CAN WE IMPROVE THE PERFORMANCE OF SPARSE ATTENTION IN GROUPED QUERY EXPERTS WITHOUT INCREASING EXPERTS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse attention architectures, such as Grouped Query Experts (GQE), significantly reduce inference costs by routing input tokens to a small, statically fixed subset of query experts. However, rigidly discarding the unselected experts inevitably leads to information loss and degraded downstream performance. To address this, we introduce HyperGQE, a dynamic attention mechanism that recovers lost representational capacity by generating an auxiliary "HyperExpert." Rather than blindly adapting selected weights, HyperGQE computes a selection embedding based on the average representation of the unselected experts, which conditions a hypernetwork to generate dynamic bottleneck matrices (D and U). This HyperExpert processes the hidden states into a supplementary query projection (q_{hyper}) that is integrated into a weighted attention slot alongside the standard routed experts. We evaluate HyperGQE on the SST-2 benchmark, observing that while standard GQE degrades accuracy to 75.92% compared to the dense GQA baseline (80.16%), HyperGQE not only fully recovers this gap but achieves 80.85% accuracy, exceeding the dense GQA baseline by 0.69 percentage points. Furthermore, hardware profiling demonstrates that HyperGQE achieves this superior accuracy using fewer active parameters (924.1M) than the dense baseline (946.0M), while maintaining competitive throughput and VRAM footprints across batch sizes. These results demonstrate that dynamically compensating for unselected experts allows sparse models to exceed the performance of dense counterparts without increasing the active parameter count beyond the dense baseline.

1 INTRODUCTION

The increasing computational cost of modern language models has motivated sparse architectures that activate only a subset of model components for each input. Mixture-of-Experts (MoE) models exploit this principle by conditionally activating a small number of experts while maintaining a substantially larger parameter space Shazeer et al. (2017); Fedus et al. (2021). However, applying sparsity to attention introduces a different challenge: the query transformation determines which information is retrieved from the key-value representations, and restricting this transformation to a small set of fixed expert parameters can limit the representational flexibility available after routing.

Grouped Query Experts (GQE) address this problem by introducing expertized query projections and routing each input toward a small subset of query experts. Given an input representation, a router computes expert scores and activates the top-K query experts. The resulting sparse query computation reduces the amount of active parameterization while allowing different inputs to use different query projections. Nevertheless, whenever a router drops the unselected experts, critical contextual information is irreversibly lost, directly causing downstream accuracy degradation.

We investigate whether the information lost during static routing can be actively recovered. We propose HyperGQE, a dynamic attention mechanism inspired by HyperMoE . Instead of merely applying a low-rank correction to the selected experts, HyperGQE explicitly compensates for what was left behind. It achieves this by taking the average embedding of the unselected experts and passing it through a selection network to condition a lightweight Hypernetwork. This Hypernetwork generates custom bottleneck matrices (D and U) to form a dynamic "HyperExpert" specifically

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.

