# Long-Context Language Modeling with Parallel Context Encoding

**Anonymous ACL submission**

## Abstract

Extending large language models (LLMs) to process longer inputs is crucial for numerous applications. However, the considerable computational cost of transformers, coupled with limited generalization of positional encoding, restricts the size of their context window. We introduce **C**ross-**A**ttention to **P**arallel **E**ncodings (CAPE), a framework that can be applied to any existing decoder-only LLMs for context expansion. CAPE leverages a small encoder to process a long input chunk by chunk and enables the frozen decoder to cross-attend to the additional contexts. CAPE is efficient, generalizable, and versatile: trained with 8K-token documents, CAPE extends the context window of LLAMA-2 to 128K tokens, offering $10\times$ of the throughput with only 1/6 of the memory. CAPE yields strong performance on language modeling and in-context learning. CAPE also excels in retrieval-augmented applications, while existing long-context models degenerate with retrieved contexts. We further introduce a CAPE variant that can extend the context window of instruction-tuned models with only unlabeled data, and showcase its effectiveness on LLAMA-2-CHAT, leading to a strong instruction-following model that can leverage very long context on downstream tasks.

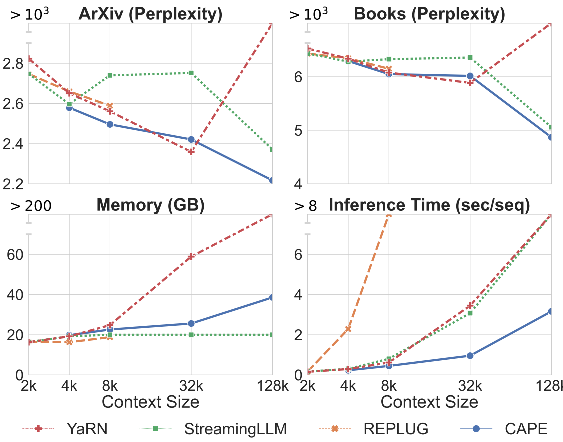


Figure 1: A comparison between CAPE and other techniques of extending LLMs' context window, including YARN (Peng et al., 2023), STREAMINGLLM (Xiao et al., 2023), and REPLUG (Shi et al., 2023b). CAPE trained on 8K tokens can generalize to 128K tokens with minimal extra computational and memory costs.

## 1 Introduction

Long and extensible context is crucial for large language models (LLMs) to effectively perform complex tasks, such as summarizing a book or answering questions with hundreds of retrieved webpages. However, there are several challenges that limit the ability of LLMs to leverage long context: (1) LLMs and popular positional encodings (Su et al., 2021; Peng et al., 2023) can not generalize to sequence lengths longer than the lengths seen during training (Press et al., 2022), even after additional fine-tuning (Computer, 2023; Chen et al., 2023a,b; Peng et al., 2023, *inter alia*). (2) Transformers—the predominant architecture of LLMs—incur a quadratic computational cost and a linear memory cost with respect to the input length, making it prohibitive to use for long sequences. (3) High quality long-context data, such as long instruction-following data, are scarce and difficult to obtain (Wang et al., 2023; Xiong et al., 2023).

A series of inference-time modification methods have been proposed recently to scale up the effective context window, either by modifying the attention mechanism (Bertsch et al., 2023; Xiao et al., 2023; Ivgi et al., 2023) or encoding chunks of context in separate forward passes (Shi et al., 2023b; Ratner et al., 2023; Lin et al., 2023). While these methods generalize to longer sequences, the model often fails to effectively leverage the extra tokens and can result in greater inference costs.

In this work, we propose an efficient and lightweight solution to extending the context window of LLMs, called **C**ross-**A**ttention to **P**arallel

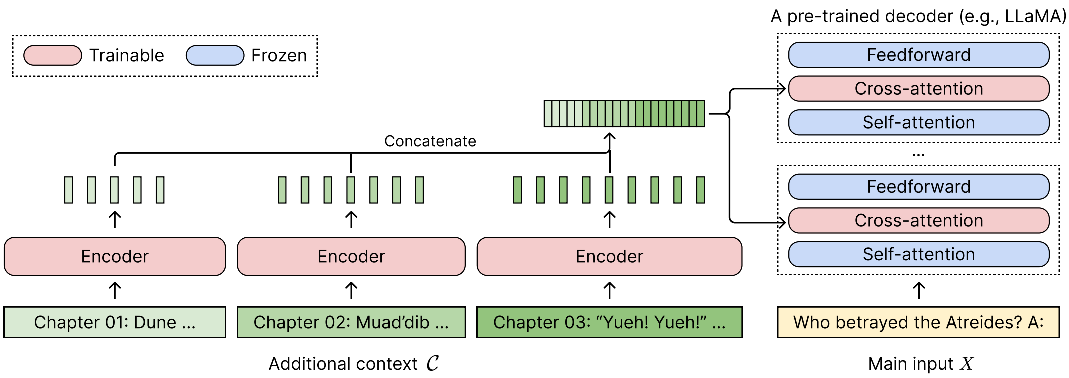


Figure 2: CAPE architecture. The encoder model encodes the additional 3 chunks ($k = 3$) of context $\mathcal{C}$ in parallel, and the final hidden representations from the encoder model are concatenated and used as inputs to the cross-attention layers in the decoder model. The cross-attention layers attend to the encoder representations between the self-attention and feed-forward layers in the decoder model.

Encodings (CAPE). CAPE is applicable to any pre-trained decoder-only LM by adding two components: a small encoder that encodes the long context in chunks, and cross attention modules inserted in each layer of the decoder to attend to the encoder representations (Figure 2). By careful selection of unlabeled training data, CAPE can be trained to leverage long-context documents effectively, and multiple retrieved contexts flexibly for downstream use.

CAPE offers several benefits: (1) **Length generalization**. CAPE is not limited by positional encoding constraints as the long context is encoded in segments, each with its own positional encoding. (2) **Efficiency.** Using a small encoder and processing context in parallel reduce computational cost. Since cross attention only attends to the last layer's representations from the encoder, CAPE requires much less memory compared to decoder-only LMs, which cache the key-value pairs of every token in every layer. (3) **Reduced training cost**. Unlike full fine-tuning approaches, we only tune the encoder and the cross attention while keeping the large decoder LM frozen; augmenting a 7B decoder with a 400M encoder and cross-attention layers can be done with a single A100 80 GB GPU.

We apply CAPE to LLaMA-2 and train it on a filtered version of RedPajama (Together, 2023) for 20B tokens. We first show that CAPE-LLaMA-2, trained with 8K input length, continues to decrease in perplexity on longer input up to 128K tokens. Then, we apply CAPE to a retrieval-augmented setting, as our larger context window allows incorporating more retrieved documents. Compared to existing methods, CAPE achieves better performance on both retrieval-augmented language modeling and open-domain question answering. Additionally, we also demonstrate that CAPE can effectively leverage more demonstrations for in-context learning (Brown et al., 2020). All the above is achieved with a much lower memory and computational cost than previous solutions.

Finally, we propose CAPE-Distilled (CAPED), which extends the context window of *instruction-tuned* models, using only unlabeled data. CAPED distills the behavior of the original instruction-tuned model to the new architecture through an auxiliary KL divergence loss, which eliminates the need to curate expensive long-context instruction-following data. We apply CAPED to LLaMA-2-Chat (Touvron et al., 2023b) models and show that while preserving their instruction understanding ability, CAPED-LLaMA-2-Chat can incorporate more context and improve performance on long-text understanding tasks (Shaham et al., 2023).

To conclude, CAPE is a lightweight framework that can extend context windows of any pre-trained and instruction-tuned LMs. We hope CAPE can empower future LLM research with cheap and effective long-context abilities.

## 2 Our Method: CAPE

We design CAPE to efficiently adapt pre-trained LLMs to perform long-context language modeling on sequences with many tokens (e.g., books). In the retrieval-augmented setting, these long contexts can contain a set of retrieved passages and a instruction or question. We first describe how CAPE modifies the architecture of LLMs to attend to representations encoded by a small encoder. Then, we describe how the added parameters are trained.

Finally, we extend CAPE to formulate CAPED, which can modify instruction-tuned models to attend to additional context with only unlabeled data.

### 2.1 Architecture

CAPE augments off-the-shelf decoder-only language models by (1) adding a small encoder model and (2) inserting cross-attention layers between the self-attention and feed-forward layers in every transformer block of the decoder model.

**Notation.** Given an input context with $T$ tokens $x_1, ..., x_T$, we consider the first $m$ tokens $x_1, ..., x_m$ as the additional context $\mathcal{C}$ and the last $n$ tokens $x_{m+1}, ..., x_T$ as the main input $X$. The additional context is split into chunks $\mathcal{C} = C_1, ..., C_k$, which can contain either segments of long documents or a set of retrieved passages. We use $\mathcal{M}_{\text{encoder}}$ to denote the small encoder model with hidden dimension $d$ and $\mathcal{M}_{\text{decoder}}$ to denote the decoder-only LLM with hidden dimension $D$.

**Encoding chunks.** We use the frozen decoder to process $X$ and encode $C_1, ..., C_k$ chunk by chunk using the small, trainable encoder $\mathcal{M}_{\text{encoder}}$:

$$\phi_i = \mathcal{M}_{\text{encoder}}(C_i) \qquad \Phi = \text{concat}(\{\phi_i\}_{i=1}^{k})$$

where $\phi_i \in \mathbb{R}^{|C_i| \times d}$ is the token-wise last layer hidden representation from $\mathcal{M}_{\text{encoder}}$ and $\Phi \in \mathbb{R}^{m \times d}$.

