RUNTIME DYNAMIC COMPRESSION OF MIXTURE OF EXPERTS

Tim Dettmers
Carnegie Mellon University
dettmers@cmu.edu

ABSTRACT

Open-weight language models are becoming more capable, but their growing size makes them difficult to host on consumer hardware. Dynamic quantization reduces memory use, but alone does not reach the extreme compression levels of between 1 to 2 bits required to fit the largest models on these devices at usable quality. We introduce runtime dynamic compression, which achieves compression levels of about 0.4 to 1 bit per weight below Unsloth Dynamic Quantization while maintaining comparable WikiText perplexity. Our method jointly allocates quantization levels and expert removal across layers and adapts which experts remain resident during inference. We achieve this with two innovations: (1) iterative sensitivity probing, which repeatedly measures sensitivities and reallocates memory as the model approaches the edge of instability, accounting for the super-additive effects between compressed layers; and (2) dynamic REAP, which exchanges experts asynchronously as the workload changes, without increasing the resident memory budget or making tokens wait for transfers. We evaluate mixture-of-experts models from 35B to 550B parameters and show the combination of expert removal, quantization, and dynamic REAP yields high-quality compressed models in the 1 to 2 bit range. The resulting resident weights fit within a 24 GB footprint for a 125B-class model and a 96 GB footprint for a 550B-class model. We release the bitsandbytes2 framework and our compressed models.

1 INTRODUCTION

As the gap between open-weight and closed-source AI model capability shrinks, the overall size of open-weight models increases. At the same time, DRAM prices increase, which makes it even more pricey and difficult to host open-weight models on one's own hardware devices. This trend necessitates extreme compression to a level of around 2 bits per weight or lower while maintaining model quality to make frontier-capability open-weight models available for consumers and small companies. While dynamic quantization, such as Unsloth GGUF, has been the most successful technique for compression, dynamic quantization alone does not reach the extreme compression levels required while maintaining the quality of the compressed model.

In this work, we introduce runtime dynamic compression, where we extend the problem of dynamic quantization to expert removal and use a runtime dynamic expert swapping algorithm to achieve compression levels 0.4 to 1 bit per weight below Unsloth quantization while maintaining the same quality.

Dynamic quantization, also known as mixed-precision quantization, is the problem where we assign different bit compression levels to different layers in an AI model. The idea is simple: certain layers are more sensitive to compression than others; if we can locate which layers are more sensitive, we can allocate more bits to the sensitive layers and remove bits from the insensitive ones, thus achieving higher quality compression without any additional bits. If the right layers are found, this scheme is very effective and leads to strong compression levels while providing stable and usable inference behavior.

The main challenge for dynamic quantization is the following: compressing the current layer not only degrades that layer, but also all future layers that depend on the current layer. Since compression has such larger-than-sum super-additive effects, the problem of finding the right allocation of bits to

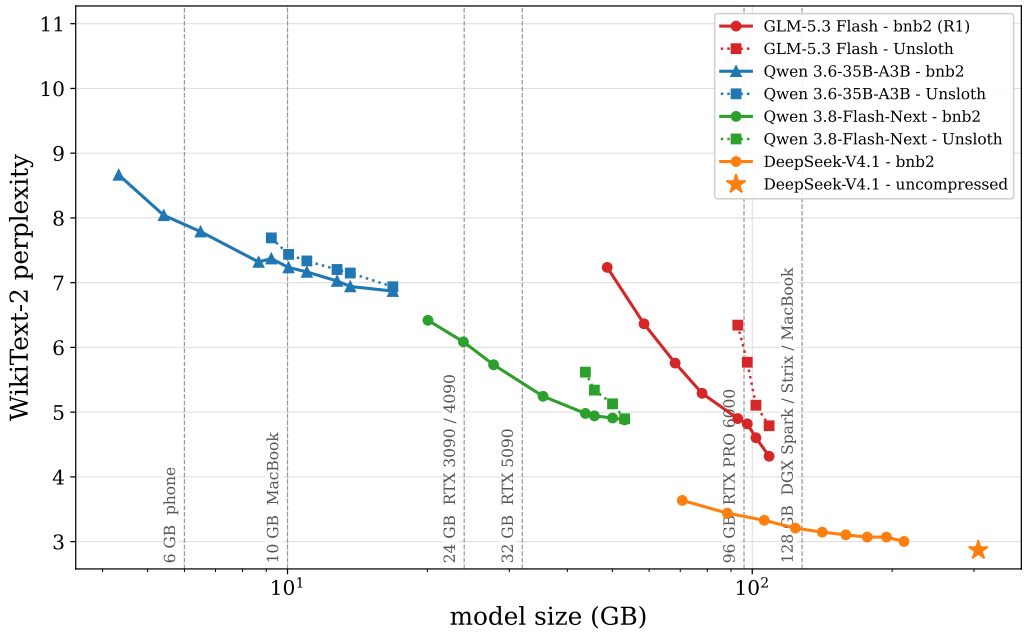

Figure 1: Perplexity-size trade-offs of different models under compression comparing our bnb2 runtime dynamic compression with Unsloth dynamic quantization. We see that due to stability at low-bit footprints, bnb2 enables the deployment of large models on small devices while maintaining adequate quality.

all layers in the AI model is a combinatorial problem where it is infeasible to evaluate all possible combinations of bit-and-layer combinations. As such, the core research problem is to devise an efficient scheme where we estimate the sensitivity of each bit-layer combination while maintaining a trade-off of computational overhead vs sensitivity estimation accuracy.

