TAP-MoE: Task-Aware Profiling for Memory-Efficient Mixture-of-Experts Deployment

Anonymous authors
Paper under double-blind review

Abstract

Mixture-of-Experts (MoE) models scale capacity through sparse activation, yet their large expert pools impose substantial storage and weight-transfer costs that hinder efficient edge deployment. When experts cannot all reside in accelerator memory, deployment must address both compression-induced quality loss and expert-loading overhead. We propose TAP-MoE, a task-aware framework that connects fine-grained expert behavior to task-level deployment decisions. It captures expert activation and sensitivity patterns through shared class-conditional statistics. Reweighting these statistics by the target task's token-class distribution yields a workload-specific profile. This profile guides sensitivity-aware mixed-precision allocation, task-conditioned expert prefetching and residency, and risk-aware output compensation. Precision allocation determines both expert sizes and quantization residuals, linking memory planning and error correction to the same task profile. The framework separates offline calibration from online execution. It preserves the pretrained routing rule and shared experts without model retraining. Our analysis bounds profile-transfer error and derives optimality conditions for a continuous precision-allocation surrogate. We evaluate TAP-MoE on OLMoE-1B-7B, Qwen1.5-MoE-A2.7B, and DeepSeek-MoE-16B-Base across three task domains under matched storage and runtime memory budgets. Compared with BF16, TAP-MoE reduces routed-expert weight storage by 75% while maintaining near-BF16 quality. Against the strongest evaluated SOTA baseline for each metric, TAP-MoE achieves 1.0–5.0% relative reductions in perplexity and 1.25–1.75$\times$ decoding speedups.

1 Introduction

Sparse mixture-of-experts (MoE) models increase parameter capacity by activating only a small subset of experts for each token (Shazeer et al., 2017; Fedus et al., 2022). Although this reduces computation, the full expert pool must still be stored. When accelerator memory cannot hold the pool, selected weights must be retrieved from slower memory. Efficient deployment therefore requires both compact representations and low-overhead access to the experts used by each token.

The choice of expert precision affects both requirements. More bits can protect sensitive computation, but they also increase the weight volume that must be stored, transferred, and cached. Quantization additionally changes expert outputs, creating an error that compensation must address. Prior work has improved expert quantization and memory management, including their co-design (Duanmu et al., 2025; Yi et al., 2023). The question for a target workload is how to use its input characteristics to guide all three decisions: which experts to protect, which weights to prepare, and which output errors to correct.

These decisions require different information about the same experts. Activation frequency describes how often an expert is needed. Loss sensitivity indicates how strongly its perturbation affects the model, while activated features help characterize its output error. Aggregating each quantity over a mixed calibration pool can obscure differences between tasks. Recollecting model-wide statistics for each workload addresses this mismatch but loses statistical reuse across tasks.

Our key insight is that reusable token-class statistics retain this input-dependent structure. Experts exhibit different activation and sensitivity patterns across character-based token classes. The target task changes the proportions of these classes. Shared class-conditional statistics can therefore be

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.

reweighted by the target distribution to obtain a task-conditioned expert profile, rather than treating expert importance as a fixed property of the model.

We propose TAP-MoE, a task-aware framework that uses this profile to coordinate mixed-precision allocation, expert access, and output compensation. As illustrated in Figure 1, precision allocation determines both expert sizes and quantization residuals. Prefetching and residency use these sizes together with task-dependent demand, while compensation uses the residuals together with task-conditioned activated means. All three mechanisms thus act on the same expert representation and serve the same workload. Calibration fixes the precision map, access-policy parameters, and correction vectors before inference, while inference retains the pretrained routing rule and shared-expert computation.

TAP-MoE makes task-aware profiling the common basis for three technical contributions:

- **Task-aware mixed precision.** We synthesize expert sensitivity from shared token-class statistics and the target distribution, then allocate precision under a fixed storage budget. A profile-transfer bound and a continuous allocation analysis explain the statistical basis and the resulting precision priorities.
- **Task-conditioned prefetching and residency.** We combine class-conditional coactivation and target-workload reuse with quantized expert sizes. The resulting access policy coordinates advance loading and repeated reuse. The complete TAP-MoE deployment achieves 1.25-1.75$\times$ decoding speedups over the fastest evaluated reference across three MoE models and three domains.
- **Risk-aware output compensation.** We construct corrections from activation-weighted task means and precision-dependent residuals, then select stable corrections offline. Together with task-aware allocation and access, this completes a deployment framework that reduces PPL by 1.0-5.0% relative to the strongest matched-storage quality reference.

2 Related Work

MoE compression and mixed precision. GPTQ, AWQ, and OmniQuant use curvature, activations, or calibrated transformations to preserve low-precision model quality (Frantar et al., 2023; Lin et al., 2024; Shao et al., 2024). For MoEs, QMoE combines compression with weight decoding, MoEQuant improves expert calibration coverage, and MxMoE considers sensitivity and activation jointly with mixed-precision execution (Frantar & Alistarh, 2024; Chen et al., 2025; Duanmu et al., 2025). GEMQ combines model-wide expert importance estimation with global mixed-precision allocation and router fine-tuning (Deng et al., 2026). MODE uses modality-decomposed expert statistics for multimodal MoEs (Chen et al., 2026). TAP-MoE instead reweights reusable token-class statistics for a target workload and uses the resulting precision map to guide both expert access and output compensation.

Expert offloading and access. MoE-Infinity uses activation traces to guide expert caching, while Fiddler orchestrates host-to-GPU execution (Xue et al., 2025; Kamahori et al., 2025). EdgeMoE combines bit adaptation and preloading, Pre-gated MoE reveals expert choices in advance, and eMoE combines expert prediction with task-aware scheduling (Yi et al., 2023; Hwang et al., 2024; Tairin et al., 2025). ST-MoE (Zhao et al., 2026), EARTH (Liu et al., 2026), and MoE-OPU (Lu et al., 2025) address related access costs through spatio-temporal prediction and hardware-oriented reuse. PagedWeight adapts precision to runtime memory demand (Yang et al., 2026). ADEPT combines domain-aware expert preloading with locality-based prefetching during decoding (Kim et al., 2026), while DyMoE integrates runtime mixed-precision quantization, depth-adaptive scheduling, and look-ahead prefetching (Huang et al., 2026). TAP-MoE plans task-conditioned access around expert representations selected offline, connecting the cost of retrieving each expert to the workload that uses it.

Quantization-error correction. Calibration can reduce output discrepancies as well as weight error. KBVQ-MoE combines input-driven redundancy elimination with bias-corrected output stabilization through an affine correction (Xu et al., 2026). TAP-MoE uses a task-conditioned constant correction derived from activated feature means and output-projection residuals. Offline selection among task, pooled, and zero correction adapts error control to the same task profile that determines precision and access.

3 Preliminaries and Problem Setup

Let l index sparse layers and $e \in \mathcal{E}_l$ their routed experts. At token position t, the pretrained router selects $S_l(t)$ with $|S_l(t)| = K_l$ and routing coefficients $g_{l,e}(t)$. For hidden state $h_l(t)$, the sparse block computes

$$y_l(t) = y_l^{\text{shared}}(t) + \sum_{e \in S_l(t)} g_{l,e}(t) F_{l,e}(h_l(t)). \tag{1}$$

The shared term is zero when absent, and routing normalization follows the checkpoint. We write $j = (l, e)$ for an expert with n_j parameters. Assigning b_j bits gives routed-weight payload $s_j = n_j b_j / 8$. The storage budget constrains $\sum_j n_j b_j$, while the runtime budget also covers resident weights, activations, and working memory.

A task τ induces a distribution over input documents and token positions. A fixed tokenizer-specific map assigns each token to one of 14 character-based classes, including alphabetic form, digits, and punctuation. The task composition is $\pi_\tau(c) = \mathbb{P}_\tau(c(t) = c)$. TAP-MoE uses this distribution to adapt expert precision, access, and output compensation while retaining the pretrained routing rule in Equation (1).

4 TAP-MoE: Task-Aware Profiling and Deployment

Figure 1 summarizes TAP-MoE, which uses a single task-conditioned expert profile as the shared interface between compression and execution. Shared class-conditional statistics and the target distribution form this profile, which guides precision allocation while supplying task-dependent demand and activated feature means to the downstream deployment mechanisms. The resulting precision map determines expert sizes for access planning and quantization residuals for output compensation, coupling the three deployment decisions. We next develop the profile and these mechanisms.

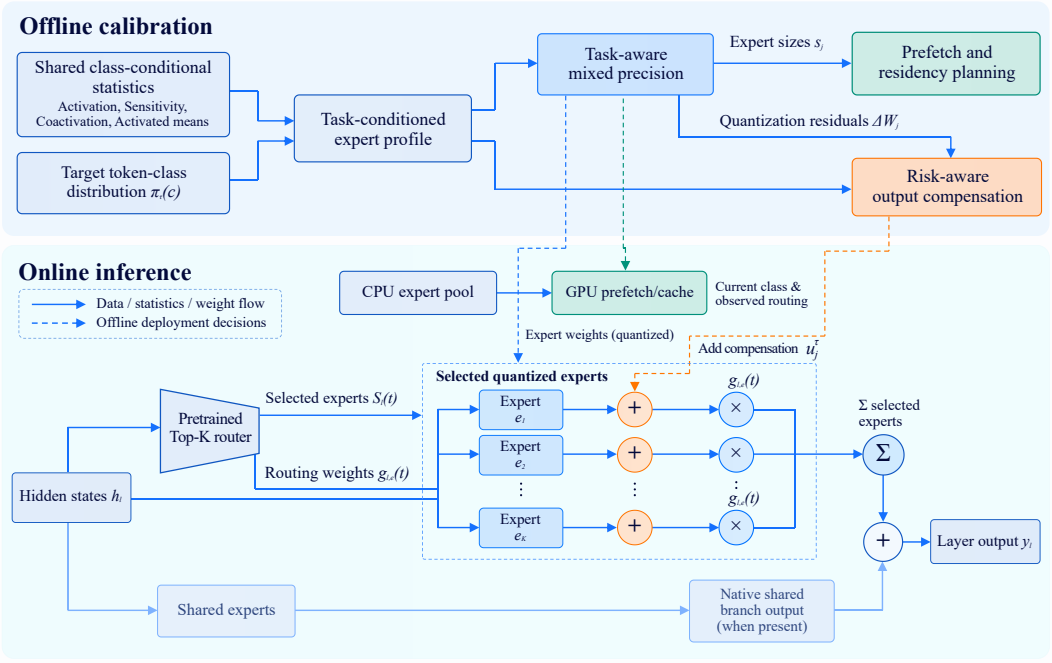


Figure 1: Overview of TAP-MoE. Offline calibration combines shared statistics and task composition to guide precision, expert access, and output compensation. Expert sizes s_j connect precision to access, and residuals ΔW_j connect precision to compensation. Dashed arrows denote offline deployment decisions.

4.1 TASK-AWARE PROFILING

An expert can have different importance for different token classes. A global average combines these responses according to the calibration pool's class proportions, which may differ from those of the target task. TAP-MoE retains class-conditional statistics so that changing the task distribution changes their contribution to each expert's priority.

Let $X_j(t) = \mathbf{1}\{e \in S_l(t)\}$, and let $z_j(t)$ be the input to expert j's output projection. Under the shared calibration distribution, we collect

$$\begin{aligned} p_j^{(c)} &= \mathbb{E}[X_j(t) \mid c(t) = c], & \gamma_j^{(c)} &= \mathbb{E}[\|a_{j,t}\|_F^2 \mid c(t) = c], \\ \mu_j^{(c)} &= \mathbb{E}[z_j(t) \mid X_j(t) = 1, c(t) = c], & a_{j,t} &= X_j(t) J_{W_j} F_j(h_l(t))^\top [g_j(t)\delta_l(t)]. \end{aligned} \tag{2}$$

Here $\delta_l(t) = \partial \mathcal{L}(x)/\partial y_l(t)$ is the adjoint of the sequence negative log-likelihood (NLL). Local gradient energy includes the invocation indicator and routing weight, so it captures both participation and loss sensitivity. We also collect adjacent-layer coactivation counts for each class. Activation estimates demand, sensitivity supplies precision priorities, coactivation describes access dependence, and activated means provide the features needed for compensation. Appendix A.3 derives the local energy without storing a gradient matrix for every invocation.

