# UniICL: An Efficient Unified Framework Unifying Compression, Selection, and Generation

Anonymous ACL submission

## Abstract

In-context learning (ICL) enhances the reasoning abilities of Large Language Models (LLMs) by prepending a few demonstrations. It motivates researchers to introduce more examples to provide additional contextual information for the generation. However, existing methods show a significant limitation due to the problem of excessive growth in context length which causes a large hardware burden. In addition, shallow-relevant examples selected by out-off-shelf tools hinder LLMs from capturing useful contextual information for generation. In this paper, we propose **UniICL**, a novel **Uni**fied **ICL** framework that unifies demonstration compression, demonstration selection, and final response generation. Furthermore, to boost inference efficiency, we design a tailored compression strategy that allows UniICL caching compression results into **Demonstration Bank** (**DB**) which avoids repeated compression of the same demonstration. Extensive out-of-domain evaluations prove the advantages of UniICL in both effectiveness and efficiency.

## 1 Introduction

In-context learning (ICL) (Brown et al., 2020; Xie et al., 2021; Wang et al., 2023b) to enhance the reasoning ability of Large Language Models (LLMs) with a few demonstrations prepended (Wang et al., 2023d; Yang et al., 2023; Wei et al., 2023; Wang et al., 2023a; Min et al., 2022). Inspired by its outstanding performance, researchers explored applying ICL on many tasks such as text summarization (Wang et al., 2023d; Yang et al., 2023; Gao et al., 2024), sentiment classification, and linguistic acceptability (Min et al., 2022; Wang et al., 2019). However, two challenges hinder the impact of ICL currently: (1) concatenated demonstrations directly surge the input length, causing a large hardware burden (2) the prepended demonstrations are randomly sampled or selected via out-off-shelf tools which tend to provide shallow relevant demonstrations, hindering LLMs from capturing useful contextual information for generation. Existing work tackles the two challenges separately.

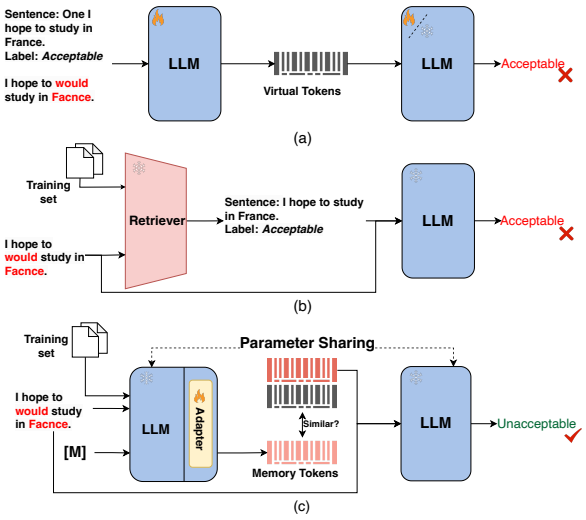


Figure 1: (a) Prompt compression methods that indiscriminately compress both demonstrations and queries.(b) Retrival-based demonstration selection methods select lexical demonstrations. (c) UniICL discriminately compresses demonstrations and performs selection upon the compression results.

To alleviate input length surge, on the one hand, many efforts are made in modifying model architecture to accommodate longer contexts (Zheng et al., 2022; Wu et al., 2022; Ding et al., 2023; Bulatov et al., 2023). These methods usually require training models from scratch and models with million context windows still struggle to overcome performance degradation (Liu et al., 2024). On the other hand, recent studies attempt to shorten inputs through prompt compression(Wingate et al., 2022; Mu et al., 2023; Jiang et al., 2023; Ge et al., 2023). However, these compression methods are not applicable to ICL because indiscriminately compress both demonstrations and queries into virtual tokens.

For instance, as illustrated in Fig. 1(a), the task entails justifying whether the query is grammatically acceptable. The latter generator makes responses only according to virtual tokens generated by the compressor, resulting in a wrong answer[1]. More importantly, current compression methods are costly to train (Wingate et al., 2022; Mu et al., 2023; Jiang et al., 2023), and compressors are either limited to compressing within the original model's allowed input length (Mu et al., 2023; Jiang et al., 2023; Ge et al., 2023) or bringing significant inference latency (Wingate et al., 2022).

Retrieval-based In-context Example Selection (RICES) methods (Alayrac et al., 2022) integrate an out-off-shelf pre-training model to select demonstrations similar to the queries in a shallow level. These demonstrations usually contain redundant information and bring minimal benefits for final generation (Liu et al., 2021; Ram et al., 2023; Wang et al., 2024). Existing work attempts to train the retrieval model and the generator in an end-to-end manner have shown better performance in in-domain datasets (Wang et al., 2023c). However, this approach still performs poorly in out-of-domain datasets. For instance, as shown in Fig. 1(b), the retriever selects an example lexically similar to queries but has *contrast* labels. Then, the LLM is misleaded and responds to a wrong answer.

In light of challenges in ICL, we turn to leverage the inherent understanding ability of LLMs developed during pre-training. We accordingly propose a **Uni**fied **ICL** (**UniICL**) framework, which unifies demonstration compression, demonstration selection, and response generation. As shown in Fig. 1(c), for lightweight training, in UniICL, both the compressor and generator are initialized from the same LLM and kept frozen. An adapter is introduced to align the compressor with the generator, and **[M]** is a learnable embedding called **Memory Slot** which is attached behind demonstrations for compression. Therefore, UniICL only contains **17M** trainable parameters. The LLM compressor first compresses each demonstration from the training set and queries into **Memory Tokens** independently on top of Memory Slots. Then, UniICL selects $n$ most relevant demonstration based on the similarity of Memory Tokens between queries and demonstrations. Finally, Memory Tokens of selected demonstrations are concatenated to formulate a global in-context sequence, together with queries fed into the generator for response generation. Due to independent compression, the compressor gets rid of the input window limitation of original LLMs with the number of demonstrations increasing. In addition to improvements in window limitation, the tailored compression strategy further makes improvements to ICL efficiency. Specifically, UniICL caches Memory Tokens of different demonstrations to configure the **Demonstration Bank** (**DB**) for future reusing as shown in Fig. 2. Therefore, repeated compression of the same demonstration is not necessary, which significantly boosts model efficiency in Fig. 8. Extensive out-of-domain evaluation indicates UniICL achieves substantial improvements compared with other baselines. Our main contributions are as follows:

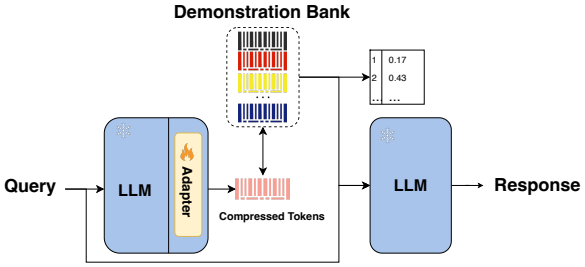


Figure 2: The workflow of Demonstration Bank.

- To our knowledge, we are the first to propose a unified ICL framework with 17M trainable parameters.
- UniICL proposes configuring the Demonstration Bank to avoid repeated compression for the same demonstration, which significantly boosts ICL efficiency.
- Different from the indiscriminate compression of previous studies, UniICL proposes a tailored compression strategy for ICL, achieving substantial improvements compared with other baselines.

## 2 Related Work

### 2.1 Soft Prompt Compression

Recently, researchers attempted to utilize soft prompts to convert actual tokens to dense-information virtual tokens. Mostly from a distillation perspective, Wingate et al. (2022) aligned the teacher model and the student model, where the teacher model accepted the actual task instruction while the student model fed the soft prompt. The

[1]I hope to ~~would~~ study in ~~Faence~~ (France)

main drawback of this approach was the lack of generalization that necessitated training for each lexically different instruction. To tackle the generalization problem, Mu et al. (2023) proposed to learn a Llama-7b to compress instruction to virtual tokens, but only compress instruction was not powerful enough since the demonstrations were much longer in practice. To compress longer prompts, Chevalier et al. (2023) proposed AutoCompressor to recurrently generate compressed virtual tokens based on a fine-tuned Llama (Zhang et al., 2022). However, AutoCompressor broke the independence of demonstrations, and the recurrent compression increased inference latency. Ge et al. (2023) proposed ICAE that employed a LoRA-adopted Llama-7b (Touvron et al., 2023) to compress the processed demonstrations to compact virtual tokens, while ICAE still struggled to overcome quite long inputs.

