Pre-computed memory or on-the-fly encoding? A hybrid approach to retrieval augmentation makes the most of your compute

Michiel de Jong [* 1] **Yury Zemlyanskiy** [* 2] **Nicholas FitzGerald** [2] **Joshua Ainslie** [2]
Sumit Sanghai [2] **Fei Sha** [2] **William W. Cohen** [2]

Abstract

Retrieval-augmented language models such as Fusion-in-Decoder are powerful, setting the state of the art on a variety of knowledge-intensive tasks. However, they are also expensive, due to the need to encode a large number of retrieved passages. Some work avoids this cost by pre-encoding a text corpus into a memory and retrieving dense representations directly. However, pre-encoding memory incurs a severe quality penalty as the memory representations are not conditioned on the current input. We propose LUMEN, a hybrid between these two extremes, pre-computing the majority of the retrieval representation and completing the encoding on the fly using a live encoder that is conditioned on the question and fine-tuned for the task. We show that LUMEN significantly outperforms pure memory on multiple question-answering tasks while being much cheaper than FiD, and outperforms both for any given compute budget. Moreover, the advantage of LUMEN over FiD increases with model size.

1. Introduction

Retrieval-augmented language models such as Fusion-in-Decoder (Izacard & Grave, 2021) achieve strong performance on knowledge intensive tasks, often outperforming much larger models (Izacard et al., 2022). They retrieve related text passages and process the passages along with the input to extract relevant context information. However, encoding retrieved passages can be computationally expensive. Recent work has found that with an optimized decoder (Shazeer, 2019; de Jong et al., 2022a; Pope et al., 2022) encoding retrieved passages makes up the bulk of total cost
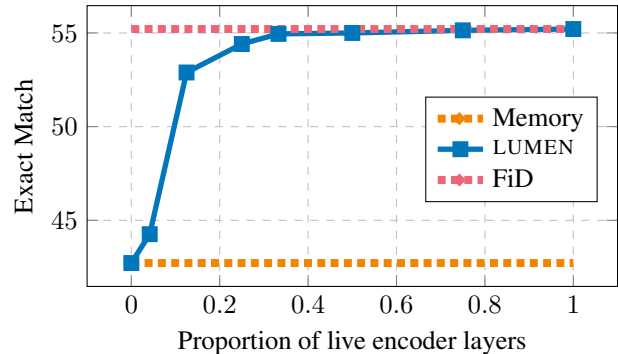

Figure 1. Exact match on Natural Questions dev set for LUMEN-XXL as a function of proportion of live (fine-tuned and conditioned on question) vs memory (pre-computed) encoder layers. LUMEN closes the gap between pure memory and FiD approaches with a fraction of live layers and therefore compute.

for finetuning and inference.

An increasingly common approach to reduce this encoding cost is to retrieve and extract information from a memory of pre-computed representations rather than raw text, amortizing the encoding of a passage over every sample that retrieves the passage entry from memory (Li et al., 2022; de Jong et al., 2022b; Wu et al., 2022a; Zhong et al., 2022; Chen et al., 2022; Wu et al., 2022b).[1]

However, memory approaches incur a large quality penalty relative to retrieval-augmented models (Li et al., 2022), because the pre-encoded memory is not conditioned on the task or on the particular input or question. Thus, the pre-encoded representation must be sufficiently comprehensive to answer *any* question, a challenging undertaking. The human analogue is the difference between memorizing an entire book and being quizzed afterwards compared to looking up the answer to a question on the fly.

Memory-based approaches therefore need to massively scale

---

[1]Here we do not refer to pre-computing representations used to select passages for retrieval (as is common practice for dense retrieval methods), but rather pre-computing the actual representations to be retrieved and incorporated into the language model.
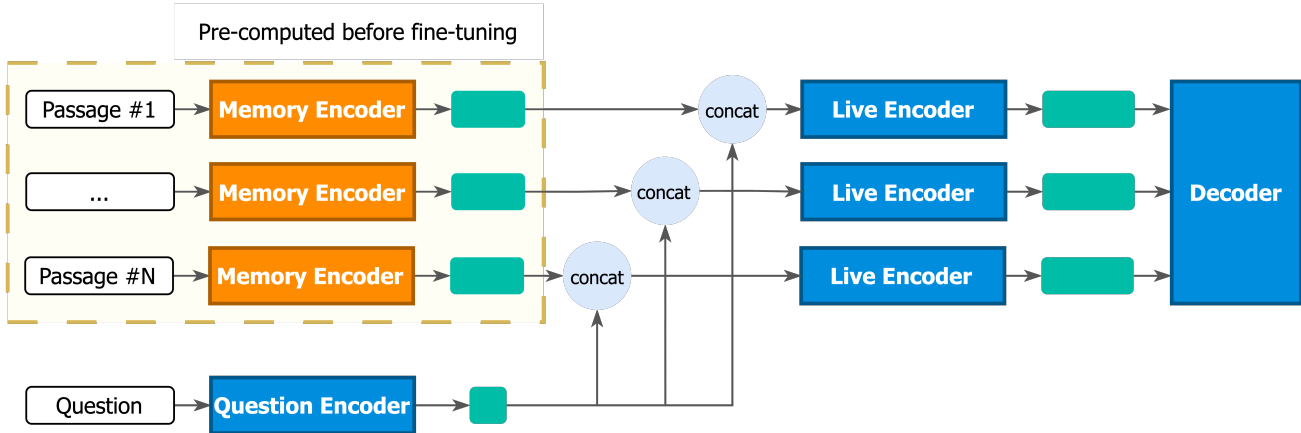
Figure 2. Overview of the LUMEN architecture. Before fine-tuning, each passage in the corpus is encoded by a memory encoder. While processing a sample, a question encoder first generates a representation of the question, which is then separately concatenated with each pre-computed passage representation. A fine-tuned live encoder then updates the passage representations conditioning on the question, which are finally fed into the decoder as in standard FiD. Frozen components are in orange, fine-tuned components in blue.

model size in order to achieve comparable performance. As we will show, this leads to higher overall net FLOPs due to cross-attention and decoding, as well as impractical increases in pre-training, pre-computation, and storage costs.

We propose LUMEN (Live Update Memory Network), a middle ground between retrieval and memory. LUMEN divides the task of encoding passages between a frozen memory encoder that pre-computes passage memory representations, and a fine-tuned live encoder that updates the memory representations conditioned on the question. Figure 2 provides a detailed overview of the architecture. As can be seen in Figure 1, a small proportion of live layers can already achieve performance close to standard Fusion-in-Decoder.

We start with a set of experiments initializing LUMEN from T5, partitioning the standard T5 encoder into a memory and live encoder. We evaluate on question-answering datasets Natural Questions (Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017). LUMEN achieves significantly stronger performance than FiD and FiD with memory given the same computational budget, and the advantage increases with model size. At T5-XXL size LUMEN performs comparably to FiD with only one third proportion of live layers and FLOPs.

Next, we experiment with improvements to the standard LUMEN setup, showing that the performance-compute trade-off can be further improved relative to FiD by transferring a trained memory and live encoder from a related task. In particular, we find that transfer from Natural Questions can close most of the gap in performance between LUMEN and FiD with smaller live encoders or on small datasets. Ultimately, LUMEN represents a desirable trade-off between retrieval and memory-based approaches, achieving better

performance for any given computational budget.

2. Background

We are interested in achieving the best possible performance for any given resource budget. However, there are different types of computational resources, and varying algorithmic approaches yield distinct trade-offs between those resources. In this section we provide background on existing retrieval-augmented models and describe the costs of those models along different computational dimensions.

