SYNCHRONIZING PROBABILITIES IN MODEL-DRIVEN LOSSLESS COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

It is well-known in the field of lossless data compression that probabilistic next-symbol prediction can be used to compress sequences of symbols. Deep neural networks are able to capture rich dependencies in data, offering a powerful means of estimating these probabilities and hence an avenue towards more effective compression algorithms. However, both compressor and decompressor must have exactly matching predictions; even small non-deterministic differences (which often happen with learned models due to hardware, software, or computation order) can lead to cascading decoding failures. In this paper, we formalize the problem of prediction mismatch in model-driven compression, and introduce Probability Matching Interval Coding (PMATIC), a model-agnostic algorithm that tolerates bounded prediction mismatch with low overhead. PMATIC works with the predicted probabilities, making it compatible as a drop-in replacement for the arithmetic encoder in model-driven compression tools. We show theoretical correctness and performance bounds for PMATIC, and validate these results on text data. These results confirm that, when paired an advanced prediction model, PMATIC is robust to prediction mismatch while achieving compression rates that out-perform standard modern compression tools.

1 INTRODUCTION

1.1 MODEL-DRIVEN LOSSLESS COMPRESSION

A key task in modern information systems is data compression, the process of reducing the size of text, images, video, or other data so it can be stored and transmitted more efficiently. In lossless compression, the data is encoded into a compact representation from which the original can be decoded exactly, in contrast to lossy compression, which only permits approximate reconstruction. Compression is generally formalized as the problem of encoding a string of discrete symbols drawn from a finite alphabet. In deep learning contexts, these symbols are often referred to as tokens. The choice of symbols is domain-dependent: for text, tokens are typically subword units or characters; for images, they may correspond to pixel intensities, color values, or transformed coefficients; and for other domains, analogous discrete representations are used.

Lossless compression works by exploiting regularities in the data: common patterns are assigned shorter codes, while rare patterns receive longer ones. These regularities may reflect simple statistics, such as symbol frequencies, or more complex and context-dependent structure and even semantic information. From this perspective, any lossless compression method implicitly defines a probabilistic model of the data source, with compression effectiveness depending on how well the model matches the true distribution. Some algorithms make this explicit, using predictive models that estimate the probability of each symbol given its context [Cleary & Witten (1984)] in order to generate the code; we refer to such algorithms as *model-driven*. Others, such as Lempel–Ziv–Welch (LZW), ZIP, or bzip2, achieve their gains through dictionary-building or transforms, but nonetheless rely on an implicit statistical model of the domain.[1]

In model-driven compression, the message is encoded sequentially, and for each symbol the model makes a probabilistic prediction based on the context of the prior symbols so the encoding can more

[1] They can even be used to create explicit predictive models [Delétang et al. (2024)].

efficiently allocate bits to potential outcomes. To convert the predictive model into a compression algorithm, the standard technique is to pair the model with arithmetic coding [Pasco (1976); Rissanen (1976); Guazzo (1980)]. Arithmetic coding represents an entire message as a subinterval of $[0, 1)$, successively narrowing the interval according to the predicted probabilities of each symbol. More probable symbols shrink the interval less and thus yield shorter average descriptions, while less probable symbols shrink it more and thus require more bits. Unlike Huffman coding [Huffman (1952)] (another commonly used technique), arithmetic coding adapts particularly well to changing and context-dependent probabilities for each symbol. If the model closely reflects the true distribution of the data, arithmetic coding yields compression rates approaching the information-theoretic limit. However, it is extremely sensitive to numerical precision, and small deviations can propagate through the algorithm.

Model-driven lossless compression has a long history, arguably going back to Shannon (1948), where Shannon tallies frequencies of characters in English, building first, second, and third order Markov model predictions. The arithmetic coding approach for model-driven compression was discussed by Cleary & Witten (1984) and further developed with a number of statistical or learned predictive models across many domains. Many of these models focus on deriving the prediction model from only the previously seen encoded symbols. In Schmidhuber & Heil (1996), it is clarified that "offline" models are those trained on a separate files and model parameters are shared among all machines responsible for encoding and decoding. In contrast "online" models use the current file to update predictions. Schmidhuber & Heil (1996) use offline neural networks and get competitive compression ratios. Many other works since, Knoll (2025); Cox (2016); Goyal et al. (2018); Bellard (2019); Liu et al. (2019) have used LSTMs and other recurrent neural networks as predictive models. Transformers were used as the predictive model in Bellard (2019; 2021); Mao et al. (2022).

This general arithmetic coding-based lossless compression technique, particularly when paired with modern neural network-driven predictive models, has been shown to have significant promise in numerous domains beyond text compression. These domains include lossless image compression Toderici et al. (2016); Schiopu et al. (2018); Mentzer et al. (2019); Rhee et al. (2022); Chen et al. (2024), compression of large numerical datasets such as time-series power data Ma et al. (2022), and neural network checkpoints Kim & Belyaev (2025).

The incredible success of modern neural networks, particularly transformers, for natural language processing has led to increased interest in using the model-driven approach to create more powerful and context-adaptive codes for natural language compression. Recent work by Delétang et al. (2024) shows that offline model-driven compression using modern models such as Llama 2 or Chinchilla with arithmetic coding can deliver significant improvement over state-of-the-art lossless compression algorithms across domains including text and vision. Concurrently, LLM-driven text compression tools such as LLMZip [Valmeekam et al. (2023)] and llama-zip [Buzanis (2024)] were introduced to take advantage of the capabilities of these advanced models.

1.2 LLM Non-Determinism and Prediction Mismatch

Despite its promise, LLM-driven compression faces serious practical obstacles. For instance, the LLM inference pipeline must run for each token during the encoding and decoding steps, which can make the process prohibitively slow, since large language models are often computationally expensive to execute. This also requires the model, which may contain many gigabytes of parameters, to be stored, thus adding a large overhead cost in memory as well. Recent work, such as has been done to address concerns about the computational performance of LLM-driven compression, such as Mittu et al. (2024) on improving speeds for LLMZip.

Another significant challenge, which we call *prediction mismatch*, arises when compressed data is transmitted between an encoder and decoder running on different machines. As noted in Witten et al. (1987), arithmetic coding with adaptive probability models, "It must be possible for the decoder to produce exactly the same probability distribution in the same context". Achieving this is unexpectedly difficult with modern machine learning models due to *LLM non-determinism*.

Non-determinism in the setting of machine learning and scientific computing means that multiple runs of the same program with identical inputs (and identical random seeds) can produce different outputs [Cooper et al. (2022); Semmelrock et al. (2025)]. One source of non-determinism occurs in GPU hardware: floating point operations which are performed in a different order may result in

different outcomes due to rounding. These small numerical deviations, in a full inference pipeline run, can cascade into large differences in what a model predicts [Shanmugavelu et al. (2025); Chen et al. (2022)]. GPU libraries, like CUDA and cuDNN, state specifically in their documentation that they do not guarantee determinism or reproducibility in many circumstances, such as when different versions or architectures are used [NVIDIA Corporation (2025a;b)]. The effects of non-determinism in CUDA is studied in Eryilmaz et al. (2024) where they note that non-determinism is likely to remain in CUDA because of the runtime benefits CUDA gains through using parallelism. Non-determinism in GPUs have also been examined and explored by Morin & Willetts (2020). Works such as Coakley et al. (2022) and Atil et al. (2025) examined the issue of non-determinism experimentally, finding significant variability, and Schlögl et al. (2023) study its causes.

