2 IN-CONTEXT AUTOENCODER

2.1 MODEL ARCHITECTURE

Like a typical autoencoder (Kramer, 1991), ICAE consists of an encoder and a decoder. Similar to the design of Gisting (Mu et al., 2023) and AutoCompressor (Chevalier et al., 2023), the ICAE performs both the encoding and decoding processes in an in-context manner, as illustrated in Figure 3.

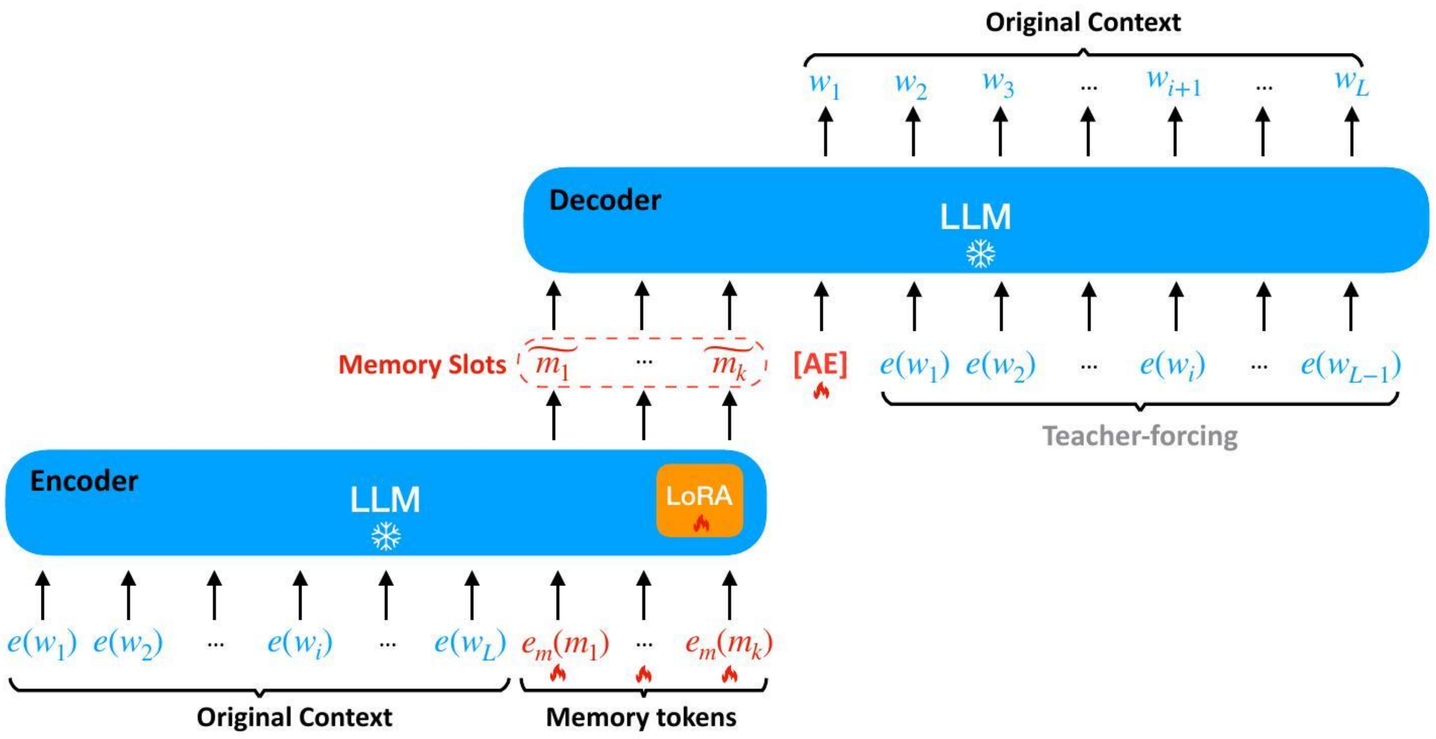

Figure 3: The encoder of the ICAE is a LoRA-adapted LLM, which is used for encoding the original context $\boldsymbol{c} = (w_1, w_2, \ldots, w_L)$ into a few memory slots $(\widetilde{m_1}, \ldots, \widetilde{m_k})$. The decoder of the ICAE is the target LLM itself that can condition on the memory slots produced by the encoder for various purposes (e.g., the autoencoding task as in this figure). $e(\cdot)$ denotes the word embedding lookup in the target LLM and $e_m(\cdot)$ denotes the learnable embedding lookup of memory tokens that are used for producing memory slots."[AE]" is a special token to indicate the autoencoding pretraining task.

Given the intuition, we propose to use a LoRA-adapted LLM as the encoder of the ICAE, as illustrated in Figure 3. When encoding a context $\boldsymbol{c} = (w_1, \ldots, w_L)$ with the length L, we first append k ($k << L$) memory tokens $(m_1, \ldots, m_k)$ to the context $\boldsymbol{c}$ to obtain their outputs $(\widetilde{m_1}, \ldots, \widetilde{m_k})$ as the memory slots for the context $\boldsymbol{c}$. Therefore, the ICAE encoder is very lightweight – it only adds a LoRA adapter and an embedding lookup for memory tokens compared with the target LLM.

As introduced above, we expect the memory slots $(\widetilde{m_1}, \ldots, \widetilde{m_k})$ to be conditioned on by the target LLM on behalf of the original context $\boldsymbol{c}$. Therefore, we use the untouched target LLM as the decoder of the ICAE to ensure the compatibility of memory slots within the target LLM.

2.2 PRETRAINING

2.2.1 AUTOENCODING

Like a typical autoencoder, one of the ICAE's pretraining objectives is to restore the original input text $\boldsymbol{c}$ of the length L from its produced memory slots $(\widetilde{m_1}, \ldots, \widetilde{m_k})$ of the length k:

$$\mathcal{L}_{\text{AE}} = \max_{\widetilde{m_1}, \ldots, \widetilde{m_k}} P(\boldsymbol{c} | \widetilde{m_1}, \ldots, \widetilde{m_k}; \Theta_{LLM}) = \max_{\Theta_{LoRA}, e_m} P(\boldsymbol{c} | m_1 \ldots m_k; \Theta_{LLM}, \Theta_{LoRA}, e_m)$$

To indicate the autoencoding task, we append a special token "[AE]" to $(\widetilde{m_1}, \ldots, \widetilde{m_k})$ in the decoder, as Figure 3 shows. As this pretraining objective does not need any extra annotation, we can use massive text data to train the In-context Autoencoder.

2.2.2 TEXT CONTINUATION

While autoencoding pretraining offers a straightforward learning objective to encode a context, its inherent simplicity and exclusive focus on the single objective may lead to suboptimal generalization. To address this issue, we incorporate an additional objective during the pretraining phase: text continuation, as illustrated in Figure 7 in Appendix A. This self-supervised task is widely acknowledged to facilitate the learning of more generalizable representations in language models:

$$\mathcal{L}_{\text{LM}} = \max_{\widetilde{m_1}, \ldots, \widetilde{m_k}} P(\boldsymbol{o} | \widetilde{m_1}, \ldots, \widetilde{m_k}; \Theta_{LLM}) = \max_{\Theta_{LoRA}, e_m} P(\boldsymbol{o} | m_1 \ldots m_k; \Theta_{LLM}, \Theta_{LoRA}, e_m)$$