Table 6: The results of pretrained ICAE (512→128) based on different target LLMs

Target LLM	AE		Text Continuation		
	BLEU(%)	Loss	PPL (original context)	PPL (memory slot)	Δ
Llama-7b	99.1	0.017	9.01	9.50	+0.49
Llama-2-7b	99.5	0.009	8.81	9.18	+0.37
Llama-2-13b	99.8	0.004	8.15	8.45	+0.30

Table 7: Latency comparison of LLM (generation) and LLM+ICAE (compression then generation)

Input (Batch×Length)	Method	Compression Time (Cachable)	Decoding Time	Total Time
8*2048	LLM	-	24.0	24.0
	LLM+ICAE	3.4	3.9	7.3 (3.3×)
8*512	LLM	-	9.3	9.3
	LLM+ICAE	0.6	3.7	4.3 (2.2×)
32*512	LLM	-	24.3	24.3
	LLM+ICAE	2.6	4.2	6.8 (3.6×)

even more significant – around 3.5× – in compute-intensive scenarios (e.g., 8×2048 and 32×512). Given that the compressed memory slots can be cached in advance (for frequently used texts like textbooks, government reports or articles of law), ICAE may introduce over 7× inference speedup in these cases. Details of the profiling are presented in Appendix B.

3.3.3 Multiple Spans of Memory Slots

Thus far, we have mainly discussed a single span of memory slots. In this section, we shall discuss multiple spans of memory slots. As illustrated in Figure 6(Left), we can segment a long context into N chunks, compress them individually, and then concatenate them to represent the original long context. However, this did not work initially, because the model had never seen multiple span concatenation patterns during training. Fortunately, we can incorporate a small number of multiple span concatenation samples during training, enabling the model to work with concatenated spans of memory slots, as OpenAI's work (Bavarian et al., 2022) on introducing the "fill in the middle" ability for the GPT. The results in Figure 6(Right) indicate that, using an equivalent length context, ICAE's memory achieves better performance – because memory can represent $4\times$ the original context length.

The ability of ICAE demonstrates great promise to handle long contexts, as it can save a significant amount of GPU memory when addressing long contexts without touching the existing LLM. As illustrated in Figure 6(Right), 2048-length memory slots can perform on par with 4096-token contexts. This means that conditioning on 2048 memory slots instead of the original 4096 context tokens can save about 20GB of GPU memory[5] with minimal quality degradation.

4 Related Work

Prompt compression and context distillation (Askell et al., 2021; Snell et al., 2022) are closely related areas to this work: Wingate et al. (2022) proposed a method to learn compact soft prompts to simulate the original natural language prompt by optimizing the KL divergence. However, this approach has a very high computational cost, as it requires performing back-propagation for each new incoming prompt to learn and obtain the compressed prompt, which severely limits its application. Qin & Van Durme (2023) propose Neural Agglomerative Embeddings named NUGGET, which encodes language into a compact representation for an encoder-decoder model.

The most closely related studies to our research are GIST (Mu et al., 2023) and AutoCompressors (Chevalier et al., 2023). GIST achieves prompt compression by fine-tuning an LLM in a similar way to ours. The resulting model can produce gist tokens as the compression of a prompt, which are similar to our memory slots. Nonetheless, this approach is limited to compressing short prompts[6] and

[5]Llama-7b (fp16) requires 24GB GPU memory for 2048 context tokens and 44GB for 4096 during inference (measured without optimization like flash attention).

[6]Prompts in Mu et al. (2023) refer to task instructions before input texts, so they are usually short.