Conformal Sparsification for Bandwidth-Efficient Edge-Cloud Speculative Decoding

Payel Bhattacharjee* **Fengwei Tian*** **Meiyu Zhong*** **Guangyi Zhang†**

Osvaldo Simeone‡ **Ravi Tandon§**

Abstract

Edge–cloud speculative decoding (SD) accelerates inference by having a cloud-based large language model (LLM) that verifies draft tokens generated by a resource-constrained small language model (SLM) at the edge. A central bottleneck is the *limited bandwidth of the edge–cloud link*, which necessitates efficient compression of draft token distributions. We first derive an information-theoretic bound that decomposes the token resampling rate into contributions from SLM–LLM distribution mismatch and from quantization distortion. Guided by this analysis, we propose the *Sparse Quantize-and-Sample SD (SQS-SD)* framework, which exploits distributional sparsity through structured sparsification and lattice-based quantization. Within this framework, K*-SQS* applies fixed top-K truncation, while *C-SQS* adaptively adjusts the retained token set via online *conformal prediction* to ensure bounded deviation from the dense distribution. Empirical results confirm that both approaches improve end-to-end latency and resampling rate in complimentary operating regimes.

1 Introduction

As edge–cloud deployment of large language models (LLMs) continues to grow, efficient inference has become a critical challenge. Techniques such as model quantization and knowledge distillation [16] are commonly employed to reduce model size and computational cost. However, these approaches often degrade performance by discarding fine-grained statistical information captured by larger models. A promising alternative is *speculative decoding* (SD) [12], where a small language model (SLM) generates multiple draft tokens that are then verified in parallel by a large LLM. This collaboration between a fast draft model and an accurate verifier boosts token throughput and lowers latency compared to standard autoregressive decoding.

In this paper, we focus on enabling bandwidth-efficient edge–cloud SD. Specifically, we assume a SLM runs on an edge device while a more powerful LLM runs on a cloud server in tandem. A central challenge in this setup is the limited communication bandwidth between edge and cloud, which makes it imperative to compress the information (such as token probability distributions) sent over the uplink. Recent studies emphasize the importance of minimizing edge–cloud communication: for example, reference [8] proposed a hybrid local–cloud inference scheme with selective offloading

*Equal Contribution. Department of Electrical and Computer Engineering; University of Arizona; Tucson, AZ, USA. {payelb,fengtian,meiyuzhong}@arizona.edu

†College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China. zhangguangyi@zju.edu.cn

‡Department of Engineering; King's College London; London, UK; osvaldo.simeone@kcl.ac.uk

§Department of Electrical and Computer Engineering; University of Arizona; Tucson, AZ, USA. tandonr@arizona.edu

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: AI and ML for Next-Generation Wireless Communications and Networking (AI4NextG).

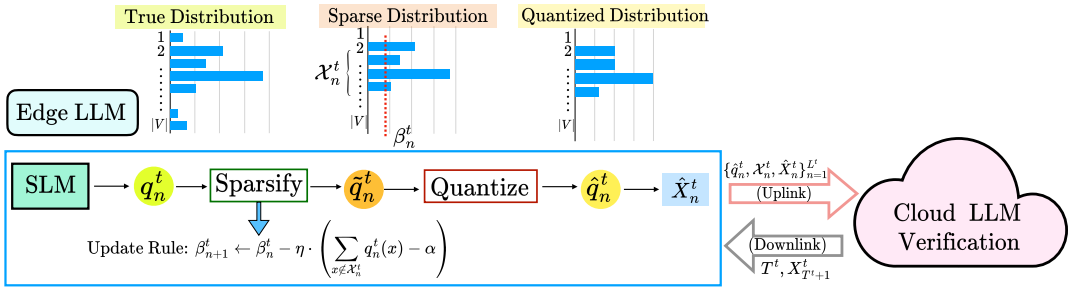

Figure 1: Illustration of *sparse quantize-and-sample (SQS) framework* for edge-cloud speculative decoding for efficient LLM inference. The edge device adaptively sparsifies and quantizes the SLM's next-token distribution with an updated threshold with a principled online update rule based on online conformal prediction.

of tokens; the work in [14] exploits uncertainty estimation to skip unnecessary uplink transmissions; and [15] explored token-level sparsification via an attention-based thresholding mechanism to reduce transmissions. Quantization is a key component of the SD pipeline for reducing communication. In the quantize-and-sample (QS) strategy introduced in [22], the edge first quantizes the SLM's token probability distribution and then samples from the quantized distribution, ensuring alignment with the cloud LLM's output while also reducing the uplink payload.

QS alleviates the distribution mismatch, but communication bottlenecks can still persist due to the high dimensionality of token distributions. Importantly, prior work has shown that SLM next-token distributions are inherently sparse, with most of the probability mass concentrated in a small top-K subset of the vocabulary [6, 9, 13], while the long tail of low-probability tokens contributes negligibly. This sparsity – often exacerbated by "attention sinks" in Transformer models [19] – motivates the development of sparse extensions to the QS approach. Motivated by these observations, we propose *sparse QS* (SQS) SD, which extends the QS framework to exploit distribution sparsity.

Main Contributions: Our contributions can be summarized as follows:

- **Information-theoretical analysis:** We start by deriving an information-theoretic performance bound on the performance of SQS that decomposes the expected token rejection and resampling rate into contributions from the SLM–LLM distribution mismatch and from quantization distortion. This analysis, based on [20], offers insight into the trade-offs introduced by sparsification.
- **K-SQS:** We first present and analyze a basic SQS variant using top-K truncation, which allocates bits to high-probability tokens, improving quantization resolution and reducing uplink cost.
- **Conformal SQS (C-SQS):** By fixing the cardinality at K, K-SQS cannot adapt to variability in next-token distributions across contexts. In practice, the SLM distribution may have widely differing effective supports, making this approach inefficient. We therefore introduce a theoretically grounded alternative, conformal QS (C-SQS), which adaptively sets the truncation threshold per token using online conformal prediction [7]. This yields a data-dependent sparsity level, guided by our theoretical bound, that ensures bounded deviation from the full (dense) QS distribution.
- **Experimental validation:** We conduct extensive experiments demonstrating that SQS and C-SQS significantly reduce required bandwidth and end-to-end latency with negligible loss in accuracy. Our results validate the effectiveness of structured sparsification and provide quantitative guidance for navigating the latency–accuracy trade-off in edge–cloud LLM inference.

2 Sparsify-Quantize-and-Sample (SQS) Speculative Decoding

This section reviews edge-cloud speculative decoding (SD) and the quantize-and-sample (QS) method from [22]. We then extend QS with sparsification, introducing sparse QS (SQS) to further reduce bandwidth. We present an information-theoretic analysis of SQS performance, forming the basis for protocol design in the following sections. Finally, we introduce a simple instantiation of SQS based on top-k selection.