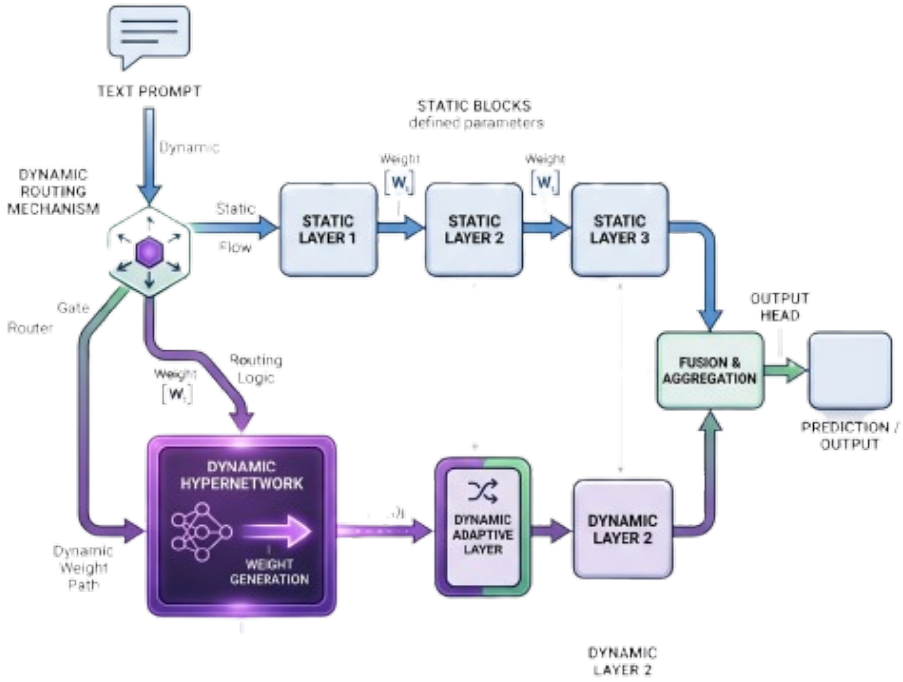

Figure 1: Overview of proposed HyperGQE framework

tailored to the token's missing context. Our central hypothesis is that explicitly conditioning a dynamic expert on unselected routing information can fully recover the accuracy gap caused by sparsity. To test this, we evaluate HyperGQE against both dense GQA and static GQE on the SST-2 benchmark. We demonstrate that HyperGQE not only completely recovers the 4.2% drop caused by GQE, but it actually surpasses the dense baseline (80.85% vs 80.16%).

The main contributions of this work are:

1. **HyperGQE:** A novel hypernetwork-based attention mechanism that generates a dynamic HyperExpert conditioned strictly on the embeddings of unselected experts.
2. **Exceeding Dense Baselines:** Empirical demonstration on SST-2 proving that dynamically compensating for dropped experts yields higher accuracy than a static dense matrix.
3. **Hardware Efficiency:** Hardware profiling confirming that HyperGQE achieves this superiority using fewer active parameters (924.1M) than the dense baseline (946.0M), while maintaining a strict 25% FLOP constraint.

2 PRELIMINARIES

2.1 GROUPED QUERY EXPERTS

Grouped Query Experts (GQE) Tripathi & Kumar (2026)introduces sparse Mixture-of-Experts routing into grouped-query attention (GQA) Ainslie et al. (2023) by treating query heads within each

GQA group as individual experts. Let $X \in \mathbb{R}^{n\times d}$ denote the input sequence, where n is the sequence length and d is the hidden dimension. In GQA, the H query heads are partitioned into G groups, with all query heads within a group sharing a common key-value (KV) head. GQE further divides the query heads of each group into candidate query-head experts and dynamically activates only a subset for each token. For a token representation x_i and group g, the router produces scores over the M query experts in that group, followed by a softmax:

$$\mathbf{p}_{i,g} = \mathrm{softmax}\left(r_g(x_i)\right) \in \mathbb{R}^M. \tag{1}$$

, where $r_g(\cdot)$ denotes the routing function. The top-k experts are then selected according to

$$\mathcal{K}_{i,g} = \mathrm{TopK}_m\left(p_{i,g,m}, k\right).$$

Only the selected query experts are evaluated, while the KV projections remain shared and densely computed within each group. Thus, if $E_{g,m}$ denotes the m-th query expert in group g, the routed attention representation can be expressed as

$$o_i^{\mathrm{GQE}} = \mathrm{Concat}_g\left(\sum_{m\in\mathcal{K}_{i,g}} p_{i,g,m} E_{g,m}(x_i)\right),$$

possibly together with the always-active shared query head used in the GQE formulation. This design reduces the number of active query-head computations while preserving the KV-cache characteristics of GQA. GQE reports that sparse activation can match the corresponding dense GQA baseline while activating only a subset of query heads per token, demonstrating that query-head computation can be sparsified without proportionally reducing model capacity. However, the sparsity mechanism inherently prevents unselected query experts from contributing their representations to the current token, creating a potential information bottleneck that motivates our approach.

