TIME SERIES COMPRESSION WITH FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Time-series volumes continue to grow across domains, making efficient lossless compression essential for storage and transmission. Classical codecs and specialized numerical compressors often miss nonlinear or long-range temporal structure, while neural compressors that learn such structure typically demand per-dataset training and extra model storage. Large pre-trained foundation models offer strong sequence modeling with zero-shot transfer, suggesting a route to compression without task-specific fine-tuning. This paper connects next-token and next-value prediction to encoding-based compression and presents a unified zero-shot framework for numerical time series: LLMs drive arithmetic coding from predictive token distributions, and TSFMs drive residual coding from point forecasts. Lightweight numeric tokenization schemes are further studied without modifying frozen tokenizers. Extensive experiments show that publicly available foundation models can be applied directly; in particular, pre-trained LLMs attain higher compression ratios than the compared general-purpose, numerical, and TSFM-based compressors on most datasets, under a shared public-checkpoint accounting.

1 INTRODUCTION

Time series data consists of a sequence of data points recorded over time, and arises across many domains such as finance, meteorology, healthcare, transportation, and industry (Figure 1(a)) Lim & Zohren (2020). Typically, it is organized into two or more columns, with one column representing timestamps and the others representing data values aligned with those timestamps. Unlike traditional datasets, where data points are independent, time series data is inherently sequential, meaning each data point relies on past observations. This temporal characteristic makes time series data particularly well-suited for analyzing trends, patterns, and dependencies that evolve over time.

1.1 MOTIVATION

IoT and related applications produce rapidly growing time-series volumes that stress storage budgets, making compression a practical necessity. As illustrated in Figure 1, the right codec depends on where computation and data live, rather than on a single universal method. For storage (Figure 1(b)), client, edge, and cloud tiers offer very different resources and thus call for different compressors. CPU-limited clients can only afford lightweight traditional encoding. Edges with GPUs have more headroom and can pursue higher ratios toward a more compact archive. Cloud GPU clusters are richest and can run foundation-model (FM) encoding for persistent archival storage at the highest compression. Figure 1(c) further indicates that a global FM can be shared via an online decompression service, so its footprint need not inflate per-file storage. Given abundant cloud compute and the strong zero-shot forecasting of pre-trained foundation models Gruver et al. (2023); Jin et al. (2024); Das et al. (2024), a natural question is how to turn that predictive capacity into compression capacity for persistent cloud storage.

1.2 INTUITION

Traditional compressors such as LZMA lzm (2026) and GZIP gzi (2026) are easy to deploy, but they treat time series as opaque byte streams. They largely ignore temporal correlation, so ratios remain limited when dependence is nonlinear or long-range Zhou et al. (2021); Nie et al. (2023).

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.

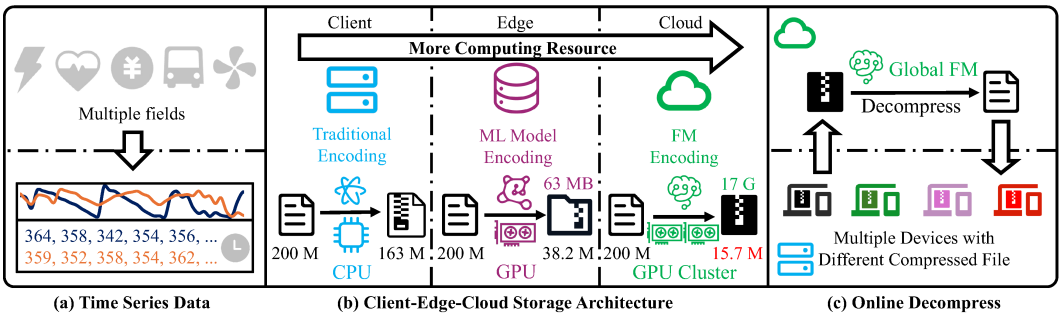


Figure 1: Time-series data sources, client-edge-cloud storage, and online decompression with a shared global FM.

Specialized numerical codecs improve on simple statistics, yet many still rely on fixed transforms or weak predictors and leave substantial redundancy.

Learned ML compressors can capture temporal dynamics and raise ratios Bommasani et al. (2021). Their drawback is adaptation cost: models are typically retrained on each corpus or file to be compressed. The resulting task-specific weights must be stored with the bitstream and cannot be omitted. Figure 1(b) makes this concrete: beyond a 38.2 MB code stream, an additional 63 MB of per-file model weights must be counted, so the effective storage is 63+38.2 MB.

Foundation models address this bottleneck through zero-shot transfer Gruver et al. (2023); Jin et al. (2024). Large-scale pre-training yields strong predictors that generalize without per-file fine-tuning, so one global checkpoint serves many datasets Zhao et al. (2023); Das et al. (2024). We use Large Language Models (LLMs) Zhao et al. (2023) and Time Series Foundation Models (TSFMs) Garza & Canseco (2023) in this role. As illustrated in Figure 1(c), the global FM is shared as an online service for multi-device decompression: devices call the same cloud-hosted model rather than bundling weights with each archive. The FM footprint is therefore not counted in per-file storage size, even though the model itself is large. Only the bitstream is persisted; Figure 1(b) already shows a strong preliminary gain, shrinking 200 MB to 15.7 MB versus 163 MB traditionally and 63+38.2 MB when ML weights are included.

1.3 Contribution

Our contributions are as follows.

(1) We connect predictive foundation models to classical encoding and present a unified zero-shot framework for time series compression (Figure 2), with LLM probability coding and TSFM residual coding.

(2) We instantiate the framework with standard LLMs and TSFMs on integer and floating-point series, without dataset-specific fine-tuning. The LLM path serializes numbers as text for next-token prediction and arithmetic coding.

(3) We study LLM tokenization of numeric inputs, propose lightweight token preprocessing, and measure how Text-Only, Number, and Space-Number schemes affect zero-shot compression.

(4) We evaluate multi-domain datasets by compression ratio and speed, and compare model families, scales, and runtime settings.

