

Latent Memory Interfaces for Scalable Multi-Hop QA

Hazim Kuniyil

Department of Computer Science

University of Texas at Austin

Austin, TX 78712, USA

hazim.a.kuniyil@gmail.com, hak835@my.utexas.edu

Abstract

Multi-hop Question Answering (QA) tasks typically require Large Language Models (LLMs) to reason over multiple pieces of evidence scattered about a large corpus. However, standard transformers have a finite context window and often struggle to use the entire context effectively. In this work, I propose a latent bottleneck architecture: instead of forcing the reader to attend to an ever-growing context window, I funnel the context through a small, fixed-size latent bottleneck before sending it to the decoder. Concretely, I introduce a simple latent memory interface that connects a Longformer context encoder to a Flan-T5-base reader. A Perceiver-style cross-attention module converts the token embeddings into a compact set of learned latent vectors, which are then passed through a scalar gate inspired by the Flamingo VLM and concatenated to the encoder states of the reader. I evaluate this approach on the HotpotQA distractor setting, adopting a two-phase training regime to stabilize the learning of the latent embedding layer. I compare this latent bottleneck to a strong long-context baseline, demonstrating that the latent bottleneck model achieves performance comparable to it. Furthermore, ablation studies that zero out or randomize the latents confirm that the reader relies on the latent bottleneck, suggesting that latent memory interfaces are a promising mediator between long-context encoders and standard seq2seq readers.

1 Introduction

Approximately half of ChatGPT usage involves Question Answering (QA) tasks (Lysø, 2025). These tasks typically require the Large Language Model (LLM) to reason over multiple pieces of evidence scattered about a large corpus of evidence, leading to the development of multi-hop QA datasets like HotpotQA (Yang et al., 2018) and Wikihop (Welbl et al., 2018), both of which test LLMs’ ability to reason over several documents,

ignore distractors, and support their answers with evidence from the provided sources.

This is a difficult task for LLMs (based on the transformer architecture), which have a finite context window, limiting the amount of evidence they can examine (Vaswani et al., 2017). While hardware advances have enabled massive context windows (Systems, 2021), empirical evidence shows that models do not use the entire context window effectively; they struggle to find information buried in the center of the input (Liu et al., 2024). Additionally, as Zewe (2025) notes, there are environmental and sustainability concerns associated with the continued scaling of generative AI models. Rather than simply scaling up the context window, a novel approach to context management is needed.

In this work, I propose a latent bottleneck architecture: instead of forcing the reader to attend to an ever-growing context window, I funnel the context through a small, fixed-size latent bottleneck before sending it to the decoder. Concretely, I use a Longformer encoder to encode the concatenated question and context/distractor paragraphs without training a separate retriever to obtain a query-aware token representation (Beltagy et al., 2020). Next, a Perceiver-style cross attention module converts the token embeddings into a compact set of learned latent vectors. Finally, these latents are passed through a scalar gate inspired by the Flamingo VLM gating mechanism (Alayrac et al., 2022) and concatenated to the encoder states of a Flan-T5-base reader. Therefore, the decoder is conditioned on both the question tokens and the compressed latent summary generated by the new memory interface.

There are four contributions that this project introduces. First, I introduce a simple latent memory interface that connects a Longformer context encoder to a Flan-T5-base reader in the HotpotQA distractor setting. Second, I compare this latent bottleneck to a strong long-context baseline, demon-

strating that the latent bottleneck model achieves performance comparable to it. Third, I adopt a two-phase training regime—initially training with the T5 model weights frozen—to learn a latent embedding layer that can capture task-relevant information. Finally, I present ablation studies that zero out, randomize, or bypass the latents at evaluation time to identify failure modes and test whether the reader could be trained to rely on the latent bottleneck.

2 Background and Related Work

Existing approaches to multi-hop QA generally fall into two distinct streams: explicit retrieval–reader pipelines and long-context transformer architectures.