Combining the target mixture with this shared basis gives a task-conditioned expert profile. Activation and sensitivity are synthesized as

$$\widehat{f}_j^{\tau} = \sum_c \widehat{\pi}_\tau(c) \widehat{p}_j^{(c)}, \qquad \widehat{\gamma}_j^{\tau} = \sum_c \widehat{\pi}_\tau(c) \widehat{\gamma}_j^{(c)}. \tag{3}$$

The first quantity estimates task-dependent expert demand, and the second determines where precision is most valuable. Coactivation retains the class-conditioned information used by prefetching. Activated means additionally condition on expert participation, so their synthesis uses activation-weighted class proportions, as derived in Section 4.4. These statistics provide different inputs to the three mechanisms while sharing the same task composition.

Reweighting accounts for a change in class proportions. Its accuracy also depends on the estimated statistics and on whether expert behavior changes within each class. The following result separates these effects.

Theorem 1 (Task-profile synthesis)**.** Let $p_j^{\tau,(c)}$ be the target inclusion probability and $f_j^{\tau} = \sum_c \pi_\tau(c) p_j^{\tau,(c)}$. For probability mixtures $\widehat{\pi}_\tau, \pi_\tau$ and $\widehat{p}_j^{(c)} \in [0,1]$,

$$\begin{aligned} |\widehat{f}_j^{\tau} - f_j^{\tau}| \le \|\widehat{\pi}_\tau - \pi_\tau\|_1 &+ \sum_c \pi_\tau(c) |\widehat{p}_j^{(c)} - p_j^{(c)}| \\ &+ \sum_c \pi_\tau(c) |p_j^{(c)} - p_j^{\tau,(c)}|. \end{aligned} \tag{4}$$

The three terms represent mixture estimation, conditional estimation, and conditional shift. Exact estimates recover target demand when shared and target conditionals agree on the target support. The result requires no independence between experts or token positions. Appendix A.2 gives the proof and the extension to bounded sensitivity statistics. The synthesized profile now supplies the quantities used to select precision, prepare expert weights, and construct output corrections.

4.2 TASK-AWARE MIXED PRECISION

TAP-MoE allocates precision according to how much each expert's perturbation matters for the target task. We normalize positive synthesized sensitivities by their geometric mean and form allocation weights $w_j = (\widetilde{\gamma}_j^{\tau})^{\rho}$, where $\rho > 0$ controls contrast. For a finite bit palette $\mathcal{B}$ and routed-weight budget B, the allocation objective is

$$\min_{b_j \in \mathcal{B}} \mathcal{J}_\tau(b) = \sum_j w_j 2^{-2b_j} \quad \text{subject to} \quad \sum_j n_j b_j \le B. \tag{5}$$

The exponential factor models the squared scale of a quantization step, and allocation weights determine where reducing that scale is most useful. This surrogate provides precision candidates that are then evaluated using the complete quantized model's calibration loss.

Theorem 2 (Continuous precision allocation)**.** For $w_j, n_j > 0$, finite bounds $b_{\min} < b_{\max}$, and a nonempty feasible set, the continuous relaxation of Equation (5) has a unique optimum. There is a $\lambda^* \geq 0$ such that each interior coordinate satisfies

$$\frac{2 \ln 2}{n_j} w_j 2^{-2b_j^*} = \lambda^*. \tag{6}$$

The left side is at most λ^* at a lower bound and at least λ^* at an upper bound. If all coordinates are interior and storage is active, then

$$b_j^* = \bar{b} + \frac{1}{2}\left[\log_2 \frac{w_j}{n_j} - \sum_i \omega_i \log_2 \frac{w_i}{n_i}\right], \qquad \omega_i = \frac{n_i}{\sum_j n_j}, \qquad \bar{b} = \frac{B}{\sum_j n_j}. \tag{7}$$

The optimum equalizes marginal sensitivity reduction per additional storage bit. Experts with larger sensitivity-to-size ratios receive more precision. Appendix A.4 proves the result, and Appendix A.5 provides the comparison with uniform precision. Under bounded multiplicative errors in the estimated allocation weights, minimizing the estimated allocation surrogate yields bounded suboptimality in the true surrogate objective (Proposition A.4).

For finite palettes, TAP-MoE ranks feasible increments by their surrogate reduction per added storage bit and selects among candidate maps using calibration NLL. The selected b^τ fixes both $s_j = n_j b_j^\tau / 8$ and the output-projection residual $\Delta W_j = W_j^o - Q_{b_j^\tau}(W_j^o)$. Crucially, precision allocation also shapes execution: expert sizes determine the byte costs of access, while output-projection residuals define the errors targeted by compensation.

4.3 Task-Conditioned Prefetching and Residency

The access policy supplies the expert representations selected by mixed precision. Their byte sizes may differ, and their demand depends on the target workload. TAP-MoE combines class-conditioned prefetching with task-conditioned residency so that memory serves both upcoming expert requests and repeated access to the same weights.

Conditional prefetching. Let $N_{l,c}(i, e)$ count coactivation of source expert i in layer $l-1$ and target expert e in layer l. Let $N_{l,c}(i)$ count source invocations. For observed source set s and current token class c, we rank target experts using

$$\widehat{\kappa}_{l,c}(i, e) = \frac{N_{l,c}(i, e) + \zeta}{N_{l,c}(i) + \zeta |\mathcal{E}_l|}, \qquad v_{l,e}(s, c) = \sum_{i \in s} \widehat{\kappa}_{l,c}(i, e), \tag{8}$$

where $\zeta > 0$ is a smoothing constant. Missing candidates are considered in score order and admitted according to their actual byte costs. Prediction uses the current token class and observed source-layer routing within the native forward computation.

Conditional information improves the best achievable coverage under a fixed budget. If $q_e(s, c)$ is the true conditional inclusion probability, the conditional expected hit count for set P is $\sum_{e \in P} q_e(s, c)$. With equal expert sizes, the largest probabilities maximize this quantity under a count budget. For the common byte-feasible family $\mathcal{F}_C = \{P : \sum_{e \in P} s_e \leq C\}$,

$$\mathbb{E}_Z\left[\max_{P \in \mathcal{F}_C} \sum_{e \in P} q_e(Z)\right] \geq \max_{P \in \mathcal{F}_C} \mathbb{E}_Z\left[\sum_{e \in P} q_e(Z)\right], \qquad Z = (s, c). \tag{9}$$

Appendix A.6 proves this relation and derives the residual-demand knapsack used for base preloading. The finite planner combines that base set with speculative coverage using empirical scores, task calibration traces, and the sizes of the selected quantized experts.

Task-conditioned residency. Residency uses calibrated task reuse, retrieval savings, and representation size to prioritize available weights. A packed hit avoids transfer, while a decoded hit also avoids unpacking. Resident weights $C(t)$ and transient prefetch weights $P(t)$ share the constraint

$$M_0(t) + C(t) + P(t) \leq M_{\text{budget}}, \tag{10}$$

where $M_0(t)$ includes fixed parameters, activations, key-value (KV) state, and workspaces. The selected policy can use residency without speculative loading when that gives lower latency. On a miss, it loads the same expert selected by the router. For fixed quantized weights and corrections, the access policy changes when expert representations become available rather than the computation performed with them; Proposition A.8 formalizes this availability invariance.

4.4 Risk-Aware Output Compensation

The same precision map also determines the output error to be corrected. For input z_j to the expert's final projection, the modeled residual is $\Delta W_j z_j$. Its mean depends on the inputs that activate the expert under the target task. TAP-MoE therefore uses the profile's activated feature means to complement sensitivity-based precision allocation.

Because $\mu_j^{(c)}$ conditions on expert participation, its synthesis weights both class prevalence and activation:

$$\widehat{\mu}_j^{\tau} = \frac{\sum_c \widehat{\pi}_{\tau}(c)\widehat{p}_j^{(c)}\widehat{\mu}_j^{(c)}}{\sum_c \widehat{\pi}_{\tau}(c)\widehat{p}_j^{(c)}}. \tag{11}$$

With matching conditionals and positive support, the population formula recovers the target activated mean. For fixed ΔW_j and finite second moment, the unique constant minimizing $\mathbb{E}[\|\Delta W_j z_j - u\|_2^2 \mid j \text{ active}]$ is $u^* = \Delta W_j \mu_j$. Appendix A.7 derives both results. They identify the task-dependent mean residual as the correction target.

TAP-MoE separates a pooled offset from the additional task-dependent component:

$$u_j^{\tau}(\eta) = \Delta W_j \left[\widehat{\mu}_j^{g} + \eta\beta_j^{\tau}(\widehat{\mu}_j^{\tau} - \widehat{\mu}_j^{g})\right]. \tag{12}$$

The factor $\beta_j^{\tau} \in [0, 1]$ shrinks residuals with weak calibration agreement or limited effective support. Calibration selects the residual strength and compares task, pooled, and zero correction through paired model-loss differences. The chosen vectors are fixed before inference, so selection requires no online optimization. Appendix B.4 gives the rule and its alternatives.

Once precision, access, and compensation have been selected, the deployed block computes

$$\widehat{y}_l(t) = \widehat{y}_l^{\text{shared}}(t) + \sum_{e \in \widehat{S}_l(t)} \widehat{g}_{l,e}(t) \left[Q_{b_{l,e}^{\tau}}(F_{l,e})(\widehat{h}_l(t)) + u_{l,e}^{\tau}\right]. \tag{13}$$

Here $Q_b(F)$ quantizes the routed projections at b bits. The pretrained router and unchanged shared branch operate on the current hidden states, yielding $\widehat{S}_l$, $\widehat{g}_l$, and $\widehat{y}_l^{\text{shared}}$. The access policy supplies the selected weights, and compensation is applied before routing-weight multiplication. Equation (13) summarizes the complete TAP-MoE computation. Algorithm B.1 provides its preparation and inference flow.

5 Experiments

We first evaluate the quality and efficiency of the complete TAP-MoE deployment, then examine the role of task-aware profiling and each deployment mechanism.

5.1 Evaluation Setup

We evaluate OLMoE-1B-7B, Qwen1.5-MoE-A2.7B, and DeepSeek-MoE-16B-Base across general text, code, and mathematics (Muennighoff et al., 2025; Qwen Team, 2024; Dai et al., 2024). At the main operating point, routed-expert weights average four bits, corresponding to a 75% reduction in their nominal routed-weight payload relative to BF16; routers, shared experts, and fixed parameters remain in BF16. Experiments run on an NVIDIA RTX A6000 in a host-to-GPU offloading setting, with matched expert-storage and model-specific runtime budgets.

Document-prefix PPL is evaluated on WikiText-2, MBPP reference code, and MATH reference solutions (Merity et al., 2017; Austin et al., 2021; Hendrycks et al., 2021b). Contiguous WikiText-2 provides a complementary language-modeling evaluation. HumanEval, GSM8K, and MMLU assess

downstream performance (Chen et al., 2021; Cobbe et al., 2021; Hendrycks et al., 2021a). Online measurements report time per output token (TPOT), time to first token (TTFT), and end-to-end request time (E2E). Quality and speed each use the strongest evaluated reference for that metric. Appendix C gives the complete comparisons and metric definitions.

5.2 Quality and Deployment Efficiency

Across all nine model-domain settings, the same TAP-MoE deployment improves quality over the strongest matched-storage quantized reference and decoding efficiency over the fastest evaluated deployment reference (Tables 1 and 2). TAP-MoE achieves the lowest quantized PPL in all nine settings, with relative reductions of 1.0-5.0%. BF16 has lower PPL as the uncompressed reference, with TAP-MoE remaining close to it. OLMoE code improves from 4.1474 to 4.0229 against EdgeMoE, while Qwen mathematics improves from 2.0633 to 1.9601 against GPTQ4. Task-aware precision and compensation preserve quality at the same routed-weight budget.

Table 1: Document-prefix PPL (↓) at four average routed-expert bits. BF16 is the uncompressed reference. Bold marks the best quantized result.

