

QuantShift: How Weight Quantization Reshapes the Per-Layer Acceptance Landscape of Self-Speculative Decoding

Jiseung Jeong

Ulsan National Institute of Science and Technology (UNIST)

CSE402 Natural Language Processing, course project

wjdwltmd1151@gmail.com

Abstract

Weight quantization lets a 7B model fit on a consumer GPU, and LayerSkip self-speculative decoding (self-SD) speeds up generation without a separate draft model. Combining them is attractive, but self-SD is only fast if its shallow-layer drafts keep agreeing with the full model, and it is unknown whether this *acceptance* survives quantization. We measure acceptance at all 32 exit layers of LayerSkip Llama-2-7B under INT8 and NF4 on HumanEval and CNN/DailyMail, using a one-pass estimator that we first cross-check against real self-SD runs. We find two things. First, **acceptance is preserved**: quantization lowers it by about one percentage point at most, at every layer, a preliminary 1B check agrees, and a per-position analysis shows why (about 97% of draft-verify matches are untouched). Second, **speed is not**: self-SD is 1.6–1.9 $\times$ faster than autoregressive decoding in fp16 and INT8, but 0.8–0.9 $\times$ in NF4. Forward-pass timings point to the verify step: in our bitsandbytes setup, an NF4 forward pass over several tokens costs more than five single-token passes. None of the exit layers we tried fixes this, which points to the multi-token 4-bit kernel as the bottleneck.

1 Introduction

Running large language models on consumer hardware means working within tight memory and latency budgets. Two techniques address these budgets:

- **Weight quantization** stores parameters in 8 or 4 bits, shrinking a 7B model’s weights from ~ 13.5 GB to about 7 GB or 4 GB (Dettmers et al., 2022, 2023).
- **Self-speculative decoding** (self-SD; Elhoushi et al., 2024) drafts several tokens with the

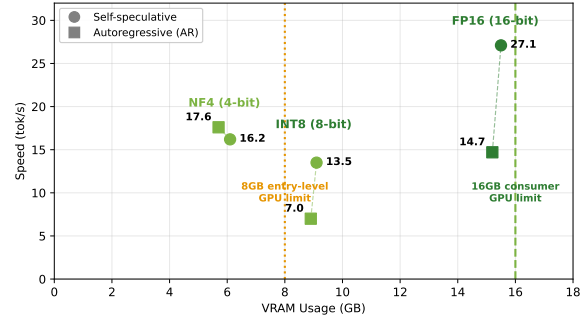


Figure 1: **self-SD helps fp16 and INT8 but hurts NF4**. Throughput vs. peak allocated GPU memory for autoregressive (AR, ■) and self-speculative (self-SD, ●) decoding (LayerSkip Llama-2-7B, CNN/DM). Sections 5–8 explain why.

model’s first E layers, then checks them all with one verify pass of the full model. The output is unchanged, and the speedup depends on *acceptance*: how often the shallow draft agrees with the full model.

Quantization makes the model fit; self-SD makes it faster without a second model. They are natural partners, yet how post-training weight quantization affects layer-wise drafting has not been measured. Our question is:

*Does self-SD acceptance survive weight quantization,
and if so, does quantized self-SD actually run faster?*

Figure 1 shows that the answer is not obvious: self-SD speeds up fp16 and INT8 but *slows down* NF4. We answer the question with a measurement study and make three contributions:

1. **A cross-checked estimator**. We adapt a one-pass estimator (Zarch et al., 2025) that reads the draft of every exit layer at once, and show that it matches real self-SD within 0.05 in all 24 configurations we tested (§4).

2. **Acceptance is preserved, and we show why.** Quantization lowers acceptance by about one percentage point at most. It rarely changes a prediction that matters for the draft–verify match, and the few changes it makes go in both directions (§5–7).
3. **Speed depends on the verify kernel.** self-SD pays off when its total time per emitted token, drafting and verification included, is lower than AR’s. In fp16 and INT8 a multi-token verify costs about as much as one decoding step, which makes this easy; in bitsandbytes NF4 it costs several steps, and none of the exit layers we tried changes that (§8).