2.1 Explicit Retrieval and Reasoning

Classical pipelines like DrQA fall into the former stream; they combine sparse retrieval with a neural reader using TF-IDF and bigram hashing to retrieve candidate passages alongside an RNN-based extractor (Chen et al., 2017). Dense Passage Retrieval (DPR) replaces sparse retrieval with a learned dense encoder to produce a substantial improvement over sparse methods (Karpukhin et al., 2020).

Beyond simple retrieval, several works have attempted to impose explicit structure on the reasoning process. Min et al. (2019) propose decomposing complex questions into simpler sub-queries to retrieve evidence iteratively. Alternatively, graph-based methods construct explicit reasoning chains between entities; Song et al. (2018) and Fang et al. (2020) utilize Graph Neural Networks to model the relationships between entities across different documents.

In contrast, Retrieval-Augmented frameworks such as RAG and REALM utilize generative readers conditioned on the outputs of neural retrievers (Guu et al., 2020; Lewis et al., 2020). Finally, Fusion-in-Decoder (FiD) and large-scale systems like Atlas and RETRO push this idea to its limits by encoding many passages independently before amalgamating them in powerful decoder systems (Izacard and Grave, 2021; Izacard et al., 2023; Borgeaud et al., 2022).

2.2 Long-Context and Latent Architectures

In the long-context transformer stream, we find models such as Longformer and BigBird that introduce sparse attention patterns to effectively parse

thousands of tokens (Beltagy et al., 2020; Zaheer et al., 2020), as well as models such as Transformer-XL and the Compressive Transformer that elongate their effective context windows through recurrence and compressed memory (Dai et al., 2019; Rae et al., 2019).

Our approach differs by compressing the context into a fixed-size latent bottleneck. This draws on the Perceiver architecture (Jaegle et al., 2021b) and its successor Perceiver IO (Jaegle et al., 2021a), which demonstrated that cross-attention can map high-dimensional byte arrays or token sequences into a low-dimensional latent array, decoupling the depth of the network from the size of the input.

3 Methodology

Concretely, the multi-hop QA task that I address in this project can be described as follows: The system needs to predict an answer string given a question string and a set of candidate context paragraphs. I use HotpotQA (Yang et al., 2018; hot) in the *distractor* configuration. In this configuration, each sample of the dataset has two supporting documents and eight distractor documents (all from Wikipedia). The dataset contains separate columns for the question, answer, a list of context documents and distractors, and a list with the titles of the two supporting documents. I didn’t train a separate sparse retriever model for this project and just treat the paragraphs supplied by HotpotQA as the context; the models trained in this project attempt to identify relevant supporting sentences in the context set provided by HotpotQA.

The baseline I implemented for this project is a classic encoder-decoder baseline based on google/Flan-T5-base. This is a 12 layer transformer with a standard encoder-decoder architecture (Vaswani et al., 2017). For each sample in the dataset, I construct a single input array by concatenating the question with a flattened array containing the context documents separated by a delimiter token and truncated to a maximum of 512 tokens. The baseline model is then trained to generate the golden answer text Using a seq2seq negative log likelihood objective. I employ teacher forcing during the training and beam search decoding at evaluation time. This baseline has no latent bottleneck or any auxiliary components; the decoder simply attends to all encoder tokens that fit in the context window in order to provide a strong long-context Flan-T5-base system to serve as a baseline with

which I will compare the latent bottleneck architecture.