Model	Domain	BF16	Uniform4	GPTQ4	EdgeMoE	TAP
OLMoE	general	7.8345	8.3536	8.2918	9.6080	**7.8772**
	code	3.9956	4.7200	4.3584	4.1474	**4.0229**
	math	3.2054	3.4467	3.3911	3.4027	**3.2215**
Qwen1.5	general	6.3344	7.0631	6.6535	6.6546	**6.4207**
	code	4.6321	5.8831	4.7127	4.7912	**4.6562**
	math	1.9523	2.2237	2.0633	2.0819	**1.9601**
DeepSeek	general	6.5022	6.7957	6.6932	6.6981	**6.5259**
	code	2.9498	3.2367	3.0271	3.0263	**2.9658**
	math	2.8233	2.9405	2.8872	2.8625	**2.8339**

The same deployments also reduce decoding latency. Table 2 reports speedups of 1.25-1.75× over the fastest evaluated deployment reference. The largest gain is on OLMoE code, where TPOT falls from 93.09 to 53.19 ms/token. Prefetching prepares likely requests, while residency avoids repeated retrieval and decoding of the selected weights. The improved quality therefore accompanies faster execution without reducing the native selected expert count.

Together, Tables 1 and 2 establish consistent quality and efficiency gains across all nine settings; the paired 95% confidence intervals remain above zero for relative PPL reduction and above one for decoding speedup (Table C.2). Contiguous-text results also favor TAP-MoE over Uniform4 and GPTQ4 for all three models, while BF16 remains the lower-PPL reference in that evaluation (Table C.7).

Table 2: Online deployment on NVIDIA RTX A6000. TPOT is in ms/token, TTFT in ms, and E2E in seconds. eMoE is the fastest evaluated reference. TAP metrics describe the same deployment.

Model	Domain	eMoE TPOT	TAP TPOT	Speedup ×	TTFT	E2E
OLMoE	general	90.71	**54.98**	1.65	390	2.149
	code	93.09	**53.19**	1.75	370	2.072
	math	89.73	**52.78**	1.70	380	2.069
Qwen1.5	general	108.51	**70.01**	1.55	920	3.160
	code	102.70	**70.82**	1.45	1020	3.286
	math	110.20	**68.87**	1.60	790	2.994
DeepSeek	general	152.97	**109.26**	1.40	1180	4.676
	code	137.60	**110.08**	1.25	1160	4.683
	math	147.79	**105.56**	1.40	1150	4.528

5.3 Effectiveness of Task-Aware Profiling

Token classes carry expert information beyond global activation frequencies. Class-conditioned prediction reduces held-out routing cross-entropy by 0.402-0.781 bits per selected expert across the nine settings (Table D.1). This information provides a basis for adapting expert priorities through the target task's class mixture.

Figure 2 illustrates how task composition enters the allocation. Panel (a) gives class proportions, panel (b) shows shared class-conditional sensitivity, and panel (c) aligns task sensitivity with assigned bits. For example, e37 receives six bits for Code and Math but four bits for General, whereas e45 receives six bits for Code and four bits for the other workloads. These assignments illustrate both shared and workload-specific precision priorities within the same expert pool. Task conditioning therefore redistributes a fixed precision budget according to which experts receive greater protection for each workload.

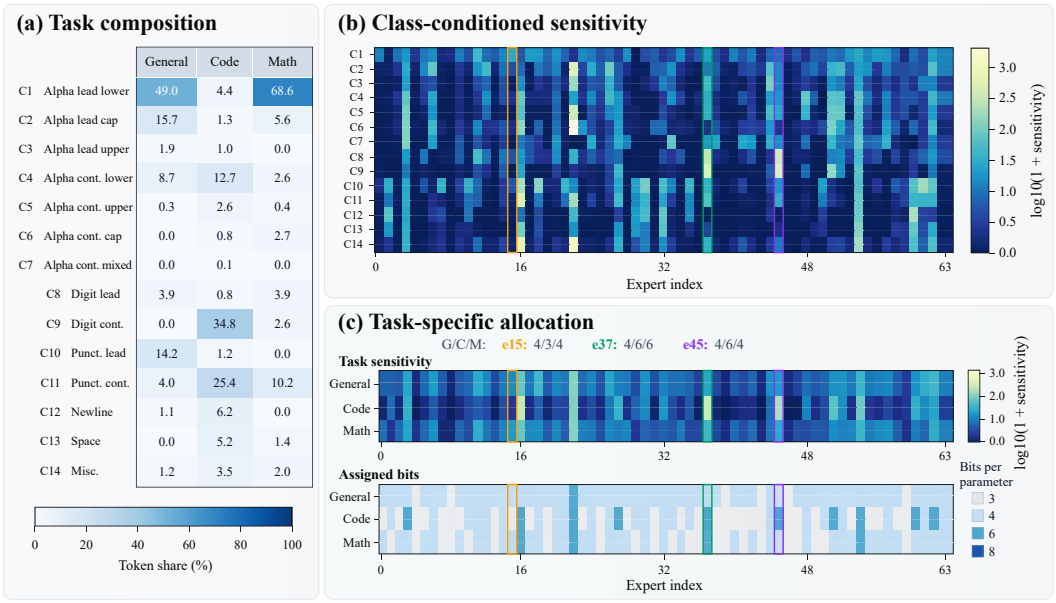


Figure 2: Illustrative expert-level allocation. Panels show (a) token-class proportions, (b) class-conditional sensitivity, and (c) task sensitivity and bit assignments. Colored outlines identify the same experts in panels (b) and (c). G/C/M denotes General/Code/Math.

TAP-MoE lowers PPL by 0.65-3.28% relative to Pooled and by 1.20-5.49% relative to Mismatch, with Frequency also worse throughout (Table D.2). These controls show that the benefit depends on matching expert sensitivity to the workload, not merely on using nonuniform bit widths. TAP-MoE also achieves lower quantized-model PPL than Direct in all nine settings (Table D.2), providing final-model evidence for the shared-statistics synthesis.

5.4 Ablation Studies and Analysis

Table 3 examines the three mechanisms within the complete framework. Pooled changes the statistic used for precision allocation, No corr. removes output compensation, and Pooled load replaces task-conditioned access. The quality controls test how the chosen representation is protected, while the access control tests how efficiently it is supplied.

Task-conditioned output compensation lowers PPL by 0.10-0.92% relative to zero correction on the same quantized weights. Its online cost is 0.15, 0.25, and 0.35 ms/token for OLMoE, Qwen, and DeepSeek, respectively (Table D.3). Thus the correction improves the selected representation with a small additional online cost.

At equal pool capacity, task-conditioned caching alone reduces TPOT by 0.75-9.36% relative to pooled caching (Table D.4). Prefetching improves the task-cache result in six settings. The other

Table 3: Ablation study of TAP-MoE. Pooled uses task-independent allocation sensitivity. No corr. removes output compensation. Pooled load replaces task-conditioned access while keeping weights and corrections fixed.

Model	Domain	Pooled PPL	No corr. PPL	TAP PPL	Pooled load ms/token	TAP ms/token
OLMoE	general	8.1442	7.9258	**7.8772**	57.54	**54.98**
	code	4.1034	4.0475	**4.0229**	61.87	**53.19**
	math	3.3307	3.2514	**3.2215**	60.73	**52.78**
Qwen1.5	general	6.5711	6.4482	**6.4207**	70.80	**70.01**
	code	4.6929	4.6618	**4.6562**	72.05	**70.82**
	math	2.0265	1.9752	**1.9601**	70.81	**68.87**
DeepSeek	general	6.6342	6.5424	**6.5259**	112.59	**109.26**
	code	3.0050	2.9748	**2.9658**	111.94	**110.08**
	math	2.8525	2.8367	**2.8339**	109.79	**105.56**

settings retain caching alone. These controls show that task information benefits both repeated reuse and advance loading, while the selected policy adapts to where each is effective.

5.5 Downstream Performance and Additional Analysis

TAP-MoE achieves higher downstream scores than GPTQ4 across all nine model-benchmark combinations (Table C.8). The gains are 2.44-3.05 percentage points on HumanEval, 1.12-2.22 points on GSM8K, and 0.63-1.28 points on MMLU. Relative to BF16, TAP-MoE remains within 0.60-1.22 percentage points on HumanEval, 0.08-0.41 points on GSM8K, and 0.15-0.57 points on MMLU. All nine paired 95% confidence intervals for gains over GPTQ4 lie above zero (Table C.9). These results complement PPL with generated-program correctness and answer-level prediction.

Across the three workloads, peak allocated GPU memory is 1.833-1.879 GiB for OLMoE, 4.523-4.530 GiB for Qwen1.5-MoE, and 3.582-3.597 GiB for DeepSeek-MoE (Table C.6). These measurements confirm that the TAP-MoE configurations reported in Tables 1 and 2 achieve their quality and latency gains while satisfying the corresponding runtime-memory budgets.

Table D.5 evaluates nearby storage budgets. Increasing average precision lowers PPL and modestly increases decoding time. This behavior reflects the two effects of precision within TAP-MoE: it changes both the quantization error and the size of the representation supplied by the access policy. Appendix D gives the complete mechanism and budget analyses.

6 Conclusion

We presented TAP-MoE, a task-aware framework for memory-efficient MoE deployment that reweights reusable class-conditional expert statistics to coordinate mixed-precision allocation, expert access, and output compensation. The resulting precision map couples compression and execution through expert sizes and quantization residuals, enabling a single workload-conditioned profile to coordinate representation fidelity, access efficiency, and error compensation. Our analysis characterizes profile transfer and budgeted precision allocation, while experiments across three MoE architectures and three workloads show that TAP-MoE reduces routed-expert weight storage by 75% relative to BF16 while maintaining near-BF16 quality, lowers PPL by 1.0-5.0% against the evaluated SOTA baseline, and achieves 1.25-1.75$\times$ decoding speedups over the fastest evaluated deployment reference. Task-aware profiling thus provides a unified basis for coordinating MoE compression and execution under memory constraints.

ETHICS STATEMENT

This work follows the ICLR Code of Ethics and does not involve human-subject studies or animal experiments. All models and evaluation datasets used in this study are publicly available research resources, including WikiText-2, MBPP, MATH, HumanEval, GSM8K, MMLU, and others. Our experiments focus on model compression and deployment efficiency and do not collect or process personally identifiable information. We have taken care to use these resources only for their intended research purposes and to report the experimental results and comparisons transparently.

REPRODUCIBILITY STATEMENT

We have made every effort to ensure that the results presented in this paper are reproducible. Sections 3 and 4 provide the methodological definitions and main deployment configurations required to reproduce TAP-MoE, while Section 5 specifies the experimental settings and evaluation protocols. The paper specifies the evaluated MoE models, task domains, storage and runtime constraints, profiling procedure, precision-allocation objective, expert-access strategy, output-compensation mechanism, and the corresponding evaluation metrics. Appendix A provides additional proofs, Appendix B presents the algorithms and implementation-related procedures, and Appendices C and D provide complete comparison tables, ablation results, uncertainty intervals, and other supporting experimental details to support independent reproduction and verification of the reported results. We believe these measures will enable other researchers to reproduce our work and further advance the field.

AI USE STATEMENT

Large Language Models (LLMs) were used to aid in the writing and polishing of the manuscript. Specifically, an LLM was used to assist with language refinement, sentence rephrasing, grammar checking, readability improvement, and overall presentation of the text. The LLM was not used to generate the research ideas, methodology, experimental design, or scientific conclusions. The authors take full responsibility for the content of the manuscript and for all text produced or revised with LLM assistance.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Yuanteng Chen, Nanxin Zeng, Peisong Wang, Zhilei Liu, Yuantian Shao, Shiqiang Lang, Tao Liu, Chuangyi Li, Qinghao Hu, Gang Li, Jing Liu, and Jian Cheng. MODE: Modality-decomposed expert-level mixed-precision quantization for MoE multimodal LLMs. In *The 2026 Conference on Empirical Methods in Natural Language Processing, Main Conference*, 2026. URL https://arxiv.org/abs/2606.17118v2. arXiv:2606.17118v2.

Zhixuan Chen, Xing Hu, Dawei Yang, Zukang Xu, Xu Chen, Zhihang Yuan, Sifan Zhou, and Jiangyong Yu. MoEQuant: Enhancing quantization for mixture-of-experts large language models via expert-balanced sampling and affinity guidance. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 8245–8260, 2025. URL https://proceedings.mlr.press/v267/chen25aa.html.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, et al. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024. URL https://arxiv.org/abs/2401.06066.

Jianing Deng, Song Wang, Dongwei Wang, Zijie Liu, Tianlong Chen, Huanrui Yang, and Jingtong Hu. GEMQ: Global expert-level mixed-precision quantization for MoE LLMs. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2605.23078.