The most popular and successful technique for dynamic quantization has been the Unsloth Dynamic Quantization (Han et al., 2023), but it uses an unknown proprietary sensitivity estimation algorithm. Published approaches that reach Unsloth's quality-compression trade-offs for the popular Qwen 3.8 27B model are GSQ (Dadgarnia et al., 2026) and RCO (Helcig & Alistarh, 2026). Both GSQ and RCO learn their compression decisions by gradient descent — GSQ the quantization grid on each layer's weights, RCO the assignment of precisions under an exact budget — rather than estimating a sensitivity per layer.

While dynamic quantization is effective, it does not provide enough compression to keep up with the ever-growing size of open-weight models where even "flash" models reach a size of 125B to 550B parameters. As such, an additional axis of compression becomes important: expert removal.

Existing approaches for expert removal either remove experts permanently before deployment, such as REAP (Lasby et al., 2025), or keep experts on NVMe disk or CPU memory to transfer them just-in-time when they are needed (Yang et al., 2026; Tang et al., 2024). Permanent removal of experts yields a fast inference solution; just-in-time transfers maintains quality but reduces inference-speed significantly and usually requires a large amount of CPU RAM – something that consumers usually do not have.

Here, we introduce runtime dynamic compression of mixture of experts (MoE), which can compress MoE models up to 550B parameters to the 1-2 bit range without major instabilities, overall being about 0.4 to 1 bit per weight more efficient compared to Unsloth Dynamic Quantization. Our runtime dynamic compression approach achieves accurate but cheap sensitivity measurements and strong compression through two innovations: (1) dynamic REAP, which takes REAP's permanent one-shot removal (Lasby et al., 2025) and extends it so asynchronous expert retrieval at runtime where experts are instantaneously swapped at decode token boundaries; (2) iterative sensitivity probing,

which iteratively drives the model closer and closer to the edge of instability where layers naturally reveal super-additive effects.

Runtime dynamic compression enables a level of accessibility of large models that was not possible before. In Figure 1 we can see the quality-size trade-off using our method and see that large models can now be run on smaller and smaller hardware without major performance degradation. For example, 125B models such as Qwen 3.8 Flash Next can be run within a 24GB footprint and models around up to 550B parameters, such as DeepSeek v4.1 can be run within a 96GB memory footprint.

Runtime dynamic compression will pave the way for a new level of accessibility for open-weight models that will help individuals and companies to choose open-source over closed-source models. We are open-sourcing our altorithms and implementations in the bitsandbytes2 library.

2 BACKGROUND

Quantization for AI models involves the compression of weights or input+weights from 16-bit to a lower-precision to save memory and improve runtime processing speed.

The main difficulty of quantizing AI models is that outliers in the model can lead to unstable quantization (Dettmers et al., 2022). Many techniques to handle outliers have been developed: isolating outliers into individual blocks (Dettmers et al., 2021); absorbing outliers into weights (Xiao et al., 2023); shuffling outliers across dimensions (Chee et al., 2023; Ashkboos et al., 2024; Malinovskii et al., 2024)

One can distinguish two main families of quantization: scalar from **vector quantization**. Scalar quantization uses bits to encode a single number which usually offers fast runtime performance, for example, NVFP4 (Abecassis et al., 2025) encodes a single number as a floating point normalized into blocks which can be accelerated with hardware support to yield speedups of 4x on modern NVIDIA GPUs. Vector quantization represent multiple values with a single encoding. For example, a 4-bit value might encode 6 different output values making it effectively a 4/6 = 0.66 bits-per-weight quantization. In our work we use HIGGS (Malinovskii et al., 2024) which combines Hadamard rotations, which shuffle outliers across random dimensions, with vector quantization.

Vector quantization is also called **codebook quantization**, since a 4-bit that maps 6 different values, is usually represented as a codebook lookup table with 2^4 keys mapping to 6-value vectors. Large codebooks quickly become too slow at runtime to be deployable. One of the most competitive quantization techniques is QTIP (Tseng et al., 2024b), which encodes its codebook through a state machine allowing impractically large codebooks to be encoded as a small state machine which makes the quantization effective and fast at runtime. The drawback is that encoding weight matrices into state machines can take days of GPU time.

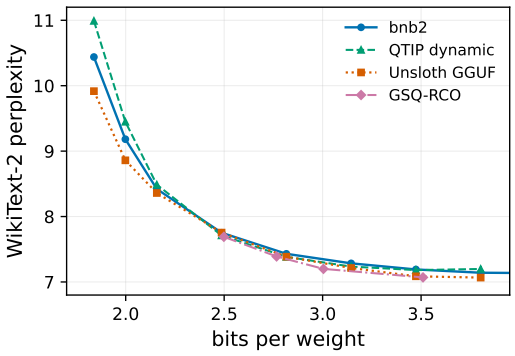

Figure 2: Dynamic quantization of a dense Qwen 3.8 27B model. We see two things: (1) dense quantization is largely solved by dynamic quantization; (2) while QTIP is a powerful technique it falls short for dynamic quantization since it is too expensive to encode a diverse codec ladder.