2.1. Computational resources for retrieval-augmented models

The usual life-cycle of current models starts with pre-training, followed by fine-tuning on multiple tasks. Finally, the model is used for inference, either online or for batch distillation to a smaller model. Each of these stages features a different cost per sample. Let N_{pt}, N_{ft} and N_I be the number of processed samples for pre-training, fine-tuning and inference, and F_{pt}, F_{ft} and F_I the compute cost per sample for each stage, measured in FLOPs (floating point operations). Then the compute costs for the model are

$$\text{FLOPs}_{\text{pre-train}} = N_{pt}F_{pt}$$
$$\text{FLOPs}_{\text{fine-tune}} = N_{ft}F_{ft} \cdot \text{number of tasks}$$
$$\text{FLOPs}_{\text{inference}} = N_I F_I$$

The methods proposed in this paper are agnostic to the method used to select retrieved passages, so we do not consider retrieval method in our comparison of computational cost. While FiD inference can be slower than what FLOPs would indicate due to decoder memory bandwidth constraints (Hofstätter et al., 2022; de Jong et al., 2022a),
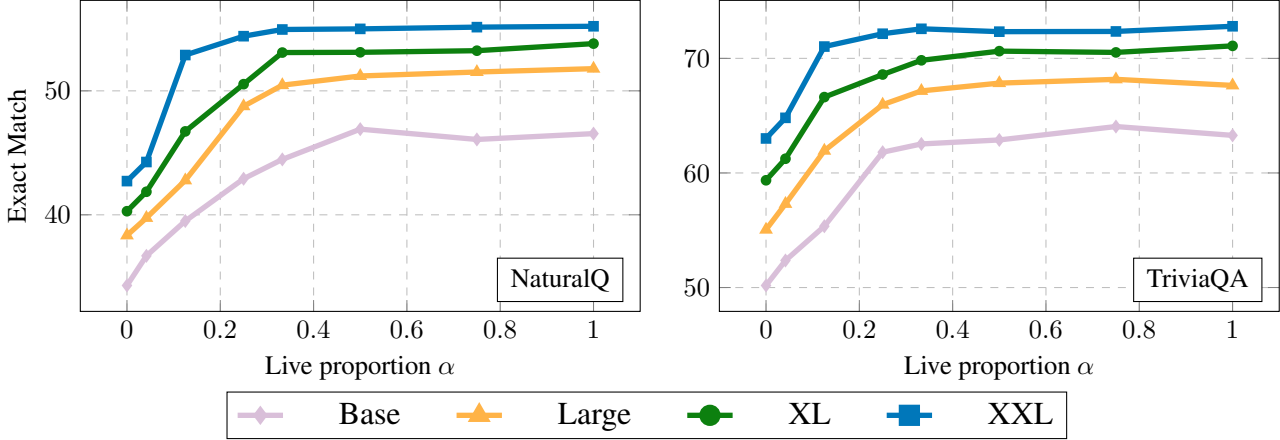
Figure 3. **MAIN RESULT: LUMEN achieves performance close to FiD with fraction of live layers. The required fraction decreases with scale.** Exact match on Natural Questions (NaturalQ) and TriviaQA validation sets as a function of proportion of live encoder layers for LUMEN Base, Large, XL, and XXL models.

modifying the decoder mitigates the gap (de Jong et al., 2022a). Hence, we use FLOPs our measure of computation cost in line with related work (Yu et al., 2022a; Varshney et al., 2022). Appendix B contains additional evidence on the relationship between FLOPs and latency.

For retrieval-augmented models there are additional costs. The retrieval set must be stored and retrievals transmitted to the accelerator. There may also be preprocessing overhead for the retrieval set, such as pre-computing memory representations. Let N_{rs} be the size of the retrieval set and F_{pc} the FLOPs associated with preprocessing a retrieval candidate. Then storage requirements and pre-computaton costs are given by

$$\text{Storage} = \text{Corpus size} \cdot \text{Size of a single sample}$$
$$\text{FLOPs}_{\text{precompute}} = \text{Corpus size} \cdot F_{\text{precompute}}$$

If retrieval representations are fine-tuned, then a different version of the retrieval set must be pre-computed and stored for each task. Required bandwidth for transmission is determined by the product of the number and size of retrieved representations.

2.2. Fusion-in-Decoder

Fusion-in-Decoder (Izacard & Grave, 2021) consists of a T5 encoder-decoder model. For each input, a number of relevant text passages are retrieved, and the input is prepended to each passage. The resulting passages are encoded separately by the encoder, and the encoded representations are then concatenated and attended to by the decoder to produce a target output. For each model, **fine-tuned** components are in blue and **frozen** components in orange.

$$G = \textbf{Dec}\Big[\textbf{Enc}(Q; \text{Passage}_1); \dots \textbf{Enc}(Q; \text{Passage}_k)\Big]$$

Let n_s be the number of source tokens, n_t the number of target tokens, L the number of layers, and d the dimension of the model. Following the analysis from de Jong et al. (2022a), the FLOPs for a single inference sample of FiD (ignoring attention score computation) is given by[2]

$$F_I = \underbrace{n_s \cdot L \cdot 14d^2}_{\text{Encoder and cross-attention}} + \underbrace{n_t \cdot L \cdot 14d^2}_{\text{Decoder}}$$

Appendix A discusses the derivation of this complexity in greater detail. F_{pt} and F_{ft} equal $3F_I$ due to the backward step. For fine-tuning and inference $n_s \gg n_t$ because of the large number of tokens from the retrieved passages. As such, FiD's fine-tuning and inference FLOPs per sample are significantly higher than for pre-training. In contrast, storage and bandwidth requirements are low as the retrieval set consists of passages of raw tokens. FiD has no pre-computation costs.

2.3. Memory

An increasing number of works reduce the cost of retrieval-augmented models by pre-computing dense representations of retrieval candidates and storing them in a memory. One such work modifies FiD by pre-computing passage encoder representations and providing the input as a prefix to the decoder (Li et al., 2022). We refer it as MemoryFiD:

$$G = \textbf{Dec}\Big[Q; \textbf{MemEnc}(\text{Passage}_1); ..\textbf{MemEnc}(\text{Passage}_k)\Big]$$

MemoryFiD saves fine-tuning and inference compute at the expense of increased pre-computation, storage, and band-