Haojie Duanmu, Xiuhong Li, Zhihang Yuan, Size Zheng, Jiangfei Duan, Xingcheng Zhang, and Dahua Lin. MxMoE: Mixed-precision quantization for MoE with accuracy and performance co-design. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 14793–14806, 2025. URL https://proceedings.mlr.press/v267/duanmu25a.html.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL https://www.jmlr.org/papers/v23/21-0998.html.

Elias Frantar and Dan Alistarh. QMoE: Sub-1-bit compression of trillion parameter models. In *Proceedings of Machine Learning and Systems*, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/c74b624843218d9b6713fcf299d6d5e4-Abstract-Conference.html.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a. URL https://arxiv.org/abs/2009.03300.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021b. URL https://arxiv.org/abs/2103.03874.

Yuegui Huang, Zhiyuan Fang, Weiqi Luo, Ruoyu Wu, Wuhui Chen, and Zibin Zheng. DyMoE: Dynamic expert orchestration with mixed-precision quantization for efficient MoE inference on edge. *arXiv preprint arXiv:2603.19172*, 2026. URL https://arxiv.org/abs/2603.19172.

Ranggi Hwang, Jianyu Wei, Shijie Cao, Changho Hwang, Xiaohu Tang, Ting Cao, and Mao Yang. Pre-gated MoE: An algorithm-system co-design for fast and scalable mixture-of-expert inference. In *ACM/IEEE International Symposium on Computer Architecture*, 2024. doi: 10.1109/ISCA59077.2024.00078. URL https://arxiv.org/abs/2308.12066.

Keisuke Kamahori, Tian Tang, Yile Gu, Kan Zhu, and Baris Kasikci. Fiddler: CPU-GPU orchestration for fast inference of mixture-of-experts models. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=N5fVv6PZGz.

Hangyeol Kim, Honguk Woo, and Younghwan Kim. Two-stage expert offloading for domain-aware MoE inference. *IEEE Access*, 14:33610–33624, 2026. doi: 10.1109/ACCESS.2026.3665697. URL https://doi.org/10.1109/ACCESS.2026.3665697.

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 *Proceedings of Machine Learning and Systems*, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html.

Fangxin Liu, Ning Yang, Jingkui Yang, Zongwu Wang, Chenyang Guan, Yu Feng, Li Jiang, and Haibing Guan. EARTH: An efficient MoE accelerator with entropy-aware speculative prefetch and result reuse. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2026. doi: 10.1145/3779212.3790155. URL https://doi.org/10.1145/3779212.3790155.

Shaoqiang Lu, Yangbo Wei, Junhong Qian, Chen Wu, Xiao Shi, and Lei He. MoE-OPU: An FPGA overlay processor leveraging expert parallelism for MoE-based large language models. In *IEEE/ACM International Conference on Computer-Aided Design*, 2025. doi: 10.1109/ICCAD66269.2025.11240807. URL https://ieeexplore.ieee.org/document/11240807.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1609.07843.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. OLMoE: Open mixture-of-experts language models. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xXTkbTBmqq.

Qwen Team. Qwen1.5-MoE: Matching 7B model performance with 1/3 activated parameters. Technical blog, 2024. URL https://qwenlm.github.io/blog/qwen-moe/.

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 *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8Wuvhh0LYW.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1701.06538.

Suraiya Tairin, Shohaib Mahmud, Haiying Shen, and Anand Iyer. eMoE: Task-aware memory efficient mixture-of-experts-based (MoE) model inference. *arXiv preprint arXiv:2503.06823*, 2025. URL https://arxiv.org/abs/2503.06823.

Zukang Xu, Zhixiong Zhao, Xing Hu, Zhixuan Chen, and Dawei Yang. KBVQ-MoE: KLT-guided SVD with bias-corrected vector quantization for MoE large language models. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2602.11184.

Leyang Xue, Yao Fu, Zhan Lu, Luo Mai, and Mahesh Marina. MoE-Infinity: Efficient MoE inference on personal machines with sparsity-aware expert cache. *arXiv preprint arXiv:2401.14361v3*, 2025. URL https://arxiv.org/abs/2401.14361v3.

Yuchen Yang, Yifan Zhao, Anisha Dasgupta, and Sasa Misailovic. PagedWeight: Efficient MoE LLM serving with dynamic quality-aware weight quantization. *arXiv preprint arXiv:2607.16184*, 2026. URL https://arxiv.org/abs/2607.16184.

Rongjie Yi, Liwei Guo, Shiyun Wei, Ao Zhou, Shangguang Wang, and Mengwei Xu. EdgeMoE: Empowering sparse large language models on mobile devices. *arXiv preprint arXiv:2308.14352*, 2023. URL https://arxiv.org/abs/2308.14352.

Yingnan Zhao, Razvan Bunescu, Ahmed Louri, Avinash Karanth, and Ke Wang. A spatio-temporal expert prefetching framework for efficient MoE-based LLM inference. *arXiv preprint arXiv:2606.15453*, 2026. URL https://arxiv.org/abs/2606.15453.

A Theoretical Analysis and Proofs

We analyze the statistical and optimization results used by TAP-MoE in the order of its deployment decisions. Task composition first determines expert statistics. These statistics guide precision allocation, whose byte costs and residuals enter expert access and output compensation.

A.1 Notation and Statistical Setting

Table A.1 summarizes the notation. Let $\mathcal{D}_g$ and $\mathcal{D}_\tau$ denote the shared and target distributions. Expectations weight documents by their counted token positions and then sample a position uniformly. The model is fixed, and randomness enters through inputs and positions.

Table A.1: Notation used in the method and theoretical analysis. Hats denote empirical estimates.

Symbol	Meaning
$l, e, j = (l, e)$	Sparse layer, routed expert, and flattened expert index.
$\mathcal{E}_l, K_l, S_l(t)$	Routed expert set, native selected count, and selected set.
$g_{l,e}(t), h_l(t)$	Native routing coefficient and sparse-block input.
$X_j(t), c(t), \mathcal{C}$	Participation indicator, token class, and class set.
π_τ, π_g	Target and shared token-class distributions.
$p_j^{(c)}, \gamma_j^{(c)}$	Class-conditional inclusion and local gradient energy.
$\mu_j^{(c)}, \mu_j^\tau$	Class-conditional and synthesized activated means.
$N_{l,c}(i,e), N_{l,c}(i)$	Adjacent-layer coactivation and source-expert counts.
n_j, b_j, s_j	Parameter count, assigned bits, and packed payload bytes.
B, M_{budget}	Routed-weight storage and runtime memory budgets.
$w_j, R_j, \mathcal{J}_\tau$	Allocation weight, precision penalty, and objective.
$\Delta W_j, u_j^\tau$	Output-projection residual and correction vector.
$R_l, \beta_l^{\text{pf}}, C_l$	Base preload set, speculative allowance, and layer window.
β_j^τ, η	Residual-stability factor and task-correction strength.

The inclusion probability records whether an expert participates, rather than the numerical coefficient multiplying its output. Since each counted position selects K_l experts,

$$\sum_{e \in \mathcal{E}_l} p_{l,e}^{(c)} = \mathbb{E}_g \left[\sum_{e \in \mathcal{E}_l} X_{l,e}(t) \mid c(t) = c \right] = K_l. \tag{A.1}$$

For supported class counts N_c and expert counts $n_{l,e,c}$, the empirical estimate is $\widehat{p}_{l,e}^{(c)} = n_{l,e,c}/N_c$. Counting the same positions in adjacent layers gives

$$\sum_e n_{l,e,c} = N_c K_l, \qquad \sum_{i,e} N_{l,c}(i,e) = N_c K_{l-1} K_l. \tag{A.2}$$

These identities follow by exchanging finite sums and do not require independent expert selections.

Sensitivity averages over all positions of a token class, including zero contributions from inactive experts. An activated feature mean instead averages over positions on which the expert participates. The first convention makes sensitivity linear in the class mixture. The second requires activation-weighted class proportions for compensation.

A.2 Task-Profile Synthesis and Transfer

Theorem 1 separates changes in class proportions from changes in expert behavior within each class.

Proof of Theorem 1. Fix an expert and suppress its index. Total probability gives

$$f^\tau = \mathbb{P}_\tau(X = 1) = \sum_c \pi_\tau(c) p^{\tau,(c)}. \tag{A.3}$$

Insert the shared conditionals with the true target weights:

$$\begin{aligned}\widehat{f} - f^{\tau} = &\sum_{c} (\widehat{\pi}_{\tau}(c) - \pi_{\tau}(c))\widehat{p}^{(c)} \\ &+ \sum_{c} \pi_{\tau}(c)(\widehat{p}^{(c)} - p^{(c)}) + \sum_{c} \pi_{\tau}(c)(p^{(c)} - p^{\tau,(c)}).\end{aligned} \tag{A.4}$$

Since $\widehat{p}^{(c)} \in [0,1]$, the absolute value of the first sum is at most $\|\widehat{\pi}_{\tau} - \pi_{\tau}\|_1$. The triangle inequality bounds the other terms by their weighted absolute differences, proving Equation (4). Exact estimates and matching conditionals on the target support make all three terms vanish.

The first term measures mixture estimation, the second shared-statistic estimation, and the third within-class distribution shift. Reweighting represents changes in task composition directly. The same decomposition applies to other bounded conditional statistics.

Corollary A.1 (Transfer of bounded conditional statistics)**.** Let $a_g(c)$ and $a_{\tau}(c)$ be shared and target conditional expectations of a scalar statistic, and suppose $|\widehat{a}_g(c)| \le M < \infty$. For $a_{\tau} = \sum_c \pi_{\tau}(c) a_{\tau}(c)$ and $\widehat{a} = \sum_c \widehat{\pi}_{\tau}(c)\widehat{a}_g(c)$,

$$\begin{aligned}|\widehat{a} - a_{\tau}| \le M\|\widehat{\pi}_{\tau} - \pi_{\tau}\|_1 &+ \sum_{c} \pi_{\tau}(c)|\widehat{a}_g(c) - a_g(c)| \\ &+ \sum_{c} \pi_{\tau}(c)|a_g(c) - a_{\tau}(c)|.\end{aligned} \tag{A.5}$$

Proof. Replace p by a in Equation (A.4). The bound on $\widehat{a}_g$ multiplies the mixture term by M. The remaining terms follow from the triangle inequality.

The corollary covers bounded local-sensitivity estimates. Activated means additionally condition on participation and are analyzed in Appendix A.7.

Proposition A.2 (Stability of mixture reweighting)**.** Fix shared sensitivities with $0 \le \gamma_j^{(c)} \le M_j$. For two class mixtures π, π',

$$|\gamma_j(\pi') - \gamma_j(\pi)| \le M_j\|\pi' - \pi\|_1. \tag{A.6}$$

Proof. Subtract the weighted sums and bound $|\sum_c(\pi'_c - \pi_c)\gamma_j^{(c)}|$ by $\sum_c |\pi'_c - \pi_c| M_j$.

The synthesized statistic is continuous in the mixture. A discrete bit assignment can nevertheless change when an expert's priority crosses an allocation boundary.

A.3 Local Sensitivity and the Distortion Surrogate

Equation (2) measures a local invocation within the complete sequence computation. We first derive its gradient energy, then explain the precision-dependent factor used in the allocation objective.

Local gradient contributions. Write $\mathcal{L}(x) = \sum_{r \in T(x)} \ell_r(x)$. For fixed forward routing choices, the chain rule gives

$$\frac{\partial \mathcal{L}}{\partial W_j} = \sum_{t} a_{j,t}, \qquad a_{j,t} = X_j(t) J_{W_j} F_j(h_l(t))^{\top} [g_j(t)\delta_l(t)]. \tag{A.7}$$

The adjoint includes later targets reached through subsequent layers and attention. One sequence backward pass supplies all local contributions, with zero contribution from inactive invocations.