The proposed system is composed of a Longformer context encoder and a Perceiver-style latent bottleneck attached to the Flan-T5-base reader (see Figure 1). First, the system generates a "query-aware" context array by prepending the question tokens to the HotpotQA paragraphs and encoding the entire sequence using `allenai/longformer-base-4096` (Beltagy et al., 2020). This converts the long input token sequence (questions, distractors, context) into hidden states. Next, a latent bottleneck module - inspired by the Perceiver IO and Set Transformer architectures (Jaegle et al., 2021a; Lee et al., 2019) - is applied. Here, a fixed set of 64 learnable latent vectors cross-attend the Longformer token embeddings, and a small latent self-attention submodule curates the data into a small output array. Ideally, the resultant latent array encodes the question and relevant context in a compact, fixed-size latent memory. These latent memory vectors are then passed through a learnable scalar gate with a tanh nonlinearity and multiplicative scaling for all latent dimensions in the style of Flamingo gated cross-attention (Alayrac et al., 2022). Finally, the gated latents - which are output to the hidden size of Flan-T5-base - are concatenated along the sequence dimension with standard Flan-T5-base encoder outputs computed over the original question tokens alone. This process conditions the Flan-T5-base model on both the original question embeddings and the context documents compressed by the Longformer-latent bottleneck module.

The primary training objective for both the baseline and latent model is sequence-to-sequence (seq2seq) negative log-likelihood loss on the answer text. Given a question and context, the model is expected to generate the exact answer text. The latent model is also trained on an auxiliary objective to encourage it to focus the latents on supporting evidence. This is done by constructing a binary mask over the Longformer input tokens, indicating which tokens belong to supporting sentences. On the last layer of the latent cross attention, the auxiliary loss is calculated as the negative log of the total attention mass over the supporting tokens. A small constant is added to this mass for numerical stability. Ideally, this auxiliary loss penalizes any latent attention distributions that don't allocate high probabilities to the true supporting sentences; this nudges the bottleneck toward correctly repre-

senting the reasoning paths grounded in the labeled evidence (Chen et al., 2019; Asai et al., 2020). The overall training objective is a weighted sum of the seq2seq loss and auxiliary loss (although the auxiliary loss is treated as a small regularizing factor).

The latent model weights were trained in two phases in order to stabilize training and avoid overwhelming the pretrained model with completely random latent features. In Phase 1, the latent model is initialized from the pretrained Longformer and Flan-T5-base checkpoints, but all the T5 parameters are frozen. I then proceed to train only the Longformer context encoder, latent module, scalar gate, and auxiliary loss parameters. The goal of Phase 1 is to ensure that the latent bottleneck produces a usable summary representation of the context while treating the decoder as a fixed reader. In Phase 2, the training is resumed from the checkpoint saved at the end of Phase 1, but all the model weights are unfrozen. In this stage, the model is fine-tuned from end-to-end, allowing the various modules to learn how to work together. The optimizer used throughout this training is AdamW, and parameter groups and learning rates are set separately for the T5 backbone, context encoder, latent parameters, and gating modules. Additionally, a linear warm-up, and gradient clipping are implemented. The batch size, epoch count, and maximum sequence length parameters were set based on the constraints of the single GPU and are kept uniform across baseline and latent runs to ensure a even comparison.

All models were trained on a single Nvidia RTX 5090 GPU. Frameworks used in this project include PyTorch and the Hugging Face Transformers and Datasets modules (Paszke et al., 2019; Wolf et al., 2020; Lhoest et al., 2021). To keep comparisons fair, both the baseline and latent models are trained using the same custom HotpotQA Dataset wrapper and tokenization pipeline. During training, the model is periodically decoded on the validation split of the dataset to compute EM and F1 using a custom evaluation module with SQuAD-style (lowercase, strip punctuation/articles) normalization (Rajpurkar et al., 2016). I used a small monitoring script that aggregates training metrics across epochs to help guide early stopping decisions. Gradient checkpointing was used on the latent model to help reduce memory usage and clock time. I also hoped to use mixed precision to improve training efficiency, but this feature was disabled due to numerical instability issues.

