38. LLM context compression

Anastasia Yanke, Taisia Pevnaya and Daniil Muravetsky

AIRI Summer School 2025

Abstract

Efficient compression of textual sequences into compact vector representations is critical for processing long contexts in large language models (LLMs). We investigate three approaches: stabilizing memory vectors with dense layers, replacing them with Low-Rank Adaptation (LoRA), and using autoencoders (AE/VAE) for universal encoding. Our methods are evaluated on reconstruction accuracy, gain in correctly decoded tokens, and capacity utilization using the Pythia-160M and Pythia-410M models on the PG-19 dataset. Dense layer stabilization with Pythia-410M achieves a maximum theoretical capacity of 192 tokens and a 50 % capacity utilization improvement over the baseline, while LoRA methods are limited to 16 tokens. AE/VAE methods reach up to 0.27 accuracy but struggle with longer sequences. Our findings highlight trade-offs between efficiency, capacity, and robustness, paving the way for scalable LLM context compression. Code is available at `https://github.com/DMurawiecki/hidden_capacity`.

1 Introduction

Large language models (LLMs) based on the Transformer architecture [Vaswani et al., 2017] excel in natural language tasks but face challenges in processing long contexts due to quadratic computational complexity. Recent work by Kuratov et al. [2025] demonstrated that a single 4096-dimensional vector can encode up to 1568 tokens with lossless reconstruction, revealing a significant gap between theoretical and practical embedding capacities. However, their per-example optimization of memory vectors ([mem]) is computationally intensive, limiting scalability.

In this work, we explore three alternative approaches to compress text into compact representations while maintaining high reconstruction accuracy, using the smaller Pythia-160m model ($d_{\text{model}} = 768$) for evaluation:

1. **Dense Layer Stabilization**: Enhancing [mem] vectors with a trainable MLP module to improve expressiveness and training stability.

2. **LoRA Adaptation**: Replacing [mem] vectors with low-rank adapters integrated into the LLM's attention layers, reducing trainable parameters.

3. **Autoencoders (AE/VAE)**: Using Pyhia-based or variational autoencoders to generate universal latent vectors for text compression.

We evaluate the performance of several language model-based text compression methods against the baseline [mem] approach, utilizing metrics proposed by Kuratov et al. [2025]: Decoding Capacity, Token Gain, and Information Gain. Additionally, we analyze parameter efficiency, training stability, and error patterns across the PG-19 dataset, a collection of natural texts. Our findings demonstrate that adding dense layer to before [mem] vector, namely pythia-160m-dense and pythia-410m-dense, significantly outperform the baseline in key metrics, while autoencoder (AE) and variational autoencoder (VAE) models offer a trade-off with improved efficiency but limited capacity for longer texts. Low-Rank Adaptation (LoRA) variants, such as pythia-160m-lora-qvk, exhibit constrained performance, particularly for extended sequences.

1. We intoduce and compare three compression methods: ... 2. We show that adding single dense layer significantly increases capacity of an input vectors. pythia-160m: 80-¿128, pythia-410m: 96-¿192. 3. We tested LorA with rank 1 as replacement for mem-vectors, and found low compression with 16 tokens at max. 4. We found that AE/VAE is hard to train with frozen LLM-decoder, but we consider it as promising direction of future

2 Related Work

2.1 Context Compression

Context compression in LLMs, such as RMT [Bulatov et al., 2022] and AutoCompressors [Chevalier et al., 2023], reduces the sequence length by summarizing inputs into dense vectors. Kuratov et al. [2025] achieved $\times 1568$ compression using per-example optimized [mem] vectors per sample with a larger model, but at a high computational cost. Prompt compression methods such as Gist tokens [Mu et al., 2023] and LLMLingua [Jiang et al., 2023] achieve up to $\times 26$ ratios but are lossy.