For a selected SwiGLU invocation, suppress indices and write

$$a = W_g h, \quad v = W_u h, \quad z = \sigma(a) \odot v, \quad F(h) = W_d z, \tag{A.8}$$

where σ is SiLU. With $r = g\delta$ and $d = W_d^{\top} r$, the local gradients are

$$G_d = r z^{\top}, \quad G_u = (d \odot \sigma(a)) h^{\top}, \quad G_g = (d \odot v \odot \sigma'(a)) h^{\top}. \tag{A.9}$$

The identity $\|uv^\top\|_F^2 = \|u\|_2^2\|v\|_2^2$ gives

$$\begin{aligned}\|a_{j,t}\|_F^2 = \|r\|_2^2\|z\|_2^2 + \|d \odot \sigma(a)\|_2^2\|h\|_2^2 \\ + \|d \odot v \odot \sigma'(a)\|_2^2\|h\|_2^2.\end{aligned} \tag{A.10}$$

The projection matrices are disjoint parameter blocks, so their squared norms add. Equation (A.10) recovers the local energy from vectors without storing a gradient matrix for every invocation.

Local gradient energy differs from the squared full-sequence gradient:

$$\left\|\sum_t a_{j,t}\right\|_F^2 = \sum_t \|a_{j,t}\|_F^2 + 2\sum_{t<u}\langle a_{j,t}, a_{j,u}\rangle. \tag{A.11}$$

The profile retains class-conditional diagonal energy. This makes task reweighting possible without storing all cross-position interactions. Complete-model calibration evaluates the finite precision candidates produced from this statistic.

Precision-dependent distortion. Let $\Delta W_j^{\mathrm{all}} = W_j - Q_{b_j}(W_j)$ contain all quantized projections of expert j. At a route-stable smooth working point, the first-order expansion is

$$\Delta\mathcal{L} = -\sum_j \langle H_j, \Delta W_j^{\mathrm{all}}\rangle + r, \qquad H_j = \sum_t a_{j,t}, \tag{A.12}$$

where r is the remainder. Consider a zero-mean analysis model with uncorrelated error coordinates and per-coordinate variance $\kappa_j 2^{-2b_j}$. Conditional on the gradients, the linear term satisfies

$$\mathbb{E}_Q[(\Delta\mathcal{L}_{\mathrm{lin}})^2] = \sum_j \kappa_j \|H_j\|_F^2 2^{-2b_j}. \tag{A.13}$$

Proof. Expand the square over expert and parameter coordinates. Zero cross-covariance removes distinct-coordinate terms. Each remaining term is its squared gradient coefficient multiplied by the corresponding error variance.

A step proportional to 2^{-b} motivates the squared dependence 2^{-2b}. TAP-MoE combines that dependence with task-reweighted local energy in Equation (5). This analysis explains the precision factor. The actual quantizer is evaluated through complete-model calibration, including changed activations and terms omitted by the local approximation.

A.4 Continuous Budgeted Precision Allocation

We derive the optimality conditions, the interior allocation, and the coordinate-bound cases of Theorem 2.

Proof of Theorem 2. Existence and uniqueness. The feasible set is a nonempty closed convex subset of the finite box $[b_{\min}, b_{\max}]^J$, hence compact. For $R_j(b) = w_j 2^{-2b}$,

$$R_j'(b) = -2\ln(2) w_j 2^{-2b}, \qquad R_j''(b) = 4(\ln 2)^2 w_j 2^{-2b} > 0. \tag{A.14}$$

Continuity gives existence, and strict convexity gives uniqueness.

Optimality conditions. Introduce nonnegative multipliers λ, α_j, ξ_j for storage and coordinate bounds. The Lagrangian is

$$\begin{aligned}\mathscr{L}(b, \lambda, \alpha, \xi) = \sum_j R_j(b_j) + \lambda\left(\sum_j n_j b_j - B\right) \\ + \sum_j \alpha_j(b_{\min} - b_j) + \sum_j \xi_j(b_j - b_{\max}).\end{aligned} \tag{A.15}$$

Stationarity and complementary slackness require

$$R_j'(b_j) + \lambda n_j - \alpha_j + \xi_j = 0, \tag{A.16}$$

$$\alpha_j(b_j - b_{\min}) = 0, \qquad \xi_j(b_{\max} - b_j) = 0, \tag{A.17}$$

$$\lambda\left(\sum_j n_j b_j - B\right) = 0. \tag{A.18}$$

At an interior coordinate, both bound multipliers vanish, giving $-R'_j(b_j)/n_j = \lambda$. At the lower bound $\xi_j = 0$, so $-R'_j(b_j)/n_j \le \lambda$. At the upper bound $\alpha_j = 0$, giving the reverse inequality.

Budget boundaries. Put $n_{\mathrm{tot}} = \sum_j n_j$. If $B = n_{\mathrm{tot}} b_{\min}$, the only feasible vector assigns the lower bound everywhere. If $B \ge n_{\mathrm{tot}} b_{\max}$, decreasing penalties make the all-upper-bound vector optimal. Between these endpoints, any slack permits an increase in an unsaturated coordinate and a decrease in the objective. Storage is therefore active. An intermediate budget admits a strictly feasible point, so the convex KKT conditions characterize the optimum. At the minimum-budget endpoint, lower-bound multipliers can satisfy the same one-sided conditions.

Interior solution. Let $r_j = w_j/n_j$ and $\omega_j = n_j/n_{\mathrm{tot}}$. Equation (6) gives

$$b_j^* = \frac{1}{2}\log_2 r_j + \frac{1}{2}\log_2 \frac{2\ln 2}{\lambda^*}. \tag{A.19}$$

Averaging with weights ω_j and using $\sum_j \omega_j b_j^* = \bar{b}$ eliminates the common constant and yields Equation (7).

For an intermediate budget, the bound-aware representation is

$$b_j(\lambda) = \mathrm{clip}_{[b_{\min}, b_{\max}]}\left[\frac{1}{2}\log_2 \frac{2\ln(2) w_j}{\lambda n_j}\right]. \tag{A.20}$$

Total storage is continuous and nonincreasing in λ, enabling scalar search. The minimizing bit vector remains unique even when an interval of multipliers describes a boundary solution.

A.5 Allocation Benefit and Stability

The interior solution gives an explicit comparison with uniform precision. We then bound the effect of allocation-weight estimation error.

Proposition A.3 (Benefit of differentiated precision)**.** Suppose the continuous optimum is interior and storage is active. With $r_j = w_j/n_j$ and $\omega_j = n_j/n_{\mathrm{tot}}$,

$$\mathcal{J}_\tau^* = n_{\mathrm{tot}} 2^{-2\bar{b}} \prod_j r_j^{\omega_j}, \qquad \mathcal{J}_\tau^{\mathrm{uniform}} = n_{\mathrm{tot}} 2^{-2\bar{b}} \sum_j \omega_j r_j. \tag{A.21}$$

Consequently,

$$\frac{\mathcal{J}_\tau^{\mathrm{uniform}}}{\mathcal{J}_\tau^*} = \frac{\sum_j \omega_j r_j}{\prod_j r_j^{\omega_j}} \ge 1, \tag{A.22}$$

with equality exactly when all sensitivity-to-size ratios agree.

Proof. Substitute Equation (7) into the objective. Each term becomes $n_j 2^{-2\bar{b}} \prod_i r_i^{\omega_i}$, giving the first identity in Equation (A.21). Uniform precision gives the second identity. The weighted arithmetic-geometric mean inequality proves Equation (A.22), including its equality condition.

Sensitivity heterogeneity is the source of improvement within this allocation model. With active coordinate bounds, the constrained optimum is still no worse than any feasible uniform allocation, although the interior closed form need not apply. Finite palettes use the same resource interpretation through the incremental priority in Appendix B.2.

Proposition A.4 (Stability under multiplicative errors in allocation weights)**.** Suppose $w_j/(1+\epsilon) \le \widehat{w}_j \le (1+\epsilon) w_j$ for $\epsilon \ge 0$. Let b^* and $\widehat{b}$ minimize the true and estimated objectives over the same nonempty feasible set. Then

$$\mathcal{J}(\widehat{b}) \le (1+\epsilon)^2 \mathcal{J}(b^*). \tag{A.23}$$

Proof. Nonnegative penalties give $\mathcal{J}(b) \le (1+\epsilon)\widehat{\mathcal{J}}(b)$ and $\widehat{\mathcal{J}}(b) \le (1+\epsilon)\mathcal{J}(b)$ for every feasible vector. Hence

$$\mathcal{J}(\widehat{b}) \le (1+\epsilon)\widehat{\mathcal{J}}(\widehat{b}) \le (1+\epsilon)\widehat{\mathcal{J}}(b^*) \le (1+\epsilon)^2 \mathcal{J}(b^*). \tag{A.24}$$

If the estimated objective is solved within additive tolerance δ, the bound gains $(1+\epsilon)\delta$. Statistical and optimization errors thus contribute separately. In the deployed finite candidate pool, complete-model calibration NLL remains the selection criterion.

On the same scored-token distribution, let $\Delta\ell = \ell(Q(W)) - \ell(W)$ and $D_{\text{true}} = \mathbb{E}[(\Delta\ell)^2] < \infty$. Then

$$|\mathbb{E}[\Delta\ell]| \le \sqrt{D_{\text{true}}}, \qquad \frac{\text{PPL}(Q(W))}{\text{PPL}(W)} = \exp(\mathbb{E}[\Delta\ell]). \tag{A.25}$$

The first relation is Cauchy-Schwarz, and the second follows from the PPL definition. These identities concern actual model loss. The surrogate provides a tractable priority, and calibration evaluates its effect on the quantized network.

A.6 Conditional Expert Access under Byte Constraints

The access analysis combines conditional expert demand with the byte costs produced by precision allocation.

For observed source assignment $s = S_{l-1}(t)$ and class c, define

$$\begin{aligned} q_{l,e}(s,c) &= \mathbb{P}(e \in S_l(t) \mid S_{l-1}(t) = s, c(t) = c) \\ &= \sum_{s' : e \in s'} \mathbb{P}(S_l(t) = s' \mid S_{l-1}(t) = s, c(t) = c). \end{aligned} \tag{A.26}$$

The implementation stores pairwise counts instead of the full set-conditioned table. Equation (8) aggregates these counts into a ranking score. Inclusion probabilities sum to K_l under native top-K_l routing. The ranking score is used for ordering, not as a normalized categorical distribution.

Proposition A.5 (Conditional coverage)**.** Let $Z = (s, c)$ and let $P(Z)$ be a selected prefetch set. Its conditional expected hit count is

$$\mathbb{E}[|P(Z) \cap S_l(t)| \mid Z] = \sum_{e \in P(Z)} q_{l,e}(Z). \tag{A.27}$$

For equal-sized experts and a count budget m, selecting the m largest conditional probabilities maximizes this quantity. Its expected coverage is at least that of any condition-independent set of m experts.

Proof. Conditioned on Z, the chosen set is fixed. Exchanging finite sums gives

$$\mathbb{E}\left[\sum_{e \in P(Z)} \mathbf{1}\{e \in S_l(t)\} \mid Z\right] = \sum_{e \in P(Z)} \mathbb{P}(e \in S_l(t) \mid Z). \tag{A.28}$$

The largest m terms maximize the sum. At each Z, this value dominates that of any fixed feasible set. Taking expectation preserves the inequality. The gain is strict when the fixed set is suboptimal on a positive-probability collection of conditions.

Normalizing by the native selected count gives per-expert recall:

$$\text{Hit}_l(m) = \frac{1}{K_l} \mathbb{E}_Z \left[\sum_{r=1}^{m} q_{l,(r)}(Z)\right], \tag{A.29}$$

where $q_{l,(r)}$ is the rth largest coordinate. Increasing m adds $\mathbb{E}[q_{l,(m+1)}]/K_l \ge 0$. The increments are nonincreasing, so recall is nondecreasing and discretely concave in admitted count.

Proof of Equation 9. For fixed positive sizes, let $\mathcal{F}_C = \{P : \sum_{e \in P} s_e \le C\}$. Every fixed $P \in \mathcal{F}_C$ and condition Z satisfy

$$\max_{A \in \mathcal{F}_C} \sum_{e \in A} q_e(Z) \ge \sum_{e \in P} q_e(Z). \tag{A.30}$$

Taking expectations and maximizing the right side over fixed sets proves the result.

The inequality describes the information advantage under a common byte budget. The finite planner uses empirical statistics to approximate the conditional choices.

