Cramming 1568 Tokens into a Single Vector and Back Again: Exploring the Limits of Embedding Space Capacity

Yuri Kuratov[1,2] **Mikhail Arkhipov**[3] **Aydar Bulatov**[1,2] **Mikhail Burtsev**[4]

[1]AIRI, Moscow, Russia
[2]Neural Networks and Deep Learning Lab, MIPT, Dolgoprudny, Russia
[3]Independent Researcher, Amsterdam, Netherlands
[4]London Institute for Mathematical Sciences, London, UK

Correspondence: yurii.kuratov@phystech.edu

Abstract

A range of recent works addresses the problem of compression of sequence of tokens into a shorter sequence of real-valued vectors to be used as inputs instead of token embeddings or key-value cache. These approaches are focused on reduction of the amount of compute in existing language models rather than minimization of number of bits needed to store text. Despite relying on powerful models as encoders, the maximum attainable lossless compression ratio is typically not higher than x10. This fact is highly intriguing because, in theory, the maximum information capacity of large real-valued vectors is far beyond the presented rates even for 16-bit precision and a modest vector size. In this work, we explore the limits of compression by replacing the encoder with a per-sample optimization procedure. We show that vectors with compression ratios up to x1500 exist, which highlights two orders of magnitude gap between existing and practically attainable solutions. Furthermore, we empirically show that the compression limits are determined not by the length of the input but by the amount of uncertainty to be reduced, namely, the cross-entropy loss on this sequence without any conditioning. The obtained limits highlight the substantial gap between the theoretical capacity of input embeddings and their practical utilization, suggesting significant room for optimization in model design.

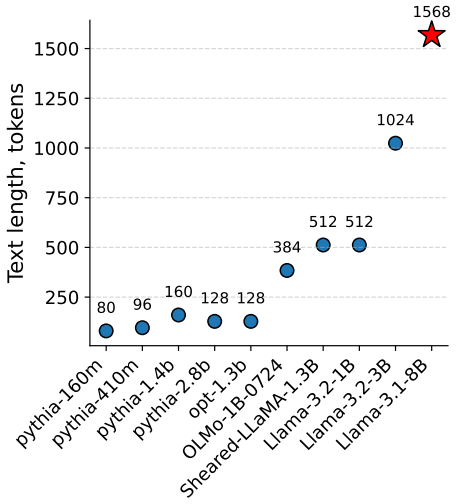

Figure 1: **How many tokens fit into a single input vector?** We estimate maximum number of tokens that can be decoded from a single input vector across various language models.

1 Introduction

Most large language models (LLMs) are built on the Transformer architecture (Vaswani et al., 2017) and have demonstrated remarkable performance as their parameters scale (Radford et al., 2019; Brown et al., 2020; Kaplan et al., 2020; Hoffmann et al., 2022). As model sizes increase, so does the dimensionality of their input embeddings. However, despite this growth, each embedding still represents only a single token, e.g., for a series of Llama models embeddings size is growing from 2,048 in 1B-parameter models to 16,384 float numbers in 405B-parameter models (Dubey et al., 2024). Remarkably, even a 2,048-dimensional vector of 16-bit floats has a theoretical capacity of 32,768 bits, which is sufficient to encode roughly 1,931 tokens from a vocabulary of size 128,256. *This observation motivates us to explore whether language models can utilize the latent capacity of input vectors more effectively, potentially encoding and processing multiple tokens with a single vector.*

Encoding multiple tokens or even entire texts into a compact latent representation has been a longstanding challenge in natural language processing. It includes approaches, such as sentence embeddings (Le and Mikolov, 2014; Kiros et al., 2015; Cer et al., 2018; Wang et al., 2024) for semantic search and retrieval, and text autoencoders (Bowman et al., 2016; Miao et al., 2016; Montero et al., 2021), aimed to capture the essential meaning of texts in a dense representations.

In the context of LLMs, the challenge of encoding prompts and long contexts is particularly important because of the quadratic computational cost of the self-attention mechanism in Transformers. Several works have explored the possibility of replacing token-based prompts with a smaller set of dense vectors (Lester et al., 2021; Li and Liang, 2021; Gao, 2024; Li et al., 2024b), thereby shortening the input sequence and reducing the computational budget. These methods have demonstrated token-to-vector lossy compression ratios on the order of x500 with 8B-parameter models, indicating that it is possible to retain the critical information in a significantly reduced number of vectors. However, lossless compression is still limited by approximately factor of 10.

In memory-augmented architectures (Weston et al., 2015; Sukhbaatar et al., 2015; Burtsev et al., 2021), these embeddings act as additional storage or as a recurrent state for passing information between time steps (Dai et al., 2019; Bulatov et al., 2022; Chevalier et al., 2023; Behrouz et al., 2024), essentially serving as an episodic memory. Moreover, recent approaches have explored the power of latent space reasoning (Hao et al., 2024) where high-capacity embeddings enable models to perform complex multi-step tasks directly in latent space. Consequently, the capacity of these vectors is crucial not only for efficient input representation, but also for increasing the overall expressiveness and computational power of models (Merrill and Sabharwal, 2023; Strobl et al., 2024; Sanford et al., 2024). Better understanding of the latent capacity of input vectors, could significantly help to improve encoding and retrieving of contextual information, episodic memory, as well as complex reasoning within large language models.

In this work, we investigate the limits of such input representations, exploring their capacity to encode and reconstruct long text sequences. By systematically quantifying how much additional information these vectors can capture, we provide insights into the efficiency and potential of latent representations in LLMs. Our main contributions:

1. We empirically study capacity limits of LLM's input representations by compressing texts into trainable `[mem]` vectors.

2. We establish a direct connection between the latent capacity of input vectors and text cross-entropy, providing a quantitative measure of the information each vector can encode.

3. We show that the capacity limits remain consistent across different text lengths and domains, including natural text and random word sequences.

4. We introduce a set of metrics that decouple the capacity of trainable input vectors from the language model's inherent prediction abilities. Using these metrics, we demonstrate a nearly linear scaling of compression capacity with the number of trainable vectors (e.g., Llama-3.2-1B compresses 7,168 tokens into just 16 vectors).

Our code is available at this URL.

2 Related Work

Approaches to compressing LLM context into a shorter sequence of input or KV-cache vectors are explored for various purposes, yet no standardized terminology or unified methodology has emerged.

Context compression. One application for input compression is connected with efficient processing of long contexts with LLMs. RMT (Bulatov et al., 2022) and AutoCompressors (Chevalier et al., 2023) train the whole language model in a recurrent manner to compress the information from input segments to summary vectors and later reuse them to solve long-context tasks. ICAE Ge et al. (2024) uses an autoencoder architecture with a frozen LLM as a decoder and adapt the same LLM for the encoder using LoRA (Hu et al., 2022). The resulting pipeline is pretrained using autoencoding and language modeling objectives, and then finetuned for language tasks, achieving the effective compression rate of x4. SelfCP (Gao, 2024) uses the base LLM itself as a compressor using a trainable adapter to aggregate compressed states across multiple segments. 500xCompressor (Li et al., 2024b) extends the autoencoding approach with layer-wise connections and additional language pretraining tasks, exploring compression ratios up to x480, though at the cost of substantial quality degradation. Alternative approaches aim to compress KV activations instead of input tokens. Some methods achieve this by estimating token relevance, either through training-free (Zhang et al., 2023; Li et al., 2024a) or training-based approaches (Qin and Van Durme, 2023; Qin et al., 2024), to prune irrelevant tokens and focus computation on the most informative ones. These strategies can yield high-quality but lossy compression with ratios up to x20. This result can be improved by finetuning models to leverage the resulting cache representations more effectively. This way, KV-Distill (Chari et al., 2025) can reduce cache size up to 100 times with

negligible loss in QA performance. In contrast, our method, applied to models of comparable size (up to 8 billion parameters), demonstrates that a compression rate of x1568 can be achieved without any loss in reconstruction quality.