**Cross-attention modules.** In every decoder layer of the transformer (Vaswani et al., 2017), we insert a cross-attention module between the self-attention and feed-forward layers (see Figure 2). To construct the cross-attention module, we provide $\Phi$ as keys and values, and the hidden representation of $X$ as the queries to the attention layer. Note that in order for $\mathcal{M}_{\text{decoder}}$ to attend to $\Phi$, the key and value projection matrices in the cross attention module also serve as an up-projection that transforms the $d$-dimensional $\Phi$ into a $D$-dimensional embedding. Figure 2 illustrates the architecture of CAPE.

**Efficiency.** $\mathcal{M}_{\text{encoder}}$ is much smaller and encodes contexts in parallel to avoid the quadratic complexity of full attention. This enables CAPE to exhibit a substantially higher training and inference speed than if we were to use $\mathcal{M}_{\text{decoder}}$ to process the entire $T$-length context. Moreover, CAPE drastically reduces memory consumption by avoiding caching $TL$ key-value pairs ($L$ is the number of layers of $\mathcal{M}_{\text{decoder}}$) and instead caching only $\Phi$. In particular, using a standard decoder-only model requires $\mathcal{O}(TLD)$ memory whereas CAPE requires $\mathcal{O}(md + nLD)$. In our setting, $m \gg n$ and $D \gg d$, so in practice, we observe a substantial gap: CAPE requires 1/256 the memory required for each additional token given to $\mathcal{M}_{\text{decoder}}$.

**Comparison with retrieval-augmented LMs.** CAPE takes the inspiration for attending to parallel encodings of additional contexts from works in retrieval-augmented architectures, such as FiD (Izacard and Grave, 2021) and Atlas (Izacard et al., 2022b). In contrast to fully fine-tuning the entire model on task specific data (e.g., open-domain QA data), we freeze the pre-trained decoder model and train on unlabeled data from a pre-training corpus. As a result, CAPE is performant on a diversity of tasks beyond retrieval-augmented applications.

### 2.2 Data

We use RedPajama (RP; Together, 2023) as our training corpus, which is an open-source reproduction of the LLaMA-1 (Touvron et al., 2023a) training data. It contains about 1 trillion tokens from seven domains: ArXiv, Books, C4-RP, CC, Github, StackExchange, and Wiki. We first separate the corpus into three splits: training, retrieval, and test. The training split is preprocessed into two subsets, each of which is a collection of 8192-token sequences: (1) In $\text{RP}_{\text{train-cat}}$, we concatenate adjacent documents together to form training sequences; (2) In $\text{RP}_{\text{train-filter}}$, we keep documents from the Arxiv and the Books domains that have at least 8192 tokens and sample the training sequences such that all tokens are from the same document.

Our qualitative analysis found that data from the ArXiv and Books domains naturally contain long documents that are especially useful when training long-context models. It is also important to use a mixture of data from all domains to ensure better generalization. Thus, we use a mixture ratio of 2:1 between $\text{RP}_{\text{train-filter}}$ and $\text{RP}_{\text{train-cat}}$ for training. Our ablation studies support this strategy (§D.1).

### 2.3 Training

We use LLaMA-2-7B (Touvron et al., 2023b) as $\mathcal{M}_{\text{decoder}}$ (originally trained on 4K length). $\mathcal{M}_{\text{encoder}}$ and the new cross attention layers described in Section 2.1 add 1.8B parameters.

**Encoder.** We first train a bi-directional masked-language model (MLM) on the RedPajama dataset. $\mathcal{M}_{\text{encoder}}$ follows the configuration of RoBERTa-large (Liu et al., 2019b) but has the vocabulary of LLaMA-2, thereby amounting to 435M parameters. We train the model for 100K steps with a batch size of 2048 and a sequence length of 512 tokens. For more details, please refer to §A.1.

**Cross-Attention.** We freeze the original weights of the decoder model and only train the added cross-attention layers as well as $\mathcal{M}_{\text{encoder}}$ using the cross-entropy loss. We adopt a warmup training stage where $\mathcal{M}_{\text{encoder}}$ and $\mathcal{M}_{\text{decoder}}$ process the same inputs, thereby teaching $\mathcal{M}_{\text{decoder}}$ to copy from $\mathcal{M}_{\text{encoder}}$ through the added cross-attention modules[1]. The warmup stage uses 131M tokens randomly sampled from the training set.

After the warmup stage, we move to standard training, where each sequence has $T = 8192$ tokens. We use the last $n = 4096$ tokens as the decoder input $X$, and chunk the first 4096 tokens into $k = 16$ contexts of $|C_i| = 256$ tokens each as the encoder input $C$. Freezing the decoder allows CAPE to be trained on a single A100 GPU, which is a significant reduction in computational cost compared to training $\mathcal{M}_{\text{decoder}}$ with sequence length $T$. In practice, we use eight A100 GPUs to train the model for 20B tokens. For more training details and hyperparameters, please refer to §A.2.

### 2.4 CAPE for Instruction-Tuned Models

In this section, we extend our method to CAPE-DISTILLED (CAPED) to augment instruction-tuned models with longer context. Instruction-tuned models (Ouyang et al., 2022; Taori et al., 2023; Touvron et al., 2023b) excel in many downstream applications, but their limited context window restricts their performance in tasks that require long documents (Shaham et al., 2023) or a large number of retrieved passages (Gao et al., 2023). It is challenging to extend these models to longer context windows directly through fine-tuning due to the scarcity of high-quality instruction data.

To this end, we propose CAPED, which uses an auxiliary distillation loss to encourage $\mathcal{M}_{\text{encoder}}$ and the cross-attention layers to learn the capabilities of the already fine-tuned $\mathcal{M}_{\text{decoder}}$. This can be especially useful for settings where the fine-tuning data are not open-sourced, which is the case for LLAMA-2-CHAT (Touvron et al., 2023b).

**Distillation Loss.** We design a distillation objective, where the original $\mathcal{M}_{\text{decoder}}$ acts as the "teacher" and the CAPED model acts as the "student". In order for $\mathcal{M}_{\text{decoder}}$ to act as a teacher, we need to use input contexts that can fit in LLAMA-2-CHAT, our choice of $\mathcal{M}_{\text{decoder}}$, which has a maximum context length of $T = 4096$. First, we feed the input context into $\mathcal{M}_{\text{decoder}}$ and record the logits. Then, we split that context equally into $\mathcal{C}$ and $X$ (i.e., $m = n = 2048$) and use that as input for CAPED. We train CAPED to minimize the KL divergence between the output logits of $X$ and the teacher logits as well as the cross-entropy loss following previous distillation works (Bai et al., 2022). We train the model for 10B tokens from the filtered RP. For more details, see §A.3

## 3 Long-context Language Modeling

In this section, we evaluate CAPE and baselines with perplexity on long documents. It reflects the basic long-context modeling ability of different approaches and provides us a reliable and quantifiable metric for comparison.

**Datasets.** We evaluate on ArXiv and Books from our RedPajama test split, as well as three long-context datasets: PG19 (Rae et al., 2020), Proof-Pile (Azerbayev et al., 2023), and CodeParrot (Wolf et al., 2023). We filter all documents to have at least 32,768 tokens, and sample 5,000 sequences for each dataset. We calculate the perplexity on the last 256 tokens of each sequence. Following Peng et al. (2023), for the experiments with 128K tokens, we filter documents to have at least 131,072 tokens, and only evaluate on 10 sequences[2] due to computational costs.

**Models.** Our baseline includes LLAMA-2-7B and its long-sequence fine-tuned versions, LLAMA-2-32K-7B (Computer, 2023), YARN-64K-7B, and YARN-128K-7B (Peng et al., 2023). We also evaluate on training-free long-context methods: STREAMINGLLM (Xiao et al., 2023) and REPLUG (Shi et al., 2023a) with LLAMA-2-7B. Note that REPLUG was originally evaluated in retrieval-augmented settings only, but we found that it also helps with long-context modeling, when viewing the long context as retrieved context. Please see §B for more details on the implementations.

For CAPE, we put 2K tokens in the decoder when total tokens is 4K, and 4K tokens in the decoder in other settings. Additional tokens are split into chunks of 256 tokens and fed into the encoder.