Base preloading and residual demand. A base set R_l and speculative allowance β_l^{pf} obey

$$\sum_{e \in R_l} s_{l,e} + \beta_l^{\text{pf}} \le C_l, \quad \sum_{e \in P_l(t)} s_{l,e} \le \beta_l^{\text{pf}}, \quad P_l(t) \cap R_l = \varnothing. \tag{A.31}$$

With the speculative coverage mask fixed, let d_e count uncovered calibration requests. The base-set subproblem is

$$\max_{R \subseteq \mathcal{E}_l} \sum_{e \in R} d_e \quad \text{subject to} \quad \sum_{e \in R} s_{l,e} \leq C. \tag{A.32}$$

Each expert contributes a fixed value and consumes its payload once, giving a 0/1 knapsack. Alternating residual-demand estimation and base-set selection evaluates their combined coverage using the expert sizes already selected by quantization.

The activation-weighted mean precision is

$$\bar{b}^{\tau}_{\text{active}} = \frac{\sum_j f_j^{\tau} n_j b_j}{\sum_j f_j^{\tau} n_j}. \tag{A.33}$$

Thus equal pool-wide average precision can produce different active payloads. Access planning uses the chosen sizes to serve the task-dependent demand.

Let $A_l(t)$ contain the experts available when their computation is required. The remaining routed-weight payload is

$$B_l^{\text{miss}}(t) = \sum_{e \in S_l(t) \setminus A_l(t)} s_{l,e}. \tag{A.34}$$

For $A_l = R_l \cup P_l$, define $\text{Miss}_l = \mathbb{E}[|S_l \setminus A_l|]/K_l$. Inclusion-exclusion gives

$$\text{Miss}_l = 1 - \text{Cov}_l^{\text{base}} - \text{Hit}_l^{\text{spec}} + \text{Ovl}_l. \tag{A.35}$$

Here $\text{Cov}_l^{\text{base}} = \mathbb{E}[|R_l \cap S_l|]/K_l$, $\text{Hit}_l^{\text{spec}} = \mathbb{E}[|P_l \cap S_l|]/K_l$, and $\text{Ovl}_l = \mathbb{E}[|R_l \cap P_l \cap S_l|]/K_l$. The overlap vanishes when speculative admission excludes the base set. With mixed precision, saved bytes depend on the identities of the covered experts as well as their count.

A.7 Activated Means and Output Compensation

Compensation uses the same task mixture as allocation, but its mean conditions on expert activation. We first derive that mean and then analyze the constant correction.

Let A_j denote expert activation and suppose $f_j^{\tau} > 0$. Bayes' rule gives

$$\mathbb{P}_{\tau}(c \mid A_j) = \frac{\pi_{\tau}(c) p_j^{\tau,(c)}}{f_j^{\tau}}. \tag{A.36}$$

Total expectation within the activated population yields

$$\mathbb{E}_{\tau}[z_j \mid A_j] = \frac{\sum_c \pi_{\tau}(c) p_j^{\tau,(c)} \mu_j^{\tau,(c)}}{\sum_c \pi_{\tau}(c) p_j^{\tau,(c)}}. \tag{A.37}$$

Matching shared and target conditional probabilities and means recovers the population version of Equation (11). The activation factor makes the estimate represent inputs seen by the expert.

For empirical class counts N_c, activated counts $n_{j,c}$, and feature sums $Z_{j,c}$, let $\pi_g(c) = N_c / \sum_{c'} N_{c'}$ and $w_c = \pi_{\tau}(c)/\pi_g(c)$ on supported classes. Substituting empirical conditionals gives

$$\widehat{\mu}_j^{\tau} = \frac{\sum_c w_c Z_{j,c}}{\sum_c w_c n_{j,c}}. \tag{A.38}$$

Both numerator and denominator must be reweighted. Averaging activated means using class proportions alone would omit differences in expert participation.

Proposition A.6 (Stability of an activated mean)**.** Let the true numerator and activation probability be N, f, with $f > 0$. If $\widehat{f} \geq a > 0$, $\|\widehat{N} - N\|_2 \leq \delta_N$, $|\widehat{f} - f| \leq \delta_f$, and $\|N/f\|_2 \leq M$, then

$$\left\| \frac{\widehat{N}}{\widehat{f}} - \frac{N}{f} \right\|_2 \leq \frac{\delta_N + M\delta_f}{a}. \tag{A.39}$$

Proof. Expand the difference as $(\widehat{N} - N)/\widehat{f} + (N/f)(f - \widehat{f})/\widehat{f}$, then apply the triangle inequality and the lower bound on $\widehat{f}$.

Limited activation support can amplify estimation error. This motivates shrinking weak task residuals and using the pooled activated mean when weighted support vanishes.

Proposition A.7 (Optimal constant projection correction)**.** Fix ΔW_j and an activation event A_j of positive probability. For an activated-input distribution with finite second moment, the unique constant minimizing

$$\mathbb{E}[\|\Delta W_j z_j - u\|_2^2 \mid A_j] \tag{A.40}$$

is $u^* = \Delta W_j \mathbb{E}[z_j \mid A_j]$.

Proof. Condition on A_j and write $a = \Delta W_j z_j$. The variance decomposition gives

$$\mathbb{E}\|a - u\|_2^2 = \mathbb{E}\|a - \mathbb{E}a\|_2^2 + \|u - \mathbb{E}a\|_2^2. \tag{A.41}$$

The second term has the unique minimum $u = \mathbb{E}a$. Linearity gives $\mathbb{E}a = \Delta W_j \mathbb{E}[z_j \mid A_j]$.

The correction target separates into a pooled offset and a task-dependent residual:

$$\Delta W_j \mu_j^\tau = \Delta W_j \mu_j^g + \Delta W_j (\mu_j^\tau - \mu_j^g). \tag{A.42}$$

Equation (12) controls the second term while retaining the pooled offset. Zero residual strength yields pooled correction, whereas removing the entire correction is a separate alternative. Appendix B.4 describes their offline selection.

If earlier expert projections are also quantized, the final projection's input changes from z_j to $\widehat{z}_j$. The full local discrepancy is

$$W_j^o z_j - Q_{b_j}(W_j^o)\widehat{z}_j = \Delta W_j z_j + Q_{b_j}(W_j^o)(z_j - \widehat{z}_j). \tag{A.43}$$

The constant-correction result identifies the mean of the first term. Complete-model calibration evaluates the correction in the presence of changed inputs and subsequent error propagation.

A.8 Execution Equivalence

Precision and compensation specify the numerical model in Equation (13). Prefetching and residency provide its expert weights. The following property explains why access policies can be compared while holding quantized-model quality fixed.

Proposition A.8 (Availability invariance)**.** Fix quantized weights, decoding metadata, correction vectors, input sequence, numerical operators, and native accumulation order. If cache hits return the exact reference-decoded weights and every selected expert is evaluated, cached and uncached availability paths produce identical model outputs.

Proof. At a selected invocation, a cache hit and a fresh decode supply the same matrices. Identical inputs, corrections, and arithmetic give identical expert outputs. Aggregation in the same order gives identical block outputs. Induction over layers yields identical logits and KV state, and induction over controlled decoding steps gives identical request outputs. The result also holds for deterministic natural decoding with fixed token selection and tie handling.

The proposition applies to a fixed quantized model. Quantization may change later hidden states, while the pretrained routing rule remains unchanged. Access changes when the selected weights become available, not the computation performed with those weights.

The results support the three decisions at their respective levels. Statistical synthesis supplies expert demand and sensitivity. Budgeted allocation determines the representation, conditional coverage explains the value of access information, and activated means determine the local correction target. The deployment experiments evaluate their combined outcome.

B Algorithms

The algorithms specify how TAP-MoE constructs a task-conditioned deployment and executes it. The shared profile, selected quantized weights, access policy, and output corrections are used throughout.

B.1 Preparation and Inference

Algorithm B.1 connects the three mechanisms through their inputs and outputs. The access planner receives the selected expert sizes, and compensation receives the corresponding output-projection residuals. The online loop then evaluates that fixed model.

Algorithm B.1 TAP-MoE preparation and inference

Require: Pretrained weights W, shared statistics $\mathcal{S}_g$, target calibration inputs, storage budget B, runtime budget M
Ensure: Outputs of the task-conditioned quantized MoE
Offline preparation
Estimate $\widehat{\pi}_\tau$ and synthesize $\widehat{f}^\tau, \widehat{\gamma}^\tau, \widehat{\mu}^\tau$
Select b^τ by Algorithm B.3 and form $Q_{b^\tau}(W)$
Compute expert payloads s_j and projection residuals ΔW_j
Plan prefetching and residency from $\mathcal{S}_g$, $\widehat{\pi}_\tau$, $\{s_j\}$, and target access traces
Select task, pooled, or zero correction by Equation (B.6)
Online inference
for each request **do**
    Initialize the request-local expert cache
    for each forward call and sparse layer in model order **do**
        Obtain native selected experts and routing coefficients
        Prepare admitted next-layer experts from current class and source routing
        Obtain each selected expert from residency, prefetch, or on-demand loading
        Compute its output, add u_j^τ, and multiply by the native coefficient
        Aggregate selected outputs and the native shared branch
    end for
    Release request-local cache state
end for

Statistical collection, precision search, and risk screening are completed offline. Online inference performs the selected access and correction operations along the native expert computation.

B.2 Shared Statistics and Precision Selection

Algorithm B.2 collects the sufficient statistics used by the profile. Class counts and activated counts have different roles, as required by the synthesis identities.

Algorithm B.2 Collecting class-conditional expert statistics

Require: Frozen weights W, shared calibration inputs $\mathcal{D}_g$, token-class map c
Ensure: Class counts, expert counts, local energies, feature sums, and coactivations
Initialize $N_c, n_{j,c}, H_{j,c}, Z_{j,c}, N_{l,c}(i,e)$ to zero
for each input sequence $x \in \mathcal{D}_g$ **do**
    Run the native forward pass and obtain sequence-loss adjoints
    for each counted position t with class c_t **do**
        $N_{c_t} \leftarrow N_{c_t} + 1$
        for each selected invocation $j = (l, e)$ **do**
            $n_{j,c_t} \leftarrow n_{j,c_t} + 1$
            $H_{j,c_t} \leftarrow H_{j,c_t} + \|a_{j,t}\|_F^2$ using Equation (A.10)
            $Z_{j,c_t} \leftarrow Z_{j,c_t} + z_{j,t}$
        end for
        Accumulate adjacent-layer coactivation and source-expert counts
    end for
end for
$\widehat{p}_j^{(c)} \leftarrow n_{j,c}/N_c$, $\widehat{\gamma}_j^{(c)} \leftarrow H_{j,c}/N_c$ on supported classes
$\widehat{\mu}_j^{(c)} \leftarrow Z_{j,c}/n_{j,c}$ on supported activated classes

Unsupported entries retain a support mask. When a synthesized activated mean has zero weighted support, the pooled mean supplies the fallback. An expert never activated in pooled calibration has zero mean under this empirical convention. Population statements retain their positive-support assumptions.

Precision candidates begin at the minimum allowed bit width. For a next-palette increment $b_j \to b'_j$, the priority is

$$d_j(b_j, b'_j) = \frac{R_j(b_j) - R_j(b'_j)}{n_j(b'_j - b_j)}, \qquad R_j(b) = w_j 2^{-2b}. \tag{B.1}$$

At each step, the highest-priority feasible increment is selected until no further increment fits. The candidate pool $\mathcal{K}_B$ includes these maps and the feasible refinements considered during offline preparation. Algorithm B.3 selects the deployed representation from that pool.

Algorithm B.3 Task-aware precision selection

Require: Shared sensitivities, target mixture, palettes and contrasts, budget B, feasible refinements, quantizer Q, calibration loss $\widehat{\mathcal{L}}_\tau$
Ensure: Selected precision map and quantized experts
1: Synthesize $\widehat{\gamma}_j^\tau$ using Equation (3)
2: Apply a positive numerical floor and geometric-mean normalization
3: Form allocation weights for each candidate contrast
4: Construct feasible maps using Equation (B.1)
5: Combine distinct maps and supplied feasible refinements into $\mathcal{K}_B$
6: **for** each $b \in \mathcal{K}_B$ satisfying the required storage budget **do**
7: Evaluate $\widehat{\mathcal{L}}_\tau(Q_b(W))$
8: **end for**
9: Select the minimum-loss candidate and fix its weights and bit map

For inputs x with scored target sets $T(x)$, the complete-model calibration loss is

$$\widehat{\mathcal{L}}_\tau(Q_b(W)) = \frac{\sum_x \sum_{r \in T(x)} \ell_r(x, Q_b(W))}{\sum_x |T(x)|}. \tag{B.2}$$

This selection evaluates the actual quantizer and network. The selected bit map is then used by access planning and compensation.