While QTIP is a strong quantization technique, it can often lag behind other techniques when it comes to dynamic quantization. For dynamic quantization to be effective, the granularity of quantization bit-precisions – the **codec ladder** – are important to fit layers well. For example, with codecs of 1/2/3/4/5 bits, it is difficult to represent layers that have jumps in sensitivity between 1 and 2 bits. This issue is seen in Figure 2 where QTIP underperforms. As such, even powerful techniques like QTIP can become unstable and poor quality. While fractional codecs for QTIP are possible – since it is based on codebook quantization – the expensive encoding procedure makes it computational infeasible to study large-model dynamic compression with QTIP quantization.

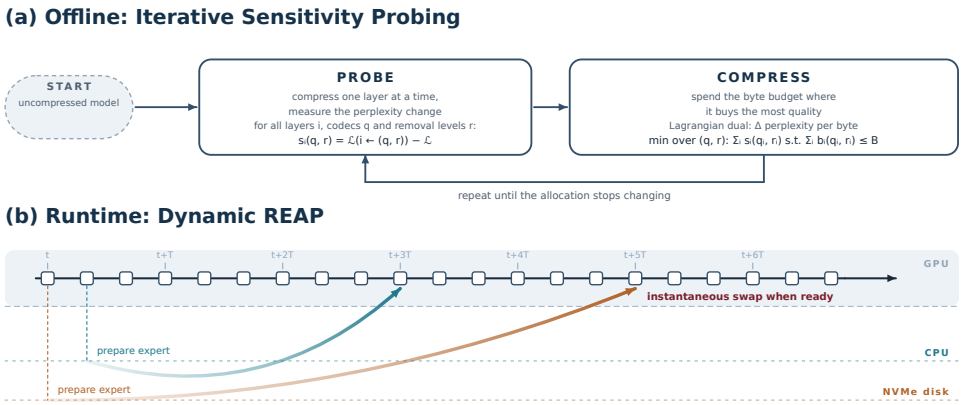

Figure 3: Schematic of our runtime dynamic compression method. (a) Offline, before running the model, we do iterative sensitivity probing: starting from the uncompressed model, then probe the sensitivities of each layer by compressing them individually and evaluating perplexity against a baseline. We then use the sensitivities to re-allocate a new iteration (or the deployment) with the Lagrangian dual that optimizes perplexity differences per byte. (b) Runtime, dynamic REAP: experts are prepared asynchronously from CPU memory or NVMe disk at a fixed cadence T and swapped in instantaneously when the preparation completes, so the resident set follows the workload while capacity never changes. With incurs practically no runtime speed overhead.

For this very reason, we use HIGGS quantization(Malinovskii et al., 2024), a simple codebook quantization technique that allows for fractional codecs while using a simple shuffling technique, Hadamard rotations, to deal with outliers. We use the following codec ladder in our work: $\{0.7, 1, 1.125, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5, 2.75, 3, 4, 5\}$ bits-per-weight. This configuration has other advantages, such that analytic distortions at known a priori (Malinovskii et al., 2024). We refer to this quantization setup as bitsandbytes2, or **bnb2** for short, which is named after the open-source framework that implements this quantization setup.

The **scaling laws of precision** (Kumar et al., 2025) state an intuitive but critical relationship: the difficulty of compressing AI models is directly related to how much training data per parameter was used. For example, 27B model trained on 20T tokens is much more difficult to compress than a 70B model trained on 20T tokens or a 7B model training on 5T tokens. Two important point immediately follows: quantization techniques should always be evaluated on the most recent high-quality models that are trained with the most tokens. Mixture of expert models are more compressible from a bits-per-weights perspective.

Finally, **mixed-precision and dynamic quantization** give different layers different precisions and/or data types, so that a target average rate is reached by spending bits where they buy the most quality. Mixed-precision quantization uses a single data type at different precisions; dynamic quantization uses specialized data types for different bit precisions. Its mixture-of-experts experiments also allocate a precision per expert, and in both settings the non-uniform allocation is what holds quality at high sparsity, where uniform removal loses most of it.

3 METHOD

There are two parts to our method: (1) probing the sensitivity of layers via iterative sensitivity probing, and (2) deployment with dynamic REAP. These components are shown in Figure 3.

3.1 ITERATIVE SENSITIVITY PROBING (ISP)

Iterative sensitivity probing (ISP) is a heuristic to solve the combinatorial problem of estimating how sensitive each layer is to quantization or expert removal while other layers are also compressed

in a particular configuration. Solving this combinatorial estimation problem is computationally infeasible which necessitates approximate algorithms.

The main problem for building a heuristic is this: a layer that is insensitive to compression when surrounding layers are not compressed, might be highly sensitive if surrounding layers are compressed. As such, often degradations remain additive under weak compression but become super-additive under strong compression. Super-additive meaning that the degradation of the sum is larger than its parts.

ISP approaches the problem from the following idea: if we drive the model to the edge of instability – the point where no catastrophic degradation has occurred yet, but any further compression leads to super-additive collapse – then a sensitivity measurement surrounding a single layer faithfully represents the sensitivity of that layer and how it relates to other layers.

