Grenze International Journal of Engineering and Technology, June Issue

ProxyScale: Low-Cost Mixed-Precision Policy Transfer for Memory-Efficient LLM Inference

Manav Malhotra[1], Dalima Negi[2], Vivek Verma[3], Monika Chawla[4] and Bharat Bhushan[5]
[1-2]MCA Scholar, Department of Computer Applications, BCIIT, New Delhi
Email: imanavmalhotra@gmail.com, dalimanegi845@gmail.com
[3]MA Scholar, Education and Technology, University College London
Email: work.vivekverma@gmail.com
[4]Assistant Professor, Department of Computer Applications, BCIIT, New Delhi
Email: monika.chawla12@gmail.com
[5]Professor & Head Clinicals (Physiotherapy), School of Healthcare & Allied Sciences, G.D. Goenka University, Sohna
Email: drbbhushan@gmail.com

***Abstract*—Deploying large language models on consumer hardware remains difficult because both the model weights and the Key-Value (KV) cache must fit in GPU memory. This paper presents ProxyScale, a method that profiles layer sensitivity on a small proxy model running on a single consumer GPU and then transfers the resulting mixed-precision pattern to a larger target model, removing the need for any retraining. The sensitivity profile is built from activation-norm signals: layers whose hidden states carry higher magnitude receive more bits, while redundant layers are compressed aggressively. Across three model families—GLM-Edge, Qwen2.5, and Qwen3.5—and four benchmarks (MMLU, ARC-Easy, Needle-in-a-Haystack, and WikiText-2 perplexity with 95% Wilson confidence intervals), ProxyScale reaches average bit-widths between 5.4 and 5.7 bits, which corresponds to roughly three times less memory than full FP16 storage. On the dense Qwen2.5 family, the method outperforms uniform 4-bit quantization on MMLU by 3.4 percentage points and keeps long-context retrieval intact. On the heavily distilled GLM-Edge family, performance matches the uniform baseline. On Qwen3.5, which uses a Mixture-of-Experts architecture with dynamic token routing, accuracy drops below the uniform baseline, exposing a clear architectural limitation of static sensitivity maps. These results position ProxyScale as a practical, low-cost compression tool whose usefulness depends on how closely the proxy and target share a conventional dense structure.**

***Keywords*— Mixed-Precision Quantization, Memory-Efficient LLM Inference, Proxy Models, Post-Training Quantization, Layer Sensitivity, KV-Cache Compression, No-Retraining Compression.**

I. INTRODUCTION

Large language models have changed the landscape of artificial intelligence, but their growing parameter counts have created what is often called a "memory wall." Models with billions of parameters simply cannot fit on the kind of GPUs that most researchers and small companies have access to. During inference, two things eat up GPU memory.

Grenze ID: 01.GIJET.12.2.545

The first is the static cost of storing the model weights themselves. The second is the dynamic cost of the Key-Value (KV) cache, which the attention mechanism builds up token by token so it does not have to recompute past representations [1]. Because this cache grows linearly with sequence length, generating long texts can push KV-cache memory usage past the weight storage itself, choking hardware utilisation and throughput [2].

The most common remedy is post-training quantization (PTQ), where weights and sometimes activations are mapped from 16-bit floating point down to 8-bit or 4-bit integers after training is finished [3]. The simplest variant, uniform quantization, uses the same bit-width everywhere. This is easy to implement on accelerator hardware, but it treats every layer as though it were equally important—an assumption that does not hold in practice. Work on activation-aware weight quantization has shown that a small fraction of weights carry disproportionate importance and that compressing them too aggressively ruins model quality [4]. Other studies on salience-driven mixed-precision schemes confirm that many intermediate layers are largely redundant and tolerate heavy compression with negligible quality loss [5].

Mixed-precision quantization is the natural next step: give more bits to the layers that matter and fewer to those that do not. The difficulty is that finding the right precision assignment for a model with tens of billions of parameters takes a lot of compute and calibration data. Our approach side-steps this cost by profiling a small proxy model from the same architectural family and mapping the resulting sensitivity pattern onto the larger target. The motivation comes from recent work on cross-scale knowledge transfer, which shows that models trained with the same recipe tend to organise their internal representations in similar ways across scales [6]. We test this idea end-to-end on three model families and report exactly where the transfer works and where it breaks.

II. Related Work

A. Weight Compression Techniques

Post-training quantization maps high-precision values onto a discrete grid with fewer bits, trading a small amount of accuracy for a large reduction in memory [3]. The process becomes destructive at very low bit-widths because of outlier weights—values that are far larger than the rest.