2 Background

Self-speculative decoding. Speculative decoding (Leviathan et al., 2023) drafts several tokens cheaply and verifies them with the target model in one forward pass. LayerSkip (Elhoushi et al., 2024) trains a model so that its intermediate layers can predict tokens through the shared LM head. A single model can then act as its own drafter. With L layers in total, each self-SD *block*:

1. drafts γ tokens one at a time using only the first E layers (the *exit layer*);
2. verifies all of them in one pass: the remaining $L - E$ layers process the whole block at once, reusing the drafts’ cached early-layer states;
3. keeps the drafts up to the first mismatch, plus one token from the verifier.

More accepted drafts per block means fewer verify passes, hence higher speed.

Weight quantization. We use bitsandbytes (Dettmers and bitsandbytes contributors, 2022) for 8-bit (INT8, LLM.int8()); Dettmers et al., 2022) and 4-bit (NF4; Dettmers et al., 2023) post-training weight quantization. Our NF4 configuration enables double quantization and bfloat16 computation; LLM.int8() quantizes non-outlier weight and activation components to 8 bits and handles outliers in 16-bit. Its main goal is saving memory. Its effect on speed depends on the kernel: decoding one token is limited by memory bandwidth, so reading fewer bytes can make it faster, but dequantizing the weights adds compute.

Measuring acceptance. Take a reference sequence and run it through the model with teacher

forcing. At each position t , pass the layer- E hidden state through the shared final norm and LM head, and let $\hat{y}_t^E$ be the top-1 token. The *acceptance* of exit layer E is the fraction of positions where this draft agrees with the full model:

$$\hat{p}(E) = \text{mean}_t \mathbf{1}[\hat{y}_t^E = \hat{y}_t^L]. \quad (1)$$

For a quantized model q , $\hat{p}_q(E)$ compares the model with *itself*, so it measures draft–verify agreement, not output quality. Our main quantity is the *acceptance erosion*, the drop caused by quantization:

$$\Delta\alpha(E) = \hat{p}_{\text{fp16}}(E) - \hat{p}_q(E), \quad (2)$$

for $q \in \{\text{INT8}, \text{NF4}\}$. A value near zero means quantization leaves self-SD acceptance intact.

3 Related Work

Prior work differs from ours in either *how it drafts* or *what it quantizes*. **Full-precision early exit.** DEL (Zarch et al., 2025) picks the exit layer dynamically at full precision. We borrow its one-pass measurement but apply it to quantized models. **Quantization as the drafter.** QSpec (Zhao et al., 2025) shares quantized weights between drafter and verifier and drafts with lower-precision activations; QuantSpec (Tiwari et al., 2025) drafts with quantized weights and a hierarchical quantized KV cache. In both, precision is what separates drafter from verifier. We instead keep precision fixed and vary draft *depth* within a quantized LayerSkip model. **Separate draft models.** SpecMQuant (Zhang et al., 2025) observes, for EAGLE-style drafting, that acceptance can stay largely intact under quantization while speculative gains shrink, because verification becomes relatively more expensive. Our results show a similar pattern for layer-wise drafting, for which we identify an implementation-specific explanation. **Quantization meets early exit.** BitSkip (Bhuvaneswaran and Liu, 2025) studies how quantization and early exit combine, but not acceptance layer by layer. To our knowledge, no prior work measures per-layer acceptance of weight-quantized, layer-wise self-SD, or explains when it gives a wall-clock speedup.

4 Method and Setup

Shadow-token estimator. Running self-SD once per exit layer would take 192 runs. Following