Prompt compression. Another line of work targets prompts compression to reduce inference costs. Gist tokens (Mu et al., 2023) are prompt representations compressed by the LLM itself, finetuned with a special mask. Gisting allows to achieve prompt compression rate up to x26 with only minor loss in model performance. LLMLingua (Jiang et al., 2023) decouples the compression operation from the LLM and introduces a coarse-to-fine prompt compression strategy with a budget controller and token-level iterative compression, achieving up to x20 compression with negligible performance loss. Jiang et al. (2024); Pan et al. (2024) extend this framework to long contexts, improving information retention through data distillation. Additionally, Morris et al. (2023, 2024) show that the original text can be reconstructed not only from its embeddings but even from the LM's predictions. In the current work we apply a per-sample optimization process instead to explore the fundamental limits of compression and establish upper bounds on compression rates that far exceed prior work.

LLM-based lossless compression pipelines. Language models have also been investigated for lossless text compression.[1] LLMZip (Valmeekam et al., 2023) improves standard compression by ranking candidates using next-token probabilities, while FineZip (Mittu et al., 2024) accelerates compression through fine-tuning and dynamic context management for better efficiency. The capabilities of LLMs in compression pipelines can be measured in bits-per-token over a representative textual corpus. Huang et al. (2024); Guo et al. (2024) provide such measurements for public LLMs and establish the connection between compression rate and model performance, measured by diverse benchmark scores. Unlike these methods, we do not rely on external compression algorithms. Instead, we achieve lossless compression using only the LLM itself, providing both theoretical insights and practical demonstrations of compression limits.

[1] Delétang et al. (2024) explore data compression (texts to sequences of bits) using Arithmetic Coding and LLMs. This line of research is orthogonal to ours, as the sequence of bits cannot be passed as an input to an LLM. We analyze our approach from the data compression perspective in Appendix F.

Trainable tokens. Some works use the trainable input tokens in other ways. Burtsev et al. (2021) uses memory tokens as additional representation storage, Beltagy et al. (2020); Zaheer et al. (2020) use similar global tokens to enhance long-range information flow. Li and Liang (2021); Lester et al. (2021); Gao (2024); Liu et al. (2022) explore trainable soft prompts as an alternative to finetuning model weights. Our findings about the representation capacity can represent the potential efficiency limits of such methods, based on how far the model behavior can be changed using trainable tokens.

3 Method

We propose a simple approach for compressing a sequence of tokens into a small set of "memory" vectors. Then with this method we analyze how many tokens can be stored and decoded from a small set of resulting vectors. Fig. 2 provides an overview of our setup.

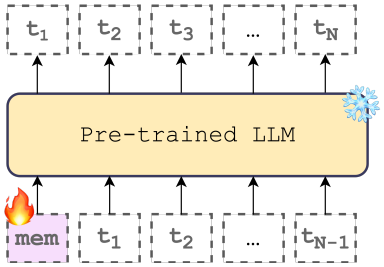

Figure 2: **Compressing text into a `[mem]` vector.** The pre-trained LLM is frozen, and we only finetune one or multiple `[mem]` vectors to decode the sequence of tokens $[t_1, t_2, \ldots, t_N]$. `[mem]` vectors are trained for each text separately.

Trainable `[mem]` vectors are inspired by Memory Transformers (Burtsev et al., 2021), but here these vectors are designed to encode an entire text sequence. The training method is similar to Prompt Tuning (Lester et al., 2021), with only a set of special input embeddings optimized while all parameters of the language model are frozen.

Formally, given a token sequence $[t_1, t_2, \ldots, t_N]$, we introduce a set of trainable vectors `[mem]` $= [m_1, \ldots, m_K]$ that are prepended to the text. These `[mem]` vectors are optimized to encode $[t_1, t_2, \ldots, t_N]$. During training, the frozen language model processes $[m_1, \ldots, m_K, t_1, t_2, \ldots, t_i]$ as the input context for predicting next token t_{i+1}. The `[mem]` vectors are optimized by minimizing the standard next-token prediction cross-entropy loss. As a result, each text sequence is associated with a unique

set of [mem] vectors. At inference time, we start generation with the learned [mem] tokens and let LM to decode the text.

Let's estimate an upper bound on the number of tokens that can be generated from a single input vector by a language model. The input vector has a dimension d_{model}, with each element represented by b bits, so that the total information content is approximately $d_{model} \times b$ bits. Given a vocabulary of size $|\mathcal{V}|$, where each token carries at most $\log_2 |\mathcal{V}|$ bits of information, the maximum number of tokens L that can be generated is bounded by:

$$L \leq \frac{d_{model} \times b}{\log_2 |\mathcal{V}|}. \quad (1)$$

Our goal is to quantify the capacity of trainable input vectors (denoted as [mem]) in terms of the amount of information they can encode and later decode. From an information-theoretic standpoint, we interpret this capacity as the ability to reduce uncertainty in the generated text. To this end, we define the following metrics.

Decoding Capacity (in Tokens): From an information-theoretic perspective, this metric represents the maximum number of tokens that can be reliably reconstructed from the compressed representation in the [mem] vectors. It is defined as the longest text sequence length L for which the token-level accuracy exceeds a predefined threshold:

$$L_{\text{max}} = \max \big\{ L \mid \text{Acc}\big(\text{LM}(t_{[1:L]} \mid \texttt{[mem]})\big) > \text{thr} \big\}, \quad (2)$$

here, *Acc* is computed via teacher-forcing when decoding text both with and without the [mem] vectors. This measure reflects the effective storage limit (in tokens) imposed by the fixed capacity of the memory vector.

Token Gain: This metric estimates the additional number of tokens that can be correctly decoded due to the presence of the [mem] vector, relative to the baseline performance of the language model (LM) without it. Formally, if $C_{\text{tokens}}^{\text{LM+[mem]}}$ is the count of tokens correctly predicted when using the memory vector and $C_{\text{tokens}}^{\text{LM}}$ is the count without it, then the gain is given by

$$\begin{aligned} C_{\text{tokens}} &= C_{\text{tokens}}^{\text{LM+[mem]}} - C_{\text{tokens}}^{\text{LM}} \\ &= \sum_{i=1}^{N} \mathbb{1}\Big(t_i = \text{LM}(t_{[1:i-1]} \mid \texttt{[mem]})\Big) \\ &\quad - \sum_{i=1}^{N} \mathbb{1}\Big(t_i = \text{LM}(t_{[1:i-1]})\Big). \end{aligned} \quad (3)$$

Viewed through an information-theoretic lens, this difference quantifies the number of tokens' worth of information (i.e., discrete units) that the memory vector adds to the decoding process.

Information Gain: Cross-entropy measures the uncertainty or the average number of bits required to encode a sequence under a given model. The *Information Gain* quantifies how much the [mem] vector reduces this uncertainty. Let $H_{\text{LM}} = H\big(P_\theta(t_{[1:N]})\big)$ be the cross-entropy (in bits) when decoding without the memory vector, and $H_{\text{LM+[mem]}} = H\big(P_\theta(t_{[1:N]} \mid \texttt{[mem]})\big)$ be the cross-entropy with the memory vector. Then, the reduction is given by

$$\text{CE-reduction} = C_{\text{H}} = H_{\text{LM}} - H_{\text{LM+[mem]}}. \quad (4)$$

This measures how many fewer bits are needed to represent the text, thus reflecting the additional information provided by the memory vector.