Speculative Decoding: Edge-cloud SD accelerates LLM inference by parallel verification of the tokens generated by a small draft model (SLM) at the edge via a larger target model (LLM) at the cloud [12]. SD operates in batches. At each batch t, the SLM takes the current prefix of accepted

tokens to generate L^t new tokens in an autoregressive manner. The n-th generated token, denoted as $\hat{X}_n^t$ is sampled from the SLM distribution q_n^t, where V is the size of the vocabulary $\mathcal{V} = \{1, ..., V\}$. The notation $q_n^t(\cdot)$ is an abbreviation for the conditional distribution $q(\cdot|C_n^t)$, where C_n^t represents the context of all previously generated tokens, including all prior batches. A similar notation will be used for the LLM distribution.

The draft token sequence $\{\hat{X}_1^t, \ldots, \hat{X}_{L^t}^t\}$ and the associated distributions $\{q_1^t, \ldots, q_{L^t}^t\}$ are conveyed to the cloud, which uses them via the target LLM for parallel verification. Specifically, to verify the n-th token $\hat{X}_n^t$ in the batch, the target LLM computes its corresponding conditional distribution p_n^t and accepts it if the inequality $q_n^t(\hat{X}_n^t) \leq p_n^t(\hat{X}_n^t)$ holds, and otherwise rejects it with probability $(1 - p_n^t(\hat{X}_n^t)/q_n^t(\hat{X}_n^t))$. Denoting T^t as the number of accepted tokens $\{X_n^t = \hat{X}_n^t\}_{n=1}^{T^t}$, with $T^t \leq L^t$, the cloud samples a new token $X_{T^t+1}^t$ from the adjusted distribution $\bar{p}_{T^t+1}^t(\cdot) \propto (\max(0, p_{T^t+1}^t(x) - q_{T^t+1}^t(x)))$ if $T^t < L^t$ and from distribution $p_n^t(\cdot)$ if $T^t = L^t$. This procedure is repeated across batches $t = 1, 2, \ldots$ producing a sequence of $T = \sum_t (T^t + 1)$ verified tokens $\{X_n^t\}_{n=1}^T$. These tokens follow the same distribution as those generated autoregressively by the cloud [12].

Quantize-and-Sample Speculative Decoding: In edge–cloud SD with a bandwidth-limited uplink channel from edge to cloud, the edge must compresses the edge-based SLM distribution q_n^t into a quantized form $\hat{q}_n^t$ prior to communicating them to the cloud. QS, introduced in [22], generates the draft tokens $\{\hat{X}_1^t, \ldots, \hat{X}_{L^t}^t\}$ using the quantized distributions $\{\hat{q}_1^t, \ldots, \hat{q}_{L^t}^t\}$. This has the key advantage that the resulting accepted tokens $\{X_n^t\}_{n=1}^T$ preserve the same distribution as for LLM-generated tokens, thus maintaining the key guarantee of SD.

Sparsify-Quantize-and-Sample Speculative Decoding: While QS reduces the communication bandwidth via quantization, further gains can be achieved by exploiting the inherent sparsity of draft token distributions. In fact, prior works [6, 9, 13] have shown that next-token probabilities are typically skewed, with most of the mass concentrated in a small subset. Building on this observation, we extend QS to allow for a preliminary sparsification step. We refer to the resulting family of procedures as SQS.

Specifically, as shown in Fig 1, the SLM probability distribution q_n^t is first sparsified into a distribution $\tilde{q}_n^t$ with support set $\mathcal{X}_n^t \subseteq \mathcal{V}$ encompassing $K_n^t \leq V$ terms, and then quantized into a distribution $\hat{q}_n^t$. By the properties of QS, the resulting protocol also guarantees that the accepted tokens have the same distribution as for the LLM in the cloud.

We adopt with sparse lattice quantization (SLQ) [18, 17], which maps the retained K_n^t probabilities onto a structured lattice within the probability simplex. SLQ is characterized by a resolution parameter ℓ_n^t, with a smaller ℓ_n^t achieves coarser quantization at reduced bit cost.

With any SQS protocol, the number of bits required to represent the quantized vector $\hat{q}_n^t$ are given by

$$b_n^t(K_n^t, \ell_n^t) = \tilde{b}_n^t(K_n^t) + \hat{b}_n^t(K_n^t, \ell_n^t), \tag{1}$$

where the first term, $\tilde{b}_n^t(K_n^t)$, represents the number of bits required to describe the subset $\mathcal{X}_n^t$, and the second term, $\hat{b}_n^t(K_n^t, \ell_n^t)$, is the number of bits needed to describe the non-zero elements in vector $\hat{q}_n^t$. The term $\tilde{b}_n^t(K_n^t)$ depends on the specific SQS scheme, while the second term is given by [18, 17]

$$\hat{b}_n^t(K_n^t, \ell_n^t) = \log_2 \binom{\ell_n^t + K_n^t - 1}{K_n^t - 1}. \tag{2}$$

Information-Theoretic Analysis of SQS: Building on the theoretical guarantees in [20], the following result establishes an upper bound on the average number of rejected tokens N_{rej}. As in [20], the number of rejected tokens, N_{rej}, represents here the number of tokens that are sampled at the edge and then rejected and resampled at the cloud. Note that there is at most one rejected and resampled token per batch. Appendix A.2 provides further discussion on this definition.

The bound presented below isolates two separate contributions to the average number of rejected (and resampled) tokens: (*i*) the intrinsic statistical mismatch between the SLM draft distribution and the LLM target distribution [20], and (*ii*) the distortion introduced by sparsification and quantization. We denote as $\text{TV}(p, q)$ as the total variation distance between p and q. For the proof of this result (Theorem 1), we refer the readers to the Appendix A.2.

Theorem 1. *Consider a sequence of T tokens $\{X_t\}_{t=1}^T$ generated by using an SQS protocol, with corresponding per-token subsets $\mathcal{X}_n$ of cardinality $K_n(\mathcal{X}_n)$ and resolution parameters ℓ_n for the tokens $n = 1, ..., T$. The expected number of rejected tokens can be upper bounded as*

$$\mathbb{E}[N_{\text{rej}}] \leq \underbrace{\sum_{n=1}^{T} \mathbb{E}_{\{X_t\}_{t=1}^{n-1} \sim p}\left[\text{TV}\big(q_n(\cdot \mid \{X_t\}_{t=1}^{n-1}),\, p_n(\cdot \mid \{X_t\}_{t=1}^{n-1})\big)\right]}_{\textit{SLM-LLM discrepancy}} + \underbrace{\sum_{n=1}^{T} \left(\alpha_n(\mathcal{X}_n) + \frac{K_n(\mathcal{X}_n)}{4\ell_n}\right)}_{\textit{SLQ-based distortion}}, \tag{3}$$

where the expectation $\mathbb{E}_{\{X_t\}_{t=1}^{n-1} \sim p}[\cdot]$ is with respect to tokens generated from the LLM model, and the notation

$$\alpha_n(\mathcal{X}_n) = \sum_{x \notin \mathcal{X}_n^t} \mathbb{E}_{\{X_t\}_{t=1}^{n-1} \sim p}[q(x \mid \{X_t\}_{t=1}^{n-1})] \tag{4}$$

represents the average total probability in the subset of dropped tokens (i.e., tokens not selected during sparsification).