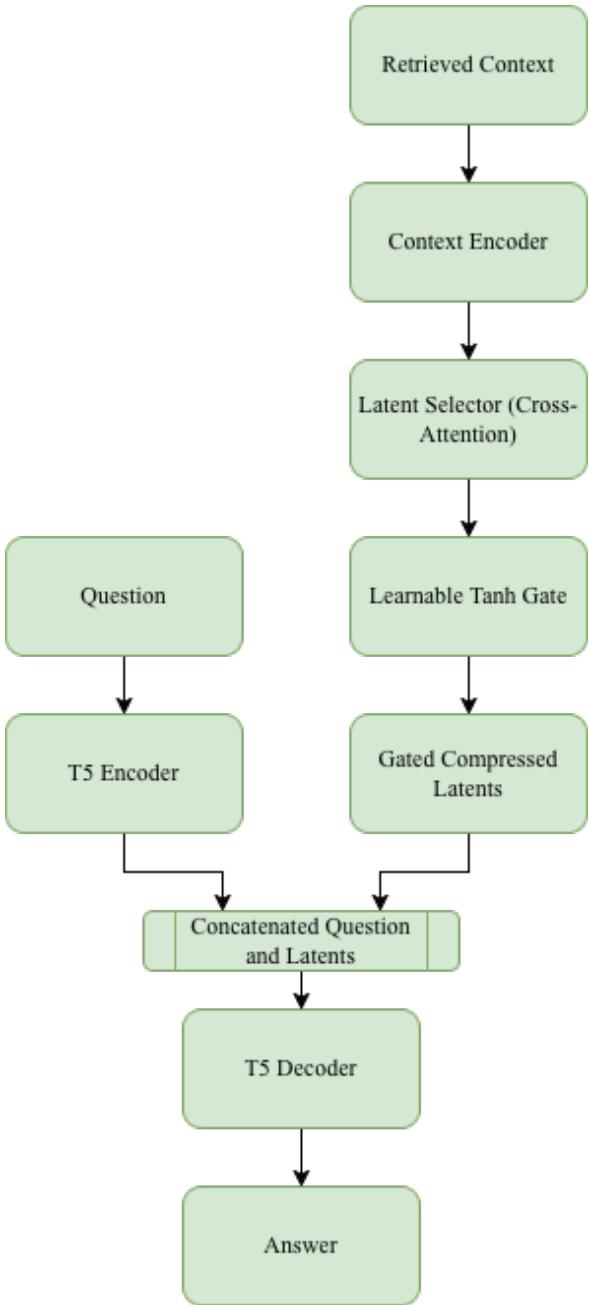


Figure 1: Simple diagram showing the latent bottleneck architecture.

4 Experiments and Results

4.1 Experimental Setup

I used the HotpotQA train and validation splits in the distractor configuration throughout this project (Yang et al., 2018). The dataset is accessed through the Hugging Face Datasets interface. Each example consists of a question string, answer string, context document sets, and annotations for the context documents (indicating what documents contain supporting evidence). The context document sets contain 8 distractors and 2 supporting documents per

question.

For the Baseline (Flan-T5), I use a custom HotpotDataset wrapper class with the collate_baseline function to flatten the paragraphs into a single string with a delimiter. The flattened context is concatenated with the question string and converted into a single input sequence using the Flan-T5 encoder. This input sequence was clipped to 512 tokens if necessary.

For the latent bottleneck model, I use the same HotpotDataset wrapper class with a different collator function: collate_latent. In this function the question is prepended to the list of HotpotQA paragraphs and tokenized with the Longformer tokenizer. The resultant sequence is clipped to 512 tokens if necessary. This "query-aware" context sequence is then sent to the Longformer encoder along with a binary mask encoding the supporting sentence labels. Simultaneously, the base question and answer sequence is also tokenized (in order to serve as the input for the T5 model) the same way as the baseline.

The evaluation metrics for all experiments are the Exact Match (EM) and F1 scores on the HotpotQA validation split. The supporting fact annotation metrics (auxiliary loss) aren't included here; those metrics are used for training purposes only. In short, the reported metrics always measure answer string correctness.

4.2 Models Compared

I compare three systems in this work.

First, in the baseline system, the Flan-T5-base encoder-decoder model directly encodes the concatenated question and context string then decodes the answer (Vaswani et al., 2017).