### 2.2 Extractive Compression

Apart from employing soft prompts, researchers also endeavored to shorten prompts by extracting informative tokens from the original ones (Li, 2023; Jiang et al., 2023), namely token pruning (Kim et al., 2022) or token merging (Bolya et al., 2022). Recent works like LLMLingua (Jiang et al., 2023) and Selective Context (Li, 2023) shared similarities but diverged on whether to eliminate tokens with high or low Perplexity (PPL). LLMLingua emphasized tokens with high PPL, attributing them as more influential, resulting in achieving outstanding performance. As mentioned in their paper, extractive compression methods encountered Out-of-Distribution (OOD) issues between the extractor and the target LLM. To reconcile this, they fine-tuned Alpaca-7b (Taori et al., 2023) using the Alpaca dataset (Taori et al., 2023) to perform the alignment.

## 3 Methodology

Previous compression methods are not tailored for ICL, and they are either bound by serious inference latency or poor performance, as demonstrated in Appendix A. We propose UniICL, a unified ICL framework that unifies demonstration compression, demonstration selection, and response generation. As for the selection of the underlying LLM, previous work has proved that the Decoder-only model performs better than the Encoder-Decoder model in prompt compression (Mu et al., 2023). We follow this conclusion and adopt Vicuna-7B (Zheng et al., 2023) as the underlying backbone in UniICL.

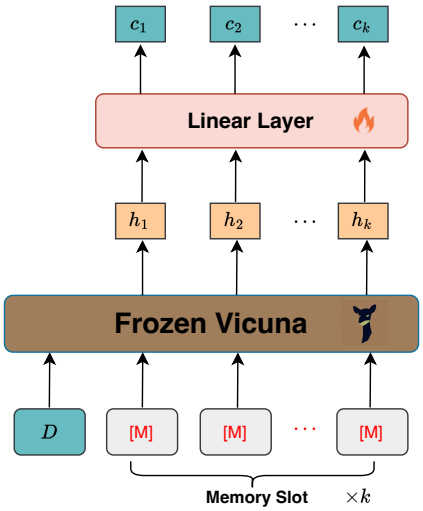


Figure 3: Demonstration compression. $k$ Memory Slots are attached behind each demonstration.

### 3.1 Demonstration Compression

UniICL introduces Memory Slots **[M]** $\in \mathcal{R}^d$, a learnable $d$-dimension embedding initialized from a rarely used embedding of the target LLM. UniICL activates the Memory Slots to extract information from demonstrations in the forward propagation $f_\theta(\cdot)$ of frozen Vicuna, as illustrated in Fig. 3. We first attach $k$ Memory Slots $M = k \times$ [M] behind each demonstration $D_i$, formatting modified prompt fed to the Vicuna. Then, frozen Vicuna infers the modified prompts and outputs the last hidden states $H^i = (h_1, h_2, ..., h_k)$ on top of the $k$ Memory Slots:

$$H^i = f_\theta(D_i^{L_i \times d} \oplus M^{k \times d}), \quad (1)$$

where $L_i$ is the $i-$th demonstration length, $d$ is the embedding dimension and $\oplus$ means token-level concatenation. Due to the attention mechanism, $H^i$ is compelled to attend to the preceding actual tokens. Then, UniICL applies a linear layer as the adapter for efficiency to convert $H^i$ to Memory Tokens $C^i = (c_1^i, c_2^i, ..., c_k^i)$, performing alignment between the compressor and the generator[2]:

$$c_j^i = W_p^{d \times d} \cdot h_j^i, \quad (2)$$

where $W_p$ is the parameters of the projection layer.

[2]Linear layer is enough for UniICL as features have interacted with each other during compression.

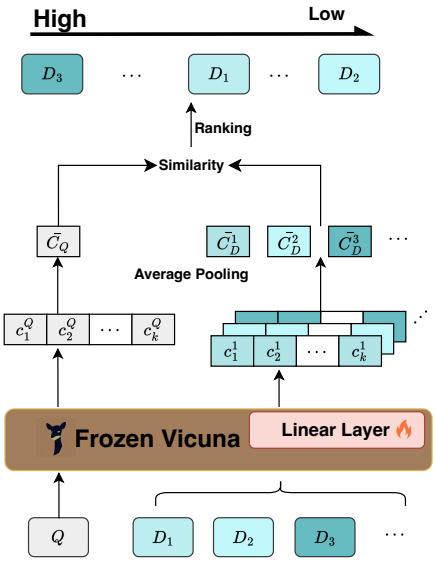


Figure 4: Demonstrations selection.

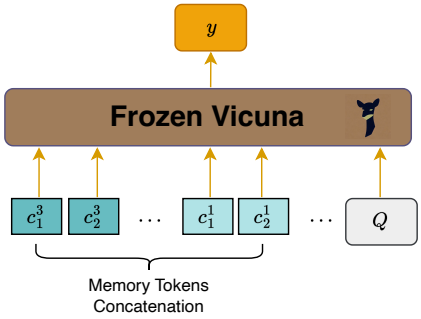


Figure 5: In-context generation. The Memory Tokens from different demonstrations are concatenated horizontally at the input end of Vicuna.

### 3.2 Demonstration Selection

Memory Tokens $C^i$ naturally summarize the demonstrations in latent space, and UniICL performs demonstration selection based on the similarity between queries and demonstrations as shown in Fig. 4. Specifically, given a query $Q$ and its candidate demonstrations $(D_1, D_2, ..., D_n)$, UniICL obtains their representations used for selection by average pooling $C_{\{Q,D\}}$:

$$\bar{C^i}_{\{Q,D\}} = \frac{1}{k}\sum_{j=1}^{k} c_j. \quad (3)$$

We define the $i$-th demonstration saliency score $S_i$ as the cosine similarity between $\bar{C_Q}$ and $\bar{D}_i$:

$$S_i = \text{cosine\_similarity}(\bar{C_Q}, \bar{C^i_D}). \quad (4)$$

### 3.3 Generation

We employ the frozen Vicuna again to generate responses with the guiding of concatenated Memory Tokens and queries, as illustrated in Fig. 5. For $m$-shot in-context learning, we obtain $m$ spans of Memory Tokens after demonstration compression and selection, denoted as $C^1$ to $C^m$. Then, we horizontally concatenate them, keeping their relative position unmodified. Finally, the concatenated Memory Tokens together with actual queries are fed into Vicuna, performing auto-regressive generation $g_\theta$ as normal:

$$y_i = g_\theta(C^1, ..., C^m; Q; y_{<i}). \quad (5)$$

Except for the generative manner, Memory Tokens apply close-ended evaluation for understanding tasks through measuring PPL[3] as normal, e.g. $(ppl^+, ppl^-)$ for sentiment classification:

$$y = \text{argmin}(ppl^+, ppl^-), \quad (6)$$

where choices with PPL closest to 1 is judged to be the current prediction.