Collectively, these metrics enable us to characterize the capacity of the trainable input vectors both in terms of discrete tokens (C_{tokens}) and entropy (C_{H}), while L_{max} provides an upper bound on the length of text that can be accurately reconstructed. In our experiments, these measures are computed over a curated set of texts and averaged to obtain robust estimates. We note that the absolute values of *Information Gain* depend on the underlying vocabulary, and therefore should not be directly compared across models with different vocabularies.

4 Experiments and Results

We evaluate capacity of trainable input vectors of the same size as dimension of input embeddings for different language models on texts from different sources.

Models We use models from Pythia suite (160M, 410M, 1.4B, 2.8B) (Biderman et al., 2023), OPT-1.3B (Zhang et al., 2022), OLMo-1B (Groeneveld et al., 2024), Sheared-LLaMA-1.3B (Xia et al., 2024), Llama-3 models (1B, 3B, 8B) (Dubey et al., 2024), and Mamba (130M, 370M, 790M, 1.4B) (Gu and Dao, 2024). List of all used models with links to HuggingFace Hub are in Appendix A.

Data As a source of texts for compression, we use texts from the PG-19 dataset (Rae et al., 2020), which consists of books extracted from the Project Gutenberg library. Given that PG-19 is publicly available and contains books, it is highly plausible

		Pythia-160M	Pythia-410M	Pythia-1.4B	Llama-3.2-1B	Llama-3.2-3B	Llama-3.1-8B
PG-19	Max, tokens	80	96	160	512	1024	1568
	Gain, tokens	70.9$_{\pm 11.0}$	81.3$_{\pm 12.0}$	158.0$_{\pm 29.1}$	426.2$_{\pm 79.2}$	720.3$_{\pm 80.2}$	1094.1$_{\pm 127.6}$
	Information Gain	396.4$_{\pm 46.0}$	431.4$_{\pm 51.6}$	792.8$_{\pm 143.4}$	2119.9$_{\pm 364.8}$	3292.2$_{\pm 320.0}$	4865.7$_{\pm 546.6}$
Fanfics	Max, tokens	80	96	192	512	1024	1568
	Gain, tokens	70.9$_{\pm 10.5}$	81.2$_{\pm 11.6}$	152.9$_{\pm 28.0}$	449.6$_{\pm 83.7}$	734.1$_{\pm 85.0}$	1071.8$_{\pm 168.6}$
	Information Gain	378.1$_{\pm 45.9}$	429.8$_{\pm 46.2}$	776.9$_{\pm 132.5}$	2213.8$_{\pm 365.8}$	3354.5$_{\pm 344.9}$	4768.9$_{\pm 622.6}$
Random	Max, tokens	65	72	139	316	460	792
	Gain, tokens	61.3$_{\pm 6.6}$	76.9$_{\pm 8.7}$	144.4$_{\pm 17.5}$	294.9$_{\pm 64.8}$	456.9$_{\pm 72.1}$	623.2$_{\pm 97.3}$
	Information Gain	500.8$_{\pm 38.9}$	630.4$_{\pm 65.2}$	1108.2$_{\pm 136.2}$	2265.2$_{\pm 498.7}$	3382.6$_{\pm 585.2}$	4541.2$_{\pm 758.6}$

Table 1: **Compression capacity across different text sources and models.** We report *Decoding Capacity (in Tokens)* ("Max, tokens" in the Table), *Token Gain*, and *Information Gain* for texts from *PG-19*, *fanfics*, *random*. Notably, *Information Gain* remains similar across all text sources for each model (except *random* for Pythia). For *PG-19* and *fanfics*, LMs leverage their ability to predict natural language, so the *Decoding Capacity (in Tokens)* generally exceeds the *Token Gain*. Furthermore, we find no evidence that the models benefit from potentially having PG-19 in their pre-training data, as their performance on *PG-19* is not significantly better than on *fanfics* published after October 2024. In contrast, random text offers no predictable structure, making these two metrics nearly identical. This allows us to distinguish how many tokens model can predict by itself compared to decoding from trainable input vector. Larger models consistently show greater compression capacity across all metrics.

that these texts were included in the pre-training data of LLMs. Notably, PG-19 is part of the Pile dataset (Gao et al., 2020), which was used to train Pythia models.

To assess the compression of texts that models have not encountered during pre-training, we collected fanfiction texts published online after October 2024 from the AO3 fanfics library[2]. Details of this collection process are provided in Appendix B.

Both the *PG-19* and *fanfics* consist of natural language texts, where language models can predict some tokens based on prior context and model parameters. To isolate the capacity of the trainable input vectors without the influence of the knowledge of natural language by language model itself, we also employed *random* texts. *Random* texts were generated by randomly sampling words from the top 100,000 words from the GloVe vocabulary[3].

We train only a set of M vectors that are prepended to the model's input. In most of the experiments, we use only one trainable vector, if not stated otherwise.

4.1 Decoding Capacity of a Single Vector

We find that a *single* trainable vector can enable language models to produce surprisingly long, *targeted* text sequences. We estimate *Decoding Capacity (in Tokens)* (Eq. (2)) on 50 texts from PG-19 for each length. We set a token-level accuracy threshold of 0.99 and evaluate across the following length grid: [64, 80, 96, 128, 160, 192, 256, 384, 512, 768, 1024, 1280, 1568, 2048, 2560, 3072].

Fig. 1 presents the results for the evaluated models. Notably, Llama-3.1-8B can accurately reconstruct texts of up to 1568 tokens from just a *single* input vector. Interestingly, among models with around 1B parameters (Pythia-1.4B, OPT-1.3B, OLMo-1B, Sheared-LLaMA-1.3B, and Llama-3.2-1B) we observe compressive capacity that ranges from 128 to 512 tokens. Pythia-2.8b, despite its larger size, has poor compression of just 128 tokens compared to smaller 1B models.

4.2 Memorization, Natural Language Understanding and Episodic Memory

Generation from the `[mem]` vector involves combining information from both the pre-trained language model parameters and memory about text specific sequence. To analyze contributions of these different types of memory, we use *Token Gain* (Eq. (3)) which measure the extra number of tokens predicted correctly, and *Information Gain* (Eq. (4)) showing the decrease in cross-entropy when decoding from memory vector. In contrast to *Decoding Capacity*, these two metrics more directly isolate the capacity contributed by the `[mem]` vector itself.

In addition to texts from *PG-19* that may have been seen by LMs during pre-training, we consider: (1) texts from *fanfics* to factor out memorization as they were published after release of the models, and