2 Data Compression Based on Models

This section develops the theoretical link between predictive modeling and lossless compression: why forecasting removes redundancy, and how training objectives align with code length Townsend et al. (2019); Delétang et al. (2024). When a model outputs a probability distribution (e.g., an LLM), entropy coding is near-optimal; when it outputs point forecasts (e.g., a TSFM), residual

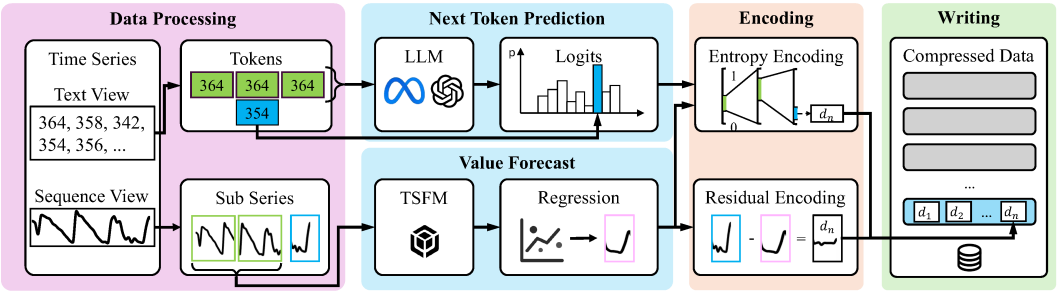


Figure 2: Overview of data types and the corresponding processing methods.

coding stores only prediction errors. Appendix A summarizes background on time-series traits and Large Sequence Models.

2.1 Data Compression by Forecasting

Data compression reduces storage via lossless or lossy methods: the former removes redundancy, while the latter discards detail within an acceptable tolerance. Machine learning models support both classification and regression. Classification discards much of the input to separate categories and is therefore poorly suited to reconstruction; regression, by forecasting the data itself, preserves enough information for recovery and thus aligns with compression. In practice, model-based compressors encode forecasts or their residuals rather than the raw stream, so that redundancy captured by the model need not be stored twice Townsend et al. (2019).

Learning theory further supports this view: model effectiveness rests on capturing the data distribution in training and generalizing to new samples Kawaguchi et al. (2017). A larger hypothesis space increases the chance of approximating the true distribution; once that distribution is well captured, new data can be predicted from context, and a finite model can represent an arbitrarily long stream more compactly than storing the stream itself. The same holds for deep networks: with enough parameters, even simple architectures can express rich distributions Zhang et al. (2017). Forecasting-based compression is therefore both principled and practical.

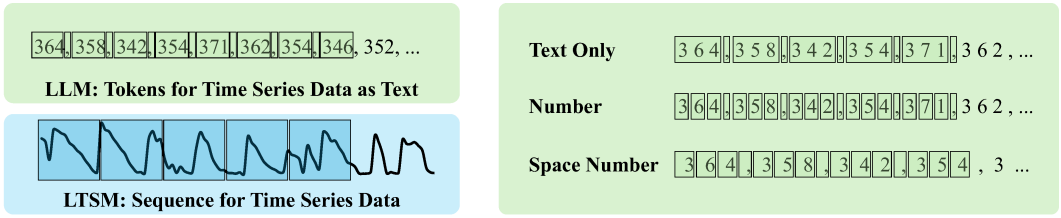
Figure 2 instantiates this idea as a dual-path pipeline over the same time series. Under a *text view*, values are discretized into tokens and fed to an LLM for next-token prediction, whose output distribution drives *entropy encoding*; under a *sequence view*, the series is partitioned into sub-series and forecast by a TSFM, after which only prediction residuals are retained via *residual encoding*. Figure 3(a) contrasts these two representations: the LLM path serializes each numerical point as text and treats the resulting symbols as discrete tokens, whereas the TSFM path keeps the series continuous and segments it into contiguous sub-sequences that serve as model inputs. Encoded units from either path are then aggregated into a compressed bitstream for storage. We defer classical coding details to Appendix B.

2.2 From Model Training to Data Compression

Prior work has linked language modeling to compression through scaling laws and empirical coding results Valmeekam et al. (2023); Delétang et al. (2024); Pan et al. (2025); Mao et al. (2025). Here we briefly recall the classical information-theoretic view that motivates using a frozen foundation model as a zero-shot probability (or residual) engine; we do not claim a new learning-theoretic result beyond this connection.

Let $x_{1:n}$ be a discrete sequence and write the autoregressive factorization $P_\theta(x_{1:n}) = \prod_{t=1}^{n} P_\theta(x_t \mid x_{<t})$. Training maximizes the likelihood of observed symbols, or equivalently minimizes the negative log-likelihood (NLL) in bits

$$\mathcal{L}(\theta) = -\sum_{t=1}^{n} \log_2 P_\theta(x_t \mid x_{<t}) = -\log_2 P_\theta(x_{1:n}).$$



(a) Tokens in LLM and LTSM. **(b) Token reprocessing methods for numerical data.**

Figure 3: Token representations for LLM/TSFM inputs and numerical preprocessing.

Under arithmetic coding matched to P_θ, the lossless codelength $\ell(x_{1:n})$ of the whole sequence satisfies Delétang et al. (2024)

$$\mathcal{L}(\theta) \;\le\; \ell(x_{1:n}) \;<\; \mathcal{L}(\theta) + O(1),$$

i.e., the bitstream length coincides with the sequence NLL up to a sequence-level additive constant. Hence minimizing $\mathcal{L}(\theta)$ is equivalent (up to $O(1)$) to minimizing the number of bits emitted by the coder.

Taking expectation under the data-generating distribution $P^\star$ further yields the cross-entropy decomposition

$$\mathbb{E}_{x\sim P^\star}\big[-\log_2 P_\theta(x)\big] = H(P^\star) + D_{\mathrm{KL}}(P^\star \,\|\, P_\theta) \;\ge\; H(P^\star),$$

with equality if and only if $P_\theta = P^\star$. Thus any reduction of the predictive KL gap tightens the achievable rate toward the Shannon limit $H(P^\star)$. When the test distribution Q differs from the pre-training distribution, the expected rate becomes $H(Q) + D_{\mathrm{KL}}(Q \,\|\, P_\theta)$; a strong public checkpoint therefore compresses well in a zero-shot regime whenever Q lies near the pre-training support, which is precisely the setting we exploit for numerical time series.

For TSFM-style value predictors that emit point forecasts rather than token probabilities, the same principle applies after residualization. If $\hat{x}_t$ denotes the forecast, the residual stream $r_t = x_t - \hat{x}_t$ typically has smaller dynamic range and lower entropy than x_t, so subsequent entropy (or general-purpose) coding of $\{r_t\}$ converts improved predictive accuracy into a shorter bitstream. Our contribution is to operationalize this principle for numerical time series with LLM and TSFM backbones, rather than to redefine the NLL–compression equivalence.

