DRAFTING WITHOUT HEARING: WHY SPECULATIVE DECODING FAILS IN HIERARCHICAL SPEECH GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Speech models such as Qwen3-Omni represent each audio frame with a primary code followed by residual codes that refine its acoustic detail. The completed frame becomes part of the history used to predict the next frame. Speculative decoding offers a natural way to accelerate this process by verifying future proposals in parallel. On frozen Qwen3-Omni, scoring three positions costs only 23% more than scoring one, yet complete speculative generation is consistently slower than ordinary decoding. We trace the lost verification saving to generating each draft's remaining acoustic codes, a step we call *acoustic completion*. Those codes are needed to verify later proposals before the draft's acceptance is known. Preparing these inputs and completing output frames account for 81.7% of measured speculative model computation. Even with a free drafter, the remaining computation nearly matches ordinary decoding at the observed acceptance rate. Truncating acoustic feedback to reduce this cost distorts subsequent target predictions. A controlled intervention raises word error rate from 11.68% to 40.97% even though every output frame retains all its codes. Drafting complete acoustic groups transfers the same prediction task to the drafter. On Moshi, more accurate complete-group drafters increase the number of accepted code groups, yet their candidate-generation cost still prevents a speedup. These results identify acoustic completion as a cost that higher proposal accuracy alone does not eliminate: a draft may be cheap to verify, yet expensive to make verifiable. Under the conditions we study, ordinary decoding is the better policy.

1 INTRODUCTION

Autoregressive speech generators produce short audio frames in sequence. Speculative decoding could reduce sequential model calls by proposing several future frames and verifying them together. On Qwen3-Omni (Xu et al., 2025), the target model scores three positions in 1.61 ms, only 23% more than the 1.31 ms needed for one. Yet the complete speculative generator is slower than ordinary decoding. Where do the verification savings go?

We show how constructing complete acoustic inputs consumes the saving from parallel verification. In text generation, a proposed token directly specifies the input needed to verify the next proposal. The hierarchical speech generators studied here represent each frame with a primary code followed by residual codes that refine its acoustic content. The completed frame becomes the input for predicting the next. A proposed primary code therefore leaves that input incomplete. Before the target can check the second primary, the first draft must be supplied with its remaining acoustic codes, even though it has yet to be accepted (Figure 1). We call producing those codes *acoustic completion*.

Our contributions are:

1. **We identify the acoustic cost that nearly exhausts verification savings.** Preparing candidate inputs and finishing output frames account for 81.7% of the measured speculative model computation. Removing all draft model computation while keeping acceptance unchanged leaves the remaining cost close to that of ordinary decoding (Section 3.2).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2. **We isolate the quality loss caused by incomplete acoustic feedback.** In a feedback-only intervention, every emitted frame retains all sixteen codes. Using seven residuals instead of fifteen for the next-step input raises word error rate from 11.68% to 40.97% (Section 3.3).
3. **We show why higher complete-group acceptance still fails to yield a speedup.** We build drafters from Moshi's original weights to propose all codes needed for each step. Retaining more layers increases output per round under strict acceptance, but the added drafting cost prevents a speedup. Costs from earlier measurements predict how much output new configurations would need to break even; their measured output falls below that requirement (Section 3.5).

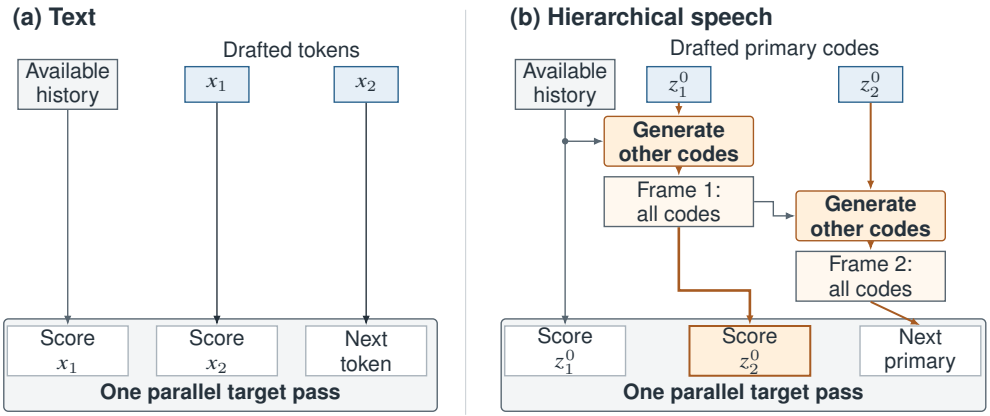


Figure 1: **Checking the second primary requires the first frame's remaining codes.** Text drafts directly specify the next verification inputs. Speech primaries z_1^0 and z_2^0 leave each frame's other codes missing. Completing frame 1 supplies the input for scoring z_2^0. To generate frame 2's remaining codes, the original generator also needs the hidden state produced when the target processes completed frame 1. Our primary-code drafter predicts that state. Frame 2 then supplies the next-primary input. All bottom scores belong to one parallel target pass; arrows show information dependencies. Text conditioning is omitted.

2 Background and Related Work

2.1 Generation and Verification

A speech codec encodes a short interval of sound, or frame, as discrete codes and decodes those codes back into sound. Residual vector quantization represents a frame with an initial code and successive refinement codes. Qwen3-Omni and Qwen3-TTS-12Hz use one primary and fifteen residual codes per frame (Xu et al., 2025; Hu et al., 2026). The temporal model advances from frame to frame, predicting one primary code at each step. A smaller autoregressive predictor generates the residuals from that primary and the temporal model's hidden state, its internal representation of the available history and text conditions. The code embeddings are summed to form the input to the next temporal step. A separate decoder turns the completed codes into sound.

In speculative decoding, a draft model proposes several tokens, and the target scores them together in a causally masked forward pass. Each position can attend only to the history and earlier proposals. Rejection sampling accepts consecutive proposals using draft and target probabilities, then supplies a replacement at the first rejection or an extra token after full acceptance (Leviathan et al., 2023). With the target's original conditional probabilities, this rule preserves its sampling distribution. The target must therefore evaluate each position using the proposed codes that precede it. For text, token IDs specify that history through embedding lookup. For the hierarchical speech models considered here, the preceding primary alone leaves its frame input incomplete.

Parallel verification is useful only after the acoustic inputs for the proposed future frames have been constructed. For the second draft, generating residuals with the original predictor requires the target hidden state reached after processing the completed first frame. Computing this state with the target

advances generation sequentially; predicting it instead moves the computation to the drafter and makes the resulting acoustic input approximate. Once these inputs are available, the target can score the proposed primaries in parallel.

