# DRAFT-AND-VERIFY: SPECULATIVE ENTROPY DECODING FOR AUTOREGRESSIVE LEARNED COMPRESSION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Autoregressive (AR) entropy models have been central to modern neural lossless compression, but their sequential decoding introduces substantial latency, often orders of magnitude higher than commercial codecs. We present SpecED, a speculative entropy decoder that reduces the number of sequential entropy-model calls while preserving bit-exact reconstruction from the bitstream. Unlike standard entropy decoding, which invokes the full AR model sequentially for each latent symbol, SpecED uses a lightweight draft model to predict the distributions of several future symbols, the arithmetic decoder recovers candidate symbols from the bitstream under these drafts, and one batched call of the base model verifies them. This draft-and-verify framework combines multi-head lookahead drafting, speculative state branching, and budgeted speculation-tree search to reduce sequential latency while balancing speculation depth, branch diversity, and verification cost. Theoretically, we establish a Rényi-divergence-based bound showing that the probability of consecutive speculation success decays exponentially in the number of drafted symbols, at a rate set by the Chernoff information between draft and base models, and derive a tractable KL-based approximation for practical draft-model design. On lossless codecs for text (LLMZip), point clouds (OctAttention), and particle-physics data (BOA Constrictor), SpecED yields ideal entropy-model speedups of $1.9$–$2.4\times$, and up to $1.8\times$ measured end-to-end wall-clock speedup on LLMZip ($2.5\times$ on enwik8). The algorithm is extended to lossy neural codecs with AR entropy models, where we obtain $1.3$–$2.3\times$ theoretical speedups in Encodec, with gains shrinking as the per-symbol entropy grows, as predicted by our theoretical bound.

## 1 INTRODUCTION

Traditional lossless compression relies on hand-crafted statistical models, such as context modelling (Cleary & Witten, 1984) and dictionary methods (Ziv & Lempel, 1977), combined with entropy coding such as arithmetic coding (Shannon, 1948; Witten et al., 1987). With the rapid development of deep learning, neural codecs have emerged as a powerful alternative. Through end-to-end optimization, learned lossless codecs replace the hand-crafted predictor with a neural probability model and arithmetic-code the resulting conditionals, achieving strong performance across modalities including point cloud (Nguyen et al., 2021a; Fu et al., 2022), text (Valmeekam et al., 2023; Deletang et al.), scientific data (Goyal et al., 2019; Gupta et al., 2026) and images (Mentzer et al., 2019; Townsend et al., 2019). These gains, however, often come with substantial decoding latency. A major bottleneck is entropy decoding (Nguyen et al., 2021b; Mittu et al., 2024), where autoregressive (AR) entropy models estimate symbol probabilities conditioned on previously decoded contexts. This sequential dependency limits parallel inference and substantially slows decoding, making efficient entropy decoding a critical challenge for practical learned compression systems.

Similar latency bottlenecks arise in AR generation (Xie et al., 2025; Deng et al., 2025; Liu et al., 2024), where speculative decoding (Leviathan et al., 2023; Xia et al., 2024; Yin et al., 2024) drafts multiple future tokens with a lightweight model and verifies them in parallel with the original large model. Speculative decoding is exact in distribution; that is, any accepted sample is a valid sample from the target model. However, entropy decoding must be exact pathwise; that is, the decoder has to reproduce the exact sequence the bitstream encodes. The arithmetic decoder's state after each symbol depends on the exact probabilities used to decode it. Therefore, even small distributional



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.

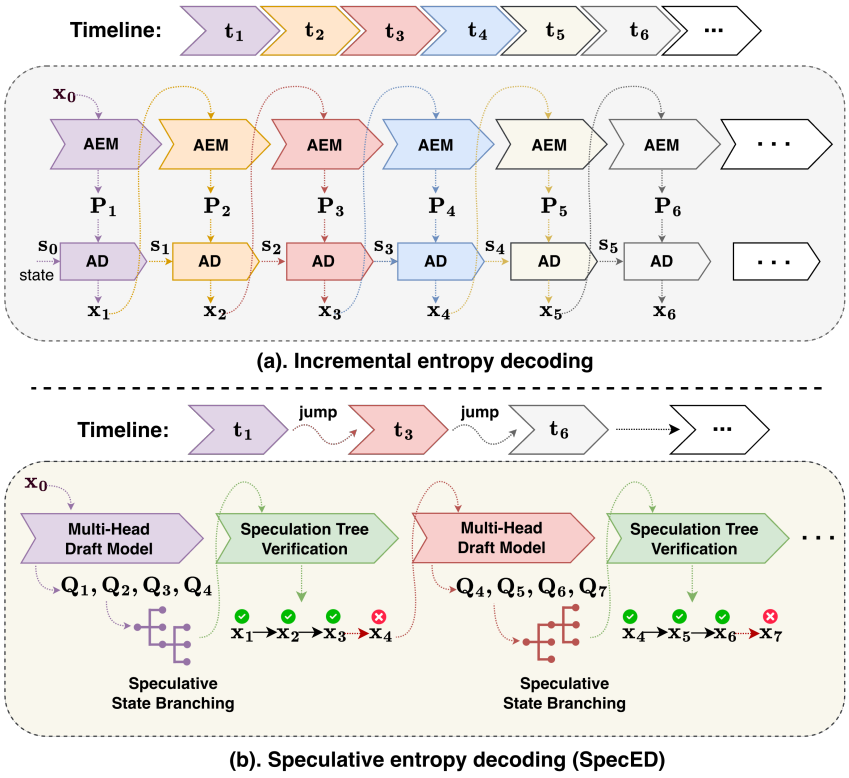



Figure 1: **Illustration of the proposed speculative entropy decoding framework.** (a) Standard entropy decoding is incremental: each autoregressive entropy model (AEM) inference produces one distribution, and arithmetic decoding (AD) recovers one symbol, leading to a sequential bottleneck. (b) SpecED uses multi-head draft models to predict future distributions in parallel and verifies candidate state branches through exact arithmetic decoding. A speculation tree maintains multiple candidates, where exact verification accepts only symbol-consistent branches, enabling multi-step decoding and breaking the autoregressive bottleneck.

discrepancies between the encoder's and decoder's models can cause arithmetic coder's intervals to mismatch, and even when the draft decodes the right symbols, it can still leave the decoder in the wrong state. This raises the question: *Can approximate draft models parallelize entropy decoding while preserving exact recovery from the bitstream?*

To parallelize AR entropy models in neural lossless compression codecs, we propose SpecED (Fig. 1), a speculative entropy decoding framework for arithmetic coding that combines parallel lookahead prediction with exact arithmetic verification. SpecED employs multiple lightweight draft heads to predict future latent distributions in parallel, inducing candidate symbols and arithmetic states. The original entropy decoder then verifies these candidates and accepts only symbol-consistent branches, preventing error propagation. To control search complexity, SpecED further introduces a budgeted speculation-tree strategy that balances speculation depth, branch diversity, and verification overhead.

The feasibility of our design stems from two observations. First, *latent symbols exhibit highly non-uniform predictability*: many lie in low-entropy regimes and can be inferred from the current context without invoking the full AR model. Second, *exact entropy decoding requires symbol-level consistency rather than exact distribution matching*. SpecED leverages these observations by maintaining a total budget of speculative branches and letting exact verification select the one that is consistent with the bitstream. This ensures the accepted candidate falls within the correct decoding intervals, and as a result, mitigates state drift and error propagation.

Theoretically, we further derive a Rényi-divergence-based upper bound on the consecutive speculation success probability, which decays exponentially with the divergence between draft and true entropy models, constraining the achievable acceleration. This result guides the speculation length choice

and multi-head draft design. We also derive a tractable KL-based approximation, which empirically predicts acceleration trends across entropy of symbols and provides a practical rule for draft-model optimization.

By combining parallel draft prediction with exact verification, SpecED preserves correctness while reducing the sequential decoding bottleneck. Importantly, it treats the base model as a black box; and therefore, applies to all AR entropy-model-based codecs with an arithmetic decoder, while our theory links its feasibility to the divergence between draft and true models. SpecED also applies to lossy codecs whose latents are coded with AR entropy models which continue to be a bottleneck (Jiang et al., 2018; Lu et al., 2020; Défossez et al.). This makes it particularly valuable for latency-sensitive and real-time decoding scenarios. More broadly, our results can guide the acceleration and analysis of prediction–compression systems (Deletang et al.), where compressors can be used to build conditional generative models. Our main contributions are summarized as follows:

- We propose speculative entropy decoding, a new model-agnostic acceleration paradigm that can apply to any codec with an AR entropy model and an arithmetic decoder. It mitigates the sequential decoding bottleneck while preserving exact reconstruction.
- We develop SpecED, a practical scheme that incorporates multi-head lookahead drafting and speculative state branching to prevent decoder state drift, and budgeted speculation-tree search for efficient parallel speculation.
- We derive a theoretical Rényi-divergence success bound, showing that acceleration decays exponentially with draft-true model divergence, and derive a tractable KL-based approximation for practical design.
- We validate SpecED across different codecs, modalities, and bit-rates, demonstrating substantial decoding acceleration, verifying our theoretical predictions, and revealing an inverse relationship between speedup and bit-rate in lossy compression.

## 2 Related Work

### 2.1 Neural compression

Autoregressive (AR) modeling has been widely adopted in modern neural compression, where symbols are ordered and each is conditioned on previously decoded ones. In learned lossless compression, AR context models are crucial for accurate probability estimation across point cloud, scientific byte streams, texts and images. Text codecs have progressed from recurrent predictors to pretrained language models (Goyal et al., 2019; Valmeekam et al., 2023; Deletang et al.), while point-cloud codecs shift from local occupancy models to large-context attention (Nguyen et al., 2021a; Fu et al., 2022). For scientific byte streams, recurrent predictors were usually employed, subsequently replaced by state-space models (Goyal et al., 2019; Gupta et al., 2026). On images, per-pixel autoregressive models give way to hierarchical auxiliary representations (Mentzer et al., 2019), and latent-variable models adopt bits-back coding (Townsend et al., 2019). Following the renewed emphasis on the equivalence between prediction and compression (Deletang et al.), pretrained generative models have been used as entropy models for other modalities as well (Li et al., 2025). Similar structure is also common in learned lossy compression, while the target of entropy coding moves from bytes/symbols to learned latents. Image codecs have progressed from spatial-neighbor contexts to joint spatial–channel modeling (He et al., 2022), while video codecs exploit temporal dependencies for motion and residual coding (Lu et al., 2019; Li et al., 2021; 2023; Mentzer et al., 2022). Despite their effectiveness, AR models incur high decoding latency from sequential probability estimation. Acceleration methods exist for specific modalities, such as image (He et al., 2021; 2022; Han et al., 2024), they sacrifice modeling capacity or impose structural constraints. Thus, accelerating AR decoding without compromising fidelity or exactness remains open. We instead preserve the AR formulation and reduce sequential entropy-model evaluations through speculative decoding.

