Decoupled Transformer for Scalable Inference in Open-domain Question Answering

Haytham ElFadeel
haythamf@fb.com

Stan Peshterliev
stanvp@fb.com

Abstract

Large transformer models, such as BERT, achieve state-of-the-art results in machine reading comprehension (MRC) for open-domain question answering (QA). However, transformers have a high computational cost for inference which makes them hard to apply to online QA systems for applications like voice assistants. To reduce computational cost and latency, we propose decoupling the transformer MRC model into input-component and cross-component. The decoupling allows for part of the representation computation to be performed offline and cached for online use. To retain the decoupled transformer accuracy, we devised a knowledge distillation objective from a standard transformer model. Moreover, we introduce learned representation compression layers which help reduce by four times the storage requirement for the cache. In experiments on the SQUAD 2.0 dataset, a decoupled transformer reduces the computational cost and latency of open-domain MRC by 30-40% with only 1.2 points worse F1-score compared to a standard transformer.

1 Introduction

Open-domain question answering (QA) aims to answer questions from a collection of text passages. It is an important and challenging task with application to several domains such as web search and voice assistants. The most popular architecture for open-domain QA is *retriever-reader* (Chen et al., 2017). Given a question, the retriever uses an information retrieval (IR) system over a collection of passages to return top-K results that are most likely to contain an answer. Then, the reader uses a machine reading comprehension (MRC) model on each of the top-K results to find an answer. In the end, the top-K MRC answers are ranked to produce a final answer.

For both the retriever and the reader, large transformer models such as BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019) and ELECTRA (Clark et al., 2020) achieve state-of-the-art results. A disadvantage of large transformer models is the high computational cost for inference which makes them hard to apply to online runtime systems, e.g. voice-assistants. Transformers' computational cost comes from three major factors. Firstly, the size of the feed-forward layers which project to an intermediate higher dimension and projects back to the original dimension. Secondly, the multi-head self-attention has quadratic computational complexity in the sequence length. Thirdly, the total number of layers.

Dense passage retrieval (DPR) (Karpukhin et al., 2020) is a retriever model which uses a transformer question encoder and transformer passage encoder to capture semantic similarity. The question and passage encoders are trained such that passages that are likely to contain an answer have a large embedding dot product with the question embedding. The embeddings for the passages are generated offline and indexed for efficient distributed KNN search (Johnson et al., 2017), and only the embedding for the question is generated at runtime. Since questions are usually short, the retriever runtime inference computational cost is low.

MRC reader models process the top-K passages returned by the retriever to get an answer. In transformer-based MRC models, each passage is encoded together with the question using a CLS and separator characters `[CLS] Question [SEP] Passage`. The encoding is followed by a prediction head which determines the answer span. If there is no answer, the model result is a zero-length span on the CLS token. The joint encoding of the document and the question produces rich interaction features but it increases the sequence length, and thus the computational coast. Since the

Proceedings of Recent Advances in Natural Language Processing, pages 386–393
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_044

MRC model inference is executed at runtime on long sequences of a question and passages, MRC is the main computational bottleneck in retriever-reader QA.

There have been several ideas to reduce the runtime inference of transformer models, such as precision reduction via quantization (Zafrir et al., 2019; Shen et al., 2020), knowledge distilling to a smaller architecture (Sanh et al., 2019; Jiao et al., 2019), and approximate multi-head attention for reducing the quadratic complexity (Wang et al., 2020; Beltagy et al., 2020). In this paper, we take an orthogonal approach, decoupling the transformer encoding for multiple inputs to improve efficiency, which can be combined with the aforementioned techniques. The motivation for the decoupled transformer is that in open-domain QA the passages are known in advance and part of the passage computation can be performed offline and stored. Then, online at the runtime the question computation can be performed only once and combined with the stored state from passages with cross-attention.

We use the decoupled transformer to reduce the computational cost of open-domain MRC by 30-40% with only 1.2 points worse than the F1-score on the SQUAD 2.0 benchmark.

Our contributions are as follows:

- We propose and evaluate a novel decoupled transformer approach for MRC in open-domain QA to reduce runtime inference cost. Our approach uses a knowledge distillation (KD) objective to bridge the gap between a standard transformer and decoupled transformer.
- We conduct experiments to understand how much cross-attention between inputs is needed in MRC and other natural language processing (NLP) tasks like paraphrasing identification and natural language inference.
- We devise an accurate representation compression approach to reduce the storage requirement for decoupled transformer offline state. The compression provides a four-fold reduction in the index storage requirement for large corpora such as Wikipedia, from 3.4 TB to 858 GB.