[2] https://archiveofourown.org/

[3] https://nlp.stanford.edu/data/glove.6B.zip

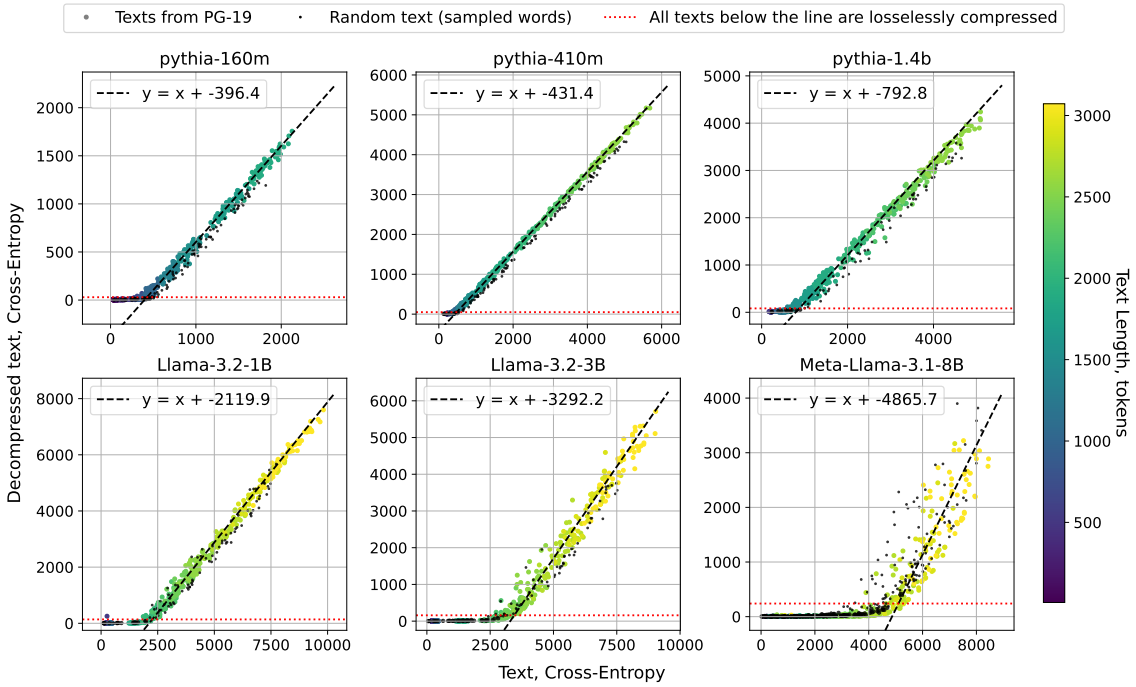

Figure 3: **Information gain of text compression to `[mem]` vector doesn't depend on language understanding capabilities of models.** Compression results for various language models show the relationship between the cross-entropy (CE) of the original and decompressed texts. If the text CE falls below a model-specific threshold (red line), the text is losslessly compressed. This value is a input vector capacity in terms of entropy (*Information Gain*, C_H). For texts that are not perfectly compressed, the compression process reduces their CE to a consistent, model-specific value (bias of the black dashed line). Larger models (e.g., Llama-3.1-8B) can handle longer texts before reaching the compression threshold, due to their greater capacity compared to smaller models (e.g., Pythia-160M). This behavior holds for both natural texts (*PG-19*) and unnatural *random* texts consisting of random word sequences.

(2) *random* sequenses of words to exclude learned natural language understanding capabilities.

Decoding Capacity (in Tokens) for texts from *PG-19* and *fanfics* was evaluated on the following length grid: [64, 80, 96, 128, 160, 192, 256, 384, 512, 768, 1024, 1280, 1568, 2048, 2560, 3072].

Table 1 summarizes the results for each model and text source. We have two main observations. The metrics for both *PG-19* and *fanfics* are remarkably similar across all models tested. This similarity implies that the presence of *PG-19* in the pre-training data does not provide much of an advantage. Thus, compression performance does not appear to be driven by direct memorization of the dataset. Notably, even for *random* texts, larger models such as Llama-3.1-8B still exhibit substantial compressing power, reliably reconstructing sequences of up to 792 tokens. This result demonstrates the impressive capacity of learnable input embeddings to control LLM generation. In particular, a single learned vector is sufficient to guide generation of nearly 800 random tokens.

A key takeaway from these results is that the model's compression ability does not depend on familiarity with specific texts or knowledge of natural language gained during pre-training. Instead, the single trainable vector itself provides language agnostic substantial capacity, allowing to store completely novel texts or random sequences of words.

4.3 Sensitivity of Compression to Text Complexity

Decoding capacity might depend on the complexity of the input text for a language model. In this section, we study how compression changes uncertainty of the model about the text.

For 50 text samples from the *PG-19* at each target length (ranging from 8 up to 1568 tokens, and to 3072 for larger models) we measured cross-entropy both before (H_{LM}) and after ($H_{\text{LM+[mem]}}$) compression (see Eq. (4)). Figure 3 compares results across Pythia and LLama models, and full results for all models are provided in Appendix D.

In Fig. 3, the models demonstrate linear relationship between cross-entropy before and after compression for not perfectly compressible texts

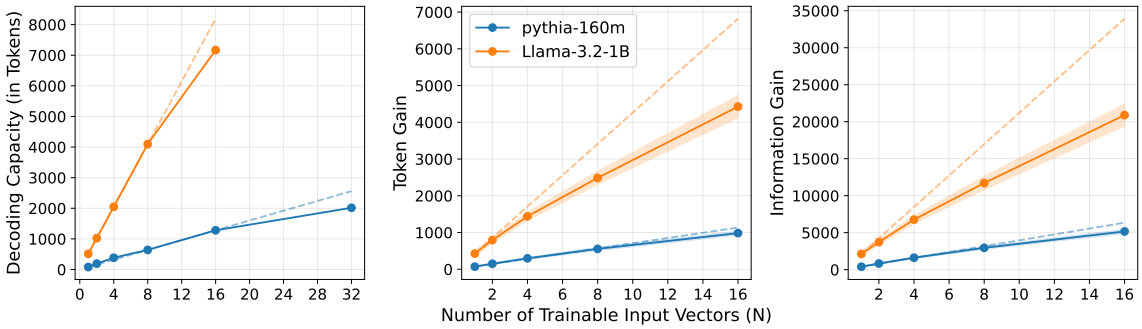

Figure 4: **Compression scales linearly with the number of trainable `[mem]` vectors.** Dashed lines represent ideal linear scaling, and shaded regions indicate ± 1 std. Pythia-160m reaches its maximum input context length of 2048 tokens and can successfully encode texts of up to 2016 tokens into 32 `[mem]` input vectors. Llama-3.2-1B can perfectly decode texts of 7168 tokens from just 16 input vectors.

(i.e., lying above the red dotted line), indicating constant value of information gain (or, reduction in cross-entropy). Texts with cross-entropy lower than a model's information gain are perfectly reconstructed.

To verify that this also holds for arbitrary texts, we used *random* word sequences and observed a similar pattern: as long as cross-entropy of a sample remains below the model-specific cutoff, it can be perfectly reconstructed. Notably, these random texts (black dots in Fig. 3) lie very close to the same linear trend as the *PG-19* texts, showing that similar compression laws apply regardless of the nature of the sequence. Thus, `[mem]` works as an episodic memory storing sequence specific information independent of natural language knowledge the model has.

4.4 Scaling Compression with More Trainable Vectors

To explore how compression scales with the number of input vectors `[mem]` $= [m_1, \ldots, m_K]$ we use the same training process as before but for different numbers of trainable vectors, from 1 to 16 for the Llama-3.2-1B model and from 1 to 32 for Pythia-160M.

The results of this series of experiments are presented in Fig. 4, demonstrating that input vector capacity scales almost linearly with the number of trainable `[mem]` vectors. This trend holds consistently across all measures of capacity, whether expressed in terms of tokens or text entropy. In particular, Pythia-160M successfully decodes texts up to 2016 tokens in length using 32 `[mem]` vectors, effectively reaching its maximum context length. Similarly, LlaMA-3.2-1B achieves perfect reconstruction for sequences as long as 7168 tokens with just 16 input vectors. However, scaling behavior for LlaMA-3.2-1B deviates from the linear trend, suggesting potential inefficiencies in the compression process or inherent model limitations in exploiting an increasing number of input vectors for information storage and extraction.

Extrapolating from these trends, we estimate that an entire text such as "The Hobbit, or There and Back Again" (approximately 120,000 tokens) could be compressed into only 128 input vectors using Llama-3.1-8B and into 256 vectors using Llama-3.2-1B.