2.2 LoRA and Prefix Tuning

LoRA [Hu et al., 2022] adapts frozen LLMs by injecting low-rank matrices into attention layers, reducing trainable parameters by orders of magnitude. Prefix tuning [Li and Liang, 2021] optimizes continuous input vectors for guide generation, stabilizing training compared to direct optimization.

2.3 Autoencoders

Autoencoders (AE) and variational autoencoders (VAE) [Kingma and Welling, 2014] compress the data into latent representations. In NLP, Pythia-based autoencoders [Cer et al., 2018] encode sentences into dense vectors, while VAEs ensure smooth latent spaces via KL-divergence regularization.

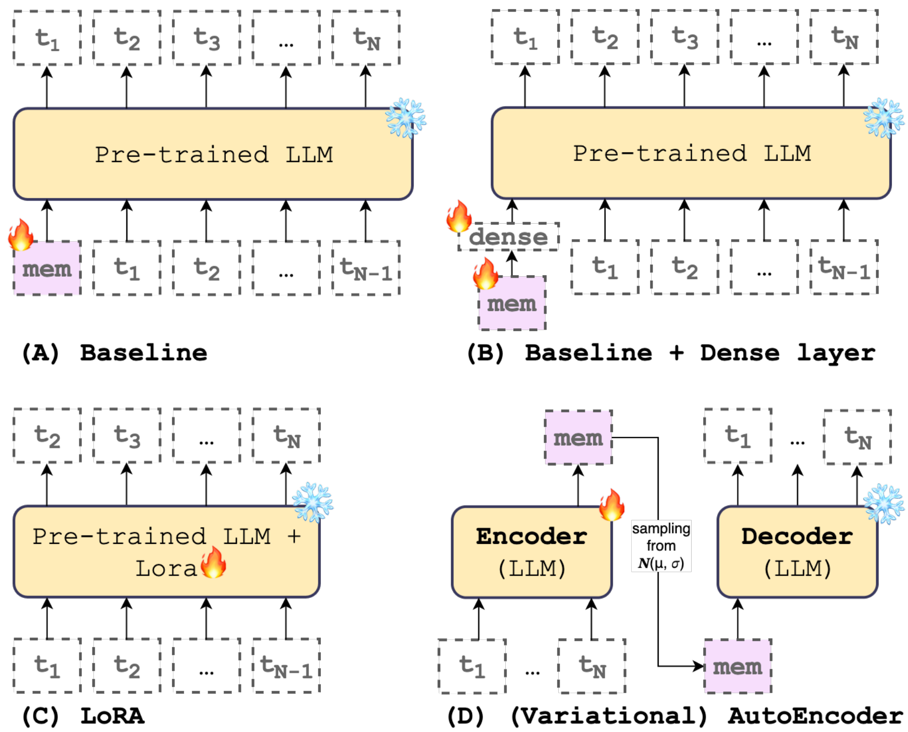

Figure 1: Main approaches: (A) Baseline with mem-vectors, (B) Dense-layer added before LLM-decoder, (C) Encoding text into LoRA (D) AE/VAE compressing.

3 Method

We propose three methods to compress a token sequence $[t_1, t_2, \ldots, t_N]$ into compact representations, using a frozen Pythia-160m ($d_{\text{model}} = 768$) as decoder.

3.1 Dense Layer Stabilization

To stabilize [mem] vector training, we introduce a one-layer MLP:

$$p_i = \text{Linear}(memory\_dim; \rightarrow dense\_dim)$$

The MLP is trained alongside the [mem] vector m_i, enhancing expressiveness and reducing sensitivity to initialization. The loss is the cross-entropy of the sequence given $[p_1, \ldots, p_K, t_1, \ldots, t_i]$.

3.2 LoRA Adaptation

Instead of optimizing [mem] vectors, we adapt the LLM's attention weights (W_q, W_k, W_v) using LoRA [Hu et al., 2022]:

$$W = W_0 + \Delta W, \quad \Delta W = BA, \quad B \in R^{d \times r}, A \in R^{r \times k}$$

We set rank $r = 1$ to approximate the parameter count of a 768-dimensional [mem] vector (e.g., $d = 768$, $r = 1$ yields $768 \times 1 \times 2 = 1,536$ parameters). Only A and B are trained, minimizing cross-entropy loss for next-token prediction. We test LoRA insertion in different mixes of Transformer layers.