While this is intuitive, this leaves us with a chicken and egg problem: To have a good dynamic compression we need to estimate sensitivities of a model close the edge of instability, but to get close to the edge of instability, we need to good dynamic compression.

We can resolve this problem as follows. First, we estimate sensitivities without any compression. Most measurements will be unreliable, since they cannot measure the supper-additive effects under mild compression. However, we are able to measure effects of highly unstable layers. We then proceed to compress layers that are stable while keeping the unstable layers intact and we re-estimate sensitivities. This reveals another set of unstable layers. Now we can repeat this procedure until we drive the model to the edge of instability. This is the procedure of ISP.

More formally, we can express ISP as follows.

Measuring sensitivity under the current compression. At iteration t, the model has compression configuration $\mathbf{c}^{(t)}$ which is made up of layer specific compression setting $\mathbf{c}[i] = (q, r)$, where q is the quantization level and r expert-removal fraction. We measure sensitivity by changing one layer while keeping every other layer at its current setting. We then evaluate perplexity in this session on a calibration set and compare differences in perplexity compared to a baseline compressed model.

Writing $\mathcal{L}(\mathbf{c})$ for the model's log-perplexity, measured on the same evaluation data, the sensitivity is:

$$s_i^{(t)}(q, r) = \mathcal{L}\big(\mathbf{c}^{(t)}[i \leftarrow (q, r)]\big) - \mathcal{L}\big(\mathbf{c}^{(t)}\big). \tag{1}$$

The expression $[i \leftarrow (q, r)]$ means "replace only layer i's compression setting." A positive value means that this replacement increases loss; a negative value means that it improves the model relative to the current allocation.

Crucially, this measures the effect on the **whole model**, not just the output of layer i. The measurement therefore includes interactions with the surrounding layers in their current compressed state. When the overall model allocation settings change, the measured sensitivity can change too.

Allocating memory from sensitivity measurements. The allocator uses these sensitivities to choose a new setting for every layer. If $b_i(q, r)$ is the number of bytes required by a layer at that setting, and B is the available memory, the allocation problem is:

$$\begin{aligned} \min_{\{q_i, r_i\}} \quad & \sum_i s_i^{(t)}(q_i, r_i) \\ \text{subject to} \quad & \sum_i b_i(q_i, r_i) \le B. \end{aligned} \tag{2}$$

Here, the sum of sensitivities is an **approximation to the change in log-perplexity** when multiple layers change together. The approximation is built from measurements under the current compression, and ISP updates the compression setting after reallocation.

We introduce a multiplier $\lambda \ge 0$ to express the memory constraint through the Lagrangian:

$$J_t(\mathbf{c}, \lambda) = \sum_i \big[s_i^{(t)}(q_i, r_i) + \lambda b_i(q_i, r_i)\big] - \lambda B. \tag{3}$$

The multiplier λ gives a common cost to each byte, measured in units of log-perplexity. Spending additional bytes is worthwhile when the estimated reduction in loss exceeds that cost.

For a fixed λ, each layer can choose its setting independently:

$$c_i^{(t+1)}(\lambda) = \arg\min_{(q,r)} \big[s_i^{(t)}(q,r) + \lambda b_i(q,r) \big]. \tag{4}$$

The dual optimizer adjusts λ: increasing it favors smaller configurations; decreasing it favors configurations with lower estimated loss. Formally, it solves:

$$\max_{\lambda \geq 0} \min_{\mathbf{c}} J_t(\mathbf{c}, \lambda). \tag{5}$$

With discrete compression settings, memory use changes in steps, so the resulting feasible allocation need not consume every available byte.

At this point, ISP increments t and starts a new iteration. ISP can be terminated once the differences in sensitivities are below a certain value $|S^{(t)} - S^{(t+1)}| \leq \epsilon$ where ϵ is an empirically chosen threshold or can be chosen when the discrete allocation of quantization and expert removal assignments no longer changes.

3.2 Dynamic REAP: runtime expert exchange

REAP (Lasby et al., 2025) permanently removes experts with the lowest routing scores. For expert e in layer i, the score is

$$S_{i,e}^{\text{REAP}} = \frac{1}{|\mathcal{X}_{i,e}|} \sum_{x \in \mathcal{X}_{i,e}} g_{i,e}(x) \left\| f_{i,e}(x) \right\|_2, \tag{6}$$

where $g_{i,e}$ is the router gate, $f_{i,e}$ the expert output, and $\mathcal{X}_{i,e}$ the calibration inputs on which the expert is active.

Let $R_i^{(t)}$ be the set of K_i GPU-resident experts available for routing in layer i at token t. For both REAP and dynamic REAP, routing is restricted to this set by masking the router logits $z_{i,e}^{(t)}$ before expert selection:

$$\widetilde{z}_{i,e}^{(t)} = \begin{cases} z_{i,e}^{(t)}, & e \in R_i^{(t)}, \\ -\infty, & e \notin R_i^{(t)}, \end{cases} \qquad |R_i^{(t)}| = K_i. \tag{7}$$

Static REAP chooses the K_i highest-scoring experts during calibration and keeps this set fixed. Dynamic REAP instead changes its membership during inference, keeping the expert count and codecs fixed.