Top-K Sparsify-Quantize-and-Sample Speculative Decoding: In the next we propose a novel instantiation of SQS. We start here with a simple approach, referred to as *K-SQS*, in which the subset $\mathcal{X}_n^t$ defining the support of the quantized distribution is selected by following the standard top-K selection rule for a fixed value of the hyperparameter K. Accordingly, the subset $\mathcal{X}_n^t$ encompasses the K terms $x \in \mathcal{V}$ in the vocabulary with the largest probabilities $q_n^t(x)$ under the SLM. For K-SQS, the overhead for representing the subset $\mathcal{X}_n^t$ is given by

$$\tilde{b}_n^t(K) = \log_2 \binom{V}{K}, \tag{5}$$

since there are $\binom{V}{K}$ possible subsets of dimension K. A performance analysis for K-SQS follows directly from the general result in Theorem 1. In particular, the SLQ-based distortion term in the upper bound (Theorem 1) is characterized by a fixed value $K_n(\mathcal{X}_n) = K$, while the value of the probability $\alpha_n(\mathcal{X}_n)$ varies across index n, as the total mass in the dropped tokens changes across token generation.

3 Conformal Sparsify-Quantize-and-Sample Speculative Decoding

Motivation for Adaptive Sparsification: By fixing the cardinality at K, K-SQS cannot adapt to variability in next-token distributions across contexts. In practice, the distribution q_n^t may have widely differing effective supports across indices t and n [11, 10], making this approach inefficient. For example, after the prompt "`The capital of France is`," the continuation is highly predictable ("`Paris`"), allowing aggressive sparsification—i.e., a small cardinality K_n^t for subset $\mathcal{X}_n^t$—without quality loss. In contrast, a context such as "`She opened the box and found`" admits more uncertainty in the next-token prediction, requiring a larger support set $\mathcal{X}_n^t$ with larger cardinality K_n^t.

Adaptive Thresholding: To overcome this limitation, this section introduces an adaptive thresholding strategy based on online conformal prediction [7, 1, 21]. In this scheme, referred to as *conformal SQS* (C-SQS), the support set is determined as

$$\mathcal{X}_n^t(\beta_n^t) = \{x \in \mathcal{V} : q_n^t(x) \geq \beta_n^t\}, \tag{6}$$

where the sequence of thresholds β_n^t is designed to control the average number of rejected tokens via the upper bound (1).

Specifically, the key idea behind C-SQS is to vary the threshold β_n^t in (6) in such a way so as to ensure that the term $\sum_{n=1}^T \alpha_n(\mathcal{X}_n)$ in the upper bound (1) does not grow too quickly with the number of tokens T. This design goal is motivated by the result in Theorem 1, which suggests that controlling this term provides a way to limit the number of rejections. We specifically aim at guaranteeing an upper bound of the form:

$$\frac{1}{T} \sum_{n=1}^{T} \alpha_n(\mathcal{X}_n) \leq \alpha + \frac{C}{T}, \tag{7}$$

where $\alpha \in (0, 1)$ is a target value, C is some arbitrary fixed positive constant, and index n runs over the accepted tokens. The requirement (7) ensures that the term in the upper bound (1) that is negatively

affected by sparsification, namely $\sum_{n=1}^{T} \alpha_n(\mathcal{X}_n)$, is asymptotically no larger than a target value α. The choice of hyperparameter α dictates the degree to which we wish C-SQS to be aggressive in sparsification. A larger α allows for a larger growth of the distortion due to sparsification, which in turn makes it possible to reduce the second term in the SLQ-based distortion in (1). The fixed value of K used in K-SQS in contrast, carries no operational significance in terms of the final performance of the algorithm, the hyperparameter α thus relates directly to the protocol's performance via Theorem 1, controlling the trade-off between sparsification distortion and bandwidth reduction.

In order to ensure (7), SQS adopts an update rule based on online conformal prediction [2], whereby the threshold is updated during token generation as follows:

$$\beta_{n+1}^t = \beta_n^t - \eta \cdot \left(\sum_{x \notin \mathcal{X}_n^t} q_n^t(x) - \alpha \right), \tag{8}$$

where η is the learning rate. The sum $\sum_{x \notin \mathcal{X}_n^t} q_n^t(x)$ corresponds to the mass of the SLM distribution that is not contained in the support set $\mathcal{X}_n^t$. When averaged over the accepted tokens, this quantity yields the term $\sum_{n=1}^{T'} \alpha_n(\mathcal{X}_n)$, where T' is the total number of tokens accepted so far. Intuitively, if the probability $\sum_{x \notin \mathcal{X}_n^t} q_n^t(x)$ exceeds the target value α, then the retained support of the distribution is too small and one needs to decrease the threshold. Conversely, one can increase the threshold.

Since the average in the upper bound (1) is only over tokens generated from the LLM, as summarized in Algorithm 1, SQS implements a check-pointing and backtracking strategy, whereby the update (8) is first applied at the edge for all tokens for each batch. Once feedback is received from the cloud, the value of the threshold is returned to the value of the last accepted token, and a further iteration is done for the new, $(T^t + 1)$-th token.

Algorithm 1 Conformal Sparse Quantize-and-Sample Speculative Decoding (C-SQS)

Input: Initial context, vocabulary $\mathcal{V}$, target deviation α, learning rate η, initial threshold β_1^1, per-batch number of tokens L^t, resolution parameters ℓ_n^t
for each batch $t = 1, 2, \ldots$ **do**
  Set $n \leftarrow 1$
  for each token $n = 1, ..., L^t$ **do**
    Compute next-token SLM distribution $q_n^t(\cdot)$
    Evaluate support $\mathcal{X}_n^t(\beta_n^t)$ and cardinality $|\mathcal{X}_n^t(\beta_n^t)|$
    Apply SLQ to $\tilde{q}_n^t$ to obtain quantized SLM distribution $\hat{q}_n^t$ and sample $X_n^t \sim \hat{q}_n^t$
    Apply threshold update (8)
  end for
  Transmit $\{\hat{q}_n^t, \mathcal{X}_n^t(\beta_n^t), X_n^t\}_{l=1}^{L^t}$ to the cloud.
  Receive T^t (number of accepted tokens) and new token $X_{T^t+1}^t$ from the cloud
  Apply the update (8) on the new token as $\beta_{T^t+1}^t = \beta_{T^t}^t - \eta \cdot \left(\sum_{x \notin \mathcal{X}_n^{T^t}} q_n^{T^t}(x) - \alpha \right)$
  Initialize next batch with $\beta_1^{t+1} \leftarrow \beta_{T^t+1}^t$
end for

Communication Overhead: Since C-SQS varies the support K_n^t along the generated tokens, the edge must communicate to the cloud both the size of the subset and the specific subset. This yields the number of bits $\tilde{b}_n^t(K_n^t) = \left\lceil \log_2 \binom{|V|}{K_n^t} \right\rceil + \lceil \log_2 |V| \rceil$,where the second term, $\log_2 |V|$ represents the additional overhead required to communicate the value K_n^t.

