# FiDO: Fusion-in-Decoder optimized for stronger performance and faster inference

**Michiel de Jong**[*†] **Yury Zemlyanskiy**[‡] **Joshua Ainslie**[‡] **Nicholas FitzGerald**[‡]
**Sumit Sanghai**[‡] **Fei Sha**[‡] **William W. Cohen**[‡]
[†] University of Southern California, [‡] Google Research

## Abstract

Fusion-in-Decoder (FiD) is a powerful retrieval-augmented language model that sets the state-of-the-art on many knowledge-intensive NLP tasks. However, the architecture used for FiD was chosen by making minimal modifications to a standard T5 model, which our analysis shows to be highly suboptimal for a retrieval-augmented model. In particular, FiD allocates the bulk of FLOPs to the encoder, while the majority of inference time results from memory bandwidth constraints in the decoder. We propose two simple changes to the FiD architecture to alleviate memory bandwidth constraints, and speed up inference by 7x. This allows us to use a much larger decoder at modest cost. We denote FiD with the above modifications as FiDO, and show that it strongly improves performance over existing FiD models for a wide range of inference budgets. For example, FiDO-Large-XXL performs faster inference than FiD-Base and achieves better performance than FiD-Large.

## 1 Introduction

A large body of work has demonstrated that language model performance on downstream tasks can be improved by augmenting the model with relevant retrieved text (Guu et al., 2020; Lewis et al., 2020; Izacard and Grave, 2021; Izacard et al., 2022). In particular, the Fusion-in-Decoder (FiD) architecture (Izacard and Grave, 2021) stands out for strong performance, even outperforming much larger models on many knowledge-intensive tasks (Izacard et al., 2022). However, FiD uses a standard T5 encoder-decoder architecture (Raffel et al., 2020) which was not designed for use as a retrieval-augmented model. In this work we propose FiDO, a modified FiD architecture optimized for the retrieval-augmented setting.

[*]Correspondence to msdejong@usc.edu. Work done at Google Research.

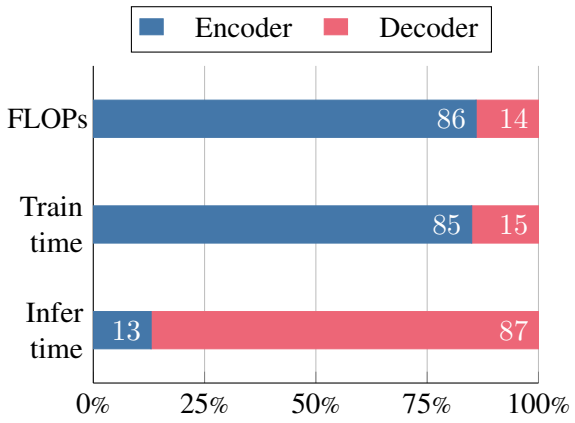


Figure 1: Shows the percentage of FLOPs in forward pass, training time and inference time for the encoder and decoder for a Fusion-in-Decoder model with 40 retrieved passages and batch size 24. The vast majority of FLOPs and training time originate from the encoder, but the decoder is much more expensive for inference.

The FiD decoder is responsible for a difficult task, assimilating information from many passages and reasoning over the information to generate an output. However, because the encoder and decoder are similar size and the encoder is applied to a large number of retrieved passages, FiD devotes an order of magnitude more Floating Point Operations (FLOPs) to the encoder than the decoder. In spite of this, the majority of inference time is actually spent in the decoder, as has been observed in prior work (Hofstätter et al., 2022). This surprising result is shown in Figure 1. Our analysis finds that for typical inference settings the FiD decoder is memory-bandwidth bound (Williams et al., 2009) due to using multi-head cross-attention (Vaswani et al., 2017) over a large input sequence.

Based on this analysis, we propose two sets of architectural changes. We first propose to reduce the cost of cross-attention over retrieved passages by removing most cross-attention layers from the decoder. This reduces cost and yields much smaller losses in performance than FiD-Light (Hofstätter



*Findings of the Association for Computational Linguistics: ACL 2023*, pages 11534–11547

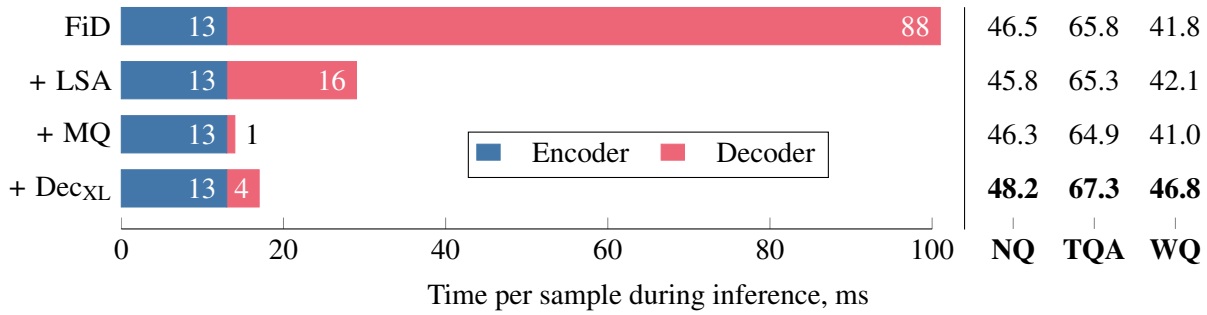


Figure 2: **MAIN RESULT. Layer-sparse cross-attention (LSA) and multi-query (MQ) attention eliminate the bulk of decoder inference cost with minor performance penalty, and the decoder can then be massively scaled up (Dec$_{XL}$) with only a modest increase in inference time.** To the left, encoder and decoder inference time per sample on a single TPUv4 with batch size 24 and 40 retrieved passages for variants of base-sized FiD model. To the right, corresponding exact match performance on Natural Questions (NQ), TriviaQA (TQA) and WebQuestions (WQ) dev sets.

et al., 2022), the best previously-proposed approach for optimizing FiD. We also replace multi-head attention with multi-query attention (Shazeer, 2019). With these modifications the memory-bandwidth bottleneck is eliminated: decoder inference is now orders of magnitude faster and most inference time is spent in the encoder, consistent with the balance of FLOPs between components.

