NarrowBERT: Accelerating Masked Language Model Pretraining and Inference

Anonymous ARR submission

Abstract

Large-scale language model pretraining is a very successful form of self-supervised learning in natural language processing, but it is increasingly expensive to perform as the models and pretraining corpora have become larger over time. We propose NarrowBERT, a modified transformer encoder that increases the throughput for masked language model pretraining by more than $2\times$. NarrowBERT sparsifies the transformer model such that the self-attention queries and feedforward layers only operate on the masked tokens of each sentence during pretraining, rather than all of the tokens as with the usual transformer encoder. We also show that NarrowBERT increases the throughput at inference time by as much as $3.5\times$ with minimal (or no) performance degradation on sentence encoding tasks like MNLI. Finally, we examine the performance of NarrowBERT on the IMDB and Amazon reviews classification and CoNLL NER tasks and show that it is also comparable to standard BERT performance.

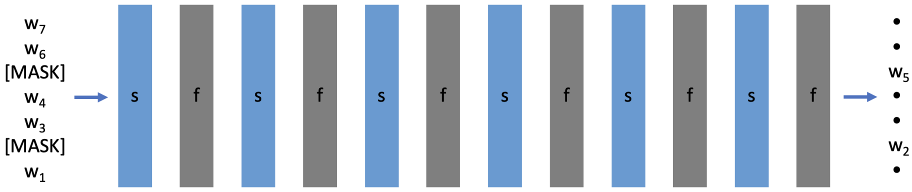
1 Introduction

Pretrained masked language models, such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and DeBERTa (He et al., 2021), have pushed the state-of-the-art in a wide range of downstream tasks in natural language processing. At their core is the transformer architecture (Vaswani et al., 2017) that consists of interleaved self-attention and feedforward sublayers. Since the former sublayer implies quadratic time complexity in the input sequence length (Vaswani et al., 2017), many have proposed methods to make the self-attention computation more efficient (Katharopoulos et al., 2020; Choromanski et al., 2021; Wang et al., 2020; Peng et al., 2021, 2022, *inter alia*).

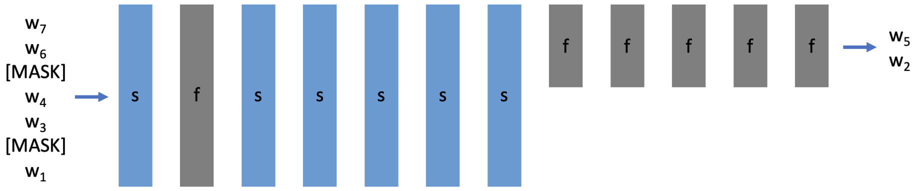
In this work, we explore an orthogonal approach to efficiency: can we make masked language models efficient by *reducing* the length of the input sequence that each layer needs to process? In particular, pretraining by masked language modeling only involves prediction of masked tokens (typically, only 15% of the input tokens; Devlin et al., 2019; Liu et al., 2019). Despite this sparse pretraining objective, each transformer layer computes a representation for every token. In addition to pretraining, many downstream applications only use a single vector representation (i.e., only the `[CLS]` token) for prediction purposes, which is much smaller than the number of input tokens (e.g., sequence classification tasks as in GLUE/SuperGLUE; Wang et al., 2018, 2019). By narrowing the input sequence for transformer layers, we can accelerate both pretraining and inference.

We present NarrowBERT, a new architecture that takes advantage of the sparsity in the training objective. We present two NarrowBERT methods in the sections that follow (Figure 1). We provide the code to reproduce our experiments at `redacted-during-review`. The first method reduces the input sequence for the feedforward sublayers by reordering the interleaved self-attention and feedforward sublayers in the standard transformer architecture (Press et al., 2020): after two standard, interleaved transformer layers, self-attention sublayers are first applied, followed only by feedforward sublayers. This way, the feedforward sublayer computations are only performed for *masked tokens*, resulting in a $1.3\times$ speedup in pretraining (§3). The second approach reduces the input length to the attention sublayers: *queries* are only computed for masked tokens in the attention mechanism (Bahdanau et al., 2015), while the *keys* and *values* are not re-computed for non-masked tokens, which leads to a greater than $2\times$ speedup in pretraining.

