More Effective LLM Compressed Tokens with Uniformly Spread Position Identifiers and Compression Loss

Anonymous ACL submission

Abstract

Compressing Transformer inputs into compressd tokens allows running LLMs with improved speed and cost efficiency. Based on the compression method ICAE, we carefully examine the position identifier choices for compressed tokens and also propose a new compression loss. We demonstrate empirically that our proposed methods achieve significantly higher compression ratios (15x compared to 4x for ICAE), while being able to attain comparable reconstruction performance.

1 Introduction

Transformers (Vaswani, 2017), as the backbone of most LLMs, faces quadratic computational demands and linear memory costs when dealing with increasingly longer contexts. To alleviate this issue, research has primarily focused on enhancing the efficiency of the attention mechanism (Beltagy et al., 2020; Ding et al., 2023) or compressing contextual information (Mu et al., 2024; Chevalier et al., 2023; Ge et al., 2024). In this work, we follow the approach that learns compressed tokens (Ge et al., 2024). Compressed tokens are a small number of tokens trained to store the information of input tokens; at inference time, LLM generation is mostly conditioned on the compressed tokens in KV cache. Therefore, compressed tokens hold the promise not only to handle long contexts but also to improve speed and cost efficiency for LLMs.

To obtain an LLM including the compressed tokens, ICAE (Ge et al., 2024) proposes to continue to pretrain LLMs on auto-encoding (AE) tasks as well as language modelling (LM) tasks using LoRA (Hu et al., 2021b). With minimal parameter adjustments, ICAE can effectively compress contextual information into a few tokens. Through further fine-tuning, the model using compressed tokens can perform on par with those using the entire historical context, and significantly outperforms out-of-the-box approaches like conditioning on auto-generated summaries. However, ICAE learns the compressed tokens only at 4x ratio (1 compressed token for 4 original tokens) and in this work, we investigate if we can optimize the approach further to enable a higher compression ratio. We mainly explore two ideas[1]:

- The compressed tokens that ICAE learns leverage default positions embeddings where the position identifiers simply follow the input. Given that widely used position embeddings such as RoPE (Su et al., 2021) guarantee to better capture the context close to the position, we design our position identifiers for the compressed tokens to spread uniformly across the input. The design gives a prior to the compressed tokens to memorize all the original input before learning the compressed tokens. We show empirically that such design significantly improves the compression ratio. For example, while most of the experiments in ICAE is carried out with 4x compression ratio, our compressed tokens retain most of the information at 15x.

- While the compressed tokens learn to retain all original input throught auto-encoding (AE), the AE task combines memorization and language modelling; in this work, we explore alternative training tasks that focus solely on memorization so that the compressed tokens are learned under improved efficiency. More specifically, our model learns to decode the original context segments directly from the compressed tokens (i.e without relying on the decoded context as in AE) with a new 'compression loss'. We observe that this compression loss can replace the AE loss, ensuring the same compression ratio with improved learning efficiency.

[1]We have released our codes at https://anonymous.4open.science/r/compress-5550.

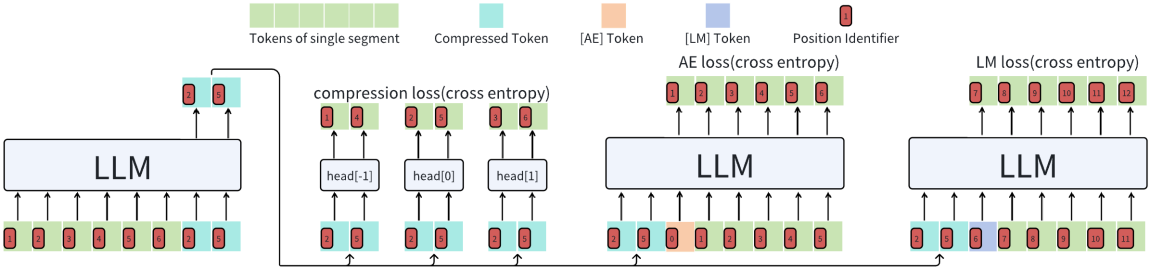


Figure 1: Proposed training architecture to train compressed tokens (shown in light blue).