Finally, we propose to partially rebalance compute towards the decoder by massively scaling decoder size, using a smaller encoder to extract information from retrieved passages and a larger decoder to assimilate the information and reason about the desired output. We refer to the resulting series of models as FiDO (Fusion in Decoder Optimized) and show that FiDO strongly outperforms standard FiD models on the question-answering datasets Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017) and WebQuestions (Berant et al., 2013) for a wide range of inference budgets and settings. Figure 2 summarizes some of these results.

## 2 Analysis

Retrieval-augmented models generally read many context tokens relative to the number of question or answer tokens, such that processing retrieved text consumes the bulk of FLOPs. However, past work has shown that most inference time for Fusion-in-Decoder (FiD) is spent in the decoder (Hofstätter et al., 2022). Our own experiments support this (Figure 1). This section investigates FiD's computational structure and decoder inference speed, and finds the slower decoder speed to be the result of memory bandwidth constraints, exacerbated by attention over retrieved documents.

### 2.1 Fusion-in-Decoder

The backbone of the Fusion-in-Decoder model (Izacard and Grave, 2021) is a T5 encoder-decoder architecture. The model is provided a question or other input, as well as a number of relevant retrieved text passages. The question is prepended to each retrieved passage, and then the encoder is applied to each passage separately. The resulting representations are concatenated. Finally, the decoder cross-attends to the large number of concatenated representations and assimilates the information from the different passages to generate an answer, hence Fusion-in-Decoder.

### 2.2 FLOPs of FiD model

Model speed is determined by the number of FLOPs and the speed at which computations are performed, typically measured in floating point operations per second (FLOP/s). Operations in a Transformer can be roughly divided into MLP layers, attention projection layers, and attention operations. For simplicity, we count only multiplication operations.

Let $d$ be the dimension of the model, $n_s$ the total number of tokens across all passages, $n_p$ the number of tokens in a single retrieved passage, $n_t$ the number of tokens in the target, $L$ the number of layers, and assume the MLP dimension is $4d$. The number of FLOPs used in an encoder layer is

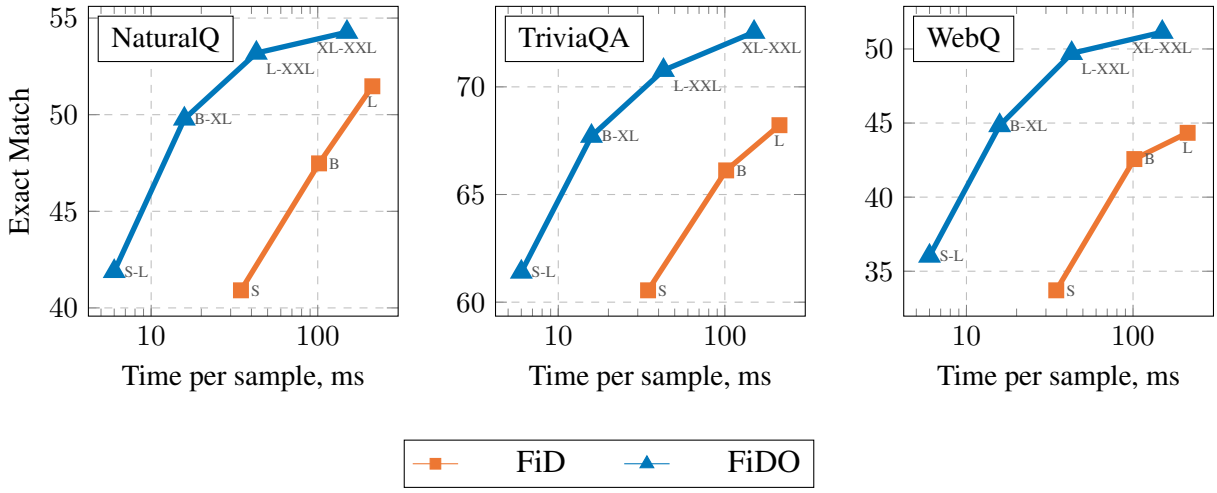


Figure 3: **MAIN RESULT. FiDO achieves much higher performance for any given inference budget.** Exact match on Natural Questions (NaturalQ), TriviaQA and WebQuestions (WebQ) test sets as a function of inference budget (log scale). Compares FiD Small, Base and Large models with FiDO Small-Large, Base-XL, Large-XXL and XL-XXL models.

approximately

$$\text{FLOPs}_{\text{enc}L} = \underbrace{8n_s d^2}_{\text{MLP}} + \underbrace{4n_s d^2}_{\text{QKVO projections}} + \underbrace{2n_s n_p d}_{\text{Attention}}$$