### 2.2 Speculative decoding

Speculative decoding accelerates AR generation by using a lightweight draft model to propose multiple tokens and a stronger target model to verify them in parallel while preserving exact sampling (Leviathan et al., 2023). It has become central to LLM inference, with extensions using

multi-head candidates, speculative sampling, tree-based verification, and lookahead strategies to improve acceptance and parallelism (Cai et al.; Chen et al., 2023; Miao et al., 2024; Kobus & Gunduz, 2025). Their acceptence rate is governed by the overlap between the draft and target distributions. However, prior work targets sequence prediction, where verification is symbol-wise, and the state is simply the generated prefix. In contrast, entropy decoding (e.g., arithmetic decoding) is inherently *stateful*, where each symbol depends on an evolving interval state. The distribution mismatch can induce state drift or error accumulation even when early symbols are correct. This makes speculative decoding nontrivial for entropy decoding and motivates our state-aware adaptation, which explicitly accounts for state drift and its impact on acceptance and efficiency.

# 3 Theoretical Foundation of Speculative Entropy Decoding

## 3.1 Arithmetic Decoding

Arithmetic coding represents a sequence $x_{1:T}$ as a nested interval in $[0, 1)$ using a probabilistic model with conditional distribution $P_t(x_t \mid x_{<t})$. Starting from a state represented as an interval $[0, 1)$, the encoder iteratively refines it via

$$[\ell_t, u_t) \;=\; \big[\ell_{t-1} + (u_{t-1} - \ell_{t-1})F_t(x_t^-), \ell_{t-1} + (u_{t-1} - \ell_{t-1})F_t(x_t)\big)\,, \tag{1}$$

where $F_t(\cdot)$ is the cumulative distribution induced by $P_t(\cdot \mid x_{<t})$, i.e., $F_t(a) := \Pr(x_t \le a | x_{<t})$, and $F_t(x_t^-) = \sum_{a<x_t} P_t(a|x_{<t})$. The encoder then transmits a bitstring whose binary expansion identifies a real number $z \in [\ell_T, u_T)$. The decoder reads the bitstring and performs the inverse recursion. Given $z$, it recovers each $x_t$ by locating the unique symbol $x_t$ whose cumulative interval under $F_t$ contains $z$, and updates $[\ell_t, u_t)$ accordingly using the same model.

A critical property is *model synchrony*: the encoder and decoder must use *identical* conditional distributions $P_t(\cdot|x_{<t})$ at every step. In practice, this requires bit-level exact agreement on the entropy models' outputs and the integer-quantized cumulative distribution $F_t$. Any mismatch perturbs the cumulative boundaries of $F_t(\cdot)$, yielding a divergent interval recursion, and can corrupt all the symbols from that step onward. The decoder retrieves the tokens following the recursive relation:

$$x_i, s_i = \text{ArithmeticDecoder}(P_i(\cdot|x_{<i}), s_{i-1}), \tag{2}$$

where $s_{i-1}$ denotes the decoder state at timestep $i-1$ comprising the interval $[\ell_{i-1}, u_{i-1})$, the value $z$ and the bitstream pointer.

## 3.2 Speculative Decoding

Speculative decoding accelerates autoregressive generation by using a lightweight draft model to propose multiple future tokens in parallel, which are then verified by the base model in a single forward pass. As illustrated in Fig. 2 (a), the draft model produces predictive distributions $\{Q_{i+1}, \ldots, Q_{i+K}\}$ at timestep $i$, from which candidate tokens $\{\hat{x}_{i+1}, \ldots, \hat{x}_{i+K}\}$ are independently sampled. The base model evaluates the corresponding conditional distributions $P_{i+k}(\cdot \mid x_{<i+k})$ in parallel, and performs rejection sampling to verify each proposal.

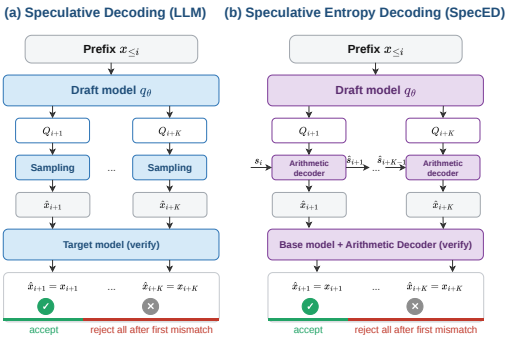



Figure 2: Comparison of conventional speculative decoding and speculative entropy decoding.

In contrast, SpecED (Fig. 2 (b)) deterministically decodes candidate tokens by feeding draft distributions and candidate arithmetic states into the arithmetic decoder, and verifies them via symbol-level consistency. In both cases, decoding follows a prefix rule: tokens are accepted sequentially until the first mismatch, after which all subsequent tokens are discarded. The efficiency is therefore governed by the expected accepted prefix length under mismatch between the draft distribution $Q$ and the true distribution $P$. However, speculative decoding relies on stochastic sampling and rejection, whereas SpecED uses deterministic, state-dependent verification, which precludes a direct application of existing analysis and motivates a dedicated treatment.

### 3.3 SPECULATIVE ENTROPY DECODING

We adopt a speculative-decoding-like design to accelerate the entropy model, which dominates decoding latency due to its AR nature. We fix a starting timestep $i$ and a speculation horizon $K$. The true distribution of the autoregressive entropy model is given by $P_t(\cdot|x_{<t}), t = i, ..., i+K$, each conditioned on all the preceding symbols. To reduce the number of inferences of the entropy model, following Fig. 2, at each timestep, $K$ draft models are run concurrently along with the base entropy model to speculate ahead, proposing $Q_{i+k}(\cdot|x_{<i})$ for $k = 1, ..., K$. Then $K$ verifications are carried out concurrently[1] via a single forward step of the base entropy model, i.e.,

$$\mathbb{1}_k = \mathbb{1}(\tilde{x}_{i+k} = \hat{x}_{i+k}) \quad \text{for } k = 1, ..., K, \tag{3}$$

where

$$\hat{x}_{i+k}, \hat{s}_{i+k} = \text{ArithmeticDecoder}(Q_{i+k}(\cdot|x_{<i}), \hat{s}_{i+k-1}) \tag{4}$$

and $\hat{s}_i = s_i$,

$$\tilde{x}_{i+k}, \tilde{s}_{i+k} = \text{ArithmeticDecoder}(P_{i+k}(\cdot|x_{<i}, \hat{x}_{<i+k}), s_{i+k-1}). \tag{5}$$

Speculation through depth $k$ is accepted if $\prod_{j=1}^{k} \mathbb{1}_j = 1$. Hence, with $k^*$ consecutive successful speculations, we retrieve $1 + k^*$ tokens per forward pass of the base entropy model.

**Theorem 1** (Speculation success rate bound for general compression scheme with arithmetic coding)**.** *Suppose a compressed bitstream is arithmetic encoded by the autoregressive entropy model $P_i(\cdot|x_{<i})$. Denoting $K$ draft models $Q_{i+k}(\cdot|x_{<i}), k = 1, ..., K$ and letting $P$ and $Q$ be the joint distributions over the $n$-symbol block, the probability of successfully decoding the first $n$ correct tokens with the draft models is bounded by*

$$\mathbb{P}_n(\text{success}) \leq \exp\Big(- \sup_{\lambda \in (0,1)} \lambda \, D_{1-\lambda}(P\|Q)\Big), \tag{6}$$

*where $D_\alpha(P\|Q)$ is the Rényi divergence.*

*Proof of Theorem 1 is provided in Appendix A.* □

**Remark 2.** *We note that unlike speculative decoding in LLMs, where an accepted token restores the context, the conditional probability of the next success in speculative entropy decoding need not stay constant with $n$, since error can accumulate in the arithmetic decoder state even if the preceding decoded tokens are correct. This makes speculative entropy coding harder than speculative sampling in practice as a correct prefix does not necessarily restore the draft's state.*

Theorem 1 provides an upper bound on speculation success governed by the Chernoff information between $P$ and $Q$.[2]. However, while the supremum over $\lambda$ is low-dimensional, computing $D_{1-\lambda}(P\|Q)$ for arbitrary $\lambda$ is typically intractable in practice. To obtain a tractable KL-based approximation:

**Corollary 1.** *With the same setting as Theorem 1, if $P$ and $Q$ are memoryless joints with per-step divergence $\delta = D_{\text{KL}}(P_1\|Q_1)$, then as $\delta \to 0$,*

$$\mathbb{P}_n(\text{success}) \leq \exp\Big(-\frac{n}{4}\delta + o(n)\Big). \tag{7}$$

*Proof of Corollary 1 is provided in Appendix A.* □

Corollary 1 is generally loose but provides a tractable design objective. Furthermore, we note that Corollary 1 is greatly effective in practice as a prediction rule for compression schemes across a range of compression ratios, by choosing and fitting an $o(n)$ term. We show with experiments and discuss this further in Section 5.

[1] Due to the dependence on $s_i$, the arithmetic decoder has to be executed sequentially. In practice, however, the entropy model has high computational cost (e.g., neural networks or diffusion models), whereas arithmetic decoding reduces to binary/interval search; the combined process can therefore be treated as concurrent.

[2] Detailed analysis and proof can be seen in Appendix A.

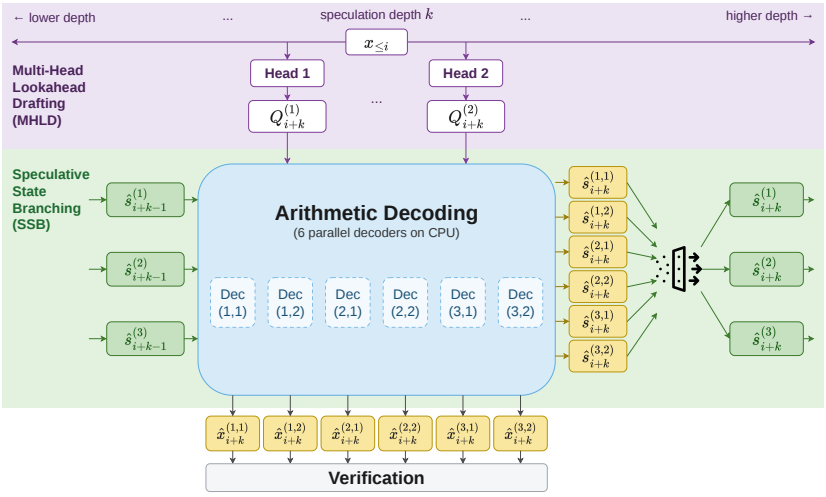



Figure 3: Illustration of multi-head lookahead drafting (MHLD) and speculative state branching (SSB) in timestep $i$ with speculation depth $k$ for number of draft models $M = 2$, SLB with limit 3.

## 4 METHODOLOGY

As illustrated in Fig. 3, we design a workflow where at timestep $i$, proposals for token $x_{i+k}$ are obtained by Multi-Head Lookahead Drafting following Speculative State Branching.