2 Methods

2.1 Continued Pretraining

Figure 1 illustrates our training architecture. The trainings starts from a pretrained language model to further learn to incorporate the compressed tokens. In ICAE (Ge et al., 2024), the authors propose to learn the compressed tokens using a combination of AE loss and LM loss. Our LLM training differs from ICAE mainly in the following two aspects:

Compressed token position identifiers Position identifiers are integers that indicate the token positions in a transformer architecture. Without specific handling, a sequence of m tokens have $1, 2, ..., m$ as position identifiers for the corresponding tokens For the compressed tokens, as is the case for ICAE. If we apply ICAE in Figure 1, the compressed tokens will have position identifiers as 7 and 8 since the context range from 1-6. [2]

We notice that the position identifiers can be more carefully chosen to better represent and thus compress the original sentence. We restrict our discussions to the popular RoPE embeddings (Su et al., 2021) which has been used in most advanced open-source LLMs such as LLama3 (et al., 2024) and Mistral (Jiang et al., 2024). For RoPE, attention weights decays in an exponential way with respect to the relative token positions, specified by the position identifiers.[3]

Based on this position embedding property, we specify the compressed token position identifiers to spread uniformly across the input tokens so that each compressed token *a priori* learns to compress a corresponding input segments. To compress a sequence of m tokens with n compressed tokens, the n position identifiers spread uniformly across range $(1, m)$ compared to $m+1, ..., m+n$ in ICAE.

compression loss The AE task is strictly easier than memorizing all context tokens. For a given token, both the prefix and what the model has memorized contribute to its correct prediction in AE. Given this observation, we propose a loss that intends to further enhance model's memorization performance for the compressed tokens. As shown in the left of Figure 1, we experiment with a loss where heads are learned to project the compressed tokens into each segments of the original input tokens. In Figure 1, the compressed token with position identifier 2 projects onto the original tokens 1-3 and the compressed token with position identifier 5 projects onto the original tokens 4-6.

2.2 Fine-tuning and Inference

After LLM continued pretraining (with the compressed tokens), one can further apply instruction fine-tuning to the LLM where the losses (Compression loss, AE loss and LM loss) are replaced by the supervised learning loss from the instruction fine tuning dataset.

During inference, most input tokens are discarded once their corresponding compressed tokens are generated[4], making the decoding process faster and more cost effective, similar to ICAE (Ge et al., 2024).

3 Experiments

We evaluate the effectiveness of our proposed architecture for continued pretraining as well as for fine-tuning. For continued pretraining, we are mainly concerned about the compression ratio at which the compressed tokens can achieve almost perfect reconstruction. We measure the downstream task performance for LLMs equipped with compressed tokens for fine-tuning tasks.

[2] Our compressed tokens receive identifiers 2 and 5 as illustrated in Figure 1.

[3] More precisely, one can prove that the upper bound for attention weights decay in an exponential way with respect to relative position difference. See 3.4.3 in (Su et al., 2021)

[4] Some context in PwC dataset (Ge et al., 2024) exceeds our compression window; in this case, we concatenate those original tokens after the compressed tokens during decoding.

Dataset For continued pretraining, we train our model including the compressed tokens using SlimPijama (Shen et al., 2024) dataset. After tokenization, we sample all passages longer than 4,096 characters and then further sample two training instances around 500 tokens from each passage to train the model with compressed tokens. In total, we train on around 1.4B tokens. We reserve 1,000 passages in SlimPijama for evaluations. For the fine-tuning experiments, we use the PwC dataset containing 18k (context, prompt, response) samples for testing to facilitate comparisons with ICAE (Mu et al., 2024).

