

Fewer Experts per Token on Apple Silicon: Speed Opportunity, Realised Speedup and Quality Cost

Two mixture-of-experts models, two weight precisions, three Macs

Magnus Lundstedt
Precisit

Technical report, October 2026

Abstract. Fine-grained Mixture-of-Experts (MoE) language models route each token to a fixed number of experts. A recent study reports that keeping two thirds of the experts a router selects preserves 98.8 % of unpruned performance on average and speeds up decoding by 1.2 to 1.7 $\times$ on GPU serving stacks [1]. We measured the same one-integer change on Apple silicon with MLX, for Qwen3-30B-A3B and DeepSeek-V2-Lite-Chat, each at bf16 and 4-bit weight precision, on three Macs, under a pre-registered protocol. Removing experts raised decode throughput by 1.08 $\times$ to 1.59 $\times$ depending on model, precision, batch size and how many experts were removed. Counting the bytes one decode step reads gives the improvement that removing experts makes available. At batch 1, bf16 models realised about 70 % of that improvement and 4-bit models about half, in both models; at batch 8 the share varied more, and in one condition the measurement exceeded the estimate. Quantization does not change the experts’ share of the bytes, so the precision difference lies in how efficiently a step uses them; on our measurements 4-bit decode moved modelled weight and cache bytes at roughly 60 % of the rate of bf16 decode. The quality cost did not transfer uniformly. On all 1,319 GSM8K test problems, Qwen3-30B-A3B at six of eight experts showed no loss we could detect (bf16: -0.61 percentage points, 95 % interval $[-1.67, +0.38]$), while DeepSeek-V2-Lite at four of six lost 4.2 points. We also report three places where our own measurements corrected us, and we release protocol, code and every raw receipt.

1 Introduction

A Mixture-of-Experts layer replaces one large feed-forward block with many small ones, the experts, and a router that sends each token to a few of them. Qwen3-30B-A3B, for example, has 128 experts per layer and routes each token to 8; DeepSeek-V2-Lite-Chat has 64 routed experts, routes each token to 6, and adds 2 shared experts that every token uses [2], [3]. The number of routed experts per token, `num_experts_per_tok` in the model configuration, is fixed in training, but nothing in the forward pass prevents an inference engine from using fewer.

Chen et al. study exactly this [1]. Truncating every token’s selection to $\lceil 2K/3 \rceil$ of the K trained experts “preserves 98.8% of unpruned performance on average” across nine models and three task families, and at each core model’s reduced budget it raises decode throughput by 1.18 to 1.65 $\times$ on vLLM and 1.44 to 1.73 $\times$ on Hugging Face Transformers, measured at batch size 4 with 1,024-token prompts and 256 generated tokens. The paper does not state weight precision or GPU type; its code runs at least one model, GPT-OSS-20B, with quantized experts.

We asked whether the result carries over to Apple silicon. There, decode is “bounded by memory bandwidth, rather than by compute ability” [4], and MoE models are commonly run with 4-bit weights. If decode time follows the bytes read, removing experts should help in proportion to how much of each step the experts account for. This report answers three questions:

1. Does quality hold at $\lceil 2K/3 \rceil$ experts on a different software stack, for two architectures?
2. What speedup does the change give, and how does it depend on weight precision and batch size?
3. How much of the speedup can be estimated before running the model, and how much is realised?

Our contributions are a pre-registered measurement on three Macs, two models and two precisions; a bytes-read estimate of the available speedup, with the finding that bf16 realises about 70 % and 4-bit about half of it at batch 1; evidence that the quality cost is model-dependent; and a record of three corrections our own measurements forced, which we think are as useful to practitioners as the headline numbers.

2 Background and related work

Routing. In both models a linear gate scores all routed experts, a softmax turns scores into weights, and the top k experts are evaluated. Qwen3-30B-A3B renormalises the kept weights to sum to one (`norm_topk_prob: true`); DeepSeek-V2-Lite does not, so with fewer experts the routed contribution’s total weight shrinks. DeepSeek-V2-Lite’s first layer is dense and its two shared experts run for every token.

Reducing experts at inference. Chen et al. compare uniform truncation with adaptive rules and find that “even the best published rules differ from it by under 1%” at conservative budgets [1]. Lu et al. prune and dynamically skip experts in Mixtral 8x7B, a coarse top-2 model; they report “ $1.2\times$ to $1.3\times$ inference speedup with $r = 6$ ” for expert pruning, and dynamic skipping alone lowers 5-shot GSM8K from 58.61 to 54.28 at $1.08\times$ [5]. Oncescu et al. show that “MoE latency is governed by the number of activated experts” and that batch-aware routing, letting tokens reuse experts already loaded for others in the batch, cuts MoE-layer decode latency by 39 % on Qwen3-30B at batch 16 without significant accuracy loss [6].

Methods that need training. Huang et al. train models with top-p routing so that harder tokens use more experts [7]. Lv et al. use two-stage self-distillation to let Qwen3-30B-A3B and GLM-4.7-Flash skip over half of their expert computation for about $1.2\times$ end-to-end speedup [8]. Neither is an inference-only setting.