Moshi likewise uses a temporal model to advance across steps and a smaller depth predictor to generate codes within each step (Défossez et al., 2024). Its audio streams are offset in time, so the code group fed back at one step contains codes from adjacent physical frames. The next temporal step consumes that completed group.

2.2 Prior Acceleration Methods

The speech generator's feedback determines what a proposal must contain. In the main single-stream configurations of prior speech-speculation studies, one proposed code specifies each new autoregressive input (Lin et al., 2025; Yanuka et al., 2026; Li et al., 2025a; Xie et al., 2026). Acceptance tolerances and groups of acoustically similar codes broaden the set of proposals that can be retained at a position (Lin et al., 2025; Yanuka et al., 2026). Hierarchical feedback introduces a further requirement: the proposed primary needs its remaining acoustic codes before later positions can be evaluated. By keeping emitted frames complete, we isolate the content errors caused by omitting residual feedback. Appendix F distinguishes the acceptance rules and the multi-codebook variant reported in prior work.

For text, speculation saves time when enough proposals are accepted to offset drafting and verification costs (Leviathan et al., 2023; Chen et al., 2026). Conditioning drafts on target-model features, proposing token blocks in parallel, and selecting proposals by confidence improve this tradeoff (Li et al., 2025b; Chen et al., 2026; Cheng et al., 2026). Whether speculation speeds up serving also depends on batch size and context length (Sadhukhan et al., 2025), and on expert activation during verification (Saxena et al., 2025). These methods operate on token sequences that already specify the target's future inputs. We identify a further failure mechanism in hierarchical speech: the measured cost of producing complete acoustic inputs consumes the saving even when target block verification remains efficient.

3 Evaluation

We compare the saving from block verification with the cost of acoustic completion. The subsequent experiments test whether reducing feedback, retaining more proposals, or drafting complete frames can recover a saving while preserving speech quality.

3.1 Experimental Setup

Our target is the mixture-of-experts (MoE) Talker in Qwen3-Omni-30B-A3B-Instruct, with 3.32B parameters including its 141.6M-parameter residual predictor. We keep these weights fixed and use an 81.8M-parameter learned drafter to compare ordinary decoding with two-frame speculation ($K = 2$). Both run in bfloat16 on two Hopper H200-class GPUs, with CUDA Graphs and the same asynchronous scheduling and streaming configuration.

Our learned drafter predicts future primaries and hidden states from available target features, the preceding complete frame, and text. The original residual predictor generates candidate codes from those predicted states. We supply either all fifteen residuals ($m = 15$) or the first seven ($m = 7$) before verification.

The target verifies primaries using these constructed histories. Accepted candidates retain their proposed residuals and receive any remaining codes. Every output frame has all sixteen codes, but the candidate residuals remain unverified. Both settings therefore use approximate acoustic feedback. Appendices A and B.1 give the training and continuation procedures.

The final evaluation uses 32 held-out English reading prompts after the configuration is fixed. Timing experiments cap output length; quality tests run until the model terminates naturally. The number of primary codes selected per round, N, includes accepted drafts and the target's replacement or extra code; we report its mean, $\mathbb{E}[N]$. Thus, $K = 2$ proposes two frames and can produce at most three outputs per round.

3.2 Verification Savings and Acoustic Cost

The complete speculative generator increases request latency at both tested concurrency limits (Table 1). Each setting processes the same ordered list of 32 prompts twice after warmup. Latency spans submission through speech completion, including text generation and waveform decoding. The speech stage processes one request at a time; the higher client concurrency allows more requests to queue.

Table 1: Mean Qwen3-Omni end-to-end request latency after warmup (seconds). Each setting completes the same 32 prompts twice, with a 64-primary-token output cap.

Concurrency limit	Ordinary decoding	Speculation ($K = 2, m = 7$)
1	0.592	0.894
32	7.653	13.403

A separate study of GPU computation asks how much of the verification saving remains after paying for acoustic generation. Ordinary decoding spends 1.31 ms on primary prediction and 5.24 ms generating residuals per frame. A three-position verification pass takes only 1.61 ms. Speculation also prepares candidate acoustic inputs and finishes the residuals of selected outputs. Together, these acoustic operations account for 81.7% of the measured speculative model computation. This share is not entirely incremental: ordinary decoding also completes output frames, while speculation additionally prepares candidate frames before their acceptance is known. The acoustic work spent preparing rejected drafts contributes no output.

Per-output accounting shows that the additional acoustic work nearly cancels the verification saving (Figure 2). With 1.77 codes selected per round, target scoring falls to 0.91 ms, while acoustic computation rises to 5.61 ms. The additional 0.37 ms of acoustic work almost cancels the 0.40 ms verification saving. The draft model adds another 0.34 ms. Total model computation is therefore 6.86 ms per code, against ordinary decoding's 6.55 ms. Excluding all draft model computation, with acceptance unchanged, leaves 6.52 ms per code, close to ordinary decoding. Acoustic completion has therefore consumed nearly all of the verification saving before any runtime overhead is added. Verification is cheap. Making the draft verifiable is not.

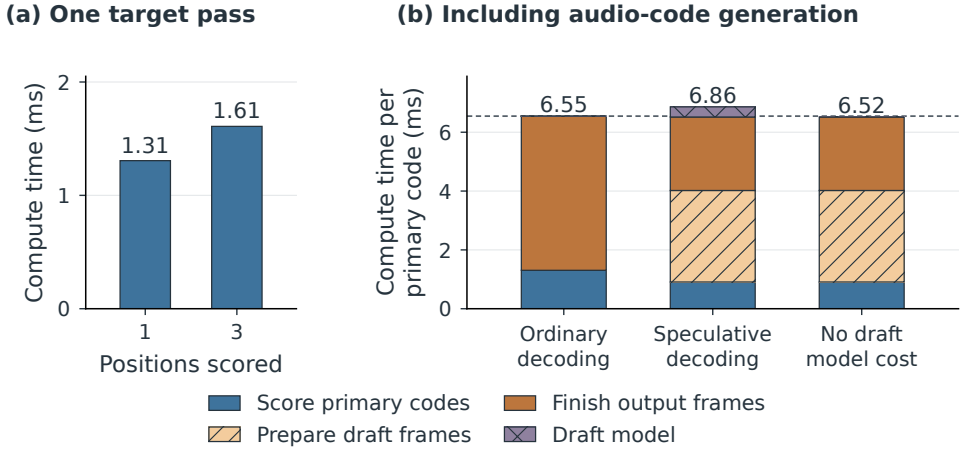


Figure 2: **Acoustic computation absorbs nearly all of the verification saving.** (a) A target pass scores three positions at little extra cost once its inputs are available. (b) Qwen3-Omni computation per primary code for $K = 2, m = 7$, obtained by dividing round costs by the observed 1.77 codes selected per round. The verification saving is almost offset by preparing candidate frames and completing output frames. The last bar excludes all draft model computation while retaining the measured acoustic work and acceptance. The dashed line marks ordinary decoding, whose residual generation is included under "Finish output frames." These GPU computation estimates exclude runtime waiting and waveform decoding. Table 6 reports both measurement orders.