Since the size of each retrieved passage $n_p \ll d$, computation of the attention score is negligible and we can approximate total FLOPs in the encoder as

$$\text{FLOPs}_{\text{enc}} \approx 12 n_s d^2 \cdot L \quad (1)$$

Decoder layers additionally have cross-attention layers, leading to FLOPs of

$$\text{FLOPs}_{\text{dec}L} = \underbrace{8n_t d^2 + 4n_t d^2 + 2n_t^2 d}_{\text{MLP and Self-attention}} + \underbrace{2n_t d^2}_{\text{Cross-attention QO}} + \underbrace{2n_s d^2}_{\text{Cross-attention KV}} + \underbrace{2n_t n_s d}_{\text{Cross-attention}}$$

The output length $n_t \ll n_s, d$, so the only non-negligible term for decoder FLOPs originates from the cross-attention key and value projections, which cost the same FLOPs as encoder key and value projections. We see that the decoder consumes roughly $\frac{1}{6}$ the FLOPs of the encoder.

$$\text{FLOPs}_{\text{dec}} \approx 2 n_s d^2 \cdot L \quad (2)$$

Figure 1 shows that actual measured training time closely mirrors this FLOPs approximation. However, the decoder is much more expensive for inference. We argue below this is because the decoder is *memory bandwidth constrained* during inference, specifically the cross-attention layers.

### 2.3 Effective computational throughput

In order to perform computations, accelerators must transmit data between global memory and registers, which can be a limiting factor. The actual FLOP/s achieved can be usefully modeled with the *roofline* model (Williams et al., 2009; Ofenbeck et al., 2014; Mohan, 2018) as the lesser of peak FLOP/s the device is capable of and how fast required data can be transferred.

$$\text{Actual FLOP/s} = \min(\text{Peak FLOP/s}, \underbrace{\text{Operational Intensity}}_{\text{Operations per byte}} \cdot \underbrace{\text{Peak Memory Bandwidth}}_{\text{bytes per second}})$$

The data constraint is given by the product of device memory bandwidth – how fast data can be transferred – and *operational intensity* – how many operations are performed per unit of data. The latter is determined by an algorithm's degree of *data reuse*, the number of operations that can be performed before new data needs to be fetched.

High operational intensity is necessary for good performance on modern GPU/TPU hardware, for which peak FLOP/s are usually two orders of magnitude times larger than memory bandwidth (Google, 2022; NVIDIA, 2022). If operational intensity is too low, the accelerator will spend the majority of its time waiting for data to be transferred to registers. Usually, that happens when the model performs minor computations with large tensors repeatedly, for example in normalization layers or during incremental decoding.

### 2.4 Operational intensity of FiD inference

Shazeer (2019) shows that the speed of incremental Transformer decoding is memory-bandwidth bound due to low operational intensity. Here we follow their analysis and derive the asymptotic *inverse* of operational intensity – the ratio of memory operations to the compute performed during each incremental decoding step – for FiD. Let $b$ be the batch size, $h$ the number of attention heads and assume that attention heads have dimension $\frac{d}{h}$.

**Operational intensity of MLP layer.** For each token the linear projections perform $O(bd^2)$ operations, and load $O(bd + d^2)$ memory, where $bd$ corresponds to activations and $d^2$ to the weight matrices. During training, sequence length effectively multiplies batch size as weights need to be loaded only once for the entire sequence, but for inference each token is processed incrementally. The inverse operational intensity is then

$$\mathcal{R}^{\text{MLP}} = \frac{1}{b} + \frac{1}{d} \tag{3}$$

Therefore, obtaining high operational intensity of MLP layer ($\mathcal{R}^{\text{MLP}} \ll 1$) during inference requires a large batch size.

**Operational intensity of attention layers.** Memory bandwidth is a more severe bottleneck for attention inference, particularly cross-attention. At each decoding step the model applies projections for a single token, and has to load all cached key and value projections from encoder tokens and prior decoder tokens into memory. This leads to very low operational intensity.

Specifically, query/key/value/output projections for a single position take $O(bd^2)$ operations. As discussed earlier, we can ignore the attention computation itself. The model needs to load projection matrices ($O(d^2)$ memory) and past keys and values ($O(bnd)$ memory). Therefore, the inverse operational intensities for self-attention layers, $\mathcal{R}^{\text{S-MHA}}$ and cross-attention layers $\mathcal{R}^{\text{C-MHA}}$ are

$$\mathcal{R}^{\text{S-MHA}} = \frac{1}{b} + \frac{n_t}{d}, \quad \mathcal{R}^{\text{C-MHA}} = \frac{1}{b} + \frac{n_s}{d} \tag{4}$$

Because the source input length $n_s$ is extremely long for FiD, the cross-attention operational intensity is very low, which bottlenecks inference.

## 3 Method

We have shown that the encoder accounts for the bulk of FiD FLOPs and training cost, while FiD spends the majority of inference time in the decoder due to low operational intensity of cross-attention layers. Next we propose several ways to alleviate the decoder bottleneck. This allows us to efficiently allocate more compute to the decoder by scaling decoder size without significantly increasing the inference speed. We denote Fusion-in-Decoder with the proposed optimizations as FiDO (Fusion-in-Decoder Optimized).

| Model | Max Batch Size |
|---|---|
| Vanilla FiD | 24 |
| + LSA | 128 |
| + MQ | 256 |
| + XL Decoder | 128 |

Table 1: Maximum batch size for QA inference with 40 retrieved passages on a single TPUv4 for FiD Base models with different FiDO components.

| Model | Pre-training | Finetuning |
|---|---|---|
| Vanilla FiD | 219.9 | 9.7 |
| + LSA | 247.0 | 11.8 |
| + MQ | 248.0 | 11.8 |
| + XL Decoder | 81.9 | 6.9 |

Table 2: Pre-training and fine-tuning samples per second per chip for FiD Base model with varying FiDO components. We use 64 TPUv4 chips and batch size 2048 for pre-training and 32 chips and batch size 64 for fine-tuning. See Section 5.1 for training information.