Where the time goes. Sakthivel et al. study DeepSeek-V2-Lite and Qwen3-30B-A3B, the two models used here, and find that inference “spends its time waiting on roughly a thousand kernel launches per forward pass”, and that INT4 quantization changes on average 0.53 of the eight selected experts per token position [9].

Runtimes. llama.cpp can override the expert count with `--override-kv <arch>.expert_used_count=int:N` [10]. SwiftLM documents a `SWIFTLM_TOP_K` setting and reports 4.95 to 6.52 tokens per second for 8 down to 2 experts on a 4-bit Qwen3.5-122B-A10B with experts streamed from SSD on an M1 Ultra, without a quality measurement [11]. We found no published replication of [1], no quality measurement of the knob on Apple silicon, and no same-model comparison across weight precisions (searched 3 October 2026).

3 Method

3.1 Models, precisions and the setting

Table 1: Models. “Routed share” is the fraction of bytes one batch-1 decode step reads that belongs to the routed experts at the shipped k , from the byte model of Section 3.5.

Model and precision	Routed experts	Shipped k	Shared	Renormalised	Routed share of bytes, batch 1
Qwen3-30B-A3B, bf16	128	8	none	yes	58 %
Qwen3-30B-A3B, 4-bit	128	8	none	yes	56 %
DeepSeek-V2-Lite-Chat, bf16	64	6	2	no	55 %
DeepSeek-V2-Lite-Chat, 4-bit (ours)	64	6	2	no	53 %

Qwen3-30B-A3B was used through the MLX conversions `mlx-community/Qwen3-30B-A3B-bf16` (revision `3a61c5fb`) and `mlx-community/Qwen3-30B-A3B-4bit` (revision `d388dead`, 4-bit affine, group size 64), which share tokenizer files. DeepSeek-V2-Lite-Chat was loaded from `deepseek-ai/DeepSeek-V2-`

Lite-Chat (revision 85864749) in bf16; its 4-bit arm is our own conversion with the same software (`mlx_lm convert -q --q-bits 4 --q-group-size 64`, 4.503 bits per weight). The paper evaluated Qwen3-30B-A3B-Instruct; we used the original hybrid model in non-thinking mode.

Experts per token was set at load time with `load(path, model_config={"num_experts_per_tok": k})`, which `mlx_lm 0.32.0` merges into the configuration before building the model (`qwen3_moe.py` line 119, `deepseek_v2.py` line 275). Every receipt records the value read back from every MoE layer. We compared the shipped k with $\lceil 2k/3 \rceil$ ($8 \rightarrow 6$ and $6 \rightarrow 4$) and one step further ($8 \rightarrow 5$ and $6 \rightarrow 3$); a 4-bit Qwen3 sweep also covered 7.

3.2 Hardware and software

Three Macs: a MacBook Pro with M1 Max and 32 GB (`m1max-32`, shared with other work and busy during our runs), a Mac mini with M5 Pro and 64 GB (`m5pro-64`) and a Mac Studio with M5 Max and 128 GB (`m5max-128`). The two M5 machines were reserved exclusively through our lab’s booking system and otherwise idle (load average below 3). All runs used Python 3.12, `mlx 0.32.3` and `mlx_lm 0.32.0`; macOS was 26.5.1 on `m1max-32`, 27.0 on `m5pro-64` and 27.0.1 on `m5max-128`. Unless stated, results are from `m5max-128`.

3.3 Speed

Each speed run was a fresh Python process: load the model, warm up with a 32-token generation on a different prompt, then one timed greedy generation. Batch 1 used a 944-token prompt and 512 new tokens; batch 8 used eight distinct essay prompts through `mlx_lm`’s `batch_generate`, 256 new tokens per row. Every run reached its token limit. The two arms of a comparison alternated in the order ABBA BAAB ABBA, six runs each, 20 seconds apart. Decode and prefill throughput are `mlx_lm`’s own counters. A difference counted as real only if the difference of arm means exceeded the larger of the two arms’ ranges (maximum minus minimum), a rule fixed before the first run.

3.4 Quality

Gate probe. The pre-registered probe used the first 200 GSM8K test problems [12] with the sampler the Qwen3 model card recommends for non-thinking mode (temperature 0.7, top-p 0.8, top-k 20), five seeds per arm, reseeded per problem. Its acceptance band was set from the baseline’s own seed variance before any pruned run: $\text{mean} \pm \max(2 \cdot \text{SD} \cdot \sqrt{2/5}, 1/200)$.

Full probe. After the gate result (Section 4.1) we added, by dated amendment, a probe over all 1,319 GSM8K test problems with greedy decoding, so arms could be compared problem by problem, decoded in fixed chunks of 8 problems with `batch_generate`, 512 new tokens. The answer is the number after the last `####`, or else the last number in the response. For each reduced k we report the accuracy difference against the shipped k with a 95 % paired bootstrap interval (10,000 resamples, fixed seed) and an exact two-sided McNemar test. We call a result “no detectable loss” if the interval contains zero and “loss” if it lies below zero; with 1,319 problems the interval is about ± 1 percentage point.