### 3.4 Training

The trainable parameters in UniICL are merely 17M originating from the projection layer $W_p$ and the introduced Memory Slot [M]. The linear layer is optimized with the language modeling objective $\mathcal{L}_{lm}$ of Vicuna to learn a base compression model. Then InfoNCE (He et al., 2020) joint with language modeling objective are used to augment the demonstration selection ability of the base compression model:

$$\mathcal{L} = \mathcal{L}_{lm} + \mathcal{L}_{ctr}. \quad (7)$$

Specifically, we slice the source input of each training instance into two parts and randomly compress one. The compressed part is denoted as $x_c$ and the uncompressed part is denoted as $x_u$. Afterward, we attach the Memory Slot sequence $M$ behind $x_c$ and get Memory Tokens $C$ on top of the Memory Slots, as described in Eq. 1 and Eq. 2. Therefore, the language modeling loss $\mathcal{L}_{lm}$ is obtained as:

$$\mathcal{L}_{lm} = -\frac{1}{|y|}\sum_{t=0} logP(y_t|x_u; C; y_{<t}), \quad (8)$$

where $y$ is the reference label of the current training instance. Additionally, to approach the large-shot settings without significant truncation, we introduce concatenation compression. When $x_c$ exceeds

[3] https://huggingface.co/docs/transformers/perplexity

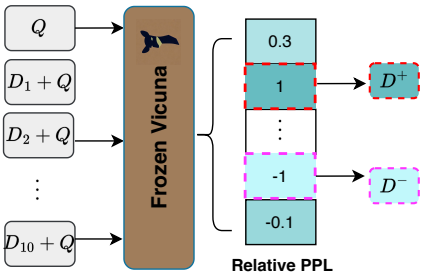


Figure 6: Contrastive examples mining pipeline. Finds demonstrations benefit/hinder the final generation according to the PPL.

the window limitation for compression, UniICL further divides $x_c$ into acceptable ranges and compresses them independently to get local Memory Tokens. Then, these Memory Tokens from different segments will be concatenated to formulate global virtual tokens to replace $x_c$, applying Eq. 8 to optimize models as well.

We obtained a base compression model that has learned to compress and understand concatenated Memory Tokens after the first-phase training mentioned. Subsequently, we utilize contrastive learning for selection augmentation and mine positives and negatives as illustrated in Fig. 6. Specifically, given each training instance $Q$ and $n$ candidate demonstrations $(D_1, D_2, ..., D_n)$ from two non-crossing training subsets, we employ Vicuna to calculate the PPL concerning the golden label of $Q$, denoted as $ppl^Q$ to find useful demonstrations for generation. Then, we provide the $i$-th demonstration and calculate PPL concerning the golden label of $Q$, denoted as $(ppl_i^D, i \in [1, n])$. We count $ppl^Q$ as the baseline and calculate candidate relative PPL gains:

$$\widetilde{ppl}_i^D = ppl^Q - ppl_i^D, i \in [1, n]. \quad (9)$$

After finding demonstrations $D^+$ ($D^-$) that furthest reduces (increases) $ppl^Q$, we obtain their representation $C_D^+$ ($C_D^-$) as processed in Eq. 3. The contrastive loss $\mathcal{L}_{ctr}$ can be formulated as:

$$\mathcal{L}_{ctr} = \frac{\exp(cos(C_Q, C_D^+))}{\exp(cos(C_Q, C_D^+)) + \exp(cos(C_Q, C_D^-))}. \quad (10)$$

In particular, if all relative PPL gains are less than 0, namely none of the candidate demonstrations help guide Vicuna to generate the golden label, we will apply the other set of candidates.

| Dataset | # words (96,512] | (512,1024] | (1024,1536] |
|---|---|---|---|
| XSum (Narayan et al., 2018) | - | 10,000 | 4,697 |
| CICERO (Ghosal et al., 2022) | 10,000 | - | - |
| SUPER-NI (Wang et al., 2022b) | - | 10,000 | 7,000 |
| XSum (Ctr) | | 5,000 | |

Table 1: The composition training set of UniICL. (m,n] represents the range of the number of words in each instance. XSum (Ctr) is used for the second-phase training in Eq. 7.

## 4 Experiment

### 4.1 Baselines

Unmodified Vicuna-7b serves as the fundamental baseline fed with actual demonstrations. AutoCompressor compresses prompts into 50 virtual tokens in different rounds recurrently as illustrated in Fig. 9(a). Previous compressed virtual tokens are put at the beginning of the current segment. Finally, virtual tokens of different compression rounds are concatenated for generation. We employ their Llama2-7b version for comparison. LLMLingua is a coarse-to-fine demonstration pruning method based on dropping uninformative words. We employ their released 7b version, of which the compressor is a fine-tuned Llama2. For a meaningful comparison, we replace target LLMs of LLMLingua (GPT-3.5-Turbo or Claude-v1.3) with the Vicuna-7b. ICAE compresses demonstrations into 128 virtual tokens via a LoRA-adapted Llama2-7b, as illustrated in Fig. 9(b). Additionally, since selection augmentation is involved in the training of UniICL, we utilize the popular Sentence-BERT (S-BERT) (Reimers and Gurevych, 2019) as the dense retriever to construct an ICL pipeline for the above methods, serving as simple but effective selection-based baselines.

### 4.2 Settings

We construct the training set by mixing up XSum, CICERO, and SUPER-NI according to their length as shown in Tab. 1 and evaluate UniICL on extensive out-of-domain datasets as listed in Tab. 2, with more details reported in Appendix H. Considering computation efficiency, we set the max allowed input length limit to 512 for both compression and generation for both training and inference. For a fair comparison, we set the allowed window of baselines to 512, and the compression ratio of default UniICL and baselines are set to 12, which is determined by the validation in Fig. 7. We fix the learning rate to 8e-5 and use Adam as the optimizer,

| Dataset | In-Domain | # Test | # Demonstrations |
|---|---|---|---|
| MS MARCO-dev | ✗ | 6,980 | - |
| XSum | ✓ | 1,500 | 204,045/20 |
| Arxiv | ✗ | 1,500 | 203,037/20 |
| CoLA-dev | ✗ | 1,041 | 67,349/20 |
| SST-2-dev | ✗ | 872 | 8,551/20 |
| IMDb | ✗ | 1,500 | 25,000/20 |
| MMLU | ✗ | 13,985 | 25,000/20 |

Table 2: The details of involved evaluation datasets. -dev represents employing development set due to their test sets are inaccessible. # Demonstrations represent the number of demonstrations to be selected in **high**/low-resource ICL settings.

and the effective batch size is 32 (8 GPUs data parallelism and 4 steps gradient accumulation). We train 10 epochs and 2 epochs respectively for the first- and second-phase training. The best checkpoints are selected according to their performance on in-domain validation sets. Additionally, we conducted all experiments on 8*NVIDIA A5000 24G GPUs based on BFloat 16 data type, and we set the evaluated shot to 8 for understanding tasks and 5 for generative tasks for illustration because of marginal ICL gains and memory costs.

We apply S-BERT to pre-rank and output the top 10 similar candidates from training sets according to each inference input for all baselines. UniICL is employed to perform selection among them in practice due to computation efficiency for high-resource ICL. On the contrary, the low-resource ICL setting utilizes the randomly sampled 20 candidate demonstrations for all inference inputs, while UniICL performs selection as normal.

To verify the universality, we further build UniICL on BlueLM-7B (Team, 2023) and Llama2-7B (Touvron et al., 2023). Results of BlueLM and Llama2 will be reported in Appendiex C and Appendiex D.

### 4.3 Results

We comprehensively evaluate the ICL performance of UniICL on the out-of-domain dataset CoLA, SST-2, and IMDb by close-ended evaluation and Arxiv by open-ended evaluation in Tab. 3. The details of involved evaluation datasets and metrics are reported in Tab. 2 and Appendix H. Specifically, UniICL outperforms unmodified Vicuna-7b fed with actual candidate demonstrations, which indicates that Memory Tokens are more efficient and informative for guiding the target LLM. Meanwhile, UniICL outperforms all the baselines by compressing the same demonstrations pre-ranked by S-BERT. Additionally, UniICL achieves further performance gains after selecting demonstrations via itself (UniICL♠). The open-ended results highlight that Memory Tokens indeed capture semantic information for ICL generation even though summarization demonstrations are much longer than understanding ones. Regarding Arxiv, the original ICL is not helpful enough due to its extremely over-length document, leaving little room for demonstrations. UniICL works as expected by compressing demonstrations into Memory Tokens and concatenating them, achieving +2.8 Rouge-1 gains in selection-augmented UniICL (+$\mathcal{L}_{ctr}$). Additionally, according to the results of +$\mathcal{L}_{ctr}$, we find that the gains brought by selection augmentation become larger with the number of demonstrations increasing. We attribute this to the fact that UniICL selects more useful demonstrations for generation after the second-phase training. The results of BlueLM are exhibited in Appendiex C. Except for understanding and generative tasks, we further evaluate UniICL on MMLU in Tab. 4. UniICL achieves stable performance gains with more demonstrations introduced. Additionally, considering ICAE and AutoCompressor are soft-prompt-based compression methods built on Llama2, we also build UniICL on Llama2 for ablation in Appendiex D.