3.3 Acoustic Feedback and Continuation Quality

An apparent way around this cost is to prepare only part of each draft frame and finish it after acceptance. The target, however, already uses the first draft's acoustic representation to score the second primary. We isolate the effect of removing residuals at five fixed generation points. Each paired comparison holds the primary code, target hidden state, preceding history, and sampled residual codes fixed, then supplies either all residual vectors or only the first m.

Removing residuals from the acoustic input changes the target's next-primary prediction. We quantify this change with total variation distance (TV), half the sum of absolute differences between the two probability distributions. Zero means identical distributions; one means no overlap. Table 2 shows the effect across all five points, with substantial variation between samples. Residuals shape the eventual sound as well as the next prediction; their omission changes the computation used to decide which drafts to accept.

Table 2: Effect of incomplete frame inputs on Qwen3-Omni's next prediction. At each of five generation points, sixteen sampled frames are evaluated with all fifteen residual vectors and with only the first m. TV measures the difference between the two next-primary distributions.

Generation	Mean TV			Largest TV
point	$m = 11$	$m = 7$	$m = 3$	$m = 7$
1	0.0313	0.0779	0.1962	0.196
2	0.0358	0.0657	0.3038	0.365
3	0.2025	0.3487	0.6030	0.511
4	0.0777	0.1426	0.2113	0.365
5	0.0484	0.1485	0.5244	0.655

We next test how feedback truncation alone affects speech content during ordinary sequential generation. The original model generates every primary and all fifteen residuals, and the waveform decoder receives complete frames. Only the residual embeddings fed into the next prediction change: all fifteen, or the first seven throughout generation. We measure word error rate (WER) using Whisper-base transcriptions and the text supplied to the speech generator (Radford et al., 2023). Reducing feedback alone raises WER by 29.29 percentage points on the 32 prompts, despite complete outputs (Table 3).

In speculative generation, by the time a frame is accepted, later predictions have already used its partial acoustic input. Adding the missing output codes does not update those predictions; doing so would require recomputing them with the complete frame. We evaluate this deferred-completion path at $m = 7$ and the full-depth path at $m = 15$ (Table 3). Both construct candidate residuals from predicted hidden states.

Table 3: Qwen3-Omni content errors on 32 paired prompts (audio demo). All settings terminate naturally and emit complete 16-code frames. The upper group isolates feedback truncation; the lower uses the fixed reference drafter. WER weights prompts equally. Increases are relative to ordinary decoding with full feedback; 95% intervals resample paired prompts.

Configuration	WER (%)	Increase (percentage points) [95% interval]
Ordinary sequential generation		
15-residual feedback	11.68	—
7-residual feedback	40.97	29.29 [14.49, 47.61]
Two-frame speculative generation		
$m = 15$	28.37	16.69 [2.71, 37.17]
$m = 7$	28.87	17.19 [12.83, 21.99]

At $m = 15$, all residual codes are present, but they come from predicted states and remain unverified. The quality gap persists at full residual depth, so residual truncation alone cannot explain it. Independent training repetitions reproduce this quality loss (Appendix A). The $m = 7$ generator also

produces more content errors and responds more slowly than ordinary decoding (Table 1). Despite the added errors, speculation remains slower than ordinary decoding.

3.4 Alternative Strategies

Acoustic completion could still pay off if each round produced enough accepted output. We examine ways to increase that output, then consider whether the drafter can avoid native completion by proposing complete frames directly.

Increasing accepted output. We retrain the drafter on its own generated histories to match the conditions it encounters online. These updates do not increase output per round ($\mathbb{E}[N]$: $1.709 \rightarrow 1.625 \rightarrow 1.600$; Table 4).

A low average acceptance rate could still hide profitable stretches of speech. Our selector therefore chooses where to speculate using signals available before candidate-frame preparation. The selected 5% of rounds average 1.902 primary codes, below the 2.122 needed to cover the separately calibrated single-GPU cost, including speech-stage runtime overhead (Appendix C).

Relaxing the acceptance rule can retain more proposals without making the drafter more accurate. To examine how the substituted primaries affect subsequent speech, we keep all residual generation with the target and test one frame at a time. Accepting any primary among the target's four most probable codes retains 67.2% of proposals. One of eight utterances then ends after its opening words, despite using the target's own residual predictor (Appendix B.5).

Predicting acoustic codes jointly. These approaches still rely on the original residual predictor to prepare candidate frames. A complete-frame drafter would replace those calls with its own predictions of the primary and residual codes. Acceptance would then depend on the complete acoustic proposal: an accepted primary can still be followed by residuals that the target rejects. Under code-by-code verification, each residual must survive before the next proposed frame can be retained. Whole-frame acceptance instead compares the target and draft probabilities of the complete code group (Appendix A.3). Either way, the drafter must predict both kinds of codes accurately while remaining cheap enough to preserve the verification saving.

The three trained models reach only 0.26–0.32 mean per-residual overlap with the target on training histories and 0.16–0.25 on validation histories, measured as one minus TV (Appendix A.3). They use 77.0M, 85.5M, and 92.8M trainable parameters to predict future primary and residual codes jointly. The variants with separate representations for each residual level reuse the original generator's embeddings and output layers; one also explicitly penalizes low residual overlap during training. These are individual residual distributions evaluated on recorded histories, before requiring acceptance of a complete proposed frame. Reusing these representations and adding the overlap penalty modestly improve training fit, while substantial mismatch remains on validation histories.

3.5 Cross-Model Evidence

To test whether complete proposals can cross the break-even boundary when acceptance is substantially higher, we turn to Moshi. The input to each new temporal step includes a *code group*: the text token and audio codes produced at the preceding step. The audio streams have different time offsets, so a group contains codes from adjacent physical frames. Proposing only the first audio code still requires one temporal target call per output step. Complete-group proposals provide the inputs needed for block verification.