Code probe. HumanEval [13], 164 tasks, greedy, 768 new tokens, scored by running each task’s tests in a separate process with 10-second limits; pass@1 with the same paired statistics. Its intervals are ± 4 to ± 8 points, so it can reveal only large losses.

3.5 Byte model

For each model and precision we summed tensor sizes from the exact checkpoints’ safetensors headers (for our own 4-bit conversion: 4.5 bits per weight for every matrix `mlx_lm` quantizes, bf16 otherwise). One decode step at batch b with k experts per token is taken to read

$$B(k, b) = W_{\text{ne}} + b(w_{\text{emb}} + c_b w_{\text{kv}}) + L \cdot E(1 - (1 - k/E)^b) \cdot w_e$$

where W_{ne} is all non-expert weight bytes except the embedding table, w_{emb} one embedding row, w_{kv} the KV-cache bytes per token at mean context c_b , L the number of MoE layers, E the routed experts per layer and w_e the bytes of one expert in one layer. The middle factor is the expected number of

distinct experts per layer if b tokens each chose k of E uniformly at random; real routing is correlated, so at batch 8 it over-counts expert bytes. The **available speedup** is $S_{\text{bytes}} = B(k_{\text{shipped}}, b)/B(k, b)$: what removing experts would give if step time were proportional to bytes read. It is an estimate of the opportunity, not a prediction of the outcome and not a strict upper bound. Because speedups are close to one, we compare improvements rather than ratios: the **share realised** is $(S_{\text{measured}} - 1)/(S_{\text{bytes}} - 1)$. (A measured $1.08\times$ against an available $1.16\times$ is 93 % as a ratio but 48 % of the improvement.) Dividing B by the measured time per step gives **useful bytes per second**, a modelled quantity, not a hardware counter.

3.6 Pre-registration and deviations

The protocol was committed before any run and amended seven times, each amendment dated and committed before the runs it governs (Section A). Deviations from the original plan, all recorded: the first host was not one of the planned ones; the 7, 5 and 3 expert points and the full probe were added after the gate result; a description of the source paper was corrected after all runs (A6); an interpretation of the precision effect was withdrawn before the second model ran (A7); the first attempt of the second-model run stopped at its quantization step because of a wrong cache path and was re-run in full; and one scoring check was added after the fact (Section 4.7).

4 Results

4.1 The pre-registered gate

On m1max-32, Qwen3-30B-A3B 4-bit scored 0.9430 at 8 experts and 0.9280 at 6 (five seeds, 200 problems), below the pre-agreed band’s lower edge of 0.9307. Batch-1 decode rose from 57.99 to 62.89 tokens per second, a difference smaller than the run-to-run range (6.74) on that busy machine. The protocol’s verdict was therefore “no transfer at matched quality”. On the idle m5pro-64 the same comparison gave $1.098\times$ at batch 1 and $1.113\times$ at batch 8 with ranges under 0.5 tokens per second, so the speed half was real; the verdict, “faster, but not at matched quality”, was recorded and has not been revised. The full probe below, which differs from the gate probe in decoding (greedy rather than sampled), batch and size, did not reproduce the quality drop.