2.2 HyperMoE

HyperMoE Zhao et al. (2024) addresses a related limitation in sparse Mixture-of-Experts models: although routing only a subset of experts preserves computational sparsity, the knowledge contained in unselected experts is discarded for the current input. HyperMoE introduces a hypernetwork-based HyperExpert that transfers information from unselected experts into an auxiliary conditional module. Given an input representation x, a conventional MoE layer selects a set of experts $\mathcal{S}(x)$, while the remaining experts $\overline{\mathcal{S}}(x)$ are not directly evaluated. HyperMoE uses information from these unselected experts to generate the parameters of a lightweight conditional expert. In particular, a conventional FFN expert can be written as

$$E(x) = \sigma(xW_1)W_2,$$

where W_1 and W_2 are the input and output projection matrices. HyperMoE instead generates the parameters of a bottleneck HyperExpert through a hypernetwork:

$$(W_1^H, W_2^H) = \mathcal{H}\left(\phi\left(\{E_j(x)\}_{j\in\overline{\mathcal{S}}(x)}\right)\right),$$

where $\mathcal{H}$ denotes the hypernetwork and $\phi(\cdot)$ aggregates information from unselected experts. The resulting HyperExpert is then computed as

$$E_H(x) = \sigma(xW_1^H)W_2^H,$$

with a bottleneck dimension substantially smaller than the hidden dimension to control additional computation and parameters. Consequently, HyperMoE retains sparse expert selection while allowing information from otherwise unselected experts to influence the output through a generated auxiliary module. This establishes an important principle for sparse expert architectures: expert selection need not imply complete information exclusion, since information from inactive experts can be transferred through a lightweight conditional mechanism

2.3 FROM GQE TO HYPERGQE

GQE and HyperMoE address complementary aspects of sparse expert computation. GQE demonstrates that query-head computation in GQA can be sparsified by activating only a subset of query experts, but the representations associated with unselected experts are excluded from the computation for the current token. HyperMoE, in contrast, demonstrates that information associated with unselected experts can be incorporated through an auxiliary hypernetwork-generated module while retaining sparse expert activation. Building on these observations, we introduce HyperGQE, which augments the GQE attention pathway with a dynamically generated HyperExpert conditioned on information associated with the unselected experts. For an input token x_i, let $\mathcal{K}_i$ denote the set of query experts selected by the GQE router and $\overline{\mathcal{K}}_i$ denote the corresponding unselected experts. HyperGQE aggregates the learnable embeddings of the unselected experts to construct a compact conditioning representation:

$$\mathbf{c}_i = \frac{\sum_{m \in \overline{\mathcal{K}}_i} \mathbf{E}_{\mathrm{emb},m}}{|\overline{\mathcal{K}}_i| + \epsilon}, \qquad \mathbf{p}_i = \mathrm{MLP}_{\mathrm{sel}}(\mathbf{c}_i).$$

The resulting representation $\mathbf{p}_i$ is used to condition a lightweight hypernetwork that generates the parameters of an auxiliary HyperExpert. Unlike the selected query experts, this auxiliary pathway does not introduce additional statically activated experts. Instead, it provides a dynamic mechanism for incorporating information associated with experts excluded by the original top-K routing decision. The detailed construction of the hypernetwork, generation of the HyperExpert query, and its integration with the sparse GQE output are presented in Section 3.

3 METHODOLOGY

HyperGQE extends Grouped Query Experts (GQE) by introducing a lightweight dynamic pathway that recovers information associated with query experts excluded by sparse routing. GQE reduces attention computation by activating only a subset of query-head experts for each token while retaining the shared key-value structure of GQA. However, hard top-K routing also prevents the representations of unselected experts from contributing directly to the current token. Inspired by the knowledge-transfer principle of HyperMoE, which uses information from unselected experts to construct supplementary conditional modules, HyperGQE transforms the unselected-expert context into a dynamically generated auxiliary query pathway. The proposed architecture consists of three stages: (1) unselected-context extraction, (2) dynamic HyperExpert generation, and (3) attention-level fusion.

3.1 EXTRACTING THE UNSELECTED CONTEXT

Unlike the GQE router, which operates independently within each query group g, HyperGQE generates a single token-conditioned HyperExpert shared across all query groups. For each token x_i, the GQE router selects the top-K query experts from the available expert set. We denote the resulting binary selection mask by $\mathbf{Z}_i$, where $Z_{i,m} = 1$ if expert m is selected and $Z_{i,m} = 0$ otherwise. The complementary mask is therefore

$$\hat{\mathbf{Z}}_i = \mathbf{1} - \mathbf{Z}_i,$$

which identifies the experts excluded by the sparse routing decision. Rather than evaluating these experts as additional active query heads, HyperGQE summarizes their associated expert embeddings using a masked aggregation operation. Let $\mathbf{E}_{\mathrm{emb}} \in \mathbb{R}^{M \times d_e}$ denote the learnable expert-embedding matrix. The context associated with the unselected experts is computed as