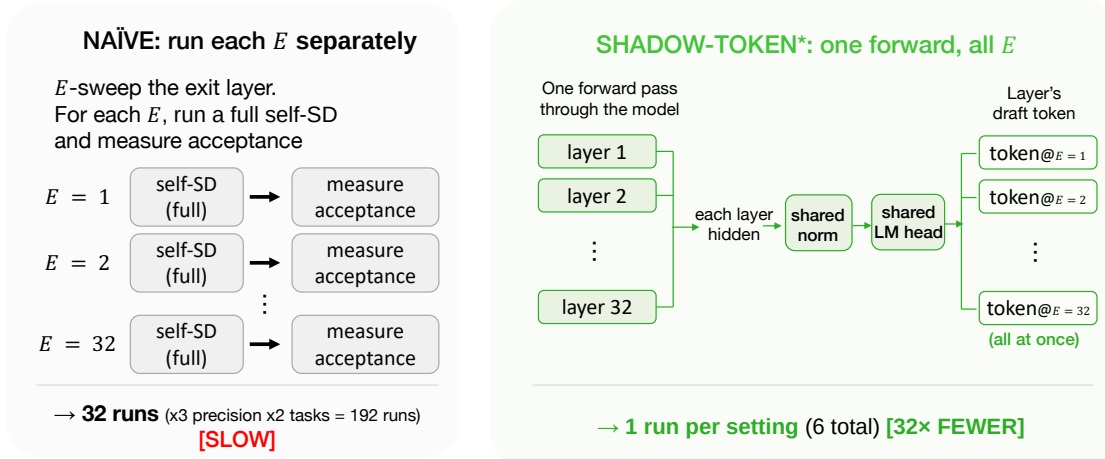


Figure 2: **Six collection runs instead of 192.** *Left:* measuring each exit layer separately takes 32 self-SD runs per setting, i.e. 192 for 3 precisions $\times$ 2 tasks. *Right:* the shadow-token estimator (Zarch et al., 2025) routes every layer’s hidden state through the shared norm and LM head, so one teacher-forced pass per reference sequence yields all 32 layers’ drafts, and one run per precision and task suffices (6 in total).

Zarch et al. (2025), we instead run *one* teacher-forced forward pass per reference sequence and read off, at every layer, the token that layer would have drafted (Figure 2). This takes 6 runs in total. All three precisions are fed the *same* fp16-generated reference text. Every comparison between precisions is therefore made at identical positions, and differences cannot come from the quantized models generating different text.

Validation. The estimator is only useful if it agrees with real self-SD. Real self-SD reports *block* acceptance (drafts count only up to the first mismatch), so we convert the estimator’s per-position matches into an approximate block statistic by replaying the accept-until-first-mismatch rule with $\gamma=6$. We compare it with real self-SD runs at 8 exit layers for each precision on HumanEval. All 24 differences are below the 0.05 threshold we set in advance; the largest is 0.047 (Figure 3). This check covers the tested configurations only. Our main quantity, $\Delta\alpha$, does not need the block conversion: it compares per-position agreement between precisions at identical positions.

Setup.

- **Models:** facebook/layarskip-llama2-7B (Elhoushi et al., 2024; Touvron et al., 2023) ($L=32$); facebook/layarskip-llama3.2-1B for replication.
- **Tasks:** HumanEval (Chen et al., 2021) (code, zero-shot, pass@1 on 100 problems) and CNN/DailyMail (CNN/DM; Nallapati et al.,

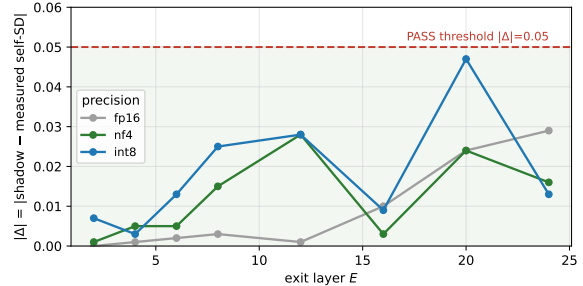


Figure 3: **The estimator agrees with real self-SD.** Absolute difference between estimated (replayed) and measured block acceptance, 3 precisions $\times$ 8 exit layers (HumanEval). All points are below the pre-set 0.05 threshold (max 0.047, INT8 at $E=20$).

2016) (summarization, one-shot, ROUGE-2).

- **Precisions:** fp16, INT8, and NF4 (our bitsandbytes configuration, §2). Greedy decoding throughout.
- **Acceptance:** a set of prompts per task, each continued greedily by the fp16 model to form the shared reference. We report per-layer means with 95% confidence intervals over prompts. The number of prompts was set from a pilot that measured the variance of paired fp16–NF4 acceptance differences in real self-SD runs.
- **Speed:** $E=8$, $\gamma=6$, up to 128 new tokens, batch size 1, mean over 100 prompts, on one NVIDIA RTX 4060 Ti (16 GB).