Theoretical Guarantee: The following theorem, as proved in Appendix A.3, justifies the use of the update rule (8).

Theorem 2. *For any learning rate $\eta > 0$, C-SQS satisfies the requirement (7) as*

$$\frac{1}{T} \sum_{n=1}^{T} \alpha_n(\mathcal{X}_n) \leq \alpha + \frac{|\beta_1^1| + 1 + \eta\alpha}{\eta T}. \tag{9}$$

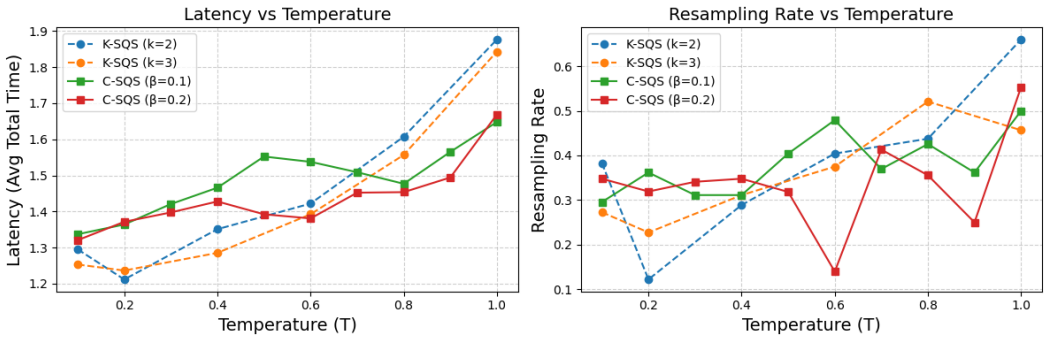

Figure 2: Latency (average total time in seconds) and resampling rate for K-SQS and C-SQS across different temperatures (T). K-SQS shows increasing latency and higher variability in resampling rate with increase in T, while C-SQS maintains more stable performance, achieving a better trade-off between latency and resampling efficiency in higher-uncertainty regimes.

4 Experiments and Discussion

Evaluation Setup and Objectives: We have conducted text completion experiments on the One Billion Word Benchmark (LM1B) dataset [3]. We have used GPT-Neo-125M [4] as the edge SLM, and GPT-Neo-1.3B [5] as the cloud LLM. For evaluating and comparing K-SQS and C-SQS, we analyze two key performance metrics: (a) the average end-to-end latency (average total time), consisting of the SLM computation time, uplink communication time, and cloud LLM verification time as detailed in [22]; and (b) average resampling rate, i.e., the ratio between the average number of rejected and resampled tokens, N_{rej} and the total number of batches. We vary the sampling temperature of the SLM and LLM between $[0, 1]$, while setting the quantization resolution ℓ and the per-batch uplink budget B (in bits) as $B = 5000$ and $\ell = 100$. For each batch t, the number of generated tokens is selected as $L^t = \max\{\, L \in \mathbb{N}_0 \;:\; \sum_{n=1}^{L} b_n^t(K_n^t, \ell) \;\leq\; B \,\}$. This condition is enforced in a sequential way, stopping token generation when the bit budget is exhausted. For C-SQS, we have used a fixed learning rate $\eta = 0.001$ and deviation parameter $\alpha = 0.0005$.

Results: Figure 2 shows the average latency and resampling rate for K-SQS and C-SQS across different temperatures T. At low temperatures, the draft distribution is sharply peaked, so the target token typically falls within the top-K set for a fixed K. In this regime, K-SQS yields fewer rejections and resampling than C-SQS, provided K is chosen appropriately. This in turn yields a lower latency for K-SQS as compared to C-SQS.

As the temperature increases, however, the SLM distribution becomes more diffuse, spreading probability mass over a wider set of tokens. This expansion of the support set happens selectively, only at the tokens for which the SLM distribution is less sharp. In this setting, C-SQS adaptively expands its support, making it more likely to include the target token and thereby reducing its latency and resampling rate. This conclusion is also reflected in the latency performance. Overall, we observe a clear crossover: K-SQS performs better at low temperatures, while C-SQS is more effective at higher temperatures. In Appendix A.4 we provide additional experimental results implementing an ablation study on K-SQS and C-SQS.

Concluding Remarks: In summary, this work demonstrates that efficient compression is critical for overcoming the bandwidth bottleneck in edge–cloud SD. By deriving an information-theoretic characterization of resampling rate and introducing the SQS-SD framework, we show how structured sparsification and quantization can effectively reduce communication costs while preserving accuracy. Both K-SQS and C-SQS significantly reduce latency, and resampling rate, with K-SQS proving more effective in low-uncertainty regimes (lower temperatures), while C-SQS achieves the most favorable balance under high-uncertainty regimes (higher temperatures). These results highlight the potential of distribution-aware compression to make edge–cloud LLM inference both practical and scalable.

Acknowledgment: This work was supported by US NSF under Grants CCF-2100013, CNS-2209951, CNS-2317192; by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing under Award DE-SC-ERKJ422; and by NIH under Award R01-CA261457-01A1. The

work of O. Simeone was supported by the Open Fellowships of the EPSRC (EP/W024101/1) and by the EPSRC project EP/X011852/1.

References

[1] Anastasios Angelopoulos, Emmanuel Candes, and Ryan J Tibshirani. Conformal pid control for time series prediction. *Advances in neural information processing systems*, 36:23047–23074, 2023.

[2] Anastasios N Angelopoulos, Rina Foygel Barber, and Stephen Bates. Theoretical foundations of conformal prediction. *arXiv preprint arXiv:2411.11824*, 2024.

[3] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

[4] EleutherAI. Eleutherai/gpt-neo-125m, 2024. Accessed: 011-2024.

[5] EleutherAI. Eleutherai/gpt-neo-1.3b, 2024. Accessed: 011-2024.

[6] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*, 2018.

[7] Isaac Gibbs and Emmanuel Candes. Adaptive conformal inference under distribution shift. *Advances in Neural Information Processing Systems*, 34:1660–1672, 2021.

[8] Zixu Hao, Huiqiang Jiang, Shiqi Jiang, Ju Ren, and Ting Cao. Hybrid slm and llm for edge-cloud collaborative inference. In *Proceedings of the Workshop on Edge and Mobile Foundation Models*, pages 36–41, 2024.

[9] John Hewitt, Christopher D Manning, and Percy Liang. Truncation sampling as language model desmoothing. *arXiv preprint arXiv:2210.15191*, 2022.

[10] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

[11] Mingyu Jin, Kai Mei, Wujiang Xu, Mingjie Sun, Ruixiang Tang, Mengnan Du, Zirui Liu, and Yongfeng Zhang. Massive values in self-attention modules are the key to contextual knowledge understanding. *arXiv preprint arXiv:2502.01563*, 2025.