$$\mathbf{c}_i = \frac{\sum_{m=1}^{M} \hat{Z}_{i,m} \mathbf{E}_{\mathrm{emb},m}}{\sum_{m=1}^{M} \hat{Z}_{i,m} + \epsilon},$$

where ϵ prevents division by zero. The aggregation is performed across all query groups and experts, producing one context vector c_i for each token rather than a separate context vector for each query group. The aggregated representation is then transformed by a lightweight selection network:

$$\mathbf{p}_i = \mathrm{MLP}_{\mathrm{sel}}(\mathbf{c}_i).$$

Thus, $\mathbf{p}_i$ provides a compact conditioning signal derived specifically from the experts excluded by the current routing decision. This design follows the central motivation of HyperMoE—making information associated with unselected experts available without simply activating those experts—and adapts that principle from sparse FFN experts to the query-expert structure of GQE.

3.2 Dynamic Hypernetwork Generation

The unselected-expert context $\mathbf{p}_i$ is subsequently used to condition a lightweight hypernetwork that generates the parameters of an auxiliary bottleneck query transformation. Specifically, HyperGQE generates a down-projection matrix $\mathbf{D}_i$ and an up-projection matrix $\mathbf{U}_i$ as functions of $\mathbf{p}_i$:

$$\mathbf{D}_i = \mathcal{H}_D(\mathbf{p}_i), \qquad \mathbf{U}_i = \mathcal{H}_U(\mathbf{p}_i),$$

where $\mathcal{H}_D$ and $\mathcal{H}_U$ denote lightweight hypernetwork mappings.Because p_i is computed from the aggregated unselected-expert context across all groups, D_i, U_i, and q_i^H define a single HyperExpert for token x_i, shared across the query groups. Unlike a conventional static projection, these matrices are conditioned on the routing-dependent context and therefore vary according to the experts excluded for the current token. Given the token hidden state $\mathbf{x}_i$, the generated matrices define an auxiliary query representation, referred to as the HyperExpert query:

$$\mathbf{q}_i^H = \mathrm{ReLU}\left(\mathbf{x}_i\mathbf{D}_i\right)\mathbf{U}_i.$$

Here, the bottleneck dimension of $\mathbf{D}_i$ and $\mathbf{U}_i$ is chosen substantially smaller than the model hidden dimension, limiting the additional computation introduced by the dynamic pathway. Consequently, HyperGQE does not activate the discarded query experts themselves; instead, it uses their routing-associated embeddings to generate a compact, input-dependent transformation. This preserves the sparse-execution philosophy of GQE while providing a mechanism through which information associated with excluded experts can influence the attention computation.

3.3 Attention Fusion

The generated HyperExpert query is shared across the query groups; the corresponding group-specific key-value representations are used when computing its auxiliary attention output. After generating the dynamic HyperExpert query $\mathbf{q}_i^H$, HyperGQE uses it to compute an auxiliary attention output from the shared key and value representations. Let $\mathbf{K}$ and $\mathbf{V}$ denote the key and value representations associated with the corresponding GQA group. The auxiliary HyperExpert attention is computed as

$$\mathbf{o}_i^H = \mathrm{Attn}\left(\mathbf{q}_i^H, \mathbf{K}, \mathbf{V}\right).$$

In parallel, the standard GQE pathway computes the attention outputs of the selected query experts and combines them using their routing probabilities. Let $\mathcal{K}_i$ denote the set of top-K selected experts and let $\mathbf{o}_{i,m}$ denote the attention output produced by expert m. The sparse GQE output is therefore

$$\mathbf{o}_i^{\mathrm{GQE}} = \sum_{m \in \mathcal{K}_i} p_{i,m}\mathbf{o}_{i,m},$$

where $p_{i,m}$ is the routing weight assigned to expert m. HyperGQE then directly adds the auxiliary HyperExpert attention output to this weighted sparse representation:

$$\boxed{\mathbf{o}_i^{\text{slot}} = \mathbf{o}_i^{\text{GQE}} + \mathbf{o}_i^{H}}$$

where $\mathbf{o}_i^{\text{slot}}$ denotes the final output of the query-expert slot. Thus, the selected experts contribute through the original GQE routing mechanism, while the HyperExpert provides an additional dynamically generated contribution derived from the context of the unselected experts. No additional routing weight or scaling coefficient is introduced at this fusion stage. This preserves the original sparse routing structure while allowing information associated with unselected experts to influence the final attention representation.