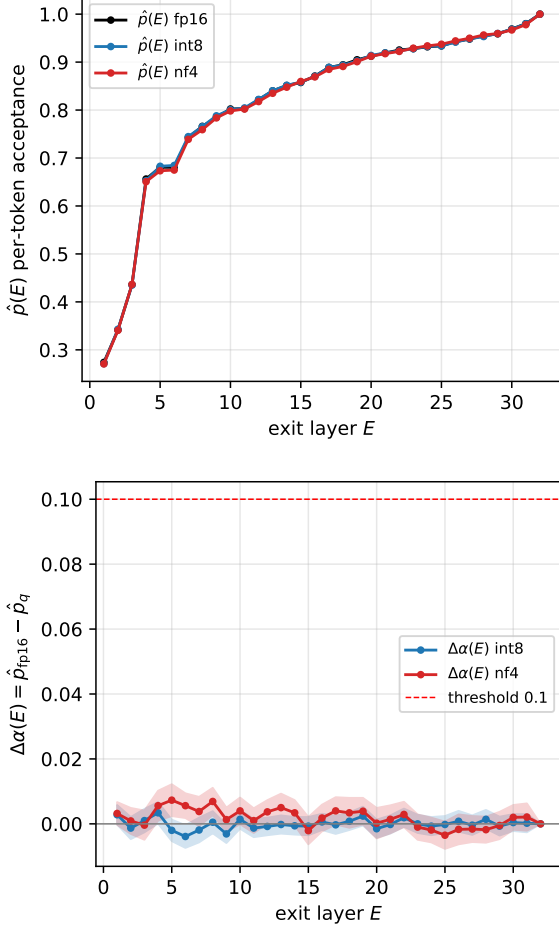


Figure 4: **Quantization barely changes acceptance** (HumanEval). *Top*: acceptance $\hat{p}(E)$ of fp16, INT8, and NF4 overlaps at every exit layer. *Bottom*: the erosion $\Delta\alpha(E)$ (shaded: 95% CI) stays near zero, far below the pre-set 0.1 threshold.

5 Acceptance Is Preserved

Finding: *quantization reduces acceptance by about one percentage point at most, at every exit layer.*

Criteria fixed in advance. To avoid reading patterns into noise, we decided before measuring what would count as erosion. Either of the following, evaluated on NF4, would have counted:

- (a) $\Delta\alpha(E) > 0.1$, with a 95% CI excluding 0, on at least half of the layers;
- (b) early layers eroding more than late ones: $\overline{\Delta\alpha}_{E \leq 8} - \overline{\Delta\alpha}_{E \geq 16} \geq 0.05$.

Results. Neither criterion is met (Figure 4). The acceptance curves of the three precisions lie on top of each other. The erosion is small but not always exactly zero: on HumanEval, NF4’s mean erosion

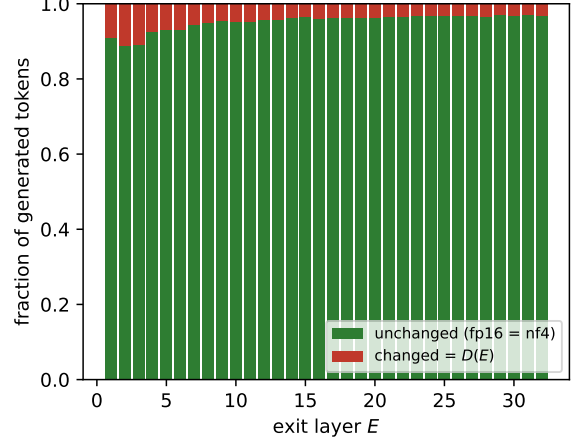


Figure 5: **Most predictions do not change.** Share of positions whose prediction changes under NF4 ($D(E)$, red) at each exit layer (HumanEval). The change peaks in early layers and shrinks with depth.

peaks at about 0.007 (at $E=5$), and the upper ends of its pointwise 95% CIs stay at or below about 0.012; on CNN/DM the erosion is at most 0.010. At a few early layers the interval excludes zero, so quantization has a measurable effect there, but it is an order of magnitude below the 0.1 threshold. A preliminary replication on Llama-3.2-1B shows the same pattern, with erosion again at most about 0.012 on both tasks.

Acceptance is not quality. Greedy self-SD reproduces the output of its verifier, so we score AR outputs and take output quality to be that of the quantized model. HumanEval pass@1 drops by 3 percentage points after quantization (from 0.25 in fp16 to 0.22), and ROUGE-2 on CNN/DM is essentially unchanged. In short, NF4 cuts peak GPU memory from ~ 15.5 GB to ~ 6 GB (Figure 1) and leaves self-SD acceptance nearly intact.

