Training Adaptive Computation for Open-Domain Question Answering with Computational Constraints

Yuxiang Wu **Pasquale Minervini** **Pontus Stenetorp** **Sebastian Riedel**
University College London
{yuxiang.wu,p.minervini,p.stenetorp,s.riedel}@cs.ucl.ac.uk

Abstract

Adaptive Computation (AC) has been shown to be effective in improving the efficiency of Open-Domain Question Answering (ODQA) systems. However, current AC approaches require tuning of all model parameters, and training state-of-the-art ODQA models requires significant computational resources that may not be available for most researchers. We propose *Adaptive Passage Encoder*, an AC method that can be applied to an existing ODQA model and can be trained efficiently on a single GPU. It keeps the parameters of the base ODQA model fixed, but it overrides the default layer-by-layer computation of the encoder with an AC policy that is trained to optimise the computational efficiency of the model. Our experimental results show that our method improves upon a state-of-the-art model on two datasets, and is also more accurate than previous AC methods due to the stronger base ODQA model. All source code and datasets are available at https://github.com/uclnlp/APE.

1 Introduction

Open-Domain Question Answering (ODQA) requires finding relevant information for a given question and aggregating the information to produce an answer. The retriever-reader architecture, popularised by Chen et al. (2017), has shown great success in this task. The retriever acquires a set of documents from external sources (e.g., Wikipedia) and the reader extracts the answer spans from these documents (Clark and Gardner, 2018; Yang et al., 2019; Wang et al., 2019; Min et al., 2019; Asai et al., 2020). Recently, Min et al. (2020); Lewis et al. (2020b); Izacard and Grave (2020b) showed that generative reader models that exploit an encoder-decoder architecture can significantly outperform previous extractive models, thanks to their better capability in aggregating and combining evidence from multiple passages. However, these generative models are much more computationally expensive than extractive models, and often need to be trained with a large number of passages, making it hard to train these models for most researchers (Schwartz et al., 2020a).

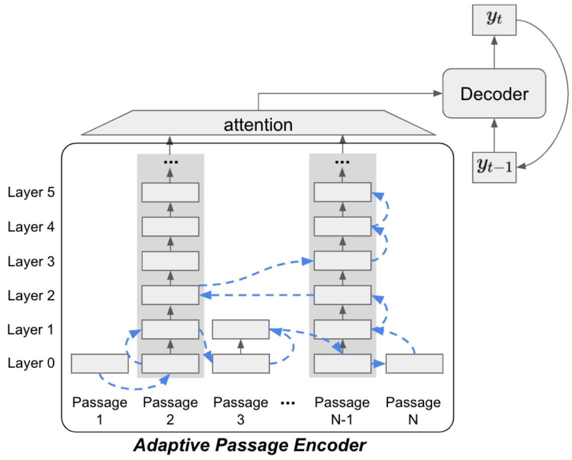

Figure 1: Overview of our approach. The adaptive passage encoder overrides the layer-by-layer computation of the encoder with an adaptive computation policy (indicated in blue dash arrows).

Wu et al. (2020) show that Adaptive Computation (AC) can significantly improve the efficiency of extractive ODQA models at inference time. However, it requires fine-tuning all model parameters with a multitask learning objective, making it computationally challenging to apply this method to current state-of-the-art models.

In this work, we explore an efficient approach to apply adaptive computation to large generative ODQA models. We introduce the *Adaptive Passage Encoder* (APE), a module that can be added to the encoder of an existing ODQA model, which has the following features: *1*) it efficiently reuses the encoder's hidden representations for calculating

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics
and the 11th International Joint Conference on Natural Language Processing (Short Papers), pages 447–453
August 1–6, 2021. ©2021 Association for Computational Linguistics

the AC priorities; *2*) it does not require tuning of the base model and hence allows efficient training under limited resource; *3*) it does not require confidence calibration. Our experimental results on NaturalQuestions and TriviaQA show that our method improves the performance of the state-of-the-art model FiD (Izacard and Grave, 2020b), while also producing more accurate results (12.4% EM) than the AC method proposed by Wu et al. (2020).

2 Related Work

Open Domain Question Answering ODQA is a task that aims to answer a factoid question given a document corpus. Most works in this domain follow a *retriever-reader* design first proposed by Chen et al. (2017). The retriever collects a set of relevant passages, then the reader comprehends and aggregates the information from multiple passages to produce the answer. Depending on the design of the reader model, these systems could be further categorised into *extractive models* and *generative models*. Extractive models (Min et al., 2019; Yang et al., 2019; Wang et al., 2019; Asai et al., 2020; Karpukhin et al., 2020) exploit an answer extraction model to predict the probabilities of answer spans, and use global normalisation (Clark and Gardner, 2018) to aggregate the answer probabilities across multiple passages.