**Results.** We show the results in Table 1. Compared to the two fully fine-tuned models, LLAMA-

[1] Preliminary experiments suggest this stage can stabilize training; ablations can be found in §D.1

[2] All 10 sequences are from different documents.

| | ArXiv | Book | PG19 | ProofPile | CodeParrot | Throughput | Mem. (GB) |
|---|---|---|---|---|---|---|---|
| **Total Tokens** $= 4,096$ | | | | | | | |
| LLAMA-2 | 2.597 | **6.282** | 7.614 | 2.409 | **1.735** | 1.00 | 19.2 |
| LLAMA-2-32K | 2.601 | 6.621 | 7.945 | 2.414 | 1.785 | 1.00 | 19.2 |
| YARN-64K | 2.651 | 6.337 | **7.326** | 2.457 | 1.764 | 1.04 | 19.2 |
| REPLUG | 2.660 | 6.343 | 7.661 | 2.465 | 1.758 | 0.12 | **16.3** |
| CAPE | **2.579** | 6.292 | 7.536 | **2.396** | 1.763 | **1.31** | 19.8 |
| **Total Tokens** $= 8,192$ | | | | | | | |
| LLAMA-2 | $> 10^3$ | $> 10^3$ | $> 10^3$ | $> 10^3$ | $> 10^3$ | - | - |
| LLAMA-2-32K | 2.505 | 6.339 | 7.744 | 2.221 | 1.729 | 1.00 | 24.9 |
| YARN-64K | 2.561 | 6.077 | **7.146** | 2.267 | **1.714** | 2.52 | 24.8 |
| REPLUG | 2.589 | 6.149 | 7.554 | 2.307 | 1.728 | 0.17 | **18.8** |
| STREAMINGLLM | 2.740 | 6.327 | 7.783 | 2.437 | 1.806 | 1.94 | 20.0 |
| CAPE | **2.496** | **6.049** | 7.372 | **2.219** | 1.715 | **3.48** | 22.6 |
| **Total Tokens** $= 32,768$ | | | | | | | |
| LLAMA-2 | $> 10^3$ | $> 10^3$ | $> 10^3$ | $> 10^3$ | $> 10^3$ | - | - |
| LLAMA-2-32K | **2.322** | 6.178 | 7.420 | **2.158** | 1.664 | 1.00 | 59.1 |
| YARN-64K | 2.359 | **5.884** | **6.809** | 2.193 | **1.640** | 1.03 | 58.9 |
| STREAMINGLLM | 2.752 | 6.358 | 7.627 | 2.503 | 1.853 | 1.16 | **20.0** |
| CAPE | 2.421 | 6.015 | 7.204 | 2.218 | 1.702 | **3.72** | 25.6 |
| **Total Tokens** $= 131,072$ | | | | | | | |
| LLAMA-2-32K | $> 10^3$ | $> 10^3$ | $> 10^3$ | $> 10^3$ | $> 10^3$ | - | - |
| YARN-64K | $> 10^3$ | $> 10^3$ | $> 10^3$ | $> 10^3$ | $> 10^3$ | - | - |
| YARN-128K | 2.359 | 5.270 | 6.306 | 2.242 | **1.264** | 1.00 | 235.6 |
| STREAMINGLLM | 2.371 | 5.058 | 6.681 | 2.270 | 1.280 | 2.56 | **20.0** |
| CAPE | **2.217** | **4.869** | **6.305** | **2.099** | 1.266 | **9.90** | 38.6 |

Table 1: Long-context language modeling text perplexity on ArXiv and Book from RedPajama, PG19, ProofPile, and CodeParrot. Throughput compares the speed (number of sequences/second) of each model with that of LLAMA-2 with the same number of total tokens. All experiments are done on 1 A100 80GB GPU, except for LLAMA-2-32K and YARN with 128K tokens, which requires model parallelism, and is conducted on 4 A100 GPUs.

2-32K and YARN-64K, CAPE achieves either lower or comparable perplexity across all datasets with lower memory usage and higher throughput. Furthermore, CAPE continues to improve on perplexity while maintaining low memory use at 128K tokens, well beyond its training lengths (8K); on the other hand, LLAMA-2-32K and YARN-64K cannot generalize beyond its training length and the memory cost increases significantly.

We also outperform REPLUG across all domains while achieving much higher throughput – we omit REPLUG at 32K tokens due to its slow speed. STREAMINGLLM maintains a low memory usage and a reasonable throughput, but the perplexity does not always decrease as the sequence length increases. Compared to STREAMINGLLM, CAPE achieves better perplexity with better throughput.

## 4 Retrieval-Augmented Applications

Retrieval-augmented settings naturally benefit from long-context LMs, as models can leverage the additional context to include more retrieval results.

### 4.1 Retrieval-augmented Language Modeling

**Datasets.** We use the test and retrieval split of RedPajama described in §2.2 for retrieval-augmented LM evaluation. Each sequence contains 2048 tokens, and the first 256 tokens are used as the query to retrieve passages from the retrieval split. The retrieval corpus contains 200M documents of 256 tokens each, and we use Contriever (Izacard et al., 2022a) to retrieve $k$ passages for each sequence.

**Models.** We evaluate full-context baselines, LLAMA-2, LLAMA-2-32K, and YARN-64K, by simply prepending the retrieved passages to the input sequence. We also evaluate REPLUG, which runs one forward pass for each retrieved passage and aggregates the results. CAPE uses 2048 tokens in the decoder and retrieved passages are fed through the encoder in parallel.

**Results.** The results are shown in Table 2. CAPE can effectively improve perplexity by using the retrieved contexts, outperforming REPLUG. Notably, CAPE extrapolates well to higher $k$ and continues

| | ArXiv | Book | C4-RP | CC | Github | StackEx | Wiki | Avg. |
|---|---|---|---|---|---|---|---|---|
| $k = 0$ ($T = 2,048$) | | | | | | | | |
| LLaMA-2 | **3.541** | **6.524** | **6.916** | **5.564** | **1.865** | **4.043** | **4.816** | **4.753** |
| LLaMA-2-32K | 3.561 | 6.892 | 7.798 | 5.931 | 1.932 | 4.262 | 4.958 | 5.048 |
| YaRN-64K | 3.633 | 6.631 | 7.164 | 5.701 | 1.930 | 4.164 | 4.837 | 4.866 |
| $k = 8$ ($T = 4,096$) | | | | | | | | |
| LLaMA-2 | 3.602 | 6.581 | 6.963 | 5.348 | 1.829 | 4.044 | 4.815 | 4.740 |
| LLaMA-2-32K | 3.642 | 6.985 | 7.767 | 5.645 | 1.893 | 4.270 | 4.988 | 5.027 |
| YaRN-64K | 3.752 | 6.718 | 7.218 | 5.466 | 1.894 | 4.178 | 4.847 | 4.868 |
| RePlug | 3.535 | 6.494 | 6.895 | 5.395 | 1.833 | 4.029 | 4.798 | 4.711 |
| CAPE | **3.486** | **6.481** | **6.884** | **5.319** | **1.793** | **3.709** | **4.302** | **4.568** |
| $k = 20$ ($T = 7,168$) | | | | | | | | |
| RePlug | 3.531 | 6.490 | 6.894 | 5.386 | 1.830 | 4.028 | 4.795 | 4.708 |
| CAPE | **3.475** | **6.463** | **6.875** | **5.266** | **1.782** | **3.703** | **4.296** | **4.551** |
| $k = 50$ ($T = 14,848$) | | | | | | | | |
| RePlug | 3.530 | 6.491 | 6.899 | 5.392 | 1.830 | 4.028 | 4.794 | 4.709 |
| CAPE | **3.467** | **6.457** | **6.881** | **5.273** | **1.777** | **3.701** | **4.292** | **4.550** |

Table 2: Retrieval-augmented language modeling. We report test perplexity on RedPajama across all domains. We calculate perplexity on the last 1792 tokens of the decoder input (to exclude the query tokens). $k$ is the number of retrieved contexts used, and $T$ is the total number of tokens. For LLaMA-2, YaRN-64K, and LLaMA-2-32K, we concatenate the contexts and prepend to the input. Avg. is the macro average across all domains.

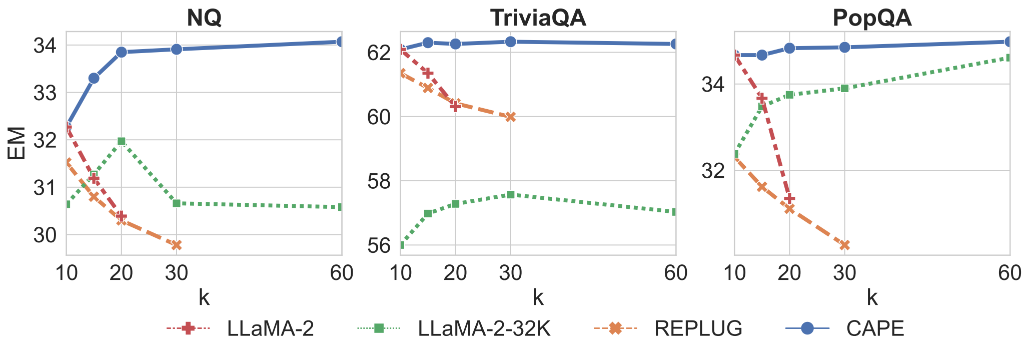


Figure 3: Open-domain QA results. LLaMA-2 is limited to 20 passages, and RePlug is limited to 30 passages due to memory constraints. For the complete results, refer to Table 6.

to improve perplexity even with $k = 50$ (trained with $k = 16$). CAPE transfers well to the retrieval-augmented setting whereas the full-context decoder models degrade in performance.

### 4.2 Open-domain Question Answering

Given a question and a large corpus of documents, open-domain question answering (QA) requires the model to retrieve relevant passages and generate the answer. A model that can leverage a large number of retrieved passages without being distracted by irrelevant ones is desirable for this task.

**Datasets.** We adopt three open-domain QA datasets: Natural Questions (NQ; Kwiatkowski et al., 2019; Lee et al., 2019a), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2023). For each question, we use Contriever to retrieve $k$ passages from Wikipedia[3].

**Models.** We compare CAPE with LLaMA-2, LLaMA-2-32K, and RePlug. For each model, we use two in-context demonstrations. For CAPE, we use 10 passages in the decoder and all other passages are encoded separately by the encoder. Refer to §C.1 for more details.