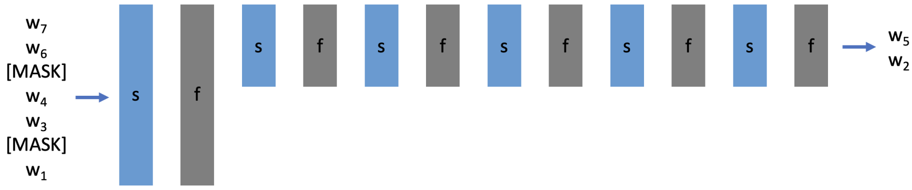
We extensively evaluate our efficient pretrained models on well-established downstream tasks (e.g., Wang et al., 2018; Tjong Kim Sang and De Meulder, 2003.) We find that our modifications result in almost no drop in downstream performance,



(a) `{6,sf}` model: standard BERT with the transformer encoder, trained on MLM loss.



(b) `sf{5,s}:{5,f}` ContextFirst model: Transformer encoder with re-ordered layers. Attentional contextualization is performed all-at-once near the beginning of the model.



(c) `sf:{5,sf}` SparseQueries model: Transformer encoder with sparsified queries. Contextualization is focused on [MASK] tokens only. (See Fig. 2.)

Figure 1: Examples of standard BERT and NarrowBERT variations. NarrowBERT takes advantage of the sparsity in the masking (i.e., only 15% of tokens need to be predicted) to reduce the amount of computation in the transformer encoder.

while providing substantial pretraining and inference speedups (§3). While efficient attention variants are promising research directions, this work presents a different and simple approach to making transformers efficient, with minimal changes in architecture.

2 NarrowBERT

In Figures 1b and 1c, we illustrate two variations of NarrowBERT. We define some notation to describe the configuration of our models. `s` refers to a **single self-attention layer** and `f` refers to a **single feedforward layer**. The colon `:` refers to the **'narrowing' operation**, which gathers the masked positions from the output of the previous layer.

The first variation ('ContextFirst' in Fig. 1b) uses attention to contextualize all-at-once at the beginning of the model. In short, the transformer layers have been rearranged to frontload the attention components. The example given in the figure specifies the model as `sf{5,s}:{5,f}`, which means that the input sentence is encoded by a self-attention layer, a feedforward layer, and 5 consecutive self-attention layers. At that point, the masked positions from the encoded sentence are gathered into a tensor and passed through 5 feedforward layers, **thereby avoiding further computations for all non-masked tokens**. Finally, the masked positions are unmasked and the MLM loss is computed.

The second variation ('SparseQueries' in Fig. 1c) does not reorder the layers at all. Instead, the `sf:{5,sf}` model contextualizes the input sentence in a more limited way. As shown in Figure 2, the input sentence is first contextualized by a `s` and a `f` layer, but the non-masked tokens are never contextualized again afterwards. Only the masked tokens are contextualized by the remaining `{5,sf}` layers.

Since the masked tokens are only about 15% of the total sentence length, the potential speedup is ~6.6× for every feedforward or attention layer downstream of a narrowing `:` operation. The memory usage can also decrease by ~6.6× for those layers since the sequence length has decreased, which allows us to use larger batch sizes during training.

For GLUE, Amazon, and IMDB text classification tasks, only the `[CLS]` token is used for prediction. When we finetune or predict with ContextFirst on a GLUE/Amazon/IMDB task, the feedforward layers only need to operate on the `[CLS]` token. When we finetune or predict with SparseQueries, only the `[CLS]` token is used in the queries of the

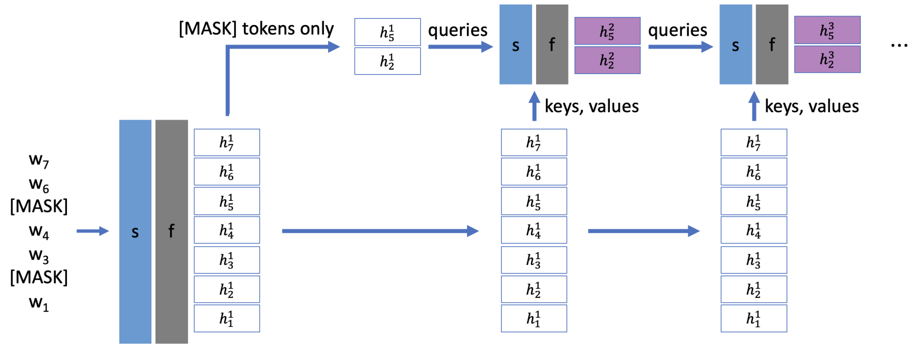


Figure 2: Sparse queries in the attention layers. Only the masked positions are contextualized as query vectors in subsequent `s` layers. The inputs are contextualized once by the first `s` layer and `f` layer, and reused as the keys and values in all subsequent attention layers.