[12] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.

[13] Georgy Noarov, Soham Mallick, Tao Wang, Sunay Joshi, Yan Sun, Yangxinyu Xie, Mengxin Yu, and Edgar Dobriban. Foundations of top-k decoding for language models. *arXiv preprint arXiv:2505.19371*, 2025.

[14] Seungeun Oh, Jinhyuk Kim, Jihong Park, Seung-Woo Ko, Tony QS Quek, and Seong-Lyun Kim. Uncertainty-aware hybrid inference with on-device small and remote large language models. *arXiv preprint arXiv:2412.12687*, 2024.

[15] Jihoon Park, Seungeun Oh, and Seong-Lyun Kim. Energy-efficient wireless llm inference via uncertainty and importance-aware speculative decoding, 2025.

[16] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

[17] Noel Teku, Sudarshan Adiga, and Ravi Tandon. Communicating classification results over noisy channels. In *ICC 2024-IEEE International Conference on Communications*, pages 2131–2136. IEEE, 2024.

[18] Noel Teku, Sudarshan Adiga, and Ravi Tandon. Latency-distortion tradeoffs in communicating classification results over noisy channels. *IEEE Transactions on Communications*, 2024.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[20] Ming Yin, Minshuo Chen, Kaixuan Huang, and Mengdi Wang. A theoretical perspective for speculative decoding algorithm. *Advances in Neural Information Processing Systems*, 37:128082–128117, 2024.

[21] Matteo Zecchin and Osvaldo Simeone. Localized adaptive risk control. *Advances in Neural Information Processing Systems*, 37:8165–8192, 2024.

[22] Guangyi Zhang, Yunlong Cai, Guanding Yu, Petar Popovski, and Osvaldo Simeone. Quantize-sample-and-verify: Llm acceleration via adaptive edge-cloud speculative decoding. *arXiv preprint arXiv:2507.00605*, 2025.

A Appendix

A.1 Sparse Lattice-based Quantization

Sparse lattice–based quantization (SLQ) [18] projects the top-K probability vector onto a structured lattice within the simplex, enabling more efficient representation and finer control over quantization error for a given bit budget. Let us consider a resource-constrained edge device running a SLM that produces a probability distribution q over a vocabulary V. If α denotes the cumulative probability mass of the least likely tokens, sparse token analysis identifies the subset of most likely tokens $\mathcal{S} \subseteq V$ that accounts for the top $(1-\alpha)$ probability mass. Let $|\mathcal{S}| = K$ be the number of such top tokens. The sparse lattice-based quantization technique [18] represents this K-dimensional probability vector as a point on a discrete lattice within the K-dimensional probability simplex $\{\hat{q} \in \mathbb{Q}^K | \sum_{i=1}^{K} \hat{q}[i] = 1\}$.

Given the original SLM distribution q, the lattice-quantized distribution over the top-K tokens is denoted $\hat{q}$ and defined as:

$$\hat{Q}_\ell = \left\{ [\hat{q}[1], \hat{q}[2], \ldots, \hat{q}[K]] \in \mathbb{Q}^K \;\middle|\; \hat{q}[i] = \frac{b[i]}{\ell}, \; \sum_{i=1}^{K} b[i] = \ell \right\},$$

where $\ell, b[i]$ are positive integers and ℓ is the lattice resolution parameter, and $b[i]$ are integer counts corresponding to each token probability. The overall procedure is shown in Algorithm 2.

A.2 Proof of Theorem 1

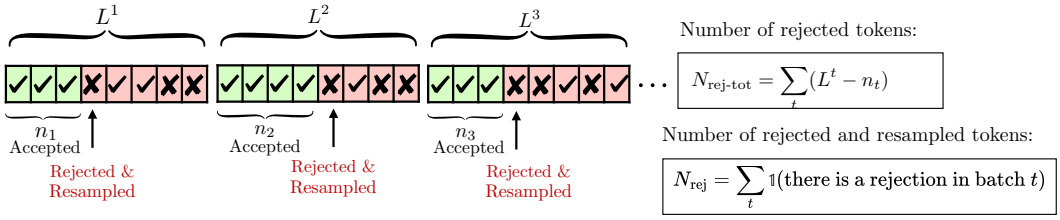

Figure 3: Illustration of the definition of rejected and resampled tokens, N_{rej}, and of the total number of rejected tokens.

As illustrated in Fig. 3, the number of rejected-and-resampled tokens by the LLM, N_{rej}, corresponds to the number of times in which the target LLM invalidates a draft proposal and resamples the rejected token. By contrast, the total number of rejected tokens ($N_{\text{tot-rej}}$ in Fig. 3) corresponds to the total number of SLM generated tokens that were rejected by the target LLM, including all tokens following the resampled token. Computing the expected number of total rejected tokens is non-trivial, since of the rejected tokens do not follow the LLM distribution. For this reason, following [20], we focus on bounding the expected value of rejected and resampled tokens.

Algorithm 2 Sparse-Lattice–Based Quantization

Input: Prompt x, vocabulary V, lattice resolution ℓ, top-K size K
Output: Sparse-lattice-based quantized probability vector $\hat{q}$
Probability Vector Generation: Generate probability distribution q over V from the SLM given prompt x
Top-K Selection: Identify the subset $\mathcal{S} \subseteq V$ of K tokens with highest probabilities. Extract $q = [q[1], q[2], \ldots, q[K]]$.
Lattice-Based Quantization:
Compute $b'[i] \leftarrow \lfloor \ell \cdot q[i] + \frac{1}{2} \rfloor$, for $i = 1, \ldots, K$
$\ell' \leftarrow \sum_{i=1}^{K} b'[i]$
if $\ell' \neq \ell$ **then**
    $\zeta[i] \leftarrow b'[i] - \ell \cdot q[i]$, for $i = 1, \ldots, K$
    Sort indices by increasing $\zeta[i]$.
    if $\ell' - \ell > 0$ **then**
        Decrease the $|\ell' - \ell|$ largest $\zeta[i]$ in $b'[i]$ by 1
    else
        Increase the $|\ell' - \ell|$ smallest $\zeta[i]$ in $b'[i]$ by 1
    end if
end if
$\hat{q}[i] \leftarrow b'[i]/\ell$, for $i = 1, \ldots, K$
Compute lexicographic index to represent $b = [b'[1], \ldots, b'[K]]$
Return: Output $\hat{q}$