**Passage Ranking** Since the virtual tokens naturally summarize semantic information of preceding sequences, we evaluate UniICL on the out-of-domain MS MARCO dataset in Tab. 5. UniICL significantly outperforms the sparse retrieval method BM25 algorithm and other compression methods. Subsequently, we fine-tune the first-phase compression model of UniICL on the training set of MS MARCO. UniICL achieves comparable performance with SIMLM (Wang et al., 2022a), which is specified in Information Retrieval (IR) and has more trainable parameters.

## 5 Analysis

### 5.1 Compression Ratio

During training, the compression ratio is dynamically sampled from 2 to 16. We mix up 2,000 instances from the in-domain validation set, 1,000 for XSum, and 1,000 for CICERO to select the compression ratio for UniICL in Fig. 7, with the backbone of Llama2, Vicuna, and BlueLM respectively. Specifically, UniICL compresses the latter cut-off

| Model | #-shots | CoLA-dev | SST-2-dev | IMDb | Arxiv | | | XSum | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Acc. | | | R-1 | R-2 | R-L | R-1 | R-2 | R-L |
| Vicuna | 0-shot | 56.2 | 91.7 | 92.6 | 34.3 | 9.1 | 27.4 | 19.9 | 5.0 | 13.5 |
| | 1-shot | 58.2 (57.4) | 90.7 (90.8) | 91.9 (91.0) | 34.4 (33.2) | 9.1 (8.5) | 27.5 (26.7) | 21.2 (20.4) | 5.8 (5.2) | 14.5 (13.9) |
| | 2-shot | 62.1 (59.8) | 92.1 (91.3) | 91.7 (91.7) | - | - | - | - | - | - |
| | 5-shot | 62.3 (61.9) | 93.0 (91.9) | 94.1 (92.5) | - | - | - | - | - | - |
| AutoCompressor | 1-shot | 42.1 (40.9) | 85.7 (84.2) | 95.0 (95.1) | 27.0 (26.4) | 8.4 (8.2) | 26.1 (25.8) | 21.3 (20.3) | 6.5 (6.3) | 13.7 (13.7) |
| | 2-shot | 58.8 (56.3) | 88.0 (86.4) | 95.0 (94.6) | 27.1 (26.2) | 8.6 (7.9) | 26.4 (25.4) | 21.9 (21.4) | 6.6 (6.4) | 14.5 (14.1) |
| | 5-shot | 59.1 (58.8) | 91.3 (89.1) | 94.7 (94.8) | 34.5 (33.7) | 9.4 (9.1) | 28.7 (27.9) | 22.4 (21.7) | 6.9 (6.7) | 14.8 (14.3) |
| LLMLingua | 1-shot | 55.5 (55.0) | 89.7 (89.6) | 91.0 (89.9) | 33.3 (33.1) | 8.9 (8.7) | 27.4 (27.1) | 20.5 (19.7) | 5.4 (5.2) | 14.5 (14.4) |
| | 2-shot | 56.7 (55.7) | 90.7 (90.2) | 91.3 (91.0) | 32.9 (32.0) | 8.2 (8.1) | 26.9 (25.9) | 20.3 (20.0) | 5.2 (5.1) | 14.3 (14.1) |
| | 5-shot | 57.2 (56.9) | 90.6 (90.2) | 90.9 (91.2) | 30.1 (29.7) | 7.9 (7.4) | 25.3 (24.6) | 19.7 (18.6) | 4.9 (4.9) | 14.1 (14.3) |
| ICAE | 1-shot | 30.9 (30.9) | 61.0 (60.1) | 85.7 (83.3) | 26.8 (24.6) | 8.2 (7.1) | 24.7 (22.9) | 23.5 (21.9) | 8.5 (7.8) | 20.9 (20.3) |
| | 2-shot | 30.9 (30.9) | 49.0 (52.8) | 85.9 (85.9) | 27.2 (25.5) | 8.4 (7.6) | 25.9 (24.3) | 24.4 (23.2) | 8.9 (8.4) | 21.3 (20.8) |
| | 5-shot | 30.9 (30.9) | 54.2 (51.0) | 85.7 (85.9) | 28.3 (26.9) | 8.7 (7.7) | 26.6 (25.8) | 25.3 (24.9) | 9.2 (8.8) | 22.5 (21.6) |
| UniICL | 1-shot | 58.7 (58.0) | 92.9 (91.7) | 94.3 (92.3) | 35.5 (34.7) | 10.5 (10.2) | 28.7 (27.9) | 27.7 (25.5) | 10.2 (9.1) | 21.2 (20.0) |
| | 2-shot | 62.4 (61.0) | 92.4 (91.6) | 94.9 (93.3) | 36.1 (35.2) | **10.8** (10.4) | 29.4 (28.2) | 29.4 (26.8) | 11.0 (9.8) | 22.3 (20.9) |
| | 5-shot | 62.6 (61.8) | 93.1 (92.3) | 94.5 (94.0) | 35.8 (35.4) | 10.6 (10.2) | 29.5 (28.1) | 30.7 (27.6) | 11.3 (10.1) | 22.8 (21.4) |
| UniICL♠ | 1-shot | 59.1 (58.7) | 93.0 (91.9) | 94.5 (91.6) | 34.8 (34.7) | 10.4 (10.3) | 28.1 (27.8) | 29.1 (26.2) | 10.8 (9.4) | 22.2 (20.7) |
| | 2-shot | 62.6 (61.2) | 94.0 (93.0) | 94.9 (92.3) | 34.6 (34.3) | 10.6 (10.4) | 28.5 (28.3) | 30.3 (28.9) | 11.3 (10.5) | 22.9 (21.7) |
| | 5-shot | 63.3 (61.5) | **94.7** (92.8) | 95.0 (93.8) | 35.6 (35.3) | 11.0 (10.8) | 29.1 (27.7) | 31.1 (30.0) | 11.7 (11.2) | 23.5 (22.3) |
| | 8-shot | 63.8 (62.6) | **94.7** (93.1) | 95.0 (94.2) | - | - | - | - | - | - |
| UniICL♠ + $L_{ctr}$ | 1-shot | 59.3 (58.9) | 93.2 (92.4) | 95.1 (92.8) | 35.6 (35.1) | 10.7 (10.5) | 28.9 (28.3) | 30.0 (27.9) | 11.3 (10.1) | 22.8 (21.5) |
| | 2-shot | 62.4 (62.0) | 94.5 (92.8) | 94.8 (93.4) | 36.8 (35.3) | 10.8 (10.6) | 29.6 (28.9) | 30.8 (29.2) | 11.4 (10.7) | 23.0 (21.9) |
| | 5-shot | 64.3 (61.8) | **94.7** (93.4) | **96.1** (94.2) | **37.1** (34.9) | **11.3** (11.2) | **30.0** (29.3) | **32.5** (30.6) | **12.3** (11.8) | **24.7** (23.8) |
| | 8-shot | **64.7** (63.3) | **94.7** (94.1) | 95.6 (95.0) | - | - | - | - | - | - |

Table 3: The high- and low-ICL results on CoLA-dev, SST-2-dev, and IMDb. Results in (bracket) represent low-resource ICL. ♠ represents the demonstrations selected by UniICL, and the others are selected by S-BERT. +$L_{ctr}$ indicates the selection augmented UniICL (optimized with Eq. 7). **Bold** (underline) represents the best performance on high- and low-resource ICL. R- indicates Rouge scores. All compression methods are evaluated with a compression ratio set to 12.

| #-Shots | S | H | SS | O | Avg. |
|---|---|---|---|---|---|
| 0-shot | 36.9 | 53.2 | 53.7 | 50.7 | 48.6 |
| 1-shot | 38.6 | 55.3 | 54.6 | 52.4 | 50.2 |
| 2-shot | 39.2 | **55.8** | **55.3** | 53.1 | 50.9 |
| 5-shot | **40.1** | 55.6 | **55.3** | **53.8** | **51.2** |

Table 4: Performance of UniICL on MMLU benchmark. We reported the Accuracy at the category level. S represents **S**TEM, H represents **H**umanities, SS represents **S**ocial **S**cience, O represents **O**ther, and Avg indicates their average performance.

| Method | # TP | MRR@10 |
|---|---|---|
| BM25† | - | 18.5 |
| Vicuna | - | 28.9 |
| AutoCompressor | - | 29.3 |
| ICAE | - | 30.2 |
| UniICL | - | **31.6** |
| SIMLM†‡ | 110M | 41.1 |
| UniICL‡ | 17M | 38.9 |

Table 5: MRR@10 results on MS MARCO. Vicuna applies the last hidden states of [EOS] to represent sentences in latent space. Results citing from Liang (Wang et al., 2022a) are denoted as †, and methods supervised trained on MS MARCO are represented as ‡. **Bold** indicates the best zero-shot performance and Underline is the best fine-tuned results. # TP indicates the number of trainable parameters.