4.2 Quality at full resolution

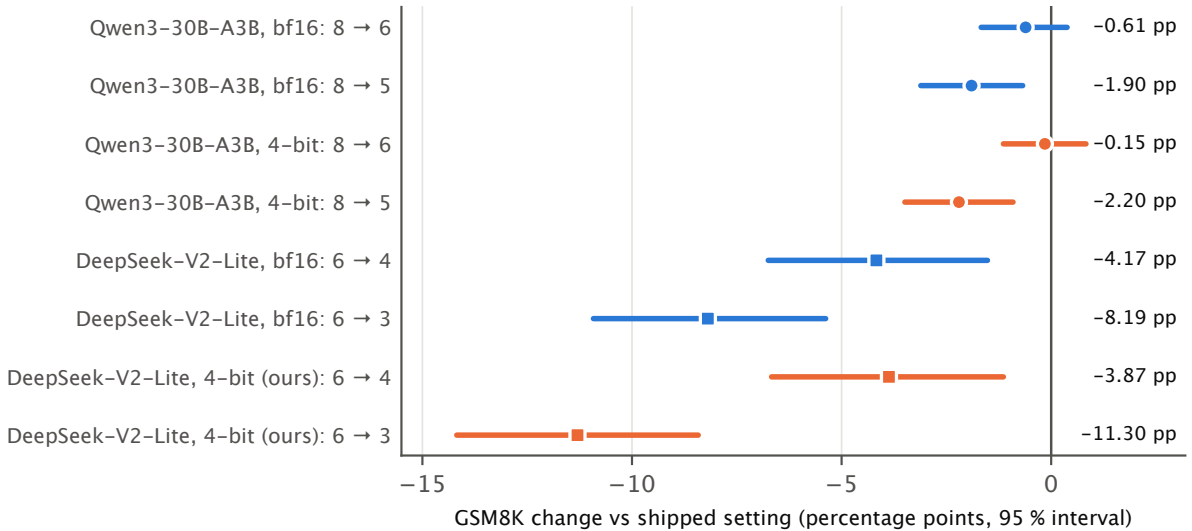


Figure 1: GSM8K accuracy change against the shipped number of experts, all 1,319 test problems, greedy decoding, m5max-128. Bars are 95 % paired bootstrap intervals. Colour: precision (blue bf16, orange 4-bit); marker: model (circle Qwen3-30B-A3B, square DeepSeek-V2-Lite).

Table 2: GSM8K, 1,319 problems, greedy, paired against the shipped k (m5max-128). The 4-bit Qwen3 run on m5pro-64 gave identical answers on every problem.

Model, precision	k	Accuracy	Change (pp)	95 % interval	McNemar p	Reading
Qwen3, bf16	8	0.9409	shipped			
	6	0.9348	−0.61	[−1.67, +0.38]	0.312	no detectable loss
	5	0.9219	−1.90	[−3.11, −0.68]	0.002	loss
Qwen3, 4-bit	8	0.9303	shipped			
	6	0.9287	−0.15	[−1.14, +0.83]	0.880	no detectable loss
	5	0.9083	−2.20	[−3.49, −0.91]	0.001	loss
DeepSeek, bf16	6	0.6967	shipped			
	4	0.6550	−4.17	[−6.75, −1.52]	0.002	loss
	3	0.6149	−8.19	[−10.92, −5.38]	< 0.001	loss
DeepSeek, 4-bit (ours)	6	0.5982	shipped			
	4	0.5595	−3.87	[−6.67, −1.14]	0.007	loss
	3	0.4852	−11.30	[−14.18, −8.42]	< 0.001	loss

The two models behave differently (Figure 1, Table 2). Qwen3-30B-A3B shows no detectable loss at 6 of 8 experts at either precision and loses about 2 points at 5. DeepSeek-V2-Lite loses about 4 points already at 4 of 6 experts and 8 to 11 points at 3. For the same models [1] reports GSM8K of 94.4, 95.0 and 93.9 for Qwen3-30B-A3B-Instruct at 8, 6 and 5 experts, which agrees with our Qwen3 result, and 63.1, 61.7 and 56.1 for DeepSeek-V2-Lite-Chat at 6, 4 and 3, a smaller drop at 4 than ours. Prompting, answer extraction and decoding differ between the studies.

4.3 Speed

Table 3: Decode throughput in tokens per second (batch 8: aggregate over the batch), mean of six fresh-process runs per arm, m5max-128. Every decode difference exceeds the larger arm’s range and the arms do not overlap; the largest range in any decode arm was 1.74 tokens per second. Per-arm minima and maxima are in Section B.

Model, precision	Change	Decode, batch 1	Decode, batch 8	Prefill, batch 1	Peak memory
Qwen3, bf16	8 → 6	70.3 → 79.0 (1.12×)	163.1 → 232.5 (1.43×)	1.10×	61.7 → 61.6 GB
	8 → 5	70.3 → 84.2 (1.20×)	163.1 → 259.4 (1.59×)	1.16×	61.7 → 61.6 GB
Qwen3, 4-bit	8 → 6	144.4 → 155.6 (1.08×)	420.7 → 474.7 (1.13×)	1.15×	17.8 → 17.9 GB
	8 → 5	144.3 → 161.0 (1.12×)	420.5 → 520.5 (1.24×)	1.24×	17.8 → 17.8 GB
DeepSeek, bf16	6 → 4	90.6 → 105.2 (1.16×)	251.6 → 316.1 (1.26×)	1.07×	34.6 → 34.6 GB
	6 → 3	90.6 → 115.0 (1.27×)	251.5 → 374.9 (1.49×)	1.14×	34.6 → 34.5 GB
DeepSeek, 4-bit	6 → 4	194.3 → 215.2 (1.11×)	538.6 → 668.3 (1.24×)	1.12×	9.7 → 9.8 GB
	6 → 3	194.3 → 228.6 (1.18×)	538.6 → 761.3 (1.41×)	1.22×	9.7 → 9.7 GB

Removing experts always made decoding faster (Table 3). At $\lceil 2k/3 \rceil$ experts the gain ranged from 1.08× (Qwen3, 4-bit, batch 1) to 1.43× (Qwen3, bf16, batch 8). bf16 gained more than 4-bit in every condition, and batch 8 more than batch 1. Prefill also became faster. Peak memory did not change, since all experts stay resident in memory.