To start, we define a sequence of random variables $R_n \in \{0, 1\}$, where $R_n = 1$ indicates that the n-th token is rejected. The total number of rejections is given by

$$N_{\text{rej}} = \sum_{n=1}^{T} R_n. \tag{10}$$

Given verified tokens $\{X_t\}_{t=1}^{n-1}$, we compute

$$\mathbb{P}(\text{Reject at } n \mid \{X_t\}_{t=1}^{n-1}). \tag{11}$$

Let $\tilde{X} \sim \hat{q}_n(\cdot \mid \{X_t\}_{t=1}^{n-1})$ be the candidate draft token. By the law of total probability,

$$\mathbb{P}(\text{Reject at } n \mid \{X_t\}_{t=1}^{n-1}) = \sum_{\tilde{X}} \mathbb{P}(\text{Reject at } n \mid \tilde{X}, \{X_t\}_{t=1}^{n-1})\, \hat{q}_n(\tilde{X} \mid \{X_t\}_{t=1}^{n-1}). \tag{12}$$

By the rejection design of SD, we have

$$\mathbb{P}(\text{Reject at } n \mid \tilde{X}, \{X_t\}_{t=1}^{n-1}) = 1 - \min\left\{1, \frac{p_n(\tilde{X} \mid \{X_t\}_{t=1}^{n-1})}{\hat{q}_n(\tilde{X} \mid \{X_t\}_{t=1}^{n-1})}\right\}. \tag{13}$$

Thus, the probability $\mathbb{P}(\text{Reject at } n \mid \{X_t\}_{t=1}^{n-1})$ can be written as

$$\sum_{\tilde{X}} \max\{0,\, \hat{q}_n(\tilde{X} \mid \{X_t\}_{t=1}^{n-1}) - p_n(\tilde{X} \mid \{X_t\}_{t=1}^{n-1})\} = \text{TV}\big(\hat{q}_n(\cdot \mid \{X_t\}_{t=1}^{n-1}), p_n(\cdot \mid \{X_t\}_{t=1}^{n-1})\big). \tag{14}$$

By the law of total expectation, we arrive at

$$\mathbb{E}[N_{\text{rej}}] = \sum_{n=1}^{T} \mathbb{E}[R_n] = \sum_{n=1}^{T} \mathbb{E}_{\{X_t\}_{t=1}^{n-1} \sim q} \left[\text{TV}\big(\hat{q}_n(\cdot \mid \{X_t\}_{t=1}^{n-1}),\, p_n(\cdot \mid \{X_t\}_{t=1}^{n-1})\big)\right]. \tag{15}$$

Therefore, the expected number of rejections satisfies

$$\begin{aligned}\mathbb{E}[N_{\text{rej}}] &= \sum_{n=1}^{T} \mathbb{E}_{\{X_t\}_{t=1}^{n-1}\sim p}\left[\mathrm{TV}\big(\hat{q}_n(\cdot \mid \{X_t\}_{t=1}^{n-1}),\ p_n(\cdot \mid \{X_t\}_{t=1}^{n-1})\big)\right] \\ &\overset{(a)}{\leq} \sum_{n=1}^{T} \mathbb{E}_{\{X_t\}_{t=1}^{n-1}\sim p}\left[\mathrm{TV}\big(\hat{q}_n(\cdot \mid \{X_t\}_{t=1}^{n-1}),\ q_n(\cdot \mid \{X_t\}_{t=1}^{n-1})\big)\right] \\ &\quad + \sum_{n=1}^{T} \mathbb{E}_{\{X_t\}_{t=1}^{n-1}\sim p}\left[\mathrm{TV}\big(q_n(\cdot \mid \{X_t\}_{t=1}^{n-1}),\ p_n(\cdot \mid \{X_t\}_{t=1}^{n-1})\big)\right], \end{aligned} \tag{16}$$

where inequality (a) holds due to the triangle inequality.

Inspired by [18], we can also bound the sparse lattice-based quantization (SLQ) term $\sum_{n=1}^{T} \mathbb{E}_{\{X_t\}_{t=1}^{n-1}\sim p}\left[\mathrm{TV}\big(\hat{q}_n(\cdot \mid \{X_t\}_{t=1}^{n-1}),\ q_n(\cdot \mid \{X_t\}_{t=1}^{n-1})\big)\right]$. To simply the notation, we denote $q(\cdot \mid \{X_t\}_{t=1}^{n-1})$ as q.

Given the inequality $\sum_{i\in k} q[i] \geq 1-\alpha, \quad \alpha \in [0,1]$, we have $\sum_{i\notin k} q[i] \leq \alpha$. Let $S = \sum_{i\in k} q[i]$. We define $\bar{q}$ as the normalized probability vector over the K entries, where

$$\bar{q}[i] = \begin{cases} \frac{q[i]}{S}, & i \in k, \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

We upper bound $\mathrm{TV}(q, \bar{q})$ as follows:

$$\begin{aligned}\mathrm{TV}(q,\bar{q}) &= \frac{1}{2}\sum_i |q[i]-\bar{q}[i]| \\ &= \frac{1}{2}\left(\sum_{i\in k}\left|q[i]-\frac{q[i]}{S}\right| + \sum_{i\notin k}|q[i]-\bar{q}[i]|\right) \\ &\overset{(a)}{=} \frac{1}{2}\left(\sum_{i\in k} q[i]\left|1-\frac{1}{S}\right| + \sum_{i\notin k} q[i]\right) \\ &= \frac{1}{2}\left(\sum_{i\in k} q[i]\frac{|S-1|}{S} + \sum_{i\notin k} q[i]\right) \\ &\overset{(b)}{=} \frac{1}{2}\left(\sum_{i\in k} q[i]\frac{1-S}{S} + \sum_{i\notin k} q[i]\right), \end{aligned} \tag{18}$$

where (a) follows from $\bar{q}[i] = 0$ for all $i \notin k$, and (b) follows from $S \geq 0$.

Following similar steps, we have

$$\begin{aligned}\mathrm{TV}(q,\bar{q}) &\overset{(a)}{=} \frac{1}{2}\left(\sum_{i\in k}\frac{q[i]}{S} - \sum_{i\in k} q[i] + \sum_{i\notin k} q[i]\right) = \frac{1}{2}\left(\frac{\sum_{i\in k} q[i]}{S} - \sum_{i\in k} q[i] + \sum_{i\notin k} q[i]\right) \\ &\overset{(b)}{=} \frac{1}{2}\left(1 - \sum_{i\in k} q[i] + \sum_{i\notin k} q[i]\right) \\ &\overset{(c)}{=} 1 - \sum_{i\in k} q[i] \overset{(d)}{\leq} \alpha, \end{aligned} \tag{19}$$

where (a) follows from $0 \leq S \leq 1$, (b) from $S = \sum_{i\in k} q[i]$, (c) from $\sum_{i\in k} q[i] + \sum_{i\notin k} q[i] = 1$, and (d) from $\sum_{i\notin k} q[i] \leq \alpha$.

From [18], the distortion due to LQ can be upper bounded as

$$\mathrm{TV}(\bar{q}, \hat{q}) \leq \frac{k}{4\ell}, \tag{20}$$

where l is a positive integer that can be set by users. Therefore $\mathrm{TV}(m, \hat{q})$ can be upper bound by $\alpha + \frac{k}{4\ell}$.