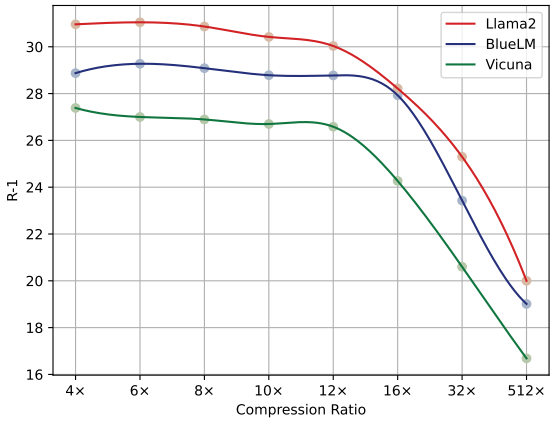


Figure 7: The compression ratio sensitivity analysis of Llama2 , BlueLM, and Vicuna.

part while keeping the former ones uncompressed. Therefore, we can measure the dense information quality of the same content with different compression ratios by ROUGE-1 since it is more sensitive to token-level differences. The performance is relative smoothing when the compression ratio changes from 4× to 12×. However, when it comes to 16×, an obvious drop occurs. In order to analyze this phenomenon more deeply, we provide a thorough analysis in Appendix G. Therefore, we set the compression ratio to 12 by default and apply this ratio

| #-shots | CoLA | SST-2 | IMDb | Arxiv |
|---|---|---|---|---|
| | | Acc. | | R-1 |
| 1-shot | 58.5 (-0.8) | 91.4 (-1.8) | 92.6 (-2.5) | 34.8 (-0.8) |
| 2-shot | 59.7 (-2.7) | 92.1 (-2.4) | 94.1 (-0.7) | 35.7 (-1.1) |
| 5-shot | 62.4 (-1.9) | 93.1 (-1.6) | 94.8 (-1.3) | 36.6 (-0.5) |

Table 6: Performance of UniICL on out-of-domain datasets, with a fixed compression ratio set to 12 during training.

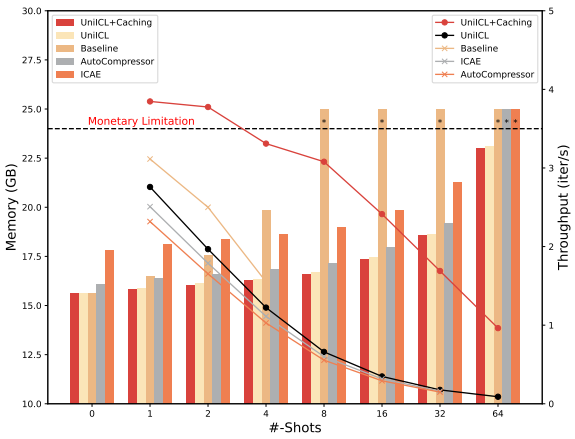


Figure 8: The efficiency comparison between UniICL and other compression methods in CoLA with the number of shots increasing from 0 to 64. Memory explodes are represented as *, corresponding to the break of the line chart. +Caching represents using DB.

to all experiments. The 512× compression ratio is equal to compressing anything to a single virtual token, due to the maximum allowed input length for compression being 512.

To explore whether it could yield additional performance gains compared with dynamic ratios, in Tab. 6, we re-train UniICL with the compression ratio fixed to 12 (Results of more fixed ratios are reported in Appendix F.). Results indicate that UniICL trained with fixed compression ratios underperforms in out-of-domain datasets as it exhibits over-fitting in in-domain sets as shown in Tab. 12.

Furthermore, we analyze whether 12× is suitable for all out-of-domain datasets in Fig. 10 in Appendix E. Results indicate that 12× outperforms other compression ratios in general across 4 out-of-domain datasets. It also points out that lower ratios still work comparable for short demonstrations and higher ratios are suitable for long demonstrations to some extent.

### 5.2 Efficiency Analysis

In UniICL, we incorporate an additional 17M trainable parameters into the 7b backbone, accounting for an approximate increase of 0.24%. We evaluate the memory costs inference latency of UniICL and other compression methods in Fig. 8. With the help of the Demonstration Bank (DB), UniICL will eliminate the extra latency if the selected demonstrations have been compressed and cached (UniICL+Caching). Despite this, parallel computation facilitates the compressing process, resulting in minimal throughput degradation (UniICL and Baseline). The unmodified 7B LLM occurs memory explosion for 8-shot settings and other compression methods perform up to 32-shot, while UniICL successfully scales up to 64-shots within 24GB CUDA allocation.

| Method | GPUHours | TFLOPs | TMACs |
|---|---|---|---|
| Vicuna | 1.5 | 86,20 | 4,309 |
| Vicuna-1k | 1.9 | 31,664 | 15,832 |
| UniICL | 1.6 | 22,437 | 11,218 |

Table 7: The computation efficiency of UniICL.

Additionally, We demonstrate the inference computation and GPU hours in Tab. 7, by using 1,024 random legal tokens as inputs and forcing models to generate 128 tokens. Notably, UniICL (without DB) compresses the former half, and the latter half is fed into the generator directly, while Vicuna and Vicuna-1k are distinguished in window limitations. Results indicate that minimal GPU hours increased due to the parallel computation of forward, although the extra compression of UniICL surges the computation. Additionally, Vicuna with a 1k window limitation surges both GPU hours and TFLOPs because long input brings significant computation and latency in generation.

## 6 Conclusion

This paper proposes UniICL, a parameter-efficient ICL framework that unifies demonstration selection, demonstration compression, and final response generation via a frozen LLM, an adapter, and a learnable embedding. Experimental results prove the advantages of UniICL in both efficiency and effectiveness. Due to 12× demonstration compression, UniICL scales up the number of demonstrations from 4 to 64 within 24 GB VRAM allocation. Finally, to avoid repeated compression of the same demonstration, UniICL configures a Demonstration Bank (DB, which significantly boosts model efficiency.

## 7 Limitations

Our study, while proposing an efficient unified ICL framework for demonstration compression and selection, still has limitations. Firstly, UniICL is limited to the realm of unmodified ICL leaving other advanced LLM prompting methods, e.g. Retrieval Augment Generation (RAG) and Chain-of-Thought (CoT) unexplored. Limited to the hardware, we deploy the underlying LLM at a scale of 7 billion parameters. Larger-scale LLMs are welcome to enrich our findings in future studies.

## References


Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. 2022. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2022. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Aydar Bulatov, Yuri Kuratov, and Mikhail S Burtsev. 2023. Scaling transformer to 1m tokens and beyond with rmt. *arXiv preprint arXiv:2304.11062*.

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

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

Jun Gao, Ziqiang Cao, Shaoyao Huang, Luozheng Qin, and Chunhui Ai. 2024. Guiding chatgpt to generate salient domain summaries. *arXiv preprint arXiv:2406.01070*.

Tao Ge, Jing Hu, Xun Wang, Si-Qing Chen, and Furu Wei. 2023. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*.

Deepanway Ghosal, Siqi Shen, Navonil Majumder, Rada Mihalcea, and Soujanya Poria. 2022. Cicero: A dataset for contextualized commonsense inference in dialogues. *arXiv preprint arXiv:2203.13926*.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*.

Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Woosuk Kwon, Joseph Hassoun, and Kurt Keutzer. 2022. Learned token pruning for transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 784–794.

Yucheng Li. 2023. Unlocking context constraints of llms: Enhancing context efficiency of llms with self-information-based content filtering. *arXiv preprint arXiv:2304.12102*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Noisy channel language model prompting for few-shot text classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5316–5330.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. *arXiv preprint arXiv:2304.08467*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *ArXiv*, abs/1808.08745.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *choice*, 2640:660.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *arXiv preprint arXiv:2302.00083*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

BlueLM Team. 2023. Bluelm: An open multilingual 7b language model. https://github.com/vivo-ai-lab/BlueLM.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

Jiaan Wang, Yunlong Liang, Fandong Meng, Haoxiang Shi, Zhixu Li, Jinan Xu, Jianfeng Qu, and Jie Zhou. 2023a. Is chatgpt a good nlg evaluator? a preliminary study. *arXiv preprint arXiv:2303.04048*.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023b. Label words are anchors: An information flow perspective for understanding in-context learning. *arXiv preprint arXiv:2305.14160*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022a. Simlm: Pre-training with representation bottleneck for dense passage retrieval. *arXiv preprint arXiv:2207.02578*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Large search model: Redefining search stack in the era of llms. In *ACM SIGIR Forum*, volume 57, pages 1–16. ACM New York, NY, USA.

Liang Wang, Nan Yang, and Furu Wei. 2023c. Learning to retrieve in-context examples for large language models. *arXiv preprint arXiv:2307.07164*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022b. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*.

Zengzhi Wang, Qiming Xie, Zixiang Ding, Yi Feng, and Rui Xia. 2023d. Is chatgpt a good sentiment analyzer? a preliminary study. *arXiv preprint arXiv:2304.04339*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2018. Neural network acceptability judgments. *arXiv preprint 1805.12471*.

Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, et al. 2023. Zero-shot information extraction via chatting with chatgpt. *arXiv preprint arXiv:2302.10205*.

David Wingate, Mohammad Shoeybi, and Taylor Sorensen. 2022. Prompt compression and contrastive conditioning for controllability and toxicity reduction in language models. *arXiv preprint arXiv:2210.03162*.

Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. 2022. Memorizing transformers. *arXiv preprint arXiv:2203.08913*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*.

Xianjun Yang, Yan Li, Xinlu Zhang, Haifeng Chen, and Wei Cheng. 2023. Exploring the limits of chatgpt for query or aspect-based text summarization. *arXiv preprint arXiv:2302.08081*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

Lin Zheng, Chong Wang, and Lingpeng Kong. 2022. Linear complexity randomized self-attention mechanism. In *International conference on machine learning*, pages 27011–27041. PMLR.

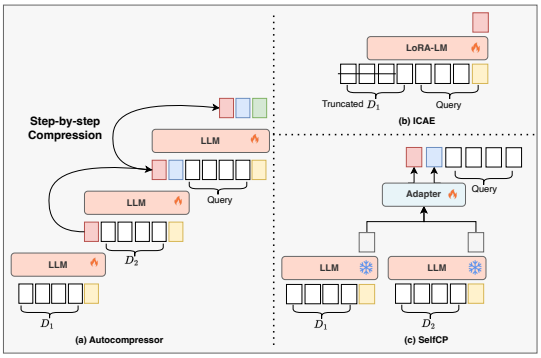


Figure 9: Differences of compression methods in formulating compressed virtual tokens in ICL. White blocks indicate the original embeddings. Yellow blocks are the compressing embedding for each method. Other colors represent virtual tokens used for generation.

| Methods | Additional Compressor | Compression Tool | # Trainable Parameters | Train Size |
|---|---|---|---|---|
| LLMLingua (Jiang et al., 2023) | YES | Pruning | 7B | 57k |
| AutoCompressor (Wingate et al., 2022) | NO | Soft Prompt | 7B | UNKNOWN |
| ICAE (Ge et al., 2023) | YES | Soft Prompt | 70M | 240k |
| UniICL | NO | Soft Prompt | 17M | 47k |

Table 8: Comparsion among recent compression methods and UniICL. Compression Tool represents the involved compression technique of different methods. Train Size represents the size of training datasets.

## A Comparsion with Existing Compression Methods

We present a comparison of training costs between UniICL and other recent compression methods in Tab. 8. Additionally, we illustrate differences in formulating virtual tokens for compression methods based on the soft prompt in Fig. 9.

To explain plainly, we ideally assume the compressor within three compression methods based on soft prompts has the window limitation of $L$, and has the same compression ratio, ignoring the length of soft prompts. In the 2-shot scenario, queries, demonstrations $D_1$ and $D_2$, each have a length of $L$. As shown in Fig. 9(a), AutoCompressor divides the concatenated demonstrations back into three segments, and then compresses each segment step-by-step, bringing three times non-parallel compression. When it comes to ICAE, merely part of $D_1$ is accessible for the compressor and others will be read by no means as illustrated in Fig. 9(b). AutoCompressor shows advantages in the readable prompt length but is short in efficiency due to step-by-step compression. ICAE has a constant compression complexity but struggles to approach relatively long inputs. Combining the advantages of AutoCompressor and ICAE, UniICL compresses $D_1$ and $D_2$ into Memory Tokens independently, utilizing an adapter to perform alignment between the compressor and the generator.

Generally, in the $N$-shot settings, the number of practical compression steps can be calculated as $\lceil \frac{N}{m} \rceil$, where $m$ indicates that a single GPU is capable of compressing $m$ demonstrations in a batch. When the GPU capacity is sufficient, $m$ equals $N$, which is the scenario of ICAE that compresses all segments in a time but UniICL drops nothing, while it degenerates to the AutoCompressor scenario that compresses segments step-by-step, when the GPU capacity is only sufficient to set $m = 1$.

## B In-Domain Evaluation

| Backbone | Method | XSum | | | CICERO | | |
|---|---|---|---|---|---|---|---|
| | | R-1 | R-2 | R-L | R-1 | R-2 | R-L |
| Vicuna-7b | Vicuna | 19.9 | 5.0 | 13.5 | 17.3 | 3.3 | 14.3 |
| | +*LoRA* | 25.4 | 7.5 | 17.3 | 28.1 | 10.5 | 25.6 |
| | Vicuna-1k | 27.3 | 8.7 | 19.7 | 30.5 | 11.3 | 27.4 |
| | +*LoRA* | 31.2 | 11.0 | 23.1 | 34.1 | 13.5 | 30.2 |
| | UniICL | 30.0 | 10.2 | 22.3 | 32.6 | 12.2 | 28.8 |
| BlueLM-7b | BlueLM | 15.0 | 3.6 | 10.4 | 17.6 | 3.1 | 15.0 |
| | +*LoRA* | 23.1 | 7.6 | 17.4 | 21.9 | 7.8 | 19.8 |
| | BlueLM-1k | 28.1 | 9.9 | 22.8 | 25.1 | 9.2 | 23.1 |
| | +*LoRA* | 30.8 | 10.5 | 24.6 | 31.2 | 10.8 | 27.4 |
| | UniICL | 30.4 | 10.2 | 23.7 | 29.2 | 10.0 | 26.6 |

Table 9: The in-domain results and ablation studies on XSum and CICERO. 1k represents the extending 1k window limitation, while others have a limitation of 512.

We conduct the zero-shot in-domain generation evaluation on the entire test set of XSum and CICERO in Tab. 9 by compressing the latter half to virtual tokens and keeping the former unmodified. UniICL significantly outperforms the baselines, indicating the compressed virtual tokens can provide the original truncated information by recovering the cut-off parts after supervised fine-tuning. Although extending the window to 1k, Vicuna and BlueLM still underperform UniICL, indicating that compressed virtual tokens filter noise information to some extent.

Additionally, to quantify the performance gains brought by the learnable projection layer. We tune Vicuna and BlueLM with comparable parameters (17M) with LoRA, setting the rank to 32 in Tab. 9. UniICL still outperforms LoRA-adapted LLMs with a 512 window limitation, indicating that the truncation indeed brings performance degradation.