Applying arithmetic coding directly under these conditions is usually immediately fatal: even subtle differences in the encoder and decoder probability distributions can result in an incorrectly decoded token, which then cascades to the rest of the message as it changes the context of subsequent tokens.

If the mismatch between the encoder and decoder distributions is arbitrarily large, recovery is impossible. However, if the mismatch is known to be small, they can exploit this closeness to reach exact agreement on a third probability distribution. This robustness incurs a cost in compression efficiency: the encoder will generally have to send extra information to ensure agreement, and the agreed probability distribution may be less accurate than the original predictions. We refer to the problem of constructing a shared distribution while minimizing the cost as *probability matching*.

1.3 Contributions

This work introduces the probability matching problem as a framework for addressing prediction mismatch in model-driven lossless compression and proposes *PMATIC* (Probability-Matched Interval Coding) to address it. PMATIC is designed to convert any predictive model into a compression algorithm which is robust to bounded prediction mismatch, and to be a drop-in replacement for arithmetic coding in model-driven compression. This work also shows the following results:

- *Theory:* We prove that PMATIC guarantees correct decoding under a simple and general model of bounded prediction mismatch (Section 2.1), and give theoretical bounds on the cost incurred to ensure this robustness.
- *Practice:* We demonstrate experimentally that LLM-driven compression using PMATIC achieves compression ratios significantly better than current standard methods, while remaining robust to prediction mismatch.

To our knowledge, this is the first work to explicitly address LLM non-determinism and prediction mismatch as fundamental obstacles to model-driven compression. In Section 5, we validate our approach by applying PMATIC with Llama 3.1 as the predictive model on Wikipedia data in the presence of synthetically generated prediction mismatch. PMATIC is a proof of concept that small amounts of variability in LLM computations can be combated in the application of data compression.

2 Problem Statement

Consider the case where an encoder and decoder are using the same model (such as a specific LLM with the same weights) to compute next-token probabilities over an input string $\boldsymbol{x} = x(1)x(2)\dots x(n)$ whose entries $x(i)$ are taken from a finite alphabet $\mathcal{A}$ of possible symbols. We use the following notation: the i-symbol prefix of $\boldsymbol{x}$ is denoted as $\boldsymbol{x}^i := x(1)\dots x(i)$; the set of all finite strings drawn from the alphabet $\mathcal{A}$ is denoted as $\mathcal{A}^* := \bigcup_{i\geq 0} \mathcal{A}^i$.

Typically, an LLM computes its next-token probabilities by computing a real-valued (or, rather, floating-point valued) weight, called a *logit*, for each outcome and then applying the *softmax* function to the vector of logits[2]Let functions $\mathrm{M}^{\mathrm{Enc}}, \mathrm{M}^{\mathrm{Dec}} : \mathcal{A}^* \to \mathbb{R}^{\mathcal{A}}$ take a string of symbols (the context) and return a logit value for each symbol in the alphabet, representing what happens when the model is run for inference at the encoder and decoder ends, respectively. We denote the predic-

[2]For simplicity we use the standard softmax with a 'temperature' parameter of 1.

tions of $\mathrm{M}^{\mathrm{Enc}}, \mathrm{M}^{\mathrm{Dec}}$ for token i (expressed as logit vectors) as

$$\boldsymbol{u}(i) := \mathrm{M}^{\mathrm{Enc}}(\boldsymbol{x}^{i-1}) \quad \text{and} \quad \boldsymbol{v}(i) := \mathrm{M}^{\mathrm{Dec}}(\boldsymbol{x}^{i-1}) \tag{1}$$

which respectively induce probability vectors $\boldsymbol{p}(i) = \mathrm{softmax}(\boldsymbol{u}(i))$ and $\boldsymbol{q}(i) = \mathrm{softmax}(\boldsymbol{v}(i))$, i.e. for any $i \in [n]$ and $k \in \mathcal{A}$,

$$p(i)_k = \mathrm{softmax}(\boldsymbol{u}(i))_k := \frac{e^{u(i)_k}}{\sum_{j\in\mathcal{A}} e^{u(i)_j}} \text{ and } q(i)_k = \mathrm{softmax}(\boldsymbol{v}(i))_k := \frac{e^{v(i)_k}}{\sum_{j\in\mathcal{A}} e^{v(i)_j}}. \tag{2}$$

When the token number i is fixed, we may drop it from the notation for clarity, so that the encoder and decoder return logit vectors $\boldsymbol{u}, \boldsymbol{v}$ which induce probability distributions $\boldsymbol{p}, \boldsymbol{q}$ respectively.

We denote the encoding and decoding algorithms (also called compressing and decompressing, respectively) as functions whose operation depends on an LLM model ($\mathrm{M}^{\mathrm{Enc}}$ and $\mathrm{M}^{\mathrm{Dec}}$ respectively) as well as on more traditional inputs. Specifically, the encoder takes LLM $\mathrm{M}^{\mathrm{Enc}}$ and input token string $\boldsymbol{x}$ and returns a bitstring $\boldsymbol{b}$ which is the encoded input. Then, the decoder takes $\boldsymbol{b}$ and its own LLM $\mathrm{M}^{\mathrm{Dec}}$ and returns a decoded string $\hat{\boldsymbol{x}}$:

$$\mathrm{Enc}(\mathrm{M}^{\mathrm{Enc}}; \boldsymbol{x}) = \boldsymbol{b} \quad \text{and} \quad \mathrm{Dec}(\mathrm{M}^{\mathrm{Dec}}; \boldsymbol{b}) = \hat{\boldsymbol{x}}\,. \tag{3}$$

Given a constraint on the difference between $\mathrm{M}^{\mathrm{Enc}}$ and $\mathrm{M}^{\mathrm{Dec}}$'s outputs on any given context, we say that the algorithm is *mismatch-tolerant* with respect to that constraint if, for all $\mathrm{M}^{\mathrm{Enc}}, \mathrm{M}^{\mathrm{Dec}}$ that satisfy the constraint,

$$\mathrm{Dec}(\mathrm{M}^{\mathrm{Dec}}; \mathrm{Enc}(\mathrm{M}^{\mathrm{Enc}}; \boldsymbol{x})) = \boldsymbol{x} \quad \text{for all } \boldsymbol{x}\,. \tag{4}$$

The goal is to design algorithms which can tolerate a given amount of mismatch between the encoder and decoder probability distributions while minimizing the cost in compression efficiency.