These results demonstrate that increasing the number of trainable `[mem]` vectors significantly enhances compression capacity, with linear scaling observed across the evaluated models. Notably, using a small number of additional vectors introduces minimal computational overhead while enabling the reconstruction of substantially longer texts.

4.5 Embedding Capacity Utilization

To measure how effectively each model uses its input embedding space, we compare the empirically measured capacity in tokens (*Token Gain*) to a theoretical maximum derived from embedding size and vocabulary size (see Eq. (1)). We define *capacity utilization* as the ratio of these two quantities.

In Fig. 5 (top), when comparing all models with roughly 1B parameters, there are two groups: (1) older models (e.g., OPT and Pythia) show lower capacity utilization, whereas (2) newer models (e.g., Llama, ShearedLlama, Mamba, and OLMo) demonstrate higher utilization despite having the same theoretical capacity. This disparity indicates that the quality of pre-training (data, compute budget, improvements in architecture) influences the extent to which a model can exploit its input vectors

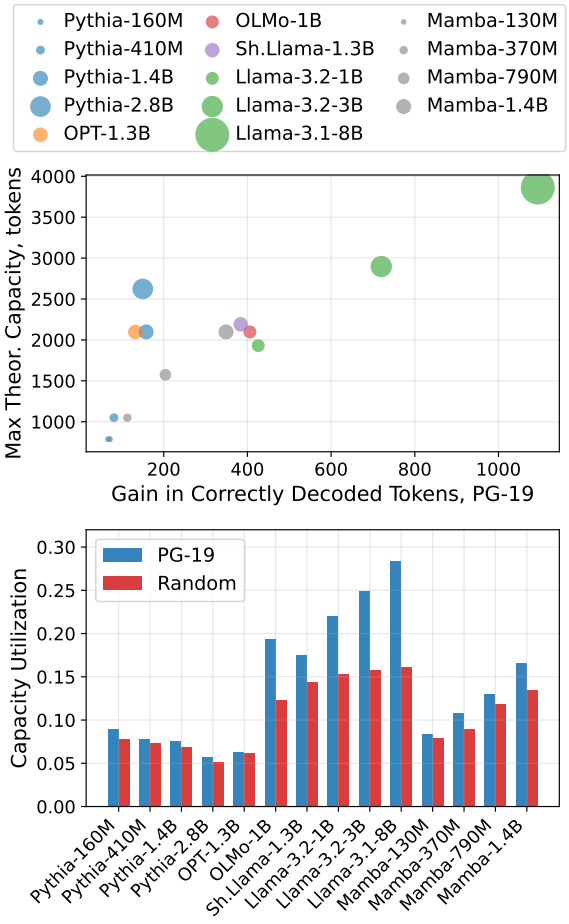

Figure 5: **Only fraction of learned input embedding information capacity can be utilized.** *Top.* Maximum token capacity (see Eq. (1)) against gain in correctly decoded tokens shows differences in utilization of learned memory embedding for studied models. *Bottom.* Capacity utilization for natural and random texts.

capacity.

In Fig. 5 (bottom) we can see three groups: (1) and (2) same as on top plot with older and newer models, and (3) group of Mamba models. The Pythia models show an interesting trend: as model size increases, capacity utilization decreases. This pattern suggests that the larger Pythia models may be under-trained relative to their theoretical potential. In contrast, Llama and OLMo models show higher capacity utilization. Based on these observations, we hypothesize that capacity utilization could serve as an indicator of the pre-training status and guide further training.

For models within the Llama family, we observe that empirical capacity utilization rises steadily with model size (1B, 3B, 8B), most noticeable on natural texts from *PG-19*. This might be fully attributed to better language understanding, gained by the larger models during pre-training, but even on *random* texts we observe a modest upward trend. This result suggests that the overall number of parameters plays an important role in determining effective capacity not only via LM capabilities but also due to better utilization of embedding space for episodic information storage.

4.6 Non-Transformer Models: Mamba

Notably, the findings from the previous sections are not limited to Transformer-based language models. State-space Mamba models can also encode an entire text into a single `[mem]` vector and perfectly reconstruct the text when its cross-entropy falls below the model's capacity threshold, e.g., Mamba-1.4B can encode texts of 512 tokens into a single `[mem]` vector, similar to other 1B models (Llama-3.2-1B, Sheared-LLaMa-1.3B). This demonstrates that obtained results are architecture-agnostic.

On capacity utilization (Fig. 5), the Mamba models exhibit a consistent pattern across both panels that depends on model size. In the top plot, the increase in correctly decoded tokens is nearly linear with respect to each model's theoretical capacity. This implies that each additional parameter in the embedding space translates into greater decoding accuracy at a nearly constant rate. In the bottom plot, capacity utilization rises monotonically from the 130M to the 1.4B model. This is the opposite of the downward trend seen in Pythia, but aligns with Llama, and shows that larger Mambas progressively make better use of their learned input space. We give more details on Mamba results in Appendix C, Fig. 6, and Table 3).

5 Discussion and Conclusions

In this work, we introduced a simple yet effective way to compress entire text sequences into a small set of trainable `[mem]` vectors without any information loss. We used this method to analyze how far we can push the latent capacity of large language models compared to its theoretical limits.

By systematically evaluating different models, we find that a surprising amount of text can be compressed to a single token, and this capacity scales linearly with the number of tokens. This highlights significant potential in practical compression pipelines and long-context processing. We demonstrate that our compression outperforms neural models as a compression method, suggesting a more efficient approach to representing infor-

mation. However, significantly more compute is needed due to optimization nature of the proposed method.

We establish a direct link between representation capacity and cross-entropy, showing that it remains independent of text length, domain, or familiarity. However, the exact model characteristics that determine capacity remain an open question. The hidden state dimension and model size play an important role along with general performance, however further analysis is required to determine the exact scaling laws for capacity.

Compression ability serves as a strong indicator of an LLM's potential. Since transformers operate entirely within their representation space, its capacity fundamentally constrains reasoning, intermediate computations, and large-scale information processing. All textual and soft prompts ultimately reside in this space, meaning its limits define how effectively models can be steered and conditioned. By mapping these boundaries, we gain deeper insight into the fundamental constraints of current architectures and the possibilities for more powerful future models.

Moreover, our findings hold significant promise for memory-augmented architectures. The ability to compress long sequences into a compact set of memory vectors shows the way for integrating efficient external memory modules that can store and retrieve detailed episodic information, potentially enhancing reasoning, long-term dependency handling, and overall model performance. We believe that incorporating such optimized memory representations could lead to novel architectures that are both computationally efficient and more capable of complex information processing.

We believe our findings present an important stepping stone to understanding the limits of modern LLMs and building more powerful models in the future.

Limitations

While our experiments push the boundaries of compression with LLMs and offer insights into their upper capacity limits, the nature of the obtained representations remains largely unclear. We have analyzed the structure of the space of trained `[mem]` vectors in Appendix E, but more in-depth analysis is needed to determine the semantic properties of the vectors and their potential value in downstream tasks. Our findings are limited to Transformer-based and Mamba models with up to 8 billion parameters due to computational constraints. Investigating the representation space of larger models, as well as exploring alternative architectures such as recurrent and memory-augmented models, remains an important avenue for future research. In our study with different text sources, we generate random text by sampling words from a dictionary. While this approach simplifies the analysis, it may slightly overestimate model capacity compared to sampling directly from a tokenizer's vocabulary, as dictionary words can be split on multiple tokens by model.