### 3.1 Layer-sparse cross-attention

The decoder cross-attention layer is the primary bottleneck for inference due to its low operational intensity. FiD-Light (Hofstätter et al., 2022) improves the operational intensity by reducing the effective input length by a factor of $K$. We instead propose to remove cross-attention from some decoder layers entirely, keeping cross-attention only in one out of every $K$ decoder layers. We call this layer-sparse cross-attention (LSA). Section 5 provides evidence that LSA achieves similar speedups without FiD-Light's drop in quality. For FiDO we use LSA with sparsity $K = 6$, which means that a Large decoder has cross-attention only at layers 6, 12, 18 and 24. In principle LSA and FiD-Light can be combined, but we find that after applying LSA and multi-query attention the remaining cross-attention makes up a small proportion of decoder inference cost and further speedups from reducing

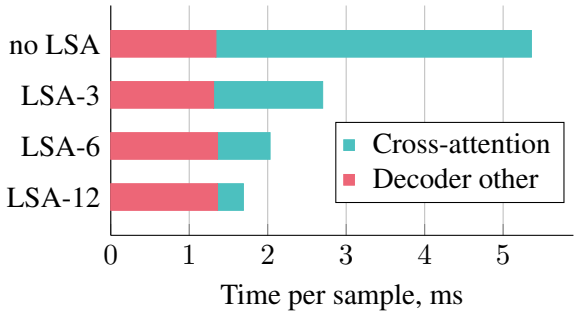


Figure 4: Cross-attention and total decoder inference time for FiDO Base-XL with varying factors of layer-sparse cross-attention. The main FiDO configuration uses LSA-6 which has cross-attention every 6 layers.

cross-attention are modest (Figure 4).

Removing cross-attention layers also reduces FiD's FLOPs and memory usage. Cross-attention layers make up approximately $\frac{1}{7}$ of total FiD FLOPs (see Eqn 2) and applying LSA-6 leads to a 12% reduction in FLOPs. Table 2 shows the reduction in FLOPs is reflected by an increase in training speed. Moreover, cross-attention keys and values make up a substantial proportion of memory usage during inference, and LSA-6 enables a much larger batch size (Table 1).

### 3.2 Multi-query attention

Shazeer (2019) proposes to increase the operational intensity of decoder attention layers by applying multi-query attention, in which keys and values share a single head each and only queries have multiple heads. With a single head, keys and values use a factor $h$ less memory and are much faster to load. With multi-query attention, keys and values occupy $O(bnd/h)$ memory, so that the inverse operational intensity of cross-attention becomes

$$\mathcal{R}^{\text{C-MQA}} = \frac{1}{b} + \frac{1}{d} + \frac{n_s}{dh} \tag{5}$$

which has the problematic term $\frac{n_s}{d}$ reduced by factor of $h$. Multi-query attention further reduces inference cost (Figure 2) and memory (Table 1) on top of layer-sparse cross-attention, though not training speed (Table 2).

### 3.3 Asymmetric Decoder

Section 5.4 showed that the FiD encoder consumes an order of magnitude more FLOPs than the decoder because the encoder and decoder are the same size but the encoder is applied to many more tokens. After applying layer-sparse cross-attention and multi-query attention, the decoder also takes up much less time for inference. Such an allocation may not be optimal, as the FiD decoder is responsible for a more challenging task than the standard T5 encoder: it has to assimilate and reason over information from many passages.

We propose to partially redress this imbalance through massively scaling the decoder up, by as much as 15x. Because the decoder is applied to fewer tokens, and because increased decoder dimension improves operational efficiency, such scaling only modestly increases inference cost. For example, Figure 2 shows that replacing the Base-sized decoder with an XL-sized decoder increases the total inference time per sample by only 21%. Fine-tuning costs also increase only modestly (Table 2). However, pre-training costs increase more (though still much less than the scaling factor of the decoder), as T5 pre-training uses a much smaller ratio of input length to output length. After reducing the decoder cross-attention memory costs scaling the decoder only mildly increases activation memory, so that FiDO can still fit much larger batch sizes than vanilla FiD (Table 1). For the FiDO method we use decoders that are typically two T5 sizes larger than the encoder: Small-Large, Base-XL, Large-XXL and XL-XXL (as XXL is the largest T5 model).

## 4 Related Work

**Retrieval-augmented models** There exists a large body of retrieval-augmented approaches. Some particularly well known models are REALM (Guu et al., 2020), RAG (Lewis et al., 2020), RETRO (Borgeaud et al., 2022) and Fusion-in-Decoder (Izacard and Grave, 2021). FiD in particular has achieved state-of-the-art performance on a wide variety of tasks (Izacard and Grave, 2021; Izacard et al., 2022; Yu et al., 2022b) and in this work we focus on improving the performance-efficiency trade-offs for FiD. RETRO is another closely related retrieval-augmented model, as it uses a small encoder for retrieved context and a larger primary decoder like FiDO does. Unlike RETRO, FiDO's efficiency improvements allow it to tractably attend to many retrieved passages with a much larger decoder.