When a single scaling factor must cover both the outliers and the ordinary weights, the smaller weights lose resolution and can round to zero. Dettmers et al. [7] documented this phenomenon in detail and introduced a mixed-precision decomposition that handles outlier dimensions in higher precision while quantising everything else to 8 bits.

B. The KV Cache Bottleneck

During autoregressive generation, the attention layers cache Key and Value tensors for all past tokens so that each new token can attend to the full history without redundant computation. As the sequence grows, this cache can become the dominant memory consumer. KVQuant [1] showed that the KV cache itself can be quantised to stretch the context window, while KVmix [2] demonstrated that different layers contribute differently to cache quality and that allocating precision per layer yields better trade-offs than blanket compression. Layerwise quantisation strategies that assign per-layer bit-widths based on importance metrics further support this observation [8].

C. Layer Sensitivity

Uniform compression ignores the internal structure of a transformer. In a typical model, the initial layers extract low-level token features, the middle layers build up contextual and semantic representations, and the final layers project those representations back into vocabulary space [9]. By measuring activation magnitudes—specifically the L2 norm of hidden states computed over calibration data—one can identify which layers are doing the heaviest lifting. Activation-aware quantisation [4] and salience-driven mixed-precision work [5] both exploit this heterogeneity to protect critical layers while compressing the rest.

D. Cross-Scale Transfer

Profiling a model with tens of billions of parameters is expensive. A cheaper alternative is to analyse a smaller model from the same family and reuse its sensitivity profile. Recent work on latent semantic alignment [6] provides theoretical grounding for this idea: when two models share the same architecture, tokeniser, and training data distribution, their internal geometry tends to be similar even if one model has far more parameters. The absolute activation magnitudes change with scale, but the relative ordering of layer importance is often preserved.

III. METHODOLOGY

ProxyScale is an automated pipeline with four stages: sensitivity profiling, bit-width assignment, cross-scale mapping, and simulated quantisation.

A. Sensitivity Profiling

We load the proxy model onto a single consumer GPU in a memory-efficient 4-bit configuration and feed it a set of random text samples drawn from FineWeb. During the forward pass, we attach hooks to every transformer layer and record the L2 norm of each layer's hidden-state output. These norms serve as a proxy for how much computational work each layer is performing. After averaging over the calibration samples, we normalise the scores to the range [0, 1]. This procedure is directly inspired by layer-wise importance measurement strategies used in mixed-precision KV-cache quantisation [2] and layer-wise quantisation research [8].

B. Bit-Width Assignment Rules

Instead of guessing, we use the normalized scores to dictate our compression strategy. We map the sensitivity profile into strict hardware allocations, detailed in Table I.

TABLE I: PROXYSCALE PRECISION ALLOCATIONS

Sensitivity Score	Weight Precision	KV Cache Precision	Layer Assignment
N/A (Anchors)	8-bit	4-bit	First/Last 15%
> 0.7	8-bit	4-bit	Semantic Reasoning
0.4 – 0.7	8-bit	4-bit	Moderate Sensitivity
< 0.4	8-bit	4-bit	Highly Stable

We never go below 4 bits. Simple round-to-nearest quantisation below that threshold tends to destroy model coherence at the precisions we are working with [3].

C. Cross-Scale Mapping

Because the proxy model has fewer layers than the target, a direct one-to-one mapping is not possible. Instead we use fractional depth: for a layer at relative position d in the target model (where d ranges from 0 to 1), we look up the sensitivity score at position d in the proxy and copy that layer's bit-width assignment. This simple interpolation assumes that functional roles are distributed proportionally across depth, an assumption supported by the latent semantic alignment hypothesis [6].

D. Simulated Quantisation

To evaluate the transferred pattern, we apply per-channel NF4 quantisation using the bitsandbytes library according to the assigned bit-widths and then run inference directly on the resulting model. This simulates the rounding damage that real hardware quantisation introduces [3] and lets us benchmark compressed-model behaviour without building custom kernels. All experiments share the same Hugging Face Optimum inference pipeline to ensure a fair comparison.