**Multi-Head Lookahead Drafting.** At each decoding step $i$, instead of invoking the full entropy model autoregressively, we employ $M$ draft models to predict future distributions in parallel. We retain the term *multi-head* to denote parallel proposal mechanisms, which may be implemented either as multiple heads on a shared backbone or as independent lightweight models. Detailed implementations can be found in Appendix C. Concretely, draft models produce a $K$-step lookahead:

$$Q^{(m)}_{i+k}(\cdot \mid x_{<i}), \quad m = 1, \ldots, M, \; k = 1, \ldots, K, \tag{8}$$

taking historical contexts $x_{<i}$ as input to output a parametric draft distribution or a direct categorical distribution over the codebook, yielding a set of $M$ candidate distributions for each future position.

Each draft distribution induces a corresponding candidate arithmetic decoding trajectory. For every position, up to $M$ candidate tokens are obtained via arithmetic decoding under the respective draft distributions, enabling parallel exploration of multiple speculative futures without committing to a single path. This parallel proposal process is illustrated in Fig. 3. For illustration, let $\hat{s}^{(a_1,\ldots,m)}_{i+k}$ denote candidate state at the end of decoding speculation for token at depth $k$, while the state is a result of arithmetic decoding $Q^{(m)}_{i+k}$ with state $\hat{s}^{(a_1,\ldots)}_{i+k}$.

**Speculative State Branching.** The use of multiple draft models induces a branching process over candidate arithmetic decoding states, as shown in Fig. 3. Each branch corresponds to a hypothesis defined by a decoding state together with its associated sequence of draft distributions. At lookahead depth $k$, each active state can expand into up to $M$ new states, resulting in at most $M^k$ candidates. Instead of committing to a single speculative trajectory, we propagate this set of candidate states in parallel, forming a speculation tree as illustrated in Fig. 4.

Verification against the true arithmetic decoding process is performed via symbol-level consistency constraints. Importantly, verification signals are not available during speculative expansion, and therefore do not influence the branching process.

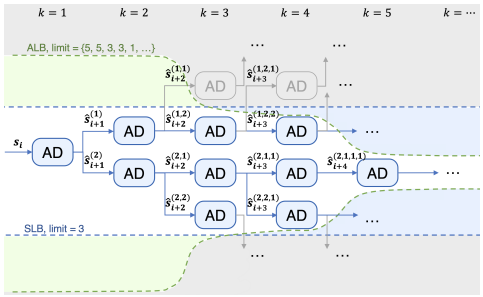



Figure 4: Illustration of Speculative State Branching for $M = 2$. AD block here represents 2 Arithmetic Decoders each with $Q^m_{i+k}, m = 1, 2$. Token outputs are not shown for clarity. One possible implementation of SLB and ALB is also shown.

Table 1: Expected consecutive acceptance $\mathbb{E}[K]$ for SpecED. BOA Constrictor is evaluated on CMS JetNtuple-116 and OctAttention on the ModelNet40 test split. Entries are mean $\pm$ standard deviation over three seeds. A moderate ALB budget of $(6, 6, 4, 2)$ achieves theoretical speedup $1 + \mathbb{E}[K]$ of $2.41\times$ and $1.93\times$ on BOA Constrictor and OctAttention respectively.

| Modality | Base model | Search | $K = 1$ | $K = 2$ | $K = 3$ | $K = 4$ |
|---|---|---|---|---|---|---|
| Physics data | BOA Constrictor | ALB $(2, 2, 1, 1)$ | $0.517_{\pm 0.005}$ | $0.849_{\pm 0.012}$ | $1.083_{\pm 0.017}$ | $1.287_{\pm 0.022}$ |
| | BOA Constrictor | ALB $(6, 6, 4, 2)$ | $0.550_{\pm 0.004}$ | $0.909_{\pm 0.010}$ | $1.173_{\pm 0.013}$ | $1.408_{\pm 0.016}$ |
| | BOA Constrictor | exhaustive oracle | $0.550_{\pm 0.004}$ | $0.914_{\pm 0.010}$ | $1.208_{\pm 0.014}$ | $1.478_{\pm 0.018}$ |
| Point cloud | OctAttention | ALB $(2, 2, 1, 1)$ | $0.363_{\pm 0.005}$ | $0.499_{\pm 0.010}$ | $0.554_{\pm 0.012}$ | $0.582_{\pm 0.014}$ |
| | OctAttention | ALB $(6, 6, 4, 2)$ | $0.548_{\pm 0.007}$ | $0.783_{\pm 0.015}$ | $0.885_{\pm 0.020}$ | $0.932_{\pm 0.022}$ |
| | OctAttention | exhaustive oracle | $0.557_{\pm 0.009}$ | $0.820_{\pm 0.021}$ | $0.964_{\pm 0.026}$ | $1.054_{\pm 0.030}$ |

**Budgeted Speculation Tree Verification.** The branching process induces combinatorial growth in candidate states, with worst-case complexity exponential in the number of draft heads $M$ and depth $k$. To make speculation tractable, we introduce budgeted tree search where static leaf budgeting (SLB) and annealed leaf budgeting (ALB) limit the number of active states after each expansion.

To constrain per-depth complexity to a constant $B$ independent of $K$, SLB caps the number of active states at $B$ after each expansion. For example, Fig. 3 shows SLB with a budget of 3, where only the first three candidates are retained after each expansion. Meanwhile, to improve computational efficiency, ALB assigns larger budgets to early lookahead steps and gradually reduces them with depth, reflecting that early speculative states are more reliable, while deeper branches accumulate errors and provide diminishing returns. One possible ALB strategy is illustrated in Fig. 4.

MHLD consists of batched neural-network evaluations, and is GPU-friendly. Speculative branching and verification involve irregular tree-based state management, better suited to CPUs. This complementary compute profile enables concurrent GPU drafting and CPU state management.

**Optimization Strategy.** Motivated by Corollary 1, we train the draft models to approximate the ground-truth entropy distribution by minimizing the KL divergence between the draft distribution $Q$ and the target distribution $P$: $\mathcal{L}_{\text{speculation}} = \mathbb{E}\big[D_{\mathrm{KL}}(P \,\|\, Q)\big]$. We freeze the base entropy model while training the draft models. This preserves baseline compression performance while enabling acceleration through speculation.

## 5 Experimental Results

### 5.1 Experiment setup

We consider various modalities in lossless compression, including point cloud (OctAttention (Fu et al., 2022)), physics data (BOA Constrictor (Gupta et al., 2026)) and text (LLMZip (Valmeekam et al., 2023)). We further show that SpecED can be extended to lossy codecs, by applying it to audio (Encodec (Défossez et al.)) and video (VCT (Mentzer et al., 2022)). The detailed experiment setups can be found in Appendix B. We report theoretical speedups via $\mathbb{E}[K]$, the expected number of consecutively accepted draft tokens per draft-and-verify iteration. The corresponding wall-time of entropy model scales as $t = \frac{t_{\text{base}}}{1+\mathbb{E}[K]}$, where $t_{\text{base}}$ denotes the baseline decoding time. We also include results on end-to-end wall-time.

### 5.2 Lossless compression

**High-energy physics data and point clouds.** In Table 1, we present $\mathbb{E}[K]$ for different speculation depths with a selection of ALB strategies, observing a theoretical speedup of $2.41\times$ in BOA Constrictor and $1.93\times$ in OctAttention. The significant difference in $\mathbb{E}[K]$ between strategies confirms the necessity and effectiveness of ALB, with a limited budget $(6, 6, 4, 2)$ approaching the oracle speedup.

Fig. 5 illustrates the empirical speedup in end-to-end latency. While the speedup in entropy model roughly follows $1 + \mathbb{E}[K]$, the proportion of latency of entropy models varies with different schemes. Hence, the effectiveness of SpecED is reduced for schemes where entropy coding is not a bottleneck.

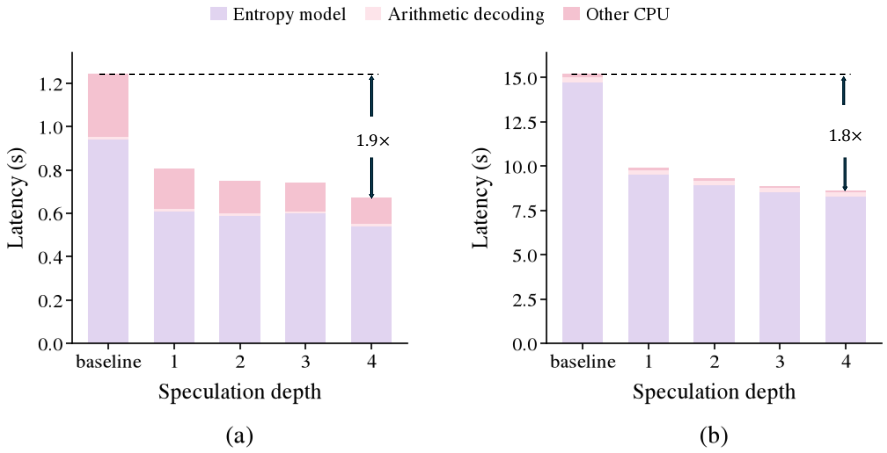



Figure 5: End-to-end latency of SpecED on (a) BOA Constrictor, (b) OctAttention with ALB of schedule $(6, 6, 4, 2)$. The latency is given per $10^4$ symbols. Speculation yields a $1.5\times$ - $1.9\times$ speedup, and depth 4 is fastest on both models, following the same trend as Table 1.