3 Time Series Compression Based on Models

This section details the compression architecture for LLMs and TSFMs, following the dual-path pipeline in Figure 2. Both paths share three stages: data processing, forecasting, and encoding. They differ in representation and coding. LLMs operate on tokens with entropy coding, while TSFMs operate on numerical subsequences with residual coding. We elaborate on each path below.

3.1 Compression Based on LLM

The LLM-based framework has three components. *Token representation* preprocesses inputs into model tokens. *Next-token prediction* yields a distribution over the vocabulary. *Encoding* compresses tokens under that distribution. Because tokenizer strongly affects compression quality, we focus on token representation to improve predictions without modifying the pre-trained tokenizer.

3.1.1 Data Processing: Token Representation

Prior work such as LLMTime Gruver et al. (2023) shows that token inputs substantially affect prediction quality. Altering the tokenizer of a pre-trained LLM is impractical, since the model was trained with a fixed vocabulary. We therefore reshape the input surface form so that the existing tokenizer segments numbers more favorably. Time series do not naturally match LLM pre-training corpora, so preprocessing is required. Consecutive values are separated by commas into a single line of text. For floating-point series we apply no quantization: each value is written as plain decimal

text and consumed as ordinary characters. Under the Text-Only arithmetic-coding baseline, each such character is a coding symbol. Symbol frequencies are counted over the entire file and used for static arithmetic coding, as in ordinary text compression.

Different LLM tokenizers handle numbers differently. LLAMA2 Touvron et al. (2023) splits numbers into individual digit tokens, whereas GPT2 Radford et al. (2019) groups them into one- to three-digit chunks. By restricting GPT2 to a set of "good" tokens, however, it can also emit one token per digit like LLAMA2. Inserting delimiters further reshapes the token stream. Figure 3(b) illustrates three representations under GPT2 tokenization. In *Text-Only*, a three-digit sequence is treated as separate tokens. In *Number*, admitting digits, signs, and commas yields per-digit segmentation. In *Space Number*, a space before each digit makes that digit a standalone token. These techniques also apply to categorical data. Tokenization changes the semantics seen by the LLM Gruver et al. (2023). Because ordinary text often relies on spaces, Space Number treats each digit as independent and breaks multi-digit integrity. Number instead emits digit tokens without inter-digit delimiters, so the model tends to read them as parts of a complete number, with commas as structural cues. LLM compression of time series varies markedly across these schemes, as measured in Section 4.

3.1.2 Forecasting: Next Token Prediction

After tokenization, the LLM performs next-token prediction. We maintain a sliding window whose size equals the context length. Window tokens are always taken from the original data, so encoding errors do not accumulate in the context. After one or more tokens are encoded, the window advances to the current position. A larger window supplies richer context and typically more accurate predictions, but raises latency. The prediction horizon is also adjustable. Forecasting several tokens at once reuses previously generated symbols, which can accumulate error while improving throughput. In practice, the final layer outputs a vocabulary distribution. Given fixed inputs and disabled randomness, sampling is deterministic, and each context maps to a well-defined distribution for encoding.

3.1.3 Encoding

Entropy coding nearly achieves the information-theoretic optimum and pairs naturally with a predictive distribution. We therefore adopt arithmetic coding. Because the sliding window starts empty, a short prefix of real tokens is written uncompressed. Tokens in the window are fed to the LLM to obtain the next-token distribution, which is projected onto the current interval in $[0, 1]$. The sub-interval of the true next token becomes the coding range for the following step. This process continues until all tokens are processed. The shortest binary string in the final interval is then emitted as the codeword.

Figure 4(a) illustrates one step with prediction length 1. When the coder occupies the interval $[0.3, 0.4]$, the window tokens from "119" through "114" are passed to the LLM. The resulting distribution is projected onto $[0.3, 0.4]$, and the true token "116" falls in $[0.32, 0.38]$. The shortest binary string in this interval (01011) is taken as the encoded fragment. By Shannon's bound, a symbol of probability p costs $-\log p$ bits, and the cost vanishes as $p \to 1$. Strong LLM predictions are often peaked. When the true token lands in a high-probability bin, its code length is minimal and encoding is efficient.

3.2 Data Compression Based on TSFM

Analogous to the LLM path, the TSFM framework has three stages: sequence representation, value forecast, and encoding. Rather than serializing numbers as text, this path keeps the series numerical and forecasts contiguous subsequences with a frozen TSFM. Because the outputs are typically point estimates rather than token probabilities, encoding focuses on residuals of these forecasts.

3.2.1 Data Processing: Sequence Representation

Numerical series have a fixed format, so TSFM pipelines leave little room for surface-form preprocessing. We describe how values enter the model directly. Typical TSFMs embed numerical inputs, model sequential dependencies in embedding space, and map the next embedding back to a scalar. TimesFM Das et al. (2024) is a representative example. Newer models such as Timer Liu et al.