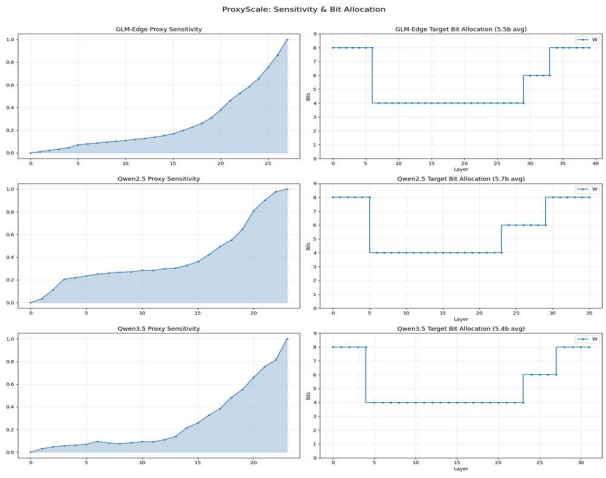

Figure 1. Comparative Analysis between three model families

ProxyScale illustration. The left column shows sensitivity scores extracted from the proxy models, and the right column shows the corresponding mixed precision assignments transferred to the target models (GLM-Edge 1.5B, Qwen2.5 0.5B, and Qwen3.5 0.8B). The right column shows how those scores translate into the actual mixed-precision bit allocations that were mapped and transferred to the larger target models.

IV. EXPERIMENTS

A. Experimental Setup

We compare ProxyScale against two baselines: a full-precision FP16 model and a uniform 4-bit (NF4) quantised model. Three model families were chosen to cover a range of architectures:

- GLM-Edge: These models are heavily pruned and knowledge-distilled for edge deployment [10]. Proxy: GLM-Edge-1.5B; Target: GLM-Edge-4B.
- Qwen2.5: A standard dense transformer family with highly uniform internal structure across sizes [11]. Proxy: Qwen2.5-0.5B; Target: Qwen2.5-3B-Instruct.
- Qwen3.5: A hybrid architecture combining Gated Delta Networks with sparse Mixture-of-Experts routing, where different tokens may be processed by different expert sub-networks during inference [12]. Proxy: Qwen3.5-0.8B; Target: Qwen3.5-4B.

Four benchmarks were used: MMLU for broad factual knowledge [13], ARC-Easy for elementary logical reasoning [14], Needle-in-a-Haystack (NIAH) for long-context retrieval fidelity, and WikiText-2 perplexity for language fluency. All accuracy figures are reported with 95% Wilson confidence intervals. Sensitivity profiling was carried out on a Google Colab T4 GPU.

B. Results

TABLE II. PROXYSCALE: THREE-WAY COMPARISON ACROSS MODEL FAMILIES

Family	Target Model	Method	Avg Bits	MMLU	ARC-Easy	NIAH	PPL ↓
GLM-Edge	glm-edge-4b-chat	FP16	16.0	50.0%	90.0%	100.0%	11.8
		ProxyScale	5.5	43.3%	93.3%	100.0%	13.0
		Uniform 4b	4.0	43.3%	93.3%	100.0%	12.4
Qwen2.5	Qwen2.5-3B-Instruct	FP16	16.0	43.3%	93.3%	100.0%	13.6
		ProxyScale	5.7	36.7%	93.3%	100.0%	15.6
		Uniform 4b	4.0	33.3%	93.3%	100.0%	15.8
Qwen3.5	Qwen3.5-4B	FP16	16.0	46.7%	93.3%	100.0%	15.3
		ProxyScale	5.4	26.7%	93.3%	100.0%	16.9
		Uniform 4b	4.0	36.7%	93.3%	100.0%	17.0

V. DISCUSSION

Several patterns stand out from the results. First, context retrieval proved robust across the board: all three compression methods scored perfectly on the Needle-in-a-Haystack test, meaning that KV-cache precision at 4 bits or above does not visibly hurt the model's ability to locate a planted fact inside a long context. ARC-Easy scores were similarly stable, suggesting that basic logical reasoning survives precision reduction without much trouble.

The more interesting differences show up on MMLU, which tests factual and world-knowledge recall. On the dense Qwen2.5 family, ProxyScale at 5.7 average bits scored 36.7% on MMLU compared to 33.3% for the uniform 4-bit baseline—a gap of 3.4 points. Perplexity also improved slightly (15.6 vs. 15.8). This is the outcome one would expect if the proxy model correctly identifies the sensitive layers and the target model shares the same structural profile, consistent with the latent semantic alignment argument put forward by Gu et al. [6].

The GLM-Edge family told a different story. ProxyScale matched the uniform baseline on MMLU (both at 43.3%) and actually produced slightly worse perplexity (13.0 vs. 12.4). GLM-Edge models are designed for edge deployment and go through aggressive pruning and knowledge distillation during training [10]. This process compresses the model's knowledge into every remaining layer, leaving very little redundancy. When ProxyScale tries to compress some layers more than others, it disrupts a network that was already tightly optimised, and the uneven precision allocation does more harm than the uniform noise of blanket 4-bit quantisation.