---

[2]We approximate the FLOPS of the MLP block as $8d^2$, the FLOPs from the original Transformer MLP. The T5 MLP has dimension between $2.5d$ and $3d$ and three matrix multiplication operations including GEGLU, yielding total FLOPs close to 8d.
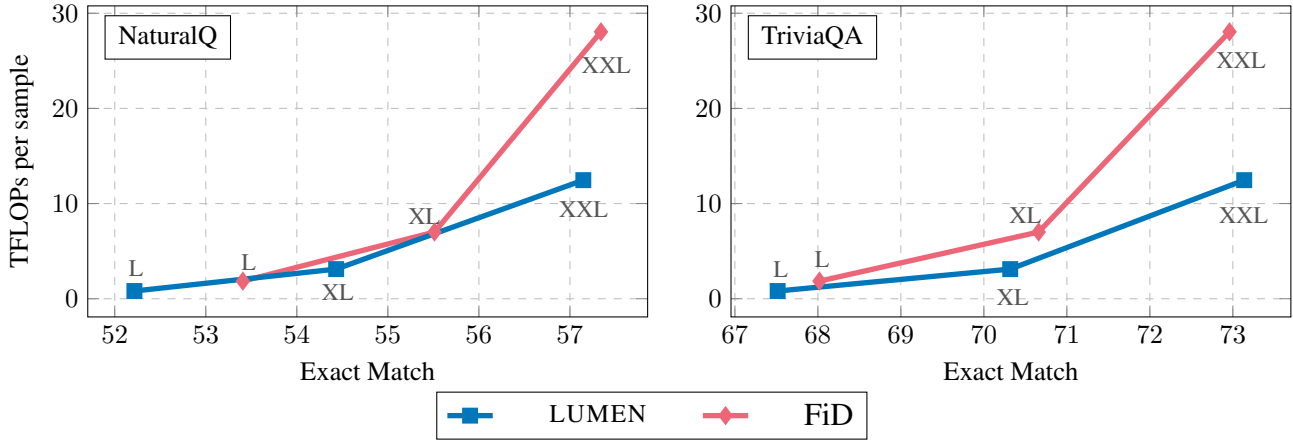
Figure 4. **MAIN RESULT: LUMEN uses significantly less compute than FiD for the same performance, and this advantage grows with scale.** TFLOPs as a function of exact match on Natural Questions (NaturalQ) and TriviaQA test sets. FLOPs are for single forward step and exclude pre-computation. Compares FiD and LUMEN with live proportion 0.33 Large, XL and XXL models. Lower is better.

width requirements. Because MemoryFiD does not encode retrieved passages on the fly, encoder costs are removed and only cross-attention and other decoder compute are left:

$$F_I = \underbrace{n_s \cdot L \cdot 2d^2}_{\text{Cross-attention}} + \underbrace{n_t \cdot L \cdot 14d^2}_{\text{Decoder}}$$

Instead, it pre-computes passage representations, using

$$\text{FLOPs}_{\text{precompute}} = \text{Corpus size} \cdot n_p L \cdot 12d^2$$

where n_p is the number of tokens in a single passage. MemoryFiD stores the final layer representations for each passage token, taking up

$$\text{Storage} = \text{Corpus size} \cdot 2n_p d$$

To give an indication of the storage overhead, applying MemoryFiD-XXL to Wikipedia requires approximately 16 terabytes. Holding model size fixed, MemoryFiD saves compute as long as the retrieval corpus is not too large relative to the number of samples processed for fine-tuning and inference. However, as passage representations are not conditioned on the question, MemoryFiD incurs a significant performance penalty relative to normal FiD. Therefore, in order to reach equivalent performance to standard FiD, MemoryFiD must use a much larger model, which incurs much larger cross-attention, decoder, pre-training, pre-computation, storage and bandwidth costs. Li et al. (2022) also fine-tune the memory encoder, which requires pre-computing and storing a separate memory for each task. This is intractable for real applications involving internet-sized corpora, so for our main results we assume the memory is pre-computed from a single model without fine-tuning on individual tasks. Without fine-tuning, the performance penalty is even higher. Figure 8 shows the effect of fine-tuning memory; LUMEN results are qualitatively similar in that setting.

Table 1. Model FLOPs per layer and storage bytes per token for FiD, MemoryFiD and LUMEN.

Model	FLOPs	Storage
FiD	$14n_s d^2 + 14n_t d^2$	10
MEMFiD	$2n_s d^2 + 14n_t d^2$	2d
LUMEN	$(2 + 12\alpha)n_s d^2 + 14n_t d^2$	2d

3. LUMEN

Intuitively when reading a passage it is helpful to know what information is needed and for what purpose. For Fusion-in-Decoder, this is achieved by prepending the input to retrieved passages and fine-tuning the passage encoder, whereas MemoryFiD does not enjoy such an advantage. With LUMEN, we explore the possibility that a similar effect can be achieved by a two-step process, in which a large model generates a general representation for each passage that can be placed in memory, and a smaller model transforms this general representation into an input-specific representation by conditioning on the input and task. Figure 2 provides an overview of the LUMEN architecture.

3.1. Architecture

LUMEN is initialized from a pre-trained T5 encoder-decoder model. The decoder functions the same as the standard FiD decoder, but LUMEN features three encoders. The T5 encoder is divided into a large **memory encoder** which contains the first $1 - \alpha$ proportion of layers, and a smaller **live encoder** with the remaining α proportion of layers. The memory encoder is applied offline to passages in the corpus to pre-compute memory representations, which are later
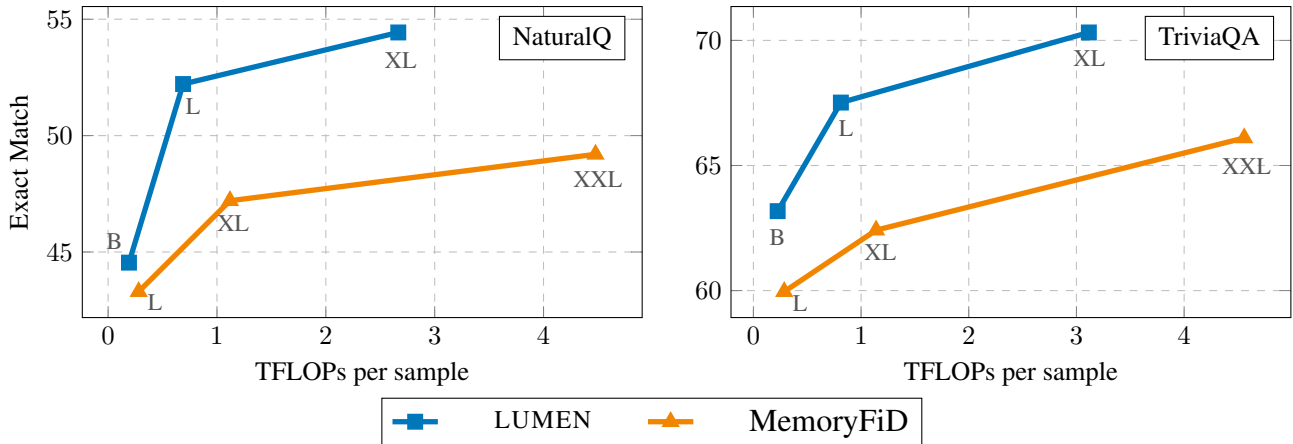
4

Figure 5. LUMEN **achieves much better performance than MemoryFiD at any compute budget.** Exact match performance on the test set of Natural Questions as a function of TFLOPs per sample comparing LUMEN $1/3$ Base, Large and XL models with MemoryFiD Large, XL, and XXL models. FLOPs are for single forward step and exclude pre-computation. Note that axes are transposed relative to Figure 4 as MemoryFiD requires too much compute to match LUMEN performance.

updated conditioned on input and task on the fly by the fine-tuned live encoder. In order to ensure that memory representations and input are compatible, LUMEN applies a **question encoder** to the input before prepending the question representation to the memory representation. The question encoder shares its structure and initial weights with the memory encoder, but is fine-tuned.

$$G = \textbf{Dec}\Big[Q; \textbf{LiveEnc}(H_1); \dots \textbf{LiveEnc}(H_k)\Big]$$

$$H_i = \Big[\textbf{QEnc}(Q); \ \textbf{MemEnc}(\text{Passage}_i)\Big]$$

Choosing $\alpha = 0$ recovers MemoryFiD, while $\alpha = 1$ yields a model very close to FiD.