6 Where the Predictions Change

Finding: *quantization changes few predictions, mostly in early layers and on content tokens.*

$\Delta\alpha \approx 0$ does not mean that quantization changes nothing. To see what it does change, we compare the fp16 and quantized predictions at each layer directly:

$$D(E) = \text{mean}_t \mathbf{1} \left[\hat{y}_t^{E, \text{fp16}} \neq \hat{y}_t^{E, q} \right]. \quad (3)$$

$D(E)$ is the fraction of positions where the two models’ layer- E predictions differ.

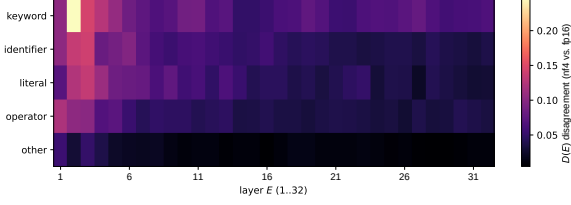


Figure 6: **Changes concentrate in early layers on content tokens.** $D(E)$ by token category and exit layer (HumanEval, NF4 vs. fp16). Content tokens: *keyword* (def, return), *identifier* (result, len), *literal* (0, 'abc'). Structural tokens: *operator* ((), +) and *other* (mostly whitespace).

By layer. Changes concentrate in early layers: $D(E)$ is $2.3\text{--}3.5\times$ higher for $E \leq 8$ than for $E \geq 16$. Even so, they are rare (Figure 5). At the early-layer peak about 10% of predictions change; at the final layer only 3.2% (HumanEval) and 2.2% (CNN/DM) do.

By token type. We assign each HumanEval reference token to one of five categories with simple lexical rules on its decoded text (Figure 6); subword fragments can be misassigned, so the categories are approximate. Keywords change the most: $D=0.118$ on average over $E \leq 8$, against 0.031 for whitespace and other tokens. In general, keywords, identifiers, and literals change more than operators and whitespace, and every category changes less with depth.

7 Why Acceptance Is Preserved

Finding: *quantization rarely breaks a draft–verify match, and the few matches it breaks are offset by new ones.*

How can acceptance stay the same while some predictions change? For each exit layer, we label every position by whether the draft matches the verifier under fp16 and under NF4 (Figure 7). The labels that matter are:

- *loss*: a match under fp16 that quantization breaks;
- *gain*: a mismatch under fp16 that quantization turns into a match.

For each prompt with N positions, the erosion equals $(\text{loss} - \text{gain})/N$, and $\Delta\alpha(E)$ averages this over prompts. So acceptance is preserved whenever losses and gains balance. What the analysis adds is that both are *small*. On average across layers, about 97% of fp16 matches survive quantization

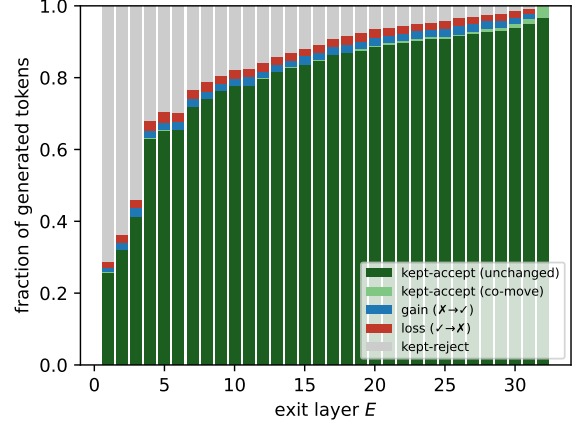


Figure 7: **Matches survive quantization.** Fate of each position’s draft–verify match from fp16 to NF4, per exit layer (HumanEval). *Kept-accept*: match under both, neither prediction changed. *Co-move*: match under both, with both predictions changing to the same new token. *Gain/loss*: a match is created/broken. *Kept-reject*: mismatch under both. Gain and loss are small and about equal.

Prec.	HumanEval			CNN/DM		
	AR	self-SD	Speedup	AR	self-SD	Speedup
fp16	17.6	27.5	$1.56\times$	14.7	27.1	$1.84\times$
INT8	7.2	12.0	$1.67\times$	7.0	13.5	$1.93\times$
NF4	18.5	14.8	$0.80\times$	17.6	16.2	$0.92\times$

Table 1: **Throughput in tokens/s.** AR vs. self-SD ($E=8$, $\gamma=6$, batch 1, 100 prompts, Llama-2-7B, RTX 4060 Ti). self-SD speeds up fp16 and INT8 but slows down NF4.

untouched, because neither the draft nor the verify prediction changes. The few losses are matched by a similar number of gains (difference at most 0.012 per layer).