4.4 The byte model against measurement

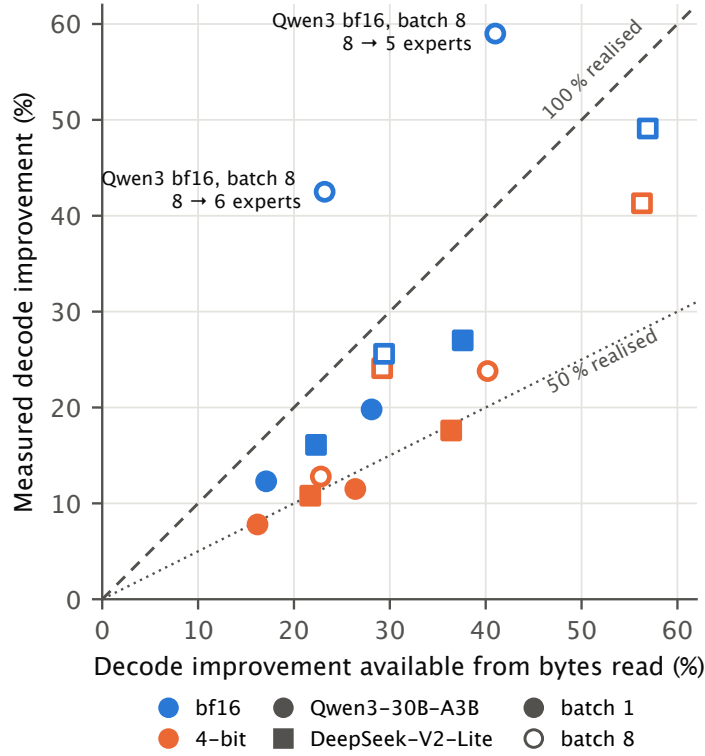


Figure 2: Measured decode improvement against the improvement available from bytes read per step (Section 3.5), m5max-128. Each point is one model, precision, batch size and expert reduction. Dashed: all of the available improvement realised; dotted: half.

At batch 1 the result is regular across both models (Figure 2, Table 4): bf16 realised 70 % to 72 % of the available improvement and 4-bit 44 % to 50 %. At batch 8 the shares are higher and more spread: 4-bit 56 % to 83 %, bf16 DeepSeek 86 % to 87 %. Qwen3-30B-A3B at bf16 and batch 8 measured $1.43\times$ against an available $1.23\times$ ($1.59\times$ against $1.41\times$ at 5 experts), more than the byte count allows for. We have no explanation; it did not recur for DeepSeek-V2-Lite at the same precision and batch, and it shows the byte estimate is not a strict bound.

Table 4: Useful bytes per second (byte-model bytes per step times measured steps per second) at the shipped k , and the decode improvement available from bytes, measured, and the share realised, at $\lceil 2k/3 \rceil$ experts, m5max-128.

Model, precision	Batch	Useful GB/s at shipped k	Available improvement	Measured improvement	Share realised
Qwen3, bf16	1	436	+17.1 %	+12.3 %	72 %
	8	530	+23.2 %	+42.5 %	184 %
Qwen3, 4-bit	1	264	+16.2 %	+7.8 %	48 %
	8	390	+22.8 %	+12.8 %	56 %
DeepSeek, bf16	1	448	+22.3 %	+16.1 %	72 %
	8	564	+29.4 %	+25.6 %	87 %
DeepSeek, 4-bit	1	276	+21.7 %	+10.8 %	50 %
	8	342	+29.2 %	+24.1 %	83 %

Quantization shrinks expert and non-expert weights alike, so the routed share of bytes is nearly the same at both precisions (Table 1). What differs is how fast the bytes are read: at batch 1 the 4-bit models reached 264 and 276 useful GB/s against 436 and 448 at bf16 (Table 4). That is the observation. One explanation consistent with it is that a 4-bit step spends a larger share of its time on work that does not scale with the bytes removed, such as dequantization, kernel launches or synchronisation; we did not profile which. In every condition 4-bit realised a smaller share of the available improvement than bf16 of the same model.

4.5 Code

Table 5: HumanEval, 164 tasks, greedy. *Our 4-bit DeepSeek-V2-Lite conversion often omitted the code block (33 of 164 answers at 6 experts, 100 at 3), so this row mostly measures answer formatting and is not used.

Model, precision	k	pass@1	Change (pp), 95 % interval	Reading
Qwen3, bf16	8	0.896	shipped	
	6	0.878	−1.8 [−4.9, +1.2]	no detectable loss
	5	0.866	−3.0 [−7.3, +1.2]	no detectable loss
Qwen3, 4-bit	8	0.902	shipped	
	6	0.902	0.0 [−4.3, +4.3]	no detectable loss
	5	0.842	−6.1 [−11.6, −1.2]	loss
DeepSeek, bf16	6	0.531	shipped	
	4	0.512	−1.8 [−9.1, +5.5]	no detectable loss
	3	0.402	−12.8 [−20.7, −4.9]	loss
DeepSeek, 4-bit (ours)*	6	0.372	shipped	
	4	0.451	+7.9 [0.6, +15.2]	gain
	3	0.159	−21.3 [−29.3, −13.4]	loss

Where usable, HumanEval agrees with GSM8K in direction (Table 5): no detectable loss at $\lceil 2k/3 \rceil$ experts for either model at bf16 or for 4-bit Qwen3, and a clear loss for DeepSeek-V2-Lite at 3 experts.