Every 256 routed tokens, we pair non-resident experts in descending rank with residents in ascending rank. If at least four pairs have an incoming score more than 5% above the outgoing score, we exchange the first four; otherwise the set stays unchanged to prevent thrashing. Exchanged experts are not eligible again until the next 256 token checkpoint.

We track each expert's unmasked routing probability $p_{i,e}^{(t)}$, so non-resident experts accumulate routing score without being evaluated. We use two exponentially smoothed averages to track changes in the routing score over time. One short-term and one long-term average:

$$m_{i,e,h}^{(t)} = \beta_h m_{i,e,h}^{(t-1)} + (1 - \beta_h) p_{i,e}^{(t)}, \tag{8}$$

with $\beta_h = 2^{-1/h}$ and half-lives $h \in \{100, 1000\}$ tokens, and rank by

$$\text{rank}_{i,e}^{(t)} = \big(0.5\, m_{i,e,100}^{(t)} + 0.5\, m_{i,e,1000}^{(t)} \big). \tag{9}$$

4 Dynamic Compression of Large Models

While Iterative Sensitivity Probing (ISP) is an efficient algorithm it can still be very costly to estimate sensitivities for large models. Therefore there are additional heuristics that we found empirically helpful.

To compress a model to low target bit-per-weight values, say in the range 1-2 bit-per-weight, it becomes important to have a fine-grained codec ladder. However, this means, that we need to estimate the sensitivities of more codecs for each layer. Having 10 codecs and 10 expert removal levels means we need to do 100 evaluations on our calibration set for each layer. This can quickly add up to hundereds of GPU hours of compute for a single ISP iteration.

From our empirical data, we find two tricks that work well: The first is, to evaluate only two extreme codecs and two extreme expert removal levels, for example a codec at 0.75 bit and at 4.0 bit and 5% and 95% expert removal. Once we have these sensitivity measurements, we assume that we can interpolate between these sensitivites to represent codecs and expert removal levels between them. We find that the perplexity degradation across codecs is model independent and follows the closed form of the mean-squared-error distortion of the data type: once its constants are fitted on our codec ladder, the distortion of any codec follows analytically from its bit-width,

$$\varphi(b) = 2^{\,a - c\,b}, \qquad a = 1.710, \quad c = 2.264, \tag{10}$$

where c is close to the 2 of the ideal Gaussian law $D(b) = \sigma^2 2^{-2b}$ (Max, 1960; Gish & Pierce, 1968).

This makes sensitivity estimation with HIGGS quantization (Malinovskii et al., 2024) highly computational efficient since we only need to find the perplexity degradation for two codecs.

For experts, we can empirically approximate perplexity cost for any fraction of expert removal to recover expert removal allocation with a Spearman rank correlation of $\rho \approx 0.9$. We do this as follows: For each layer ℓ we measure the removal curve at three levels $\mathcal{F}$ — no removal, 25% removed, and 95% removed and record the cost of removing a fraction f of a layer's experts as a log-perplexity difference against the same model un-removed,

$$d_\ell(f) \;=\; \ln \mathrm{PPL}_\ell(f) \;-\; \ln \mathrm{PPL}_\ell(0). \tag{11}$$

We model that curve as an origin-fixed power law and fit it *jointly across all layers*, so that one exponent is shared by the whole network while each layer keeps its own amplitude,

$$d_\ell(f) \;=\; a_\ell\, f^{\,p}, \qquad \big(p, \{a_\ell\}\big) \;=\; \underset{p,\,\{a_\ell\}}{\arg\min} \underbrace{\sum_{\ell \in \mathcal{L}}}_{\text{all MoE layers}} \underbrace{\sum_{f \in \mathcal{F}}}_{\text{measured levels}} \big(a_\ell f^{\,p} - d_\ell(f)\big)^2, \tag{12}$$

With the combination of both interpolations, we only need 2×2 sensitivity estimates per layer and one evaluation of the allocated unaltered model. This allows us to allocate the combination of 16 codecs and 20 expert removal levels at 80x less computational cost.

5 Experimental setup

Our aim is to evaluate our method against mixture of expert (MoE) models that are currently used and compare with the best other methods.

Currently used mixture of expert models at scales that are computationally feasible for us include Qwen 3.8 Flash Next (125B), GLM 5.3 Flash (320B), DeepSeek v4.1 Flash (552B). We also evaluate Qwen 3.6 (35B) although it is now rarely used in practice.

The only methods evaluating at this scale are Unsloth (Han et al., 2026) and RCO (Helcig & Alistarh, 2026). RCO uploaded their Qwen 3.8 Flash-Next model 3 days ago and we are unable to compare with their approach in the current version of this draft.

We evaluate with the standard llama.cpp (Gerganov & llama.cpp contributors, 2023) perplexity evaluation tool which evaluates the perplexity in chunks of 512 tokens but only evaluates the perplexity of the last 256 tokens. We only evaluate on WikiText perplexity.

Usually, KLD evaluation is standard, but we found the correlation between KLD and perplexity evaluations to have a correlation of $r = 0.992$ and logits for most models would cost 130 GB of disk space per model which makes evaluation more difficult with limited resources. As such, we stick with perplexity evaluations for all of our work in this paper.