8 Wall-Clock Speed Is Conditional

Finding: *self-SD speeds up fp16 and INT8 but slows down NF4; the timings point to NF4’s expensive multi-token verify step.*

Table 1 compares self-SD with AR decoding at the same precision. self-SD is faster for fp16 ($1.56\text{--}1.84\times$) and INT8 ($1.67\text{--}1.93\times$), but slower for NF4 ($0.80\text{--}0.92\times$). The similar agreement rates (§5) and the forward-pass timings below suggest that execution cost drives the difference.

Why NF4 loses, step by step. Figure 8 breaks the cost down. The center panel times a full 32-layer forward pass; the right panel times the parts of a self-SD block in a separate run.

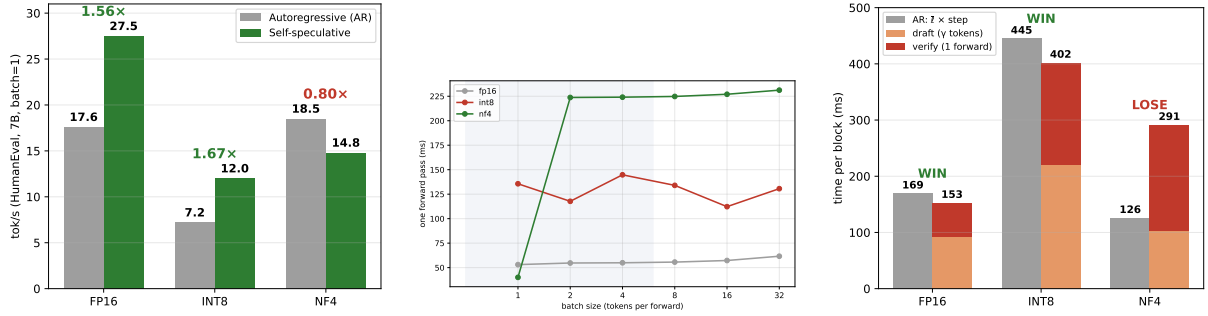


Figure 8: **Why NF4 self-SD loses (HumanEval).** *Left:* AR vs. self-SD throughput. *Center:* time for one forward pass vs. number of tokens processed. fp16 and INT8 are nearly flat, while NF4 is the fastest for one token but the slowest for two or more. *Right:* time per self-SD block (draft + verify) vs. AR producing the same tokens, from a separate block-timing run. Its implied speedups differ from Table 1, so we use it only for the composition of a block, not to quantify end-to-end speedup. NF4 drafts as cheaply as fp16, but its verify is expensive and its AR baseline is the cheapest.

- Drafting is cheap.** One might expect 4-bit dequantization to make drafting slow. It does not. For a single token, NF4 is the *fastest* precision: $\sim 40\text{--}45$ ms per forward pass, against ~ 53 ms for fp16 and $\sim 135\text{--}143$ ms for INT8 (center). Six NF4 drafts cost about as much as six fp16 drafts (right).
- Verifying is expensive.** The verify step pushes the whole block of $\gamma+1=7$ tokens through the remaining $L-E$ layers at once. For fp16 and INT8, a multi-token pass costs about the same as a single-token one, because decoding is limited by memory bandwidth (fp16 weights take ~ 47 ms to read at the GPU’s 288 GB/s, close to the measured 53 ms). For NF4, a full pass jumps from ~ 40 ms for one token to ~ 225 ms for two or more. This is consistent with how bitsandbytes has dispatched 4-bit layers: a fused 4-bit matrix–vector kernel for single-token inputs, and dequantization of the full weight matrices for multi-token inputs (Dettmers and bitsandbytes contributors, 2022). Newer releases add fused multi-token paths, so this cost is specific to the implementation we used.
- The block does not pay off.** In the block-timing run, an NF4 block takes more than twice as long as NF4 AR needs for the same tokens, while for fp16 and INT8 a block is cheaper than the matching AR time. The run’s absolute values do not match Table 1, so we read only this ordering from it.