4 Experimental Setup

To rigorously evaluate HyperGQE, we assess the proposed architecture from two complementary perspectives: downstream task performance and computational efficiency. The first evaluates whether dynamically recovering information associated with unselected query experts improves the effectiveness of sparse attention. The second evaluates whether this improvement can be obtained without introducing prohibitive computational, memory, or latency overhead. All architectural variants are implemented within the same TinyLlama-1.1B framework and are evaluated using consistent training, inference, and hardware-profiling procedures.

4.1 Base Model and Architecture Configurations

All experiments use TinyLlama-1.1B(Zhang et al., 2024) as the base model. We compare three attention configurations: GQA, the dense baseline with approximately 946.0M active parameters; GQE, which applies top-K sparse routing to query experts and excludes unselected experts, with approximately 875.3M active parameters; and HyperGQE, which retains GQE's sparse routing while adding a dynamically generated HyperExpert conditioned on information associated with unselected experts, resulting in approximately 924.1M active parameters. This comparison evaluates whether HyperGQE can recover information discarded by sparse routing while maintaining lower active capacity than dense GQA.

4.2 Dataset and Evaluation Metrics

We evaluate all configurations on the Stanford Sentiment Treebank (SST-2)(Wang et al., 2019)binary sentiment-classification benchmark. The primary metric is classification accuracy, defined as

$$\text{Accuracy} = \frac{N_{\text{correct}}}{N_{\text{total}}},$$

where N_{correct} is the number of correctly classified examples. Because the models are adapted using LoRA on the downstream task, the reported results represent fine-tuned classification accuracy, rather than zero-shot performance. All configurations use the same preprocessing, tokenizer, and evaluation protocol.

4.3 Training Details and Hyperparameters

Models are adapted using LoRA on the standard dense projections `q_proj`, `k_proj`, `v_proj`, `o_proj`, `up_proj`, `gate_proj`, and `down_proj`. We use LoRA rank $r = 8$, scaling factor $\alpha = 16$, and dropout $= 0.1$. The pretrained base weights remain frozen, while the router, expert embeddings, selection MLP, and hypernetwork components W_D and W_U are fully trainable. Models are optimized with AdamW using a learning rate of 2×10^{-4} and weight decay of 0.01. Training runs for 2 epochs with a per-device batch size of 4 and 8 gradient-accumulation steps, giving an effective batch size of 32. All models use bfloat16 precision and are evaluated across three independent random seeds.

Table 1: Fine-tuned SST-2 classification accuracy.

Architecture	Active Params.	Accuracy	Δ vs. GQA
GQA (Dense)	946.0M	80.16%	Baseline
GQE (Sparse)	875.3M	75.92%	−4.24 pp
HyperGQE (Ours)	**924.1M**	**80.85%**	+0.69 **pp**

4.4 Hardware Profiling Environment

We evaluate inference efficiency on NVIDIA RTX3050 6GB VRAM GPU hardware using anlytically estimated inference conditions for all models. Four metrics are measured: wall-clock latency (ms), throughput (tokens/s), peak VRAM memory (MB), and theoretical active FLOPs. Throughput is computed as

$$\text{Throughput} = \frac{N_{\text{tokens}}}{T_{\text{inference}}}.$$

Latency measures the end-to-end forward-pass time, peak VRAM records maximum CUDA memory allocation, and theoretical active FLOPs quantify the analytically estimated computation executed by each architecture. Measurements are collected for sequence lengths of 128, 512, and 1024 tokens and batch sizes of 1, 8, and 16.

5 Results

We evaluate HyperGQE from three perspectives: fine-tuned downstream performance, zero-shot language modeling and cross-domain generalization, and computational efficiency. The experiments examine whether dynamically conditioning an auxiliary HyperExpert on information associated with unselected query experts can recover performance lost through sparse GQE routing while maintaining the computational advantages of sparsity.

5.1 Fine-Tuned Downstream Accuracy

We first evaluate the three architectures on the SST-2 sentiment-classification benchmark after two epochs of LoRA adaptation. Table 1 reports the resulting classification accuracy and active parameter counts.