**Results.** The results are shown in Figure 3. CAPE consistently outperforms all models across all datasets and $k$. Notably, CAPE outperforms LLaMA-2-32K on NQ and TriviaQA by over 3 and 4 points in exact match, respectively. Furthermore, CAPE does not degrade in performance as the number of retrieved passages increases, while other models often performs worse at larger $k$, as

[3]Snapshot from 2018-12-20, and each passage is 100 words (Karpukhin et al., 2020).

| | $k$ | SST2 | MR | AGNews | SST5 | TREC | TREC-F | DBPedia | NLU-S | NLU-I | BANKING | CLINIC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LLAMA-2 | 2 | 89.1 | 96.7 | 72.7 | 3.9 | **48.0** | 16.7 | **94.0** | 42.3 | 22.3 | 38.4 | 59.1 |
| + CAPE | 2 + 18 | 90.7 | **98.4** | 71.9 | **46.7** | 47.1 | 22.8 | **94.0** | **48.9** | 30.4 | 42.5 | 62.4 |
| | 2 + 38 | **92.9** | 98.0 | **73.2** | 45.5 | 47.5 | **25.1** | 93.3 | 48.8 | **31.6** | **46.0** | **62.8** |
| LLAMA-2† | 40 | 94.3 | 98.7 | 74.7 | 52.3 | 87.7 | 54.8 | 95.1 | 76.7 | 62.1 | 50.4 | 72.0 |

Table 3: ICL results averaged across 3 seeds. $k$ is the number of demonstrations. All models uses 2 demonstrations in the decoder, and we add $+m$ demonstrations to the encoder for CAPE. † denotes the oracle setting with $k = 40$ demonstration in the decoder.

| | | Question Answering | | | Summarization | | |
|---|---|---|---|---|---|---|---|
| | Total tokens | NQA | Qspr | QALT | GvRp | SSFD | QMSum |
| LLAMA-2-CHAT | 2K | 17.1 | 14.6 | 28.6 | 16.0 | 16.4 | 19.3 |
| + CAPED | 2K + 2K | 19.5 | **20.5** | **30.2** | **16.5** | 16.4 | **19.6** |
| | 2K + 30K | 21.6 | 19.9 | 29.6 | 15.8 | **16.7** | 19.5 |
| | 2K + All | **21.9** | 19.9 | 29.6 | 15.9 | **16.7** | 19.5 |
| LLAMA-2-32K INSTRUCT | 32K | 12.2 | 18.1 | 41.6 | 19.9 | 10.0 | 10.3 |

Table 4: ZeroSCROLLS validation results. The total number of tokens includes both the input and generated tokens. For NarrativeQA (NQA) and Qspr (QASPER), we report the F1 scores. For QALT (QuALITY), we report accuracy. For GovReport (GvRp), SummScreenFD (SSFD), and QMSum, we report the ROUGE-L scores. CAPED uses 2K tokens in the decoder, and additional tokens are inputted through the encoder.

they are sensitive to the large amount of redundant or irrelevant passages.

## 5 In-Context Learning

In-context learning (ICL; Brown et al., 2020) is one of the most important emerging qualities of LLMs. In this experiment, we examine whether CAPE can effectively utilize the demonstrations from the encoder context and improve the performance.

Specifically, we use a range of classification tasks that contains a large number (up to 150) of categories, where the model can benefit from additional demonstrations. Following previous work (Ratner et al., 2023), we use a test set size of 250 examples for each dataset.

**Models.** Our baseline is LLAMA-2 with 2 demonstrations. For CAPE, we add additional demonstrations in the encoder. We also compare with an "oracle", where the LLAMA-2 decoder takes 40 demonstrations. Note that the oracle is significantly more expensive. More details are in §C.2.

**Results.** The results are shown in Table 3. We first observe that compared to the decoder-only baseline, CAPE can effectively use the additional demonstrations from the encoder context; the performance further increases or remains consistent with more demonstrations in the encoder. However, there is still a large gap to the 40-demonstration oracle. Our hypothesis is that in-context learning requires both query-demonstration interactions and demonstration-demonstration interactions, which CAPE cannot provide. Regardless, CAPE can be always applied on top of the decoder-only model to add additional demonstrations, with little extra computational and memory cost.

## 6 Instruction-tuned Models for Long Text Understanding

**Dataset.** ZeroSCROLLS (Shaham et al., 2023) is a collection of zero-shot long-context understanding tasks that require instruction-following abilities. Specifically, we test on NarrativeQA, QASPER, QuALITY, GovReport, SummScreenFD, and QMSum, which all have large validation sets made available. We follow the formats and instructions of Shaham et al. (2023) for each dataset, except the long text is placed before the instructions.

**Models.** For CAPED, we use 2048 tokens in the decoder and put the remaining tokens to the encoder as chunks of 256 tokens. We compare with LLAMA-2-CHAT and LLAMA-2-32K INSTRUCT[4], which was fine-tuned on multi-round conversational data as well as long-context summarization and QA data. We allow the model to

[4] https://huggingface.co/togethercomputer/Llama-2-7B-32K-Instruct

generate 1024 tokens for the summarization tasks and 50 tokens for the question answering tasks.

**Results.** Table 4 shows that CAPED improves upon LLAMA-2-CHAT with 2K tokens across all tasks. The performance of CAPED improves or remains consistent as we scale up the number of tokens in the context window. Notably, CAPED improves upon LLAMA-2-CHAT by 3 points in F1 scores on NQA, which has the longest average number of words among all ZeroSCROLLS tasks. Furthermore, CAPED outperforms LLAMA-2-32K INSTRUCT on 4 out of the 6 tasks, despite being trained on unlabeled data. We provide qualitative examples and more analysis in §C.3.

## 7 Ablation Studies

We conduct comprehensive ablations to show the effectiveness of our training data mixture, pre-training and fine-tuning the encoder, and the warmup training stage. We also ablate to verify the effectiveness of the KL divergence loss in CAPED. All the ablations can be found in §D.

## 8 Related Work

**Long-context language models.** Many recent works on long-context language models aim to solve the problem of positional embedding extrapolation in transformers (Peng et al., 2023; Chen et al., 2023a). Others simply fine-tune LMs on longer sequences (Xiong et al., 2023; Chen et al., 2023b). Distinctively, several recent papers propose to extend the context window of LMs by modifying the attention mechanism: Xiao et al. (2023) discover the use of "sink tokens" in sliding windows and Bertsch et al. (2023) retrieve relevant tokens from a cache instead of attending to all tokens This results in memory-efficient long-context LMs, but they can degrade with longer contexts, as the same positional embedding may be seen multiple times. The key advantage of CAPE is that it does not degrade for inputs longer than the training sequence, while achieving greater memory efficiency than full fine-tuning approaches. Novel architectures and pre-training techniques, such as S4 (Gu et al., 2022), RPT (Rubin and Berant, 2023), and Mamba (Gu and Dao, 2023), also extend the context window of LMs at greater efficiency. However, pre-training is extremely expensive at scale and, thus, these methods cannot leverage existing powerful pre-trained LLMs. It is also unclear if state-space models can replace transformers (Jelassi et al., 2024).

**Retrieval-augmented language models.** Augmenting language models with retrieval has been useful in a number of applications. It's often applied to question answering tasks, where retrieval can enrich LMs with external knowledge (Lee et al., 2019b; Karpukhin et al., 2020). Recently, combining LMs with retrieval systems for more generalized purposes, such as language modeling, has been explored: Guu et al. (2020); Borgeaud et al. (2022); Izacard et al. (2022b); Min et al. (2023) pre-train LMs with retrieval, and Shi et al. (2023b); Lin et al. (2023) use logits interpolation from separate forward passes to incorporate retrieval information.

Our architecture is similar to Atlas (Izacard et al., 2022b) and RETRO (Borgeaud et al., 2022), which encode retrieved passages independently and uses cross-attention to incorporate them into the decoder. However, they are both models trained from scratch on retrieval-augmented data, which are expensive to acquire at the pre-training scale. CAPE only requires fine-tuning on long document data, which are much more efficient to obtain; CAPE is also applicable to any decoder-only LM, allowing us to extend context windows for pre-existing strong models. RETRO-Fitting (Borgeaud et al., 2022) similarly fine-tunes a pre-trained model but requires retrieval-augmented data.

## 9 Conclusion

We propose CAPE as a way to extend the context window of existing language models. The key idea behind CAPE is to leverage a small encoder and cross-attention to process long inputs and achieve low memory and computational complexity. Compared to existing methods, CAPE extrapolates to input lengths well beyond the training length, while remaining efficient and effective. Consequently, CAPE augments pre-trained models to be performant on both long-context and retrieval-augmented applications. We also show that CAPED can be applied to off-the-shelf models with additional contexts using an auxillary loss with only unlabeled data. We believe that there is still much room for improvements in terms of curating better data for training flexible and robust models. Ultimately, we hope our work can be a useful, efficient, and accessible tool for the community to study long-context models in diverse settings.

## Limitations

One limitation of our work include the focus on LLaMA-2-7B. Due to computational resource constraints and the cost of training, we only applied CAPE to LLaMA-2-7B. We hope that future work can investigate the applicability of our framework to a wider variety of LLMs of different sizes. Similarly, we only applied CAPED to LLaMA-2-Chat-7B, but we look forward to members in the community to apply it to other instruction-tuned or other fine-tuned models.