3.2. Computational analysis

During fine-tuning and inference LUMEN applies only a proportion α of the layers, leading to a fraction α of FiD reader FLOPs for any given model size.

$$F_I = \underbrace{n_s \cdot \alpha L \cdot 12d^2}_{\text{Encoder}} + \underbrace{n_s \cdot L \cdot 2d^2}_{\text{Cross-attention}} + \underbrace{n_t \cdot L \cdot 14d^2}_{\text{Decoder}}$$

Pre-computation costs at the same model size are a factor $1 - \alpha$ of MemoryFiD pre-computation costs (*without* fine-tuning the memory encoder). Storage and bandwidth costs are the same as for MemoryFiD (at same model size and without fine-tuning the memory encoder). Table 1 shows a comparison of FLOPs and storage requirements of FiD, MemoryFiD and LUMEN. As we will show, LUMEN can match FiD performance with only a modest increase in size, leading to a large decrease in computational cost without the commensurate increases in pre-training, pre-computation, and storage requirements incurred with MemoryFiD.

4. Experiments

4.1. Experiment Setup

Training procedure All experiments use models based on the T5.1.1 architecture (Raffel et al., 2020). The main experiments use models initialized from the public T5 checkpoints (Google, 2022). FiD is trained according to the standard recipe (Izacard & Grave, 2021). For LUMEN, given proportion of live layers α, the memory encoder and question encoder are each initialized with the first 1 - α proportion of layers of the T5 encoder, and the live encoder is initialized with the last α proportion of layers of the T5 encoder.

Models are fine-tuned with the T5X framework (Roberts et al., 2022) based on JAX (Bradbury et al., 2018) and FLAX (Heek et al., 2020) using the Adafactor (Shazeer & Stern, 2018) optimizer with batch size 64 and learning rate 0.0001. Test results are generated from checkpoints with the best dev results. Experiments in Section 4.4 pre-train models from scratch. Pre-training follows the standard T5 training recipe except that we train for 500k steps, and disable the Adafactor second moment update schedule and factoring.

Data We evaluate LUMEN on open-domain question-answering datasets Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and WebQuestions (Berant et al., 2013) (in Section 4.3). For all datasets, each sample is paired with the 20 most relevant 100-word Wikipedia passages ranked by DPR (Karpukhin et al., 2020) score. For FiD, the concatenated question and passage pairs are truncated to 256 tokens. For LUMEN, the question and passage are individually truncated to 48 and 208 tokens to provide a fair comparison, as they are processed separately.
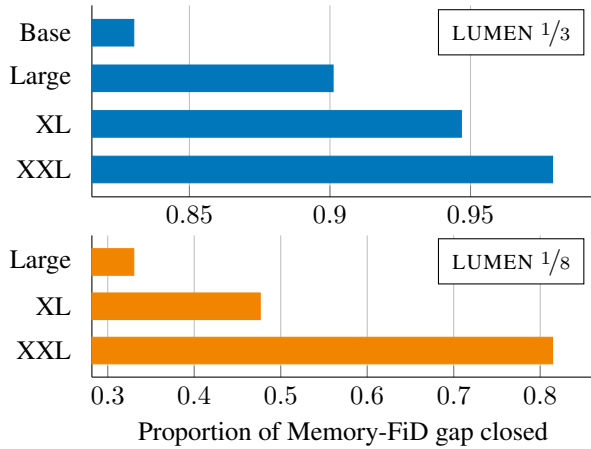
Figure 6. LUMEN **closes the gap with FiD as scale increases.** Proportion of exact match difference on Natural Questions between MemoryFiD and FiD closed by LUMEN as a function of model scale.

4.2. Main results

Figure 3 shows LUMEN performance as a function of live proportion for varying model sizes. The first key observation is that a relatively small proportion of live layers is sufficient to achieve quality close to FiD. The second key observation is that as the model size increases, the required live proportion to recover FiD performance decreases. This pattern is further supported by results from Figure 6, which explicitly measures how much of the gap between MemoryFiD and FiD is closed by LUMEN and shows this gap increases with scale.

Figure 4 compares FLOPs as a function of performance for LUMEN and FiD, demonstrating that LUMEN achieves similar performance at lower FLOPs for fine-tuning and inference (assuming pre-computation is sufficiently amortized to be effectively free). Moreover, the advantage becomes more pronounced with larger model size, consistent with the findings from Figure 3 and 6. Figure 5 shows that LUMEN also has much stronger performance than MemoryFiD for any FLOP value. Finally, Table 3 compares LUMEN with published results in the literature.

4.3. Transfer

Since the memory encoder is not fine-tuned on each individual task, the live encoder must adapt the memory representations to the task in addition to conditioning on the input. Especially for smaller live encoders, this may be difficult to learn while fine-tuning on a single task. Here we evaluate whether LUMEN can benefit from transferring from other knowledge-intensive tasks.

In particular, we consider two transfer settings. In the *Live* setting, we transfer the Live Encoder by training on Natural

Questions with frozen memory before transferring to the target task. In the *Memory* setting, the model is trained on Natural Questions with fine-tuned memory before transferring both the Live and Memory encoder to the target task. The *Memory* setting follows the intuition that, although it is infeasible to use a different memory for every task, it may be possible to perform multi-task fine-tuning before encoding memory.

Figure 7 shows the results of transfer from Natural Questions to TriviaQA and WebQuestions. We note several interesting patterns. *First*, gains from transfer are higher for smaller live proportion, with minimal gains for FiD and large gains for LUMEN 1/8. *Second*, transferring memory is only helpful for small live proportion, where the Live Encoder does not contain sufficient layers to fully adapt the memory to the task. *Third*, gains from transfer are significantly higher for WebQuestions, a task with a very small amount of data.

4.4. Memory shape

In our main experiments we initialize LUMEN from public T5 checkpoints to avoid costly pre-training and partition the encoder into a memory encoder and live encoder. Can we achieve a better a trade-off by pre-training a model with a custom configuration? Fixing the output of the live encoder to have low model dimension allows us to *scale the memory encoder without using more FLOPs*, as the cross-attention FLOPs are not affected by the size of the memory encoder. Table 2 shows the effect of adding a memory encoder consisting of 24 additional Base layers to an existing T5-Base configuration, yielding increasing performance without increasing compute. Taken to an extreme, these results suggest that combining a large language model with a moderately sized live encoder could yield strong results at modest cost.

Table 2. **Adding memory to FiD leads to significant performance gains without additional fine-tuning or inference FLOPs.** Exact match performance on Natural Questions and TriviaQA for FiD-Base and LUMEN 1/3 with Base decoder and live encoder, and memory encoder with 24 Base layers.