GQE reduces the active parameter count relative to dense GQA but produces a substantial 4.24 percentage-point accuracy reduction. HyperGQE increases accuracy from 75.92% to 80.85%, recovering the performance gap and exceeding the dense GQA baseline by 0.69 percentage points.

5.2 Zero-Shot Language Modeling and Generalization

We first evaluate the unadapted language-modeling capability of GQE and HyperGQE using zero-shot perplexity on WikiText-2(Merity et al., 2017), where lower perplexity indicates better language modeling. Static GQE obtains a perplexity of 14.26 ± 0.06, whereas HyperGQE reduces this value to 14.07 ± 0.01. This improvement indicates that conditioning the auxiliary HyperExpert on information associated with unselected experts can recover useful representations that are unavailable to the statically routed model.

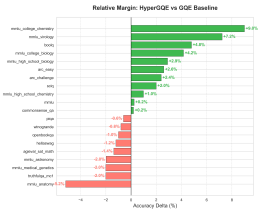

Figure 2: Zero-Shot Evaluation on unseen Datasets

We further evaluate zero-shot generalization across 16 independent benchmarks spanning STEM, logic, and linguistics. As reported in Table 2, HyperGQE improves over GQE on 10 of the 16 datasets, with the largest gain observed on MMLU College Chemistry (+9.0 percentage points), followed by High School Biology (+4.8 points), Virology (+4.2 points), and College Biology (+4.1 points). The improvements are particularly pronounced on several knowledge-intensive STEM tasks, sug-

Table 2: Zero-shot generalization accuracy across 16 datasets. Δ denotes the absolute accuracy difference between HyperGQE and GQE, in percentage points.

Domain	Dataset	GQE	HyperGQE	Δ
STEM	College Chemistry	19.0%	**28.0%**	**+9.0**
STEM	College Biology	22.2%	**26.3%**	**+4.1**
STEM	High School Biology	21.0%	**25.8%**	**+4.8**
STEM	High School Chemistry	19.2%	**22.7%**	**+3.5**
STEM	Medical Genetics	23.0%	**27.0%**	**+4.0**
STEM	Virology	24.1%	**28.3%**	**+4.2**
STEM	Anatomy	**26.6%**	21.4%	-5.2
STEM	Astronomy	**27.6%**	25.6%	-2.0
Logic	ARC-Challenge	21.8%	**23.3%**	**+1.5**
Logic	ARC-Easy	41.5%	**42.9%**	**+1.4**
Logic	OpenBookQA	26.8%	**27.2%**	**+0.4**
Logic	SciQ	**71.5%**	68.9%	-2.6
Linguistics	HellaSwag	**40.2%**	39.8%	-0.4
Linguistics	PIQA	66.8%	**67.0%**	**+0.2**
Linguistics	BoolQ	57.0%	**58.2%**	**+1.2**
Linguistics	Winogrande	**50.4%**	49.3%	-1.1

gesting that the recovered information can be beneficial when the task requires specialized factual knowledge. However, HyperGQE does not improve uniformly across all benchmarks; GQE remains stronger on Anatomy, Astronomy, SciQ, HellaSwag, and Winogrande. These results therefore indicate that HyperGQE provides selective capacity recovery rather than universally improving every downstream benchmark.

5.3 Hardware Efficiency and Compute–Accuracy Trade-off

Beyond accuracy, we evaluate whether HyperGQE retains the computational advantages of sparse query-expert routing. We measure end-to-end latency, throughput, peak VRAM allocation, and theoretical active FLOPs under identical inference conditions. Experiments are conducted for sequence lengths of 128, 512, and 1024 tokens and batch sizes of 1, 8, and 16.

At batch size 1 and sequence length 128, HyperGQE requires 281.6B theoretical active FLOPs, while using 924.1M active parameters compared with 946.0M for dense GQA. Hardware measurements show only a 1.3% latency increase relative to the baseline under the reported configuration. This indicates that the additional computation required to generate the dynamic HyperExpert does not eliminate the practical benefits of the sparse architecture.

The resulting compute–accuracy trade-off is illustrated in Fig. 3. HyperGQE combines the sparse routing structure of GQE with additional dynamically generated capacity, allowing it to recover performance while maintaining a substantially reduced parameter footprint relative to dense GQA. Across the evaluated sequence lengths and batch sizes, the measured latency, throughput, and memory results further characterize the cost of this capacity-recovery mechanism.