Table 2: Expected consecutive acceptance $\mathbb{E}[K]$ and end-to-end latency speedup for SpecED on text compression baseline LLMzip. Entries are mean $\pm$ standard deviation. The average theoretical speedup $1 + \mathbb{E}[K]$ achieves up to $2.27\times$ at $K{=}7$, with most of the gain realized by $K{=}3$ except on enwik8. Average empirical latency speedup yields $1.78\times$ at $K{=}7$.

| Metric | Dataset | $K = 1$ | $K = 3$ | $K = 5$ | $K = 7$ |
|---|---|---|---|---|---|
| $\mathbb{E}[K]$ | text8 | $0.521_{\pm 0.033}$ | $0.867_{\pm 0.089}$ | $0.941_{\pm 0.115}$ | $0.952_{\pm 0.116}$ |
| | enwik8 | $0.670_{\pm 0.149}$ | $1.423_{\pm 0.550}$ | $1.806_{\pm 0.887}$ | $2.106_{\pm 1.239}$ |
| | wikitext2 | $0.566_{\pm 0.061}$ | $1.012_{\pm 0.192}$ | $1.123_{\pm 0.255}$ | $1.150_{\pm 0.302}$ |
| | gutenberg | $0.486_{\pm 0.002}$ | $0.798_{\pm 0.034}$ | $0.852_{\pm 0.041}$ | $0.856_{\pm 0.039}$ |
| | **macro-average** | $\mathbf{0.561_{\pm 0.080}}$ | $\mathbf{1.025_{\pm 0.280}}$ | $\mathbf{1.181_{\pm 0.432}}$ | $\mathbf{1.266_{\pm 0.573}}$ |
| Latency speedup | text8 | $1.308_{\pm 0.030}$ | $1.557_{\pm 0.076}$ | $1.531_{\pm 0.134}$ | $1.496_{\pm 0.138}$ |
| | enwik8 | $1.409_{\pm 0.105}$ | $1.963_{\pm 0.407}$ | $2.259_{\pm 0.603}$ | $2.465_{\pm 0.868}$ |
| | wikitext2 | $1.335_{\pm 0.062}$ | $1.665_{\pm 0.168}$ | $1.713_{\pm 0.219}$ | $1.665_{\pm 0.183}$ |
| | gutenberg | $1.268_{\pm 0.024}$ | $1.478_{\pm 0.045}$ | $1.496_{\pm 0.034}$ | $1.505_{\pm 0.067}$ |
| | **macro-average** | $\mathbf{1.330_{\pm 0.059}}$ | $\mathbf{1.666_{\pm 0.212}}$ | $\mathbf{1.750_{\pm 0.353}}$ | $\mathbf{1.783_{\pm 0.461}}$ |

We also note that BOA Constrictor is constituted with Mamba (Gu & Dao, 2024), and a concurrent verification of $K$ tokens by parallel scan admit latency of $O(\log K)$, hence the speedup in entropy model of BOA Constrictor is not guaranteed to be monotonic to speculation depth.

**Text.** In Table 2, we observe that speedups vary greatly with (entropy of) the data, similar to in speculative decoding, as suggested by Theorem 1. End-to-end latency loosely follows the theoretical prediction of $1 + \mathbb{E}[K]$. This is further studied in Fig. 6, where we see that arithmetic decoding costs substantially more with speculation depth, further confirming the necessity of establishing a budgeted verification strategy.

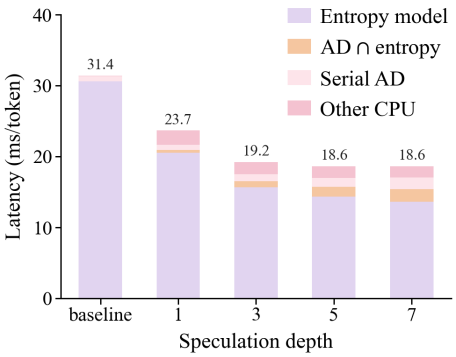
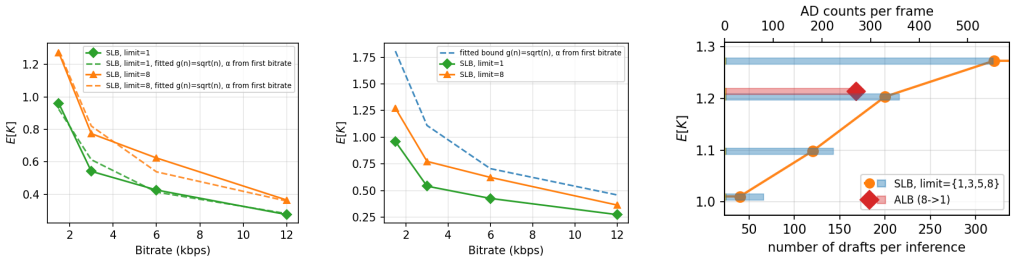



Figure 6: Average end-to-end latency of LLMzip as a function of the speculation depth $K$ with decomposition in modules.

### 5.3 Extension to lossy compression

**Theoretical prediction.** While SpecED is directly applicable on lossy compression schemes with autoregressive entropy models, Corollary 1 suggests an inherit limitation in high-rate regime. In




(a) (b) (c)

Figure 7: Audio experiments: (a) Audio empirical speedups and Corollary 1 based predictions. (b) general performance and Corollary 1 based indicative bounds, (c) speedups for different strategies of SLB/ALB. Arithmetic decoding counts are in bar chart. Frame rate: 75Hz, $M = 4, K = 10$.

Fig. 7a, we plot speedup as a function of the compressed bit-rate in Encodec. We instantiate the $o(n)$ term in Corollary 1 as $\sqrt{n}$, and estimate its coefficient $\alpha$ via least squares on the 1.5 kbps setting. Fixing $\alpha$, we obtain theoretical predictions (dashed lines) for other bitrates ($\{3, 6, 12\}$ kbps). The predictions closely match empirical trends, indicating that the correction term remains approximately constant across bit-rates when using a shared pretrained entropy model (as in Encodec). The empirical agreement with Corollary 1 suggests that, under a fixed entropy model, higher-order corrections to $\mathbb{E}[K]$ are largely invariant to bit-rate, enabling reliable extrapolation from a single operating point.

**Audio.** In Fig. 7b, we observe a theoretical speedup of 1.30 - 2.25× in Encodec, across a range of bitrates from 1.5 to 12kbps, with SLB strategy. Achievable speedup decreases superlinearly with bit-rate, reflecting the diminishing predictability of high-entropy symbols. Fig. 8 quantifies the effect of state error accumulation. Under an oracle setting with access to the ground-truth state, performance degrades slowly with depth. In contrast, practical implementations exhibit a sharp drop due to state contamination. This effect is pronounced even at the first speculative step, reflecting the sequential dependence of arithmetic decoding within vector-latent components. State drift emerges as a primary bottleneck: even small discrepancies in the decoder state propagate through the sequential arithmetic updates, leading to rapid degradation in acceptance. This verifies the necessity for multi-state speculation strategies.

In Fig. 7c, we examine the trade-off between speedup and energy. While SLB induces a direct trade-off, ALB achieves more efficient energy utilization, though with diminishing marginal returns as tree size increases. The performance of SLB is upper bounded by decoding with the ground-truth state. Empirically, the 1.5 kbps-fitted prediction of oracle speculation offers a tighter indicative bound across different tree configurations in Fig. 7b.

We defer the latency on Encodec, results on video (VCT) and further discussion on the dependence of speculation acceptance on entropy of symbols to Appendix D.

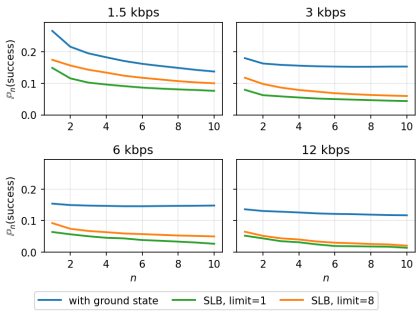



Figure 8: Speculation success rate against speculation depth on Encodec.

## 6 Conclusion

We present SpecED, a model-agnostic speculative entropy decoding framework that accelerates autoregressive entropy models in learned lossless compression. By combining multi-head lookahead drafting, speculative state branching, and budgeted speculation-tree verification, SpecED mitigates the sequential bottleneck of AR decoding without modifying the underlying bitstream or base model. We further provide a Rényi-divergence-based analysis of speculation success and derive a practical KL-based objective for draft-model design. Experiments across a variety of codecs demonstrate consistent decoding speedups while validating the trade-offs between draft accuracy and efficiency. For future work, an important direction is the design of principled selection strategies under budget constraints, as well as improved draft architectures that better balance accuracy and cost. Exploring more efficient verification pipelines and tighter theoretical characterizations of multi-step speculation and branching policies remain open problems.

### AI USE STATEMENT

In this work, we used generative AI tools to help implement the proposed method. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis or interpret results.

Additionally, we used generative AI tools for brainstorming and suggesting improvements on readability. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by 3 authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

We open-source both lossless and lossy variants of speculative entropy decoding library for different modalities to facilitate future research and development for learned compression (which is included in the supplementary materials and to be released publicly after review).

## REFERENCES

Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *ICLR 2018-6th International Conference on Learning Representations*, 2018.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. In *Forty-first International Conference on Machine Learning*.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Herman Chernoff. A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations. *The Annals of Mathematical Statistics*, 23(4):493 – 507, 1952. doi: 10.1214/aoms/1177729330. URL https://doi.org/10.1214/aoms/1177729330.

J. Cleary and I. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396–402, 1984. doi: 10.1109/TCOM.1984.1096090.

Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *Transactions on Machine Learning Research*.

Gregoire Deletang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, et al. Language modeling is compression. In *The Twelfth International Conference on Learning Representations*.

Jingcheng Deng, Zhongtao Jiang, Liang Pang, Zihao Wei, Liwei Chen, Kun Xu, Yang Song, Huawei Shen, and Xueqi Cheng. Following the autoregressive nature of llm embeddings via compression and alignment. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12672–12688, 2025.

Chunyang Fu, Ge Li, Rui Song, Wei Gao, and Shan Liu. Octattention: Octree-based large-scale contexts model for point cloud compression. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 625–633, 2022.

Mohit Goyal, Kedar Tatwawadi, Shubham Chandak, and Idoia Ochoa. Deepzip: Lossless data compression using recurrent neural networks. In *2019 Data Compression Conference (DCC)*, pp. 575–575, 2019. doi: 10.1109/DCC.2019.00087.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Akshat Gupta, Caterina Doglioni, and Thomas J Elliott. Boa constrictor: a mamba-based lossless compressor for scientific data. *Machine Learning: Science and Technology*, 7(3):035014, 2026.

Minghao Han, Shiyin Jiang, Shengxi Li, Xin Deng, Mai Xu, Ce Zhu, and Shuhang Gu. Causal context adjustment loss for learned image compression. *Advances in Neural Information Processing Systems*, 37:133231–133253, 2024.

Dailan He, Yaoyan Zheng, Baocheng Sun, Yan Wang, and Hongwei Qin. Checkerboard context model for efficient learned image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14771–14780, June 2021.

Dailan He, Ziming Yang, Weikun Peng, Rui Ma, Hongwei Qin, and Yan Wang. Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5718–5727, June 2022.

Feng Jiang, Wen Tao, Shaohui Liu, Jie Ren, Xun Guo, and Debin Zhao. An end-to-end compression framework based on convolutional neural networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):3007–3018, 2018. doi: 10.1109/TCSVT.2017.2734838.

Szymon Kobus and Deniz Gunduz. Speculative sampling via exponential races. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18189–18204, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.936. URL https://aclanthology.org/2025.findings-acl.936/.

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

Jiahao Li, Bin Li, and Yan Lu. Deep contextual video compression. *Advances in Neural Information Processing Systems*, 34:18114–18125, 2021.

Jiahao Li, Bin Li, and Yan Lu. Neural video compression with diverse contexts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22616–22626, June 2023.

Ziguang Li, Chao Huang, Xuliang Wang, Haibo Hu, Cole Wyeth, Dongbo Bu, Quan Yu, Wen Gao, Xingwu Liu, and Ming Li. Lossless data compression by large models. *Nature Machine Intelligence*, 7(5):794–799, 2025.

Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Autotimes: Autoregressive time series forecasters via large language models. *Advances in Neural Information Processing Systems*, 37:122154–122184, 2024.

Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Guo Lu, Xiaoyun Zhang, Wanli Ouyang, Li Chen, Zhiyong Gao, and Dong Xu. An end-to-end learning framework for video compression. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3292–3308, 2020.

Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Practical full resolution learned lossless image compression. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10621–10630, 2019. doi: 10.1109/CVPR.2019.01088.

Fabian Mentzer, George Toderici, David Minnen, Sergi Caelles, Sung Jin Hwang, Mario Lucic, and Eirikur Agustsson. Vct: A video compression transformer. In *Advances in Neural Information Processing Systems*, 2022.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 932–949, 2024.

Fazal Mittu, Yihuan Bu, Akshat Gupta, Ashok Devireddy, Alp Eren Ozdarendeli, Anant Singh, and Gopala Anumanchipalli. Finezip: Pushing the limits of large language models for practical lossless text compression. *arXiv preprint arXiv:2409.17141*, 2024.

Dat Thanh Nguyen, Maurice Quach, Giuseppe Valenzise, and Pierre Duhamel. Learning-based lossless compression of 3d point cloud geometry. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4220–4224. IEEE, 2021a.

Dat Thanh Nguyen, Maurice Quach, Giuseppe Valenzise, and Pierre Duhamel. Multiscale deep context modeling for lossless point cloud geometry compression. In *2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pp. 1–6. IEEE, 2021b.

C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3): 379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

James Townsend, Thomas Bird, and David Barber. Practical lossless compression with latent variables using bits back coding. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryE98iR5tm`.

Chandra Shekhara Kaushik Valmeekam, Krishna Narayanan, Dileep Kalathil, Jean-Francois Chamberland, and Srinivas Shakkottai. Llmzip: Lossless text compression using large language models. *arXiv preprint arXiv:2306.04050*, 2023.

Ian H Witten, Radford M Neal, and John G Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 7655–7671, 2024.

Desai Xie, Zhan Xu, Yicong Hong, Hao Tan, Difan Liu, Feng Liu, Arie Kaufman, and Yang Zhou. Progressive autoregressive video diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 6322–6332, 2025.

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.

J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977. doi: 10.1109/TIT.1977.1055714.

## APPENDIX CONTENTS


- A. **Proofs**
  - A.1. **Proof of Theorem 1**
  - A.2. **Proof of Corollary 1**
- B. **Baseline implementation**
  - B.1. **Lossless compression**
    - B.1.1. **Scientific data**
    - B.1.2. **Point cloud**
    - B.1.3. **Text**
  - B.2. **Lossy compression**
    - B.2.1. **Audio**
    - B.2.2. **Video**
- C. **Method implementation**
- D. **Additional experiments**
- E. **Limitations**


# A PROOFS

**Outline.** We prove Theorem 1, Corollary 1, and the identification of the theorem's exponent with Chernoff information. We first sketch a brief outline:

1. Arithmetic decoding recovers the block only when the codepoint lies in both the encoder interval and the draft interval. The conditional success probability is therefore at most $\min(1, Q/P)$.
2. The inequality $\min(1, e^{-x}) \le e^{-\lambda x}$ converts the bound into a Rényi divergence of the two joints. Optimizing the order produces the Chernoff information. This is Theorem 1. The source may be autoregressive, as the argument uses only the block joints.
3. If the source and the draft are memoryless, the joint Chernoff information equals $n$ times the single-letter Chernoff information.
4. As the single-letter KL divergence $\delta$ tends to 0, the single-letter Chernoff information equals $\delta/4 + o(\delta)$. Steps 2–4 give Corollary 1.

**Notation.** $\log$ is the natural logarithm, and every alphabet is finite. For $\alpha \in (0, 1)$ the Rényi divergence is

$$D_\alpha(P\|Q) = \frac{1}{\alpha - 1} \log \sum_z P(z)^\alpha Q(z)^{1-\alpha}, \tag{9}$$

with $D_\alpha = \infty$ if the sum is zero. For $\lambda \in (0, 1)$ this rearranges to

$$\lambda\, D_{1-\lambda}(P\|Q) = -\log \sum_z P(z)^{1-\lambda} Q(z)^\lambda. \tag{10}$$

The Chernoff information (Chernoff, 1952) is the same supremum that appears in Theorem 1:

$$C(P, Q) := \sup_{\lambda \in (0,1)} \lambda\, D_{1-\lambda}(P\|Q). \tag{11}$$

## A.1 PROOF OF THEOREM 1

Let $P$ and $Q$ be the joints of one $n$-symbol block. Arithmetic coding under $P$, in a fixed symbol order, assigns to each block $z$ an interval $I_P(z) \subset [0, 1)$ of length $P(z)$. The draft intervals $I_Q(z)$ likewise partition $[0, 1)$ and have length $Q(z)$. Draw $Z \sim P$ and, given $Z = z$, draw the codepoint $X$ uniformly from $I_P(z)$. Decoding under $Q$ returns $z$ if and only if $X \in I_Q(z)$, so

$$\mathbb{P}(\text{success} \mid Z = z) = \frac{|I_P(z) \cap I_Q(z)|}{P(z)} \tag{12}$$

whenever $P(z) > 0$. This is the success event of the theorem: one draft trajectory reproduces the block.

**Lemma 1** (Length bound)**.** *If $P(z) > 0$, then* $\mathbb{P}(\text{success} \mid Z = z) \le \min\big(1, Q(z)/P(z)\big)$.

*Proof.* The intersection of two intervals is no longer than the shorter one, so $|I_P(z) \cap I_Q(z)| \le \min\big(P(z), Q(z)\big)$. Divide by $P(z)$. If $Q(z) = 0$, the draft interval is empty and both sides are zero. □

**Lemma 2.** *For every $\lambda \in [0, 1]$ and every $x \in \mathbb{R}$,* $\min(1, e^{-x}) \le e^{-\lambda x}$.

*Proof.* If $x \ge 0$, then $\min(1, e^{-x}) = e^{-x} \le e^{-\lambda x}$ because $\lambda \le 1$. If $x < 0$, then $\min(1, e^{-x}) = 1$ and $e^{-\lambda x} = e^{\lambda |x|} \ge 1$. □

Apply Lemma 1 and take the expectation under $Z \sim P$. On $\{Q = 0 < P\}$ both the length bound and the summand below are zero, so those atoms may be included by continuity. For every $\lambda \in (0, 1)$,

$$\begin{aligned} \mathbb{P}(\text{success}) &\le \mathbb{E}_P\big[\min(1, Q/P)\big] \\ &\le \mathbb{E}_P\big[(Q/P)^\lambda\big] = \sum_z P(z)^{1-\lambda} Q(z)^\lambda. \end{aligned} \tag{13}$$

The second step is Lemma 2 with $x = \log(P/Q)$. Identity equation 10 rewrites the sum as $\exp\big(-\lambda D_{1-\lambda}(P\|Q)\big)$, where both sides are zero if the sum vanishes. Therefore

$$\mathbb{P}(\text{success}) \le \exp\big(-\lambda\, D_{1-\lambda}(P\|Q)\big) \qquad \text{for every } \lambda \in (0, 1). \tag{14}$$

The right-hand side is minimized by maximizing the exponent. By definition equation 11 that maximum is the Chernoff information of the joints, and

$$\mathbb{P}(\text{success}) \le \exp\big(-C(P, Q)\big), \tag{15}$$

which is precisely Theorem 1.

### A.2 Proof of Corollary 1

The corollary adds a memoryless assumption: $P = P_1^{\otimes n}$ and $Q = Q_1^{\otimes n}$, with per-step divergence $\delta = D_{\mathrm{KL}}(P_1\|Q_1) \to 0$.

**Lemma 3** (Product formula)**.** *For every $\lambda \in (0, 1)$, $D_{1-\lambda}(P\|Q) = n\, D_{1-\lambda}(P_1\|Q_1)$, and therefore $C(P, Q) = n\, C(P_1, Q_1)$.*

*Proof.* Independence factors the sum in equation 10:

$$\sum_{z_{1:n}} P(z_{1:n})^{1-\lambda} Q(z_{1:n})^\lambda = \Big(\sum_a P_1(a)^{1-\lambda} Q_1(a)^\lambda\Big)^n. \tag{16}$$

Take $-\lambda^{-1} \log$ of both sides. Multiplying by $\lambda$ and taking the supremum over $\lambda \in (0, 1)$ gives the claim for $C$. □

**Lemma 4** (Local expansion)**.** *Let $P_1, Q_1$ be distributions on a fixed finite alphabet with $P_1(a) \ge m > 0$ on their common support, and set $\varepsilon_a = (Q_1(a) - P_1(a))/P_1(a)$. As $\delta = D_{\mathrm{KL}}(P_1\|Q_1) \to 0$,*

$$C(P_1, Q_1) = \frac{1}{4}\delta + o(\delta). \tag{17}$$

*In particular $C(P_1, Q_1)/\delta \to 1/4$.*

*Proof.* Pinsker's inequality gives $\|Q_1 - P_1\|_1 \le \sqrt{2\delta}$. Thus $|\varepsilon_a| \le m^{-1}\|Q_1 - P_1\|_1 \to 0$, and it is enough to expand in $\|\varepsilon\|_\infty$. Write $\chi^2 = \sum_a P_1(a)\varepsilon_a^2$. The identity $\sum_a P_1(a)\varepsilon_a = 0$ holds because both distributions sum to one.

For the KL divergence, Taylor's theorem with Lagrange remainder yields, for $|\varepsilon| \le 1/2$ and some $\xi$ between 0 and $\varepsilon$,

$$\log(1+\varepsilon) = \varepsilon - \frac{\varepsilon^2}{2} + \frac{\varepsilon^3}{3(1+\xi)^3}. \tag{18}$$

Since $1+\xi \ge 1/2$, the remainder is at most $(8/3)|\varepsilon|^3$. Therefore

$$\delta = -\sum_a P_1(a)\log(1+\varepsilon_a) = \frac{1}{2}\chi^2 + r, \qquad |r| \le \frac{8}{3}\|\varepsilon\|_\infty \chi^2. \tag{19}$$

The linear term cancelled. Hence $\delta = \chi^2/2 + o(\chi^2)$ and $\chi^2 = 2\delta + o(\delta)$ as $\|\varepsilon\|_\infty \to 0$.