Model	NQ	TQA
FiD Base	47.3	64.4
LUMEN Base 24-12	**48.9**	**65.4**

4.5. Ablations

The two main differences between FiD, LUMEN, and MemoryFiD are the extent to which retrieved passages are conditioned on the input and the extent to which passage encoders are fine-tuned on particular tasks. Our first ablation investigates how performance differences between LUMEN and
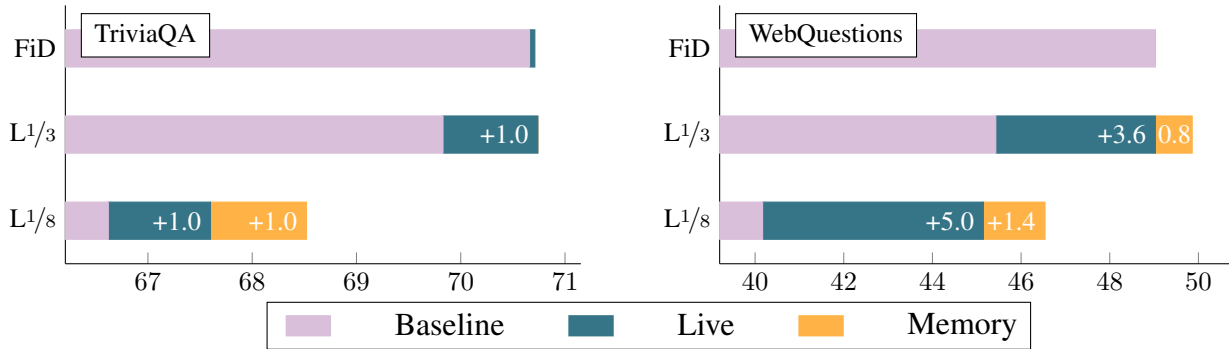
Figure 7. **Transferring memory and especially live encoder from a related dataset can partially close the gap with FiD, with increased gains for lower live proportion and smaller final dataset.** Exact match on TriviaQA and WebQuestions dev sets with and without transfer from Natural Questions for FiD and LUMEN XL models with live proportion $1/3$ and $1/8$. Live keeps the memory encoder frozen during training on Natural Questions while Memory also trains the memory on Natural Questions (still frozen after transfer). The gains from transfer are much more pronounced for smaller live proportion and on WebQuestions, the smaller dataset.
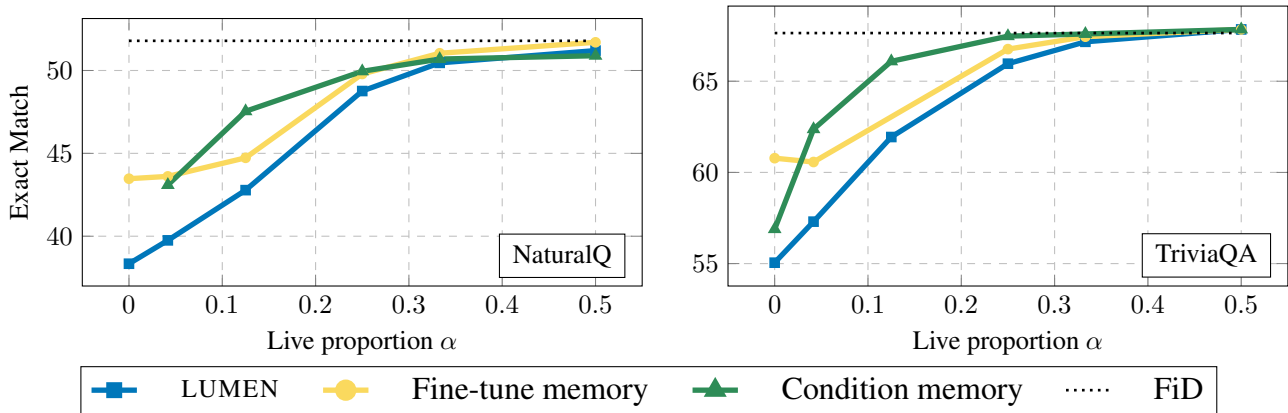

Figure 8. **Neither conditioning memory on input nor fine-tuning memory are sufficient to recover FiD performance. Both ingredients are important, although conditioning appears to contribute more.** Exact match on Natural Questions (NaturalQ) and TriviaQA dev sets as a function of proportion of live encoder layers for LUMEN-Large and two relaxations: one in which the memory layers are fine-tuned, and another in which the memory layers are conditioned on the question.

MemoryFiD on the one hand and FiD on the other hand result from conditioning on the input and fine-tuning. We construct two ablation settings as intermediate models between LUMEN and FiD: fine-tuning the memory encoder, and conditioning the memory encoder on the question (but without fine-tuning it). Figure 8 compares performance as a function of live proportion for these settings. Neither conditioning memory on the input nor fine-tuning the memory come close to recovering FiD performance by themselves: both are necessary. However, it seems that conditioning may be more helpful by itself than fine-tuning memory.

The LUMEN live encoder jointly processes concatenated passage and input representations. The decoder therefore receives passages conditioned on the input as well as the input on the passage. In order to disentangle these conditioning effects, we experiment with ablations that disallows question tokens to attend to the passage ("no q2mem") or

passage tokens to question ("no mem2q"). Figure 9 presents results that show that conditioning the passage on the input is critical, although the passage-conditioned question is still helpful.

Finally, LUMEN also uses a fine-tuned question encoder to generate a question representation that is optimized for the live encoder to condition the passage memories on. Figure 10 compares performance between fine-tuning and freezing this question encoder, demonstrating the importance of adapting the question encoder to the task.

5. Related Work

Retrieval-augmented models There is a significant amount of research on retrieval-augmented language models. Some notable approaches include REALM (Guu et al., 2020), RAG (Lewis et al., 2020), kNN-LM (Khandelwal et al., 2020), RETRO (Borgeaud et al., 2022), and Fusion-in-
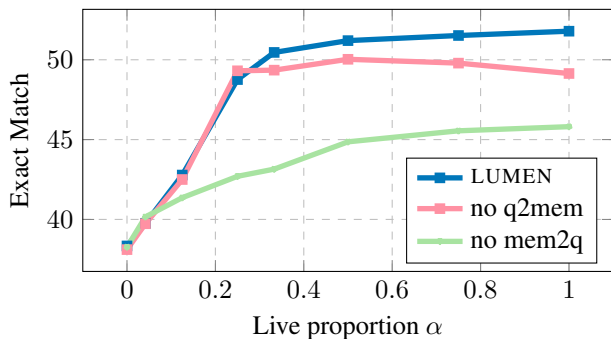
Figure 9. **The primary gains from the live encoder in LUMEN result from updating memory representations conditioned on the question.** Exact match on Natural Question dev set as a function of the proportion of live encoder layers for LUMEN-Large and two modifications with restricted encoder self-attention. In the 'no q2mem' setting question tokens cannot attend to passage tokens, and vice versa for 'no mem2q'.
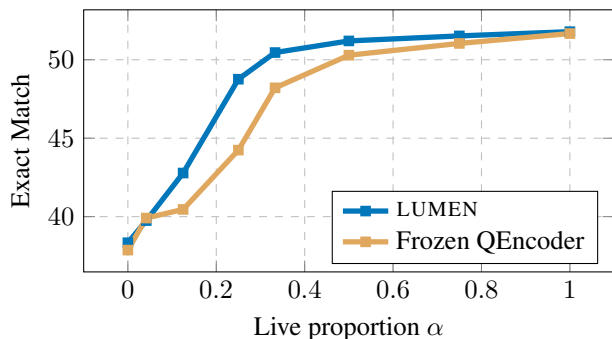

Figure 10. **Fine-tuning the question encoder improves performance significantly.** Exact match on Natural Question dev set as a function of the proportion of live encoder layers for LUMEN-Large and a modification for which the question encoder is frozen (so that the memory encoder and question encoder are shared).

Decoder (FiD) (Izacard & Grave, 2021). FiD in particular has demonstrated state of the art performance across a range of tasks (Izacard & Grave, 2021; Izacard et al., 2022; Yu et al., 2022b). This work focuses on improving the efficiency of FiD through a hybrid memory approach.