To evaluate Unsloth GGUF we download the official checkpoints and use llama.cpp to evaluate the checkpoints using the standard evaluation tool.

For our setup, we allocate to the same bit-footprint as Unsloth checkpoints by using our dual Lagrangian solver. We then evaluate with dynamic REAP enabled where we swap in 4 experts per layer for every 256 tokens.

We evaluate models after two rounds of ISP where we estimate the sensitivity constants on RedPajama (Weber et al., 2024) over After the second probe, we evaluate.

For GLM 5.3 Flash and DeepSeek v4.1 we start ISP with 4.5 bits, the second iteration at 2.25 bits and then 2 bits. We then evaluate.

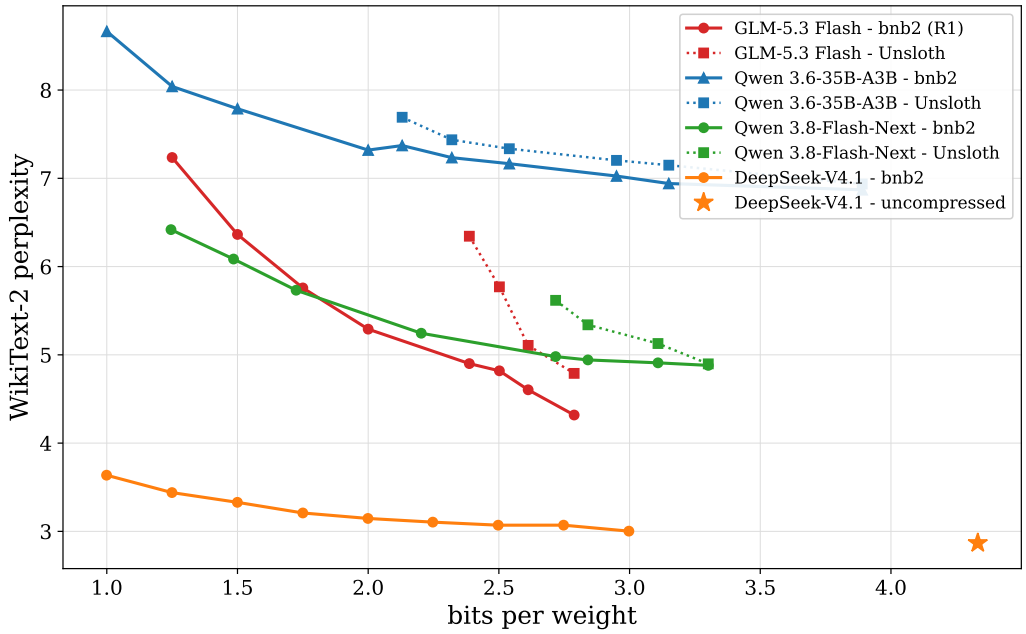

Figure 4: Evaluation perplexity on WikiText-2 for diverse models at a certain bits-per-weights average. We compare with Unsloth. We see that bnb2 compression achieves the same quality as Unsloth dynamic quantization while using between 0.4 and 1.0 bit less.

6 Results

Results are shown in Figure 4 and Figure 1. We can see that our runtime dynamic compression bnb2 is more effective than Unsloth, particularly at lower bit footprints. It is noteworthy that increases in bnb2 compression remain relatively flat even at extreme compression levels around 1-2 bits, while Unsloth dynamic quantization tends to increase exponentially in perplexity at lower bit widths. The notable exception is GLM 5.3 Flash, which is a model that is very difficult to quantize.

Comparing the perplexity level for a given bit footprint, we can see that bnb2 compression is 0.4 to 1 bit more efficient than Unsloth dynamic quantization. For example, Unsloth Qwen 3.6 at 2.2 bits has similar perplexity to 1.5 bits for bnb2; Qwen 3.8 Flash Next at 2.7 bits is similar to 1.7 bits for bnb2; and for GLM 5.3 Flash 2.8 bits is similar to 2.5 bits.

It appears that models with engram parameters, for example, Qwen 3.8 Flash Next with 51B, and DeepSeek v4.1 with 196B engram parameters degrade less when heavily quantized. This is in line with the scaling laws for precision (Kumar et al., 2025) which finds that compressibility is roughly related to the ratio of training tokens to total parameters.

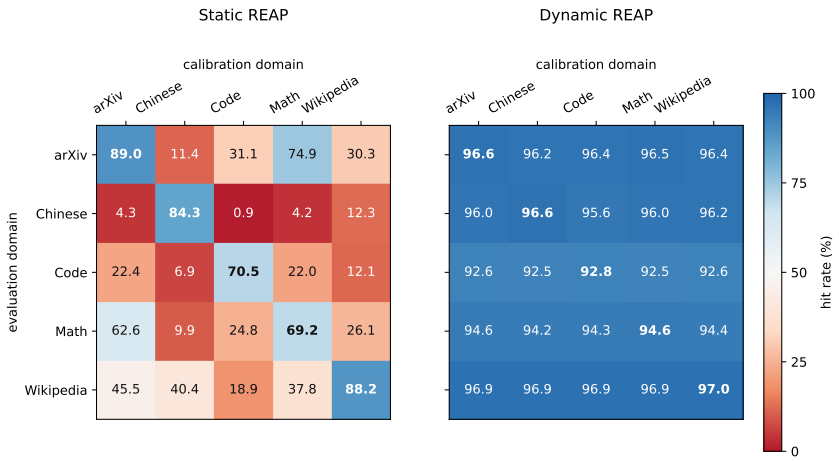

Figure 5: Cross-domain hit rate for resident experts. Static REAP shows poor out-of-distribution generalization while dynamic REAP adjust to any out-of-distribution data. Furthermore, hit-rates within a dataset are also higher, since routing distributions change over the sequence of tokens.