A.3 Proof of Theorem 2

Recall the threshold update rule introduced in Section 3 via equation (8):

$$\beta_{n+1}^t = \beta_n^t - \eta \cdot \left(\sum_{x \notin \mathcal{X}_n^t} q_n^t(x) - \alpha \right), \tag{21}$$

where β_{n+1}^t is the threshold used for token n in batch t, $\alpha_n^t(\mathcal{X}_n^t) = \sum_{x \notin \mathcal{X}_n^{T^t}} q_n^{T^t}(x)$ is the dropped probability mass due to sparsification at step n, $\alpha \in (0,1)$ is the target deviation, and $\eta > 0$ is the learning rate. The pre-batch threshold initialization follows Algorithm 1. Let the total number of accepted tokens across all batches be $T = \sum_t^T L_t$. Throughout this section, the index $t = 0, 1, \ldots, T-1$ enumerates the T accepted tokens in chronological order.

Lemma 1. *Let V denote the vocabulary, and consider any sparsification strategy. The total probability mass outside the sparsified token set $\mathcal{X}_n^t$ can be expressed in terms of the total variation (TV) distance between the true distribution q^t and the sparsified distribution $\tilde{q}^t$:*

$$\sum_{x \notin \mathcal{X}_n^t} q^t(x) \;=\; \mathrm{TV}\big(q^t, \tilde{q}^t\big)\,. \tag{22}$$

Proof. We start with the definition of $\mathrm{TV}(q^t, \tilde{q}^t)$:

$$\begin{aligned}
\mathrm{TV}(q^t, \tilde{q}^t) &= \frac{1}{2} \sum_{x \in \mathbb{V}} |q^t(x) - \tilde{q}^t(x)| = \frac{1}{2} \sum_{x \in \mathcal{X}_n^t} |q^t(x) - \frac{q^t(x)}{\sum_{x \in \mathcal{X}_n^t} q^t(x)}| + \frac{1}{2} \sum_{x \notin \mathcal{X}_n^t} q^t(x) \\
&= \frac{1}{2} \sum_{x \in \mathcal{X}_n^t} q^t(x) \left(\frac{1}{\sum_{x \in \mathcal{X}_n^t} q^t(x)} - 1 \right) + \frac{1}{2} \sum_{x \notin \mathcal{X}_n^t} q^t(x) \\
&= \frac{1}{2} \left(1 - \sum_{x \in \mathcal{X}_n^t} q^t(x) + \sum_{x \notin \mathcal{X}_n^t} q^t(x) \right) = \sum_{x \notin \mathcal{X}_n^t} q^t(x)
\end{aligned} \tag{23}$$

□

This completes the proof of Lemma 1. Using this lemma, we can re-write the update rule of 8 as presented in Lemma 2. We next prove the following result which uses the new alternative form of the threshold update rule.

Lemma 2. *Given the Lemma 1 and assume the threshold is updated at each accepted token as*

$$\beta^{t+1} \;=\; \beta^t \;-\; \eta \big(\mathrm{TV}(\tilde{q}^t, q^t) \;-\; \alpha\big), \qquad \eta > 0,\ \alpha \in (0,1). \tag{24}$$

Then the cumulative sparsification distortion satisfies the identity

$$\sum_{t=0}^{T-1} \mathrm{TV}(\tilde{q}^t, q^t) \;=\; \alpha\, T \;+\; \frac{|\beta^0 - \beta^T|}{\eta}. \tag{25}$$

This follows via an algebraic telescoping sum. Any asymptotic statement such as $\frac{1}{T} \sum_{t=0}^{T-1} \mathrm{TV}(\tilde{q}^t, q^t) \to \alpha$ requires a separate argument showing that $|\beta^0 - \beta^T|$ is $O(1)$; this will be established by bounding the iterates $\{\beta^t\}$.

Lemma 3 (Step-size envelope)**.** *Since $0 \le \mathrm{TV}(\tilde{q}_n, q_n) \le 1$, each update satisfies the inequalities*

$$-\eta(1-\alpha) \;\le\; \beta_{n+1} - \beta_n \;\le\; \eta\alpha. \tag{26}$$

Thus, every step moves β by at most η in either direction.

Lemma 4 (Universal bound on β)**.** *The sequence (β_n) is uniformly bounded:*

$$-\eta(1-\alpha) \;\le\; \beta_n \;\le\; 1+\eta\alpha, \qquad \forall n \ge 0. \tag{27}$$

Proof. If $\beta_n < 0$, then thresholding keeps the full support and hence $\mathrm{TV}(\tilde{q}_n, q_n) = 0$. The update becomes $\beta_{n+1} = \beta_n + \eta\alpha > \beta_n$, so β is forced upward. By Lemma 3, the largest one-step overshoot below 0 is $-\eta(1-\alpha)$.

If $\beta_n > 1$, then thresholding discards all but the top outcome, so $\mathrm{TV}(\tilde{q}_n, q_n) = 1$. Since $\alpha < 1$, we obtain $\beta_{n+1} = \beta_n - \eta(1-\alpha) < \beta_n$, so β is forced downward. By Lemma 3, the largest one-step overshoot is $1 + \eta\alpha$. Combining the two cases yields the stated bound. □

Plugging Lemma 4 into Lemma 2 gives

$$\sum_{n=1}^{T} \mathrm{TV}(\tilde{q}_n, q_n) \;\le\; \alpha T \;+\; \frac{|\beta_0| + 1 + \eta}{\eta}, \tag{28}$$

and hence

$$\frac{1}{T}\sum_{n=1}^{T} \mathrm{TV}(\tilde{q}_n, q_n) \;\le\; \alpha \;+\; O(T^{-1}), \tag{29}$$

as desired. Furthermore, if $\eta = c\,T^{-1/2}$ and $\alpha = d\,T^{-1/2}$ with $c, d > 0$, then

$$\sum_{n=1}^{T} \mathrm{TV}(\tilde{q}_n, q_n) = \mathcal{O}(\sqrt{T}), \qquad \frac{1}{T}\sum_{n=1}^{T} \mathrm{TV}(\tilde{q}_n, q_n) = \alpha + \mathcal{O}(T^{-1/2}). \tag{30}$$

This completes the proof.

A.4 Additional Experimental Results

In this part of the Appendix, we present an ablation study that investigates the effects of the parameters K and β for the K-SQS and C-SQS methods, respectively, while varying the temperature T. We also report results demonstrating that C-SQS without adaptivity (i.e., learning rate $\eta = 0$) exhibits higher latency and re-sampling rates compared to the adaptive version ($\eta > 0$), thereby highlighting the benefits of adaptive parameter updates for β.

A.4.1 Impact of Hyperparameters K and β

First we have analyzed the effect of the parameters K and β over varying temperature for K-SQS and C-SQS methods. Figure 4 shows the latency performance for K-SQS and C-SQS across varying temperature settings. Both schemes demonstrate consistent performance trends, with latency generally increasing as the temperature and consequently the sampling uncertainty-rises.