2 Related Work

DC-BERT (Zhang et al., 2020) is a decoupled transformer that has dual BERT models. An online BERT encodes the question and an offline BERT pre-encodes all the passages and stores their encodings. Conceptually, DC-BERT goals and architecture of combining local and global context are similar to our work with the following major differences:

- We apply decoupled transformers to MRC and DC-BERT is designed and evaluated for the retrieval passage ranking. With the recent introduction of DPR, passage ranking as explored in DC-BERT is less important, so MRC becomes the primary bottleneck.
- We investigate how much cross-attention is needed for MRC and other NLP tasks.
- We introduce compression and decompression layers to reduce representation storage requirements.

Another model where the query and the passage are encoded independently using a transformer is ColBERT (Khattab and Zaharia, 2020). The main modeling applications for ColBERT are retrieval and passage ranking. After encoding the query and the passage independently, late interactions are introduced using an efficient sum of maximum similarity computations. ColBERT representations are used for retrieval, so it combines the strengths of DPR and DC-BERT. However, the efficient late interactions in ColBERT do not have enough representation power for complex tasks like MRC.

3 Decoupled Transformer

In the decoupled transformer, Figure 1, we split the transformer model M into two components.

1. Input-component M_{input} (the lower N layers) which processes the inputs independently and produces a representation. The representation for the inputs that are known in advance, i.e. the passages, is stored and used without computation.
2. Cross-component M_{cross} (the higher M layers) which processes the inputs jointly (after concatenation) and produces the final output.

3.1 Workflow

The workflow is depicted in Figure 2. Offline, we run the input-component M_{input} on each passage from the collection of passages and store the representation in the search index. Moreover, we compress the stored passage representation to reduce

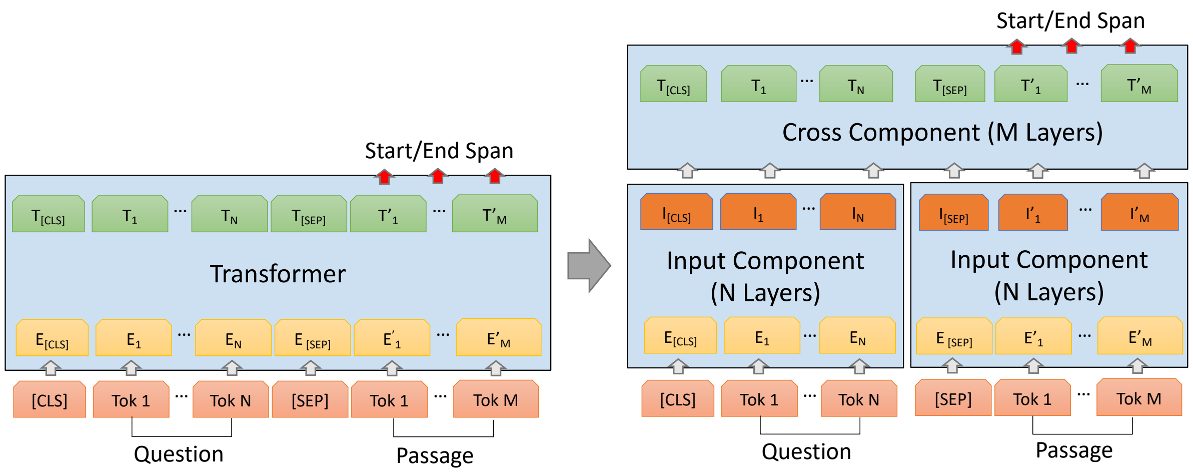
Start/End Span
Transformer
Question
Passage
Start/End Span
Cross Component (M Layers)
Input Component (N Layers)
Input Component (N Layers)
Question
Passage

Figure 1: On the left, standard transformer model for MRC. On the right, decoupled transformer model with input-component and cross-component.

storage requirements. The offline step is performed together with the DPR indexing.