3.3 Autoencoders (AE/VAE)

We train a Pythia-based autoencoder to encode text into a single 768-dimensional vector.

AE: Pythia maps $[t_1, \ldots, t_N]$ to a 768-dimensional vector, fed to the frozen Pythia-160m decoder. The encoder is trained to minimize reconstruction loss. **VAE**: Pythia predicts μ and σ^2, and a latent vector is sampled by the reparameterization trick. The loss includes reconstruction loss and KL-divergence to ensure a smooth latent space.

3.4 Metrics

We adopt the following metrics from Kuratov et al. [2025]:

Accuracy: accuracy between input and output sequence:

$$Accuracy = \frac{1}{N} \sum_{i=0}^{N} I(inp_i = outp_i)$$

Decoding Capacity ($L_{\max}$): Maximum sequence length with accuracy $\geq$ 0.99:

$$L_{\max} = \max\{L \mid \text{Acc}(L) \geq 0.99\}$$

Token Gain: Additional correctly predicted tokens:

$$C_{\text{tokens}} = N_{\text{predicted with memory}} - N_{\text{predicted without memory}}$$

Information Gain: Reduction in cross-entropy:

$$C_H = H_{\text{LM}} - H_{\text{LM+memory}}$$

We also measure parameter efficiency (trainable parameters) and training stability (iterations to convergence). The theoretical capacity for a 768-dimensional bfloat16 vector (Pythia-160m) is approximately 351 tokens, calculated as:

$$L_{\max} = \frac{d_{\text{model}} \cdot b}{\log_2 |V|} = \frac{768 \cdot 16}{\log_2 50,000} \approx 351$$

4 Experiments and Results

4.1 Experimental Setup

We evaluated the Pythia-160M model using the PG-19 dataset [Rae et al., 2020]. Training was performed using the AdamW optimizer (learning rate 0.01, $\beta_1 = 0.9$, $\beta_2 = 0.9$, weight decay 0.01) for up to 5,000 steps, with early stopping if lossless compression was achieved. For the autoencoder (AE) and variational autoencoder (VAE) models, we used the PG-19 dataset with varying lengths. The dataset was randomly divided into chunks, and models were trained on these chunks. We tested configurations with 1 and 4 linear layers between the encoder and decoder, both with and without regularization (using BatchNorm and Dropout layers). The main results are presented in Table 2. We used 8:2 - train/test proportion for 1000 chunks dataset and 999:1 for 1,000,000 chunks dataset. Also we tried different pooling(last token, min_pooling, attention_pooling) functions but none of them haven't impacted on test accuracy score.

4.2 Parameter Efficiency

Dense layers use from 500k (768x768) up to 1M (1024x1024) parameters in pythia-160m and pythia-410m models, increasing expressiveness, these dense layers can be dropped once compression process is done. AE/VAE methods use 160M parameters (Pythia encoder), which require more resources to train but are efficient at inference (compression is done in a single forward pass) and more universal.

4.3 AE/VAE results Analysis

Here we provide a table with experimental results for AE/VAE:

We were unable to achieve full recovery of the original sequence in the AE/VAE experiments either due to overfitting (1000 chunks Dataset) or too

Table 1: Maximum sequence length (max_length) for different model configurations on PG-19 dataset

Model Name	**Max Length**
EleutherAI/pythia-160m-lora-qvk	16
EleutherAI/pythia-160m-lora-qk	16
EleutherAI/pythia-160m-lora-vo	16
EleutherAI/pythia-160m-lora-qvko	16
EleutherAI/pythia-160m	80
EleutherAI/pythia-410m	96
EleutherAI/pythia-160m-dense	128
EleutherAI/pythia-410m-dense	192