However, thanks to recent advances in sequence-to-sequence pretrained language models (Raffel et al., 2020; Lewis et al., 2020a), generative ODQA models (Min et al., 2020; Lewis et al., 2020b; Izacard and Grave, 2020b) achieve significant improvement upon extractive models, demonstrating stronger capability in combining evidence from multiple passages. We focus on generative models in this work.

Passage Retrieval and Re-Ranking Passage retrievers in ODQA systems are initially based on sparse vector representations. Chen et al. (2017) use TF-IDF, whereas Yang et al. (2019); Karpukhin et al. (2020); Wang et al. (2019) rely on BM25 for ranking passages (Robertson, 2004). Recently, Karpukhin et al. (2020); Lewis et al. (2020b); Izacard and Grave (2020a) achieved substantial increase in retrieval performance using dense representations. Our work is based on the retrieval results from a dense retriever (Izacard and Grave, 2020b), but we show that the proposed method can still improve the quality of the support passages despite the strong retrieval performance.

Nogueira and Cho (2019); Qiao et al. (2019); Mao et al. (2021) show that adding a separate cross-encoder re-ranker can improve the performance, but that comes with a significant increase of the computation at train or inference time. Despite that our proposed adaptive passage encoder can be viewed as an encoder with an integrated re-ranker, the focus of our work is to improve the computational efficiency, namely, enhancing the performance without a substantial increase in computation.

Adaptive Computation Adaptive computation allows the model to condition the computation cost on the input. For example, Schwartz et al. (2020b); Liu et al. (2020); Xin et al. (2020) propose models that can dynamically decide to early exit at intermediate layers when the confidence at the layer exceeds a threshold. They show that adaptively early exiting can significantly reduce the computational cost for various sequence classification tasks. Closest to our work, Wu et al. (2020) introduced adaptive computation for extractive ODQA models. We extend adaptive computation to generative ODQA models, and our approach can be incorporated in existing generative ODQA models without finetuning the base model.

3 Method

In this section, we will introduce the base model and how our proposed adaptive passage encoder works with it.

3.1 Base Model

Large generative ODQA models (Lewis et al., 2020b; Izacard and Grave, 2020b) share a similar encoder-decoder architecture. They first concatenate the question with all retrieved passages. Then the encoder encodes all passages and produces their hidden representations $h_1^L, \cdots, h_N^L$, where L is the number of encoder layers and N is the number of retrieved passages. We denote the hidden representation of the i-th passage at its j-th encoder layer as h_i^j. The decoder will attend to these hidden representations and generate the answer tokens sequentially.

3.2 Adaptive Passage Encoder

As shown in Fig. 1, the adaptive passage encoder overrides the layer-by-layer computation of the encoder of the base model with an adaptive computation policy. It adds two components on top of the base encoder to define the policy: an answerability prediction model HasAnswer and a scheduler.

The HasAnswer model predicts the probability that a passage contains an answer to the question, given its hidden representation h_i^j. It first pools hidden representation h_i^j into a vector, then feeds the pooled representation to a multi-layer perceptron to produce the probability p_i^j.

The scheduler is then responsible for the selection and prioritisation of passages that are likely to contain the answer (Wu et al., 2020). As shown by the blue arrows in Fig. 1, the scheduler learns a scheduling policy to allocate encoder layer computation to passages. The scheduler will exit in early layers for those spurious passages while allocating more layers to the ones that it finds promising.

To achieve this goal, the scheduler produces a *priority* score q_n for each passage:

$$q_n = \sigma(g(p_n^{l_n}, n, l_n))p_n^{l_n} + f(p_n^{l_n}, n, l_n) \qquad (1)$$

where n is the passage rank by the retriever, l_n is the index of its current encoder layer, g and f are two multi-layer perceptrons that learn the weight and bias respectively. Starting at the initial layer for all passages, the scheduler will select a passage with the maximum priority, forward one encoder layer for it $l'_n = l_n + 1$, and updates its priorities q_n with its new hidden representation $h_n^{l'_n}$ and has-answer probability $p_n^{l'_n}$. This process will iterate for B (budget) steps, and only k passages with the most layers computed are retained in the end.