Dynamic REAP cross-domain routing generalization. Figure 5 shows static and dynamic REAP side by side at 50% residency (128 of 256 experts per layer) on Qwen3.6-35B-A3B. Each column calibrates on one domain and each row evaluates on another. Both panels share one color scale (red = low, blue = high). The static panel's strong diagonal and weak off-diagonal cells show that a frozen resident set tracks its calibration corpus. The dynamic panel stays blue almost everywhere: routing-mass coverage remains between 92.5% and 97.0% for all 25 source–evaluation pairs, including pairs where static selection collapses below 5%. Notably, dynamic REAP outperforms static reap even if static REAP is fitted to the test set. This aligns with our observations that routing distributions shift over time and the average routing distribution – even over a single dataset – is a poor proxy for how routing changes within a single document.

7 Related work

Uniform quantization. Quantization methods represent all weights of a model at one target rate. GPTQ (Frantar et al., 2022) rounds with error compensation, AWQ (Lin et al., 2023) rescales activation-salient channels, and SpQR (Dettmers et al., 2023) isolates outliers into a sparse high-precision residue. Vector codebooks push rates below one bit per weight: lattice and additive codes (QuIP# (Tseng et al., 2024a), AQLM (Egiazarian et al., 2024)), trellises (QTIP (Tseng et al., 2024b)), and the fine rate ladder of HIGGS (Malinovskii et al., 2024). These methods choose how weights are represented, not how a byte budget is spent: the codec is not the bottleneck for deployment, the menu of available rates is.

Dynamic and mixed-precision quantization. Allocation spreads a fixed budget across the model by importance. RCO (Helcig & Alistarh, 2026) is the closest work: it meets an exact bit budget at every optimization step and extends the same machinery to expert pruning and MoE allocation. GSQ (Dadgarnia et al., 2026) learns per-coordinate rates through a Gumbel-Softmax relaxation; salience-driven methods keep important weights high (Huang et al., 2024; Lee et al., 2023); Unsloth's dynamic GGUFs make the choice per tensor (Han et al., 2026); EvoPress (Sieberling et al., 2024) searches profiles against measured loss; a recent survey taxonomizes the space (Rakka et al., 2025). All of these settle the assignment offline, before deployment: none prices the embedding or LM head, none covers the KV cache, and none revisits the decision while the model runs.

Expert compression. The expert axis is compressed offline as well. REAP (Lasby et al., 2025) prunes experts in one shot from router statistics, and MxMoE (Duanmu et al., 2025) solves a joint program over expert removal and bitwidths from per-expert sensitivities. In both, the kept set is fixed at compression time — from a workload that has not yet arrived.

Memory efficient serving of mixture of expert models. Offloading systems choose exactness instead of approximation at a trade-off being slower during inference and consuming more CPU memory. HOBBIT (Tang et al., 2024) and FreeToken (Yang et al., 2026) are two systems that prefetching and caching mechanisms to improve the overall throughput of the inference system.

8 LIMITATIONS

A main limitation is the narrow evaluation of our method. We mostly evaluate on WikiText-2 perplexity. While we find that the correlation of perplexity and KLD is $r = 0.98$ and while it has also been shown that the correlation between perplexity and zero-shot tasks is $r = -0.94$ (Dettmers & Zettlemoyer, 2023) the community often notes that KLD and perplexity evaluations are not enough. The largest problem is often out-of-distribution generalization of dynamic quantization methods and while dynamic REAP seems to be effective at out-of-distribution generalization we cannot know for sure without such evaluations.

Another limitation is missing comparison with most methods. We only compare with Unsloth and with RCO for a dense Qwen 3.8 27B model. This is because we believe dynamic quantization methods need to generalize to real cases rather than small MoEs with 1-7B parameters and other methods do not evaluate large MoE models. This makes our method largely uncomparable to the current literature and unfairly disregards the work of other academics. While we try to keep the related work current, we likely missed crediting some work relevant to this area.

Furthermore, we evaluated on only four MoE models in the range of 35 to 550B parameters. And it is not clear that our method generalizes to other MoE or larger MoE models. Dynamic compression of GLM 5.3 Flash was not very successful, which hints that our method might fail to quantize very large models.

Acknowledgment I want to thank my students for their help, support, and patience with the open-source week. This work was supported by Schmidt Sciences, Jane Street Group, and by a Laude Institute Slingshot.