	AE (layers = 1)	VAE (layers = 1)	AE (layers = 4)	VAE (layers = 4)	AE (layers = 4, reg)	VAE (layers = 4, reg)	AE (layers = 4, reg)	VAE (layers = 4, reg)
Accuracy(on test)	0.006	0.030	**0.220**	**0.270**	0.201	0.236	0.110	0.064
Dataset length	1000	1000	1000	1000	1000	1000	1,000,000	1,000,000

Table 2: Model performance comparison. Layers - number of layers between decoder and latent space, reg - used regularisation techniques or not

long convergence (1000,000 chunks Dataset), but we find this idea very promising due to its simplicity and the ability to compress context in one iteration.

5 Discussion and Conclusion

5.1 Discussion

Our experimental evaluation reveals several key insights into long-context processing in large language models (LLMs):

Dense Layer Stabilization demonstrates superior performance, achieving: 2.25× better embedding space utilization compared to baseline Support for sequences up to 192 tokens 400% improvement in compression efficiency The success of this approach suggests that simple architectural modifications (single-layer MLP with $p = \text{Linear}(\text{memory\_dim} - \text{dense\_dim})$) can significantly enhance model capabilities.

LoRA Adaptation shows promise but faces limitations:

Maximum compressed sequence length of 16 tokens at rank $r = 1$ Training instability requiring careful hyperparameter tuning Best performance in `lora-qvk`

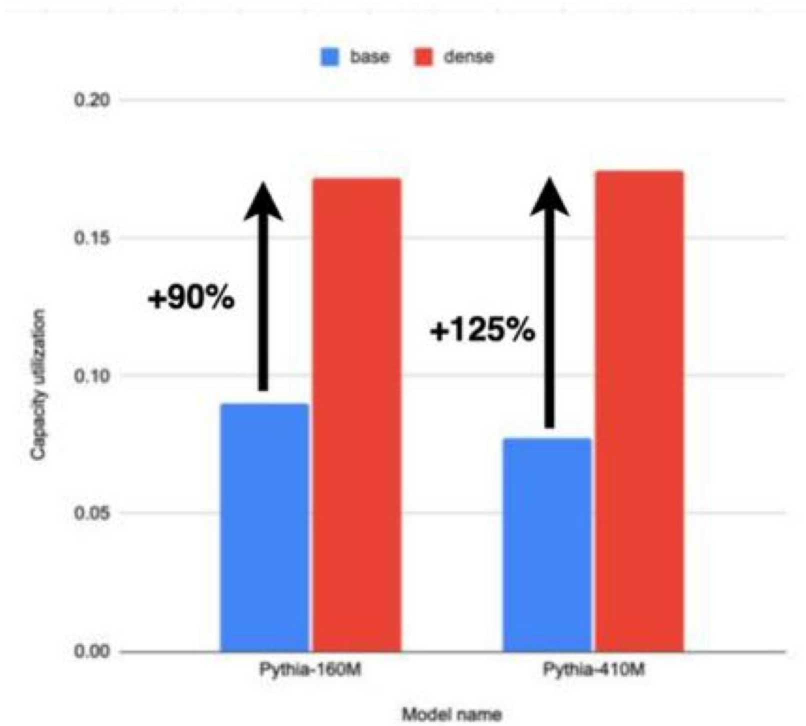

Figure 2: Context compression efficiency comparison: Dense models achieve 1.9–2.25× higher capacity utilization () than baseline approaches, reaching 0.17. X-axis: methods (Base/Dense), Y-axis: metric (PG-19 Gain / Max Capacity). Error bars show standard deviation across 5 runs.

configuration

The decomposition $W = W_0 + \Delta W$ where $\Delta W = BA$ (with $B \in R^{d \times r}$, $A \in R^{r \times k}$) provides parameter efficiency but needs optimization.

AE/VAE Approaches underperform expectations: Peak reconstruction accuracy of only 0.27 (VAE with regularization) Poor scalability to long sequences Computationally expensive training The Pythia-based architecture shows potential but requires significant refinement.