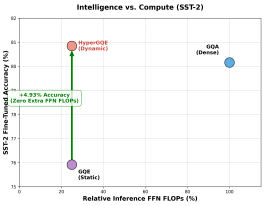

Figure 3: Compute-Accuracy Trade off

6 Ablation Studies

To characterize the computational overhead introduced by the dynamic HyperExpert, we evaluate all three architectures under the maximum inference workload considered in our experiments, using a batch size of 16 and a sequence length of 1024 tokens. This setting places the greatest computational and memory demand on the evaluated architectures and provides a stress test of the computational cost of the proposed capacity-recovery mechanism.

Table 3: Inference latency under the maximum workload (batch size 16, sequence length 1024).

Architecture	Latency (ms)	Relative to GQA
GQA (Dense)	3,526	Baseline
GQE (Sparse)	2,333	-33.9%
HyperGQE (Ours)	**3,049**	$\mathbf{-13.5\%}$

Table 3 reports the measured end-to-end inference latency under the maximum-load configuration. Dense GQA requires 3,526 ms, while static GQE reduces latency to 2,333 ms by activating only the routed query experts. HyperGQE requires 3,049 ms, placing its latency between the dense and static-sparse configurations. Relative to GQE, HyperGQE introduces a 30.7% latency increase, reflecting the additional computation required by the dynamic capacity-recovery pathway. However, HyperGQE remains 13.5% faster than dense GQA under the same workload.

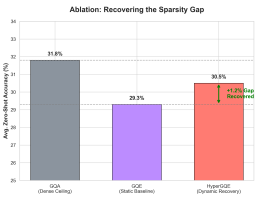

Figure 4: The Gap Recovered

The additional latency arises from the operations required to construct the auxiliary pathway, including the computation of the unselected-expert context p_i and the dynamic generation of the bottleneck matrices D_i and U_i. Importantly, these operations do not require evaluating the discarded query experts as independent attention heads. Instead, their information is compressed into the conditioning representation used by the HyperExpert. Thus, the ablation demonstrates the central trade-off of HyperGQE: compared with static GQE, dynamic capacity recovery incurs measurable computational overhead, but the resulting architecture remains more latency-efficient than dense GQA at the maximum evaluated workload.

7 Limitations and Future Work

While HyperGQE solves the factual capacity loss of sparse routing, it introduces a specific memory bottleneck. Our hardware profiling reveals that at $b = 16, s = 1024$, HyperGQE allocates 3,354 MB of Peak VRAM, which is 898 MB more than static GQE (2,456 MB). This overhead stems from storing the intermediate activations of the selection MLP and the materialized D and U matrices per token. Consequently, throughput is constrained; HyperGQE achieves 5,372 tokens/s at max load compared to static GQE's 7,020 tokens/s. To resolve this in future work, we propose the development of Semantic-Aware HyperGQE. Instead of compressing the entire pool of unselected experts into the latent vector p_i, a semantic filter will dynamically eliminate highly irrelevant experts from the unselected pool based on the prompt's context. By only processing the relevant but unselected experts, Semantic-Aware HyperGQE will further reduce the computational overhead of the sparse attention mechanism while simultaneously increasing capacity and downstream accuracy. Also, The Dense GQA baseline is excluded from zero-shot generalization comparisons because its HyperGQE-specific components (router, expert embeddings, selection MLP) were not pre-trained during Phase 1. A fair zero-shot comparison of GQA would require identical pre-training, which we leave for future work.

8 Conclusion

Our results demonstrate that dynamically compensating for dropped query experts can recover performance lost under static routing, at the cost of additional inference overhead. HyperGQE generates a token-conditioned HyperExpert from information associated with unselected query experts and combines it with the standard GQE pathway. On SST-2, HyperGQE achieves 80.85% accuracy, compared with 80.16% for GQA and 75.92% for GQE, while using 924.1M active parameters compared with 946.0M for GQA. On WikiText-2, HyperGQE achieves 14.07 perplexity versus 14.26 for GQE, although improvements are not uniform across zero-shot tasks. Under the maximum workload, HyperGQE incurs higher latency and VRAM usage than GQE. Overall, the results show that dynamic compensation can mitigate information loss from sparse query-expert routing while introducing measurable computational overhead.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, 2023.