For the Chernoff information, expand $(1+\varepsilon)^\lambda$ the same way. For $|\varepsilon| \le 1/2$, $\lambda \in (0,1)$, and some $\xi$ between 0 and $\varepsilon$,

$$(1+\varepsilon)^\lambda = 1 + \lambda\varepsilon + \frac{\lambda(\lambda-1)}{2}\varepsilon^2 + \frac{\lambda(\lambda-1)(\lambda-2)}{6}(1+\xi)^{\lambda-3}\varepsilon^3. \tag{20}$$

The coefficient of $\varepsilon^3$ has absolute value at most $1/3$, and $(1+\xi)^{\lambda-3} \le 2^{3-\lambda} \le 8$, so the remainder is at most $(8/3)|\varepsilon|^3$ uniformly in $\lambda$. Summing against $P_1$ gives

$$\sum_a P_1(a)^{1-\lambda}Q_1(a)^\lambda = 1 + \frac{\lambda(\lambda-1)}{2}\chi^2 + \theta_\lambda, \qquad |\theta_\lambda| \le \frac{8}{3}\|\varepsilon\|_\infty \chi^2. \tag{21}$$

Call the right-hand perturbation $u_\lambda$. Then $|\lambda(\lambda-1)/2| \le 1/8$, so $|u_\lambda| \le 1/2$ for all $\lambda$ once $\|\varepsilon\|_\infty$ is small. For such $u$, $-\log(1+u) = -u + \rho(u)$ with $|\rho(u)| \le 2u^2$, because the Lagrange remainder of $-\log(1+u)$ is $u^2/(2(1+\eta)^2)$ and $1+\eta \ge 1/2$. Substituting $-u_\lambda = \lambda(1-\lambda)\chi^2/2 - \theta_\lambda$ produces

$$-\log\sum_a P_1(a)^{1-\lambda}Q_1(a)^\lambda = \frac{\lambda(1-\lambda)}{2}\chi^2 + r_\lambda, \tag{22}$$

where $\sup_\lambda |r_\lambda| = o(\chi^2)$. The coefficient $\lambda(1-\lambda)/2$ has maximum $1/8$ at $\lambda = 1/2$, and changing the objective by a uniform $o(\chi^2)$ changes its supremum by $o(\chi^2)$. Hence $C(P_1,Q_1) = \chi^2/8 + o(\chi^2)$. Insert $\chi^2 = 2\delta + o(\delta)$ to obtain equation 17. The ratio of the two sides tends to $1/4$ because the error is $o(\delta)$. □

Lemma 3 and Lemma 4 give

$$C(P,Q) = n\,C(P_1,Q_1) = \frac{n}{4}\delta + o(n\delta). \tag{23}$$

The little-o depends only on the single-letter pair, so $n$ times it is $o(n\delta)$ as $\delta \to 0$, for every block length $n$. Theorem 1 is $\mathbb{P}(\text{success}) \le \exp(-C(P,Q))$, hence

$$\mathbb{P}(\text{success}) \le \exp\Big(-\frac{n}{4}\delta + o(n\delta)\Big). \tag{24}$$

As $\delta \to 0$ one has $o(n\delta) = o(n)$, which is equation 7.

# B Baseline implementation

Since SpecED is plug-and-play and can be integrated into codecs across different modalities and application scenarios, this section details the baseline implementations and how SpecED is integrated into each codec.

## B.1 Lossless compression

### B.1.1 Scientific data

For lossless compression of CMS particle-physics byte streams, we apply our scheme to BOA Constrictor (Gupta et al., 2026). BOA Constrictor applies a frozen Mamba entropy model mapping the previous byte and its recurrent state to a 256-way categorical, and an arithmetic coder that encode the source through the distribution into the bitstream. SpecED attaches a standalone draft that, from a fixed 32-byte context, proposes four future bytes with four heads in one forward. Detailed setup can be found at Table 3.

Table 3: Details and hyperparameters for SpecED on BOA Constrictor, trained on an NVIDIA RTX 3090.

| Component | Configuration |
|---|---|
| Codec | BOA Constrictor (Gupta et al., 2026) |
| Source | CMS Open Data, DOI 10.7483/OPENDATA.CMS.RY2V.T797 |
| Alphabet | 256 bytes |
| Train / validation / test | `JetNtuple_100.h5` / `104` / `116` |
| Events | 2,729 / 1,071 / 540, source-file and event-ID disjoint |
| Training windows | 50,803,302 valid drafted starts, sampled uniformly |
| Test starts | all 9,895,952 held-out drafted starts |
| Base entropy model | Official V1.3.1 Mamba-v1, frozen |
| Width / depth | 128 / 1 layer |
| Checkpoint | `cms_experiment_final_model_fp32.pt` |
| Output | 256-way categorical per recurrent step |
| Draft entropy model | Standalone residual MLP |
| Context | 32 bytes |
| Embedding / width / depth | 16 / 128 / 3 residual blocks |
| Heads $\times$ horizon | $M{=}4$, $H{=}4$ logits over 256 |
| Training objective | Best-of-heads cross-entropy $+\,0.05$ KL to the base |
| Optimizer | AdamW, learning rate $3 \times 10^{-4}$, weight decay $10^{-4}$ |
| Gradient clipping | $\ell_2 = 1$ |
| Batch size / epochs | 256 / 5 (992,252 steps; $2.54 \times 10^8$ examples per seed) |
| Checkpoint | Lowest validation loss |
| Seeds | 20260907, 20260908, 20260909 |
| Search | ALB $(2, 2, 1, 1)$ and $(6, 6, 4, 2)$; exhaustive oracle $4^4{=}256$ chains |
| Oracle subsample | $10^5$ evenly spaced starts |
| Arithmetic coding | `constriction` 0.4.1 |
| Guard symbols | 128 uniform symbols per window, excluded from $\mathbb{E}[K]$ |
| Verification | One fused step $+\,\lceil \log_2 K \rceil$ combine rounds on $K$ devices |

**Base entropy model** We use the official V1.3.1 release, a 128-dimensional one-layer Mamba-v1 (Gu & Dao, 2024), fp32 checkpoint. One recurrent step emits a categorical over 256 bytes. The base is frozen during training. Arithmetic coding uses `constriction==0.4.1`. Each evaluated window appends 128 uniform guard symbols, which are excluded from $\mathbb{E}[K]$.

**Draft entropy model** The design of the draft model is as follow. A 16-dimensional embedding of the 32-byte context is flattened into width 128. Three residual blocks follow, each LayerNorm, $\mathrm{Linear}(w, 2w)$, SiLU, $\mathrm{Linear}(2w, w)$, with a residual connection. A final LayerNorm feeds $M{=}4$ independent linear heads, and each head emits $H{=}4$ logits over the byte vocabulary. We freeze the base and train with a best-of-heads cross-entropy plus KL regularization of $\lambda{=}0.05$ against the frozen teacher. At depth $h$ the search continues only when the draft and the base decode the same symbol. We report budgets $(2, 2, 1, 1)$ and $(6, 6, 4, 2)$ on every held-out start. The exhaustive oracle decodes all $4^4{=}256$ head sequences on $10^5$ evenly spaced starts from the same pool.

Verification of $K$ drafted bytes is a parallel prefix scan, where one fused step plus $\lceil \log_2 K \rceil$ associative combine rounds, with a 19 KB state copy per round. Arithmetic coding runs only on the committed chain.

### B.1.2 Point cloud

For lossless compression of point clouds, we apply our scheme to OctAttention (Fu et al., 2022). In OctAttention, coordinates are quantized and serialized as an octree. A frozen transformer then predicts the next non-empty occupancy from $K{=}4$ parent nodes, and an arithmetic coder writes that symbol. SpecED attaches a standalone draft with six causal trunks that propose the next four occupancies in one forward. Table 4 records the setup.

Table 4: Details and hyperparameters for SpecED on OctAttention, trained on an NVIDIA RTX 3090.

| Component | Configuration |
|---|---|
| Codec | OctAttention (Fu et al., 2022) |
| Dataset | ModelNet40 official splits |
| Quantization | 10-bit isotropic, $q_s = \max /1023$, unique rows |
| Serialization | `GenOctree`, `GenKparentSeq` with $K$=4 parents |
| Alphabet | 255 non-empty occupancies |
| Train / validation / test objects | 8,859 / 984 / 2,468 |
| Validation carve | 10% of the official train split, ordered by SHA-256 |
| Trace length | capped at 2,500 symbols |
| Available test starts | $6.16 \times 10^6$ |
| Base entropy model | Released object checkpoint, 3-layer causal transformer, frozen |
| Checkpoint | `encoder_epoch_00800093.pth` |
| Conditioning | 4 parent nodes |
| Output | 255-way categorical |
| Draft entropy model | Standalone encoder plus six causal trunks |
| Encoder input | Occupancy, mask, level, octant, xyz, $\sin(\pi\,\mathrm{xyz})$ (248-d) |
| Width / MLP depth | 160 / 2, GELU, LayerNorm |
| Windows | $(1, 1, 2, 4, 6, 8)$, one trunk each |
| Heads $\times$ horizon | $M$=6, $H$=4 logits over 255 |
| Copy residual | Parent and previous occupancy, added on trunk 0 |
| Training objective | Min-trunk cross-entropy + mode cross-entropy + 0.2 KL |
| Optimizer | AdamW, learning rate $3 \times 10^{-4}$, weight decay $10^{-4}$ |
| Gradient clipping | $\ell_2 = 1$ |
| Batch size / window / steps | 8 / 128 / 20,000 |
| Checkpoint | Lowest validation loss |
| Seeds | 20260907, 20260908, 20260909 |
| Search | ALB $(2, 2, 1, 1)$ and $(6, 6, 4, 2)$ on $10^5$ evenly spaced starts |
| Exhaustive oracle | $6^4$=1296 chains on 4,096 evenly spaced starts |
| Arithmetic coding | 16-bit range coder with state snapshots |
| Verification | One teacher forward over batch $K$ |

**Base entropy model** We use the authors' released object-point-cloud checkpoint of transformer entropy model. The vocabulary has 255 symbols (non-empty 8-bit occupancy). Points are quantized isotropically to 10 bits, with per-axis minimum offset and scale $q_s = \max /1023$, duplicate rows removed, then converted with the repository `GenOctree` and `GenKparentSeq` ($K$=4 parents). The base is frozen. The original arithmetic coder does not expose states, which is necessary for drafts, hence evaluation uses a 16-bit range coder.

**Draft entropy model** A shared encoder embeds each of the four parents: a 32-dimensional occupancy embedding, an 8-bit occupancy mask, level, octant, normalized coordinates, and $\sin(\pi\,\mathrm{xyz})$. The 248-dimensional concatenation is projected to width 160 with LayerNorm. Six causal windowed MLP trunks, with windows $(1, 1, 2, 4, 6, 8)$, depth 2, and GELU, each emit $H$=4 occupancy logits. A learned copy residual from the parent and the previous occupancy is added on trunk 0 and broadcast over the horizon. The trunks run concurrently and the slowest trunk stays below one teacher forward. We freeze the base and train with the minimum cross-entropy over trunks, an additional cross-entropy to the teacher's mode, and KL regularization of $\lambda$=0.2. Search uses the same ALB rule as BOA, with budgets $(2, 2, 1, 1)$ and $(6, 6, 4, 2)$, on $10^5$ evenly spaced starts. The exhaustive oracle decodes all $6^4$=1296 head sequences on 4,096 evenly spaced starts.