We also acknowledge that certain hyperparameters are not studied in depth due to training costs – such as the ratio between $\text{RP}_{\text{train-filter}}$ and $\text{RP}_{\text{train-cat}}$, learning rate, and the size of the small encoder model. We also fixed Contriever (Izacard et al., 2022a) to be the retriever of choice in this work, but it would be useful to study a greater range of retrievers.

## Ethics Statement

LLMs are known to potentially output harmful and/or offensive languages, and the LLaMA-2-based models we use in this work are no exceptions. Since these models are trained on internet-size corpora (e.g., RedPajama), it can be difficult and expensive to filter out such offensive languages.

Our models are also trained on RP, which could be misused in certain contexts. Although addressing this issue in large-scale pre-training corpus is out of the scope for this work, we hope that future work will carefully resolve possible misuse issues in these models.

## References


Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. 2021. A general language assistant as a laboratory for alignment.

Zhangir Azerbayev, Edward Ayers, and Bartosz Piotrowski. 2023. Proofpile: A pre-training dataset of mathematical texts.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback.

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R. Gormley. 2023. Unlimiformer: Long-range transformers with unlimited length input. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. 2022. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning (ICML)*, volume 162, pages 2206–2240.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders.

Mingda Chen, Zewei Chu, Sam Wiseman, and Kevin Gimpel. 2022. SummScreen: A dataset for abstractive screenplay summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8602–8615, Dublin, Ireland. Association for Computational Linguistics.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023a. Extending context window of large language models via positional interpolation.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023b. Longlora: Efficient fine-tuning of long-context large language models.

Together Computer. 2023. Llama-2-7b-32k.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of* the Association for Computational Linguistics: Human Language Technologies, pages 4599–4610, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional Transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023. Enabling large language models to generate text with citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6465–6488, Singapore. Association for Computational Linguistics.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces.

Albert Gu, Karan Goel, and Christopher Re. 2022. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-augmented language model pre-training. In *International Conference on Machine Learning (ICML)*.

Zhixiong Han, Yaru Hao, Li Dong, Yutao Sun, and Furu Wei. 2023. Prototypical calibration for few-shot learning of language models. In *The Eleventh International Conference on Learning Representations*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations (ICLR)*.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 2021. Surface form competition: Why the highest probability answer isn't always right. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7038–7051, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1419–1436, Online. Association for Computational Linguistics.

Maor Ivgi, Uri Shaham, and Jonathan Berant. 2023. Efficient Long-Text Understanding with Short-Text Models. *Transactions of the Association for Computational Linguistics*, 11:284–299.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022a. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022b. Atlas: Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. 2024. Repeat after me: Transformers are better than state space models at copying.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316, Hong Kong, China. Association for Computational Linguistics.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019a. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics (ACL).

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019b. Latent retrieval for weakly supervised open domain question answering. In *Association for Computational Linguistics (ACL)*, pages 6086–6096.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Scott Yih. 2023. Ra-dit: Retrieval-augmented dual instruction tuning.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2019a. Benchmarking natural language understanding services for building conversational agents. *ArXiv*, abs/1903.05566.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Association for Computational Linguistics (ACL)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Sewon Min, Weijia Shi, Mike Lewis, Xilun Chen, Wen-tau Yih, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2023. Nonparametric masked language modeling. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2097–2118, Toronto, Canada. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:27730–27744.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel Bowman. 2022. QuALITY: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5336–5358, Seattle, United States. Association for Computational Linguistics.

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

Ofir Press, Noah Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations (ICLR)*.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*.

Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. Parallel context windows for large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6383–6402, Toronto, Canada. Association for Computational Linguistics.

Ohad Rubin and Jonathan Berant. 2023. Long-range language modeling with self-retrieval.

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. 2023. ZeroSCROLLS: A zero-shot benchmark for long text understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7977–7989, Singapore. Association for Computational Linguistics.

Uri Shaham, Elad Segal, Maor Ivgi, Avia Efrat, Ori Yoran, Adi Haviv, Ankit Gupta, Wenhan Xiong, Mor Geva, Jonathan Berant, and Omer Levy. 2022. SCROLLS: Standardized CompaRison over long language sequences. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 12007–12021, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. 2023a. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning (ICML)*.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. 2023b. Replug: Retrieval-augmented black-box language models.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding.

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.

Together. 2023. Redpajama: An open source recipe to reproduce llama training dataset.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models.

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

Ellen M. Voorhees and Dawn M. Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, page 200–207, New York, NY, USA. Association for Computing Machinery.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Thomas Wolf, Loubna Ben Allal, Leandro von Werra, Li Jia, and Armel Zebaze. 2023. A dataset of python files from github. `https://github.com/huggingface/blog/blob/main/codeparrot.md?version=codeparrot/codeparrot-valid-v2-near-dedup`.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma. 2023. Effective long-context scaling of foundation models.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Neural Information Processing Systems*.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir Radev. 2021. QMSum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5905–5921, Online. Association for Computational Linguistics.

# A Training Details

## A.1 Pre-training Encoder

The encoder follows the configuration of RoBERTa-large (Liu et al., 2019b) – it has 24 layers, a hidden size of 1024, and 16 attention heads. However, we use the architecture of LLAMA-2, which means that the vocabulary size is different and the attention module contains an additional output projection. We refer to Liu et al. (2019b) and Touvron et al. (2023a) for more details.

We pre-trained the encoder for 100K steps on RP using the masked language modeling objective (Devlin et al., 2019). We used a batch size of 2048 sequences, where each sequence consisted of 512 tokens. The learning rate was set to $10^{-3}$ with a warm-up of the first 4% of the steps. We used eight A6000 GPUs with gradient accumulation of 16. Furthermore, we employed a masking rate of 30% and disabled the next sentence prediction objective. We always replace token with the `[MASK]` token if it is masked instead of replacing it with a random token or the original token. Finally, we used the AdamW optimizer (Loshchilov and Hutter, 2019) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$, as implemented by the HuggingFace Transformers library (Wolf et al., 2020).

## A.2 Training CAPE

The attention module in LLAMA-2 is consisted of four projection matrices: key, value, query, and output. In contrast to original transformers (Vaswani et al., 2017), the output projection matrix is used as an additional attention output projection. When we first insert the cross-attention layers into the decoder, we initialize the weights of the key, value, and query projection matrices with the respective weights from the decoder's self-attention layer in the same transformer block. Furthermore, since the hidden dimension of the encoder is smaller than the hidden dimension of the decoder, $d < D$, we use only copy the first $d$ rows of the key and value projection matrices from the self-attention module to the cross-attention module. Lastly, the output projection matrix is initialized with all zeros. While we did not investigate this initialization in detail, the intuition is that we want the model should use the tokens from the encoder using a similar mechanism as the decoder uses for its own tokens. However, we want the model to learn the output projection from scratch, as it may be too disruptive to have doubled the number of attention modules.

Then, we employ a warmup initialization method that simply trains the model to copy the input tokens from the encoder to the decoder. Specifically, we use the same inputs $X = \mathcal{C}$ for both the encoder and the decoder, and $X$ consists of $n = 256$ tokens. However, for the encoder, we chunk $X$ into $k = 4$ sequences of 64 tokens to construct $\mathcal{C}$. This step was trained for 4K steps with a batch size of 128 and peak learning rate of $5 \times 10^{-4}$, which totals to 131M tokens. We noticed that the model quickly learned to copy the input tokens from the encoder to the decoder, and the loss was close to zero after just 1K steps. The intuition behind this initialization strategy was to instill strong inductive bias between the encoder input and decoder outputs. From our early experiments, we found that this initialization strategy helped stabilize the later training.

Finally, we train CAPE for 20K steps with a batch size of 128. We use eight A100 80 GB GPUs with a per-device batch size of 2 and gradient accumulation of 8, which took approximately 750 GPU hours. We also use a peak learning rate of $3 \times 10^{-4}$ with a warm-up of 4% of the steps, and a cosine learning rate schedule. We use the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$.

## A.3 Training CAPED

We leverage an additional distillation loss for CAPED. For encoder input $C$ and decoder input $X$, we first calculate the logits $\theta(\text{concat}(C, X))$ by running forward passes with the original model parameterized by $\theta$. Due to storage constraints, we only store the top 50 likelihoods and their indices in the vocabulary for each token in $X$, following (Askell et al., 2021; Bai et al., 2022).

Then, during training, we define the distillation loss as the KL Divergence between the teacher model's probability distribution and the student model's probabilities distribution for the previously stored top 50 tokens. Concretely, our distillation loss is defined as follows:

$$\mathcal{L}_{\text{KL}} = D_{\text{KL}}(\mathcal{M}_{\text{decoder}}(S) || \mathcal{M}_{CAPE}(\mathcal{C}, X))$$

where $S = \text{concat}(\mathcal{C}, X)$, $\mathcal{M}_{\text{decoder}}(S)$ is the probability distribution of the top 50 tokens for $X$, and $\mathcal{M}_{CAPE}(\mathcal{C}, X)$ takes $\mathcal{C}$ as the encoder input and $X$ as the decoder input and outputs the probability distribution of the same 50 tokens on $X$.