Retaining more of Moshi's temporal layers raises output per round, but generation remains slower than ordinary decoding. We construct these drafters from the pretrained weights, varying the retained layers on separate calibration questions. All drafters use the original depth predictor. Under strict acceptance of complete groups, retaining 24 or 30 temporal layers yields 1.985 or 2.598 output groups per round, yet generation takes 1.704 or 1.525 times as long as ordinary decoding, respectively.

No tested drafter crosses the break-even boundary (Figure 3). To save computation, output per round must exceed the round's cost divided by ordinary decoding's cost per output group. We estimate this requirement for each calibrated drafter using earlier costs from the other evaluation questions;

acceptance is estimated on separate calibration questions. For every drafter, even the calibration estimate remains below the output required to cover computation; measured online output is lower still. For the 28- and 30-layer drafters, even the favorable cost estimate requires more than three outputs per round, exceeding the maximum for $K = 2$. Increasing acceptance alone cannot make these two drafters profitable at these costs (Appendix E.3).

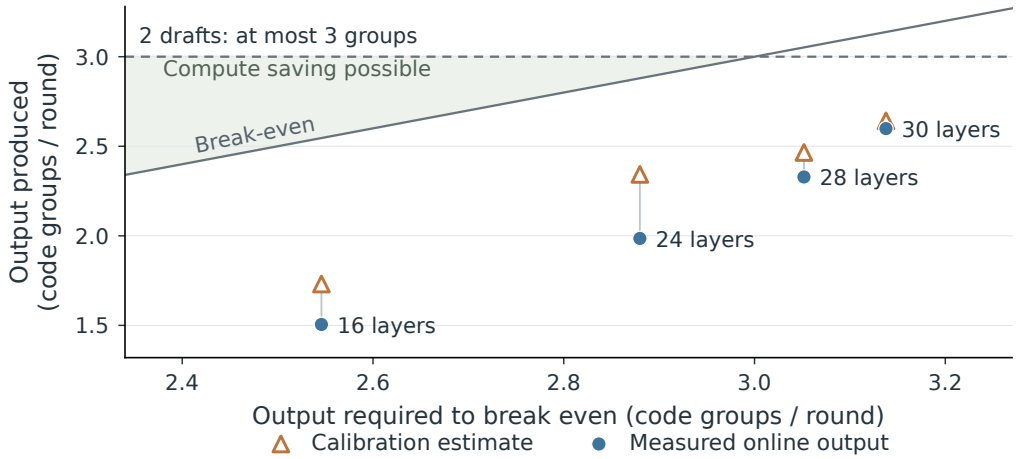


Figure 3: **Complete-group drafting raises both acceptance and the output needed to break even.** Triangles show calibration estimates; circles show measured online output. Each pair represents a Moshi drafter, labeled by retained temporal layers. A computation saving requires output above the diagonal. Horizontal positions use break-even estimates from earlier cost measurements. Online output is averaged over four questions and includes the target's replacement or extra group, so $K = 2$ allows at most three groups per round. Appendix E.3 gives the cost estimates.

Qwen3-TTS: obtaining the acoustic inputs. The Moshi drafters compute future states while producing their proposals. We use Qwen3-TTS to ask how much computation remains when these conditions are supplied for free, and how the cost changes when the target obtains them. We supply future primaries and hidden states recorded in a completed target run for free, assuming both drafts are accepted. Residual generation, block verification, and output completion in Qwen3-TTS then take 13.087 ms per frame, below ordinary decoding's 15.606 ms. Having the target obtain these states sequentially, while reusing its primary scores, costs 15.776 ms per frame. Both paths produce complete sixteen-code frames. The lower cost relies on an impractical online assumption: future target primaries and hidden states are available for free (Appendix D).

To our knowledge, training a low-cost drafter that predicts complete acoustic frames accurately enough to make speculation profitable remains an open challenge for hierarchical speech generators.

4 Limitations

The speech-quality experiments use English read speech. The speculative-generation comparison combines predicted hidden states, unverified residuals, and continued use of states computed during verification; their separate contributions to WER are unmeasured. WER measures transcribed content, and no multi-listener perceptual study was conducted.

The Qwen3-TTS controls measure computation under specified future-state conditions. Moshi uses four calibration questions and four evaluation questions at batch one.

5 Conclusion

In our GPU measurements, preparing candidate acoustic inputs and finishing output frames consume nearly all of the verification saving. Reducing feedback alone damages speech content despite complete output frames. Drafting complete frames transfers this work to the drafter, whose accuracy and cost must be evaluated together. Ordinary decoding achieves both lower response latency and fewer content errors than the tested approximate generator. In hierarchical speech, the cost of a draft includes supplying the acoustic history needed to verify what follows.

Reproducibility Statement

Appendices A–E describe the data, generation procedures, and statistical measures. The supplementary ZIP includes code, measurements, and recordings.

Audio demo. Extract the supplementary ZIP and open `index.html` in a browser. The page presents all paired recordings, including the feedback-only control, with reference texts and transcripts. Playback works offline without installation.

Reproduce results. Run `python3 reproduce.py` from the extracted folder to recompute statistics from saved measurements; no GPU or model download is required. Open `results.html` for the tables. `README.md` maps data to paper results; `EXPERIMENTS.md` gives requirements and commands for model runs. Trained drafter checkpoints are not included.

Ethics Statement

The study uses generated English read speech and publicly available utterance texts. It includes no human-subject study or collection of private speech. Experimental recordings can contain unintended words and should not be used as faithful readings of the supplied text. This work evaluates inference behavior and does not assess the broader safety of the underlying speech models.

Statement of AI use

OpenAI Codex and Anthropic Claude Code assisted with literature search, experiment and hypothesis development, prompt preparation, implementation, debugging, analysis, and manuscript drafting and revision. Source papers and recorded experiments provide the evidence for the claims. The authors are responsible for reviewing the manuscript, references, and artifacts before submission.

References

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.06036.

Xin Cheng, Xingkai Yu, Chenze Shao, Jiashi Li, Yunfan Xiong, Yi Qian, Jiaqi Zhu, Shirong Ma, Xiaokang Zhang, Jiasheng Ye, Qinyu Chen, Chengqi Deng, Jiping Yu, Damai Dai, Zhengyan Zhang, Yixuan Wei, Yixuan Tan, Wenkai Yang, Runxin Xu, Yu Wu, Zhean Xu, Xuanyu Wang, Muyang Chen, Rui Tian, Xiao Bi, Zhewen Hao, Shaoyuan Chen, Huanqi Cao, Wentao Zhang, Anyi Xu, Huishuai Zhang, Dongyan Zhao, and Wenfeng Liang. DSpark: Confidence-scheduled speculative decoding with semi-autoregressive generation. *arXiv preprint arXiv:2607.05147*, 2026. URL https://arxiv.org/abs/2607.05147.