At runtime, using DPR we retrieve the candidate passages with their stored representation which we decompress. Then, we compute the question representation using the input-component M_{input}. Finally, we concatenate the question representation with the representation of the passage and process them with the cross-component M_{cross}.

3.2 Benefits

The decoupled transformer reduces per question transformer complexity in lower N layers from $O(N_p(L_q + L_p)^2)$ to $O(L_q^2 + N_pL^2p)$ where N_p denotes the number of top-K passages per question, L_q and L_p denote the average number of tokens of each question and passage.

At runtime, the computation for the lower N layers for the passage is eliminated because it is performed once offline and reused. Moreover, the computation for the lower N layers for the question is done only once for the top-K retrieved passages, and not repeated, as opposed to the normal transformer which uses all layers on both the question and the top-K retrieved passages.

3.3 Initialization

To build a decoupled transformer model, we start from a standard transformer such as BERT, RoBERTa, ELECTRA model which is fine-tuned on a target dataset such as SQUAD 2.0. Then, we create the decoupled transformer model by splitting the encoder layers into input and cross components which are initialized with the fine-tuned MRC model weights. In addition to the standard

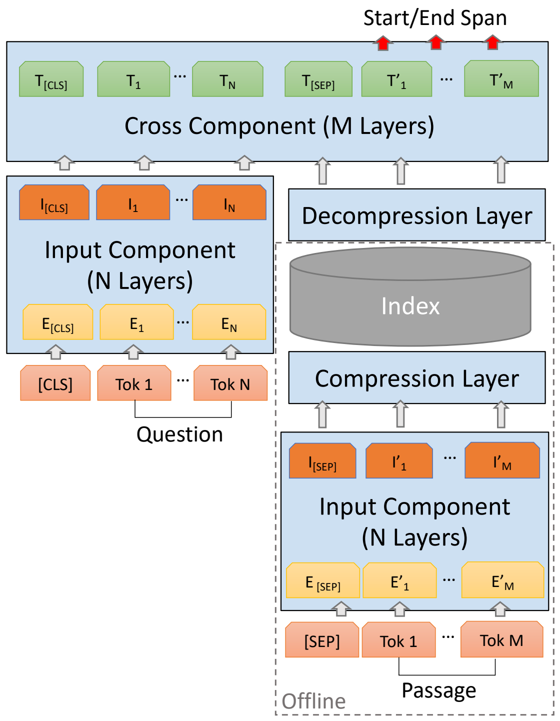
Start/End Span
Cross Component (M Layers)
Input Component (N Layers)
Question
Decompression Layer
Index
Compression Layer
Input Component (N Layers)
Passage
Offline

Figure 2: Decoupled model offline indexing and runtime. The passage is processed with the input module, then the state representations are compressed using a projection layer and stored in the index. At runtime, the passage is retrieved by DPR from the index, decompressed to the original representation size and used by the cross-component.

transformer weights, we create a global position embedding and segment embedding layers at the start of the cross-component and initialize them to the same weight as the local position and segment embedding from the input-component. The global position and segment embedding re-encode the tokens for the new position in the concatenated question-document encoding sequence. The segment embedding differentiates whether the encoded token is from the question or document.

3.4 Training Objective

During decoupled transformer training, we aim to preserve the standard transformer model accuracy. To achieve that, we propose a knowledge distillation (KD) (Hinton et al., 2015) objective from the standard transformer to decoupled transformer which helps preserve the original representation.

The objective function is the sum of four terms:

$$\begin{aligned} L = &(1-\lambda)CE(\text{y}, \text{target}) && (1)\\ &+ \lambda KL(\text{logits/T}, \text{teacher-logits/T}) && (2)\\ &+ \sigma MSE(\text{represent}_n, \text{teacher-represent}_n) && (3)\\ &+ \sigma MSE(\text{attention}_n, \text{teacher-attention}_n) && (4) \end{aligned}$$

1. A standard cross-entropy (CE) loss with the prediction y and hard targets from ground truth labels.
2. KD loss based on Kullback–Leibler (KL) divergence with logits from the teacher standard transformer model. We scale the targets with the same temperature T for both the teacher and student.
3. The mean square error (MSE) between the decoupled model final layer representation with the original model final layer representation.
4. The MSE between the decoupled model final layer multi-head self-attention output with the standard model final layer multi-head self-attention output.

The parameter λ determines the relative contribution of CE and KL losses. And, σ is a weight for the MSE losses.