For K-SQS, performance is highly dependent on the choice of the K-value: smaller K values yield lower latency but may reduce stability, while larger K values improve robustness at the cost of increased computation. In contrast, C-SQS leverages adaptive threshold tuning through the parameter β, enabling dynamic control over resampling and thereby achieving smoother latency–accuracy trade-offs.

Collectively, these findings indicate that K-SQS and C-SQS are complementary: K-SQS performs optimally in low-uncertainty regimes with well-chosen K, whereas C-SQS excels under higher-uncertainty conditions due to its adaptive thresholding mechanism.

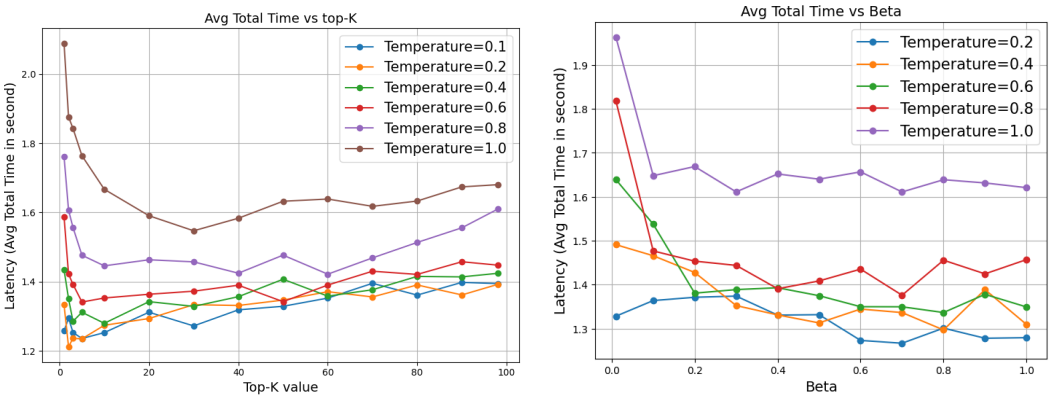

Figure 4: Latency for K-SQS and C-SQS methods versys K and β, respectively, across varying temperature settings.

A.4.2 Benefits of Adaptivity in C-SQS

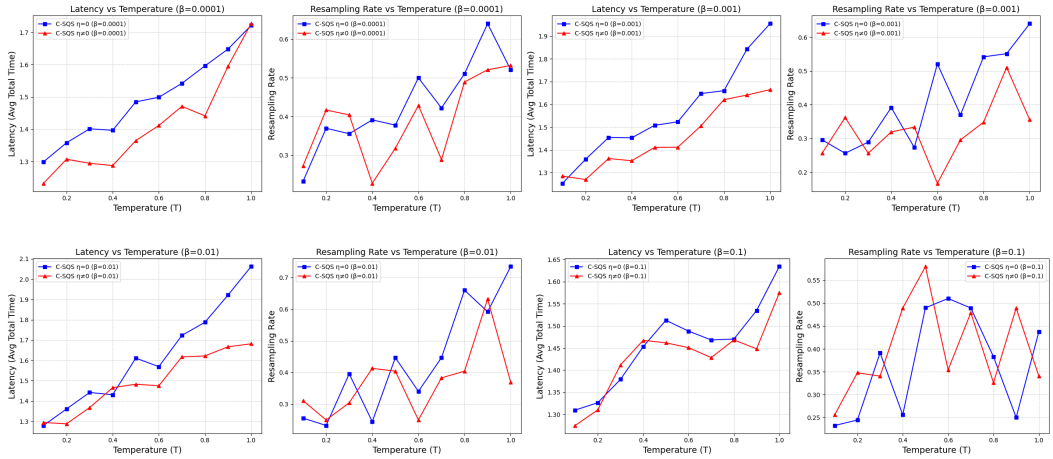

Figure 5: Latency and resampling rate as a function of temperature for C-SQS with and without adaptivity.

Figure 5 presents the relationship between temperature and (a) latency, and (b) resampling rate, across different initial threshold (β) values, comparing adaptive ($\eta > 0$) and non-adaptive ($\eta = 0$) configurations of C-SQS. It is observed that incorporating adaptivity during β-updates significantly improves efficiency. Notably, for smaller β values that represent more conservative acceptance thresholds, the adaptive variant consistently yields lower latency and reduced resampling rates relative to the non-adaptive baseline. This indicates that adapting the threshold β enables C-SQS to better regulate resampling behavior, effectively balancing stability and responsiveness under varying uncertainty (temperature) conditions.

A.4.3 K-SQS vs C-SQS

In Figure 6, we compare the performance of K-SQS and C-SQS under varying temperature (T) values, illustrating their latency and resampling characteristics. As temperature increases, indicating higher sampling uncertainty, both methods show a general rise in latency and resampling rate due to more frequent token rejections. For K-SQS, performance is highly influenced by the selection of the K-value: smaller K values yield faster but less stable performance, while larger K values improve reliability at the cost of increased latency.

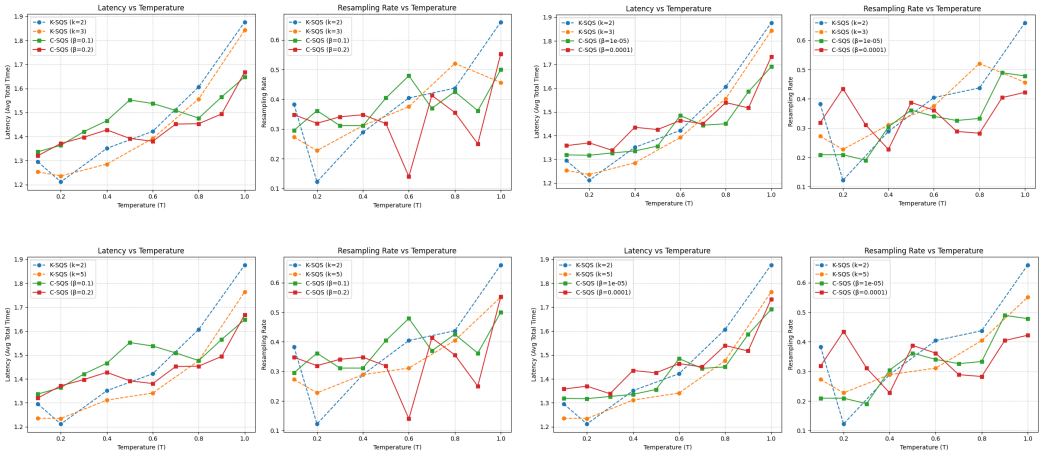

Figure 6: Latency and resampling rate of K-SQS and C-SQS across varying temperature (T) settings.

In contrast, C-SQS leverages its adaptive threshold β, allowing it to dynamically adjust to temperature changes and maintain a more balanced latency, accuracy trade-off. Overall, these results highlight that K-SQS is preferable in low-uncertainty regimes (lower temperature regimes), whereas C-SQS demonstrates greater robustness and efficiency in higher-uncertainty conditions due to its adaptive mechanism.