INT8 shows the opposite case. Its decoding kernel is slow, but its verify costs about one decoding step,

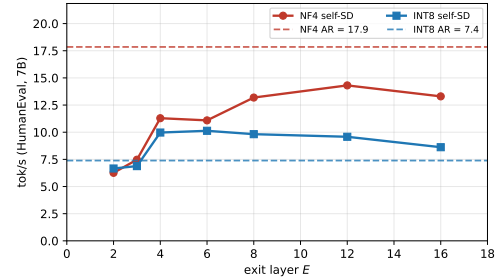


Figure 9: **No tested exit layer makes NF4 self-SD beat AR.** self-SD throughput vs. exit layer E (HumanEval, $\gamma=6$). NF4 peaks at $0.80\times$ of AR ($E=12$); INT8 beats AR from $E=4$ on. Measured in a separate run from Table 1: at $E=8$, self-SD throughput is $\sim 11\%$ (NF4) and $\sim 18\%$ (INT8) lower than in the table, so compare values within this figure only.

so self-SD still wins.

No tested exit layer rescues NF4. A deeper exit raises acceptance and shortens the multi-token part of the verify, but makes each draft more expensive. We sweep $E \in \{2, 3, 4, 6, 8, 12, 16\}$ with $\gamma=6$ on HumanEval (Figure 9). For NF4, every exit layer stays below AR; the best, $E=12$, reaches $0.80\times$. An adaptive per-token policy, which we did not test, would need $\sim 25\%$ more throughput than this best fixed layer just to break even. For INT8, self-SD beats AR from $E=4$ on, but the curve is flat, so tuning E adds little. We therefore expect the multi-token 4-bit kernel, rather than the exit policy, to be the main lever for NF4; batched 4-bit kernels such as Marlin (Frantar et al., 2025) are a natural next step.

9 Conclusion

We asked whether LayerSkip self-speculative decoding still works after 8- and 4-bit weight quantization. For *acceptance*, the answer is yes. Quantization lowers it by about one percentage point at most, at every exit layer, on both tasks (with a preliminary 1B check agreeing), because it rarely breaks a draft–verify match. For *speed*, the answer depends on the kernel. self-SD pays off when its total time per emitted token is lower than AR’s. In fp16 and INT8 the multi-token verify is cheap enough for this; in bitsandbytes NF4 it is not, and none of the exit layers we tried changes this.

Practical takeaways.

- Quantization changes draft–verify agreement only slightly, so quantized LayerSkip models keep self-SD’s acceptance without retraining the draft.
- Before enabling self-SD on a quantized model, compare its time per emitted token with AR at the same precision; a quick check is whether a multi-token forward pass is much slower than a single-token one.
- For 4-bit models, a faster multi-token 4-bit kernel is likely to matter more than a smarter exit policy.

Limitations

Speed results depend on the kernel. All timings use bitsandbytes on one GPU. Its INT8 decoding is unusually slow and its NF4 kernel is fast only for single tokens. The cross-precision comparison (“INT8 wins, NF4 loses”) reflects these kernels and may change with others, such as GPTQ or AWQ with Marlin. The acceptance results do not depend on kernel speed, although kernel numerics and computation dtype (bfloat16 for NF4, float16 for fp16) could slightly affect predictions.

One model family. Both models are LayerSkip-trained Llama models, and the 1B analysis is preliminary: its predictions may have been stored in a 16-bit integer format that cannot represent every Llama-3 token id, which the released code now avoids. Models without early-exit training, and other quantization methods, are untested.

Narrow evaluation. We measure greedy acceptance on two tasks with short generations and moderate sample sizes. We also estimate acceptance on

an fp16 reference text rather than on each quantized model’s own output; in the validation of §4, the observed discrepancy stayed below 0.05, but this was checked on HumanEval only, and the block-acceptance replay used for Figure 3 omitted the verifier’s extra token after a fully accepted block, so its block boundaries are approximate. The released code fixes this replay; recomputing the check is left for an update of this report. Sampling, longer generations, and larger models are untested.

No realized NF4 speedup. We did not test whether a faster multi-token 4-bit kernel actually turns NF4 self-SD into a speedup; that is a systems effort beyond this measurement study.