Efficient retrieval-augmented models Retrieval augmentation can be expensive for training and inference, and a large body of work investigates more efficient retrieval-augmented models. The computational cost of retrieval-augmented models can be partitioned into the cost from reading retrieved passages, decoding, and long-range attention. Recent work has shown that FiD spends the majority of inference time in the decoder (Hofstätter et al., 2022) due to memory bandwidth constraints in cross-attention (de Jong et al., 2022a). However, with the appropriate modifications (de Jong et al., 2022a) the constraint can be amelio-

Table 3. Comparison of LUMEN with published results on Natural Questions and TriviaQA test sets. We focus on comparing with FiD as other works enhance performance with improved retrieval (such as ATLAS), which is orthogonal and complementary to our contributions. LUMEN is agnostic to retrieval method, and can be used with ATLAS retrieval.

Model	NQ	TQA
REALM (Guu et al., 2020)	40.4	-
RAG (Lewis et al., 2020)	44.5	56.8
RETRO (Borgeaud et al., 2022)	45.5	-
T5-XXL (Roberts et al., 2020)	35.2	51.9
ATLAS (Izacard et al., 2022)	60.4	79.8
FiD-L (Izacard & Grave, 2021)	51.4	67.6
FiD-XXL (ours)	57.3	73.0
LUMEN-XXL	57.1	73.1

rated, after which the majority of training and inference costs result from reading retrieved passages.

The computational burden from encoding retrieved passages can be reduced by reranking and making use of only the best retrievals (Yu et al., 2022a; Wang et al., 2018; Mao et al., 2021). Alternatively, the resources devoted to retrieval can be adapted to the difficulty of the input, retrieving fewer or no passages if the model is confident it already knows the answer (Kratzwald & Feuerriegel, 2018; Varshney et al., 2022). In order to efficiently model interaction between different retrieved passages it is common to employ sparse long-range attention (Guo et al., 2022; Ainslie et al., 2020; Zemlyanskiy et al., 2021). Finally, there is a large body of work that attempts to improve the efficiency of Transformer models in general. Efficient fine-tuning methods (Zaken et al., 2022; Hu et al., 2022) update only a fraction of parameters during fine-tuning, but unlike LUMEN these methods still employ the full model for inference. Other general efficiency improvements include parallelization (Pope et al., 2022), quantization (Dettmers et al., 2022; Zeng et al., 2022), and distillation (Hinton et al., 2015; Gou et al., 2021).

Memory models LUMEN is most nearly related to the literature on *memory*. Another method to reduce encoding cost of retrieval-augmented models is to pre-compute representations for the retrieval corpus and collect these representations into a memory, thereby amortizing the encoding cost over all the instances for which a sample is retrieved. In particular, LUMEN is closely connected to Li et al. (2022), who propose a memory FiD model with pre-computed encoder representations. LUMEN can be seen as a hybrid of Li et al. (2022) and FiD that partially pre-computes encoder representations for efficiency, and finalizes the encoder representations on-the-fly conditioned on question and task to avoid the strong performance penalty from pre-computation.

This partial pre-computation is a form of late interaction, which has previously been used to improve the selection of retrieved passages (Khattab & Zaharia, 2020; Santhanam et al., 2022). We instead employ late interaction in the reader model. Milbauer et al. (2023) also employs late interaction in the reader model, but in the context of NLI as opposed to retrieval-augmented generation.

LUMEN uses memory in a straightforward manner, simply pre-computing token representations from a pre-trained model and retrieving passages with a standard dense passage retriever. Other memory models can be more involved, incorporating end-to-end retrieval within the model (de Jong et al., 2022b; Wu et al., 2022a), storing higher-level latent representations (de Jong et al., 2022b; Chen et al., 2022; Wu et al., 2022b), and specific pre-training for memory (de Jong et al., 2022b; Zhong et al., 2022). The main idea behind LU-MEN to update retrieved memory representations conditioning on the input is complementary to and can be combined with these more complex memory models.

6. Conclusion

Retrieval-augmented language models such as Fusion-in-Decoder are powerful but expensive. Pre-computing encoder representations into dense memory, a popular method for reducing computation costs of retrieval-augmented models, leads to a sharp decrease in performance. We propose LUMEN, a hybrid between Fusion-in-Decoder and dense memory. Passage representations are partially pre-encoded into a dense memory, and then reprocessed on the fly by a fine-tuned encoder that conditions on the question. We show that LUMEN achieves stronger performance for the same FLOPs, and that this advantage increases with scale.

Acknowlegements

References

Ainslie, J., Ontañón, S., Alberti, C., Cvicek, V., Fisher, Z., Pham, P., Ravula, A., Sanghai, S., Wang, Q., and Yang, L. ETC: encoding long and structured inputs in transformers. In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 268–284. Association for Computational Linguistics, 2020. doi: 10.18653/v1/ 2020.emnlp-main.19. URL `https://doi.org/10. 18653/v1/2020.emnlp-main.19`.

Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., and Sanghai, S. Gqa: Training generalized multi-query transformer models from multi-head checkpoints, 2023.

Berant, J., Chou, A., Frostig, R., and Liang, P. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1533–1544. ACL, 2013. URL `https://aclanthology.org/D13-1160/`.

Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., van den Driessche, G., Lespiau, J., Damoc, B., Clark, A., de Las Casas, D., Guy, A., Menick, J., Ring, R., Hennigan, T., Huang, S., Maggiore, L., Jones, C., Cassirer, A., Brock, A., Paganini, M., Irving, G., Vinyals, O., Osindero, S., Simonyan, K., Rae, J. W., Elsen, E., and Sifre, L. Improving language models by retrieving from trillions of tokens. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S. (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2206–2240. PMLR, 2022. URL `https://proceedings.mlr.press/ v162/borgeaud22a.html`.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Chen, W., Verga, P., de Jong, M., Wieting, J., and Cohen, W. W. Augmenting pre-trained language models with qa-memory for open-domain question answering. *CoRR*, abs/2204.04581, 2022. doi: 10.48550/arXiv. 2204.04581. URL `https://doi.org/10.48550/ arXiv.2204.04581`.

de Jong, M., Zemlyanskiy, Y., Ainslie, J., FitzGerald, N., Sanghai, S., Sha, F., and Cohen, W. Fido: Fusion-in-decoder optimized for stronger performance and faster inference. *arXiv preprint arXiv:2212.08153*, 2022a.

de Jong, M., Zemlyanskiy, Y., FitzGerald, N., Sha, F., and Cohen, W. W. Mention memory: incorporating textual

knowledge into transformers through entity mention attention. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022b. URL `https://openreview.net/forum?id=OY1A8ejQgEX`.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *CoRR*, abs/2208.07339, 2022. doi: 10.48550/arXiv.2208.07339. URL `https://doi.org/arXiv.2208.07339`.

Google. Pre-trained t5 models. `https://github.com/google-research/t5x/blob/main/docs/models.md`, 2022. Accessed: 2022-12-20.

Gou, J., Yu, B., Maybank, S. J., and Tao, D. Knowledge distillation: A survey. *Int. J. Comput. Vis.*, 129(6):1789–1819, 2021. doi: 10.1007/s11263-021-01453-z. URL `https://doi.org/10.1007/s11263-021-01453-z`.

Guo, M., Ainslie, J., Uthus, D. C., Ontañón, S., Ni, J., Sung, Y., and Yang, Y. Longt5: Efficient text-to-text transformer for long sequences. In Carpuat, M., de Marneffe, M., and Ruíz, I. V. M. (eds.), *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pp. 724–736. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.findings-naacl.55. URL `https://doi.org/10.18653/v1/2022.findings-naacl.55`.

Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M. REALM: retrieval-augmented language model pretraining. *CoRR*, abs/2002.08909, 2020. URL `https://arxiv.org/abs/2002.08909`.