Broader Impact

We train a set of `[mem]` vectors so that arbitrary texts can be accurately reconstructed from them. This process not only allows us to analyze the capacity of these vectors, but also demonstrates that any kind of text can be compressed into compact latent representations and later decoded. Such a capability may have far-reaching implications: it could lead to more efficient methods for storing and transmitting text, while also raising important considerations regarding the potential misuse of compressed information and issues related to data security, intellectual property, and altering the behavior of aligned models.

Acknowledgments

We are thankful to SberDevices for granting us access to additional computational resources. This work was partially supported by the Ministry of Economic Development of the Russian Federation (code 25-139-66879-1-0003).

References

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. 2024. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. 2022. Recurrent memory transformer. In *Advances in Neural Information Processing Systems*, volume 35, pages 11079–11091. Curran Associates, Inc.

Mikhail S. Burtsev, Yuri Kuratov, Anton Peganov, and Grigory V. Sapunov. 2021. Memory transformer. *Preprint*, arXiv:2006.11527.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder for english. In *Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations*, pages 169–174.

Vivek Chari, Guanghui Qin, and Benjamin Van Durme. 2025. Kv-distill: Nearly lossless learnable context compression for llms. *arXiv preprint arXiv:2503.10337*.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3829–3846, Singapore. Association for Computational Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

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. 2024. Language modeling is compression. In *ICLR*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Jun Gao. 2024. Selfcp: Compressing long prompt to 1/12 using the frozen large language model itself. *arXiv preprint arXiv:2405.17052*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Tao Ge, Hu Jing, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2024. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations*.

Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, William Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah Smith, and Hannaneh Hajishirzi. 2024. OLMo: Accelerating the science of language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15789–15809, Bangkok, Thailand. Association for Computational Linguistics.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*.

Albert Gu, Karan Goel, and Christopher Ré. 2022. Efficiently modeling long sequences with structured state spaces. In *The International Conference on Learning Representations (ICLR)*.

Peijia Guo, Ziguang Li, Haibo Hu, Chao Huang, Ming Li, and Rui Zhang. 2024. Ranking llms by compression. *arXiv preprint arXiv:2406.14171*.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *Preprint*, arXiv:2412.06769.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric

Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. 2022. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, volume 35, pages 30016–30030. Curran Associates, Inc.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1419–1436, Online. Association for Computational Linguistics.

Yuzhen Huang, Jinghan Zhang, Zifei Shan, and Junxian He. 2024. Compression represents intelligence linearly. In *First Conference on Language Modeling*.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376, Singapore. Association for Computational Linguistics.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1658–1677, Bangkok, Thailand. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1188–1196, Bejing, China. PMLR.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024a. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970.

Zongqian Li, Yixuan Su, and Nigel Collier. 2024b. 500xcompressor: Generalized prompt compression for large language models. *arXiv preprint arXiv:2408.03094*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

William Merrill and Ashish Sabharwal. 2023. The parallelism tradeoff: Limitations of log-precision transformers. *Transactions of the Association for Computational Linguistics*, 11:531–545.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1727–1736, New York, New York, USA. PMLR.

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

Ivan Montero, Nikolaos Pappas, and Noah A. Smith. 2021. Sentence bottleneck autoencoders from transformer language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1822–1831, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

John Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander Rush. 2023. Text embeddings reveal (almost) as much as text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12448–12460, Singapore. Association for Computational Linguistics.

John Xavier Morris, Wenting Zhao, Justin T Chiu, Vitaly Shmatikov, and Alexander M Rush. 2024. Language model inversion. In *The Twelfth International Conference on Learning Representations*.

Jesse Mu, Xiang Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36:19327–19352.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 963–981, Bangkok, Thailand. Association for Computational Linguistics.

Guanghui Qin, Corby Rosset, Ethan Chau, Nikhil Rao, and Benjamin Van Durme. 2024. Dodo: Dynamic contextual compression for decoder-only lms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9961–9975.

Guanghui Qin and Benjamin Van Durme. 2023. Nugget: Neural agglomerative embeddings of text. In *International Conference on Machine Learning*, pages 28337–28350. PMLR.

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

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*.

Clayton Sanford, Daniel J Hsu, and Matus Telgarsky. 2024. Representational strengths and limitations of transformers. *Advances in Neural Information Processing Systems*, 36.

Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. 2024. What formal languages can transformers express? a survey. *Transactions of the Association for Computational Linguistics*, 12:543–561.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in neural information processing systems*, pages 5998–6008.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2024. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *The Twelfth International Conference on Learning Representations*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *Preprint*, arXiv:2205.01068.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

A Models and Training Details

We provide list of all models that we used in our experiments in Table 2.

Trainable vectors are initialized randomly. We use the AdamW optimizer (Loshchilov and Hutter, 2019) with a learning rate of 0.01, β_1, and β_2 both set to 0.9, and a weight decay of 0.01. Training proceeds for a maximum of 5,000 steps, with early stopping if the text is compressed losslessly, i.e., achieving a token-level accuracy of 1.0. All models are loaded from the HuggingFace Transformers library with the PyTorch framework.

Each compression experiment was run on a single A100 80GB GPU. The time required to compress text using 5,000 optimization steps ranged from a dozen of seconds for small models and short contexts to 10–20 minutes for larger models and longer contexts. We used up to 4 GPUs to run several experiments in parallel.

B Collecting Texts from the Fanfics Library

We used the AO3 fanfiction library `https://archiveofourown.org/` as a source of texts that were not present in the language models' pretraining data. To ensure novelty, we manually downloaded 21 fanfics from various fandoms (including Harry Potter, Star Wars, Transformers, Lord of the Rings, and others) that contained more than 20,000 words and were published after October 2024.

We preprocessed the HTML pages to extract only the main text content, removing any irrelevant elements. We then sampled passages from these texts to evaluate the capacity of trainable input vectors. Throughout our experiments, we refer to this dataset as *fanfics*.

From each of the *PG-19* and *fanfics*, we sampled texts and set their lengths to match the desired token counts. We ensured that each text began with complete sentences to maintain coherence. As a result, to estimate the capacity of the input vectors, we used 50 texts for each length.

C Non-Transformer Language Models: Mamba

Mamba (Gu and Dao, 2024) is a state space language model (SSM) (Gu et al., 2022), in contrast to attention-based Transformers. We applied the same compression procedure used for the Transformer models and evaluated the official Mamba checkpoints from the Hugging Face Hub (Table 2). We found that Mamba can also encode texts into a single `[mem]` vector and successfully reconstruct them whenever the text's cross-entropy falls below model's capacity threshold, indicating the effect is not unique to only Transformer models (Fig. 6 and Table 3).

D Results of Evaluating Text Compression for All Models

Here we provide results for all evaluated models in Fig. 6 and Table 3. Results are discussed in Section 4.3.

E Understanding the Structure of Compressed Vectors

To better understand the structure of the space formed by the learned embeddings, we collect a dataset of embeddings for 64-token sequences from the GovReport dataset (Huang et al., 2021). The optimization is performed until a reconstruction accuracy of 1.0 is achieved. Additionally, for each sequence, we compute multiple embeddings using different random initializations of the `[mem]` vectors.

First, we observe that the optimization process can yield different solutions; the resulting vectors for the same text may lie in completely different parts of the space. To visualize this phenomenon, we plot histograms of cosine similarity between embeddings of the same text (intra-sample) and between embeddings of different texts (inter-sample) in Fig. 7. Notably, almost no high cosine similarities (above 0.8) are observed in the intra-sample case. Moreover, the intra-sample similarities significantly overlap with the inter-sample ones, implying that the embeddings are considerably scattered throughout the space.