**Efficient Transformers** Our work builds heavily on existing insights into neural network and particularly Transformer speed. Previous work has found that data movement is often a constrain-

| Model | Total TPS | Decoder TPS | NaturalQ | TriviaQA | WebQ |
|---|---|---|---|---|---|
| FiDO (base-XL) | 15.8 | 2.0 | 48.2 | 67.3 | 46.8 |
| no LSA | 19.2 | 5.4 | 47.9 | 67.4 | 46.3 |
| no MQ | 60.8 | 47.0 | 48.2 | 67.5 | 45.4 |
| no Asym (base-base) | 14.4 | 0.6 | 46.3 | 64.9 | 41.0 |

Table 3: Inference time per sample, decoder time per sample (ms) and downstream QA exact match for FiDO base-XL with different components ablated separately. FiDO is evaluated on dev sets for ablation results.

ing factor for computations on modern devices (Williams et al., 2009; Dao et al., 2022; Shazeer, 2019). Shazeer (2019) shows that autoregressive Transformers are particularly bandwidth bound during inference, and proposes multi-query attention as a partial solution. We find that this is exacerbated by the FiD setting, and adopt multi-query attention for FiDO to ameliorate the problem. Pope et al. (2022) also investigates multi-query attention, primarily in the context of efficient inference and parallelization for very large language models, whereas we focus on performance/cost trade-offs for the retrieval-augmented setting.

Another way to alleviate memory bandwidth constraints is to quantize model parameters and possibly activations (Dettmers et al., 2022; Zeng et al., 2022). Quantizing models reduces data that needs to be sent to device registers, and also reduces overall memory usage which allows for larger, more efficient batch sizes. Finally, it is possible to distill (Hinton et al., 2015; Gou et al., 2021) models into a smaller student model, which is cheaper for inference. However, knowledge distillation requires labeling a very large number of samples with the larger model, so reducing the inference costs of larger models is highly valuable.

**Efficient retrieval-augmented models** FiDO lies in a body of work that attempts to improve the efficiency of retrieval-augmented or long-input models. One direction focuses on reducing the cost of the attention mechanism. LongT5 (Guo et al., 2022) routes long-range attention through a small number of global tokens. FiD-Light (Hofstätter et al., 2022), the most closely related work to FiDO, employs a similar mechanism for FiD, as the decoder attends to only the first $\frac{1}{K}$ proportion of representations of each retrieved passage. We opt to introduce sparsity in attention layers as in ReadTwice (Zemlyanskiy et al., 2021) instead of attention patterns. FiDO applies cross-attention from the decoder to the encoder in one out of every K layers, which achieves a similar speedup to FiD-Light but with only minor performance penalty. FiDO also incorporates multi-query attention leading to a further order of magnitude reduction in decoder inference cost, and takes advantage of this to massively scale the decoder.

A different and complementary direction is to reduce the cost of reading retrieved passages. KG-FiD (Yu et al., 2022a) reranks retrieved passages and reads only the top passages, while Varshney et al. (2022) reads more retrieved passages only if it is not confident in its answer. Another approach is to pre-compute and store encoder representations in a memory and directly retrieve representations from memory, rather than re-encoding retrieved text (de Jong et al., 2022; Wu et al., 2022; Li et al., 2022). For standard FiD, the decoder actually makes up the bulk of the inference cost. FiDO reduces the cost of the decoder such that encoding retrieved passages becomes the bottleneck, increasing the benefit of the above approaches.

## 5 Experiments

### 5.1 Experiment Setup

**Pre-training** All models are based on the T5.1.1 architecture (Raffel et al., 2020), pre-trained from scratch on C4 (Dodge et al., 2021) using JAX (Bradbury et al., 2018), FLAX (Heek et al., 2020), and T5X (Roberts et al., 2022). We employ the standard T5 training recipe except for a modified Adafactor (Shazeer and Stern, 2018) optimizer. Appendix A describes training in greater detail.

**Downstream evaluation** We evaluate FiDO on open-domain question-answering datasets Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017) and WebQuestions (Berant et al., 2013). We report results on the open-domain QA splits from Lee et al. (2019). For all datasets, each sample is paired with a set of 100-word Wikipedia passages ranked by DPR (Karpukhin et al., 2020) score. The question is prepended to each retrieved

passage, and then truncated to 256 tokens. The experiments in the paper use 40 retrieved passages to balance performance and speed, but our results hold across a wide range of retrieved passages.

**Inference setup** For our main results we choose a setting that we believe is most representative for common use of retrieval-augmented models. We perform inference on a single TPUv4 and report inference time per sample (TPS) as measured by xprof (Google, 2020). We use a batch size of 64 (or the largest batch size that fits, if smaller) for the main experiments. Figure 1 and 2 use batch size 24 to ensure a like-for-like comparison, as it is the largest batch size that fits for vanilla FiD. All experiments use 40 passages of 256 tokens and output size of 32 tokens. Predictions are generated with greedy decoding as we found beam search did not meaningfully improve performance for considered tasks. Analysis in Section 5.4 investigates how trade-offs change with input and output length, low batch size and different sampling methods.

### 5.2 Main results

Figure 3 shows performance as a function of inference time for FiD and FiDO. FiDO strongly outperforms FiD at any inference budget and achieves the same performance with order of magnitude faster speed. The following section investigates how each component of FiDO contributes to its performance. Table 5 compares FiDO to published results.

### 5.3 Components

| Model | TPS | NQ | TQA | WebQ |
|---|---|---|---|---|
| FiD | 101.8 | 46.5 | 65.8 | 41.83 |
| FiD-Light | 28.3 | 36.3 | 54.5 | 30.8 |
| FiD-LSA | 29.5 | 45.8 | 65.3 | 41.0 |

Table 4: Time per sample (ms) and QA exact match for FiD, FiD-Light, and FiD Base-sized models with layer-sparse cross-attention.