Training details For continued pretraining, we train our model with the compressed tokens from Llama2-7B-hf (et al., 2023). Our training is performed on a 8xH20 machine where we feed 4 examples to each device (thus with 32 as batch size). We perform training using LoRA (Hu et al., 2021a) for embedding layers as well as linear layers; we train also RMSNorm (Zhang and Sennrich, 2019) layers. Without further specifications, the linear LoRA rank is set to 16 and the embedding LoRA rank is set to 128. At each optimization step, the AE loss, LM loss and compression loss are optimized with equal weights. We use AdamW (Loshchilov and Hutter, 2017) to optimize the whole architecture.[5] The fine-tuning stage shares the same training strategy as the continued pretraining, with the loss optimization soly focusing on the downstream tasks.

3.1 Compression Ratio evaluations

We first study the compression ratio for our proposed architecture. Table 1 shows the results where 31x means compressing 31 input tokens into 1 compressed token. It is natural to observe that as we increase the ratio, reconstruction metrics (autoencoding (AE) loss, language model loss and BLEU4) measured on the test set degrade. Overall our architecture is able to reconstruct with low error rate at 15x (96.9 BLEU4 score); comparatively, ICAE reports only 4x compression ratio.

We conduct experiments to see what factors influence most the compression effectiveness. The full setting is our proposed settings and we perform ablation studies by removing the compression loss or the AE loss as well as reproducing ICAE by removing both position identifiers and the compression loss [6]. The experimental results are shown in Table 2, Pretrain metric.

Compress Ratio	AE loss	LM loss	BLEU4
5x	0.01	2.01	99.7
15x	0.03	2.08	96.9
31x	0.11	2.18	84.4

Table 1: Various loss by training our full architecture (with AE loss, LM loss and Compression Loss) using different compression ratios.

Either without AE loss or without compression loss (CL loss), the LM loss reamins at a relatively low level (2.06) compared to the full setting (2.08), suggesting that CL might be a viable alternative to AE loss for enhancing LLM model memorization during learning[7], for which we further confirm for the fine-tuning experiments. Our ICAE reproduction (wo CL + PI) shows that our conceived position identifiers contribute significantly to model's memorization; when incorporating the identifiers alone (i.e. wo CL), the LM loss decreases from 2.10 to 2.06 and BLEU4 improves from 52.7 to 95.8.

3.2 Downstream task performance

In this section, we examine downstream fine-tuning performance across different architectures using a 15x compression ratio during continued pretraining. As shown in Table 2, the downstream metrics on the PwC test set align with pretraining results. The full setting delivers strong BLEU4 scores, and similar performance is possible without compression or AE loss, confirming compression loss as a viable substitute. However, removing the designed position identifiers leads to a significant drop in BLEU4 performance (26.0 vs 22.1).

We also compare win rates using LLM-based evaluation, following ICAE (Mu et al., 2024), as it may capture nuances beyond n-gram metrics like BLEU (Papineni et al., 2002) or ROUGE (Lin, 2004). We use GPT-4o as the judge with the same prompts as ICAE. Results are shown in Table 3.[8] The comparison between AE loss and CL loss shows that GPT-4o prefers the model with CL loss,

[5] For AdamW, $lr = 0.0001$, $betas = (0.9, 0.95)$, $weight_decay = 0.1$.

[6] Our implementations mainly differ with ICAE in lora choices and we conduct our experiments based on Llama2 (et al., 2023).

[7] Particularly, wo AE experiments finishes in less time than wo CL experiments (24.3h vs 34.4h).

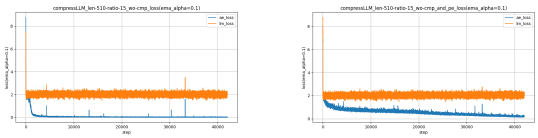
[8] The percentages do not add up to 100% due to some cases where GPT-4o failed to generate or produced invalid JSON.

Settings	Pretrain metric				Downstream metric
	CL loss	AE loss	LM loss	AE BLEU4	PwC BLEU4
full	0.17	0.03	2.08	**96.9**	26.2
wo CL	-	**0.02**	**2.06**	95.8	26.0
wo AE	0.15	-	**2.06**	-	**26.5**
wo CL + PI	-	0.17	2.10	52.7	22.1
full (rank↑)	0.09	0.01	2.18	99.6	26.4

Table 2: Various loss by training our architecture using different settings under same compress ratio(15x). Loss that are not trained are also not shown.