REFERENCES

Felix Abecassis, Anjulie Agrusa, Dong Ahn, Jonah Alben, Stefania Alborghetti, Michael Andersch, Sivakumar Arayandi, Alexis Bjorlin, Aaron Blakeman, Evan Briones, et al. Pretraining large language models with nvfp4. *arXiv preprint arXiv:2509.25149*, 2025.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, et al. QuaRot: Outlier-free 4-bit inference in rotated LLMs. *arXiv preprint arXiv:2404.00456*, 2024.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. *arXiv preprint arXiv:2307.13304*, 2023.

Alireza Dadgarnia, Soroush Tabesh, Mahdi Nikdan, Michael Helcig, Eldar Kurtic, Maximilian Kleinegger, and Dan Alistarh. GSQ: Highly-accurate low-precision scalar quantization for LLMs via Gumbel-Softmax sampling. *arXiv preprint arXiv:2604.18556*, 2026.

Tim Dettmers and Luke Zettlemoyer. The case for 4-bit precision: k-bit inference scaling laws. In *International Conference on Machine Learning*, pp. 7750–7774. PMLR, 2023.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*, 2021.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35: 30318–30332, 2022.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. *arXiv preprint arXiv:2306.03078*, 2023.

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. *arXiv preprint arXiv:2505.05799*, 2025.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. *arXiv preprint arXiv:2401.06118*, 2024.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Georgi Gerganov and llama.cpp contributors. llama.cpp: LLM inference in C/C++. `https://github.com/ggml-org/llama.cpp`, 2023. Accessed 21 September 2026.

Herbert Gish and John N. Pierce. Asymptotically efficient quantizing. *IEEE Transactions on Information Theory*, 14:676–683, 1968.

Daniel Han, Michael Han, and Unsloth team. Unsloth, 2023. URL `https://github.com/unslothai/unsloth`.

Daniel Han, Michael Han, and Unsloth team. Unsloth Dynamic 3.0 GGUFs. Unsloth documentation, `https://unsloth.ai/docs/basics/dynamic-3.0-ggufs`, 2026. Accessed 21 September 2026.

Michael Helcig and Dan Alistarh. Model compression with exact budget constraints via riemannian manifolds. *arXiv preprint arXiv:2605.00649*, 2026.

Wei Huang, Haotong Qin, Yangdong Liu, Yawei Li, Qinshuo Liu, Xianglong Liu, et al. SliM-LLM: Salience-driven mixed-precision quantization for large language models. *arXiv preprint arXiv:2405.14917*, 2024.

Tanishq Kumar, Zachary Ankner, Benjamin Spector, Blake Bordelon, Niklas Muennighoff, Mansheej Paul, Cengiz Pehlevan, Christopher Ré, and Aditi Raghunathan. Scaling laws for precision. In *International Conference on Learning Representations*, volume 2025, pp. 71833–71864, 2025.

Mike Lasby, Ivan Lazarevich, Nish Sinnadurai, Sean Lie, Yani Ioannou, and Vithursan Thangarasa. REAP the experts: Why pruning prevails for one-shot MoE compression. *arXiv preprint arXiv:2510.13999*, 2025.

Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. OWQ: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models. *arXiv preprint arXiv:2306.02272*, 2023.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, et al. AWQ: Activation-aware weight quantization for LLM compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

Vladimir Malinovskii, Andrei Panferov, Ivan Ilin, Han Guo, Peter Richtárik, and Dan Alistarh. Pushing the limits of large language model quantization via the linearity theorem. *arXiv preprint arXiv:2411.17525*, 2024.

Joel Max. Quantizing for minimum distortion. *IRE Transactions on Information Theory*, 6:7–12, 1960.

Mariam Rakka, Marios Fournarakis, Olga Krestinskaya, Jinane Bazzi, Khaled N. Salama, Fadi Kurdahi, Ahmed M. Eltawil, and Mohammed E. Fouda. Mixed-precision quantization for language models: Techniques and prospects. *arXiv preprint arXiv:2510.16805*, 2025.

Oliver Sieberling, Denis Kuznedelev, Eldar Kurtic, and Dan Alistarh. EvoPress: Accurate dynamic model compression via evolutionary search. *arXiv preprint arXiv:2410.14649*, 2024.

Peng Tang, Jiacheng Liu, Xiaofeng Hou, Yifei Pu, Jing Wang, Pheng-Ann Heng, Chao Li, and Minyi Guo. HOBBIT: A mixed precision expert offloading system for fast MoE inference. *arXiv preprint arXiv:2411.01433*, 2024.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better LLM quantization with hadamard incoherence and lattice codebooks. *arXiv preprint arXiv:2402.04396*, 2024a.

Albert Tseng, Qingyao Sun, David Hou, and Christopher De Sa. QTIP: Quantization with trellises and incoherence processing. *arXiv preprint arXiv:2406.11235*, 2024b.

Maurice Weber, Daniel Y Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, et al. Redpajama: an open dataset for training large language models. *Advances in neural information processing systems*, 37: 116462–116492, 2024.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pp. 38087–38099. PMLR, 2023.

Shuo Yang, Xiaoze Fan, Melissa Pan, Haocheng Xi, Zhe Wang, Shanlin Sun, Kurt Keutzer, and Song Han. FreeToken: Efficient edge-native MoE serving with bandwidth-adaptive execution. *arXiv preprint arXiv:2608.16157*, 2026.