4.6 Output length and wall time

Table 6: Total tokens generated and wall time to answer all 1,319 GSM8K problems (batch-8 chunks), relative to the shipped k . One run per configuration.

Model, precision	k	Tokens generated vs shipped	Wall time vs shipped
Qwen3, bf16	6	+4.0 %	−18.9 %
	5	+7.1 %	−25.0 %
Qwen3, 4-bit	6	+5.2 %	−8.3 %
	5	+9.6 %	−12.4 %
DeepSeek, bf16	4	+3.8 %	−16.7 %
	3	+10.7 %	−24.0 %
DeepSeek, 4-bit	4	+2.2 %	−15.2 %
	3	+4.3 %	−23.1 %

With fewer experts both models wrote longer answers (Table 6). The wall-time column is complementary evidence, not a direct check of the speed series: it is a different workload (problems of varying length, prefill included, batches in which finished rows wait for the longest one). For orientation, the batch-8 speed series for 4-bit Qwen3 at 6 experts ($1.128\times$) corresponds to 11.4 % less decode time for unchanged output, and the GSM8K run took 8.3 % less time while generating 5.2 % more tokens. Longer outputs plausibly account for part of such gaps; we did not separate them from the other differences between the workloads. Benchmarks that generate a fixed number of tokens cannot show the change in output length at all.

4.7 Other observations

Answer extraction. DeepSeek-V2-Lite rarely writes the requested ##### line; 76 to 95 % of its answers were scored by the fallback rule. A post-hoc re-scoring with a different rule (the number after the last “answer”) changed the verdict on 18 to 30 problems per arm and none of the conclusions; the $6 \rightarrow 4$ change becomes −5.31 points at bf16 and −3.34 at 4-bit.

Our 4-bit DeepSeek-V2-Lite conversion. At the shipped k it scored 9.9 points below bf16 on GSM8K, against 1.1 for the Qwen3 4-bit conversion, and it produced stray tokens in code answers. Comparisons between expert counts within this conversion remain paired and valid, but it is a weak 4-bit model.

Perplexity. For 4-bit Qwen3 on m1max-32, teacher-forced perplexity on 8,176 tokens of GSM8K training text was lower at 6 experts than at 8 (3.110 against 3.214), the opposite direction to task accuracy in that probe.

Determinism. The 4-bit Qwen3 GSM8K run produced byte-identical answers to all 1,319 problems on m5pro-64 and m5max-128. Between m1max-32 and m5pro-64, a re-run of one seed of the sampled gate probe gave different text for most problems (46 and 13 of 200 identical at 8 and 6 experts) while accuracy agreed within one problem.

5 Discussion

The opportunity can be estimated in advance; the outcome cannot, yet. In both models the routed experts are a little over half of the bytes a decode step reads, at either precision. Going to $\lceil 2k/3 \rceil$ removes a quarter of the routed experts for Qwen3-30B-A3B ($8 \rightarrow 6$) and a third for DeepSeek-V2-Lite ($6 \rightarrow 4$), so the bytes read per batch-1 step fall by about a seventh and a fifth: an available $1.17\times$ and $1.22\times$. A practitioner can compute this from the checkpoint before trying the setting. How much of it appears depends on precision: at batch 1, about 70 % at bf16 and about half at 4-bit in our measurements. At batch 8 tokens share experts, the relative saving in bytes is larger, and the realised share was higher but less regular.

Observation and explanation for 4-bit. Observed: 4-bit decode moved modelled bytes at roughly 60 % of the rate of bf16 decode on the same machine, and realised a smaller share of the available improvement in every condition. Not established: why. Dequantization work, kernel-launch overhead or other per-step costs that do not shrink with fewer experts would each produce this pattern; launch-bound inference has been reported for the same two models [9], but we did not profile ours. The practical consequence holds either way: the configuration most people run on a Mac, a 4-bit model at batch 1, gains the least, about 1.08 to $1.11\times$ at $\lceil 2k/3 \rceil$ experts.

Quality is model-dependent. Qwen3-30B-A3B tolerated six of its eight experts; DeepSeek-V2-Lite did not tolerate four of six on our setup, although [1] reports a smaller drop for it. Two architectural differences may matter and were not tested: DeepSeek-V2-Lite does not renormalise the kept experts' weights, and with 64 rather than 128 routed experts each expert carries more of the computation. The general lesson is that the quality side must be measured per model, on enough problems to resolve a one-point change.