Although Bai et al. (2022) also use an additional category that represents the sum of all other tokens' probabilities, we found that this may cause the KL Divergence to be undefined for when the sum of other probabilities is 0. For our main model, we use a coefficient of 2 in front of $\mathcal{L}_{\text{KL}}$ when adding to the cross-entropy loss to calculate the total loss. We experiment with this coefficient in §D.1.

# B Baseline Implementations

**REPLUG.** Although REPLUG (Shi et al., 2023b) was introduced as a method to augment language models with retrieval, we found that the technique of interpolating logits from separate forward passes can also transfer well to the long context setting. Among the methods that we compare to, REPLUG uniquely improves performance upon the base model in both the long-context and the retrieval-augmented LM settings. This gives us an additional point of comparison across the two settings.

Following the original authors, we use Contriever (Izacard et al., 2022a) to calculate the scores for each previous context by using the first 256 tokens following the previous context as the query in the long-context setting. We did not include the additional memory and inference time costs of calculating the Contriever scores in our evaluation.

**STREAMINGLLM.** We follow the implementation of STREAMINGLLM from the original authors[5] (Xiao et al., 2023). Specifically, we use their best settings, where we enable the positional shifts and cache 4 sink tokens and 2044 recent tokens.

The original code evaluates the model using a stride of 1 token at a time, where the cache is updated after every token, but this is not feasible for our large-scale evaluation. Therefore, we use a stride of 2048 tokens, and we update the cache after each stride. We show the difference in performance between the two settings in Table 5, and we found that STREAMINGLLM benefits from using a larger stride. We leave future exploration in this direction to future work.

# C Evaluation Settings

## C.1 Open-domain Question Answering

The full results for the open-domain question answering experiments are shown in Table 6. REPLUG only uses up to $k = 30$ passages due to memory constraints, and LLAMA-2 has a window size of 4096, which limits $k$ to 20. While LLAMA-2-32K can use more than $k = 60$ passages with a context size of 32K, we only use up to 60 passages due to cost of generation. For each demonstration, we only show the the top 1 retrieved passages instead of the top $k$ passages.

## C.2 In-context Learning

For our in-context learning experiments, we use the datasets commonly used in previous works (Zhao et al., 2021; Lu et al., 2022; Han et al., 2023; Ratner et al., 2023): SST-2 (Socher et al., 2013), MovieReview (MR Pang and Lee, 2005), AGNews (Zhang et al., 2015), SST-5 (Socher et al., 2013), TREC (Voorhees and Tice, 2000), DBPedia (Zhang et al., 2015), NLU (Liu et al., 2019a), BANKING77 (Casanueva et al., 2020), CLINIC150 (Larson et al., 2019). We follow the prompts used in Ratner et al. (2023) for all datasets. During evaluation, we first calculate the log-likelihood of each option and select the option with the highest likelihood. We sample the in-context learning demonstrations from the training set such that each label has equal number of demonstrations (except for possible reminders).

Furthermore, we first calculate the accuracy for each dataset using four different metrics: likelihood, likelihood normalized for length, calibrated likelihood, and calibrated likelihood normalized for length. We calibrate using Domain Conditional PMI (Holtzman et al., 2021), but use the empty string as the domain string for all dataset for simplicity. We then choose the metrics that yields the highest score for the LLAMA-2 model in the two-shot, and apply the same metrics to all other models. The metrics used for each dataset are shown in Table 7. In this work, we did not investigate how to best calibrate CAPE in ICL settings. We leave these explorations for future work.

## C.3 ZeroSCROLLS

We use a subset of the ZeroSCROLLS (Shaham et al., 2023) with large validation sets: NarrativeQA (Kočiský et al., 2018), Qasper (Dasigi et al., 2021), QuALITY (Pang et al., 2022), GovReport (Huang et al., 2021), SummScreenFD (Chen et al., 2022), and QMSum (Zhong et al., 2021). Specifically, we use the validation sets of these datasets made available by SCROLLS (Shaham et al., 2022).

However, we follow the same evaluation set up as ZeroSCROLLS, where models are evaluated in the zero-shot setting. We also use the same evaluation metrics as ZeroSCROLLS for each dataset.

[5] `https://github.com/mit-han-lab/streaming-llm`

| | Stride | ArXiv | Book | PG19 | ProofPile | CodeParrot |
|---|---|---|---|---|---|---|
| **Total Tokens** = 8192 | | | | | | |
| STREAMINGLLM | 1 | 2.823 | 6.381 | 7.817 | 2.522 | 1.848 |
| | 2048 | 2.740 | 6.327 | 7.783 | 2.437 | 1.806 |

Table 5: Performance of STREAMINGLLM with different stride lengths.

| | $k$ | NQ | TQA | PopQA |
|---|---|---|---|---|
| LLAMA-2 | 1 | 28.37 | 56.44 | 27.17 |
| | 5 | 31.91 | 61.08 | 33.83 |
| | 10 | 32.27 | 62.09 | 34.67 |
| | 15 | 31.19 | 61.35 | 33.67 |
| | 20 | 30.39 | 60.31 | 31.35 |
| LLAMA-2-32K | 10 | 30.64 | 56.00 | 32.38 |
| | 15 | 31.27 | 56.98 | 33.48 |
| | 20 | 31.97 | 57.28 | 33.75 |
| | 30 | 30.66 | 57.57 | 33.90 |
| | 60 | 30.58 | 57.03 | 34.61 |
| REPLUG | 5 | 31.27 | 61.21 | 32.40 |
| | 10 | 31.52 | 61.35 | 32.31 |
| | 15 | 30.80 | 60.89 | 31.62 |
| | 20 | 30.30 | 60.41 | 31.11 |
| | 30 | 29.78 | 59.99 | 30.27 |
| CAPE | 10 | 32.27 | 62.09 | 34.67 |
| | 15 | 33.30 | 62.30 | 34.67 |
| | 20 | 33.85 | 62.26 | 34.83 |
| | 30 | 33.91 | 62.33 | 34.85 |
| | 60 | 34.07 | 62.26 | 34.98 |

Table 6: Open-domain QA results. We report exact match scores for the Natural Questions(NQ) test set, TriviaQA(TQA) validation set, and PopQA test set. All models use two-shot in-context learning. $k$ is the number of retrieved passages, and CAPE uses the top 10 passages in the decoder and all passages in the encoder.

| | Normalized | Calibrated |
|---|---|---|
| SST2 | No | Yes |
| MR | No | No |
| AGNews | No | No |
| SST5 | No | Yes |
| TREC | No | No |
| TREC-F | No | No |
| DBPedia | Yes | Yes |
| NLU-S | Yes | Yes |
| NLU-I | No | No |
| BANKING | No | No |
| CLINIC | No | No |

Table 7: Metrics used for each dataset. For normalization, we divide the log-likelihood by the length of the prompt. For calibration, we use Domain Conditional PMI (Holtzman et al., 2021) with the empty string as the domain string for all datasets for simplicity.

For the question answering datasets (NarrativeQA, Qasper, and QuALITY), we allow the model to generate up to 50 tokens, and we use greedy decoding.

For the summarization datasets (GovReport, SummScreenFD, and QMSum), we allow the model to generate up to 1024 tokens, following the original authors. For SummScreenFD and QMSum, we use greedy decoding, and for GovReport we use nucleus sampling (Holtzman et al., 2020) with a temperature of 1.0 and top-$p$ of 0.95 and a minimum generation length of 10 tokens.

This is because GovReport has a much longer gold summary than the other datasets, and sampling methods are typically used in long generation settings, and that greedy decoding may degenerate. The minimum generation length helps prevent trivial outputs, such as empty string. To account for the randomness in the sampling method, we average GovReport performance over 3 seeded runs, and we found that the standard deviation is less than 0.20 ROUGE-L scores in all settings.

We also show some generation examples in Table 8 and 9. We find that CAPED can especially benefit from the additional contexts in the encoder in the QA datasets, where the answer may be localized to just one small part of the entire input. On the other hand, summarization tasks do not catastrophically fail when the model only has access to only part of the input, as the model can still generate a coherent summary for the provided context, achieving reasonable ROUGE-L scores.

# D Ablations

## D.1 Training Settings

**Training with retrieved documents.** Even though CAPE was trained with long documents, it also achieves strong performance on retrieval-augmented applications. In this subsection, we study a different data strategy: training CAPE with retrieved documents. Specifically, we pair the training sequences described in §2.2 with retrieved passages from the retrieval split of RP using Contriever. Following our setup for retrieval-augmented LM evaluation, we use the first 256 tokens of the decoder input $X$ as the query, and retrieve $k = 16$ passages to form the additional contexts $C$. Then, we train CAPE with retrieved passages (RetDoc) using the same training settings as in §2.3.

As shown in Table 10, we find that training on RetDoc results in slightly stronger performance in the retrieval-augmentation setting, but the results on long documents are worse. Augmenting a pre-training corpus with retrieval contexts can be extremely computationally and storage expensive, as well as a large implementation overhead; CAPE's simple long-document data strategy achieves a good balance between efficient training and strong performance on both long-context and retrieval-augmented applications.

**Choices of unlabeled data.** In Table 10, we show the results of training CAPE with only the filtered documents from the ArXiv and Books domains (**$\text{RP}_{\text{train-filter}}$**), and only the concatenated RP documents (**$\text{RP}_{\text{train-cat}}$**). We find that training on $\text{RP}_{\text{train-filter}}$ is more beneficial for the long-document setting and training on $\text{RP}_{\text{train-cat}}$ is better for the retrieval setting, but using a mixture of both leads to a more balanced and generalizable model.