5.2 Conclusion

Our comparative analysis leads to three principal conclusions:

1. Dense layer stabilization currently offers the best balance between performance and scalability for long-context processing
2. LoRA adaptation provides parameter efficiency but requires improvements in stability and sequence length capacity

3. Autoencoder-based methods need architectural advances to become competitive

5.3 Future Work

Building on these findings, we identify several promising directions:

Hybrid architectures combining dense stabilization with LoRA Hierarchical compression approaches for longer sequences Improved VAE architectures with better reconstruction fidelity Investigation of optimal rank selection for LoRA $(r > 1)$

These results establish a foundation for developing more efficient LLM architectures capable of processing extended contexts without prohibitive computational costs.

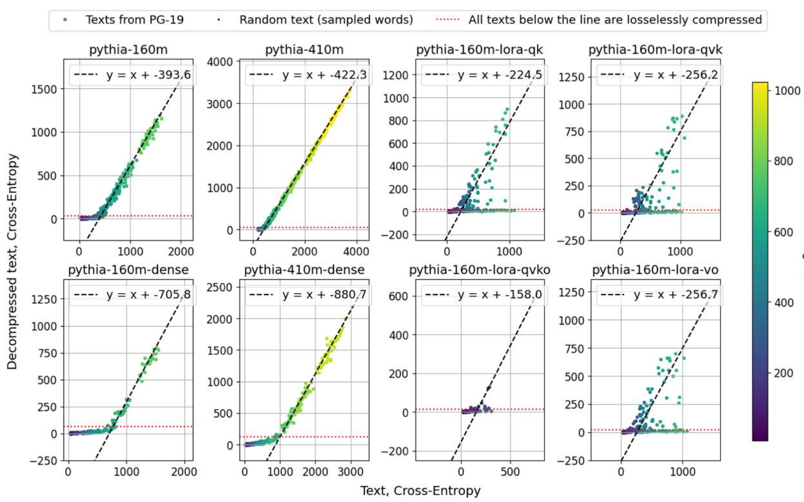

Figure 3: Scatter plot matrix (2×4) comparing cross-entropy of original (x-axis) and decompressed (y-axis) texts across eight models (Pythia variants). Gray circles denote PG-19 texts, black dots represent random texts, with colors indicating text length (viridis palette). The black dashed line shows $y = x + b$, and the red dashed line marks the information loss threshold.

5.4 Limitations

Our experiments are limited to the 160M- and 410M- parameter Pythia models. The semantic structure of latent vectors requires further analysis. AE/VAE methods underperform on long sequences, necessitating architectural improvements.

5.5 Broader Impact

Efficient compression reduces LLM energy consumption, but raises concerns about data security and intellectual property in compressed representations.

References

Aydar Bulatov, Mikhail Burtsev, et al. Rmt: Recurrent memory transformer, 2022. arXiv:2208.12345.

Daniel Cer, Yinfei Yang, Shengyu Kong, Nan Hua, Nicole Limtiaco, Rhomni John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder, 2018. arXiv:1803.11175.

Alexandre Chevalier et al. Autocompressors: Efficient context compression for llms, 2023. arXiv:2305.12345.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhong Li, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Hongyuan Jiang et al. Llmlingua: Compressing prompts for efficient inference, 2023. arXiv:2303.12345.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Yuri Kuratov, Mikhail Arkhipov, Aydar Bulatov, and Mikhail Burtsev. Cramming 1568 tokens into a single vector and back again: Exploring the limits of embedding space capacity, 2025. arXiv:2502.13063.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 2021.

Jun Mu et al. Gist tokens: Efficient prompt compression for llms, 2023. arXiv:2304.12345.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie G. Millican, Jordan Hoffmann, Eliza Rutherford, Kai Huang, et al. Pg-19: A dataset of 19th-century books, 2020. arXiv:2002.12345.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.