2.1 The Bounded Prediction Mismatch Setting

Since the encoder and decoder are using the same LLM on the same inputs, it is reasonable to assume that they obtain logits whose difference is bounded by some reasonably small $\varepsilon > 0$. A natural choice for this is to assume that their difference has bounded L_∞ norm (i.e. elementwise): $\|\boldsymbol{u} - \boldsymbol{v}\|_\infty := \max_{k\in\mathcal{A}} |u_k - v_k| \le \varepsilon$. We first define the following a measure of difference between two probability distributions:

Definition 1. *The* conditional total variation distance *(d_{CTV}) between two probability distributions $\boldsymbol{p}, \boldsymbol{q}$ on an alphabet $\mathcal{A}$ is defined as the maximum total variation distance (d_{TV}) of $\boldsymbol{p}$ and $\boldsymbol{q}$ after conditioning on some (nonempty) $S \subseteq \mathcal{A}$, i.e.*

$$d_{\mathrm{CTV}}(\boldsymbol{p}, \boldsymbol{q}) := \max_{\emptyset\neq S\subseteq\mathcal{A}} d_{\mathrm{TV}}(\boldsymbol{p}(\cdot|S), \boldsymbol{q}(\cdot|S)) \tag{5}$$

where $\boldsymbol{p}(\cdot|S)$ and $\boldsymbol{q}(\cdot|S)$ are, respectively, $\boldsymbol{p}$ and $\boldsymbol{q}$ conditioned on the outcome being in S.

Note that there is no divide-by-zero issue with conditioning on any (nonempty) S since all probabilities are induced via the softmax function and hence strictly positive. Bounded prediction mismatch then bounds conditional TV distance:

Proposition 1. *If $\boldsymbol{u}, \boldsymbol{v}$ induce probability distributions $\boldsymbol{p} = \mathrm{softmax}(\boldsymbol{u})$ and $\boldsymbol{q} = \mathrm{softmax}(\boldsymbol{v})$ over $\mathcal{A}$, and $\|\boldsymbol{u} - \boldsymbol{v}\|_\infty \le \varepsilon$, then $d_{\mathrm{CTV}}(\boldsymbol{p}, \boldsymbol{q}) \le \frac{\varepsilon}{2}$.*

Proof. Using the definition of TV distance, the conditional TV distance can be rewritten as

$$d_{\mathrm{CTV}}(\boldsymbol{p}, \boldsymbol{q}) = \max_{\substack{\emptyset\neq S\subseteq\mathcal{A}\\ S^*\subseteq S}} |p(S^*|S) - q(S^*|S)| \tag{6}$$

$$\implies \max_{\|\boldsymbol{u}-\boldsymbol{v}\|_\infty\le\varepsilon} d_{\mathrm{CTV}}(\boldsymbol{p}, \boldsymbol{q}) = \max_{\|\boldsymbol{u}-\boldsymbol{v}\|_\infty\le\varepsilon} \max_{\substack{\emptyset\neq S\subseteq\mathcal{A}\\ S^*\subseteq S}} |p(S^*|S) - q(S^*|S)| \tag{7}$$

$$= \max_{\substack{\emptyset\neq S\subseteq\mathcal{A}\\ S^*\subseteq S}} \max_{\|\boldsymbol{u}-\boldsymbol{v}\|_\infty\le\varepsilon} |p(S^*|S) - q(S^*|S)| \tag{8}$$

So, if $\max_{\|\boldsymbol{u}-\boldsymbol{v}\|_\infty \le \varepsilon} |p(S^*|S) - q(S^*|S)| \le \frac{\varepsilon}{2}$ for all $S^* \subseteq S \subseteq \mathcal{A}$, then $\max_{\|\boldsymbol{u}-\boldsymbol{v}\|_\infty \le \varepsilon} d_{\mathrm{CTV}}(\boldsymbol{p}, \boldsymbol{q}) \le \frac{\varepsilon}{2}$.

In other words, the conditional TV distance between $\boldsymbol{p}, \boldsymbol{q}$ induced by $\|\boldsymbol{u}-\boldsymbol{v}\|_\infty \le \varepsilon$ can be bounded by first fixing $S^* \subseteq S \subseteq \mathcal{A}$ and then bounding $|p(S^*|S) - q(S^*|S)|$ over all $\boldsymbol{p}, \boldsymbol{q}$ whose logits are within ε of each other in L_∞ distance. We assume WLOG that the $\boldsymbol{u}, \boldsymbol{v}$ maximizing $|p(S^*|S) - q(S^*|S)|$ has $q(S^*|S) > p(S^*|S)$ (otherwise S^* can be changed into $S \backslash S^*$), so the goal is to maximize $q(S^*|S) - p(S^*|S)$ given the logit L_∞ bound. This is achieved by letting

$$v_k = u_k + \varepsilon \text{ for } k \in S^* \quad \text{and} \quad u_k - \varepsilon \text{ for } k \notin S^* \tag{9}$$

Let $p^* := p(S^*|S)$ and $q^* := q(S^*|S)$, which are both scalars in $[0, 1]$. Then, given (9),

$$q^* = \frac{\sum_{k \in S^*} e^{u_k + \varepsilon}}{\sum_{k \in S^*} e^{u_k + \varepsilon} + \sum_{k \in S \backslash S^*} e^{u_k - \varepsilon}} = \frac{e^\varepsilon p^*}{e^\varepsilon p^* + e^{-\varepsilon}(1 - p^*)} \tag{10}$$

$$\implies q^* - p^* \le \max_{p^* \in [0,1]} \left(\frac{e^\varepsilon p^*}{e^\varepsilon p^* + e^{-\varepsilon}(1 - p^*)} - p^* \right) = \tanh\left(\frac{\varepsilon}{2}\right) \le \frac{\varepsilon}{2}\,. \tag{11}$$

Tracing this bound back to the conditional TV distance concludes the proof. □

3 THE PMATIC ALGORITHM

The Probability Matching Interval Coding (PMATIC) algorithm addresses prediction mismatch by ensuring that the encoder and decoder use a common probability distribution for each token. The first step of PMATIC is to convert the input token string into a bitstring using a dictionary that associates each token with a length-$\ell := \lceil \log_2(|\,\mathcal{A}\,|) \rceil$ bitstring. We will call each bit in this bitstring a *token bit*. PMATIC encodes these tokens bits with arithmetic coding using next-bit conditional probabilities derived from the token's encoder prediction vector. At each step, the next-bit prediction is a scalar in $[0, 1]$ giving the probability that the token bit equals 1. The key idea is to divide the interval $[0, 1]$ into a set of *bins* (disjoint equal-length intervals which cover $[0, 1]$). Then, instead of using their exact predictions, the encoder and decoder use either the center of the bin their predictions fall into or the nearest boundary between two bins; which one to use is decided by the encoder and communicated to the decoder by use of auxiliary 'helper' bits, which are also sent via arithmetic coding. This procedure can be viewed as quantizing the probability of each token bit.

