We use Llama3.2-1B on a Qualcomm Snapdragon 8 Elite Gen 5 (`SM8850-AC`) through Genie. The decoder uses 4-bit per-channel quantization with FlexRound (Lee et al., 2023) and the input embedding table uses 8-bit per-tensor quantization (RTN). For each LM-head configuration, we perform five separate runs, each generating 2,500 output tokens with the following prompt:

> "Explain the details of Min Heap algorithm to get top-k largest value in $V$ elements, and write down the python code. Repeat the answer 2 times."

We allow the device to cool for two minutes before each run and approximately one hour when switching model configurations. All configurations use the same device and runtime setup. Reference sampling uses temperature 1.0 and top-$p = 1.0$. Q2F samples from the refined candidate logits, and all reported Q2F throughput measurements include this sampling step. The custom sampler executes on a CPU performance core. We report the mean and standard deviation of Genie's token-generation rate across the five runs.

**Exposing the final hidden state.** Genie's standard sampler callback receives full-vocabulary logits but does not expose the final decoder hidden state required for refinement. We therefore extend the NPU graph with an additional output, `last_hidden_states`, and expose the corresponding buffer to the sampler through a lightweight API. The output contains the final normalized decoder representation for the current decoding step and is consumed synchronously by the custom sampler.

**Reusing the embedding artifact.** To avoid introducing additional model artifacts, refinement reuses the deployed input embedding table. The sampler memory-maps the same embedding weights already used for token lookup and accesses them in read-only mode during decoding. As a result, Q2F requires no separate high-precision LM-head weights.

**Candidate selection, refinement, and sampling.** At each decoding step, the NPU executes the decoder and coarse LM head, producing full-vocabulary logits. The custom sampler then (i) selects the top-$K$ candidates from the coarse logits, (ii) gathers the corresponding embedding vectors, (iii) computes refined candidate scores using the exported hidden state, and (iv) samples the next token from the refined candidate set. Refinement is implemented with fixed-size candidate storage and requires no dynamic memory allocation. The custom sampler therefore replaces Genie's standard sampling stage with a candidate-selection, refinement, and sampling pipeline while reusing existing model artifacts.

**CPU microbenchmark.** Figure 4 isolates the CPU operations for candidate selection and refinement. We measure full-vocabulary Top-$K$ selection and candidate-row gather plus GEMV on both prime and performance CPU cores, using 50 warm-up iterations followed by 1,000 timed iterations for each candidate-set size. The stacked bars report mean component latency, and the dashed lines report the p99 latency of the combined selection-and-refinement path. Sampling and NPU model execution are excluded.

Top-$K$ selection dominates at small candidate budgets because it scans all $V$ coarse logits. The gather-and-GEMV cost grows approximately linearly with $K$, processing $Kd$ embedding elements. At $K = 128$, the combined selection-and-refinement path takes approximately 0.135 ms on the prime core and 0.195 ms on the performance core.

These standalone timings are consistent with the small integrated overhead in Table 4. At $K = 128$, mean throughput decreases from 72.03 tokens/s for VGMP + FT to 71.34 tokens/s with Q2F, corresponding to an increase from 13.883 to 14.017 ms/token, or approximately 0.134 ms/token. Adding the standalone performance-core latency of 0.195 ms serially to the baseline predicts 71.03 tokens/s, close to the observed throughput.