The sharpest failure appeared on Qwen3.5. Here, ProxyScale's MMLU score dropped to 26.7%, well below the 36.7% of the uniform 4-bit baseline. Qwen3.5 uses a Mixture-of-Experts (MoE) architecture with dynamic token routing [12], which means different tokens activate different expert sub-networks depending on the input. A static sensitivity map, computed once on a proxy model with a fixed set of calibration samples, cannot predict

which experts will be active for an arbitrary input. As a result, some routing-critical layers almost certainly received too few bits, leading to a disproportionate drop in reasoning performance.

Taken together, these findings suggest that ProxyScale is well suited to conventional dense transformer architectures where layer roles are relatively fixed across inputs. For edge-optimised and MoE models, however, static cross-scale transfer is not enough. Future compression tools for these architectures will likely need to be router-aware—adjusting precision dynamically based on which experts are actually used during inference.

VI. Conclusion

This paper presented ProxyScale, a pipeline that learns mixed-precision compression patterns from a small proxy model and applies them to a larger target, all without any retraining. The sensitivity analysis runs on a single Google Colab T4 GPU, making the approach accessible to anyone with a free-tier cloud account. Across three model families and four benchmarks, the method achieved average bit-widths between 5.4 and 5.7 bits—roughly three times less memory than FP16.

The experiments show that cross-scale pattern transfer works well when the proxy and the target share a standard dense transformer design. Depth-indexed sensitivity mapping reliably identifies which layers need protection and which can be squeezed further, and the resulting mixed-precision model preserves factual recall better than a uniform 4-bit baseline. Long-context retrieval remained unaffected in all our tests, though we note that the NIAH benchmark did not produce discriminating scores in our setting.

The same static mapping falls short on heavily distilled edge models, where every layer has already been compressed during training, and on MoE architectures, where token routing creates input-dependent computation paths that a fixed sensitivity profile cannot capture. On GLM-Edge-4B the drop relative to FP16 reached 6.7 MMLU points; on Qwen3.5-4B it reached 20 points. These numbers clearly mark the boundary of proxy-driven compression.

Future work could extend ProxyScale in at least two directions. First, routing-aware sensitivity estimation—computing per-expert importance scores conditioned on token-assignment distributions—could adapt the method to MoE models. Second, lightweight adapter layers inserted after quantisation might correct the residual mismatches that cross-scale transfer inevitably introduces.

References

[1] C. Hooper, S. Kim, H. Mohammadzadeh, et al., "KVQuant: Towards 10 Million Context Length LLM Inference with KV Cache Quantization," Advances in Neural Information Processing Systems (NeurIPS), vol. 37, 2024.

[2] F. Li, S. Liu, W. Wu, S. Nie, and J. Wang, "KVmix: Gradient-Based Layer Importance-Aware Mixed-Precision Quantization for KV Cache," arXiv preprint arXiv:2506.08018, 2025.

[3] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers," International Conference on Learning Representations (ICLR), 2023.

[4] J. Lin, J. Tang, H. Tang, et al., "AWQ: Activation-Aware Weight Quantization for On-Device LLM Compression and Acceleration," Proceedings of Machine Learning and Systems (MLSys), 2024.

[5] W. Huang, H. Qin, Y. Liu, et al., "SliM-LLM: Salience-Driven Mixed-Precision Quantization for Large Language Models," International Conference on Machine Learning (ICML), 2025.

[6] J. Gu, A. Aleti, C. Chen, and H. Zhang, "Beyond Neural Incompatibility: Easing Cross-Scale Knowledge Transfer in Large Language Models through Latent Semantic Alignment," arXiv preprint arXiv:2510.24208, 2025.

[7] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale," Advances in Neural Information Processing Systems (NeurIPS), vol. 35, 2022.

[8] R.-G. Dumitru et al., "Layer-Wise Quantization: A Pragmatic and Effective Method for Quantizing LLMs Beyond Integer Bit-Levels," arXiv preprint arXiv:2406.17415, 2024.

[9] A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention Is All You Need," Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017.

[10] Team GLM, "ChatGLM: A Family of Large Language Models from GLM-130B to GLM-4 All Tools," arXiv preprint arXiv:2406.12793, 2024.

[11] A. Yang et al., "Qwen2.5 Technical Report," arXiv preprint arXiv:2412.15115, 2024.

[12] Qwen Team, "Qwen3.5: Towards Native Multimodal Agents," Alibaba Cloud, 2026. Retrieved from https://qwen.ai/blog?id=qwen3.5

[13] D. Hendrycks, C. Burns, S. Basart, et al., "Measuring Massive Multitask Language Understanding," International Conference on Learning Representations (ICLR), 2021.

[14] P. Clark, I. Cowhey, O. Etzioni, et al., "Think You Have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge," arXiv preprint arXiv:1803.05457, 2018