**Encoder training.** Finally, we investigate how to best train the encoder. To this end, we train CAPE with (1) freezing the encoder after pre-training and the warmup stage, (2) training with a randomly initialized encoder, and (3) using the pre-trained model without the warmup stage. As shown in Table 10, we find that the copying warmup training and fine-tuning the encoder during training are both crucial for strong performance.

### D.2 KL Divergence

The key component of CAPED is the KL Divergence loss. To understand the importance of this auxillary loss, we explore the performance of CAPED when train without the KL Divergence loss as well as with difference coefficients for each loss. Let $\mathcal{L}_{CE}, \mathcal{L}_{KL}$ be the cross entropy loss and the KL Divergence loss, respectively. Then, the total loss is $\mathcal{L} = c_{CE}\mathcal{L}_{CE} + c_{KL}\mathcal{L}_{KL}$.

The results are shown in Table 13. We find that the KL Divergence loss is crucial for the performance of CAPED on summarization tasks as well as QALT, where the performance may decrease compared to LLAMA-2-CHAT with 2K tokens.

We show the full results for the ablation studies in Table 11, 12, and 13.

**Encoder Input** $C_1$**:**
We propose a novel pre-training method called BRLM, which can effectively alleviates the distance between different source language spaces. Our proposed approach significantly improves zero-shot translation performance, consistently surpassing pivoting and multilingual approaches. Meanwhile, the performance on supervised translation direction remains the same level or even better when using our method. Related Work In recent years, zero-shot translation in NMT has attracted widespread attention in academic research. Existing methods are mainly divided into four categories: pivot-based method, transfer learning, multilingual NMT, and unsupervised NMT. Pivot-based Method is a common strategy to obtain a source→target model by introducing a pivot language. This approach is further divided into pivoting and pivot-synthetic. While the former firstly translates a source language into the pivot language which is later translated to the target language BIBREF4, BIBREF5, BIBREF12, the latter trains a source→target model with pseudo

**Encoder Input** $C_2$**:**
, NMT heavily relies on large-scale parallel data, resulting in poor performance on low-resource or zero-resource language pairs BIBREF3. Translation between these low-resource languages (e.g., Arabic→Spanish) is usually accomplished with pivoting through a rich-resource language (such as English), i.e., Arabic (source) sentence is translated to English (pivot) first which is later translated to Spanish (target) BIBREF4, BIBREF5. However, the pivot-based method requires doubled decoding time and suffers from the propagation of translation errors. One common alternative to avoid pivoting in NMT is transfer learning BIBREF6, BIBREF7, BIBREF8, BIBREF9 which leverages a high-resource pivot→target model (parent) to initialize a low-resource source→target model (child) that is further optimized with a small amount of available parallel data. Although this approach has achieved success in some low-resource language pairs, it still performs very poorly in extremely low-resource or zero-resource translation scenario. Specifically, BIBREF8 reports that without any child model training data,

**Encoder Inputs** $[C_3, \ldots, C_{17}]$ **Omitted...**

**Decoder Input** $X$**:**
tokens are selected to be masked. Among the selected tokens, $80\%$ of them are replaced with [MASK] token, $10\%$ are replaced with a random BPE token, and $10\%$ unchanged. The prediction accuracy of masked words is used as a stopping criterion in the pre-training stage. Besides, we use fastalign tool BIBREF34 to extract word alignments for BRLM-HA. Experiments ::: Main Results Table TABREF19 and TABREF26 report zero-shot results on Europarl and Multi-UN evaluation sets, respectively. We compare our approaches with related approaches of pivoting, multilingual NMT (MNMT) BIBREF19, and cross-lingual transfer without pretraining BIBREF16. The results show that our approaches consistently outperform other approaches across languages and datasets, especially surpass pivoting, which is a strong baseline in the zero-shot scenario that multilingual NMT systems often fail to beat BIBREF19, BIBREF20, BIBREF23. Pivoting translates source to pivot then to target in two steps, causing inefficient translation process. Our approaches use one encoder-decoder model to translate between any zero-shot directions, which is more efficient than pivoting. Regarding the comparison between transfer approaches, our cross-lingual pretraining based transfer outperforms transfer method that does not use pretraining by a large margin. Experiments ::: Main Results ::: Results on Europarl Dataset. Regarding comparison between the baselines in table TABREF19, we find that pivoting is the strongest baseline that has significant advantage over other two baselines. Cross-lingual transfer for languages without shared vocabularies BIBREF16 manifests the worst performance because of not using source↔pivot parallel data, which is utilized as beneficial supervised signal for the other two baselines. **Additional Decoder Input Omitted...**
You are given a scientific article and a question. Answer the question as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the article, write "unanswerable". If the question is a yes/no question, answer "yes", "no", or "unanswerable".
Question:
what are the pivot-based baselines?
Answer:

**Model Outputs:**
LLAMA-2-CHAT output: unanswerable.
CAPE output with encoder contexts: pivot-based baselines include pivoting and pivot-synthetic.
Gold answers: pivoting, pivoting$_{\mathrm{m}}$

Table 8: ZeroSCROLLS generation example on the Qasper dataset. CAPE sees the entire article through the decoder and the encoder, whereas LLAMA-2-CHAT only sees a 2K token window. For brevity, we omit part of the decoder input and only show 2 out of $k = 17$ encoder inputs for CAPE.

**Encoder Input $C_1$:**
Phoebe: Almost sunrise. Do you think you're ready to try the window again? Prue: Yeah, yeah, but Abraxas will be ready for us here. We have to take him by surprise, go where we're most powerful, where we're most connected. [Cut to the park. Prue, Piper and Phoebe have joined hands around a stone.] Prue, Piper and Phoebe: "Hear now the words of the witches, the secrets we hid in the night, the oldest of Gods are invoked here, the great work of magic is sought." [Cut to Abraxas undoing the spell that gave them their powers.] [Cut back to the girls.] Prue, Piper, Phoebe: "In this night, and in this hour we call upon the ancient power." [Cut back to Abraxas. The pages of the Book of Shadows turn over. He turns back to the spell and continues to read it backwards.] [Cut back to the girls.] Prue, Piper, Phoebe: "Hear

**Encoder Input $C_2$:**
hoebe: Honey, we've done a lot of good as witches too. You know that. But that doesn't mean that bad things still aren't gonna happen. But just because we can't help that, doesn't mean that is our fault. Piper: Or yours. Phoebe: Come here. (They all hug.) Grams' Voice: The Power of Three. (They look around.) Phoebe: Okay, did you guys hear that? Piper: Mm-hmm. Prue: Grams? Grams' Voice: The Power of Three. Prue: How is that possible? Phoebe: I don't know but it's gotta to be a good thing if we're all hearing it. Piper: Maybe it means we're connected. Phoebe: To be as one. Maybe that's what the Book meant by the Power of One. Piper: The three of us working as one. Prue: Which is what the triquetra represents. (Phoebe looks at her watch)

**Encoder Inputs $[C_3, \ldots, C_{44}]$ Omitted...**

**Decoder Input $X$:**
the secrets we hid in the night, the oldest of Gods are invoked here, the great work of magic is sought." [Cut back to Abraxas. The Book disappears.] [Cut back to the girls. The Book drops out from the sky and lands on the rock.] Phoebe: Prue, Piper... Prue: Keep chanting. Prue, Piper, Phoebe: Hear now the words of the witches, the secrets we hid in the night..." (Abraxas appears above them) Abraxas: What are you doing? How'd you bring me here? Prue, Piper, Phoebe: "The oldest of Gods are invoked here, the great work of magic is sought. In this night, and in this hour we call upon the ancient power. In this night, and in this hour we call upon the ancient power. In this night, and in this hour we call upon the ancient power." (Abraxas is vanquished. Phoebe grabs the Book.) **Additional Decoder Input Omitted...**

You are given a script of a TV episode. Summarize the episode in a paragraph.

Summary:

**Model Outputs:**
LLAMA-2-CHAT output (ROUGE-L $= 19.1$): Prue, Piper, and Phoebe find an old Book of Shadows in the attic of their home, which they believe to be the original Book of Shadows. They start to read it, but Abraxas, the demon who wrote the book, appears to them, warning them to stop reading the book because he has returned to Earth and is seeking the power in the book to take over. The three sisters are able to defeat Abraxas and close the book, but the experience causes them to reflect on their lives as witches and their bond as sisters..
CAPE output with encoder contexts (ROUGE-L $= 19.3$): In this episode of Charmed, Prue is worried about the future because Andy is not coming back and they don't know what their new life will be like. Prue and Piper have a hard time coming to terms with the fact that their powers are no longer needed and they are losing their sense of purpose and identity. At the same time, they are struggling to cope with their feelings for each other and their relationship. Phoebe, meanwhile, has found a new source of power and is learning to control her abilities in a new way. As the three sisters work together to defeat an evil spirit that is threatening their home, they are able to come to terms with their powers and their relationship and find a new sense of purpose and identity.
Gold answers: The sisters face a demon named Abraxas , who steals the Book of Shadows and undoes some of the spells they've cast (in reverse order), releasing old vanquished enemies of the Charmed Ones and forcing them to remember the vanquishing spells from memory alone. The sisters have one chance to recapture the Book of Shadows or they will lose their powers forever. The sisters also meet their new neighbors, Jenny and her uncle Dan Gordon . Phoebe and Piper learn through the Wiccan community that because their anniversary of activating their inherent powers falls on an equinox , a wiccan holy day , each of their powers will be more developed and greatly magnified but only temporarily.