**Layer-sparse cross-attention** First, Table 3 shows that layer-sparse cross-attention significantly reduces inference cost with modest performance degradation. Separately, Table 4 compares the inference speed and performance impact of layer-sparse cross-attention with the token-sparse cross-attention from FiD-Light. Reducing cross-attention layers and inducing encoder output sparsity by the same factor lead to similar speedups, but layer-sparse cross-attention achieves the inference speedup with much lower performance penalty.

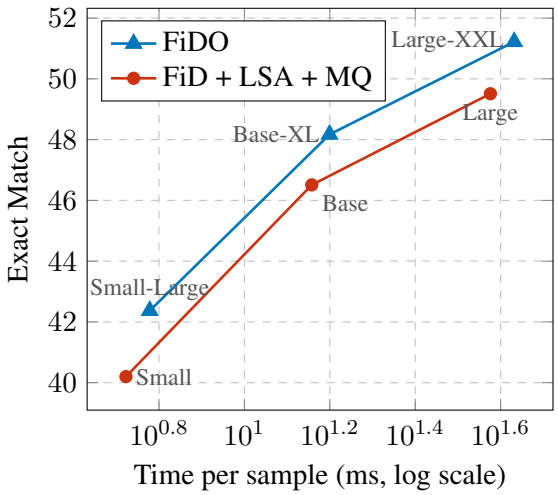


Figure 5: Performance on Natural Questions dev set as a function of inference time for FiDO Small, Base and Large models with and without asymmetric decoder.

Note that we find a much larger performance degradation from compressing the encoder output in our setting compared to the experiments in Hofstätter et al. (2022). Some exploratory experiments suggest that multi-task training fine-tuning on large amounts of data as done in FiD-Light may ameliorate the performance penalty from compressing encoder output; however even with such training Hofstätter et al. (2022) still report significant peformance degradation, in contrast to LSA.

Layer-sparsity over a factor of 6 incurs greater performance penalties. However, as shown in Table 4, with LSA-6 cross-attention already makes up a small proportion of total decoder inference cost.

**Multi-query attention** Table 3 shows that multi-query attention achieves a large cost reduction on top of layer-sparse cross-attention with minimal performance degradation, consistent with our analysis and findings from Shazeer (2019).

**Decoder scale** We can see in Table 3 that increasing the size of the decoder leads to a significant improvement in performance at the cost of a modest increase in inference time. Figure 5 provides a visual comparison of the performance-inference profile for FiDO with and without asymmetric decoders and shows that asymmetric large decoders achieve a better trade-off.

### 5.4 Other analysis

| Model | NQ | TQA | WQ |
|---|---|---|---|
| REALM (Guu et al., 2020) | 40.4 | - | 40.7 |
| RAG (Lewis et al., 2020) | 44.5 | 56.8 | 45.2 |
| RETRO (Borgeaud et al., 2022) | 45.5 | - | - |
| T5-XXL (Roberts et al., 2020) | 35.2 | 51.9 | 42.8 |
| ATLAS (Izacard et al., 2022) | 60.4 | 79.8 | - |
| FiD-L (Izacard and Grave, 2021) | 51.4 | 67.6 | - |
| FiD-L (ours) | 51.5 | 68.2 | 44.3 |
| FiDO (L-XXL) | 53.2 | 70.7 | 49.7 |

Table 5: Comparison of FiDO with published results on Natural Questions, TriviaQA and WebQuestions test sets. We focus on comparing with FiD as other works enhance performance with improved retrieval (such as ATLAS), which is orthogonal to our contributions.

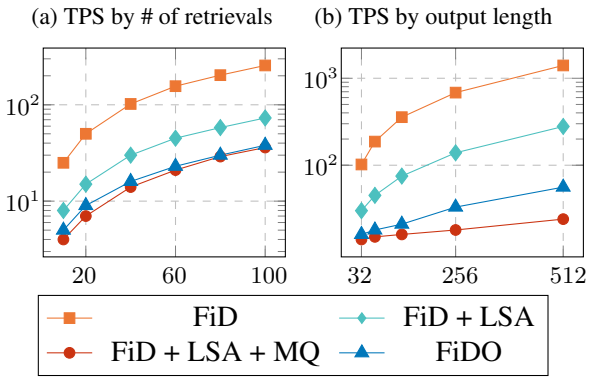


Figure 6: Time per sample (TPS) as a function of retrieved passages (left) or the number of generated tokens (right) for Base FiD variants and FiDO-Base-XL.

**Varying input and target length** Our main results use a middle-of-the-road setting for FiD applications with a medium number of retrievals and a relatively short output, reflecting common knowledge-intensive tasks. However, it is interesting to ask how FiDO components affect speed for other settings. Figure 6 shows time per sample as a function of retrieved passages and length of the target output for each step from FiD to FiDO.

We first note that layer-sparse cross-attention and multi-query attention are critical across all settings. For standard output length, the asymmetric decoder is cheap for any reasonable number of retrieved passages, becoming negligible as a fraction of total inference time as the number of retrievals increases. As output length increases, the cost of the disproportionately large decoder rises, although it only becomes a substantial proportion of inference time for output length of 256-512 and above. For tasks with long outputs, such as summarization, one may want to reduce the level of decoder asymmetry (e.g. Base-Large rather than Base-XL).