Heek, J., Levskaya, A., Oliver, A., Ritter, M., Rondepierre, B., Steiner, A., and van Zee, M. Flax: A neural network library and ecosystem for JAX, 2020. URL `http://github.com/google/flax`.

Hinton, G. E., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015. URL `http://arxiv.org/abs/1503.02531`.

Hofstätter, S., Chen, J., Raman, K., and Zamani, H. Fidlight: Efficient and effective retrieval-augmented text generation. *CoRR*, abs/2209.14290, 2022. doi: 10.48550/arXiv.2209.14290. URL `https://doi.org/10.48550/arXiv.2209.14290`.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL `https://openreview.net/forum?id=nZeVKeeFYf9`.

Izacard, G. and Grave, E. Leveraging passage retrieval with generative models for open domain question answering. In Merlo, P., Tiedemann, J., and Tsarfaty, R. (eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pp. 874–880. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.eacl-main.74. URL `https://doi.org/10.18653/v1/2021.eacl-main.74`.

Izacard, G., Lewis, P., Lomeli, M., Hosseini, L., Petroni, F., Schick, T., Dwivedi-Yu, J., Joulin, A., Riedel, S., and Grave, E. Few-shot learning with retrieval augmented language models. *CoRR*, abs/2208.03299, 2022. doi: 10.48550/arXiv.2208.03299. URL `https://doi.org/10.48550/arXiv.2208.03299`.

Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In Barzilay, R. and Kan, M. (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pp. 1601–1611. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1147. URL `https://doi.org/10.18653/v1/P17-1147`.

Karpukhin, V., Oguz, B., Min, S., Lewis, P. S. H., Wu, L., Edunov, S., Chen, D., and Yih, W. Dense passage retrieval for open-domain question answering. In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 6769–6781. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.550. URL `https://doi.org/10.18653/v1/2020.emnlp-main.550`.

Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., and Lewis, M. Generalization through memorization: Nearest neighbor language models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=HklBjCEKvH`.

Khattab, O. and Zaharia, M. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In Huang, J. X., Chang, Y., Cheng, X., Kamps, J., Murdock, V., Wen, J., and Liu, Y. (eds.), *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China,*

July 25-30, 2020, pp. 39–48. ACM, 2020. doi: 10.1145/3397271.3401075. URL https://doi.org/10.1145/3397271.3401075.

Kratzwald, B. and Feuerriegel, S. Adaptive document retrieval for deep question answering. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J. (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 576–581. Association for Computational Linguistics, 2018. doi: 10.18653/v1/d18-1055. URL https://doi.org/10.18653/v1/d18-1055.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A. P., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M., Dai, A. M., Uszkoreit, J., Le, Q., and Petrov, S. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://doi.org/10.1162/tacl_a_00276.

Lewis, P. S. H., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Li, Z., Guo, R., and Kumar, S. Decoupled context processing for context augmented language modeling. *CoRR*, abs/2210.05758, 2022. doi: 10.48550/arXiv.2210.05758. URL https://doi.org/10.48550/arXiv.2210.05758.

Mao, Y., He, P., Liu, X., Shen, Y., Gao, J., Han, J., and Chen, W. Reader-guided passage reranking for open-domain question answering. In Zong, C., Xia, F., Li, W., and Navigli, R. (eds.), *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pp. 344–350. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.findings-acl.29. URL https://doi.org/10.18653/v1/2021.findings-acl.29.

Milbauer, J. L., Louis, A., Hosseini, J., Fabrikant, A., Metzler, D., and Schuster, T. Lait: Efficient multi-segment encoding in transformers with layer-adjustable interaction. In *Proceedings of the Association for Computational Linguistics: ACL 2023*, 2023.

Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Levskaya, A., Heek, J., Xiao, K., Agrawal, S., and Dean, J. Efficiently scaling transformer inference. *CoRR*, abs/2211.05102, 2022. doi: 10.48550/arXiv.2211.05102. URL https://doi.org/10.48550/arXiv.2211.05102.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Roberts, A., Raffel, C., and Shazeer, N. How much knowledge can you pack into the parameters of a language model? In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 5418–5426. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.437. URL https://doi.org/10.18653/v1/2020.emnlp-main.437.

Roberts, A., Chung, H. W., Levskaya, A., Mishra, G., Bradbury, J., Andor, D., Narang, S., Lester, B., Gaffney, C., Mohiuddin, A., Hawthorne, C., Lewkowycz, A., Salcianu, A., van Zee, M., Austin, J., Goodman, S., Soares, L. B., Hu, H., Tsvyashchenko, S., Chowdhery, A., Bastings, J., Bulian, J., Garcia, X., Ni, J., Chen, A., Kenealy, K., Clark, J. H., Lee, S., Garrette, D., Lee-Thorp, J., Raffel, C., Shazeer, N., Ritter, M., Bosma, M., Passos, A., Maitin-Shepard, J., Fiedel, N., Omernick, M., Saeta, B., Sepassi, R., Spiridonov, A., Newlan, J., and Gesmundo, A. Scaling up models and data with t5x and seqio. *arXiv preprint arXiv:2203.17189*, 2022. URL https://arxiv.org/abs/2203.17189.

Santhanam, K., Khattab, O., Saad-Falcon, J., Potts, C., and Zaharia, M. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In Carpuat, M., de Marneffe, M., and Ruíz, I. V. M. (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pp. 3715–3734. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.naacl-main.272. URL https://doi.org/10.18653/v1/2022.naacl-main.272.

Shazeer, N. Fast transformer decoding: One write-head is all you need. *CoRR*, abs/1911.02150, 2019. URL http://arxiv.org/abs/1911.02150.

Shazeer, N. and Stern, M. Adafactor: Adaptive learning rates with sublinear memory cost. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4603–4611. PMLR, 2018. URL `http://proceedings.mlr.press/v80/shazeer18a.html`.

Varshney, N., Luo, M., and Baral, C. Can open-domain QA reader utilize external knowledge efficiently like humans? *CoRR*, abs/2211.12707, 2022. doi: 10.48550/arXiv.2211.12707. URL `https://doi.org/10.48550/arXiv.2211.12707`.

Wang, S., Yu, M., Guo, X., Wang, Z., Klinger, T., Zhang, W., Chang, S., Tesauro, G., Zhou, B., and Jiang, J. R^3: Reinforced ranker-reader for open-domain question answering. In McIlraith, S. A. and Weinberger, K. Q. (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 5981–5988. AAAI Press, 2018. URL `https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16712`.

Wu, Y., Rabe, M. N., Hutchins, D., and Szegedy, C. Memorizing transformers. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022a. URL `https://openreview.net/forum?id=TrjbxzRcnf-`.

Wu, Y., Zhao, Y., Hu, B., Minervini, P., Stenetorp, P., and Riedel, S. An efficient memory-augmented transformer for knowledge-intensive NLP tasks. *CoRR*, abs/2210.16773, 2022b. doi: 10.48550/arXiv.2210.16773. URL `https://doi.org/10.48550/arXiv.2210.16773`.

Yu, D., Zhu, C., Fang, Y., Yu, W., Wang, S., Xu, Y., Ren, X., Yang, Y., and Zeng, M. Kg-fid: Infusing knowledge graph in fusion-in-decoder for open-domain question answering. In Muresan, S., Nakov, P., and Villavicencio, A. (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 4961–4974. Association for Computational Linguistics, 2022a. doi: 10.18653/v1/2022.acl-long.340. URL `https://doi.org/10.18653/v1/2022.acl-long.340`.