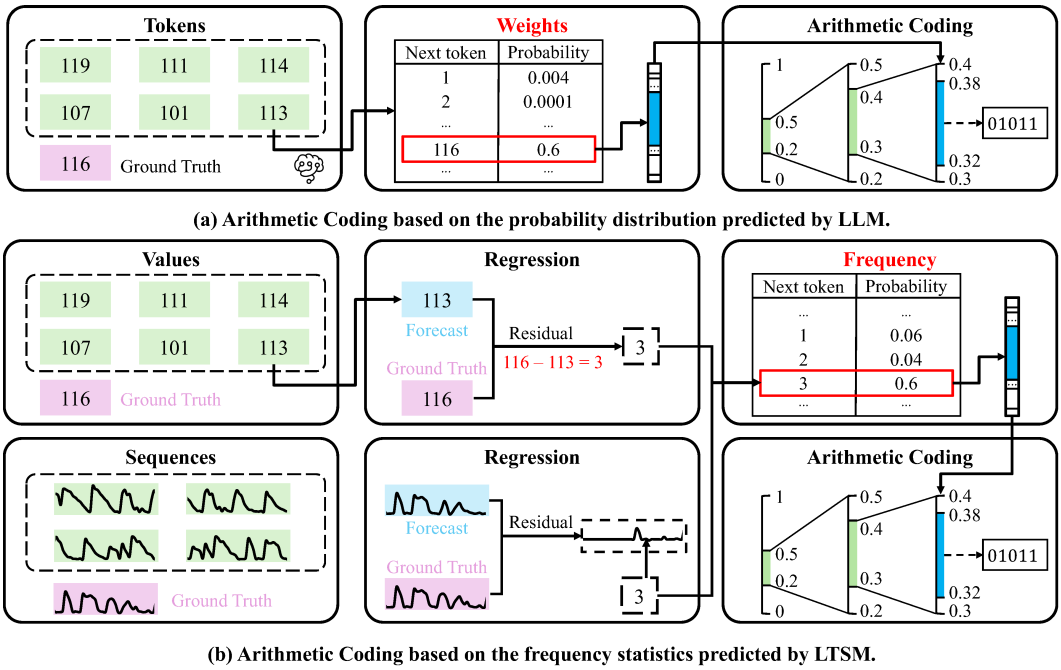


Figure 4: Arithmetic coding based on foundation-model predictions.

(2024) segment the series into fixed-length subsequences. Each subsequence is embedded as a unit akin to an LLM token and forecast at that level.

3.2.2 Forecasting: Value Forecast

TSFM forecasts operate on these minimal units. Unlike LLM next-token prediction, the output is typically a point estimate from regression rather than a full distribution. Some models, such as Sundial, produce distributions but expose only averages over samples, which limits their use for entropy coding. Given point forecasts, we keep a sliding window of raw data points so that residuals do not feed back into the context. The context length is aligned with the LLM setting.

3.2.3 Encoding

Because forecasts are definite values rather than distributions, arithmetic coding cannot be applied directly to model outputs. We instead store differences between predicted and actual values, known as *residual encoding*. Exact matches need no bits. Even small residuals shrink the value range and make the stream more compressible Xiao et al. (2022). Residual encoding is a transform rather than a complete codec, so a later numerical or general-purpose compressor can further reduce the residual stream. For a fair comparison with the LLM path, we also apply arithmetic coding to the residuals.

Figure 4(b) illustrates the process. Whether forecasts are scalar- or subsequence-valued, residuals are formed against the ground truth. One value-based arithmetic coder estimates frequencies over residual symbols. A simpler character-based variant serializes residuals as comma-delimited text and applies the same Text-Only rule. Each character is a coding symbol, and frequencies are counted over the full residual text for static arithmetic coding. Both variants use fixed distributions, so their bit rates follow the residual empirical entropy.

4 Experimental Evaluation

We evaluate foundation-model compressors on numerical time series and compare them with classical baselines, following the LLM and TSFM pipelines in Sections 3.1 and 3.2. Experiment code is available at sou (2026). Key settings are summarized below; full details appear in Appendix C.

Table 1: Compression performance on numerical datasets. **Bold** / underline: best / second-best in each Compression-ratio and Speed column.

Token Processing	Model	Compressor	Compression Ratio						Average Speed (Points/s)
			City-T	TH-G	ZH-R	USGS-E	TH-C	Cyber-V	
-	-	Lzma	4.312	7.077	7.099	3.924	10.614	3.836	$\mathbf{3.314 \times 10^5}$
-	-	Gorilla	0.724	1.226	0.717	1.363	3.864	1.750	$\underline{2.310 \times 10^5}$
Text Only	-	AC	2.352	2.135	2.347	2.638	2.668	1.993	8.257×10^4
	LLAMA2-7b-hf	AC	<u>7.064</u>	<u>12.482</u>	<u>11.943</u>	4.657	**11.864**	<u>5.458</u>	6.767
	LLAMA2-13b-hf		**7.081**	**12.692**	**12.219**	<u>4.682</u>	<u>11.705</u>	**5.471**	4.347
	GPT2		6.524	7.886	7.682	4.106	8.522	4.570	19.459
	GPT2-medium		6.783	8.541	8.154	4.174	9.551	4.731	14.202
	GPT2-large		6.789	8.875	8.521	4.295	10.219	4.747	9.603
Number	GPT2	AC	5.265	3.918	4.210	3.223	5.625	3.455	22.959
	GPT2-medium		2.788	3.289	2.890	2.561	5.735	3.331	11.055
	GPT2-large		5.391	4.013	5.343	3.299	6.481	3.896	6.062
Space Number	GPT2	AC	5.969	8.082	7.046	4.104	7.565	4.618	22.248
	GPT2-medium		6.189	8.395	7.621	4.079	9.326	4.732	11.239
	GPT2-large		6.294	9.236	7.721	4.219	9.814	4.967	6.169
Value	-	AC	1.270	1.357	1.002	**4.753**	4.714	1.779	1.090×10^5
	TimesFM	AC	1.308	1.305	1.111	4.067	5.997	1.694	1.664
	Chronos2		1.322	1.433	1.110	4.161	6.052	1.655	39.38
Sequence	Timer-XL	AC	1.250	1.363	1.076	3.811	5.786	1.520	276.301

4.1 Experimental Settings

We evaluate six numerical series that comprise three float and three integer datasets from public and private sources: City-Temp Ustinian (2020), USGS-Earthquakes USG (2024), TH-GPS, ZH-Root, TH-Climate, and Cyber-Vehicle (one column per dataset; Table 2). Classical baselines are LZMA, character-level static arithmetic coding (AC), and Gorilla. Foundation-model compressors use frozen LLMs (LLaMA 2, GPT-2) with AC under Text-Only / Number / Space-Number tokenization, and TSFMs (TimesFM, Timer-XL, Chronos2) with residual coding followed by AC (Table 3). We report compression ratio (original / compressed; higher is better) and throughput (points/s). Model weights are excluded under the shared public-checkpoint assumption (Section 1); measured speeds are inference-dominated (Figure 1). Full hardware, dataset, baseline, and metric details, including Tables 2 and 3, are in Appendix C.

4.2 Performance on Time Series Data

We evaluate numerical time series under the same LLM tokenization schemes and under value-based arithmetic coding for TSFMs; results appear in Table 1. City-T(emp), TH-G(PS), and ZH-R(oot) are float series; USGS-E(arthquake), TH-C(limate), and Cyber-V(ehicle) are integer series. Baselines include the numerical codec Gorilla, character-level Text-Only AC (per-character frequencies over the full file, covering digits, signs, and decimal points for floats), and value-based AC.