The MSE losses on the final layer representation and the final layer self-attention are similar to TinyBERT (Jiao et al., 2019) approach to smaller model distillation. Unlike TinyBERT, we only apply the MSE losses only on the last layer. The motivation for the MSE losses is that we are aiming to make the representation at the end of the decoupled transformer to match the representation of the standard transformer.

4 Representation Compression

In open-domain QA the collection of passages are known in advance. So, with decoupled transformer, we run the input-component M_{input} on each passage offline and store the passage representation in the index. For a large corpus, the representation storage can be a significant amount. In the case of QA over Wikipedia, the storage requirement is around 3.4TB given around 32 million passages, averaging 150 tokens per passage, and 768 token dimensions for the BERT-base model with 16-bit precision.

To reduce the storage requirements for the passage representation of the decoupled transformer, we introduce a *compression layer* at the end of the input-component and a *decompression layer* at the start of the cross-component, see Figure 2. The compression layer is a linear projection from the original dimension to a compression dimension. The decompression layer is a linear projection from the compression dimension to the original dimension. These layers are similar to an autoencoder with a bottleneck.

4.1 Training Procedure

To train the compression and decompression layers we start from a decoupled transformer model. Then, we perform training in two phases:

- **Phase 1.** We train the randomly initialized compression and decompression layers to reconstruct the input-component output representation without updating the decoupled transformer model itself.
- **Phase 2.** We train the compression and decompression layers together with the decoupled transformer jointly. This means the cross-component receives the decompressed representation.

The intuition behind the two-phase approach is that since compression and decompression layers are randomly initialized, it is beneficial to first train the compression and decompression layers independent from the decoupled transformer to get near-optimal weights. Then, train the cross-component of the model to understand the slightly different decompressed representation.

5 Experiments and Results

5.1 Datasets

We evaluate the decoupled transformer on SQUAD 2.0 (Rajpurkar et al., 2018) which is a popular MRC dataset over Wikipedia articles. In addition to MRC, we evaluate models on the datasets below to understand how many cross-components layers are needed for tasks of different complexity and dataset size.

- QQP (Chen et al., 2018) and MRPC (Dolan and Brockett, 2005) datasets for paraphrasing identification. The task is given two sentences, recognizing if they are paraphrases or not.
- MNLI (Williams et al., 2018) dataset for natural language inference datasets. The task is given two sentences the "premise" and the "hypothesis", to determine if the hypothesis entails, contradicts, or is neutral given the premise.

Hyperparameter	**Value**
Warmup steps	200
Learning Rate (LR)	5e-5
Layer-wise LR Multiplier	0.95
Batch size per GPU	32
Number of GPUs	2
Adam ϵ, β_1, β_2	1e-6, 0.9, 0.999
Attention Dropout	0.1
Dropout	0.1
Weight Decay	Linear
Gradient Clipping	3.0
Epochs	4
KD temperature T	3.0
Loss weight λ	0.95
Loss weight σ	0.5

Table 2: Model training hyperparameters.

5.2 Setup

Models. We use ROaD-base (ElFadeel and Peshterliev, 2021) for the MRC experiments on SQUAD 2.0. ROaD is an ELECTRA model pretrained and distilled using multi-task learning. For the experiments on QQP, MRPC, and MNLI we use ELECTRA-base. All models are implemented in PyTorch and optimized using Adam (Kingma and Ba, 2014).

Hyperparameters. Table 2 shows the hyperparameters that we use for fine-tuning the standard transformer and training the decoupled transformer. We searched different values for the temperature T, and the weights λ and σ. For λ, we experimented with 0.5, 0.7, 0.9, 0.95 values and we found that for decoupled transformer training a large λ that biases towards the KL divergence objective work best. For σ, we experimented with $0.25, 0.5, 0.75, 1.0$ values and we found that smaller values work better because otherwise the KL divergence objective is given less weight which leads to worse models.

Hardware. We perform the experiments and benchmarks on Nvidia Titan RTX with tensor cores GPU and AMD Ryzen Threadripper 3960X - 24 cores CPU.

5.3 Decoupled Transformer

First, we perform a set of experiments on a decoupled transformer without compression. For each experiment, we denote the decoupled transformer split configuration as x-y, where x is the number of input-component layers, and y is the number of cross-component layers.