For any token x_i in the message, we denote its corresponding bitstring by $\boldsymbol{b}_i := b_i(1) \dots b_i(\ell)$ and define the following parameters and notation:

- $\delta > 0$, which represents the amount of prediction mismatch (per bit) which the algorithm can tolerate, as measured by conditional TV distance.
- $r > 0$ is the radius of the quantization bins (so the width of a bin is $2r$); r will be chosen to maximize performance given δ. We will assume that $r = 1/(2m)$ for some integer m; in practice this entails rounding r up or down slightly to the nearest such value, which will still be at the correct scale relative to δ. We also always set $r > 2\delta$.
- Let $h(p) := p \log\left(\frac{1}{p}\right) + (1 - p) \log\left(\frac{1}{1-p}\right)$ be the binary entropy function (the entropy of a Bernoulli random variable with probability p), and $H(\boldsymbol{p})$ be the more general entropy function for a probability vector $\boldsymbol{p}$ over a finite set.
- Let $D_{\mathrm{KL}}(p\|q) := p \log\left(\frac{p}{q}\right) + (1 - p) \log\left(\frac{1-p}{1-q}\right)$ be the binary Kullback Liebler divergence (the divergence between two Bernoulli random variables with proabilities p, q respectively).
- Let $S_{\boldsymbol{b}_i^{j-1}} := \{\boldsymbol{a} \in \{0, 1\}^\ell : \boldsymbol{a}^{j-1} = \boldsymbol{b}_i^{j-1}\}$ denote the set of length-ℓ bitstrings whose first $j - 1$ bits agree with $\boldsymbol{b}_i$.

3.1 THE PMATIC ENCODER

Consider encoding the jth bit, $b_i(j)$, of token x_i. Let the encoder and decoder prediction vectors for token i be, respectively, $\boldsymbol{p}(i) := \mathrm{softmax}(\mathrm{M}^{\mathrm{Enc}}(\boldsymbol{x}^{i-1}))$ and $\boldsymbol{q}(i) := \mathrm{softmax}(\mathrm{M}^{\mathrm{Dec}}(\boldsymbol{x}^{i-1}))$. The predictions for the jth bit $b_i(j)$ for the encoder and decoder, conditional on the prior bits in $\boldsymbol{b}_i$, are

$$p_i(j) := \mathbb{P}_{\boldsymbol{p}(i)}\left[b_i(j) = 1 \,|\, S_{\boldsymbol{b}_i^{j-1}}\right] \quad \text{and} \quad q_i(j) := \mathbb{P}_{\boldsymbol{q}(i)}\left[b_i(j) = 1 \,|\, S_{\boldsymbol{b}_i^{j-1}}\right] \tag{12}$$

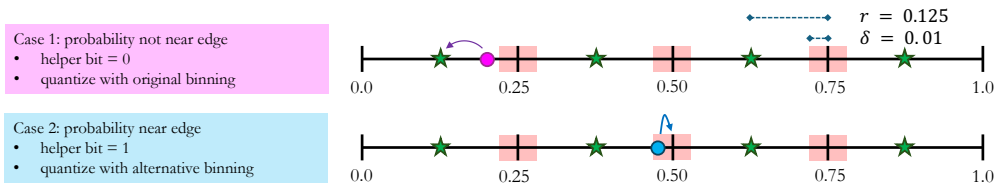

Figure 1: Examples of PMATIC helper-bit and quantization logic for two cases, one where the helper bit is 0 and one where the helper bit is 1.

These can be computed directly using $\boldsymbol{p}(i)$ (or $\boldsymbol{q}(i)$) by setting all values outside of $S_{\boldsymbol{b}_i^{j-1}}$ to 0 and renormalizing to get the conditional probability distribution.

The interval $[0,1]$ is then split into radius-r intervals, which we call *bins*, $I_1, I_2, \ldots I_m$, where $m = 1/(2r)$ (which, as assumed above, is an integer) and $I_k = [2r(k-1), 2rk]$. The center of bin I_k is therefore $c_k := 2r(k-1) + r$, and we denote the δ-*interior* of I_k (the set of points in I_k at least δ away from any point outside I_k) by

$$I_k^\delta = \begin{cases} [0, 2r - \delta] & \text{if } k = 1 \\ [2r(m-1) + \delta, 1] & \text{if } k = m \\ [2r(k-1) + \delta, 2rk - \delta] & \text{if } k \neq 1, m \end{cases} \tag{13}$$

I_1^δ, I_m^δ need special definition as they have an edge next to the edge of $[0,1]$ instead of another bin.

Note that if $p_i(j) \in I_k^\delta$ and $|p_i(j) - q_i(j)| \leq \delta$, then $q_i(j) \in I_k$, and that if $p_i(j) \notin I_k^\delta$ for all k, then there is instead a unique integer $k \neq 1, m$ for which $|2rk - p_i(j)| < \delta$.

In addition to the token bit $b_i(j)$, PMATIC encodes (prior to the token bit) a *helper bit*

$$b_i'(j) = \begin{cases} 0 & \text{if } b_i(j) \in I_k^\delta \text{ for some } k \\ 1 & \text{otherwise} \end{cases} \tag{14}$$

This encoding is done with arithmetic coding using probabilities $p' := \delta/r$ for the helper bit and

$$\hat{p}_i(j) = \begin{cases} c_k = 2r(k-1) + r & \text{if } p_i(j) \in I_k^\delta \\ 2rk \text{ for the integer } k \text{ s.t. } |2rk - p_i(j)| < \delta & \text{otherwise} \end{cases} \tag{15}$$

for the token bit. The probability $\hat{p}_i(j)$ is the common probability of token bit j that both the encoder and decoder agree to use. See Figure 1 for an example.

The intuition for the helper bits is that when $p_i(j) \in I_k^\delta$, the encoder knows that the decoder's probability $q_i(j) \in I_k$ (the same bin), so the encoder quantizes to the bin center and tells the decoder to do the same by sending the helper bit $b_i'(j) = 0$. If $p_i(j)$ is not in the δ-interior of its bin, the encoder no longer knows that the decoder probability lies in the same bin. However, in this case both probabilities must be near the same boundary point between two bins, so the encoder quantizes to the nearest boundary point and tells the decoder to do the same by sending the helper bit $b_i'(j) = 1$. In either case, they agree on the probability to use for encoding and decoding. The probability of being in the δ-interior of a bin is $\approx \delta/r$, which is very small if $r \gg \delta$. This gives the helper bits low entropy and hence makes them highly compressible via arithmetic coding.

To summarize, given a token x_i and context $\boldsymbol{x}^{i-1}$, the PMATIC encoder does the following:

1. Computes $\boldsymbol{p}(i) = \mathrm{softmax}(\mathrm{M}^{\mathrm{Enc}}(\boldsymbol{x}^{i-1}))$, gets the bitstring $\boldsymbol{b}_i$ corresponding to x_i, and computes the conditional next-bit probabilities $p_i(1), \ldots, p_i(\ell) \in [0,1]$.
2. Computes for each bit $b_i(j)$ the helper bit $b_i'(j)$ and quantized probability $\hat{p}_i(j)$ ((14), (15)).
3. Encodes the bitstring $b_i'(1)b_i(1)\ldots b_i'(\ell)b_i(\ell)$ using arithmetic coding, where the encoding probability for each helper bit $b_i'(j)$ is $p' = \delta/r$ and the encoding probability for each token bit $b_i(j)$ is $\hat{p}_i(j)$.