The second and third systems are include the latent bottleneck module after the first and second phases of training respectively. The first phase of training is run with the T5 model weights frozen; only the Longformer context encoder, Perceiver-style latent module, gate, and auxiliary-loss parameters are trained in this stage. The second phase resumes training from the checkpoint created at the end of the first phase, but trains the entire system jointly (ie the T5 model weights are unfrozen). The full architecture described at the end of Section 3 is attained here, and this is the final model used in the ablation studies.

4.3 Training Dynamics

I originally planned to run this project in a single stage, but early training metrics show that this approach results in the loss plateauing at ≈ 2.7 and the EM at $\approx 11\%$. As shown in Figure 2, the two stage training configuration allows the validation loss to steadily breach this limit in the first phase, nudging the EM score into the lower 12% range. Unfreezing the T5 model weights in the second phase allows answer loss to reach ≈ 2.38 and the peak EM to reach $\approx 12.7\%$.

4.4 Main Results

The final EM and F1 scores on the HotpotQA validation set are reported on Table 1. The *Baseline T5* row corresponds to the bypass mode of the phase 2 checkpoint. Here, the latent pathway is disabled and only the T5 encoder state is fed to the decoder. This baseline scores 11.69% EM and 18.64 F1.

After Phase 1, the latent model scores 11.88% EM and 18.67 F1, representing a small improvement over the baseline. The final model (after Phase 2) scores 11.71% EM and 18.46 F1. Although these differences are small, the model showed continuous improvement. Given additional compute resources and training time, I am hopeful that the model could continue to improve. By the end of the experimentation, the latent bottleneck system was able to approximate the T5-only baseline despite conditioning the T5 decoder on 64 latent vectors instead of all the Longformer tokens.

4.5 Ablation Studies via Latent-Removal Modes

To test whether the T5 reader actually uses the latent bottleneck, I evaluated the Phase 2 checkpoint under the latent-removal modes implemented in the evaluation script (Figure 3):

- **Full latent model:** 11.71% EM, 18.46 F1.
- **Zeros ablation** (latents replaced with zeros): 10.99% EM, 16.82 F1.
- **Random ablation** (latents replaced with random noise): 10.97% EM, 16.85 F1.
- **Bypass baseline** (latent path removed; T5 encoder only): 11.69% EM, 18.64 F1.

Replacing the latents with zeros or random noise produced a consistent drop of approximately 0.7–0.8 EM and 1.6–1.7 F1 points relative to the full

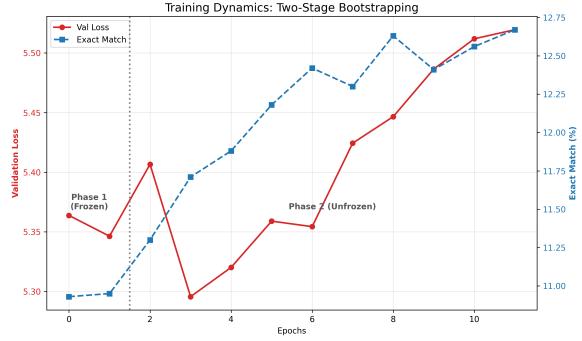


Figure 2: Training progression using Two-Stage Bootstrapping. The T5 reader is frozen to initialize the latent selector in phase 1 (Epochs 0-1). The reader is unfrozen in Phase 2 (Epochs 2-11). Note the steady rise in Exact Match (blue) even as Loss (red) stabilizes, reaching a peak EM of 12.67%.