Table 9: ZeroSCROLLS generation example on the SummScreenFD dataset. CAPE sees the entire TV script through the decoder and the encoder, whereas LLAMA-2-CHAT only sees a 2K token window. For brevity, we omit part of the decoder input and only show 2 out of $k = 44$ encoder inputs for CAPE.

| | Long-context (Total Tokens) | | Retrieval-augmented ($k$) | |
|---|---|---|---|---|
| | **8K** | **32K** | **8** | **50** |
| CAPE | 3.97 | 3.91 | 4.57 | 4.55 |
| w/ RetDoc | 4.01 | 3.99 | **4.53** | **4.50** |
| w/ $\text{RP}_{\text{train-cat}}$ only | 4.01 | 3.96 | 4.56 | 4.54 |
| w/ $\text{RP}_{\text{train-filter}}$ only | **3.96** | **3.89** | 4.75 | 4.72 |
| w/ Frozen Encoder | 4.01 | 3.99 | 4.62 | 4.61 |
| w/ Random Encoder | 4.03 | 4.02 | 4.60 | 4.60 |
| w/ No Warmup | 4.03 | 4.02 | 4.61 | 4.61 |

Table 10: Test perplexity in long-context and retrieval-augmented language modeling, averaged over all datasets. Full results are in Table 11 and Table 12.

| | ArXiv | Book | PG19 | ProofPile | CodeParrot |
|---|---|---|---|---|---|
| **Total Tokens** $= 4096$ | | | | | |
| CAPE | 2.579 | 6.292 | 7.536 | 2.396 | 1.763 |
| w/ RetDoc | 2.649 | 6.340 | 7.586 | 2.465 | 1.775 |
| w/ RP Only | 2.633 | 6.335 | 7.604 | 2.446 | 1.766 |
| w/ AB Only | 2.569 | 6.287 | 7.525 | 2.386 | 1.772 |
| w/ Frozen Encoder | 2.631 | 6.353 | 7.603 | 2.446 | 1.785 |
| w/ Random Encoder | 2.680 | 6.374 | 7.617 | 2.488 | 1.797 |
| w/ No Copy Init | 2.678 | 6.372 | 7.613 | 2.487 | 1.796 |
| **Total Tokens** $= 8192$ | | | | | |
| CAPE | 2.496 | 6.049 | 7.372 | 2.219 | 1.715 |
| w/ RetDoc | 2.553 | 6.089 | 7.417 | 2.278 | 1.724 |
| w/ RP Only | 2.543 | 6.085 | 7.434 | 2.262 | 1.718 |
| w/ AB Only | 2.485 | 6.040 | 7.357 | 2.208 | 1.720 |
| w/ Frozen Encoder | 2.541 | 6.099 | 7.430 | 2.261 | 1.734 |
| w/ Random Encoder | 2.571 | 6.108 | 7.439 | 2.291 | 1.739 |
| w/ No Copy Init | 2.572 | 6.113 | 7.439 | 2.292 | 1.739 |
| **Total Tokens** $= 32768$ | | | | | |
| CAPE | 2.421 | 6.015 | 7.204 | 2.218 | 1.702 |
| w/ RetDoc | 2.546 | 6.088 | 7.280 | 2.332 | 1.726 |
| w/ RP Only | 2.497 | 6.059 | 7.271 | 2.288 | 1.709 |
| w/ AB Only | 2.396 | 5.995 | 7.178 | 2.195 | 1.702 |
| w/ Frozen Encoder | 2.520 | 6.091 | 7.282 | 2.297 | 1.739 |
| w/ Random Encoder | 2.571 | 6.108 | 7.303 | 2.346 | 1.752 |
| w/ No Copy Init | 2.571 | 6.110 | 7.301 | 2.346 | 1.752 |

Table 11: Test perplexity for all ablation settings in the long-context language modeling evaluation setting.

| | ArXiv | Book | C4-RP | CC | Github | StackEx | Wiki | Avg. |
|---|---|---|---|---|---|---|---|---|
| **Total Tokens** $= 2048$ ($k = 0$) | | | | | | | | |
| CAPE | 3.486 | 6.481 | 6.884 | 5.319 | 1.793 | 3.709 | 4.302 | 4.568 |
| w/ RetDoc | 3.413 | 6.399 | 6.854 | 5.263 | 1.788 | 3.694 | 4.287 | 4.528 |
| w/ RP Only | 3.485 | 6.479 | 6.901 | 5.313 | 1.777 | 3.700 | 4.281 | 4.562 |
| w/ AB Only | 3.505 | 6.504 | 7.185 | 5.444 | 1.859 | 4.018 | 4.763 | 4.754 |
| w/ Frozen Encoder | 3.501 | 6.495 | 6.933 | 5.505 | 1.821 | 3.734 | 4.323 | 4.616 |
| w/ Random Encoder | 3.426 | 6.442 | 6.904 | 5.541 | 1.838 | 3.728 | 4.338 | 4.602 |
| w/ No Copy Init | 3.452 | 6.459 | 6.914 | 5.546 | 1.842 | 3.732 | 4.344 | 4.613 |
| **Total Tokens** $= 7168$ ($k = 20$) | | | | | | | | |
| CAPE | 3.475 | 6.463 | 6.875 | 5.266 | 1.782 | 3.703 | 4.296 | 4.551 |
| w/ RetDoc | 3.413 | 6.393 | 6.839 | 5.169 | 1.779 | 3.693 | 4.286 | 4.510 |
| w/ RP Only | 3.479 | 6.467 | 6.894 | 5.249 | 1.767 | 3.696 | 4.276 | 4.547 |
| w/ AB Only | 3.491 | 6.481 | 7.140 | 5.401 | 1.846 | 4.004 | 4.738 | 4.729 |
| w/ Frozen Encoder | 3.485 | 6.482 | 6.930 | 5.500 | 1.815 | 3.727 | 4.318 | 4.608 |
| w/ Random Encoder | 3.426 | 6.442 | 6.904 | 5.540 | 1.837 | 3.727 | 4.337 | 4.602 |
| w/ No Copy Init | 3.447 | 6.457 | 6.913 | 5.545 | 1.841 | 3.721 | 4.332 | 4.608 |
| **Total Tokens** $= 14848$ ($k = 50$) | | | | | | | | |
| CAPE | 3.467 | 6.457 | 6.881 | 5.273 | 1.777 | 3.701 | 4.292 | 4.550 |
| w/ RetDoc | 3.413 | 6.392 | 6.835 | 5.098 | 1.776 | 3.692 | 4.287 | 4.499 |
| w/ RP Only | 3.472 | 6.465 | 6.900 | 5.243 | 1.762 | 3.693 | 4.274 | 4.544 |
| w/ AB Only | 3.480 | 6.471 | 7.114 | 5.412 | 1.838 | 3.994 | 4.719 | 4.718 |
| w/ Frozen Encoder | 3.474 | 6.474 | 6.930 | 5.509 | 1.814 | 3.723 | 4.316 | 4.606 |
| w/ Random Encoder | 3.426 | 6.442 | 6.904 | 5.540 | 1.837 | 3.726 | 4.337 | 4.602 |
| w/ No Copy Init | 3.445 | 6.456 | 6.913 | 5.545 | 1.841 | 3.716 | 4.329 | 4.606 |

Table 12: Test perplexity on RedPajama across all domains in the retrieval-augmented setting for all ablation experiments. $k$ is the number of additional contexts used. Avg. is the macro average across all domains.

| | Question Answering | | | Summarization | | |
|---|---|---|---|---|---|---|
| $c_{KL}$ | NQA | Qspr | QALT | GvRp | SSFD | QMSum |
| **Total Tokens** $= 4\text{K}$ | | | | | | |
| 2 | 19.5 | 20.5 | 30.2 | 16.5 | 16.4 | 19.6 |
| 1 | 21.6 | 20.7 | 27.2 | 16.3 | 5.3 | 4.7 |
| 0 | 21.3 | 21.0 | 27.4 | 14.6 | 14.9 | 15.6 |
| **Total Tokens** $= 16\text{K}$ | | | | | | |
| 2 | 20.6 | 19.9 | 29.6 | 15.9 | 16.8 | 19.4 |
| 1 | 21.8 | 20.6 | 26.8 | 16.0 | 15.2 | 16.1 |
| 0 | 22.9 | 20.6 | 26.4 | 14.8 | 5.2 | 4.7 |
| **Total Tokens** $= 32\text{K}$ | | | | | | |
| 2 | 21.6 | 19.9 | 29.6 | 15.8 | 16.7 | 19.5 |
| 1 | 22.7 | 20.6 | 26.8 | 16.0 | 15.2 | 15.8 |
| 0 | 22.3 | 20.6 | 26.4 | 14.6 | 5.2 | 4.7 |
| **All Tokens** | | | | | | |
| 2 | 21.9 | 19.9 | 29.6 | 15.9 | 16.7 | 19.5 |
| 1 | 22.6 | 20.6 | 26.8 | 15.9 | 15.2 | 15.8 |
| 0 | 23.0 | 20.6 | 26.4 | 14.6 | 5.2 | 4.7 |

Table 13: ZeroSCROLLS results using different losses during training, where $c_{KL}$ is the coefficient of the KL Divergence loss.