Comparisons	Win	Lose	Tie
Full vs wo CL	24.8	25.0	49.0
wo CL vs wo CL + PI	37.1	16.7	45.0
wo AE vs wo CL	25.2	24.0	49.9

Table 3: GPT-4o evaluation comparing different pretrained models after fine-tuning.



(a) with PI (b) without PI

Figure 2: Learning curves for LM loss (orange) and AE loss (blue) with and without PI.

aligning with the BLEU4 results in Table 2. Therefore, the importance of PI is reaffirmed.

4 Analysis

Higher Lora ranks We investigate whether higher LoRA rank can enable higher compression rate and consequently better downstream performance. We conduct experiments by changing the LoRA rank from 16 to 256 for the linear layers. For the same compress ratio (15x), Table 2 Pretrain metric shows that increasing rank can still improve the compression performance as the BLEU4 improves from 96.9 to 99.6. The improved compression performance also transfers to downstream; as Table 2 Downstream metric shows, the BLEU4 score for the corresponding PwC answers also improve from 26.2 to 26.4.

Learning curve We plot the learning curve with and without position identifiers (PI) to investigate how PI improves the performance using the ICAE architecture with LM loss and AE loss. Figure 2 shows that the AE loss converges much quickly with architectures including designed PI. With PI, the AE loss (which is used to enhance model memorization) converges to 0 quickly while it takes much longer for the architectures without PI to converge. In our experiments, the AE loss for the architecture without PI does not yet converge to 0.[9]

[9]Though AE loss might be further minimized by training on more tokens as (Ge et al., 2024) training on tens of billions of tokens.

5 Related Work

Our work is most related to (Ge et al., 2024) as both works leverage AE as an additional task to learn the compressed tokens; we achieve further improvement by carefully choosing the position identifiers for the compressed tokens. Our position identifier choices are based on the RoPE (Su et al., 2021) embedding property that guarantees to assign larger attention weights to closer tokens; such property is also extensively used in context extension literature (Jin et al., 2024; Peng et al., 2023; Chen et al., 2023).

6 Conclusion

Following ICAE (Mu et al., 2024), we propose to learn compressed tokens to improve LLM efficiency. We achieve higher compress ratio by introducing position identifiers for the compressed tokens; we propose a compression loss that proves to be a more efficient alternative to the autoencoding loss for enhancing model memorization. We confirm that architectures that compress better the context also achieves better downstream performance when fine-tuned and factors such as LoRA ranks also need to be carefully chosen to achieve the best performance. Future research involves further investigating the strength of each loss to propose the best training settings.

Limitations

This work has two main limitations. First, we only explored the possibility of higher compression rates within a single segment, without conducting experiments on compressing multiple segments like ICAE's multi-span. Second, we didn't explore different weight distributions for task losses in multi-task training, which could potentially be optimized.

References

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

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. *Preprint*, arXiv:2306.15595.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. *Preprint*, arXiv:2305.14788.

Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. 2023. Longnet: Scaling transformers to 1,000,000,000 tokens. *Preprint*, arXiv:2307.02486.

Abhimanyu Dubey et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Hugo Touvron et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2024. In-context autoencoder for context compression in a large language model. *Preprint*, arXiv:2307.06945.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021a. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021b. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. 2024. Llm maybe longlm: Self-extend llm context window without tuning. *Preprint*, arXiv:2401.01325.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*, page 10.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2024. Learning to compress prompts with gist tokens. *Preprint*, arXiv:2304.08467.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *Preprint*, arXiv:2309.00071.

Zhiqiang Shen, Tianhua Tao, Liqun Ma, Willie Neiswanger, Zhengzhong Liu, Hongyi Wang, Bowen Tan, Joel Hestness, Natalia Vassilieva, Daria Soboleva, and Eric Xing. 2024. Slimpajama-dc: Understanding data combinations for llm training. *Preprint*, arXiv:2309.10818.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding. *CoRR*, abs/2104.09864.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *CoRR*, abs/1910.07467.