References

- Ramshankar Bhuvaneswaran and Handan Liu. 2025. BitSkip: An empirical analysis of quantization and early exit composition. *arXiv preprint arXiv:2510.23766*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Tim Dettmers and bitsandbytes contributors. 2022. bitsandbytes: 8-bit and 4-bit quantization primitives for PyTorch. <https://github.com/bitsandbytes-foundation/bitsandbytes>.
- Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, volume 35, pages 30318–30332.
- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient fine-tuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 36, pages 10088–10115.
- Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed A Aly, Beidi Chen, and Carole-Jean Wu. 2024. LayerSkip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12622–12642.
- Elias Frantar, Roberto L Castro, Jiale Chen, Torsten Hoefler, and Dan Alistarh. 2025. MARLIN: Mixed-precision auto-regressive parallel inference on large language models. In *Proceedings of the 30th ACM*

SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming, pages 239–251.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, pages 19274–19286. PMLR.

Ramesh Nallapati, Bowen Zhou, Cicero Dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.

Rishabh Tiwari, Haocheng Xi, Aditya Tomar, Coleman Hooper, Sehoon Kim, Maxwell Horton, Mahyar Najibi, Michael W Mahoney, Kurt Keutzer, and Amir Gholami. 2025. QuantSpec: Self-speculative decoding with hierarchical quantized KV cache. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*. PMLR.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shrutu Bhosale, and others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Hossein Entezari Zarch, Lei Gao, Chaoyi Jiang, and Murali Annavaram. 2025. DEL: Context-aware dynamic exit layer for efficient self-speculative decoding. *arXiv preprint arXiv:2504.05598*.

Yudi Zhang, Weilin Zhao, Xu Han, Tiejun Zhao, Wang Xu, Hailong Cao, and Conghui Zhu. 2025. Speculative decoding meets quantization: Compatibility evaluation and hierarchical framework design. *arXiv preprint arXiv:2505.22179*.

Juntao Zhao, Wenhao Lu, Sheng Wang, Lingpeng Kong, and Chuan Wu. 2025. QSpec: Speculative decoding with complementary quantization schemes. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 4779–4795.

A Additional Results

Comparison with fp16 AR. A practitioner may care less about beating AR at the same precision than about beating fp16 AR. Quantized self-SD beats fp16 AR only for NF4 on CNN/DM ($16.2/14.7 = 1.10\times$, 95% CI [1.05, 1.15]). All other quantized configurations remain slower than fp16 AR ($0.68\text{--}0.92\times$; Table 1).

INT8 transitions. The analysis of §7 gives the same picture for INT8, with even smaller changes: losses and gains differ by at most 0.004 per layer,

and 1.3% / 0.9% of final-layer predictions change on HumanEval / CNN/DM (NF4: 3.2% / 2.2%).

Draft cost. Measured directly, one NF4 draft step costs $0.42\times$ an INT8 draft step at every exit layer tested, confirming that dequantization does not make NF4 drafting expensive. Raising the bitsandbytes INT8 outlier threshold reduces INT8 decoding time by about 20% ($123 \rightarrow 99$ ms); this comparison alone does not isolate the source of the remaining latency.

Choosing the exit layer without fp16. Because $\Delta\alpha \approx 0$, a quantized model can pick its exit layer from its *own* acceptance curve, without an fp16 reference. In our measurements, choosing the exit layer from the quantized model’s own curve gave the same choice as using fp16’s curve; picking the fastest layer additionally requires measuring draft and verify costs.

B Reproducibility Details

Hardware. One NVIDIA RTX 4060 Ti (16 GB, 288 GB/s memory bandwidth); batch size 1 for all timings.

Models and quantization. [facebook / layerskip - llama2 - 7B](https://github.com/facebook/llama2-7B) and [facebook / layerskip-llama3.2-1B](https://github.com/facebook/llama3.2-1B), loaded with Hugging Face Transformers. fp16 runs use torch.float16. NF4 uses bitsandbytes 4-bit loading with double quantization and bfloat16 compute; INT8 uses bitsandbytes 8-bit (LLM.int8()) loading with default settings.

Decoding. Greedy decoding. self-SD uses $\gamma=6$ drafts per block; throughput runs use $E=8$ and up to 128 new tokens over 100 prompts. The exit-layer sweep uses $E \in \{2, 3, 4, 6, 8, 12, 16\}$. HumanEval is zero-shot; CNN/DM is one-shot.

Code. Measurement, analysis, and plotting scripts are available at <https://github.com/jiseungjeong/QuantShift>. The results in this report were produced with the code at tag report-v1, before the replay and storage fixes described in the Limitations section.