Different probes, different answers. Our pre-registered probe (200 problems, sampled decoding, five seeds, batch 1, on m1max-32) measured a 1.5-point loss for Qwen3 at 6 experts. The later probe (1,319 problems, greedy decoding, batch-8 chunks, on M5 machines) did not reproduce it, and a perplexity check pointed the other way. Sample size, decoding method, batching and machine all changed between the first two, so we cannot attribute the difference to any one of them; whether sampled decoding is more sensitive to pruning is untested at the larger size. We kept the gate verdict as recorded and report both.

6 Limitations

Two models, of 16 and 30 billion parameters with 2.4 and 3.3 billion active per token. Greedy, non-thinking decoding for the large probes, and one maths benchmark plus a small code benchmark; nothing here measures open-ended writing, other languages or long contexts. Batch size at most 8 and prompts up to about 1,000 tokens. One 4-bit recipe per model, one of them our own and lossy. Bandwidth figures are modelled bytes over measured time, not hardware counters, and the batch-8 byte model assumes uniform routing. One run per configuration for the quality probes, which are deterministic on a given machine. Three Macs, two of them from the same generation.

7 Reproducibility

The pre-registered protocol with all amendments, the code that was run and every raw receipt (environment, per-run speed records, every model output, analysis summaries) are in the repository github.com/precisit/apple-silicon-moe-experts-per-token, tag report-2026-10 (commit 902fb96),

under the MIT licence. All model revisions, dataset revisions (GSM8K 740312ad, HumanEval 7dce6050) and package versions are pinned there, and each table and figure in this report is generated from those receipts.

8 Acknowledgements

The measurement harness, the runs on the three machines, the analysis and the drafts of this report were produced with AI coding agents working under the author’s direction, and an editorial review improved several interpretations (Section 3.5). Machine time was provided by Precisi’s lab cluster.

References

- [1] Y. Chen *et al.*, “You Only Need 2/3 of the Chosen Experts: An Empirical Study of Dynamic Expert Pruning in Fine-Grained MoE LLMs.” [Online]. Available: <https://arxiv.org/abs/2609.25809>
- [2] Qwen Team, “Qwen3 Technical Report.” [Online]. Available: <https://arxiv.org/abs/2505.09388>
- [3] DeepSeek-AI, “DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model.” [Online]. Available: <https://arxiv.org/abs/2405.04434>
- [4] Apple Machine Learning Research, “Exploring LLMs with MLX and the Neural Accelerators in the M5 GPU.” [Online]. Available: <https://machinelearning.apple.com/research/exploring-llms-mlx-m5>
- [5] X. Lu *et al.*, “Not All Experts are Equal: Efficient Expert Pruning and Skipping for Mixture-of-Experts Large Language Models,” in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. [Online]. Available: <https://aclanthology.org/2024.acl-long.334>
- [6] C.-A. Onescu *et al.*, “Opportunistic Expert Activation: Batch-Aware Expert Routing for Faster Decode Without Retraining.” [Online]. Available: <https://arxiv.org/abs/2511.02237>
- [7] Q. Huang *et al.*, “Harder Tasks Need More Experts: Dynamic Routing in MoE Models.” [Online]. Available: <https://arxiv.org/abs/2403.07652>
- [8] X. Lv *et al.*, “Post-Trained MoE Can Skip Half Experts via Self-Distillation.” [Online]. Available: <https://arxiv.org/abs/2605.18643>
- [9] G. Sakthivel, J. Wu, A. Rajendra, and G. Radhakrishnan, “Launch-Bound and Substitutable: Why Three Inference Optimizations Fail to Pay Off in Mixture-of-Experts Models.” [Online]. Available: <https://arxiv.org/abs/2608.26612>
- [10] ggml-org/llama.cpp contributors, “Feature Request: Support per-layer active expert count (top-k) for MoE architectures.” [Online]. Available: <https://github.com/ggml-org/llama.cpp/issues/28380>
- [11] SharpAI, “SwiftLM README: SWIFTLM_TOP_K.” [Online]. Available: <https://github.com/SharpAI/SwiftLM>
- [12] K. Cobbe *et al.*, “Training Verifiers to Solve Math Word Problems.” [Online]. Available: <https://arxiv.org/abs/2110.14168>
- [13] M. Chen *et al.*, “Evaluating Large Language Models Trained on Code.” [Online]. Available: <https://arxiv.org/abs/2107.03374>

A Protocol amendments

Amendment	Committed (UTC)	Content
Protocol	2026-10-01 11:33	Question, single variable, measures, acceptance bar, kill criterion.
A1	2026-10-01 11:50	Host, pins, verification of the setting in mlx-lm 0.32.0, speed harness, noise rule, quality-band rule. Before any run.
A2	2026-10-01 13:31	Numeric quality band from the baseline’s seed variance. Before any reduced- k run.
A3, A3a	2026-10-01 16:43, 21:30	Quiet-host repeat of the speed series on m5pro-64 ; verdict rule for it.
A4	2026-10-01 22:09	Trade-off sweep (7, 6, 5 experts), full GSM8K probe and its statistics, stated as a new question that cannot change the gate verdict.
A5	2026-10-02 23:48	Precision arm (bf16 and 4-bit) and HumanEval on m5max-128 .
A6	2026-10-03 06:22	Correction of the background description of [1] (precision not stated in the paper). No measurement changed.
A7	2026-10-03 11:32	Second model, the byte model, withdrawal of an earlier byte-share interpretation, and written predictions for the second model, before it ran.