3.3 Training the Adaptive Passage Encoder

Differently from Wu et al. (2020), our method does not require tuning the underlying base model. Since the number of parameters introduced by the HasAnswer model and the scheduler is less than 4% of the base model, APE can be trained very efficiently. The HasAnswer model is first trained with cross-entropy loss, supervised by the has-answer labels of the passages. Then we fix HasAnswer and train the scheduler with REINFORCE algorithm (Williams, 1992) to maximise the expected return, which is defined to encourage selection and prioritisation of passages that contain the answer. The selection action gains a positive reward $(1 - c)$ if it selects a relevant passage, otherwise a negative reward $-c$. Since the weight g and bias f in Eq. (1) are automatically learned during the training of the scheduler, our method does not require confidence calibration of the HasAnswer model, unlike the method proposed by Wu et al. (2020).

	Train	Validation	Test
NaturalQuestions	79,168	8,757	3,610
TriviaQA	78,785	8,837	11,313

Table 1: Number of samples of the evaluated datasets.

4 Experiments

4.1 Experimental Setup

Datasets Following (Lee et al., 2019; Izacard and Grave, 2020b), we evaluate our method on NaturalQuestions (Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017) whose statistics are shown in Table 1.

Evaluation Metrics Following Wu et al. (2020), we conduct the evaluation under different computational costs at inference time. Since the number of passages k is almost linearly correlated with memory consumption and number of operations, we evaluate the performances with various number of passages $k \in \{5, 10, 20\}$. To evaluate the end performance of ODQA models, we use the standard Exact Match (EM) score, which is the proportion of questions whose predicted answer matches exactly with the ground truth. We also include the unrestricted setting to compare the best performances of different models.

Technical Details We use FiD (Izacard and Grave, 2020b) as our base model. FiD-base and FiD-large contain $L = 12$ and 24 layers respectively, and we set the budget $B = Lk$. For the pooling operation in the HasAnswer model, we found max-pooling works better than mean-pooling and the [CLS] token, so max-pooling is used in all our experiments. We use discount factor $\gamma = 0.8$ and step penalty $c = 0.1$ during the REINFORCE training of the scheduler. More hyperparameters are presented in Appendix A.1.

Computational Feasibility Tuning a FiD-base model with $k = 20$ or a FiD-large model with $k = 10$ (batch size=1) would yield out-of-memory errors on a V100 (16GB) GPU. Hence, it is infeasible to train FiD with the previous AC method (Wu et al., 2020) in our setting. However, training with our proposed approach can be done in the same setting with a batch size 4 or larger within 8-15

	NaturalQuestions				**TriviaQA**			
	Top-5	Top-10	Top-20	Unrestricted	Top-5	Top-10	Top-20	Unrestricted
SkylineBuilder (Wu et al., 2020)	34.4	34.2	-	34.2	-	-	-	-
DPR (Karpukhin et al., 2020)	-	40.8	-	41.5	-	-	-	57.9
DPR (our implementation)	38.4	40.2	40.2	40.2	-	-	-	-
RAG (Lewis et al., 2020b)	**43.5**	44.1	44.1	44.5	-	-	-	56.1
FiD-base (Izacard and Grave, 2020b)	39.5	42.9	45.3	48.2	53.9	57.9	60.7	65.0
Ours (APE+FiD-base)	**40.3**	**43.7**	**46.0**	48.2	**55.4***	**59.0***	**62.0***	65.0
FiD-large (Izacard and Grave, 2020b)	42.5	45.8	48.3	51.4	57.2	60.6	63.7	67.6
Ours (APE+FiD-large)	**43.4**	**46.6**	**49.1**	51.4	**57.9**	**61.4***	**64.1***	67.6

Table 2: Exact match scores on NaturalQuestions and TriviaQA test sets. * indicates statistical significance.

	NaturalQuestions				**TriviaQA**			
	Top-5	Top-10	Top-20	Top-100	Top-5	Top-10	Top-20	Top-100
BM25 (Lee et al., 2019)	-	-	59.1	73.7	-	-	66.9	76.7
DPR (Karpukhin et al., 2020)	67.1	-	78.4	85.4	-	-	79.4	85.0
FiD (Izacard and Grave, 2020b)	66.2	73.9	79.2	86.1	69.8	74.9	78.9	84.8
Ours (APE+FiD-base)	**67.4***	75.1*	**80.4***	86.1	**70.8***	**75.8***	**79.5**	84.8
Ours (APE+FiD-large)	67.2	**75.4***	80.2*	86.1	70.4	75.6*	79.2	84.8

Table 3: Top-k retrieval accuracy scores on NaturalQuestions and TriviaQA test sets. * indicates statistical significance.

hours.

4.2 Experimental Results

As shown in Table 2 under restricted top-k, our proposed method improves upon the FiD model on both datasets, and by a statistically significant margin on TriviaQA. It also outperforms the previous AC method (Wu et al., 2020) by 12.4% when $k = 10$ due to the stronger base model. The addition of APE allows FiD to significantly outperform RAG (Lewis et al., 2020b) on NaturalQuestions when $k \in \{10, 20\}$.