	Pretrain Speedup	Finetune Speedup	Inference Speedup	GLUE MNLI	QNLI	SST2	STS-B	QQP	WNLI
Baseline BERT (`{12,sf}`)	1×	1×	1×	0.83	0.91	0.93	0.89	0.87	0.56
Funnel Transformer (B4-4-4)	0.88×	0.86×	0.78×	0.78	0.87	0.88	0.86	0.86	0.56
ContextFirst (`sfsf{10,s}:{10,f}`)	1.33×	1.24×	1.64×	0.82	0.90	0.91	0.89	0.87	0.56
SparseQueries:									
`{1,sf}:{11,sf}`	2.47×	4.73×	4.64×	0.77	0.87	0.89	0.84	0.80	0.56
`{2,sf}:{10,sf}`	2.34×	2.82×	3.49×	0.81	0.88	0.91	0.88	0.87	0.59
`{3,sf}:{9,sf}`	2.15×	2.43×	2.79×	0.81	0.89	0.91	0.86	0.87	0.56
`{4,sf}:{8,sf}`	1.63×	2.13×	2.33×	0.82	0.88	0.91	0.89	0.87	0.57

Table 1: Test scores on various GLUE tasks. ('MNLI' scores refer to the MNLI matched dev set.) Finetuning and inference speedups refer to speeds on the MNLI task.

attention layers. Everything after the narrowing : operation only operates on the [CLS] token, which dramatically speeds up the NarrowBERT variants.

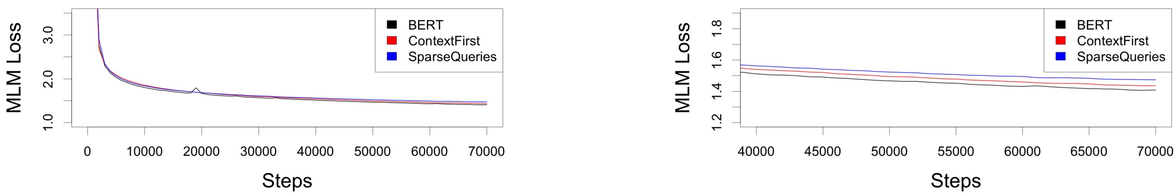
3 Experiments

We focus on 2 models in our experiments: ContextFirst (`sfsf{10,s}:{10,f}`) and SparseQueries (`{1,sf}:{11,sf}`, $\cdots$, `{4,sf}:{8,sf}`). Our NarrowBERT models all contain 12 self-attention and 12 feedforward layers in total, with the narrowing operation used at different points in the model. We compare NarrowBERT with the baseline BERT model and the Funnel Transformer model (Dai et al., 2020), which is a pretrained encoder-decoder transformer model where the encoder goes through a sequence of length bottlenecks.

In our experiments, we use 15% masking in masked language model (MLM) training. Following Liu et al. (2019), we do not use next sentence prediction as a pretraining task. We use large batch sizes and high learning rates to fully utilize GPU memory, as suggested in Izsak et al. (2021). Batches are sized to be the largest that fit in GPU memory. We use a learning rate of 0.0005. Models are trained for 70k steps, where each step contains 1728 sequences of 512 tokens, and gradient accumulation is used to accumulate the minibatches needed per step. Models were trained on hosts with 8 Nvidia A100 GPUs. We used the Hugging Face implementations of the baseline BERT and Funnel Transformer models. We pretrained the baseline BERT, Funnel Transformer, and NarrowBERT models using the same Wikipedia and Books corpora and total number of steps.

In Figure 3, we see the evolution of the development MLM loss over the course of model training. The BERT and NarrowBERT models all converge to similar values, with the NarrowBERT models reaching a slightly higher MLM loss near the end of training.

We report the accuracy for MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), SST2 (Socher et al., 2013), WNLI (Levesque et al., 2012), IMDB (Maas et al., 2011), and English Amazon reviews (Keung et al., 2020), F1 for QQP (Sharma et al., 2019) and CoNLL-2003 NER (Tjong Kim Sang and De Meulder, 2003), and Spearman correlation for STS-B (Cer et al., 2017). For the Amazon reviews corpus, we consider both



(a) All training steps.