On most datasets, suitably configured LLMs with arithmetic coding outperform Gorilla, character-level AC, LZMA, and TSFM-based compressors, showing that LLMs can extract temporal structure even when numbers are presented as text. Across families, LLaMA 2 typically yields higher ratios than GPT-2. Within GPT-2, scaling GPT-2 $\rightarrow$ medium $\rightarrow$ large improves ratios in most Text-Only and Space-Number settings, indicating that extra capacity along this controlled ladder helps next-token modeling for compression. Cross-family or near-saturated comparisons need not be strictly monotonic (e.g., LLaMA 2-7B vs. 13B on TH-C), so scale is useful but not decisive alone. An exception is USGS-E, where value-based AC attains the best ratio owing to a small discrete domain and a skewed frequency distribution.

Tokenization likewise matters. For GPT-2, Text-Only, which keeps each numeric literal as a coherent textual unit before tokenization, usually compresses better than digit-level Number splitting, with Space-Number typically in between (Table 1). Aggressive Number splitting breaks multi-digit and floating-point structure, forcing the model to decide whether the next token continues a number or begins a delimiter and thereby weakening probability estimates. Prefixed spaces reduce spurious merges and partially restore digit-wise predictability, yet still discard the default multi-digit grouping that Text-Only preserves. For numerical time series, retaining complete numeric literals in the serialized text is therefore generally preferable to forcing per-digit segmentation.

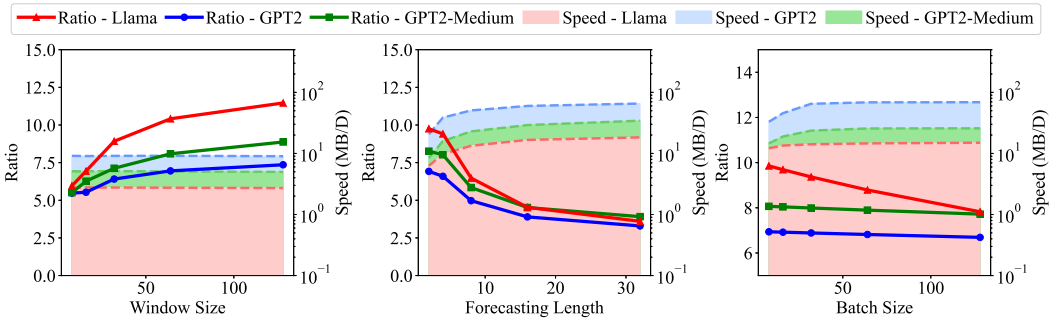


Figure 5: Compression performance of variable parameters on TH-Climate dataset.

4.3 Parameter Tuning on Time Series Data

Figure 5 reports the effect of window size, forecasting length, and batch size on TH-Climate. Across models, larger windows steadily raise the compression ratio until saturation, while throughput stays roughly flat or declines only modestly. Numerical series are dominated by short- to mid-range autocorrelation rather than the long, template-level repetitions that reward very long context, which explains the limited gains. Extending the forecasting length consistently hurts compression while slightly improving speed: multi-step rollouts accumulate error, though the penalty is milder than on log data, where discrete template dependencies amplify error propagation. Batch size mainly affects system throughput: larger batches raise points-per-second while leaving the ratio nearly unchanged. Together with Table 1, these trends confirm that foundation-model ratio gains come with substantially lower throughput than classical codecs, so deployment should follow the cost trade-off in Figure 1.

5 Related Work

Existing studies related to this work can be categorized into three interconnected areas: data forecasting, data encoding and compression, and model-based compression. Data forecasting examines temporal dependencies in time-series data, establishing patterns that improve compression efficiency. Encoding and compression techniques focus on reducing redundancy through statistical or transformation-based methods, but often struggle with complex, non-stationary sequences. Model-based compression merges predictive modeling with encoding principles for more adaptive and efficient data representation.

5.1 Data Forecasting

In recent years, large pre-trained models have excelled in time series prediction due to their strong extrapolation and generation capabilities. Some research proposes using large language models for time series tasks. FPT Zhou et al. (2023) employs GPT-2 to extract sequence representations. LLMTime Gruver et al. (2023) encodes time series as text, predicting the next word in the sequence. This indicates that models like GPT and LLAMA can extrapolate time series from zero samples, achieving performance on par with or better than specialized time series models. Time-LLM Jin et al. (2024) aligns time series with text prototypes and inputs them into a frozen LLM, excelling in few-shot and zero-shot scenarios.

Additionally, some research focuses on training large models specifically for time series data Garza & Canseco (2023); Dooley et al. (2023); Liu et al. (2024); Das et al. (2024). TimeGPT-1 Garza & Canseco (2023) is the first foundational model trained on extensive time series datasets. Various fundamental time series models have since emerged, excelling in zero-shot prediction, including ForecastPFN Dooley et al. (2023), TimesFM Das et al. (2024), Timer Liu et al. (2024), Timer-XL Liu et al. (2025a), Sundial Liu et al. (2025b) and Chronos2Ansari et al. (2025).

5.2 Data Encoding and Compressing

Time series data compression is crucial for efficient data storage and transmission. Dictionary-based methods represent recurring segments with shorter codes from a pre-built dictionary. Although effective for repetitive data, their performance declines with novel patterns, making dictionary maintenance challenging.

Other methods combine simple encoding techniques like delta encoding Suel (2019), run-length encoding (RLE) Golomb (1966), and XOR coding. Gorilla Pelkonen et al. (2015) uses delta-of-delta for timestamps and XOR for values to eliminate common zeros. TS2DIFF Xiao et al. (2022) applies second-order differencing and bit-packing to minimize the value range. While efficient, these methods depend on fixed patterns, limiting their adaptability to complex data.

Prediction-based methods leverage temporal autocorrelation by forecasting the next value and encoding the residual (the difference). Sprintz Blalock et al. (2018) uses delta-encoding and FIRE for predictions, with smaller residuals improving compression. This approach closely relates to our work, bridging forecasting and compression.

5.3 Compressing by Models

Following the rise of deep neural networks, a few works have emerged that focus on data compression using neural network-based approaches. DZip Goyal et al. (2020) is a general-purpose compressor for sequential data that uses neural networks for probability prediction followed by arithmetic coding. NNCP Bellard (2019; 2021), based on Transformers Vaswani et al. (2017), excels in text compression through adaptive model fine-tuning. However, these models often require retraining when faced with new data, significantly reducing the applicability of these methods on small-scale datasets.