Previous adaptive computation methods (Wu et al., 2020; Schwartz et al., 2020b) was reported to have plateaued or degraded performances in the unrestricted setting. However, Table 2 shows that our approach does not have this issue.

4.3 Analysis of Passage Quality

To understand how APE outperforms the baselines, we analyse the quality of the final top-k passages retained by APE. Table 3 reports the top-k retrieval accuracy of the top-k passages. The results show that the top-k accuracy of the selected collection of documents by APE is significantly better than BM25, DPR, and FiD, which are strong retrieval baselines for ODQA. Combined with Table 2, it indicates that the better passage quality yielded by APE helps to improve the end ODQA performance of the model.

5 Conclusions

In this work, we explore an adaptive computation method that can be efficiently applied to an existing generative ODQA model. We find that, by replacing the encoder of generative ODQA models with our proposed adaptive passage encoder, we can train an effective adaptive computation policy without tuning the base model. This allows applying adaptive computation to large state-of-the-art generative models, which was previously challenging computation-wise. Our experimental results show that our method produces more accurate results than a state-of-the-art generative model on both NaturalQuestions and TriviaQA, and it outperforms the previous AC method by a large margin. The analysis also shows that our approach achieves better passage quality that leads to improvements in ODQA performance.

Acknowledgments

The first author would like to thank his wife Jane for her love and support throughout the years. We would also like to thank Gautier Izacard and Edouard Grave for their help with using FiD. This research was supported by the European Union's Horizon 2020 research and innovation programme under grant agreement no. 875160.

References

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *ICLR*. OpenReview.net.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *ACL (1)*, pages 1870–1879. Association for Computational Linguistics.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *ACL (1)*, pages 845–855. Association for Computational Linguistics.

Gautier Izacard and Edouard Grave. 2020a. Distilling knowledge from reader to retriever for question answering. *CoRR*, abs/2012.04584.

Gautier Izacard and Edouard Grave. 2020b. Leveraging passage retrieval with generative models for open domain question answering. *CoRR*, abs/2007.01282.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL (1)*, 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 *EMNLP (1)*, pages 6769–6781. Association for Computational Linguistics.

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 *ACL (1)*, pages 6086–6096. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *ACL*, pages 7871–7880. 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. 2020b. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. 2020. Fastbert: a self-distilling BERT with adaptive inference time. In *ACL*, pages 6035–6044. Association for Computational Linguistics.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021. Reader-guided passage reranking for open-domain question answering. *CoRR*, abs/2101.00294.

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. A discrete hard EM approach for weakly supervised question answering. In *EMNLP/IJCNLP (1)*, pages 2851–2864. Association for Computational Linguistics.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. Ambigqa: Answering ambiguous open-domain questions. In *EMNLP (1)*, pages 5783–5797. Association for Computational Linguistics.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. *CoRR*, abs/1901.04085.

Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the behaviors of BERT in ranking. *CoRR*, abs/1904.07531.

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.

Stephen Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation*, 60(5):503–520.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020a. Green AI. *Commun. ACM*, 63(12):54–63.

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A. Smith. 2020b. The right tool for the job: Matching model and instance complexities. In *ACL*, pages 6640–6651. Association for Computational Linguistics.

Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. 2019. Multi-passage BERT: A globally normalized BERT model for open-domain question answering. In *EMNLP/IJCNLP (1)*, pages 5877–5881. Association for Computational Linguistics.

R. J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Yuxiang Wu, Sebastian Riedel, Pasquale Minervini, and Pontus Stenetorp. 2020. Don't read too much into it: Adaptive computation for open-domain question answering. In *EMNLP (1)*, pages 3029–3039. Association for Computational Linguistics.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. Deebert: Dynamic early exiting for accelerating BERT inference. In *ACL*, pages 2246–2251. Association for Computational Linguistics.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with bertserini. In *NAACL-HLT (Demonstrations)*, pages 72–77. Association for Computational Linguistics.

A Experimental Details

A.1 Hyper-parameters

Hyper-parameter	Value
learning rate	1e-4
batch size	24
epoch	2
optimiser	Adam
Adam ϵ	1e-6
Adam (β_1, β_2)	(0.9, 0.999)
max sequence length	256
pooling	max-pooling
number of passages	5/10/20
device	Nvidia V100

Table 4: Hyper-parameters for the HasAnswer model training.

Hyper-parameter	Value
learning rate	0.01
batch size	24
epoch	1
optimiser	Adam
max number of steps	240
step cost c	0.1
discount factor γ	0.8
hidden size of MLPs	64
number of passages	20/30/50

Table 5: Hyper-parameters for scheduler model REINFORCE training.