B.3 Prefetching and Residency

The prefetch planner evaluates byte-feasible divisions between base preloading and speculative candidates. Current class and source assignments determine the ranking in Equation (8). Uncovered requests provide values for the knapsack in Equation (A.32). The chosen window satisfies Equation (A.31).

Request-local residency retains packed or decoded representations according to target reuse, retrieval savings, and size. Together with transient prefetch weights, these representations satisfy Equation (10). Algorithm B.4 uses the calibrated residency policy as an offline input and applies it to the native selected experts.

A zero speculative allowance bypasses ranking but leaves residency available. Packed hits save transfer, and decoded hits save transfer and unpacking. On a miss, the same selected expert is obtained. Proposition A.8 therefore applies to access-policy comparisons with fixed weights and corrections.

B.4 Output Compensation and Risk Screening

The task correction adds a residual to the pooled offset. Its reliability is assessed from separate calibration subsets. Let $r_j^{A,\tau} = \widehat{\mu}_j^{A,\tau} - \widehat{\mu}_j^{A,g}$ and let $r_j^{B,\tau}$ be the corresponding estimate from the other subset. Their agreement in output space gives

$$\beta_j^\tau = \max\{0, \cos(\Delta W_j r_j^{A,\tau}, \Delta W_j r_j^{B,\tau})\} \frac{n_{\mathrm{eff},j}}{n_{\mathrm{eff},j} + \kappa}, \qquad \kappa > 0, \tag{B.3}$$

where

$$n_{\mathrm{eff},j} = \frac{(\sum_c w_c n_{j,c})^2}{\sum_c w_c^2 n_{j,c}}. \tag{B.4}$$

Algorithm B.4 Expert access during inference

Require: Quantized experts, base sets, speculative allowances, calibrated residency policy, correction vectors, runtime budget
Ensure: Sparse-block outputs of the fixed quantized model

```
for each sparse layer l in model order do
    Obtain native selected sets and coefficients for current hidden states
    Stage base weights in the corresponding layer window
    if the next-layer speculative allowance is positive then
        Rank missing candidates using current classes and source assignments
        Request candidates that fit the remaining byte allowance
    end if
    for each selected expert in native accumulation order do
        if its exact decoded representation is resident then
            Reuse decoded weights
        else
            Obtain packed weights from cache or on-demand loading
            Wait for readiness and decode the selected representation
        end if
        Compute expert output and add the selected correction
        Apply the native routing coefficient and accumulate
        Retain representations admitted by the fixed residency policy
    end for
    Add the native shared-branch output when present
    Release completed transient storage
end for
```

The weights w_c are defined in Appendix A.7. Agreement measures whether the task-residual direction is stable after projection by ΔW_j, while effective support controls shrinkage. Zero denominators or zero-vector cosine arguments yield zero stability.

Calibration NLL selects the residual strength η. Validation then compares task, pooled, and zero correction. Define

$$\Delta_{t0} = \widehat{\mathcal{L}}_0 - \widehat{\mathcal{L}}_t, \qquad \Delta_{tg} = \widehat{\mathcal{L}}_g - \widehat{\mathcal{L}}_t, \qquad \Delta_{g0} = \widehat{\mathcal{L}}_0 - \widehat{\mathcal{L}}_g. \tag{B.5}$$

Positive differences favor task over zero, task over pooled, and pooled over zero, respectively. With paired lower confidence bounds L_{t0}, L_{tg}, L_{g0}, the workload-level choice is

$$u_j^{\text{selected}} = \begin{cases} u_j^{\tau}(\eta), & L_{t0} > 0 \text{ and } L_{tg} > 0, \\ \Delta W_j \widehat{\mu}_j^{g}, & \text{otherwise if } L_{g0} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{B.6}$$

The pooled branch keeps the shared offset, whereas the zero branch removes the correction operation. The chosen vector is added after the output projection and before the routing coefficient, as shown in Figure 1 and Equation (13).

B.5 Computational Complexity

For $|\mathcal{C}|$ classes and layerwise expert counts $|\mathcal{E}_l|$, scalar activation and sensitivity statistics require $O(|\mathcal{C}| \sum_l |\mathcal{E}_l|)$ entries. Adjacent-layer pair counts require $O(|\mathcal{C}| \sum_l |\mathcal{E}_{l-1}||\mathcal{E}_l|)$ entries. Feature sums additionally scale with each expert's intermediate width. Task synthesis is linear in the stored arrays.

Enabled prefetch ranking costs $O(K_{l-1}|\mathcal{E}_l| + |\mathcal{E}_l| \log |\mathcal{E}_l|)$ operations, followed by a linear byte-admission scan. Output compensation is a vector addition per selected invocation. Sensitivity collection, precision candidate evaluation, and correction screening remain offline.

C Additional Experimental Results

This appendix expands the complete-deployment comparisons in Section 5.2 with metric definitions, uncertainty intervals, resource measurements, and downstream results.

C.1 Models and Metrics

Table C.1 summarizes the three native architectures. They differ in expert count, top-K selection, and shared-expert structure. TAP-MoE quantizes routed gate, up, and down projections while retaining routers and fixed branches. The tokenizer-specific map distinguishes 14 character-based classes. Experiments use NVIDIA RTX A6000 for general-text, code, and mathematics workloads.

Table C.1: Native model architecture. Shared* counts fine-grained expert equivalents.

Model	Sparse layers	Experts per layer	Top-K	Shared*	Routed total
OLMoE	16	64	8	0	1024
Qwen1.5-MoE	24	60	4	4	1440
DeepSeek-MoE	27	64	6	2	1728

Document-prefix PPL uses WikiText-2, MBPP reference programs, and MATH reference solutions. Contiguous WikiText-2 retains the same task-specific precision maps. HumanEval, GSM8K, and MMLU evaluate generated programs, mathematical answers, and multiple-choice predictions, respectively.

For document NLL sums L_i and scored-token counts T_i,

$$\mathrm{PPL} = \exp\left(\frac{\sum_i L_i}{\sum_i T_i}\right). \tag{C.1}$$

Within each setting, let P^* and T^* be the best reference PPL and TPOT in the matched-storage comparison set. Relative PPL reduction and decoding speedup are

$$g = 100\left(1 - \frac{\mathrm{PPL}_{\mathrm{TAP}}}{P^*}\right), \qquad s = \frac{T^*}{\mathrm{TPOT}_{\mathrm{TAP}}}. \tag{C.2}$$

The abstract reports the minimum and maximum across the nine model-domain settings. Table C.2 gives the corresponding point estimates and paired 95% intervals.

Table C.2: Relative PPL reduction g and decoding speedup s with paired 95% intervals. The speed reference is eMoE throughout.

Model	Domain	PPL ref.	g (%)	95% CI (%)	s	95% CI (×)
OLMoE	general	GPTQ4	5.0	[4.46, 5.65]	1.65	[1.63, 1.67]
	code	EdgeMoE	3.0	[2.69, 3.31]	1.75	[1.69, 1.80]
	math	GPTQ4	5.0	[4.77, 5.25]	1.70	[1.67, 1.72]
Qwen1.5	general	GPTQ4	3.5	[3.15, 3.85]	1.55	[1.53, 1.57]
	code	GPTQ4	1.2	[0.37, 2.05]	1.45	[1.41, 1.48]
	math	GPTQ4	5.0	[4.59, 5.43]	1.60	[1.56, 1.64]
DeepSeek	general	GPTQ4	2.5	[2.21, 2.78]	1.40	[1.32, 1.49]
	code	EdgeMoE	2.0	[1.71, 2.28]	1.25	[1.24, 1.26]
	math	EdgeMoE	1.0	[0.84, 1.15]	1.40	[1.35, 1.47]

GPTQ4 or EdgeMoE supplies the quality reference, while eMoE is the fastest evaluated deployment reference. The next tables report both outcomes for all evaluated methods.

C.2 Complete Deployment Comparisons

Tables C.3, C.4, and C.5 report OLMoE, Qwen1.5-MoE, and DeepSeek-MoE. For methods shared with Table 1, Tables C.3–C.5 use the same PPL results while additionally reporting deployment latency and the full set of deployment references. EARTH retains its larger expert-weight payload and remains outside the matched-storage ranking.

Table C.3: OLMoE deployment comparisons. PPL and TPOT (ms/token) are reported for each domain. EARTH uses a larger expert-weight payload.

Method	General PPL	General TPOT	Code PPL	Code TPOT	Math PPL	Math TPOT
Uniform4	8.3536	155.80	4.7200	151.30	3.4467	151.10
GPTQ4	8.2918	156.20	4.3584	151.70	3.3911	150.90
ST-MoE	8.3460	206.24	4.7125	188.58	3.4480	179.82
EdgeMoE	9.6080	127.46	4.1474	104.56	3.4027	113.51
EARTH	7.9031	1161.12	4.0613	1169.81	3.2903	1123.62
MoE-OPU	8.8200	635.26	5.2433	595.26	4.3448	583.78
Pre-gated	9.3200	149.35	5.2200	140.86	3.8300	140.12
eMoE	10.6500	90.71	5.9000	93.09	4.2400	89.73
TAP	**7.8772**	**54.98**	**4.0229**	**53.19**	**3.2215**	**52.78**

Table C.4: Qwen1.5 deployment comparisons. PPL and TPOT (ms/token) are reported for each domain. EARTH uses a larger expert-weight payload.

Method	General PPL	General TPOT	Code PPL	Code TPOT	Math PPL	Math TPOT
Uniform4	7.0631	160.60	5.8831	139.40	2.2237	165.30
GPTQ4	6.6535	161.10	4.7127	138.90	2.0633	165.90
ST-MoE	6.9578	208.56	5.7179	232.18	2.1760	213.58
EdgeMoE	6.6546	122.50	4.7912	120.83	2.0819	123.62
EARTH	6.4869	948.57	4.7154	944.17	1.9685	989.10
MoE-OPU	8.6398	778.03	5.6977	731.33	2.3766	840.59
Pre-gated	7.4900	169.47	5.9500	135.03	2.3700	134.67
eMoE	8.4400	108.51	6.6800	102.70	2.6500	110.20
TAP	**6.4207**	**70.01**	**4.6562**	**70.82**	**1.9601**	**68.87**

Table C.5: DeepSeek deployment comparisons. PPL and TPOT (ms/token) are reported for each domain. EARTH uses a larger expert-weight payload.

Method	General PPL	General TPOT	Code PPL	Code TPOT	Math PPL	Math TPOT
Uniform4	6.7957	206.70	3.2367	203.50	2.9405	202.70
GPTQ4	6.6932	205.90	3.0271	204.10	2.8872	203.30
ST-MoE	6.7520	297.16	3.2158	290.12	2.9338	325.63
EdgeMoE	6.6981	165.99	3.0263	168.79	2.8625	178.77
EARTH	6.5681	1045.72	3.0204	1060.92	2.8449	1066.79
MoE-OPU	7.0600	867.48	3.3100	866.92	3.0600	853.35
Pre-gated	7.3300	205.76	3.4300	211.70	3.1700	188.70
eMoE	8.0700	152.97	3.8700	137.60	3.4900	147.79
TAP	**6.5259**	**109.26**	**2.9658**	**110.08**	**2.8339**	**105.56**

TAP-MoE improves on the strongest matched-storage reference for both metrics across all three models. DeepSeek code gives the smallest decoding speedup, 1.25×, and DeepSeek mathematics gives the smallest PPL reduction, 1.0%. These complete comparisons correspond to the ranges summarized in the abstract.

C.3 Runtime Memory and Contiguous-Text Quality

Table C.6 reports peak allocated and reserved memory for the same TAP-MoE deployments used in the latency comparison. Every reported run satisfies its model-specific runtime budget. The measurements include the selected representation and access policy.

Table C.6: Peak allocated and reserved memory for the reported TAP deployments.

Model	Domain	Peak allocated GiB	Peak reserved GiB
OLMoE	general	1.833	1.961
	code	1.863	1.963
	math	1.879	1.982
Qwen1.5	general	4.530	4.844
	code	4.528	4.844
	math	4.523	4.883
DeepSeek	general	3.582	3.980
	code	3.597	3.980
	math	3.587	3.980

The quality improvement extends to contiguous text (Table C.7). With unchanged task-specific precision maps, TAP-MoE lowers WikiText-2 PPL by approximately 1.0-3.3% relative to GPTQ4 and improves on Uniform4 in all three models. BF16 has the lowest PPL in this evaluation, showing the remaining quality gap at the selected compression level.