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. Moshi: a speech-text foundation model for real-time dialogue. *arXiv preprint arXiv:2410.00037*, 2024. URL https://arxiv.org/abs/2410.00037.

Hangrui Hu, Xinfa Zhu, Ting He, Dake Guo, Bin Zhang, Xiong Wang, Zhifang Guo, Ziyue Jiang, Hongkun Hao, Zishan Guo, Xinyu Zhang, Pei Zhang, Baosong Yang, Jin Xu, Jingren Zhou, and Junyang Lin. Qwen3-TTS technical report. *arXiv preprint arXiv:2601.15621*, 2026. URL https://arxiv.org/abs/2601.15621.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023. URL https://arxiv.org/abs/2309.06180.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Bohan Li, Hankun Wang, Situo Zhang, Yiwei Guo, and Kai Yu. Fast and high-quality auto-regressive speech synthesis via speculative decoding. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1–5. IEEE, 2025a. doi: 10.1109/ICASSP49660.2025.10888194. URL https://arxiv.org/abs/2410.21951.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Advances in Neural Information Processing Systems*, volume 38, pp. 136737–136756, 2025b. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/c7b5a35ea98b62512a869c19ea7b03cb-Abstract-Conference.html.

Zijian Lin, Yang Zhang, Yougen Yuan, Yuming Yan, Jinjiang Liu, Zhiyong Wu, Pengfei Hu, and Qun Yu. Accelerating autoregressive speech synthesis inference with speech speculative decoding. In *Interspeech 2025*, pp. 5533–5537, 2025. doi: 10.21437/Interspeech.2025-2447. URL https://www.isca-archive.org/interspeech_2025/lin25h_interspeech.html.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. LibriSpeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5206–5210. IEEE, 2015. doi: 10.1109/ICASSP.2015.7178964. URL https://www.danielpovey.com/files/2015_icassp_librispeech.pdf.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 28492–28518. PMLR, 2023. URL https://proceedings.mlr.press/v202/radford23a.html.

Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian En-Hsu Yen, Avner May, Tianqi Chen, and Beidi Chen. MagicDec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/13f972adf12bdf886583d48cd528002f-Abstract-Conference.html.

Anish Saxena, Po-An Tsai, Hritvik Taneja, Aamer Jaleel, and Moinuddin Qureshi. Utility-driven speculative decoding for mixture-of-experts. *arXiv preprint arXiv:2506.20675*, 2025. URL https://arxiv.org/abs/2506.20675.

Hanke Xie, Xiaming Ren, Dake Guo, Ruonan You, Wenhao Li, Jingbin Hu, Guobin Ma, Huakang Chen, Kejie Xu, Rui Huang, Weiguo Tan, Xianrong Wang, and Lei Xie. FlashTTS: Fast streaming TTS with MTP acceleration and x-pred mean flow distillation. *arXiv preprint arXiv:2606.09141*, 2026. URL https://arxiv.org/abs/2606.09141.

Jin Xu, Zhifang Guo, Hangrui Hu, Yunfei Chu, Xiong Wang, Jinzheng He, Yuxuan Wang, Xian Shi, Ting He, Xinfa Zhu, Yuanjun Lv, Yongqi Wang, Dake Guo, He Wang, Linhan Ma, Pei Zhang, Xinyu Zhang, Hongkun Hao, Zishan Guo, Baosong Yang, Bin Zhang, Ziyang Ma, Xipin Wei,

Shuai Bai, Keqin Chen, Xuejing Liu, Peng Wang, Mingkun Yang, Dayiheng Liu, Xingzhang Ren, Bo Zheng, Rui Men, Fan Zhou, Bowen Yu, Jianxin Yang, Le Yu, Jingren Zhou, and Junyang Lin. Qwen3-Omni technical report. *arXiv preprint arXiv:2509.17765*, 2025. URL https://arxiv.org/abs/2509.17765.

Moran Yanuka, Paul Dixon, Eyal Finkelshtein, Daniel Rotman, and Raja Giryes. Principled coarse-grained acceptance for speculative decoding in speech. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2026. URL https://arxiv.org/abs/2511.13732.

Peiqi Yin, Jiangyun Zhu, Han Gao, Chenguang Zheng, Yongxiang Huang, Taichang Zhou, Ruirui Yang, Weizhi Liu, Weiqing Chen, Canlin Guo, Didan Deng, Zifeng Mo, Cong Wang, James Cheng, Roger Wang, and Hongsheng Liu. vLLM-Omni: Fully disaggregated serving for any-to-any multimodal models. *arXiv preprint arXiv:2602.02204*, 2026. URL https://arxiv.org/abs/2602.02204.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft & verify: Lossless large language model acceleration via self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11263–11282, 2024. doi: 10.18653/v1/2024.acl-long.607. URL https://aclanthology.org/2024.acl-long.607/.

A Data and Training

A.1 Datasets

We use 200 training prompts, 24 validation prompts, and 32 reserved test prompts. Training combines 185 utterance texts from LibriSpeech `train-clean-100` (Panayotov et al., 2015) with fifteen existing English development prompts. Validation and test texts come from the same public training partition. No prompts are shared between splits; entries with source metadata also have disjoint speakers and chapters.

Qwen3-Omni generates the training speech by reading these texts. We use the validation set for model development and evaluate the reserved test set only after fixing the model and generation settings.

A.2 Primary-Code Drafter

The target speech generator is the Talker component of Qwen3-Omni-30B-A3B-Instruct. Its temporal model has twenty MoE layers, each selecting six of 128 experts, and its residual predictor has five Transformer layers. The full code generator has 3.32B parameters, including 141.6M in the residual predictor. Generator parameter counts include embeddings and output layers and exclude waveform decoders and upstream encoders. The drafter is a six-layer Transformer with 81.8 million trainable parameters. It receives hidden states from four Talker layers, the current complete audio frame, and available text features. The target's frozen output layer converts its predictions into primary-code probabilities. Predicted hidden states condition the original residual predictor. Prediction of the second frame also uses the first proposal's primary and initial residual codes.