3.2 The PMATIC Decoder

The PMATIC decoder takes the encoded message $\boldsymbol{y}$ and decodes it sequentially in pairs of bits. Each pair consists of a helper bit and a token bit; the helper bit is decoded first using $p' = \delta/r$ as the probability (since helper bits are always encoded using this probability), and determines the quantized probability to use. Analogous to the encoder next-bit prediction, let the decoder next-bit prediction for bit j of token i be denoted $q_i(j)$. Then the decoder decodes the token bit using the quantized probability:

$$\hat{q}_i(j) = \begin{cases} c_k \text{ for } k \text{ s.t. } q_i(j) \in I_k & \text{if } b'_i(j) = 0 \\ 2rk \text{ for } k \in \{1, \ldots, m-1\} \text{ s.t. } |2rk - q_i(j)| \text{ is minimized} & \text{if } b'_i(j) = 1 \end{cases} \tag{16}$$

After decoding all the bits, the helper bits are discarded and the token bits are converted back into a token string using the token-bitstring dictionary. PMATIC is successful when $\hat{q}_i(j)$ is the same as $\hat{p}_i(j)$ produced in the encoder step. This is discussed in greater detail in the next section.

4 Analysis

We wish to: (i) show that PMATIC ensures correct decoding if the conditional TV distance between the encoder and decoder token predictions is at most δ; (ii) show (in expectation) theoretical performance bounds. Since we compress in bits, all logarithms are base-2 unless noted otherwise.

4.1 Correctness

Theorem 1. *If $d_{\mathrm{CTV}}(\boldsymbol{p}(i), \boldsymbol{q}(i)) \leq \delta$, then $\hat{q}_i(j) = \hat{p}_i(j)$ for all j (i.e. the encoder and decoder will agree on the quantized probabilities for all bits corresponding to token x_i).*

Proof. Consider bit j of token i; without loss of generality we can assume that all previous bits in i and all previous tokens were decoded correctly (since, if any bits are incorrectly decoded, there must be a first one). Since $d_{\mathrm{CTV}}(\boldsymbol{p}(i), \boldsymbol{q}(i)) \leq \delta$, we know that $|p_i(j) - q_i(j)| \leq \delta$ (since $p_i(j), q_i(j)$ are derived by conditioning $\boldsymbol{p}(i), \boldsymbol{q}(i)$ on the outcome coming from the set $S_{\boldsymbol{b}_i^{j-1}}$).

First, the decoder will decode the helper bit $b'_i(j)$ using the probability δ/r. Since δ/r is fixed and used for all helper bits, the encoder and decoder probabilities match and the helper bit is decoded correctly. Now we consider two cases: $b'_i(j) = 0$, and $b'_i(j) = 1$.

If $b'_i(j) = 0$, this means that computed encoder next-bit predictor $p_i(j)$ falls in the δ-interior I_k^δ of some bin; since $|p_i(j) - q_i(j)| \leq \delta$, this means $q_i(j) \in I_k$ (not necessarily the δ-interior, just the bin itself). Thus, since both the encoder and decoder quantize to the center c_k of the bin, we have $\hat{p}_i(j) = \hat{q}_i(j) = c_k$ and the token bit is encoded correctly.

If $b'_i(j) = 1$, then there is some $k \in \{1, \ldots, m-1\}$ such that $|p_i(j) - 2rk| \leq \delta$ (note that $2rk$ here is the boundary between two bins). Then, since we set $r > 2\delta$ and bins have width $2r$:

$$|p_i(j) - 2rk| \leq \delta \implies |q_i(j) - 2rk| \leq 2\delta \tag{17}$$

$$\implies |q_i(j) - 2rk'| \geq 2r - 2\delta > 2\delta \text{ for any integer } k' \neq k \tag{18}$$

$$\implies \hat{q}_i(j) = 2rk = \hat{p}_i(j). \tag{19}$$

Thus, in either case, the encoder and decoder agree on the next-bit probability for $b_i(j)$ and the decoder will decode the bit and update the arithmetic code interval correctly. □

Note that, by Theorem 1 and Proposition 1, if the LLM has logits that differ by at most ε between the encoder and decoder, then compressing with PMATIC using $\delta = \varepsilon/2$ guarantees correctness.

4.2 Compression Loss

For the compression performance analysis, we make the following simplifying assumptions:

- The encoder's next-token probabilities are the true probabilities. This means that the expected increase in message length for each bit $b_i(j)$ is $D_{\mathrm{KL}}(p_i(j)\|\hat{p}_i(j))$ (Cover & Thomas, 2006, Thm 5.4.3), and that the no-mismatch optimal expected message length per token x_i is $H(\boldsymbol{p}_i)$ where $H(\cdot)$ is the entropy.
- Within each individual bin, the encoder next-bit probability is roughly uniformly distributed (so e.g. it's not disproportionately likely to fall next to the bin boundary and the probability of being within δ of a bin boundary is $\approx \delta/r$).[3]

We consider the *compression loss* of PMATIC over traditional (non-mismatch-tolerant) arithmetic coding, i.e. the extra message length incurred by PMATIC in order to tolerate a conditional TV distance bound of δ with a bin width of r. This loss comes from two sources:

1. *Helper bit encoding:* the helper bits require extra message bits to send. Since all helper bits are (approximately) Bernoulli with parameter δ/r, the expected extra encoding length per helper bit is the binary entropy $h(\delta/r) = \frac{\delta}{r}\log\left(\frac{r}{\delta}\right) + \left(\frac{r-\delta}{r}\right)\log\left(\frac{r}{r-\delta}\right)$.

   If $r \gg \delta$, then the first term dominates and the entropy of each helper bit is $\approx \frac{\delta}{r}\log\left(\frac{r}{\delta}\right)$.
2. *Quantization loss:* the quantized probability $\hat{p}_i(j)$ is different than the true probability $p_i(j)$, incurring a quantization loss of $D_{\mathrm{KL}}(p_i(j)\|\hat{p}_i(j))$.
   Since $r \le \hat{p}_i(j) \le 1-r$ (all bin centers and boundaries are in $[r, 1-r]$) and $|p_i(j)-\hat{p}_i(j)| \le r$, the quantization loss satisfies $D_{\mathrm{KL}}(p_i(j)\|\hat{p}_i(j)) \le 2\log(e)r$, since KL divergence is bounded above by χ^2 divergence (Polyanskiy & Wu, 2025, Ch. 7):

$$D_{\mathrm{KL}}(p_i(j)\|\hat{p}_i(j)) \le \log(e)\frac{(p_i(j)-\hat{p}_i(j))^2}{\hat{p}_i(j)(1-\hat{p}_i(j))} \le 2\log(e)r \tag{20}$$

   since the numerator is $\le r^2$ and the denominator is $\ge (1/2)r$.