Table C.7: Contiguous WikiText-2 PPL (↓). Bold compares quantized methods.

Model	BF16	Uniform4	GPTQ4	TAP
OLMoE	7.4908	7.9348	7.9773	**7.7128**
Qwen1.5	7.2211	7.5179	7.3768	**7.2989**
DeepSeek	6.5120	6.6683	6.6152	**6.5490**

Both document-prefix and contiguous-text evaluation favor TAP-MoE over the quantized references. Their distinct scoring contexts provide complementary measurements of the same task-specific precision maps.

C.4 Downstream Performance

Table C.8 reports complete downstream scores. HumanEval uses pass@1 on generated code, GSM8K uses strict answer extraction, and MMLU uses question-level accuracy. TAP-MoE improves on GPTQ4 in every listed model and benchmark combination, extending the quality comparison beyond reference-text likelihood.

Table C.8: Downstream scores (%). HumanEval uses pass@1, GSM8K strict answer extraction, and MMLU question-level accuracy. BF16 is uncompressed. Bold marks the best quantized result.

Model	Method	HumanEval	GSM8K	MMLU
OLMoE	BF16	17.68	9.72	52.59
	Uniform4	12.80	8.30	51.32
	GPTQ4	13.41	8.52	51.39
	TAP	**16.46**	**9.64**	**52.02**
Qwen1.5	BF16	37.80	18.53	61.07
	Uniform4	34.15	14.90	58.24
	GPTQ4	34.76	15.90	59.64
	TAP	**37.20**	**18.12**	**60.92**
DeepSeek	BF16	28.66	17.30	44.86
	Uniform4	26.22	14.84	42.63
	GPTQ4	25.61	15.37	43.88
	TAP	**28.05**	**17.16**	**44.65**

The paired differences in Table C.9 retain the uncertainty of these comparisons. TAP-MoE improves on GPTQ4 in all nine model-benchmark combinations, with positive lower bounds for the reported paired 95% confidence intervals.

Table C.9: Paired downstream differences relative to GPTQ4. Differences and 95% intervals are in percentage points.

Model	Task	TAP (%)	GPTQ4 (%)	Δ (pp)	95% CI (pp)
OLMoE	HumanEval	16.46	13.41	+3.05	[0.43, 5.72]
	GSM8K	9.64	8.52	+1.12	[0.46, 1.73]
	MMLU	52.02	51.39	+0.63	[0.52, 0.78]
Qwen1.5	HumanEval	37.20	34.76	+2.44	[0.10, 4.79]
	GSM8K	18.12	15.90	+2.22	[1.61, 2.82]
	MMLU	60.92	59.64	+1.28	[1.12, 1.41]
DeepSeek	HumanEval	28.05	25.61	+2.44	[0.11, 4.81]
	GSM8K	17.16	15.37	+1.79	[1.12, 2.45]
	MMLU	44.65	43.88	+0.77	[0.62, 0.93]

PPL uncertainty is paired over documents, downstream uncertainty over questions, and runtime pairing retains repeated observations within document clusters. These intervals describe uncertainty within a comparison. The ranges in the abstract describe variation across model-domain settings.

D Ablation Studies and Additional Analysis

We examine the information in token classes and its contribution to precision allocation, compensation, and expert access within TAP-MoE.

D.1 Class-Conditional Expert Information

Let $q_l(e \mid c)$ and $q_l(e)$ be class-conditioned and global categorical predictors fitted on calibration data. Both predict an expert drawn uniformly from the native selected set. For held-out selected-invocation counts $n^{\text{eval}}_{l,e,c}$,

$$H_l^{\text{class}} = -\frac{\sum_{e,c} n^{\text{eval}}_{l,e,c} \log_2 q_l(e \mid c)}{\sum_{e,c} n^{\text{eval}}_{l,e,c}}. \tag{D.1}$$

The global score substitutes $q_l(e)$. These categorical predictors sum to one, unlike the inclusion probabilities in Equation (A.1). The metric is therefore measured in bits per selected expert.

Class conditioning improves every pooled and same-source comparison in Table D.1. For the pooled fit, reductions range from 0.402 to 0.781 bits per selected expert, showing information beyond global expert frequency before a bit map is considered.

Table D.1: Held-out routing cross-entropy in bits per selected expert (↓). Global and class-conditioned predictors are compared within each fitting pool.

Model	Domain	Pooled global	Pooled class	Same-source global	Same-source class
OLMoE	general	6.414	5.633	5.797	5.352
	code	5.103	4.618	4.996	4.533
	math	6.385	5.726	5.654	5.234
Qwen1.5	general	5.956	5.455	5.857	5.239
	code	5.739	5.127	5.723	5.089
	math	5.948	5.457	5.721	5.072
DeepSeek	general	6.016	5.543	5.902	5.328
	code	5.840	5.327	5.825	5.288
	math	5.950	5.548	5.709	5.158

Same-source fitting further improves prediction. TAP-MoE instead reuses shared conditionals and adapts their task-level weights. We next evaluate this allocation.

D.2 Task-Aware Precision Allocation

Figure 2 illustrates how class-weighted sensitivity guides precision. Sensitivities are combined on their original scale, and the budget is model-wide.

Table D.2 compares alternative allocation statistics. Frequency uses expert occurrence, while the other controls vary the source or task conditioning of sensitivity. TAP-MoE improves PPL by 0.65-3.28% over Pooled and by 1.20-5.49% over Mismatch at the same routed-weight budget. Frequency is also worse in every setting. These controls show that the assignment of precision to task-relevant experts matters beyond the use of mixed precision itself.

Shared synthesis gives lower PPL than Direct in every setting. Unlike same-source cross-entropy in Table D.1, this comparison evaluates quantized-model quality rather than expert-occurrence prediction.

Figure D.1 reports the bit compositions. At the same average precision, OLMoE code assigns six or eight bits to 186 experts, compared with 23 on general text.

Table D.2: Precision-allocation ablations at the same routed-weight budget (PPL, ↓). Pooled uses task-independent sensitivity, Direct uses target-specific estimation, and Mismatch another task profile.

Model	Domain	TAP	Pooled	Frequency	Direct	Mismatch
OLMoE	general	**7.8772**	8.1442	8.2054	7.9327	8.3344
	code	**4.0229**	4.1034	4.1285	4.0410	4.1660
	math	**3.2215**	3.3307	3.3592	3.2447	3.4085
Qwen1.5	general	**6.4207**	6.5711	6.6181	6.4561	6.6911
	code	**4.6562**	4.6929	4.7042	4.6663	4.7212
	math	**1.9601**	2.0265	2.0449	1.9736	2.0739
DeepSeek	general	**6.5259**	6.6342	6.6678	6.5551	6.7241
	code	**2.9658**	3.0050	3.0172	2.9747	3.0361
	math	**2.8339**	2.8525	2.8582	2.8382	2.8683

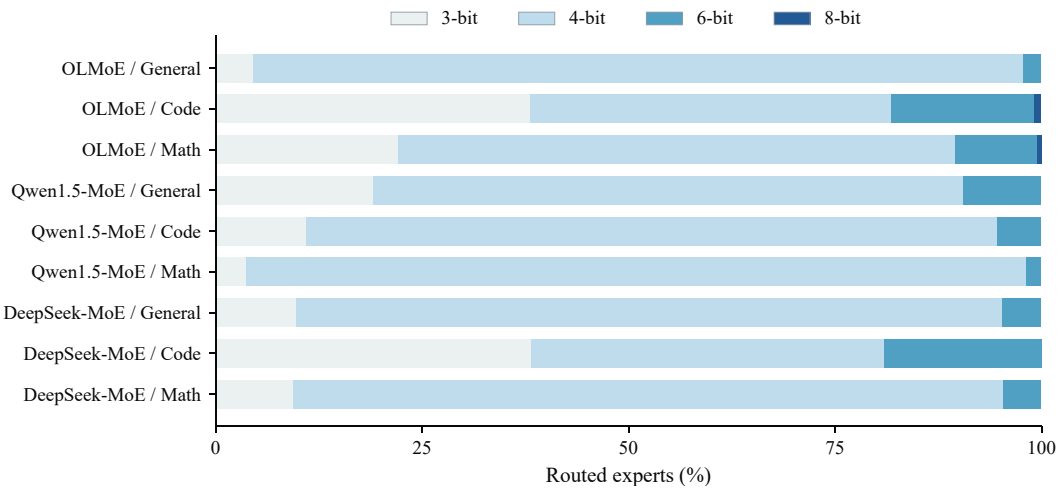


Figure D.1: Task-dependent precision composition at four average routed-expert bits. Bars give the fraction of experts at each bit width. The expert totals are 1024, 1440, and 1728 for OLMoE, Qwen, and DeepSeek.

D.3 Output Compensation

Table D.3 compares output corrections on fixed quantized weights. Task correction gives the lowest PPL throughout, improving by 0.10-0.92% relative to zero correction after precision has been selected.

All listed settings select task correction, costing 0.15, 0.25, and 0.35 ms/token for OLMoE, Qwen, and DeepSeek. Proposition A.7 gives its local mean target, and Equation (B.6) specifies offline selection.

Table D.3: Output-compensation PPL (↓) on identical quantized weights. The final column gives the online cost of the selected task correction.

Model	Domain	Zero	Pooled	Task	Correction cost ms/token
OLMoE	general	7.9258	7.9122	**7.8772**	0.150
	code	4.0475	4.0332	**4.0229**	0.150
	math	3.2514	3.2415	**3.2215**	0.150
Qwen1.5	general	6.4482	6.4419	**6.4207**	0.250
	code	4.6618	4.6582	**4.6562**	0.250
	math	1.9752	1.9707	**1.9601**	0.250
DeepSeek	general	6.5424	6.5333	**6.5259**	0.350
	code	2.9748	2.9721	**2.9658**	0.350
	math	2.8367	2.8350	**2.8339**	0.350

D.4 Prefetching and Residency

The access controls keep quantized weights and corrections fixed while changing how experts become available. At equal total pool capacity, task-conditioned caching reduces TPOT by 0.75-9.36% relative to pooled caching (Table D.4). The largest gain is on OLMoE code, from 61.87 to 56.08 ms/token before prefetching is added. Prefetching further improves six settings. OLMoE code decreases from 56.08 to 53.19 ms/token. Qwen code, Qwen mathematics, and DeepSeek code instead select caching alone. Task-conditioned residency thus remains useful when speculative loading does not further reduce latency. Proposition A.8 explains why these access choices preserve the fixed quantized computation.

Table D.4: Expert-access latency at equal total pool capacity (ms/token, ↓). PF denotes prefetching.

Model	Domain	No cache	Pooled	Task cache	Task+PF	Selected
OLMoE	general	58.39	57.54	55.83	**54.98**	cache+PF
	code	64.76	61.87	56.08	**53.19**	cache+PF
	math	63.38	60.73	55.43	**52.78**	cache+PF
Qwen1.5	general	71.06	70.80	70.27	**70.01**	cache+PF
	code	72.46	72.05	**70.82**	71.53	cache
	math	71.45	70.81	**68.87**	69.56	cache
DeepSeek	general	113.70	112.59	110.37	**109.26**	cache+PF
	code	112.56	111.94	**110.08**	110.74	cache
	math	111.20	109.79	106.97	**105.56**	cache+PF

D.5 Sensitivity to the Storage Budget

Table D.5 evaluates general text around the main four-bit operating point. Runtime and residency budgets remain fixed within each model. Increasing average precision from 3.75 to 4.25 bits lowers PPL in all three models, with a modest rise in TPOT. More precision reduces representation error but increases the weight volume served by the access policy.

Table D.5: Sensitivity to average routed-expert precision on general text with fixed runtime and residency budgets. Bold marks the lowest PPL and TPOT within each model.

Model	Mean bits	PPL	TPOT (ms/token)
OLMoE	3.750	7.9638	**52.23**
	4.000	7.8772	54.98
	4.250	**7.8181**	56.67
Qwen1.5	3.750	6.4721	**67.56**
	4.000	6.4207	70.01
	4.250	**6.3854**	72.11
DeepSeek	3.750	6.5651	**106.53**
	4.000	6.5259	109.26
	4.250	**6.5031**	111.45