Training combines total variation and cross-entropy against target probabilities, with residual-prediction and acceptance-confidence objectives. It begins with target-generated histories and progressively substitutes the drafter's own preceding predictions. Two further updates use target probabilities on histories generated with the drafter. Table 4 reports their online validation results.

Table 4: Mean primary codes selected per round during generation on 24 validation prompts. Each row evaluates one trained model; 95% intervals resample prompts.

Training stage	$\mathbb{E}[N]$	95% interval
Before retraining on generated histories	1.709	$[1.662, 1.754]$
After retraining round 1	1.625	$[1.574, 1.679]$
After retraining round 2	1.600	$[1.544, 1.652]$

The full-depth quality gap persists across independent training runs. We repeat the initial distillation and training on the drafter's own preceding predictions with three independent initializations. At $K = 2, m = 15$, these models achieve online $\mathbb{E}[N]$ of 1.743, 1.801, and 1.769 on the same 32 test prompts. Their WERs are 23.47%, 27.05%, and 28.12%, against the ordinary-decoding reference of 11.68%. All 96 generations terminate naturally; each paired WER increase has a 95% prompt-bootstrap interval above zero.

A.3 Joint-Code Drafters

Three auxiliary models predict primary and residual codes jointly. The first uses two passes of a masked-block Transformer to predict two frames together, followed by a causal recurrent decoder. It shares embeddings and output weights across residual codebooks. The other two retain the temporal drafter and use a recurrent residual decoder with the original generator's separate codebook embeddings and output layers. The third also learns a separate projection for each codebook and adds a negative-log-overlap loss, which penalizes poorly matched residual distributions. These alternatives test parallel block prediction, codebook-specific representations, and stronger penalties for low-overlap positions. We train each model for eight epochs on target-generated histories, followed by four epochs with increasing use of its own preceding predictions.

We compare target probabilities p and draft probabilities q on the same recorded histories. Table 5 averages their overlap, $1 - \mathrm{TV}(p, q)$, over the first eleven residual codebooks for which target probabilities were recorded. The two future frames are reported separately. This evaluates individual residual distributions under the recorded conditions.

Table 5: Residual-distribution overlap, $1 - \mathrm{TV}(p, q)$, for joint-code drafters. A value of 1 denotes identical target and draft probabilities. All models are evaluated on the same recorded histories.

Model	Trainable parameters	Training histories Frame 1	Training histories Frame 2	Validation histories Frame 1	Validation histories Frame 2
Shared codebook representations	77.0M	0.269	0.258	0.170	0.162
Separate codebook representations	85.5M	0.304	0.291	0.252	0.248
Separate representations + projections	92.8M	0.319	0.305	0.232	0.227

Fix the history generated so far, s, and list proposed primary and residual codes in generation order as $x_1, x_2, \ldots$. Write $p_i = p(x_i \mid s, x_{<i})$ and $q_i = q(x_i \mid s, x_{<i})$ for the target and draft probabilities of the same proposed code under the same preceding codes. Standard code-by-code rejection sampling accepts that code with probability $\min(1, p_i/q_i)$ (Leviathan et al., 2023). With independent acceptance draws, the probability that the first r proposals all survive is exactly

$$\begin{aligned} S_r(s) &= \mathbb{E}_{x_{1:r} \sim Q(\cdot|s)} \left[\prod_{i=1}^{r} \min\left(1, \frac{p_i}{q_i}\right) \right] \\ &= \sum_{x_{1:r}} \prod_{i=1}^{r} \min(p_i, q_i). \end{aligned} \tag{1}$$

Each probability is evaluated along the candidate's own prefix. The code distributions may depend on all earlier proposed codes. For two drafted frames with L codes each, the expected number of complete output frames per round is $1+S_L(s)+S_{2L}(s)$, including one target-completed replacement or extra frame and before end-of-speech trimming. A rejection within the first frame discards the later candidates that used it as history. The recorded-history averages in Table 5 measure local distribution fit; evaluating Equation 1 requires probabilities on the sampled candidate prefixes.

For whole-frame acceptance, let $P(\cdot \mid s)$ and $Q(\cdot \mid s)$ be the joint distributions over a frame $F = (x_1, \ldots, x_L)$. The acceptance probability is

$$\begin{aligned} A_{\text{frame}}(s) &= \mathbb{E}_{F \sim Q(\cdot|s)} \left[\min\left(1, \prod_{i=1}^{L} \frac{p_i}{q_i}\right) \right] \\ &= 1 - \mathrm{TV}(P(\cdot \mid s), Q(\cdot \mid s)) \;\geq\; S_L(s). \end{aligned} \tag{2}$$

The inequality follows by comparing the two acceptance probabilities on each proposed frame. Whole-frame acceptance combines the likelihood ratios before applying the acceptance test; code-by-code acceptance applies the test at every position. The Moshi experiment uses the whole-group rule (Appendix E).

Scoring proposed residuals is cheap compared with producing them. Given candidate codes and conditioning states, the frozen residual predictor can score all fifteen residual positions in one causal pass. An isolated Qwen3-Omni measurement uses fixed synthetic inputs for two sequences with two candidate frames each, in bfloat16 with CUDA Graphs on an H200-class GPU. Scoring all four candidates together takes approximately 0.85 ms, compared with 10.8 ms for fifteen sequential scoring passes on the same inputs. This computation evaluates supplied candidates; producing their codes and conditioning states remains the drafter's task.

B Generation and Measurement

B.1 Speculative Generation

Each round begins with the last selected frame. The drafter proposes two future primary codes and hidden states. The original residual predictor generates the first m residuals for each candidate. The

target scores both primaries and a possible extra code in one pass over the selected frame and the two prepared candidates.

Rejection sampling selects consecutive primary proposals. Accepted candidates keep their proposed residuals and receive any remaining codes. Replacement and extra frames obtain complete residuals from the hidden states produced during verification. All output frames therefore have complete code groups.

The target continues from the attention states computed during verification, which reflect the candidate inputs used in that pass. Adding the missing output codes does not update those states. With $m = 15$, all residuals are present before verification, but they were generated from predicted hidden states and remain unverified after primary-code acceptance. The complete-group Moshi experiment uses joint acceptance instead (Appendix E).

B.2 Performance Measurement

Serving uses a vLLM-Omni fork with vLLM 0.25.0 (Kwon et al., 2023; Yin et al., 2026) on two Hopper H200-class GPUs in bfloat16. The text stage occupies one GPU; speech-code generation and waveform decoding share the other. Both compared implementations retain CUDA Graphs, asynchronous scheduling, and streaming chunk transfer. The text-generating Thinker stage uses greedy decoding; primary speech codes use temperature 0.9, top-$k = 50$, and a repetition penalty of 1.05.