Although the embeddings appear to be spread out, one might hope they form a basin in which all linear interpolations between vectors would yield perfect reconstruction. To test this, we computed the reconstruction accuracy along linear interpolation trajectories between embeddings of the same sequence. However, in all cases we examined, errors were present along the interpolation trajectory (see Fig. 8). Thus, the embeddings obtained by the proposed procedure do not form a continuous basin.

These observations have several implications:

Model Name	Link to HuggingFace	Params (B)	Input Hidden Size	Vocabulary Size
Pythia-160M	EleutherAI/pythia-160m	0.16	768	50304
Pythia-410M	EleutherAI/pythia-410m	0.41	1024	50304
Pythia-1.4B	EleutherAI/pythia-1.4b	1.4	2048	50304
Pythia-2.8B	EleutherAI/pythia-2.8b	2.8	2560	50304
OPT-1.3B	facebook/opt-1.3b	1.3	2048	50272
OLMo-1B	allenai/OLMo-1B-0724-hf	1.0	2048	50304
Sheared-LLaMA-1.3B	princeton-nlp/Sheared-LLaMA-1.3B	1.3	2048	32000
Llama-3.2-1B	meta-llama/Llama-3.2-1B	1.0	2048	128256
Llama-3.2-3B	meta-llama/Llama-3.2-3B	3.0	3072	128256
Llama-3.1-8B	meta-llama/Llama-3.1-8B	8.0	4096	128256
Mamba-130M	state-spaces/mamba-130m-hf	0.13	768	50280
Mamba-370M	state-spaces/mamba-370m-hf	0.38	1024	50280
Mamba-790M	state-spaces/mamba-790m-hf	0.79	1536	50280
Mamba-1.4B	state-spaces/mamba-1.4b-hf	1.4	2048	50280

Table 2: List of used language models and their parameters.

Model	PG-19			Fanfics			Random		
	Max, tok.	Gain, tok.	Information Gain	Max, tok.	Gain, tok.	Information Gain	Max, tok.	Gain, tok.	Information Gain
Pythia-160M	80	$70.9_{\pm 11.0}$	$396.4_{\pm 46.0}$	80	$70.9_{\pm 10.5}$	$378.1_{\pm 45.9}$	65	$61.3_{\pm 6.6}$	$500.8_{\pm 38.9}$
Pythia-410M	96	$81.3_{\pm 12.0}$	$431.4_{\pm 51.6}$	96	$81.2_{\pm 11.6}$	$429.8_{\pm 46.2}$	72	$76.9_{\pm 8.7}$	$630.4_{\pm 65.2}$
Pythia-1.4B	160	$158.0_{\pm 29.1}$	$792.8_{\pm 143.4}$	192	$152.9_{\pm 28.0}$	$776.9_{\pm 132.5}$	139	$144.4_{\pm 17.5}$	$1108.2_{\pm 136.2}$
Pythia-2.8B	128	$150.1_{\pm 50.7}$	$740.3_{\pm 234.9}$	-	-	-	141	$134.5_{\pm 24.7}$	$1026.3_{\pm 211.4}$
OPT-1.3B	128	$132.2_{\pm 23.8}$	$712.8_{\pm 143.3}$	-	-	-	116	$129.3_{\pm 16.4}$	$1068.0_{\pm 181.3}$
OLMo-1B	384	$406.3_{\pm 61.7}$	$1901.0_{\pm 254.5}$	-	-	-	249	$257.3_{\pm 55.0}$	$1852.2_{\pm 395.0}$
Sh.LLaMa-1.3B	512	$383.6_{\pm 38.4}$	$1835.1_{\pm 162.9}$	-	-	-	382	$315.1_{\pm 35.1}$	$1893.0_{\pm 210.0}$
Llama-3.2-1B	512	$426.2_{\pm 79.2}$	$2119.9_{\pm 364.8}$	512	$449.6_{\pm 83.7}$	$2213.8_{\pm 365.8}$	316	$294.9_{\pm 64.8}$	$2265.2_{\pm 498.7}$
Llama-3.2-3B	1024	$720.3_{\pm 80.2}$	$3292.2_{\pm 320.0}$	1024	$734.1_{\pm 85.0}$	$3354.5_{\pm 344.9}$	460	$456.9_{\pm 72.1}$	$3382.6_{\pm 585.2}$
Llama-3.1-8B	1568	$1094.1_{\pm 127.6}$	$4865.7_{\pm 546.6}$	1568	$1071.8_{\pm 168.6}$	$4768.9_{\pm 622.6}$	792	$623.2_{\pm 97.3}$	$4541.2_{\pm 758.6}$
Mamba-130M	32	$65.5_{\pm 18.1}$	$371.4_{\pm 94.6}$	-	-	-	84	$62.6_{\pm 17.1}$	$535.7_{\pm 121.8}$
Mamba-370M	128	$113.0_{\pm 27.6}$	$585.3_{\pm 131.1}$	-	-	-	106	$94.5_{\pm 25.5}$	$743.2_{\pm 186.9}$
Mamba-790M	256	$204.2_{\pm 33.0}$	$1011.6_{\pm 138.8}$	-	-	-	198	$187.0_{\pm 26.3}$	$1421.3_{\pm 179.1}$
Mamba-1.4B	512	$348.9_{\pm 42.9}$	$1599.5_{\pm 164.5}$	-	-	-	288	$282.7_{\pm 36.5}$	$2062.3_{\pm 257.3}$

Table 3: **Compression capacity across different text sources and for all evaluated models.** We report *Decoding Capacity (in Tokens)* ("Max, tokens" in the Table), *Token Gain*, and *Information Gain* for texts from *PG-19*, *fanfics*, *random*.

1. A lossless compression algorithm ideally assigns a unique decoding to each vector; multiple valid embeddings for the same object limit the achievable compression rate.

2. The spread and entangled structure of the embeddings may render them less useful as representations.

3. This non-unique, scattered structure could make it more challenging to extract important information when these compressed representations are used as context in an LM.

F Data Compression Analysis

In the present work, we focus on exploring phenomena that could be potentially helpful in building more efficient LLMs. Despite our method can serve as a general-purpose compression approach, we currently treat lossless compression as a tool to better understand LLMs hidden state capacity in contrast to achieving the highest possible compression rate.

We ran experiments on three datasets (PG-19, fanfics, and randomly sampled sequences of words) using zlib, bz2, lzma, pure Huffman coding, and Arithmetic Coding(AC) with LM. We report their mean compression ratios (original text size in bits / compressed size) in Table 4.

We see that conventional entropy coders can compress texts by about a factor of two, and combining Arithmetic Coding with pythia-160m can lead to much higher ratios: 6-7x. With our approach we can encode up to 1568 tokens into a single 4096-dimensional bfloat16 vector (using

Method	PG19	Fanfics	Random
zlib	2.28 ± 0.16	2.34 ± 0.07	1.80 ± 0.06
bz2	2.46 ± 0.16	2.56 ± 0.09	1.94 ± 0.11
lzma	2.28 ± 0.16	2.33 ± 0.07	1.86 ± 0.13
Huffman	1.81 ± 0.07	1.86 ± 0.04	1.77 ± 0.01
AC, pythia-160m	6.77 ± 0.75	6.73 ± 0.41	2.83 ± 0.08

Table 4: Compression ratios (in bits) comparison for classic compression algorithms and arithmetic coding (AC) using pythia-160m for a range of corpora.

LLaMA-3.1-8B) and reconstruct them losslessly. If we treat a single 4096-dimensional vector as a "compressed file" and compute (original text size as string) / (size of vector), the ratio actually appears to be about 0.8x, so it does not take less bits on disk than a raw text.