Note that these losses respond in opposite directions when the bin radius r is increased: larger bins mean lower helper bit entropy but a bigger difference between the true probability and the quantized probability. This means that setting r to balance the loss terms gives an approximate minimizer of the objective function: any other $r' \neq r$ will make one of the loss terms larger, so balancing the loss terms incurs a total loss of at most 2 times the optimal. This is achieved (approximately) with

$$2\log(e)r = \frac{\delta}{r}\log\left(\frac{r}{\delta}\right) \implies r \approx \frac{\sqrt{\delta\log\left(\frac{1}{\delta}\right)}}{\sqrt{2\log e}} \tag{21}$$

and yields a total loss on the order of $O\left(\sqrt{\delta\log\left(\frac{1}{\delta}\right)}\right)$ (note that one of the loss terms is itself $O(r)$, so the total loss should be proportional to r when balancing the loss terms).

5 Experiments

We test PMATIC on text using different (quantized) LLMs as the predictive model and synthetic mismatched probabilities, and compare to standard baselines and non-mismatch-robust LLM-driven compression.

5.1 Setup

The three models we use are LLaMA 3.1 8B (4-bit quantized), Mistral 7B v0.1 (3-bit quantized) and Qwen 2.5 Instruct 7B (3-bit quantized). Llama 3.1 8B [Grattafiori et al. (2024)] includes a tokenizer with a $128,256$-token vocabulary, whereas the Mistral and Qwen models we used respectively have a $32,000$-token vocabulary and a $151,643$-token vocabulary.

[3]This assumption is reasonable and holds up well in practice if the bins are relatively small. If desired, PMATIC can modified so that this assumption is not necessary for the analysis, by adding a PRNG (pseudo-random number generator) to the encoder and decoder to (pseudo)randomly translate the bins by a value within $[-r, r]$. This ensures that for any $p_i(j)$, it has at most δ/r chance of being within δ of a bin boundary.

Parameter Settings	Fraction of Helper Bits Equal to 1	Helper Bit Fraction of Encoded Text
no PMATIC	0	0
$\delta = 10^{-5},\ r = 0.005$	0.00051	0.04594
$\delta = 10^{-3},\ r = 0.05$	0.00368	0.18947
$\delta = 10^{-2},\ r = 0.125$	0.01145	0.34073

Figure 2: Helper bit behavior averaged across different PMATIC robustness parameter settings.

We run our experiment on several datasets. One of these datasets is the first (nearly) 10 MB of the enwik8 benchmark Hutter (2006), consisting of a collection of Wikipedia articles from 2006 with non-ASCII characters removed. The second dataset consists of 1000 randomly selected articles from Wikipedia (pulled in September 2025) with non-ASCII characters removed. The other datasets include two additional English texts, *Hamlet* by Shakespeare and *Emma* by Austen. We also use *Candide* by Voltaire in French, and *Dream of the Red Chamber* by Cao Xuewin in Chinese. For each dataset, except randomly selected Wikipedia articles, we split all our files into smaller files of size ≈ 5 KB and compress each file separately.

We run our algorithm with three choices of r, δ. One setting uses $\delta = 0.001$ and $r = 0.05$. Given $\delta = 0.001$, the approximate minimizer r given in (21) would be $r \approx 0.047$; we choose $r = 0.05$ since it is close and divides $[0, 1]$ into intervals evenly. The small mismatch setting uses $\delta = 0.00001$ and $r = 0.005$, with r chosen again to be on the right scale and to divide the $[0, 1]$ evenly. The large mismatch setting uses $\delta = 0.01$ and $r = 0.125$, and is chosen similarly. [4]

To speed up the algorithm, we use a rolling context window of maximum size 512 which resets every 256 tokens via truncation: each time the context length reaches 512, we drop the oldest 256 tokens. Our program ran on a high-performance computing system with Xeon CPU nodes and Volta GPUs.

The first step of our processing is to tokenize each file. Since PMATIC assigns each token in the alphabet a unique fixed-length bitstring, we assign a random ℓ-bit representation (where ℓ depends on the model used[5]) for each token and convert the file to a bitstring. The same bitstring dictionary is used across all files in each setting.

The maximum logit mismatch is selected in each setting to be $\varepsilon = 2\delta$, to ensure the conditional TV distance is always at most δ (as per Proposition 1). The encoding of the text is done with the probabilities given by the model. We then create a synthetic mismatched probability distribution for the decoder by adding IID uniform noise from $[-\varepsilon, \varepsilon]$ to each logit of the encoder's distribution.

Encoding and decoding the helper and token bits generated by PMATIC is done with arithmetic coding, using the implementation from Buzanis (2024).

For comparison, we include several standard compression algorithms. Among these, cmix is a compressor known for achieving the best compression ratios before LLM-driven compression. For other standard compression algorithms, we chose the compression level that optimizes for smallest file size.

5.2 Results

PMATIC successfully decoded all files, validating our theoretical correctness result, and its performance validates the theoretical performance analysis. Our results show that PMATIC provides robustness to numerical deviations without significant loss over the extremely good rates of LLM-based compression on text. PMATIC increases the number of bits per token by ≈ 0.2 to tolerate a 0.00002 mismatch per logit, by ≈ 2 to tolerate a 0.002 mismatch per logit and by ≈ 5 to tolerate a 0.02 mismatch per logit. Even in the setting combating the most mismatch, compression with PMATIC still achieves significantly better compression rates than the baselines used.

[4] Since we chose r only through a rough approximation of the optimization process, it may be possible to improve performance by further refining the value of r.

[5] For LLaMA 3.1, each token requires a length-17 bitstring representation since $\lceil \log(128, 256) \rceil = 17$.

		Enwik8	Randomly selected Wikipedia articles	Hamlet	Emma	Candide *French*	Dream of the Red Chamber *Chinese*
LLM-based Compression (with and without PMATIC)							
Meta LLaMA 3.1	no PMATIC	0.0780	0.0700	0.0878	0.0606	0.1024	–
	$\delta = 10^{-5}, r = 0.005$	0.0847	0.0878	0.0952	0.0660	0.1102	–
	$\delta = 10^{-3}, r = 0.05$	0.1353	0.1330	0.1514	0.1099	0.1683	–
	$\delta = 10^{-2}, r = 0.125$	0.2492	0.2085	0.2772	0.2113	0.2971	–
Mistral 7B	no PMATIC	0.0867	0.0737	0.1501	0.1066	–	–
	$\delta = 10^{-5}, r = 0.005$	0.0940	0.0794	0.1587	0.1134	–	–
	$\delta = 10^{-3}, r = 0.05$	0.1481	0.1198	0.2167	0.1595	–	–
	$\delta = 10^{-2}, r = 0.125$	0.2699	0.2106	0.3447	0.2610	–	–
Qwen2.5 7B Instruct	no PMATIC	0.0881	0.0824	0.1102	0.1183	0.1150	0.1268
	$\delta = 10^{-5}, r = 0.005$	0.0951	0.0880	0.1177	0.1253	0.1231	0.1345
	$\delta = 10^{-3}, r = 0.05$	0.1501	0.1315	0.1755	0.1738	0.1831	0.1879
	$\delta = 10^{-2}, r = 0.125$	0.2751	0.2297	0.3067	0.2825	0.3171	0.3073
Standard Baselines							
cmix		0.3558	0.3644	0.3865	0.3797	0.3709	0.3824
brotli	level 11	0.3524	0.3546	0.4361	0.3918	0.4442	0.4733
bzip2	level 9	0.4537	0.4605	0.4636	0.4539	0.4494	0.4727
xz	level 9	0.4647	0.4904	0.5111	0.4973	0.4866	0.5192
gzip	level 9	0.4601	0.4759	0.5007	0.4852	0.4768	0.5305
zstd	level 22	0.4676	0.4773	0.5034	0.4882	0.4851	0.5544

Figure 3: Compression ratio across different models, PMATIC parameters, and datasets; compression ratios are given by $\frac{\text{compressed file size}}{\text{uncompressed file size}}$. PMATIC overhead (cost of gaining mismatch robustness through PMATIC) for each parameter setting can be computed by subtracting the corresponding 'no PMATIC' compression ratio from the PMATIC compression ratio.