For Table 1, each configuration completes the same ordered 32-prompt list twice, with speech-stage capacity one and client concurrency limits of 1 and 32. Every request is included. These finite workloads drain completely; mean in-flight concurrency is approximately 1 and 16, respectively. Warm request latency spans submission through completion. The two methods use the same GPU pair and produce matching text for each prompt. Timing excludes profiling.

The separate component study measures GPU model operations. We sum component medians, weight output completion by observed acceptance, and divide by mean primary codes selected per round. That count includes the target's replacement or extra code before end-of-speech trimming. Waveform decoding and runtime overhead are excluded. Figure 2 uses the measurement order more favorable to speculation; both orders are reported below.

Table 6: GPU computation for $K = 2, m = 7$, in ms per round. Each column uses a different measurement order. Finishing output frames is weighted by observed acceptance. Waveform decoding and runtime overhead are excluded.

Operation	Baseline-first	Speculation-first
Draft model	0.609	0.609
Prepare draft frames	5.596	5.508
Score primary codes	1.604	1.609
Finish output frames	4.571	4.424

B.3 Cost without the Draft Model

Table 7 removes all draft model computation while leaving acceptance and the other measured costs unchanged. Generating candidate residuals, verification, and completing outputs remain in the calculation.

Table 7: Computation if the draft model were free, with acceptance unchanged. Costs are milliseconds per selected primary code and exclude runtime overhead and waveform decoding.

Measurement order	Codes/round	Ordinary	Speculation	Draft model free
Baseline first	1.6842	6.539	7.350	6.989
Speculation first	1.7706	6.547	6.862	6.518

B.4 Speech Quality

All configurations in the speech-quality comparison terminate naturally on the 32 test prompts. Whisper-base transcribes each recording (Radford et al., 2023); the reference is the text supplied to the speech generator. Before computing WER, we lowercase both texts and replace punctuation other than apostrophes with spaces. Prompts receive equal weight. Confidence intervals resample paired prompts.

The feedback-only control uses ordinary sequential generation. Every frame's full residual sequence is sampled by the original predictor; the intervention restricts the embedding sum passed to the next temporal step. It applies at every step, while all sixteen codes are retained for waveform decoding. There is no learned future-state prediction or speculative acceptance in this comparison. Seven-residual feedback increases WER on 21 prompts, leaves five unchanged, and improves six.

The mean number of primary codes selected per round, averaged equally across prompts, is 1.775 $[1.734, 1.812]$ for $m = 15$ and 1.713 $[1.672, 1.755]$ for $m = 7$. The direct WER difference ($m = 7$ minus $m = 15$), expressed as a fraction, is $0.0050\,[-0.1998, 0.1556]$. Brackets give 95% confidence intervals.

B.5 Relaxed Primary-Code Acceptance

The eight-prompt substitution test accepts a proposed primary if it is among the target's four most probable codes. Otherwise, it samples from the target. The target generates all residuals from the selected primary and its current hidden state. Each round emits one frame, isolating primary substitution from residual approximation.

The rule accepts 67.2% of proposals. Mean WER, expressed as a fraction, changes from 0.2737 to 0.3888, with paired increase $0.1152\,[-0.0187, 0.2710]$. Four utterances worsen, three are unchanged, and one improves. For *I am sure Rosalind will be safe with you*, the output lasts 0.297 seconds and is transcribed as *I am.*... This test uses lowercase, whitespace-tokenized text for WER.

C Selective Speculation

The learned selector reads available target features, the current primary, and visible text before generating drafts or residuals. Its inputs exclude future codes and acceptance outcomes. Evaluation uses cross-validation with prompts kept separate between training and evaluation in each fold.

The cost calibration measures speech-stage elapsed time with all three serving stages on one GPU and one active request. CUDA Graphs and asynchronous serving remain enabled. Including runtime overhead, ordinary decoding costs 11.787 ms per output and speculation costs 25.018 ms per round. This is a separate calibration from the two-GPU component measurements in Figure 2.

At these fixed costs, a selected round must average more than $25.018/11.787 = 2.1225$ primary codes to save time before paying for selection. Selecting 5% of recorded rounds yields 1.902 codes per round, below that requirement. A hypothetical selector given the future acceptance outcome could select only rounds in which both drafts are accepted and obtain a positive saving. The learned selector does not identify them accurately enough. This calculation holds recorded histories and per-round costs fixed.

D Qwen3-TTS Computation Controls

Qwen3-TTS-12Hz-0.6B-CustomVoice generates one primary and fifteen residual codes per frame (Hu et al., 2026). Its code-generation checkpoint contains 905.8M parameters, including the text embeddings and the 141.6M-parameter residual predictor. We use the official model with bfloat16, batch one, and CUDA Graphs on one Hopper H200-class GPU. Sampling uses temperature 0.9, top-$k = 50$, and a primary-code repetition penalty of 1.05. We record sixteen states from ordinary generation on eight English texts. Configurations are interleaved across three timing repetitions.

Table 8 compares how future acoustic conditions are obtained while producing three complete output frames. With recorded future primaries and hidden states supplied for free, we measure residual

generation, block verification, and output completion with both drafts assumed accepted. These future values are recorded from a completed target run before timing. An online drafter would have to compute or predict them; that work is excluded from this control. The target-computed control obtains successive hidden states through the original model and reuses the primary scores generated along the way. At full residual depth, this control executes the original sequential generation and supplies a cost reference for the calculation with free future conditions. The free-condition control holds recorded hidden states fixed while resampling residuals. Timings exclude waveform decoding and serving overhead.

Table 8: Qwen3-TTS computation in ms per complete output frame, with two draft positions. The last row is an offline control: future primaries and hidden states come from a completed target run, their acquisition cost is excluded, and both drafts are assumed accepted. With $m < 15$, the target reads incomplete frames during verification; final outputs still contain all codes.

Execution	$m = 15$	$m = 11$	$m = 7$
Ordinary decoding	15.606	15.606	15.606
Target computes future states	15.776	13.628	11.609
Offline: future primaries and states supplied free	13.087	11.070	8.956

E Strict Speculation on Moshi

E.1 Model and Workload

Moshiko-BF16 has 7.69B parameters. Its temporal Transformer has 32 layers and its six-layer depth predictor contains 714.4M parameters including its embeddings and output heads (Défossez et al., 2024). Each step produces one text token and eight audio codes. Seven audio streams have a one-step delay, so the completed group contains codes from adjacent physical frames.