**Low batch size setting** For our primary investigation we focus on medium batch sizes (24+). There are two reasons one might care about smaller batch sizes: either because larger batches do not fit in memory or because they lead to excessive latency. The first constraint is not binding for FiDO: due to FiDO's memory efficiency we are able to fit larger batches even for the XL-XXL model, and if necessary model size can be further extended with quantization (Zeng et al., 2022) and parallelism (Pope et al., 2022).

For real-time serving latency can be a constraint, but in those settings it is common practice to use much smaller models which are distilled from larger teacher models (Gou et al., 2021). The student models can utilize a higher batch size, while the teacher models do not have latency constraints, so FiDO also applies to this use case.

For rare cases where a lower batch size is required layer-sparse and multi-query attention are still important, but cannot fully eliminate the decoder as a bottleneck for inference (Table 6). The $\frac{1}{b}$ term in Equation 5 dominates, reflecting the fact that the model has to repeatedly load model parameters without spreading the cost over many samples.

Instead of scaling the decoder, it would be more cost-effective to apply more expensive sampling methods, because sampling methods increase the effective batch size. For example, beam search with large beams is nearly free at lower batch sizes.

| Model | Total TPS | Decoder TPS |
|---|---|---|
| Vanilla FiD | 135 | 123 |
| + LSA | 51 | 39 |
| + MQ | 35 | 23 |
| + Beam 16 | 35 | 23 |
| + XL Decoder | 117 | 105 |

Table 6: Inference time per sample (ms) with batch size 1 for Base FiD with varying FiDO components.

**Sampling** We do not apply beam search for our main experiments as decoder inference time is proportional to beam width for medium batch sizes and beam search does not improve performance on the considered set of tasks. Instead, we find that scaling decoder size provides a more cost-efficient way to add decoder capacity. Table 7 compares the performance vs time trade-offs from beam search and scaling the decoder for Natural Questions, and shows that scaling the decoder is significantly more

effective. Beam search may be more important for other tasks, such as tasks with longer outputs.

| Model | Decoder TPS | NaturalQ |
|---|---|---|
| FiD with LSA, MQ | 0.6 | 46.3 |
| + Beam 4 | 2.4 | 46.2 |
| FiDO | 2.0 | 48.2 |

Table 7: Decoder inference time (ms) and QA exact match for FiD Base models, comparing the trade-offs of beam search versus scaling decoder size.

## 6 Conclusion

We perform analysis of the performance-inference speed tradeoff for FiD, showing that the encoder uses more FLOPs but most time is spent in the decoder due to memory bandwidth constraints. We propose FiDO, an extension of FiD which removes most cross-attention layers and employs multi-query attention to vastly reduce the cost of the decoder. The resulting model spends most time in the encoder, consistent with compute analysis, which FiDO takes advantage of by strongly increasing the size of the decoder. We show that FiDO achieves much stronger performance for the same inference budget relative to existing FiD models.

## Acknowlegements

We thank Livio Baldini Soares, Kenton Lee, Pat Verga, Iftekhar Naim and others at Google Research for insightful advice and discussion. Michiel de Jong is partially supported by NSF Awards IIS-1513966/ 1632803/1833137, CCF-1139148, DARPA Awards#: FA8750-18-2-0117, FA8750-19-1-0504, DARPA-D3M - Award UCB-00009528, Google Research Awards, gifts from Facebook and Netflix, and ARO# W911NF-12-1-0241 and W911NF-15-1-0484.

## Limitations

One of the advantages of the Fusion-in-Decoder approach is that it uses the off-the-shelf T5 architecture with publicly available checkpoints. The proposed FiDO modifications strongly improve performance and inference speed for retrieval-augmented question-answering, but require pre-training from scratch. It is in general preferable to have a small number of checkpoints that can be fine-tuned for any application. For example, it may not be feasible to train different giant language models for use in the retrieval-augmented setting. Instead, the architectures for such large models may need to be a compromise for different use cases.

## Ethics

In general the ethics concerns for this paper are similar to those for the large body of work studying retrieval-augmented language models. One distinction worth pointing out is that this work proposes a model with faster inference, which makes retrieval-augmented models more feasible to apply in practical settings and serve to users and inherently carries higher risk.

## References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. 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*, pages 1533–1544. ACL.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George 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 W. Rae, Erich Elsen, and Laurent Sifre. 2022. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. JAX: composable transformations of Python+NumPy programs.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *CoRR*, abs/2205.14135.

Michiel de Jong, Yury Zemlyanskiy, Nicholas FitzGerald, Fei Sha, and William W. Cohen. 2022. 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.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *CoRR*, abs/2208.07339.

Jesse Dodge, Maarten Sap, Ana Marasovic, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 1286–1305. Association for Computational Linguistics.

Google. 2020. Profile your model with cloud tpu tools. https://cloud.google.com/tpu/docs/cloud-tpu-tools. Accessed: 2022-11-11.

Google. 2022. System architecture tpu vm. https://cloud.google.com/tpu/docs/system-architecture-tpu-vm. Accessed: 2022-11-19.

Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *Int. J. Comput. Vis.*, 129(6):1789–1819.

Mandy Guo, Joshua Ainslie, David C. Uthus, Santiago Ontañón, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2022. Longt5: Efficient text-to-text transformer for long sequences. In *Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 724–736. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938. PMLR.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. 2020. Flax: A neural network library and ecosystem for JAX.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Sebastian Hofstätter, Jiecao Chen, Karthik Raman, and Hamed Zamani. 2022. Fid-light: Efficient and effective retrieval-augmented text generation. *CoRR*, abs/2209.14290.

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, EACL 2021, Online, April 19 - 23, 2021*, pages 874–880. 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. 2022. Few-shot learning with retrieval augmented language models. *CoRR*, abs/2208.03299.