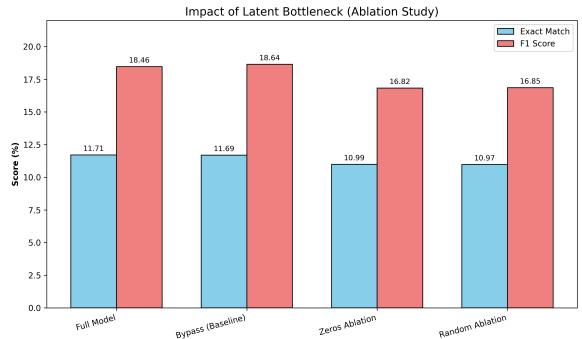


Figure 3: Ablation study on the validation set. Replacing learned latents with Zeros or Random noise causes a significant performance drop ($\sim 0.7\%$), suggesting that the T5 reader attends to the latent bottleneck. The Full Model performs comparably to the Bypass baseline, suggesting successful mechanism integration but limited information capacity in the bottleneck.

model. This indicates that the reader is sensitive to the *content* of the latent vectors rather than merely their presence. At the same time, the performance of the bypass baseline, which is effectively equal to that of the full model, shows that the current latent bottleneck does not yet yield clear gains over a well-tuned Flan-T5-only reader; instead, it provides a compressed interface that can approximately recover the baseline.

5 Discussion and Future Work

The results of these experiments strongly suggest that a simple latent bottleneck interface is viable, despite the dramatic reduction in total context content. The latent model matches or slightly improves upon the EM and F1 scores in the validation split, and the ablation testing suggests that the T5 model

Model	EM	F1	Notes
Baseline T5	11.69	18.64	Flan-T5-base, max seq len = 512 (bypass evaluation without latents)
Latent (Phase 1)	11.88	18.67	Frozen T5 encoder-decoder; train Longformer + latents + gate only
Latent (Phase 2)	11.71	18.46	Joint fine-tuning of Longformer, latents, gate, and T5 (final checkpoint)

Table 1: Final HotpotQA dev performance.

utilizes the latent pathway. However, these results come with a major caveat: the absolute gains produced by the new architecture are rather modest and potentially insignificant.

I am hopeful that, given the success of the two stage training regime, splitting the training into several stages and training each of the components of the system individually will produce even greater results. Additionally, I would be interested in seeing whether the system would be more successful with a smaller compression ratio; the system I used in this experiment compresses the 512 token output of the Longformer model into 64 latent vectors (an 8x reduction in sequence length).

There are several natural continuations to this experiment. One could vary the capacity and structure of the latent bottleneck (number of latents, depth of the latent stack, and where the latents are injected into T5) and to experiment with richer (ie non-scalar) gating mechanisms. Also, one could clarify whether the bottleneck is more effective with other question types or evidence distributions using benchmarks other than HotpotQA. Finally, integrating a discrete retriever module upstream of the latent module and pretraining the latent interface on larger corpora may produce latent bottlenecks genuinely outperform strong long-context baselines rather than merely matching them.

6 Conclusion

In this project, I introduced a simple latent bottleneck architecture that converts context embeddings from a Longformer context encoder to a small set of learned latent vectors that the Flan-T5-base reader can use for multi-hop QA tasks such as those in the HotpotQA dataset. The model compresses hundreds of context tokens into a small set of learned latent vectors, which are gated (Flamingo-style) and concatenated to the T5 encoder state, yielding a system that can approximate the baseline performance of the long-context T5 model.

Although the latent bottleneck system discussed here doesn’t deliver a clear improvement over the baseline, ablation testing indicates that the reader was able to learn to use the latent representations. These results suggest that latent bottlenecks comprise a promising mediator between long-context encoders and standard seq2seq readers. I hope that this work will serve as a first step toward more principled context management techniques and a template for future research that will push latent memory interfaces beyond parity with the baseline.

Limitations

The experiments conducted for this project are limited to the HotpotQA dataset. The scope was limited to a fixed pair of models (Longformer and Flan-T5-base). The training runs were restricted by limited compute (a single RTX 5090 and limited time). Finally, the only metrics available are EM and F1; the results should be interpreted cautiously in lieu of more rigorous metrics and significance testing.

Acknowledgments

I thank Kailash Subramanian for reviewing my project proposal. This work relies on the HotpotQA dataset, as well as open-source code from PyTorch and the Hugging Face transformers and datasets libraries and the model weights of the Longformer and Flan-T5 models.