6 Future Work

While PMATIC has good theoretical and experimental performance in enabling model-driven compression to tolerate bounded prediction mismatch, there remain significant challenges. While PMATIC was validated using synthetic prediction mismatch, it remains to be validated on datasets that include real-world LLM prediction mismatch. A related problem is to extend PMATIC (or design another mismatch-robust coding technique) to allow for stochastically-bounded mismatch rather than mismatch with a hard upper bound for all logits. For practical usage, the tradeoff between prediction model size, compression efficiency, and computational performance should be characterized so that a model exhibiting a good balance of these characteristics can be chosen.

In addition, the fundamental limits of model-driven compression under prediction mismatch (that is, how much additional message length is mathematically required to correct a given amount of potential mismatch) remain unknown, and theoretical research into the mathematical and information-theoretic properties of the problem will be needed to address these fundamental limits.

Finally, LLM (and large neural network) non-determinism remains a significant challenge in several contexts outside of model-driven compression, such as ensuring reproducibility of experimental results in machine learning. It may be interesting to explore whether PMATIC, or similar approaches, might offer certain tools to address LLM non-determinism in these contexts.

REFERENCES

Berk Atil, Sarp Aykent, Alexa Chittams, Lisheng Fu, Rebecca J. Passonneau, Evan Radcliffe, Guru Rajan Rajagopal, Adam Sloan, Tomasz Tudrej, Ferhan Ture, Zhe Wu, Lixinyu Xu, and Breck Baldwin. Non-determinism of "deterministic" llm settings, 2025. URL https://arxiv.org/abs/2408.04667.

Fabrice Bellard. Lossless data compression with neural networks. *URL: https://bellard. org/nncp/nncp. pdf*, 2019.

Fabrice Bellard. Nncp v2: Lossless data compression with transformer. *Preprint at Fabrice Bellard https://bellard. org/nncp/nncp_v2. pdf*, 2021.

Alexander Buzanis. llama-zip: An llm-powered lossless compression tool. https://github.com/AlexBuz/llama-zip, 2024. Commit on main branch; accessed 2025-05-19.

Boyuan Chen, Mingzhi Wen, Yong Shi, Dayi Lin, Gopi Krishnan Rajbahadur, and Zhen Ming (Jack) Jiang. Towards training reproducible deep learning models. In *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22, pp. 2202–2214. ACM, May 2022. doi: 10.1145/3510003.3510163. URL http://dx.doi.org/10.1145/3510003.3510163.

Kecheng Chen, Pingping Zhang, Hui Liu, Jie Liu, Yibing Liu, Jiaxin Huang, Shiqi Wang, Hong Yan, and Haoliang Li. Large language models for lossless image compression: Next-pixel prediction in language space is all you need. *arXiv preprint arXiv:2411.12448*, 2024.

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.

Kevin Coakley, Christine R. Kirkpatrick, and Odd Erik Gundersen. Examining the effect of implementation factors on deep learning reproducibility. In *2022 IEEE 18th International Conference on e-Science (e-Science)*, pp. 397–398. IEEE, October 2022. doi: 10.1109/escience55777.2022.00056. URL http://dx.doi.org/10.1109/eScience55777.2022.00056.

A. Feder Cooper, Jonathan Frankle, and Christopher De Sa. Non-determinism and the lawlessness of machine learning code. In *Proceedings of the 2022 Symposium on Computer Science and Law*, CSLAW '22, pp. 1–8. ACM, November 2022. doi: 10.1145/3511265.3550446. URL http://dx.doi.org/10.1145/3511265.3550446.

T.M. Cover and J.A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006. ISBN 0471241954.

David Cox. Syntactically informed text compression with recurrent neural networks. *arXiv preprint arXiv:1608.02893*, 2016.

Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, Marcus Hutter, and Joel Veness. Language modeling is compression, 2024. URL https://arxiv.org/abs/2309.10668.

Bahadir Eryilmaz, Osman Alperen Koras, Jorg Schlotterer, and Christin Seifert. Investigating the Impact of Randomness on Reproducibility in Computer Vision: A Study on Applications in Civil Engineering and Medicine . In *2024 IEEE 6th International Conference on Cognitive Machine Intelligence (CogMI)*, pp. 265–274, Los Alamitos, CA, USA, October 2024. IEEE Computer Society. doi: 10.1109/CogMI62246.2024.00042. URL https://doi.ieeecomputersociety.org/10.1109/CogMI62246.2024.00042.

Mohit Goyal, Kedar Tatwawadi, Shubham Chandak, and Idoia Ochoa. Deepzip: Lossless data compression using recurrent neural networks. *CoRR*, abs/1811.08162, 2018. URL http://arxiv.org/abs/1811.08162.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

M. Guazzo. A general minimum-redundancy source-coding algorithm. *IEEE Transactions on Information Theory*, 26(1):15–25, 1980. doi: 10.1109/TIT.1980.1056143.

David A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952. doi: 10.1109/JRPROC.1952.273898.

Marcus Hutter. The hutter prize for lossless compression of human knowledge. http://prize.hutter1.net/, 2006. Accessed: 2025-09-11.

Yuriy Kim and Evgeny Belyaev. An efficient compression of deep neural network checkpoints based on prediction and context modeling, 2025. URL https://arxiv.org/abs/2506.12000.

Byron Knoll. cmix: A data compression program. https://www.byronknoll.com/cmix.html, 2025. Accessed: 2025-09-23.

Qian Liu, Yiling Xu, and Zhu Li. Decmac: A deep context model for high efficiency arithmetic coding. In *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pp. 438–443. IEEE, 2019.

Zhoujun Ma, Hong Zhu, Zhuohao He, Yue Lu, and Fuyuan Song. Deep lossless compression algorithm based on arithmetic coding for power data. *Sensors*, 22(14), 2022. ISSN 1424-8220. doi: 10.3390/s22145331. URL https://www.mdpi.com/1424-8220/22/14/5331.

Yu Mao, Yufei Cui, Tei-Wei Kuo, and Chun Jason Xue. Trace: A fast transformer-based general-purpose lossless compressor. In *Proceedings of the ACM Web Conference 2022*, pp. 1829–1838, 2022.

Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Practical full resolution learned lossless image compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10629–10638, 2019.

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.

Miguel Morin and Matthew Willetts. Non-determinism in tensorflow resnets, 2020. URL https://arxiv.org/abs/2001.11396.

NVIDIA Corporation. *NVIDIA cuBLAS Library Documentation*, 2025a. URL https://docs.nvidia.com/cuda/cublas/. Version: CUDA cuBLAS.

NVIDIA Corporation. *NVIDIA cuDNN Backend API: Odds and Ends (Determinism and Reproducibility)*, 2025b. URL https://docs.nvidia.com/deeplearning/cudnn/backend/latest/developer/misc.html. cuDNN Developer Documentation.

Richard Clark Pasco. *Source coding algorithms for fast data compression*. PhD thesis, Stanford University CA, 1976.

Yury Polyanskiy and Yihong Wu. *Information Theory: From Coding to Learning*. Cambridge University Press, 2025.

Hochang Rhee, Yeong Il Jang, Seyun Kim, and Nam Ik Cho. Lc-fdnet: Learned lossless image compression with frequency decomposition network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6033–6042, 2022.

Jorma J Rissanen. Generalized kraft inequality and arithmetic coding. *IBM Journal of research and development*, 20(3):198–203, 1976.

Ionut Schiopu, Yu Liu, and Adrian Munteanu. Cnn-based prediction for lossless coding of photographic images. In *2018 Picture Coding Symposium (PCS)*, pp. 16–20. IEEE, 2018.

Alex Schlögl, Nora Hofer, and Rainer Böhme. Causes and effects of unanticipated numerical deviations in neural network inference frameworks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 56095–56107. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/af076c3bdbf935b81d808e37c5ede463-Paper-Conference.pdf.

J. Schmidhuber and S. Heil. Sequential neural text compression. *IEEE Transactions on Neural Networks*, 7(1):142–146, 1996. doi: 10.1109/72.478398.

Harald Semmelrock, Tony Ross-Hellauer, Simone Kopeinik, Dieter Theiler, Armin Haberl, Stefan Thalmann, and Dominik Kowald. Reproducibility in machine learning-based research: Overview, barriers and drivers, 2025. URL https://arxiv.org/abs/2406.14325.

Sanjif Shanmugavelu, Mathieu Taillefumier, Christopher Culver, Oscar Hernandez, Mark Coletti, and Ada Sedova. Impacts of floating-point non-associativity on reproducibility for hpc and deep learning applications. In *Proceedings of the SC '24 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*, SC-W '24, pp. 170–179. IEEE Press, 2025. ISBN 9798350355543. doi: 10.1109/SCW63240.2024.00028. URL https://doi.org/10.1109/SCW63240.2024.00028.

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.

George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. Full resolution image compression with recurrent neural networks. *CoRR*, abs/1608.05148, 2016. URL http://arxiv.org/abs/1608.05148.

Chandra Shekhara Kaushik Valmeekam, Krishna Narayanan, Dileep Kalathil, Jean-Francois Chamberland, and Srinivas Shakkottai. Llmzip: Lossless text compression using large language models, 2023. URL https://arxiv.org/abs/2306.04050.

Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Commun. ACM*, 30:520–540, 1987. URL https://api.semanticscholar.org/CorpusID:3343393.

A Appendix

A.1 PMATIC Full Token Example

Illustrated here is an example of PMATIC encoding when there are 8 total tokens (each token can be described by a 3-bit bitstring). PMATIC has parameter settings of $\delta = 0.01$ and $r = 0.125$ (so there are four quantization bins in total).

Token	Bitstring	Prob
A	010	0.097
B	000	0.249
C	101	0.206
D	111	0.047
E	010	0.045
F	100	0.200
G	011	0.127
H	110	0.029

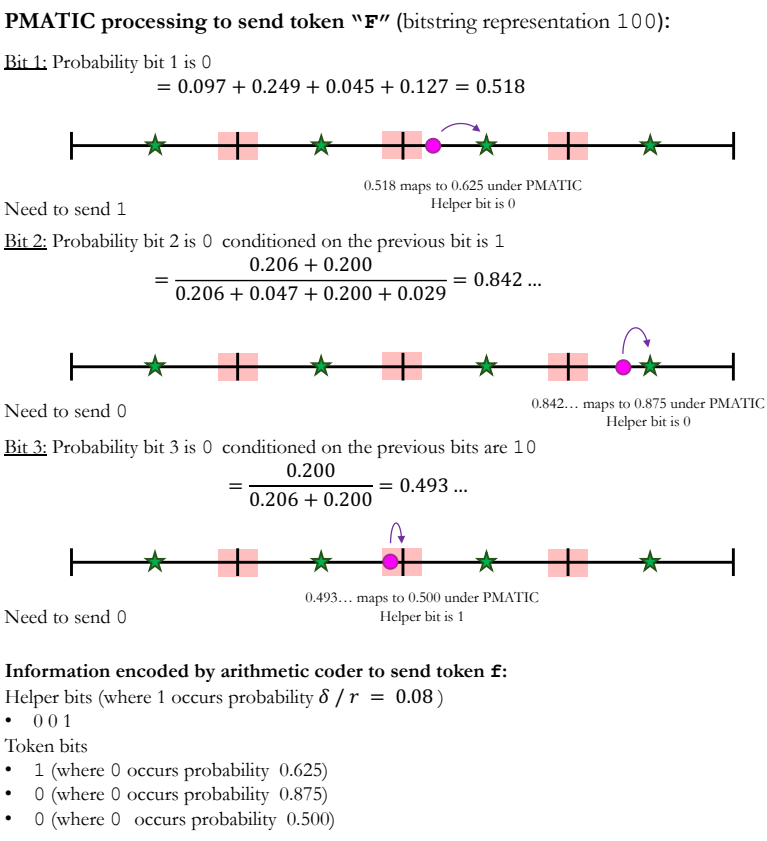

Figure 4: Illustration of PMATIC encoding for token 'F'. The table in the figure shows the corresponding bitstring and model-computed probability for each token (the model-computed probability depends on context).

A.2 LLM Usage

We used LLMs (specifically GPT-5) as an assistant for the background literature search, writing, and coding. This entailed asking the LLM to: search for and summarize related papers; write sample paragraphs, which we could then use as a guideline for our own writing or take phrases from; and explain any terms we came across which we were unsure of.

For the code for our experiments, we consulted LLMs in several different ways. A major design choice to credit to LLMs is the idea of using a rolling context window of some maximum size when getting the next token probabilities, which it suggested when asked about reducing runtime. We also asked LLMs to write various small parts of the code which are standard operations, for instance, a script to aggregate statistics for the experiments to be printed on the screen, a function that changes byte arrays to bitstrings, some helper functions to setup arithmetic coding when running without PMATIC, and even a one line function to compute entropy. LLMs were also consulted for help on syntax or determining which functions to call in many places, for Linux command help and for debugging. In the earlier iterations of our code, we used LLM generated code to setup the Llama

model, but later many of those critical parts were replaced. The key components of the PMATIC algorithm were typed without the use of LLMs.