Outcome of the A7 predictions: the byte model was meant as an estimate of the available speedup; for DeepSeek-V2-Lite the measured improvement was 48 to 87 % of the available improvement, below it in every condition (held as an estimate, but A7 stated it as a ratio of speedups, which overstates agreement; see Section 3.5); bf16 gained more than 4-bit at both batch sizes (held, small gap); 4 of 6 experts within 1.5 points on GSM8K (failed, -4.2 at bf16); 3 of 6 experts a clear loss (held).

B Speed series, all arms

Host	Model	Change	Batch	Shipped arm	Reduced arm	Ratio	Larger range
m5pro-64	Qwen3, 4-bit	8 $\rightarrow$ 5	1	98.22 (97.86–98.50)	113.88 (113.69–114.09)	1.159	0.64
m5pro-64	Qwen3, 4-bit	8 $\rightarrow$ 5	8	235.14 (235.02–235.35)	292.42 (292.23–292.54)	1.244	0.33
m5pro-64	Qwen3, 4-bit	8 $\rightarrow$ 6	1	98.19 (97.97–98.37)	107.79 (107.67–107.94)	1.098	0.40
m5pro-64	Qwen3, 4-bit	8 $\rightarrow$ 6	8	235.41 (235.17–235.58)	262.09 (261.95–262.26)	1.113	0.41
m5pro-64	Qwen3, 4-bit	8 $\rightarrow$ 7	1	98.16 (98.03–98.31)	102.73 (102.64–102.93)	1.047	0.29
m5pro-64	Qwen3, 4-bit	8 $\rightarrow$ 7	8	235.09 (234.97–235.46)	238.44 (238.31–238.60)	1.014	0.50
m5max-128	Qwen3, 4-bit	8 $\rightarrow$ 5	1	144.34 (144.15–144.79)	160.96 (160.36–161.66)	1.115	1.31
m5max-128	Qwen3, 4-bit	8 $\rightarrow$ 5	8	420.52 (420.27–420.79)	520.51 (520.01–520.90)	1.238	0.90
m5max-128	Qwen3, 4-bit	8 $\rightarrow$ 6	1	144.40 (144.10–145.01)	155.61 (155.14–156.26)	1.078	1.12
m5max-128	Qwen3, 4-bit	8 $\rightarrow$ 6	8	420.65 (420.51–420.80)	474.66 (474.36–474.91)	1.128	0.55
m5max-128	Qwen3, bf16	8 $\rightarrow$ 5	1	70.30 (70.26–70.38)	84.19 (84.07–84.32)	1.198	0.26
m5max-128	Qwen3, bf16	8 $\rightarrow$ 5	8	163.14 (162.97–163.34)	259.41 (259.27–259.52)	1.590	0.37
m5max-128	Qwen3, bf16	8 $\rightarrow$ 6	1	70.32 (70.22–70.39)	78.99 (78.92–79.08)	1.123	0.17
m5max-128	Qwen3, bf16	8 $\rightarrow$ 6	8	163.11 (162.99–163.34)	232.49 (232.44–232.53)	1.425	0.36
m5max-128	DeepSeek, 4-bit	6 $\rightarrow$ 3	1	194.34 (193.84–194.73)	228.61 (227.66–229.21)	1.176	1.55
m5max-128	DeepSeek, 4-bit	6 $\rightarrow$ 3	8	538.64 (538.53–538.79)	761.34 (760.93–761.64)	1.413	0.71
m5max-128	DeepSeek, 4-bit	6 $\rightarrow$ 4	1	194.25 (193.44–194.68)	215.16 (214.23–215.97)	1.108	1.74
m5max-128	DeepSeek, 4-bit	6 $\rightarrow$ 4	8	538.60 (538.42–538.76)	668.34 (667.83–668.61)	1.241	0.78
m5max-128	DeepSeek, bf16	6 $\rightarrow$ 3	1	90.57 (90.50–90.64)	115.02 (114.94–115.15)	1.270	0.21
m5max-128	DeepSeek, bf16	6 $\rightarrow$ 3	8	251.54 (251.49–251.58)	374.93 (374.86–375.03)	1.491	0.18
m5max-128	DeepSeek, bf16	6 $\rightarrow$ 4	1	90.61 (90.44–90.78)	105.17 (104.95–105.27)	1.161	0.34
m5max-128	DeepSeek, bf16	6 $\rightarrow$ 4	8	251.64 (251.60–251.72)	316.11 (315.98–316.19)	1.256	0.21

Decode tokens per second, mean (minimum–maximum) of six fresh-process runs per arm. “Larger range” is the larger of the two arms’ ranges, the noise threshold of the pre-registered rule. Every series in this table exceeds it.