Table 1 shows the performance and FLOPs starting from the baseline standard transformer model to decoupled transformer with extreme 11-1 split. We observe that tasks with a large dataset (QQP, SQUAD, MNLI contain over 100K samples each) have similar behavior with a noticeable drop when moving from decoupled transformer 5-7 split to 6-6 split, and another big drop when the number of cross-component layers becomes less than 3. While in MRPC, a small dataset with around 5K sample, the drop of performance was significant

Task	**Baseline**	**1-11**	**2-10**	**3-9**	**4-8**	**5-7**	**6-6**	**7-5**	**8-4**	**9-3**	**10-2**	**11-1**
SQUAD 2.0	87.6	87.5	87.2	87.1	87.0	86.7	85.4	85.2	84.8	84.0	80.6	62.6
QQP	91.5	91.1	91.0	90.9	90.9	90.9	90.5	90.4	90.4	90.0	89.0	86.4
MNLI	88.9	87.3	87.1	86.9	86.7	86.7	86.7	86.4	86.4	85.6	77.0	73.5
MRPC	89.5	87.7	86.3	85.5	83.0	80.1	78.1	77.9	77.8	77.7	71.8	71.6
FLOPs	1.0	.91	.83	.75	.66	.58	.50	.41	.33	.25	.16	0.08

Table 1: Decoupled transformer results with variable number of input-component and cross-component layers. Baseline is a standard transformer model. The x-y columns indicate the number of input-component and cross-component layers. We use F1 score for SQUAD 2.0 and accuracy for QQP, MNLI, and MRPC. There is a consistent trend for performance to degrade as we increase the number of input-component layers and decrease the number of cross-component layers. FLOPs is floating point operations for inference as a measure of computational cost.

and bigger than the other large datasets even with the decoupled transformer with 1-11 split.

With every layer, we moved from the cross-component to the input-component, the FLOPs decreased by about 8% and performance dropped by a small amount until the number of input-component layers equals to or bigger than the cross-component layers. The results show that choosing the right setting is application-specific and the best option depends on the particular performance and latency trade-offs.

For the following experiments, we use the 5-7 split because it provides the best trade-offs between accuracy and FLOPs across the evaluated datasets.

MRC Model	**SQUAD 2.0**	
	F1	**EM**
Decoupled 5-7	86.7	84.1
- SQUAD 2.0 pretraining	84.2	81.5
- training position and segment embedding in the cross-model	82.1	80.0
- KL objective	84.0	80.4
- MSE on representation and attention final layer	86.5	83.7
+ MSE on hidden and attention applied to all layers	86.0	83.2

Table 3: Decoupled transformer ablation study for SQUAD 2.0 MRC decoupled transformer with 5-7 split. We remove one row at a time except for the last row where we add MSE losses to all layers and not just the final layer.

Ablations. We perform an ablation study to understand the effect of the different modeling techniques on the decoupled transformer performance. Table 3 shows the results. First, we remove the SQUAD 2.0 pretraining and start with regular ELECTRA-base which reduces F1 significantly by 2.5 points. Then, we tried keeping the position and segment embeddings in the cross-component frozen which hurt F1 as expected. If we remove the distillation KL objective, F1 degrades significantly by 2.7 points. On the other hand, removing the MSE losses on the representation and attention does not cause a significant reduction in F1. However, adding MSE losses on all layers actually causes a reduction in F1 because the CE and KL objectives receive less weight.

Compression Rate	**Dim Size**	**SQUAD 2.0**	
		F1	**EM**
No compression	768	86.7	84.1
2.0x	386	86.6	84.0
3.0x	256	86.5	83.8
4.0x	192	86.4	83.8
4.8x	160	86.2	83.6
6.0x	128	85.2	82.4

Table 4: Decoupled transformer compression rates for SQUAD 2.0 MRC decoupled transformer with 5-7 split. The dim size is the number of embedding dimensions per token. EM is exact match accuracy. FLOPs is floating-point operations for inference as a measure of computational cost.

5.4 Compression

To evaluate compression, we conducted experiments on a decoupled transformer with 5-7 split using the MRC model for SQUAD 2.0. Our goal is to understand how much impact different levels of compression have on the storage requirement and model performance.