Recent large pre-trained models significantly expand the "model-as-compressor" paradigm, with sequence modeling capabilities making them ideal for probability estimation. LLMZip Valmeekam et al. (2023) demonstrated that models like LLAMA-7B, combined with arithmetic coding, achieve strong text compression ratios. Further, Delétang et al. (2024) established that "Language Modeling Is Compression," showing that large LMs can rival domain compressors when next-token prediction is used as coding. Despite this progress, applying foundation models to *numerical time-series* compression in a zero-shot setting, including TSFM residual pipelines and numeric tokenization choices, remains comparatively underexplored. While studies like LLMTime Gruver et al. (2023) adapt LLMs for time-series forecasting, their use as compressors has not been fully investigated.

6 Conclusion

Lossless compression of numerical time series is studied with frozen foundation models in a zero-shot regime in this paper. Building on the view that predictive modeling and coding are two sides of the same coin, a unified dual-path framework is presented: LLMs serialize values as text and drive arithmetic coding via next-token distributions, while TSFMs emit point forecasts whose residuals are subsequently encoded. Without altering pre-trained tokenizers, lightweight numeric preprocessing (Text-Only, Number, and Space-Number) is further examined for its effect on token boundaries and compression quality. On multi-domain integer and floating-point datasets, appropriately configured LLMs with arithmetic coding outperform, on most series, the compared general-purpose, numerical, and TSFM-based compressors, under a shared public-checkpoint accounting that excludes model weights from the bitstream. Compression quality tracks predictive fit and is highly sensitive to serialization: retaining complete numeric literals (Text-Only) generally beats forced digit-level splits. Within a single family such as GPT-2, larger variants tend to compress better, and longer context further improves ratios until saturation. Across families, however, compression depends more on architecture and pre-training than on parameter count alone. These results indicate that foundation-model compressors offer strong ratio gains for archival storage and bandwidth-bound pipelines that can amortize a shared public checkpoint.

AI USE STATEMENT

In this work, we used generative AI tools for editing the research paper to improve readability and for retrieving bibliographic references. After completing the full draft, the authors utilized generative AI to polish the text; the role of the AI tool was limited to checking technical wording and grammar so as to improve precision, clarity, and stylistic consistency. We have reviewed all AI-assisted work. Every suggested change was examined and approved by the authors, and all references suggested by generative AI were read and screened by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies lossless compression of time-series data with foundation models and does not involve human subjects, sensitive personal data, or dual-use applications. We have read the ICLR Code of Ethics and believe that this submission does not raise ethics concerns under that code.

REPRODUCIBILITY STATEMENT

We provide anonymous source code and experiment scripts for reproducing the reported results at sou (2026). Experimental settings are summarized in Section 4, with further details in Appendix C. Using the released repository together with the publicly available models and datasets described in the paper, the main experiments can be reproduced.

REFERENCES

https://www.kaggle.com/datasets/rupindersinghrana/usgs-earthquakes-2024, 2024.

https://www.gzip.org, 2026.

https://www.gnu.org/software/gzip/, 2026.

https://anonymous.4open.science/r/LLM4TSC-5C72, 2026.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting. *CoRR*, abs/2510.15821, 2025. doi: 10.48550/ARXIV.2510.15821. URL https://doi.org/10.48550/arXiv.2510.15821.

Fabrice Bellard. Lossless data compression with neural networks. 2019. URL https://api.semanticscholar.org/CorpusID:211241644.

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

Davis W. Blalock, Samuel Madden, and John V. Guttag. Sprintz: Time series compression for the internet of things. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(3):93:1–93:23, 2018. doi: 10.1145/3264903. URL https://doi.org/10.1145/3264903.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ B. Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren E. Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang,

Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021. URL https://arxiv.org/abs/2108.07258.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=jn2iTJas6h.

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. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=jznbgiynus.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/V1/N19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Samuel Dooley, Gurnoor Singh Khurana, Chirag Mohapatra, Siddartha V. Naidu, and Colin White. Forecastpfn: Synthetically-trained zero-shot forecasting. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/0731f0e65559059eb9cd9d6f44ce2dd8-Abstract-Conference.html.

Jarek Duda. Asymmetric numeral systems as close to capacity low state entropy coders. *CoRR*, abs/1311.2540, 2013. URL http://arxiv.org/abs/1311.2540.

Philip Gage. A new algorithm for data compression. *C Users Journal*, 12(2):23–38, 1994.

Azul Garza and Max Mergenthaler Canseco. Timegpt-1. *CoRR*, abs/2310.03589, 2023. doi: 10.48550/ARXIV.2310.03589. URL https://doi.org/10.48550/arXiv.2310.03589.

Solomon W. Golomb. Run-length encodings (corresp.). *IEEE Trans. Inf. Theory*, 12(3):399–401, 1966. doi: 10.1109/TIT.1966.1053907. URL https://doi.org/10.1109/TIT.1966.1053907.

Mohit Goyal, Kedar Tatwawadi, Shubham Chandak, and Idoia Ochoa. Dzip: Improved general-purpose lossless compression based on novel neural network modeling. In Ali Bilgin, Michael W. Marcellin, Joan Serra-Sagristà, and James A. Storer (eds.), *Data Compression Conference, DCC 2020, Snowbird, UT, USA, March 24-27, 2020*, pp. 372. IEEE, 2020. doi: 10.1109/DCC47342.2020.00065. URL https://doi.org/10.1109/DCC47342.2020.00065.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/3eb7ca52e8207697361b2c0fb3926511-Abstract-Conference.html.

Paul G. Howard and Jeffrey Scott Vitter. Parallel lossless image compression using huffman and arithmetic coding. *Inf. Process. Lett.*, 59(2):65–73, 1996. doi: 10.1016/0020-0190(96)00090-7. URL https://doi.org/10.1016/0020-0190(96)00090-7.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-llm: Time series forecasting by reprogramming large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Unb5CVPtae.

Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. *CoRR*, abs/1710.05468, 2017. URL http://arxiv.org/abs/1710.05468.

Bryan Lim and Stefan Zohren. Time series forecasting with deep learning: A survey. *CoRR*, abs/2004.13408, 2020. URL https://arxiv.org/abs/2004.13408.

Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Generative pre-trained transformers are large time series models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=bYRYb7DMNo.

Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer-xl: Long-context transformers for unified time series forecasting. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025a. URL https://openreview.net/forum?id=KMCJXjlDDr.

Yong Liu, Guo Qin, Zhiyuan Shi, Zhi Chen, Caiyin Yang, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Sundial: A family of highly capable time series foundation models. *CoRR*, abs/2502.00816, 2025b. doi: 10.48550/ARXIV.2502.00816. URL https://doi.org/10.48550/arXiv.2502.00816.

Yu Mao, Holger Pirk, and Chun Jason Xue. Lossless compression of large language model-generated text via next-token prediction. *CoRR*, abs/2505.06297, 2025. doi: 10.48550/ARXIV.2505.06297. URL https://doi.org/10.48550/arXiv.2505.06297.

Detlev Marpe, Gabi Blattermann, Guido Heising, Thomas Wiegand, and Heiko Schwarz. Context-based adaptive binary arithmetic coding in JVT/H.26L. In *Proceedings of the 2002 International Conference on Image Processing, ICIP 2002, Rochester, New York, USA, September 22-25, 2002*, pp. 513–516. IEEE, 2002. doi: 10.1109/ICIP.2002.1040000. URL https://doi.org/10.1109/ICIP.2002.1040000.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=Jbdc0vTOcol.

Ghim Hwee Ong and Shell-Ying Huang. A data compression scheme for chinese text files using huffman coding and a two-level dictionary. *Inf. Sci.*, 84(1&2):85–99, 1995. doi: 10.1016/0020-0255(94)00108-N. URL https://doi.org/10.1016/0020-0255(94)00108-N.

Zhixuan Pan, Shaowen Wang, and Jian Li. Understanding LLM behaviors via compression: Data generation, knowledge acquisition and scaling laws. *CoRR*, abs/2504.09597, 2025. doi: 10.48550/ARXIV.2504.09597. URL https://doi.org/10.48550/arXiv.2504.09597.

Tuomas Pelkonen, Scott Franklin, Paul Cavallaro, Qi Huang, Justin Meza, Justin Teller, and Kaushik Veeraraghavan. Gorilla: A fast, scalable, in-memory time series database. *Proc. VLDB Endow.*, 8(12):1816–1827, 2015. doi: 10.14778/2824032.2824078. URL http://www.vldb.org/pvldb/vol8/p1816-teller.pdf.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016. doi: 10.18653/V1/P16-1162. URL https://doi.org/10.18653/v1/p16-1162.

Claude E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423, 1948. doi: 10.1002/J.1538-7305.1948.TB01338.X. URL https://doi.org/10.1002/j.1538-7305.1948.tb01338.x.

Torsten Suel. Delta compression techniques. In Sherif Sakr and Albert Y. Zomaya (eds.), *Encyclopedia of Big Data Technologies*. Springer, 2019. doi: 10.1007/978-3-319-63962-8_63-1. URL https://doi.org/10.1007/978-3-319-63962-8_63-1.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023. doi: 10.48550/ARXIV.2307.09288. URL https://doi.org/10.48550/arXiv.2307.09288.

James Townsend, Thomas Bird, and David Barber. Practical lossless compression with latent variables using bits back coding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=ryE98iR5tm.

Ustinian. https://www.heywhale.com/mw/dataset/5ee4750d71a691002d548906, 2020.

Chandra Shekhara Kaushik Valmeekam, Krishna Narayanan, Dileep Kalathil, Jean-François Chamberland, and Srinivas Shakkottai. Llmzip: Lossless text compression using large language models. *CoRR*, abs/2306.04050, 2023. doi: 10.48550/ARXIV.2306.04050. URL https://doi.org/10.48550/arXiv.2306.04050.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Jinzhao Xiao, Yuxiang Huang, Changyu Hu, Shaoxu Song, Xiangdong Huang, and Jianmin Wang. Time series data encoding for efficient storage: A comparative analysis in apache iotdb. *Proc. VLDB Endow.*, 15(10):2148–2160, 2022. doi: 10.14778/3547305.3547319. URL https://www.vldb.org/pvldb/vol15/p2148-song.pdf.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Sy8gdB9xx.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *CoRR*, abs/2303.18223, 2023. doi: 10.48550/ARXIV.2303.18223. URL https://doi.org/10.48550/arXiv.2303.18223.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 11106–11115. AAAI Press, 2021. doi: 10.1609/AAAI.V35I12.17325. URL https://doi.org/10.1609/aaai.v35i12.17325.

Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pretrained LM. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/86c17de05579cde52025f9984e6e2ebb-Abstract-Conference.html.

A Background

This appendix provides background for foundation-model compression of time series. We first summarize numerical series traits that affect codec choice, then introduce *Large Sequence Models* as an umbrella for LLMs and TSFMs, covering representation, sequence modeling, and a compression perspective.

A.1 Time Series Data

Numerical time series consist of integers or floating-point values recorded over time. Their statistical structure strongly affects which codecs work well. Prior work groups the most relevant traits into four categories: Scale, Delta, Repeat, and Increase Xiao et al. (2022). Scale governs bit-packing efficiency: wider ranges need more bits, while narrower ranges admit denser encodings. Small and stable deltas favor differential coding. Repeated values improve run-length encoding Golomb (1966) by exposing redundancy. Monotonically increasing series yield non-negative deltas, which simplifies unsigned packing and related encodings. These traits jointly guide the choice and tuning of compression methods.

A.2 Large Sequence Model

The foundation models studied here are mainly LLMs and TSFMs. We use *Large Sequence Model* as an umbrella term for both, emphasizing a shared sequential modeling stack across modalities. Despite modality differences, their design can be organized into three components: representation, sequence modeling, and a compression view of model outputs.

A.2.1 Representation

Raw sequences must first be mapped into model inputs. For language, this is typically tokenization via word units, subword methods such as Byte Pair Encoding (BPE) Gage (1994); Sennrich et al. (2016) or WordPiece Devlin et al. (2019), or character-level units. For time series, continuous signals are often quantized into discrete tokens or split into fixed-length patches that are embedded as vectors. These strategies cast heterogeneous data as token sequences under a unified modeling interface.