Importantly, unlike conventional compression algorithms that output arbitrary bitstreams, our approach must encode the text into a single vector, which LLMs interpret as an embedding. This constraint is stricter than simply minimizing file size, since the output must lie within the input space of the model. Our focus, therefore, is on analyzing representational capacity within the input space of large language models. For instance, a single 4096-dimensional vector in LLaMA-3.1-8B can guide the model to generate up to 1568 tokens exactly. Reconstructing 1568 tokens from just one vector results in a 1568x reduction in the number of embeddings that would otherwise be used to represent the text.

Our goal is not to outperform standard compressors in bits-per-byte efficiency, but rather to show that LLMs can store significant amounts of text with only a single embedding.

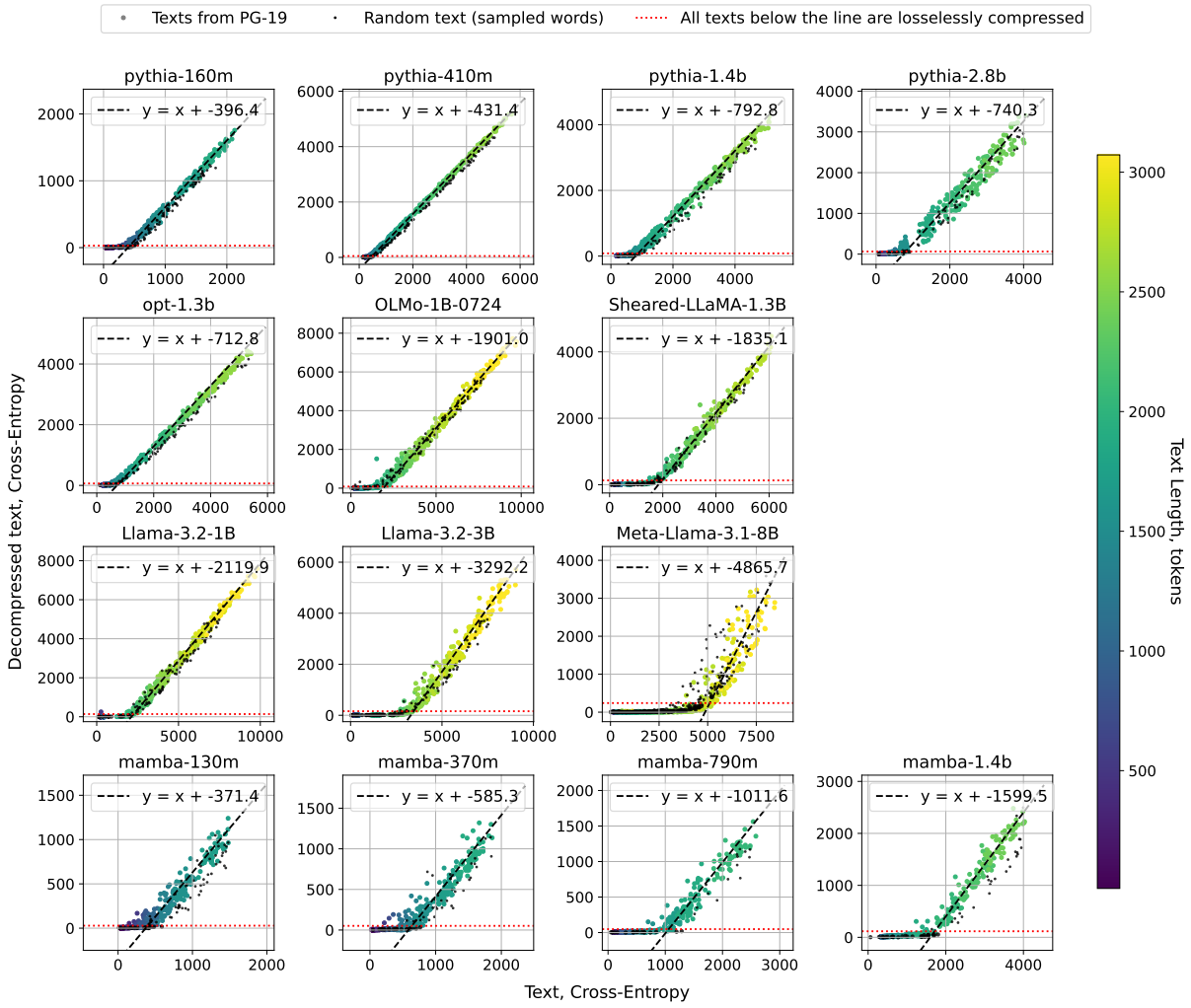

Figure 6: **Information gain of text compression to `[mem]` vector doesn't depend on language understanding capabilities of models.** Compression results for various language models show the relationship between the cross-entropy (CE) of the original and decompressed texts. If the text CE falls below a model-specific threshold (red line), the text is losslessly compressed. This value is a input vector capacity in terms of entropy (*Information Gain*, C_H). For texts that are not perfectly compressed, the compression process reduces their CE to a consistent, model-specific value (bias of the black dashed line). Larger models (e.g., Llama-3.1-8B) can handle longer texts before reaching the compression threshold, due to their greater capacity compared to smaller models (e.g., Pythia-160M). This behavior holds for both natural texts (*PG-19*) and unnatural *random* texts consisting of random word sequences. The result is not limited to transformer-based architectures: non-transformer models, such as *Mamba*, exhibit the same pattern.

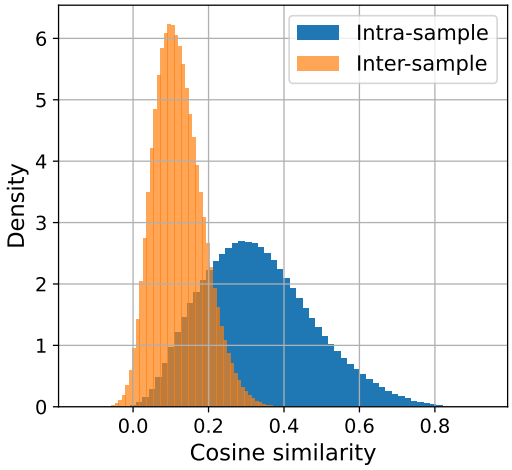

Figure 7: **Intra/inter-sample embeddings cosine similarity.** Empirical probability densities of cosine similarity between intra-sample and inter-sample embeddings. Intra-sample similarities are measured between of the same sequence of tokens, while inter-sample between different ones. Measured on GovReport (Huang et al., 2021) and Sheared-Llama-1.3B (Xia et al., 2024).

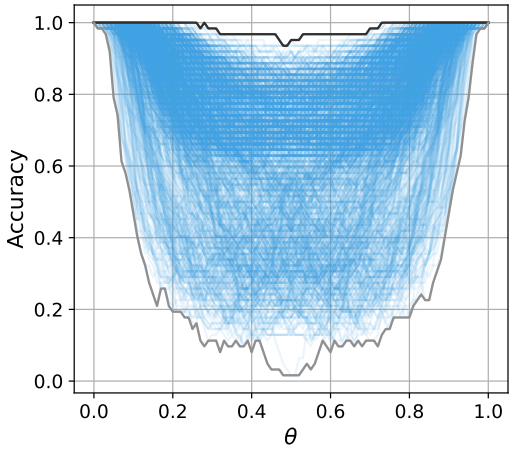

Figure 8: **Intra-sample Interpolation Accuracies.** Interpolation lines are provided for all pairs between 32 embeddings of the same input sequence. All interpolation lines are printed with high transparency to show denser regions. Grey lines depict minimums and maximums of the accuracy for a given interpolation parameter θ.