## C Results on BlueLM

We extra conduct experiments on BlueLM (Team, 2023) to verify the generality of UniICL. We demonstrate the result of understanding tasks in Tab. 10, of the generative tasks in Tab. 11.

| Model | #-shots | CoLA-dev | SST-2-dev | IMDb |
|---|---|---|---|---|
| | | | Acc. | |
| BlueLM | 0-shot | 71.6 | 81.2 | 48.8 |
| | 1-shot | 69.6 | 82.6 | 64.8 |
| | 2-shot | 70.0 | 87.0 | 65.6 |
| | 5-shot | 70.5 | 88.6 | 68.7 |
| UniICL | 1-shot | 69.6 | 81.2 | 65.4 |
| | 2-shot | 68.7 | 82.6 | 67.0 |
| | 5-shot | 71.7 | 87.0 | 70.4 |
| UniICL♠ | 1-shot | 69.8 | 80.0 | 62.0 |
| | 2-shot | 70.1 | 80.8 | 67.0 |
| | 5-shot | 71.8 | 85.6 | 69.6 |
| | 8-shot | 72.3 | 87.4 | 69.4 |
| UniICL♠ + $L_{ctr}$ | 1-shot | 70.1 | 80 | 69.6 |
| | 2-shot | 70.3 | 87.2 | 70.6 |
| | 5-shot | 71.1 | 89.2 | 71.0 |
| | 8-shot | 72.5 | 90.4 | 76.8 |

Table 10: The ICL results of understanding tasks with the backbone of BlueLM.

| Method | #-shots | XSum | | | Arxiv | | |
|---|---|---|---|---|---|---|---|
| | | R-1 | R-2 | R-L | R-1 | R-2 | R-L |
| BlueLM | 0-shot | 15.0 | 3.6 | 10.4 | 30.9 | 7.7 | 24.7 |
| | 1-shot | 19.1 | 4.8 | 12.1 | 23.0 | 3.6 | 19.0 |
| UniICL | 1-shot | 24.0 | 6.9 | 18.0 | 31.4 | 7.7 | 25.2 |
| | 2-shot | 25.0 | 7.3 | 18.8 | 30.8 | 7.3 | 24.8 |
| | 5-shot | 25.3 | 7.4 | 19.1 | 31.9 | 7.8 | 26.0 |
| UniICL♠ | 1-shot | 25.2 | 7.4 | 18.9 | 31.6 | 7.9 | 25.4 |
| | 2-shot | 25.4 | 7.6 | 19.1 | 31.9 | 8.0 | 25.6 |
| | 5-shot | 26.5 | 7.9 | 20.3 | 32.1 | 8.0 | 25.5 |
| UniICL♠ + $L_{ctr}$ | 1-shot | 24.7 | 7.2 | 18.5 | 31.0 | 7.5 | 24.9 |
| | 2-shot | 25.1 | 7.4 | 19.0 | 31.2 | 7.7 | 25.1 |
| | 5-shot | 26.3 | 7.6 | 20.0 | 31.5 | 7.9 | 25.3 |

Table 11: The ICL results of generative tasks with the backbone of BlueLM.

## D Supplementary Ablation on Llama2

AutoCompressor (Wingate et al., 2022) and ICAE (Ge et al., 2023) are built on Llama2-7B (Touvron et al., 2023), which are soft-prompt-based methods similar to UniICL. Therefore, we evaluate UniICL with Llama2 as the backbone. As shown in Tab 12 and Tab. 13, UniICL achieves substantial improvements compared with unmodified Llama2 and outperforms ICAE and AutoCompressor demonstrated in Tab. 3.

| Model | #-shots | CoLA-dev | SST-2-dev | IMDb |
|---|---|---|---|---|
| | | | Acc. | |
| Llama2 | 0-shot | 73.4 | 93.0 | 85.3 |
| | 1-shot | 74.8 | 94.0 | 85.5 |
| | 2-shot | 75.6 | 94.9 | 87.8 |
| | 5-shot | 84.3 | 97.2 | 92.7 |
| UniICL | 1-shot | 74.9 | 94.1 | 94.6 |
| | 2-shot | 75.9 | 95.1 | 96.1 |
| | 5-shot | 85.4 | 95.7 | 96.5 |

Table 12: The ICL results of understanding tasks with the backbone of Llama2.

| Method | #-shots | XSum | | | Arxiv | | |
|---|---|---|---|---|---|---|---|
| | | R-1 | R-2 | R-L | R-1 | R-2 | R-L |
| Llama2 | 0-shot | 27.4 | 7.6 | 20.1 | 32.9 | 8.9 | 29.2 |
| | 1-shot | 27.7 | 7.9 | 20.3 | 30.1 | 8.0 | 28.4 |
| UniICL | 1-shot | 27.8 | 8.0 | 20.5 | 33.3 | 9.2 | 29.7 |
| | 2-shot | 28.4 | 8.6 | 21.3 | 34.0 | 9.4 | 30.3 |
| | 5-shot | 29.3 | 9.1 | 22.0 | 34.5 | 9.7 | 30.8 |

Table 13: The ICL results of generative tasks with the backbone of LLama2.

## E Compression Ratio Selection on Different Tasks

We illustrate suitable ratio selection across four out-of-domain datasets in Fig. 10. For tasks with relatively short inputs, such as CoLA and SST2, UniICL tends to perform better with a compression ratio set to 4. While in IMDb and Arxiv, which are longer, UniICL performs better with higher compression ratios. UniICL with a 12× compression ratio substantially outperforms other settings on four datasets. Additionally, we are curious about if it is necessary to introduce more demonstrations with a higher compression ratio. In Fig. 11, we find

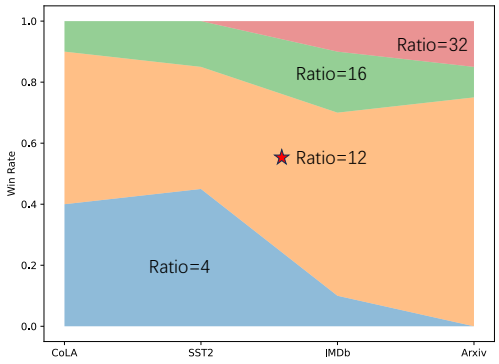


Figure 10: Winrate of different compression ratios on out-of-domain evaluation in 1-shot settings.

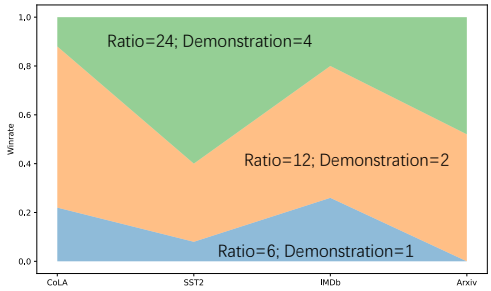


Figure 11: Winrate of UniICL with a fixed number of Memory Tokens.

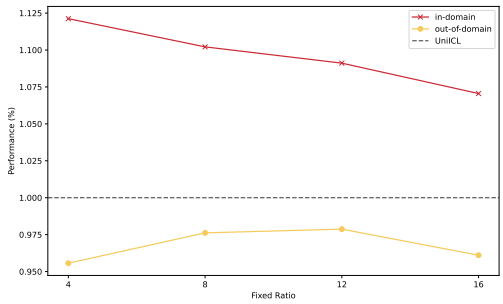


Figure 12: The relative performance on in-domain and out-of-domain datasets, with UniICL trained with a fixed ratio. Out-of-domain evaluation applies 1-shot settings.

that the performance of compressing 2 demonstrations with 12× ratio is stable and outperforms other settings across 3/4 datasets. 6× compression ratio with 1 demonstration compressed performs worst in general. When compressing 4 demonstrations with a 24× ratio, its performance is comparable and it slightly outperforms the 12× ratio in SST2.

## F Fixed Compression Ratio Training

To verify the effectiveness of the dynamic sampled compression ratio of UniICL, we train models with more extensive fixed compression ratios and perform out-of-domain evaluation with the same ratio in Fig. 12. Results indicate that fixed compression ratios work better than dynamic sampled ratios in in-domain evaluation, but underperform in out-of-domain evaluation. We attribute this to the fixed compression ratio makes models exhibit overfitting during training, and demonstration compression degrades to Prefix Tuning.