Figure 3(a) contrasts the two representations used in this work. LLMs consume text, so numerical series are serialized with delimiters before tokenization. Different LLMs tokenize the resulting string differently, but values and delimiters are treated as independent tokens to preserve numeric structure. TSFMs instead operate on numerical inputs and segment the series into patches or related units for embedding and subsequent modeling.

A.2.2 SEQUENCE MODELING

At the core of large sequence models is next-step prediction. LLMs predict the next text token, such as the next subword in a sentence. TSFMs predict the next measurement or the next interval of a numerical series. Transformer architectures dominate these tasks because self-attention captures both short- and long-range dependencies across modalities Vaswani et al. (2017).

A.2.3 COMPRESSION PERSPECTIVE

From a compression standpoint, the two model families contribute differently. For LLMs, predictive token distributions can be paired directly with entropy coding. TSFMs typically emit point forecasts rather than discrete probabilities, so compression stores only residual errors after prediction. This residual path reduces redundancy while remaining lossless when residuals are coded exactly. For decoder-side reconstruction, the pipeline must be free of sampling noise so that identical inputs and checkpoints yield identical forecasts.

B ENTROPY CODING AND RESIDUAL CODING

This appendix reviews the two classical coding ingredients used in our dual-path framework. Entropy coding compresses discrete symbols under a probability model and pairs naturally with LLM next-token distributions. Residual coding stores prediction errors of numerical forecasts and is the counterpart for TSFM point predictors. We summarize each technique below.

B.1 ENTROPY CODING

Entropy coding is a standard lossless technique that assigns code lengths according to symbol probabilities. By Shannon's source coding theorem, the average code length cannot fall below the source entropy Shannon (1948). Efficient coders therefore give shorter codewords to more frequent symbols.

Classical instantiations include Huffman coding Ong & Huang (1995) and arithmetic coding Howard & Vitter (1996). Huffman coding builds a variable-length prefix code, while arithmetic coding represents a message as a subinterval of $[0, 1)$ and can approach the entropy bound with fractional-bit precision. Related methods such as range coding, CABAC Marpe et al. (2002), and asymmetric numeral systems (ANS) Duda (2013) trade compression efficiency against computational cost. In neural and LLM-based compressors, entropy coding turns predictive distributions into bitstreams; the closer the model matches the true distribution, the closer the rate approaches the entropy limit.

B.2 RESIDUAL CODING

Residual coding is effective for numerical time series when a predictor can estimate future values accurately. Instead of storing the raw signal, one stores residuals, namely the differences between actual and predicted values. Good predictors capture temporal structure, so residuals usually have smaller dynamic range and lower entropy than the original series Xiao et al. (2022).

Mapping the series to residuals concentrates mass near zero and reduces the bit width needed for representation. Later stages such as run-length encoding or entropy coding therefore become more efficient. In short, residual coding turns a high-variance series into a more compressible stream that standard codecs handle more readily.

C FULL EXPERIMENTAL SETTINGS

This appendix expands the experimental summary in Section 4.

C.1 HARDWARE

Experiments run on a server with two 32-core AMD EPYC3 7542 CPUs at 2.9 GHz, 256 GB RAM, and four NVIDIA A100 GPUs used for model inference.

Table 2: Real-world numerical datasets.

Dataset	Type	Points	Size (KB)
City-Temp (Ustinian (2020))	Float	100,000	502.6
TH-GPS	Float	3,441	33.2
ZH-Root	Float	35,666	205.6
USGS-Earthquakes (USG (2024))	Int	75,809	223.4
TH-Climate	Int	131,747	514.2
Cyber-Vehicle	Int	5,751	20.9

Table 3: Baseline models.

Model	Parameter	Model Size	Vocabulary Size
LLAMA2-7b-hf	7 B	≈ 13 GB	32,000
LLAMA2-13b-hf	13 B	≈ 25 GB	
GPT2	117 M	≈ 0.5 GB	50,257
GPT2-medium	345 M	≈ 1.3 GB	
GPT2-large	774 M	≈ 3 GB	
TimesFM	200 M	≈ 1.86 GB	-
Timer-XL	84 M	≈ 0.77 GB	-
Chronos2	0.1B	≈ 478 MB	-

C.2 Datasets

We evaluate three integer and three floating-point series drawn from both public and private sources. City-Temp and USGS-Earthquakes are public, and their download links appear in Table 2. The other four series, namely TH-GPS, ZH-Root, TH-Climate, and Cyber-Vehicle, are private. Among them, TH-GPS collects campus GPS traces from Tsinghua, TH-Climate records weather measurements, and Cyber-Vehicle contains vehicle-sensor readings with a low repeat rate. For each dataset we use one column, and Table 2 further reports sizes and types.

C.3 Baselines and Models

Baselines include character-level arithmetic coding and the general-purpose compressor LZMA on all datasets, together with Gorilla for numerical series. For the Text-Only arithmetic-coding baseline, integer and floating-point series are written as comma-separated plain text without quantization. Each character (digits, signs, decimal points, and delimiters) is one symbol; frequencies are estimated on the whole file for static arithmetic coding. Foundation models used in the benchmark are listed in Table 3. They include LLMs from LLaMA 2 Touvron et al. (2023) and GPT-2 Radford et al. (2019), as well as TSFMs TimesFM Das et al. (2024), Timer-XL Liu et al. (2025a), and Chronos2 Ansari et al. (2025). While both LLM families use subword tokenization, their tokenizers differ: LLaMA 2 uses SentencePiece BPE, whereas GPT-2 uses byte-level BPE. TSFM backends also differ: TimesFM uses patched decoding, Chronos2 yields probabilistic forecasts that we reduce to point values for residual coding, and Timer-XL predicts over subsequence patches.

C.4 Evaluation Metrics

The primary metrics are compression ratio and compression speed. The compression ratio is original size divided by compressed bitstream size (original / compressed); larger is better. Under the shared-model setting in Section 1, foundation-model weights are excluded from the compressed size: both ends use the same public checkpoint, and no dataset-specific parameters are stored. In foundation-model pipelines, both compression and decompression require a prediction step. Because inference dominates classical encoding cost, we treat decompression throughput as matching compression throughput and report a single speed figure. Reported speeds therefore mainly reflect model inference: LLM- and TSFM-based pipelines are much slower than LZMA or Gorilla, consistent with the archival and bandwidth-bound scenarios in Figure 1.