Table 4 compares the results with five different levels of compression. We observed the performance degradation is minimal for 2x, 3x and 4x compression, and then it starts to degrade significantly. At 4x compression, the required storage for open-domain QA over Wikipedia with the previous assumptions is 3.4 TB which could be reduced to 858 GB.

Ablations. We evaluate the effectiveness of the two-stage training of the compression and decompression layers. Table 5 shows the results. First, we remove the training of the compression independent from the model fine-tuning which causes a significant 6.6 F1 score reduction. Second, we remove the joint training of compression and MRC layers which cause 1.6 F1 sore drops. Overall, both stages are necessary for training effective compression layers.

5.5 Inference Performance

In addition to FLOPs computational cost analysis, we run inference benchmarks on GPU and CPU. For the benchmarks, we use FP16 PyTorch models without TorchScript. We test in two settings: long and short inputs. Long inputs are 64 words for the question and 448 words for the passage. Short inputs are 16 words for the question and 150 words for the passage. For each setting, we perform four runs and take the average time.

MRC Model	SQUAD 2.0	
	F1	EM
Decoupled 5-7, 4x compress	86.4	83.8
- training compression independent from the model	79.8	77.3
- joint training	84.8	82.3

Table 5: Compression ablation study for MRC SQUAD 2.0 model with 5-7 split. We remove either training compression independent of the model or join training.

Table 6 shows the benchmark results. For CPU the results are close to our FLOPs analysis, and for GPU the we get lower runtime reduction due to the GPU parallelism.

MRC Model	GPU	CPU
	Long / Short (diff)	Long / Short (diff)
Baseline	9.1 / 9.0	3200 / 920
Decoupled 5-7	6.3 / 6.2 (31%)	1890 / 490 (40-46%)
+ 4x compress	6.4 / 6.3 (30%)	1950 / 520 (39-43%)

Table 6: Decoupled transformer inference performance. Baseline is a standard transformer MRC model. Long and short indicate the input length, and the times are in milliseconds. Diff is the difference with the baseline.

Model	FLOPs	SQUAD 2.0	
		F1	EM
DeBERTa	1.2x	86.2	83.1
ROaD	1.0x	87.6	85.1
Decoupled ROaD 5-7, 4x compress	0.6x	86.4	83.8

Table 7: Decoupled transformer with DeBERTA-base and ROaD-base on SQUAD 2.0 model. EM is exact match accuracy. FLOPs is floating point operations for inference as a measure of computational cost.

5.6 Results

Table 7 compares the decoupled ROaD-base 5-7 split model with 4x compression with the ROaD-base model and DeBERTa-base model on SQUAD 2.0 MRC task. The DeBERTa model introduces additional positional embeddings that increase the computational cost by 20%. Still, the decoupled ROaD model achieves comparable accuracy with DeBERTa while requiring two times fewer FLOPs.

6 Conclusion and Future Work

We presented the decoupled transformer model for reducing runtime latency of MRC models in open-domain QA. The decoupling allows for part of the representation computation to be performed offline and cached for online use. To bridge the accuracy gap between a standard transformer and decoupled transformer, we devised knowledge distillation objectives for both model logits and features. Moreover, we introduced a representation compression approach that allows for a four-times reduction in representation storage requirements for open-domain QA without significant loss of accuracy. We use the decoupled transformer to reduce the computational cost of open-domain MRC by 30-40% with only 1.2 points worse than the F1-score on the SQUAD 2.0 benchmark.

In the future, we are planning to extend the decoupled model with a DPR objective. The goal is for the input-component to also produce DPR-like embeddings suitable for similarity search. This way, we can have a single model that acts as both retrieval and reader.

References

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150.*

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051.*

Zihan Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. 2018. Quora question pairs. *University of Waterloo.*

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555.*

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805.*

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005).*

Haytham ElFadeel and Stan Peshterliev. 2021. Robustly optimized and distilled training for natural language understanding. *arXiv preprint arXiv:2103.08809.*

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 39–48.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8815–8821.

Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. The multi-genre nli corpus.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8bert: Quantized 8bit bert. *arXiv preprint arXiv:1910.06188*.

Yuyu Zhang, Ping Nie, Xiubo Geng, Arun Ramamurthy, Le Song, and Daxin Jiang. 2020. Dc-bert: Decoupling question and document for efficient contextual encoding. *arXiv preprint arXiv:2002.12591*.