Mandar Joshi, Eunsol Choi, Daniel S. 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, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1601–1611. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. 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 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. 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. *Trans. Assoc. Comput. Linguistics*, 7:452–466.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6086–6096. Association for Computational Linguistics.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Zonglin Li, Ruiqi Guo, and Sanjiv Kumar. 2022. Decoupled context processing for context augmented language modeling. *CoRR*, abs/2210.05758.

Ankur Mohan. 2018. Understanding roofline charts.

NVIDIA. 2022. Nvidia a100 tensor core gpu. https://www.nvidia.com/en-us/data-center/a100/. Accessed: 2022-12-06.

Georg Ofenbeck, Ruedi Steinmann, Victoria Caparrós Cabezas, Daniele G. Spampinato, and Markus Püschel. 2014. Applying the roofline model. In *2014 IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2014, Monterey, CA, USA, March 23-25, 2014*, pages 76–85. IEEE Computer Society.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2022. Efficiently scaling transformer inference. *CoRR*, abs/2211.05102.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Adam Roberts, Hyung Won Chung, Anselm Levskaya, Gaurav Mishra, James Bradbury, Daniel Andor, Sharan Narang, Brian Lester, Colin Gaffney, Afroz Mohiuddin, Curtis Hawthorne, Aitor Lewkowycz, Alex Salcianu, Marc van Zee, Jacob Austin, Sebastian Goodman, Livio Baldini Soares, Haitang Hu, Sasha Tsvyashchenko, Aakanksha Chowdhery, Jasmijn Bastings, Jannis Bulian, Xavier Garcia, Jianmo Ni, Andrew Chen, Kathleen Kenealy, Jonathan H. Clark, Stephan Lee, Dan Garrette, James Lee-Thorp, Colin Raffel, Noam Shazeer, Marvin Ritter, Maarten Bosma, Alexandre Passos, Jeremy Maitin-Shepard, Noah Fiedel, Mark Omernick, Brennan Saeta, Ryan Sepassi, Alexander Spiridonov, Joshua Newlan, and Andrea Gesmundo. 2022. Scaling up models and data with `t5x` and `seqio`. *arXiv preprint arXiv:2203.17189*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 5418–5426. Association for Computational Linguistics.

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *CoRR*, abs/1911.02150.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *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*, pages 4603–4611. PMLR.

Neeraj Varshney, Man Luo, and Chitta Baral. 2022. Can open-domain QA reader utilize external knowledge efficiently like humans? *CoRR*, abs/2211.12707.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Samuel Williams, Andrew Waterman, and David A. Patterson. 2009. Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76.

Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy. 2022. Memorizing transformers. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Donghan Yu, Chenguang Zhu, Yuwei Fang, Wenhao Yu, Shuohang Wang, Yichong Xu, Xiang Ren, Yiming Yang, and Michael Zeng. 2022a. Kg-fid: Infusing knowledge graph in fusion-in-decoder for open-domain question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 4961–4974. Association for Computational Linguistics.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2022b. Generate rather than retrieve: Large language models are strong context generators. *CoRR*, abs/2209.10063.

Yury Zemlyanskiy, Joshua Ainslie, Michiel de Jong, Philip Pham, Ilya Eckstein, and Fei Sha. 2021. Readtwice: Reading very large documents with memories. In *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*, pages 5189–5195. Association for Computational Linguistics.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Peng Zhang, Yuxiao Dong, and Jie Tang. 2022. GLM-130B: an open bilingual pre-trained model. *CoRR*, abs/2210.02414.

## A Training

All experiments are built on the T5.1.1 architecture with the training recipe from T5 (Raffel et al., 2020). The first exception is the optimizer; we find that the second moment factoring and mixing schedule from Adafactor (Shazeer and Stern, 2018) can lead to instability, especially with unbalanced encoder and decoder sizes. Instead, we disable factoring and second moment mixing, leading to an

optimizer that is a hybrid between Adafactor and Adam (Kingma and Ba, 2015).

The second difference to the training recipe arises from the observation that FiDO XL-XXL is unstable for the standard training regimen. We solve the instability by restarting from a recent healthy checkpoint with a 10x decreased learning rate, which happened once.

During fine-tuning, we load not only model weights but also second moment estimates, which we find leads to better fine-tuning in general and particularly for asymmetric models. We finetune with learning rate 0.001 and batch size 64 for all datasets. For evaluation on test sets we select the checkpoint with the best validation performance.

## ACL 2023 Responsible NLP Checklist

### A For every submission:

☑ A1. Did you describe the limitations of your work?
*After conclusion*

☑ A2. Did you discuss any potential risks of your work?
*After limitations*

☑ A3. Do the abstract and introduction summarize the paper's main claims?
*1*

☒ A4. Have you used AI writing assistants when working on this paper?
*Left blank.*

### B ☑ Did you use or create scientific artifacts?

*5*

☑ B1. Did you cite the creators of artifacts you used?
*5*

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
*Not applicable. Left blank.*

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
*Not applicable. Left blank.*

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
*Not applicable. Left blank.*

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
*Not applicable. Left blank.*

☒ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
*Used and referred to standard use in past literature*

### C ☑ Did you run computational experiments?

*5*

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
*Parameter count not relevant, long discussions of computational budget and infrastructure*

*The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.*

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
*Left blank.*

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
*5*

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
*Not applicable. Left blank.*

**D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?**

*Left blank.*

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
*No response.*

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
*No response.*

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
*No response.*

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
*No response.*

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
*No response.*