Verification of $K$ drafted occupancies is one teacher forward over a batch of $K$ windows. Arithmetic coding is issued only on the committed chain.

### B.1.3 TEXT

For lossless text compression, we apply our scheme to LLMZip (Valmeekam et al., 2023), which uses a pretrained large language model as an autoregressive probability model. Given the preceding token

Table 5: Details and hyperparameters for SpecED on LLMzip, evaluated on an NVIDIA RTX A6000. Both the base and draft models are frozen pretrained checkpoints; no additional training is performed.

| **Component** | **Configuration** |
|---|---|
| Codec | LLMzip (Valmeekam et al., 2023) |
| Alphabet | LLaMA SentencePiece, 32,000 tokens |
| Arithmetic coding | 24-bit coder, $\mathrm{min_range} = 2$, round-off $10^{-8}$ |
| Context length | Sliding window of 512 tokens |
| Base entropy model $P$ | Pretrained LLaMA-7B, frozen |
| Architecture | 32-layer decoder-only Transformer |
| Hidden width | 4096 |
| Attention heads | 32 |
| Feed-forward expansion | SwiGLU, $\times 4$ |
| Precision / attention | IEEE fp16, scaled dot-product attention |
| Implementation | Hugging Face Transformers |
| Draft entropy model $Q$ | TinyLlama-1.1B, frozen standalone draft |
| Architecture | 22-layer LLaMA, width 2048, 32 heads |
| Quantization | llama.cpp GGUF Q4_0 |
| Tokenizer | Shared 32k vocabulary with $P$ |
| Drafting | Autoregressive; one TinyLlama step per draft symbol |
| Draft temperature | $q_{\mathrm{temp}} = 0.7$ ($P$ is not tempered) |
| Speculation depths | $K \in \{1, 3, 5, 7\}$ |
| Bonus symbol | First mismatch is committed |
| Cache update | $Q$ key–value cache cropped to the accepted prefix |
| Training | None |
| Evaluation corpora | text8, enwik8, WikiText-2, Gutenberg #1342 |
| Windows | Four 20,000-byte slices per corpus |
| text8 / enwik8 offsets | 0, 200k, 800k, 1.5M bytes |
| WikiText-2 / Gutenberg | Equally spaced through the file |
| Tokens per window | First 512 tokens (prefix length 1) |
| Reported statistic | Mean $\pm$ sample standard deviation over four windows |
| Macro-average | Unweighted mean across corpora |

sequence, LLaMA-7B predicts a probability distribution over the next token, which is subsequently used by an arithmetic coder to encode the observed token into a lossless bitstream. During decoding, the same pretrained language model is used to reproduce the conditional probability distributions required by the arithmetic decoder. In our experiments, we retain the pretrained LLaMA-7B model and tokenizer from the original LLMZip implementation and use them without modification. Table 5 summarizes the setup.

**Base entropy model** We use the pretrained 7B LLaMA, a decoder-only Transformer with 32 layers, model width 4096, 32 attention heads, SwiGLU MLP expansion $4\times$, and a 32,000-token SentencePiece vocabulary. Decoding uses Hugging Face Transformers in IEEE fp16 with scaled-dot-product attention (SDPA) on a single NVIDIA RTX A6000. The arithmetic coder is with $\mathrm{min_range} = 2$ and round-off $10^{-8}$. The context is a sliding window of 512 tokens.

**Draft entropy model** We use TinyLlama-1.1B as a standalone draft, loaded as a llama.cpp GGUF Q4_0 graph. TinyLlama is a 22-layer LLaMA-architecture model with width 2048, 32 heads, and the same 32k tokenizer as $P$, so draft and verify operate in a shared discrete alphabet. Draft logits are tempered with $q_{\mathrm{temp}} = 0.7$ before softmax. Each of the $K$ draft symbols is produced by an autoregressive TinyLlama step followed by arithmetic decoding of that step's CDF, i.e. $\hat{x}_{i+k} = \mathrm{AD}(Q(\cdot \mid x_{<i}, \hat{x}_{<k}))$. On an idle A6000 the $K$-step draft chain remains shorter than one LLaMA-7B forward for the horizons we report ($K \in \{1, 3, 5, 7\}$).

### B.2 LOSSY COMPRESSION

#### B.2.1 AUDIO

For audio compression, we apply our scheme on Encodec (Défossez et al.), a neural audio compression scheme that is approximately real-time across a number of compression ratios. In Encodec, a continuous audio signal sequence is converted to discrete indices by Residual Vector Quantization (RVQ). The indices are then arithmetic encoded into a bitstream via a pretrained small language model, which predicts a codebook for each symbol, given the previous symbol sequence. In our experiments, pretrained RVQ codebooks from the original repository are retained and frozen, which are used in encoding and decoding. Table 6 shows the detailed settings.

**Base entropy model** We use the official 24kHz checkpoint of causal language model: a 5-layer streaming Transformer with hidden width 200, 8 attention heads, and feed-forward expansion ×4, operating on the sum of $n_q$ learned codebook embeddings and producing $n_q$ softmax classifiers over 1024 RVQ bins per frame with frame rate of 75Hz, where $n_q$ in $\{2, 4, 8, 16\}$ determines the bitrate $\{1.5, 3, 6, 12\}$ kbps of compressed binary sequence. Attention is causally masked with a finite past context of 3.5 seconds.

**Draft entropy model** We train a standalone lightweight draft model to propose symbol distributions. The design or draft model is with the same $n_q$-way input embedding and sum, followed by four parallel, non-shared temporal backbones. Each backbone is a stack of causal dilated 1-D convolutions with kernel 5, residual connections, GELU, and LayerNorm; dilations grow as $2^l$ over $l = 0 \dots 3$ trunk layers. Each of the four models emits 10 horizon-specific predictors: for every speculative offset s = $1 \dots 10$ and every codebook k, a linear map from trunk width (160) to the RVQ vocabulary (1024). We freeze the parameters of base entropy model and train draft model with a best-of-models cross-entropy objective and KL regularization of $\lambda = 0.05$ against the frozen teacher distribution.

We note that as the official repository implemented an arithmetic coding scheme in Python, the arithmetic coding component is inevitably slow sequentially in CPU. However in practice arithmetic decoding is fully parallelizable among different speculations. To ensure a fair comparison, we use the built-in arithmetic coding implementation in Encodec, while only reporting the maximum wall time of arithmetic decoding, where full parallelization between speculations are possible.

#### B.2.2 VIDEO

For video compression, we consider Video Compression Transformer (VCT) (Mentzer et al., 2022) as the base compression scheme. Different from classical neural video coding models which highly rely on complex hand-crafted components for accurate motion and residual information estimation, VCT simplifies this process by replacing these components with a purely transformer-based temporal entropy model. It leverage a pretrained image autoencoder to independently map each frame $\boldsymbol{x}_i$ into a quantized latent representation $\hat{\boldsymbol{y}}_i$. Then, VCT models the temporal dependencies between frames by predicting the conditional distribution $P(\hat{\boldsymbol{y}}_i|\hat{\boldsymbol{y}}_{i-1}, \hat{\boldsymbol{y}}_{i-2})$ of the current frame $\hat{\boldsymbol{y}}_i$ based on the previous two frames $\hat{\boldsymbol{y}}_{i-1}$ and $\hat{\boldsymbol{y}}_{i-2}$. Finally, $\hat{\boldsymbol{y}}_i$ can be encoded and decoded by arithmetic coding given $P(\hat{\boldsymbol{y}}_i|\hat{\boldsymbol{y}}_{i-1}, \hat{\boldsymbol{y}}_{i-2})$. Detailed setting is summarized in Table 7.

In our experiments, we use pretrained analysis and synthesis transforms from (Ballé et al., 2018) as the image autoencoder to extract $16\times$ downsampled latent $\boldsymbol{y}_i \in \mathbb{R}^{C\times H'\times W'}$ from original frame $\boldsymbol{x}_i \in \mathbb{R}^{3\times H\times W}$ where $H' = \frac{H}{16}$ and $W' = \frac{W}{16}$. $\boldsymbol{y}_i$ is then quantized as $\hat{\boldsymbol{y}}_i$ and spatially partitioned into $p = \frac{H'\times W'}{4\times 4}$ non-overlapping patches with size of $4 \times 4$, yielding 16 tokens of dimension $C$. Each patch is processed in parallel, where the temporal context for each current patch is derived from the corresponding $6 \times 6$ overlapping regions in the previous two frames. Within each patch, VCT predicts entropy parameters of every token in an AR manner. Specifically, the prediction of the $j$-th token is conditioned on the complete temporal context and the previously encoded spatial tokens $\boldsymbol{y}_{i,<j}$.

**Base Entropy Model** Temporal context is derived from two preceding quantized latents, each independently processed by a separate Transformer $T_{\text{sep}}$ (3 layers, hidden width $d/2 = 96$, 16 heads). Their representations are concatenated along the channel dimension and further refined by a joint

Table 6: Details and hyperparameters for SpecED on Encodec audio compression, where model is trained on NVIDIA RTX 3090 GPU and Intel Core i9-10980XE CPU @ 3.00GHz.

| **Component** | **Configuration** |
|---|---|
| Codec | Encodec (Défossez et al.) |
| Sampling rate | 24 kHz |
| Quantization | Residual Vector Quantization (RVQ) |
| RVQ codebooks | Official pretrained codebooks |
| Frame rate | 75 Hz |
| Bitrates | $\{1.5, 3, 6, 12\}$ kbps |
| Number of codebooks $n_q$ | $\{2, 4, 8, 16\}$ (bitrate-dependent) |
| RVQ vocabulary size | 1024 bins per codebook |
| Base entropy model | Official causal language model, 24kHz checkpoint |
| Architecture | 5-layer streaming Transformer |
| Hidden width | 200 |
| Attention heads | 8 |
| Feed-forward expansion | $\times 4$ |
| Output heads | $n_q$ softmax classifiers over 1024 bins |
| Context length | 3.5 s finite causal context ($\approx$ 262 frames at 75 Hz) |
| Draft entropy model | Standalone lightweight draft ensemble |
| Input embedding | Same $n_q$-way codebook embedding and summation |
| Backbones | 4 parallel *independent* temporal backbones |
| Backbone type | Causal dilated 1-D convolutional stacks |
| Kernel size | 5 |
| Trunk layers | 4 residual blocks per backbone; dilations $2^\ell$, $\ell = 0, \ldots, 3$ |
| Trunk width | 160 |
| Activations / normalization | GELU, LayerNorm |
| Residual connections | Used |
| Prediction horizons | 10 speculative offsets, $s = 1, \ldots, 10$ |
| Output heads | Per-model, per-horizon, per-codebook linear predictors ($4 \times 10 \times n_q$ heads) |
| Draft parameters | $\approx 15.57$ M |
| Training objective | Best-of-models cross-entropy + KL to base LM |
| KL weight | $\lambda = 0.05$ |
| Optimizer | AdamW, learning rate $3 \times 10^{-4}$ |
| Batch size | 8 |
| Epochs | 100 (checkpoint = best validation loss) |
| Training data | 500 FSD50K dev shards; validation 100 eval shards |
| Arithmetic coding | Built-in Encodec Python implementation |

Transformer $T_{\text{joint}}$ (2 layers, hidden width $d = 192$, 16 heads) to produce a fused temporal context $z_{\text{joint}}$. Tokens of the current frame are encoded autoregressively by a masked encoder $T_{\text{cur}}$ (5 layers, hidden width $d = 192$, 16 heads), which predicts the mean and scale of a Laplace distribution for each token. The model ensures decoding causality by conditioning the current token on both the complete temporal context $z_{\text{joint}}$ and all spatially preceding tokens within the patch. The integration of spatiotemporal information is achieved through cross-attention layers, where queries are derived from $z_{\text{joint}}$ while keys and values are derived from spatially previous tokens.