## G Visualization of Memory Tokens

To explore how Memory Tokens work within UniICL across different compression ratios, we visualize the cosine similarity between Memory Tokens and original embeddings in Fig. 13 and attention scores of the first generation step in Fig. 14.

Intuitively, the 4× compression ratio should retain more information due to more Memory Tokens. However, as shown in Fig. 13(a), the cosine similarity is relatively sparser than the 4× compression ratio illustrated in Fig. 13(b). This tendency is aligned with the first step attention scores in Fig. 14(a). According to merely 0.3% average attention occupied in a generation, we can conclude that more Memory Tokens fail to provide models with more information. We attribute this phenomenon to the given semantic information being distributed over all Memory Tokens as models attend to each Memory Token equally in Fig. 14(a). Fewer Memory Tokens are enough to concentrate this information, represented as relatively concentrated similarity distribution in Fig. 13(b) and higher attention scores in Fig. 14(b), both of each indicates denser information retained. When the compression ratio becomes higher, such as 16 or 32, Memory Tokens become fewer and therefore sparse information retrained as shown in Fig. 13(c), Fig. 13(c), Fig. 14(c), and Fig. 14(c). This also provides an explanation for the slow performance degradation with ratios varying from 4 to 12 and drops sharply at 16 in Fig. 7.

## H Datasets & Metrics

**Datasets** We mix up three public datasets for compression and selection augmentation training, described in Tab. 1. The training set includes an instruction dataset SUPER-NI, which we used to make UniICL respond to various instructions. Notably, we don't perform an in-domain evaluation on SUPER-NI as it only contains a training set. After training, we extensively evaluate UniICL on out-of-domain evaluation, involving text summarization (Narayan et al., 2018), passage ranking (Nguyen et al., 2016), sentiment classification (Maas et al., 2011; Socher et al., 2013), linguistic acceptability (Warstadt et al., 2018), and a popular reasoning benchmark (Hendrycks et al., 2020), more details referring to Tab. 2. MS MARCO is popularly used in Information Retrieval (IR), we use this dataset to evaluate the ability of UniICL to capture document-level information. Specifi-

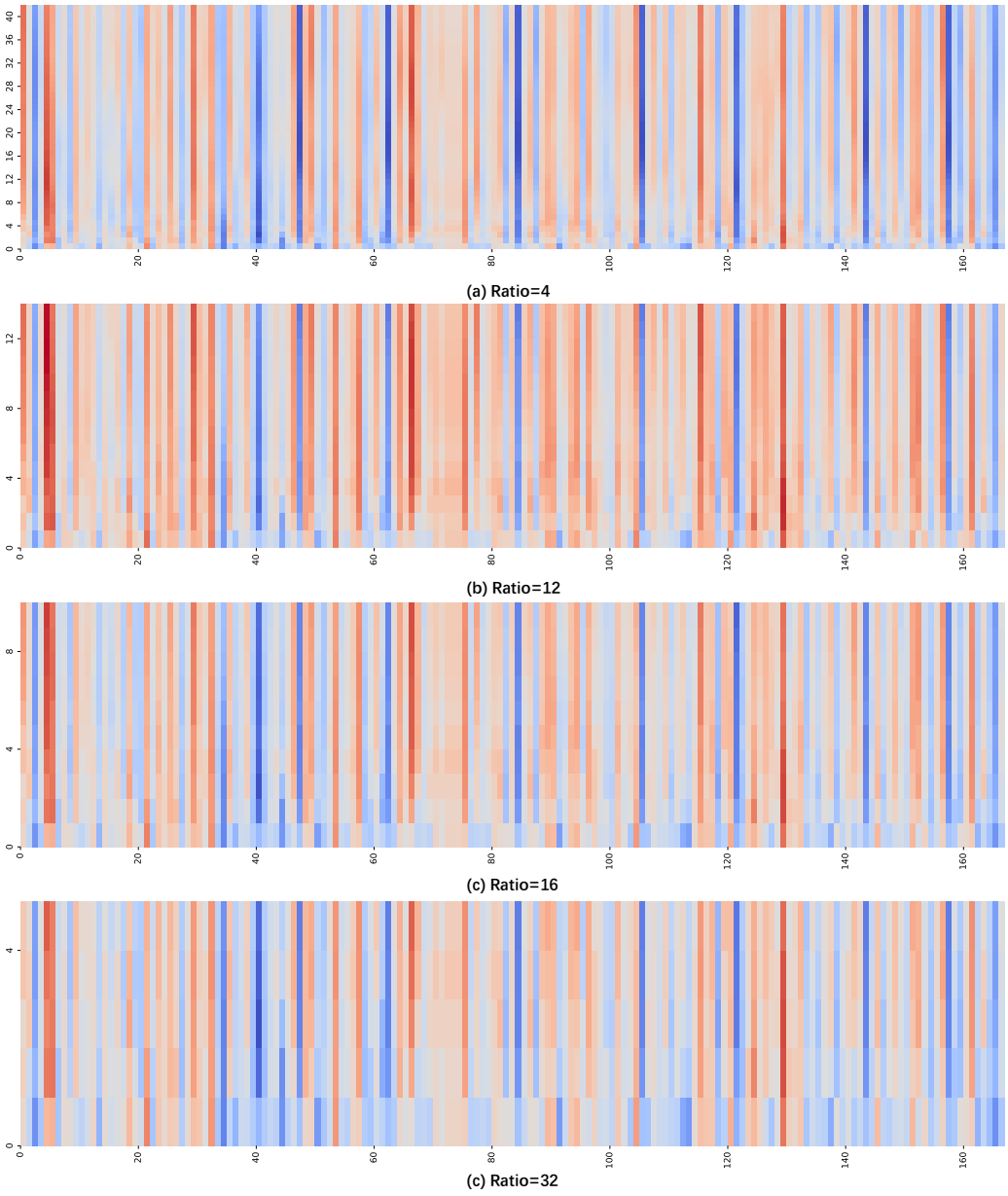


Figure 13: Cosine similarity between Memory Tokens (vertical axis) and original embeddings (horizon axis).

cally, MMLU (Massive Multitask Language Understanding) is a comprehensive evaluation benchmark including 57 subjects from STEM, Humanities, Social Sciences, and Other fields. We use this benchmark to evaluate the reasoning ability of UniICL. UniICL selects demonstrations from its training set in high-resource ICL, and we fixed the number of candidate demonstrations to 20 for low-resource ICL evaluation.

**Evaluation Metrics** ROUGE (Lin, 2004) is a widely adopted metric in many generative tasks that evaluate how similar the generated hypothesis is to the golden label. Therefore, ROUGE is used in our experiments to evaluate the quality responses generated conditioned on compressed virtual tokens, and we report the F-1 scores of ROUGE-1, ROUGE-2, and ROUGE-L (abbreviated R-1, R-2, R-L in the following), and we employed the files2rouge [4] library in practice. Following the previous works, we report the accuracy of close-ended evaluation and MRR@10 for passage ranking.

[4] https://github.com/pltrdy/files2rouge.

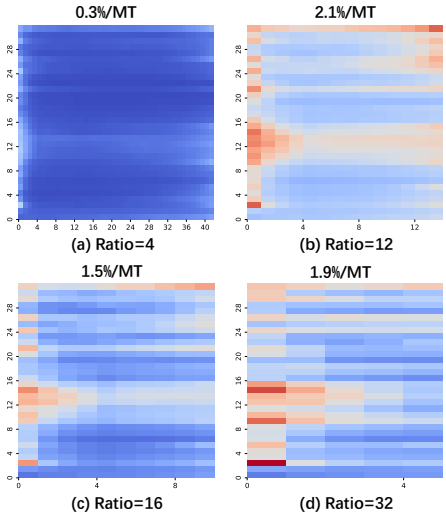


Figure 14: Attention scores on Memory Tokens in the first step generation. The vertical axis describes the 32 LLM layer, and the horizon axis indicates the number of Memory Tokens across different compression ratios. Above each figure, **%/MT** represents the average proportion of the attention score occupied by memory tokens in each LLM layer.