Yu, W., Iter, D., Wang, S., Xu, Y., Ju, M., Sanyal, S., Zhu, C., Zeng, M., and Jiang, M. Generate rather than retrieve:

Large language models are strong context generators. *CoRR*, abs/2209.10063, 2022b. doi: 10.48550/arXiv.2209.10063. URL `https://doi.org/10.48550/arXiv.2209.10063`.

Zaken, E. B., Goldberg, Y., and Ravfogel, S. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In Muresan, S., Nakov, P., and Villavicencio, A. (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 1–9. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.acl-short.1. URL `https://doi.org/10.18653/v1/2022.acl-short.1`.

Zemlyanskiy, Y., Ainslie, J., de Jong, M., Pham, P., Eckstein, I., and Sha, F. Readtwice: Reading very large documents with memories. In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tür, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y. (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pp. 5189–5195. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.408. URL `https://doi.org/10.18653/v1/2021.naacl-main.408`.

Zeng, A., Liu, X., Du, Z., Wang, Z., Lai, H., Ding, M., Yang, Z., Xu, Y., Zheng, W., Xia, X., Tam, W. L., Ma, Z., Xue, Y., Zhai, J., Chen, W., Zhang, P., Dong, Y., and Tang, J. GLM-130B: an open bilingual pre-trained model. *CoRR*, abs/2210.02414, 2022. doi: 10.48550/arXiv.2210.02414. URL `https://doi.org/10.48550/arXiv.2210.02414`.

Zhong, Z., Lei, T., and Chen, D. Training language models with memory augmentation. *CoRR*, abs/2205.12674, 2022. doi: 10.48550/arXiv.2205.12674. URL `https://doi.org/10.48550/arXiv.2205.12674`.

A. Complexity Derivation

Table 1 shows the FLOPs per layer for FiD, MemoryFiD and LUMEN. We go into more detail on those values here. For background, we have a source input of length n_s, a target output of length n_t, and a model with L layers and dimensionality d. A linear layer from k to l uses kl FLOPs.

Each Transformer layer consists of an attention layer, a feed-forward layer, and (in case of the decoder) a cross-attention layer. Each attention layer performs query, key, value and output projections, as well as attention score and value computations. The projections map from d to d and use d^2

FLOPs per token, totaling $4d^2$ per token. The feedforward layer of a vanilla Transformer maps each token from d to $4d$ and back, consuming $8d^2$ FLOPs per token.

The attention score and value computation involve taking the inner product of $n_q \cdot n_v$ pairs of vectors of dimension d, totaling $n_q n_v d$ FLOPs each. For an FiD encoder layer, the number of queries is equal to the source length, while the number of values is equal to the length of each passage and question (since FiD encodes each passage separately). Therefore the attention score and value computation total $2n_s n_p d$ FLOPs, leading to total encoder layer FLOPs of

$$12n_s d^2 + 2n_s n_p d$$

Since retrieved passages are short relative to model dimension ($n_p << d$), especially for larger models, the attention score computation term is very small and we ignore it in our estimate. The derivation for the decoder is similar, except for the presence of cross-attention layers. For cross-attention, the key and value projections apply to the source input and the query and output projections apply to the target output. Therefore, decoder layer FLOPs are given by

$$12n_t d^2 + 2n_t d^2 + 2n_s d^2 + 2n_t n_s d$$

where the attention computation term is again negligible. Totaling FLOPs from both components, we get FLOPs per layer of

$$14n_s d^2 + 14n_t d^2$$

Finally, MemoryFiD simply removes the encoder FLOPs, while LUMEN multiplies encoder FLOPs by live proportion α, yielding the figures in Table 1.

B. FLOPs vs Latency

Our main experimental results measure computational cost in terms of FLOPs. However, depending on the model configuration and hardware setting, inference can be slower than FLOPs would suggest due to memory bandwidth constraints. In this section we show that LUMEN FLOPs efficiency gains translate to actual throughput improvements.

We consider two settings. First, we investigate how LUMEN affects training speed, as memory bandwidth is primarily a bottleneck for autoregressive inference. Second, FiDO (de Jong et al., 2022a) showed that the memory bandwidth overhead from decoder inference can be drastically reduced with minimal reduction in quality through layer-sparse cross-attention or multi-query attention. At the time of our main experiments we did not have access to T5 checkpoints with multi-query attention, but in production LUMEN would be deployed with multi-query attention. Moreover, recent work has shown that multi-query checkpoints

can be efficiently obtained from existing multi-head checkpoints (Ainslie et al., 2023). Therefore, we report inference measurements for T5 with multi-query attention.

Figure 11 shows latency as a function of live proportion, demonstrating that latency varies nearly linearly with the number of remaining live layers. Figure 12 instead shows the trade-off between latency and quality, yielding the same pattern as Figure 1.
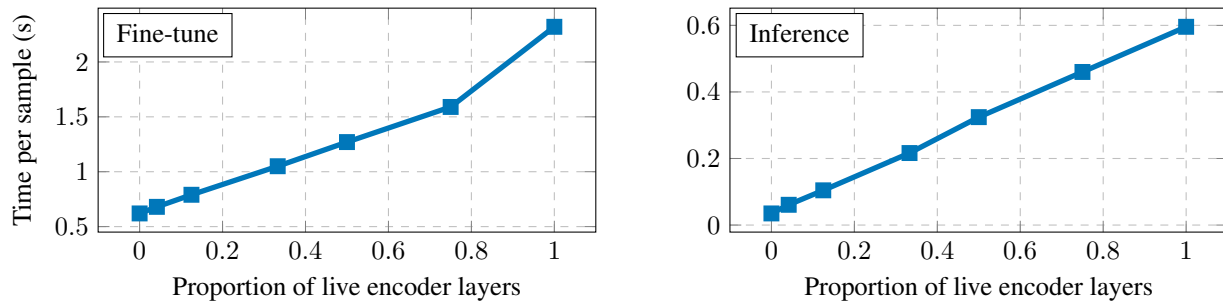
Figure 11. Fine-tuning time per sample for LUMEN-XXL as a function of live proportion.
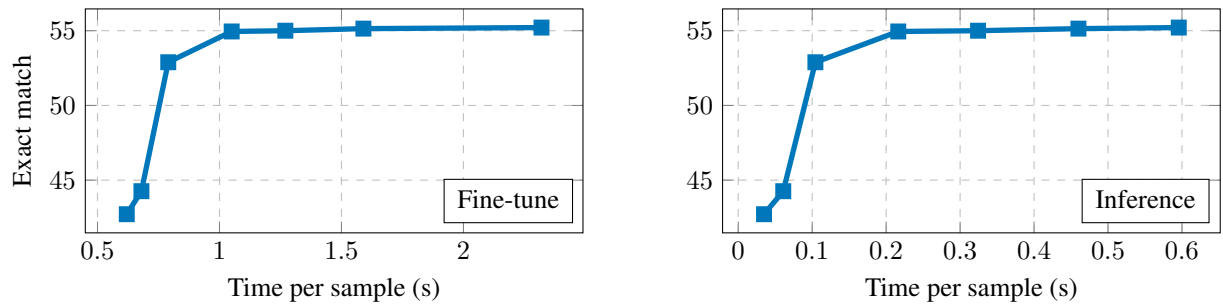

Figure 12. Exact match on Natural Questions as a function of inference time per sample for LUMEN-XXL, sweeping over live proportion. Assumes multi-query for inference measurements.