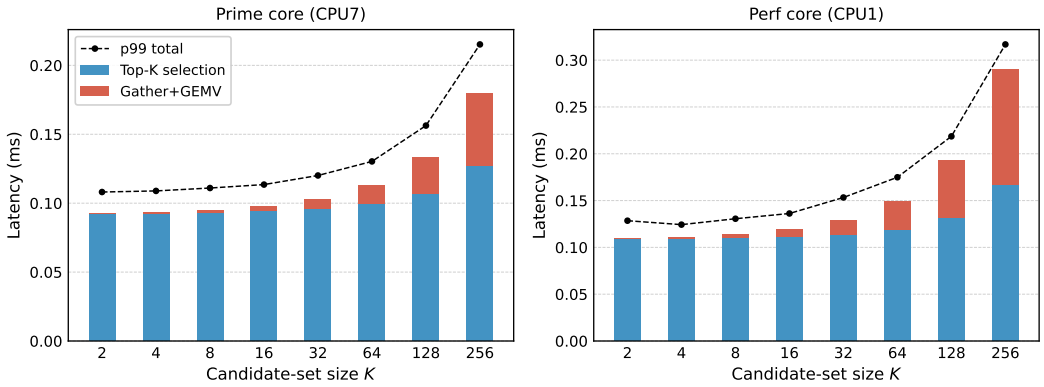



Figure 4: CPU latency breakdown of Q2F candidate selection and refinement for Llama3.2-1B ($V = 128{,}256$, $d = 2{,}048$). Stacked bars show the mean latency of full-vocabulary Top-$K$ selection and candidate-row gather plus GEMV, while dashed lines indicate the p99 latency of the combined path. Measurements are based on 1,000 timed iterations following 50 warm-up iterations on both prime and performance CPU cores. Sampling is excluded from this microbenchmark. The additional CPU work introduced by Q2F remains negligible, contributing only 1.4% of end-to-end decoding latency at $K = 128$ with VGMP head.

The integrated overhead also reflects a change in sampling cost. The baseline samples over the full vocabulary, whereas Q2F samples only among the refined candidates. Under an additive timing model,

$$\Delta t \approx t_{\text{TopK+gather+GEMV}}(K) + t_{\text{sample}}(K) - t_{\text{sample}}(V).$$

Adding the full microbenchmark latency therefore omits potential savings from candidate-restricted sampling. These savings provide a plausible explanation for the measured overhead being smaller than the standalone kernel cost. We interpret the additive projection as a conservative latency estimate; sampling savings were not separately profiled.

Larger candidate budgets can substantially increase CPU refinement cost. Using the approximate gather-and-GEMV scaling $t(K) \approx 0.0005K$ ms observed over the measured range, we examine the cost of a budget comparable to FlashHead's default Llama3.2-1B configuration. That configuration probes 512 clusters of 16 tokens, producing 8,192 candidates per decoding step (Tranheden et al., 2026). Extrapolating our kernel scaling to this budget gives approximately 4.096 ms for gathering and rescoring alone. Adding this cost serially to the same VGMP + FT decoding path, while holding other costs fixed, gives

$$\frac{1000}{1000/72.03 + 4.096} \approx 55.62 \text{ tokens/s},$$

below the measured 59.91-token/s eight-bit-head baseline. Under these assumptions, the refinement cost would outweigh the throughput gain from reducing the size of the head. This extrapolation illustrates the importance of a small CPU candidate budget. It uses our INT8 kernels and omits potential sampling savings; determining FlashHead's end-to-end performance on this platform would additionally require profiling its cluster-based memory access and centroid projection.

### A.10 Selective Low-Rank Compensation

We investigated vocabulary permutation with selective low-rank compensation as an alternative to mixed-precision row promotion, combined with the methods that integrates low-rank and quantization, such as CALDERA (Saha et al., 2024), LoftQ (Li et al., 2024), and SVDQuant (Li et al., 2025). This approach shares ARCHead's use of low-rank correction for a quantized LM-head (Kocabay et al., 2026), while restricting the correction to selected vocabulary rows.

Our variant retains the full 2-bit LM-head and applies an additive rank-64 correction to $M = 10\text{K}$ vocabulary rows. An offline vocabulary permutation places these rows in a contiguous region. Both low-rank factors use 8-bit per-channel quantization; the remaining vocabulary rows receive no

correction. Thus, the additional precision is allocated through a selective low-rank branch rather than by promoting head rows to 4 bits.

We profile this selective low-rank variant to assess its deployment cost, but do not report a matched downstream accuracy comparison with VGMP. On-target profiling reported 555.21 MiB of NPU memory and $69.73 \pm 0.89$ tokens/s, compared with 560.60 MiB and $72.03 \pm 0.69$ tokens/s for VGMP + FT (Table 4). The low-rank variant therefore reduced reported NPU memory by 5.39 MiB while lowering mean throughput by 3.19%.

This trade-off motivated our choice of mixed-precision row promotion for deployment. Although the low-rank representation uses less memory, its correction branch introduces additional projections and accumulation, whose efficiency depends on the available kernels. We did not pursue further optimization of this variant. These measurements characterize our selective low-rank implementation and provide deployment context for the broader design family represented by ARCHead.