References

- Hotpotqa dataset homepage. <https://hotpotqa.github.io/>. Accessed 2025-11-27.
- Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, and 1 others. 2022. [Flamingo: A visual language model for few-shot learning](#). *arXiv preprint arXiv:2204.14198*.
- Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. [Learning](#)

- to retrieve reasoning paths over wikipedia graph for question answering. In *Proceedings of ICLR*.
- Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. **Longformer: The long-document transformer.** *arXiv preprint arXiv:2004.05150*.
- Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, and 1 others. 2022. **RETRO: Improving language models by retrieving from trillions of tokens.** *arXiv preprint arXiv:2112.04426*.
- Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. **Reading wikipedia to answer open-domain questions.** In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.
- Jifan Chen, Shih-Ting Lin, and Greg Durrett. 2019. **Multi-hop question answering via reasoning chains.** *arXiv preprint arXiv:1910.02610*.
- Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. **Transformer-XL: Attentive language models beyond a fixed-length context.** In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.
- Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuhang Wang, and Jingjing Liu. 2020. **Hierarchical graph network for multi-hop question answering.** In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.
- Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. **REALM: Retrieval-augmented language model pre-training.** *arXiv preprint arXiv:2002.08909*.
- Gautier Izacard and Edouard Grave. 2021. **Leveraging passage retrieval with generative models for open domain question answering.** *arXiv preprint arXiv:2007.01282*.
- Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. **Atlas: Few-shot learning with retrieval augmented language models.** *Journal of Machine Learning Research*, 24(160).
- Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, and 1 others. 2021a. **Perceiver IO: A general architecture for structured inputs & outputs.** *arXiv preprint arXiv:2107.14795*.
- Andrew Jaegle, Felix Gimeno, Andrew Brock, Oriol Vinyals, Andrew Zisserman, and 1 others. 2021b. **Perceiver: General perception with iterative attention.** *arXiv preprint arXiv:2103.03206*.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. **Dense passage retrieval for open-domain question answering.** In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.
- Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Koziorek, Seungjin Choi, and Yee Whye Teh. 2019. **Set transformer: A framework for attention-based permutation-invariant neural networks.** In *Proceedings of the 36th International Conference on Machine Learning*.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kütterer, 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*.
- Quentin Lhoest, Clement Delangue, Julien Plu, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patel, Julien Chaumond, Mariama Drame, Stas Bekman, and 1 others. 2021. **Datasets: A community library for natural language processing.** In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184.
- Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. **Lost in the middle: How language models use long contexts.** *Transactions of the Association for Computational Linguistics*, 12:157–173.
- Ben Lysy. 2025. **34 ChatGPT statistics for 2026.** <https://zapier.com/blog/chatgpt-statistics/>. Zapier Blog. Accessed: 2025-11-27.
- Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. **Multi-hop reading comprehension through question decomposition and rescoring.** In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. **Pytorch: An imperative style, high-performance deep learning library.** In *Advances in Neural Information Processing Systems* 32, pages 8024–8035.
- Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. 2019. **Compressive transformers for long-range sequence modelling.** *arXiv preprint arXiv:1911.05507*.

- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.
- Linfeng Song, Zhiguo Wang, Mo Yu Zhang, Yue Yu, and Dong Yu. 2018. Exploring graph-structured passage representation for multi-hop reading comprehension. In *Proceedings of the 27th International Conference on Computational Linguistics*.
- Cerebras Systems. 2021. Context is everything: Why maximum sequence length matters. <https://www.cerebras.ai/blog/context-is-everything-why-maximum-sequence-length-matters>. Blog post.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.
- Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierrick Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, and Jamie Brew. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Yang, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.
- Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*.
- Adam Zewe. 2025. Explained: Generative AI’s environmental impact. <https://news.mit.edu/2025/explained-generative-ai-environmental-impact-0117>. MIT News, accessed 2025-11-27.

A Code Repository

The full code implementation, including training/evaluation scripts and model definitions for the experiments described in this paper, is available at: <https://github.com/Hazim-Kuniyil/latent-compression>.