**Draft Entropy Model** We train speculative heads, which are added in parallel with base output MLP heads. Each head share the same design as the output head: 3-layer 192-width MLP with LeakyReLU activation. The heads are trained with the loss of the KL Divergence between the Laplace distribution of its output parameter and the ground parameters of the corresponding timestep. A total of 5 heads are trained to predict 5 steps ahead.

## C METHOD IMPLEMENTATION

In practice, draft models in MHLD can be instantiated in two ways, depending on the application scenario: (i) standalone draft models and (ii) on-top speculative heads.

Table 7: Details and hyperparameters for five-head joint finetuning of the video learned compression model, run on NVIDIA RTX A6000 GPU and Intel Xeon Gold 5220R CPU @ 4.00GHz.

| **Component** | **Configuration** |
|---|---|
| Task | Neural video compression with latent-space temporal prediction |
| Input | RGB clips, $256 \times 256$, 3 frames per training step |
| Training / test sources | Vimeo-style triplets for training; UVG clips for validation |
| Batch size | 8 |
| Rate–distortion weights $\lambda_{\mathrm{RD}}$ | $\{10^{-4}, 5{\times}10^{-4}, 2{\times}10^{-3}, 3.5{\times}10^{-3}\}$ |
| Speculative KL weight $\lambda_{\mathrm{KL}}$ | $10^{-4}$ |
| Distortion | MSE in 255-scaled pixel domain |
| KL aggregation | Mean over elements |
| Analysis–synthesis backbone | Convolutional analysis and synthesis transforms |
| Latent channels | $C = 16$ from width $N = 128$ |
| Initialization | Weights initialized from a pretrained image compression checkpoint |
| Temporal entropy model | Transformer stack over quantized latents with causal masking |
| Model dimension | $d_{\mathrm{model}} = 192$ |
| Attention heads | 16 |
| Feed-forward expansion | $\times 4$ |
| Dropout | 0 |
| Temporal context | 2 past latents |
| Encoder / decoder windows | Analysis window 6 blocks; synthesis window 4 blocks |
| Depths $(T_{\mathrm{sep}}, T_{\mathrm{joint}}, T_{\mathrm{cur}})$ | $(3, 2, 5)$ transformer sections |
| Latent normalization | Fixed scale factor 35 on latents fed to entropy |
| Speculative heads | One base predictor plus five speculative predictors (six heads total) |
| Per-head output MLP | Three linear layers on $d_{\mathrm{model}}$ with LeakyReLU |
| Optimizer | Adam, base learning rate $\eta_0 = 1.8 \times 10^{-4}$ |
| Learning-rate schedule | Piecewise: linear warmup to $\eta_0$ over 200 steps;<br>constant $\eta_0$ until step 1200; then $0.8\,\eta_0$ until step 17000;<br>thereafter multiply by 0.9 every 1000 steps, with floor $10^{-5}$ |
| Epochs | 200 |
| Logging / validation | Every 100 steps / every 500 steps |
| Checkpoint selection | Best validation objective |
| Random seed | 1314 |

**Standalone models.** Independent lightweight draft models are trained on the same conditioning inputs as the base entropy model to perform lookahead prediction. This design is fully decoupled and enables parallel execution, allowing draft computation to overlap with decoding when sufficiently lightweight. Its main potential drawback is weaker alignment with the base model, which may reduce acceptance rates and increase sensitivity to draft training quality. It also introduces additional training and deployment overhead.

**On-top speculative heads.** Alternatively, draft models can be implemented as additional output heads on the entropy model, each predicting a future token or distribution at a fixed offset. Shared representations improve alignment with the base model and typically increase acceptance rates, while simplifying training. However, these heads are coupled to the base forward pass and introduce extra latency that cannot be fully parallelized, which may reduce speedup for lightweight models.

In general, the two designs target different regimes. On-top speculative heads are more suitable when the base entropy model is heavy, or the prediction task is difficult, where sharing representations reduces training and deployment overhead, e.g., in video compression. Standalone models are preferable for lightweight entropy models or simpler prediction tasks, where parallel draft computation can effectively hide latency and the additional capacity improves speculation quality, e.g., in audio compression.

Table 8: End-to-end decoding latency of Encodec and SpecED, averaged per 2.0 s clip on the same test set as Section 5. Speculation uses SLB with leaf limit 1. speedup is the ratio of sequential Encodec latency to SpecED latency.

| **Bitrate (kbps)** | $\mathbb{E}[K]$ | **Encodec (ms)** | **SpecED (ms)** | **speedup** |
|---|---|---|---|---|
| 1.5 | 0.96 | 954.9 | 699.7 | 1.36× |
| 3 | 0.54 | 1314.6 | 1125.5 | 1.17× |
| 6 | 0.43 | 1752.9 | 1587.0 | 1.10× |
| 12 | 0.28 | 2804.6 | 2667.9 | 1.05× |

Table 9: Cumulative speculative accuracy of SpecED on VCT over a range of PSNR and speculation depth.

| $\lambda_{\text{train}}$ | PSNR (dB) | $P(K \geq 1)$ | $P(K \geq 2)$ | $P(K \geq 3)$ | $P(K \geq 4)$ | $P(K \geq 5)$ |
|---|---|---|---|---|---|---|
| 0.0001 | 26.78 | 23.89% | 10.71% | 6.65% | 5.30% | 4.55% |
| 0.0005 | 29.82 | 12.57% | 3.57% | 1.84% | 0.78% | 0.28% |
| 0.002 | 33.33 | 4.24% | 0.67% | 0.00% | 0.00% | 0.00% |
| 0.0035 | 33.95 | 2.08% | 0.15% | 0.08% | 0.00% | 0.00% |

## D Additional Experiments

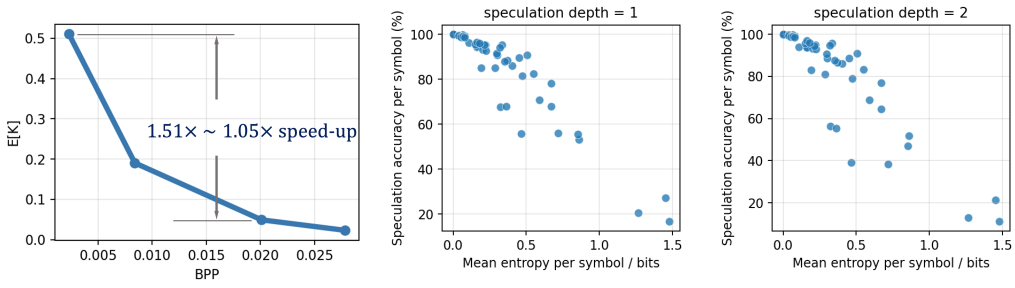
**End-to-end latency on Encodec.** We report end-to-end decoding latency for Encodec on the same test set as the Encodec evaluation in Section 5, averaged per clip of 2.0 s. Speculation uses static leaf budgeting (SLB) with leaf limit 1. As shown in Table 8, the realized speedup decreases with bitrate, from 1.36× at 1.5 kbps to 1.05× at 12 kbps, following the decline in $\mathbb{E}[K]$.

**Video results** A theoretical speedup of 1.05-1.51× is observed in VCT under the low BPP region, as shown in Fig. 9a. Table 9 confirms the exponential decay of cumulative acceptance with speculation depth, and Fig. 9b shows the same superlinear reduction of speedup at high bit-rates observed in audio experiments. Consistently, Fig. 9b, 9c show that per-symbol speculation accuracy is inversely correlated with the corresponding average Shannon entropy. This can be attributed to reduced overlap between draft and target distributions, as the uncertainty of target distribution grows while the capability of draft model stays the same for bitrate-adaptive compression schemes.

We note that one possible variant of SpecED is to relax the constraint of freezing base entropy model and jointly train base model and draft models, with the target function to be a Lagrange multiplier of KL divergence between draft distribution and base model distribution, on the usual rate-distortion loss function. We illustrate the speedup difference between joint and separate training strategy in Fig. 10a. With the freedom to improve near future contexts in the entropy model, allowing joint training yields double speedup compared with freezing base entropy model and training speculative heads alone. In addition, the loss objective biases the entropy model toward longer horizon predictions, which appears to improve generalization to unseen samples (Fig.10b). In Fig. 10c, we also see that by allowing joint finetuning speculative heads with both entropy model and predictor model, we retrieve a similar compression performance for PSNR/BPP ratio.

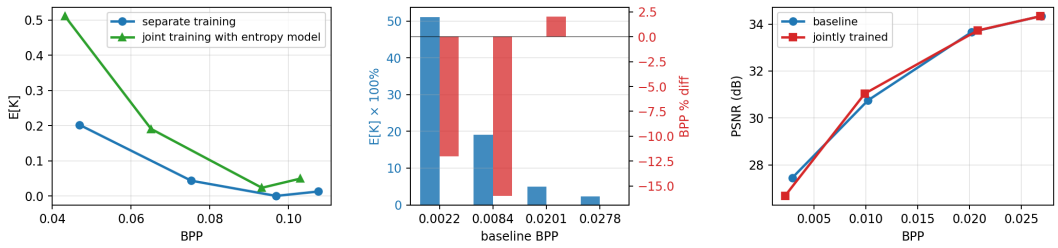
## E Limitations

To enable parallelization and ultimately mitigate the AR bottleneck in decoding, SpecED introduces additional algorithmic and system complexity due to concurrency, similar to existing speculative decoding methods. This overhead could potentially be reduced through further lightweight design and engineering optimizations, which we leave for future work.




(a) (b) (c)

Figure 9: Video experiments: (a) general performance, (b,c) per-symbol acceptance rate vs entropy.




(a) (b) (c)

Figure 10: Video experiments: comparison between joint and separate training strategies, (a) speedup vs. bitrates, (b) bitrate difference between 1) jointly trained entropy-speculative-head model and 2) baseline, speedup achieved by corresponding speculative heads. For fair comparison, the baseline is not finetuned with predictor for a consistent PSNR performance. (c) Rate-distortion performance for finetuned baseline and predictor-entropy-draft jointly trained model.