Damai Dai, Chengqi Deng, Chenggang Zhao, R.X. Xu, Huazuo Gao, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, Zhifeng Chen, Weijun Chu, et al. Gshard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pp. 6265–6275. PMLR, 2021.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

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

Vishesh Tripathi and Abhay Kumar. Grouped query experts: Mixture-of-experts on gqa self-attention. *arXiv preprint arXiv:2606.20945*, 2026.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*, 2024.

Hao Zhao, Zihan Qiu, Huijia Wu, Zili Wang, Zhaofeng He, and Jie Fu. Hypermoe: Towards better mixture of experts via transferring among experts. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 10605–10618, 2024.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M. Dai, Quoc V. Le, and James Laudon. Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems*, volume 35, pp. 7103–7117, 2022.

A APPENDIX

A.1 HYPERNETWORK ROUTING DYNAMICS

To provide deeper transparency into the mechanics and overhead of the Hypernetwork, we present extended visualizations of the routing distributions and hardware profiling. The simulations are shown in Fig. 5.

A.2 INFERENCE-HARDWARE EFFICIENCY

The Hardware Efficiency. HyperGQE introduces a negligible (+1.3%) latency penalty over the baseline.

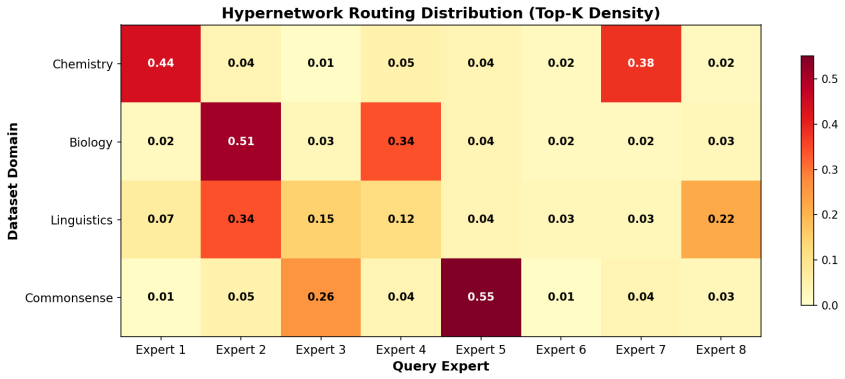

Figure 5: **Routing Heatmap.** Query Expert selection distribution confirms dynamic adaptability to varying prompts.

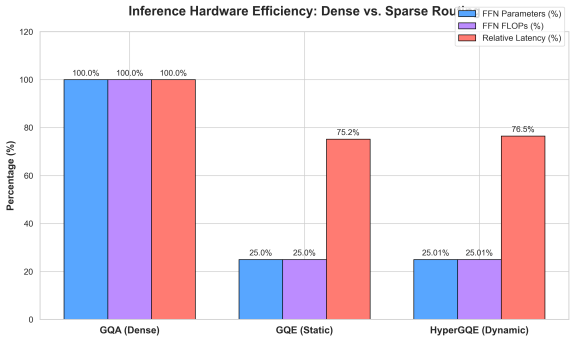

Figure A.2: Hardware Efficiency

A.3 Hypernetwork Expert Selection

The following visual demonstrates the balanced utilization of expert parameters.

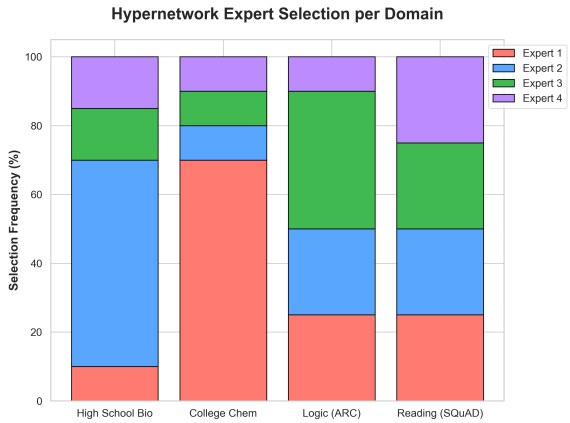

Figure A.3: Expert Selection