We measure active speech continuation after four completed spoken questions. Starting histories are selected before speculative outcomes are observed. Every continuation produces 64 complete code groups. The repetition-based comparison uses ten repetitions per input; the calibrated-drafter comparison uses five, with method order alternated. Each comparison includes its own ordinary-decoding baseline. All methods use audio temperature 0.8 and top-$k = 250$, text temperature 0.7 and top-$k = 25$, batch one, bfloat16, and CUDA Graphs on a Hopper H200-class GPU. Timing includes proposal generation, verification, rejection correction, and generation-state updates. Input encoding and waveform synthesis are excluded.

E.2 Repetition-based Proposals

The proposer repeats the preceding output. The single-code strategy proposes the first, semantic audio code and leaves text and the other codes to the target. The complete-group strategy proposes two groups, each containing one text token and eight audio codes. The target scores the proposed code sequences in parallel under a causal mask. A group's probability is the product of its conditional code probabilities.

Repetition assigns probability one to its proposed value d. Strict rejection sampling accepts it with target probability $P(d)$. On rejection, the target samples from its distribution excluding d; all retries are included in the measured time. Rejection discards the remaining drafts. Accepting both groups yields an extra target-generated group. Complete groups are accepted or rejected as units.

The first repeated complete group has zero probability after the target's sampling filters in 69.0% of rounds, averaging inputs equally. Complete-group speculation makes 0.968 temporal calls per output step; single-code speculation makes 1.000. These inexpensive proposals rarely supply a complete group that the target accepts.

Table 9: Strict Moshi decoding with repetition-based proposals. Elapsed time in ms for 64 generation steps, averaged over four inputs and ten repetitions each. Relative time is the ratio to ordinary decoding; 95% intervals resample inputs. $\mathbb{E}[N]$ counts generation steps per round. Input encoding and waveform decoding are excluded.

Strategy	Time (ms)	Relative time [95% interval]	$\mathbb{E}[N]$
Ordinary decoding	979.7	1.000	1.000
Draft one audio code	988.7	1.009 [1.008, 1.010]	1.188
Draft complete groups	1087.7	1.110 [1.085, 1.135]	1.036

E.3 Calibrated Complete-Group Drafters

To obtain more accurate proposals, we form drafters from the target's own pretrained weights by skipping temporal layers, following the self-speculative approach of Zhang et al. (2024). The complete original depth predictor is retained. The 16/24/28/30-layer drafters reuse 4.40/6.04/6.87/7.28B parameters, respectively, shared with the target; no auxiliary weights are trained. Four additional spoken questions provide sixteen two-step calibration windows. We preserve the first and last temporal layers and greedily remove intermediate layers to maximize estimated complete groups per round. The second group's probabilities use the drafter's own computation on the first group's codes. The selected layer sets are then fixed for the four evaluation questions.

During generation, each drafter computes its own future states and samples two complete groups. The target applies the whole-group rule in Equation 2. On rejection, we sample a group from P and retain it with probability $(1 - Q/P)_+$, producing the required normalized $(P - Q)_+$ correction distribution. All proposal computation, correction scoring and retries enter the elapsed time. Retaining all 32 temporal layers makes the drafter identical to the target and provides a full-model accuracy reference.

Cost of candidate generation. For a favorable fixed-cost estimate, we count candidate generation, verification, state updates, and one target-generated output group per round. We omit additional correction work and other runtime overhead. The output-generation cost comes from the full-model reference in the same comparison. Dividing this round cost by ordinary decoding's device time per step gives the required $\mathbb{E}[N]$ in Table 10.

Table 10: Complete-group self-speculation on Moshi with independently calibrated layer selection. Four evaluation inputs, five interleaved repetitions each; 95% intervals resample inputs. Ordinary decoding takes 994.6 ms for 64 steps. Required $\mathbb{E}[N]$ is the favorable cost estimate described in the text. Relative elapsed time above one denotes a slowdown.

Temporal layers retained	Draft cost (ms/round)	Actual $\mathbb{E}[N]$	Required $\mathbb{E}[N]$ (est.)	Relative elapsed time [95% interval]
16	21.97	1.505	2.543	1.899 [1.798, 2.002]
24	27.11	1.985	2.874	1.704 [1.639, 1.770]
28	29.67	2.329	3.038	1.588 [1.557, 1.621]
30	30.85	2.598	3.111	1.525 [1.497, 1.546]
32 (full model)	32.27	3.000	3.205	1.068 [1.054, 1.083]

The 30-layer drafter increases output to 2.598 groups per round, compared with 1.050 for repetition in the same run. Its generation cost raises the output needed to break even to an estimated 3.111 groups per round, above the three-output maximum for $K = 2$. Retaining fewer layers reduces the cost and acceptance together; all selected configurations remain slower than ordinary decoding.

After subtracting all measured GPU time for generating replacement and extra groups, the 30-layer drafter still requires 7.3% more GPU computation than ordinary decoding on average, and 5.4–10.3% more across the four inputs. The deduction includes repeated correction draws and the mandatory first draw. This subtraction holds proposals, acceptance and visited histories fixed.

Cost transfer across drafters. We retrospectively transfer cost profiles from earlier drafters that retained the leading temporal layers. Each evaluation question uses costs from the other three questions. The 16- and 24-layer configurations use profiles at the same depth; the 28- and 30-layer configurations interpolate between the earlier 24- and 31-layer profiles. Acceptance estimates use the four separate calibration questions.

The predicted requirements are 2.546, 2.880, 3.052, and 3.138 groups per round. Requirements computed from direct GPU timing of the calibrated configurations appear in Table 10. The question-averaged predictions agree with these measurements within 0.9%; the largest error for an individual held-out question is 1.95%.

F Acceptance Rules in Related Speech Work

Speech-token acceptance criteria include target-token matching and top-ranked alternatives (Li et al., 2025a), a probability bias (Lin et al., 2025), and rejection sampling over overlapping groups of similar alternatives (Yanuka et al., 2026). In the last case, the exactness guarantee applies to the group distribution at the current history; the retained draft token represents the accepted group. These groups contain alternatives for one position, whereas the acoustic groups in our experiments contain several codes needed together for the next input.

VADUSA's main HuBERT experiments include verification with a fixed target and no added tolerance (Li et al., 2025a). Its EnCodec variant combines eight codebooks with target tuning and relaxed acceptance, but leaves the codebook feedback order unspecified. The streaming speech-token experiments with trained future-token prediction also leave the precise acceptance rule unspecified (Xie et al., 2026).