(b) Near the end of training.

Figure 3: Development MLM loss over the course of pretraining. At the end of training, the BERT, ContextFirst, and SparseQueries (`{2,sf}:{10,sf}`) dev MLM losses are 1.41, 1.43, and 1.47 respectively.

	CoNLL NER	IMDB	Amazon2	Amazon5
Baseline BERT (`{12,sf}`)	0.90	0.93	0.96	0.66
Funnel Transformer	0.87	0.92	0.95	0.65
ContextFirst (`sfsf{10,s}:{10,f}`)	0.89	0.93	0.95	0.65
SparseQueries:				
`{1,sf}:{11,sf}`	0.87	0.91	0.94	0.65
`{2,sf}:{10,sf}`	0.89	0.91	0.95	0.65
`{3,sf}:{9,sf}`	0.89	0.92	0.95	0.65
`{4,sf}:{8,sf}`	0.89	0.93	0.95	0.65

Table 2: Test scores on CoNLL NER, IMDB, binarized Amazon reviews, and 5-star Amazon reviews tasks.

the usual 5-star prediction task and the binarized (i.e., 1-2 stars versus 4-5 stars) task.

In Table 1, we present the results for our extrinsic evaluation on various GLUE tasks. The reduction in performance is small or non-existent, and on WNLI, the NarrowBERT variations perform better than the baseline. For SparseQueries, it is clear that using more layers prior to the narrowing operation improves performance, though the training and inference speedups become smaller. We note that the Funnel Transformer implementation in Pytorch is slower than the baseline BERT model; this may be due to the fact that the original implementation was written in Tensorflow and optimized for Google TPUs.[1]

In Table 2, we provide results on the IMDB and Amazon reviews classification tasks and the CoNLL NER task. Generally, NarrowBERT is close to the baseline in performance, and the SparseQueries performance improves as more layers are used before the narrowing operation.

It is well known that the variability in the performance of BERT on certain GLUE tasks is extreme (Mosbach et al., 2020; Dodge et al., 2020; Lee et al., 2019), where the differences in performance between finetuning runs can exceed 20% (absolute). We have also observed this extreme variability in the course of our own GLUE finetuning experiments. While many techniques have been proposed to address this issue, it is not the goal of this work to apply finetuning stabilization methods to maximize BERT's performance. For this reason, we have excluded the RTE, MRPC, and COLA tasks (which are high-variance tasks studied in the aforementioned papers) from our evaluation.

4 Discussion and Conclusion

We have explored two straightforward ways of exploiting the sparsity in the masked language model loss: rearranging the layers of the transformer encoder to allow the feedforward components to avoid computations on the non-masked positions, and sparsifying the queries in the attention mechanism to only contextualize the masked positions. The NarrowBERT variants can speed up training by a factor of ~2× and inference by a factor of ~3×, while maintaining very similar performance on GLUE, IMDB, Amazon, and CoNLL NER tasks. Based on the favorable trade-off between speed and performance seen in Section 3, we recommend that practitioners consider using the SparseQueries NarrowBERT model with 2 or 3 layers before narrowing.

[1] See https://github.com/laiguokun/Funnel-Transformer. In their paper, the Funnel Transformer authors claim to have a finetuning FLOPs that is 0.58× of the BERT baseline's.

Limitations

Due to our budget constraint, we only performed pretraining and downstream experiments with base-sized transformer models. We also only applied the masked language modeling objective, but there are other effective pretraining objectives (e.g., Clark et al., 2020). Nonetheless, since we introduced minimal changes in architecture, we hope that subsequent work will benefit from our narrowing operations and conduct a wider range of pretraining and downstream experiments. While pretrained models can be applied to even more downstream tasks, we designed a reasonable task suite in this work, consisting of both GLUE sentence classification and the CoNLL NER sequential classification tasks.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. 2021. Rethinking attention with Performers. In *Proc. of ICLR*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *Proc. of ICLR*.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. 2020. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. In *Proc. of NeurIPS*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: decoding-enhanced bert with disentangled attention. In *Proc. of ICLR*.

Peter Izsak, Moshe Berchansky, and Omer Levy. 2021. How to train BERT with an academic budget. In *Proc. o EMNLP*.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proc. of ICML*.

Phillip Keung, Yichao Lu, György Szarvas, and Noah A Smith. 2020. The multilingual amazon reviews corpus. *arXiv preprint arXiv:2010.02573*.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2019. Mixout: Effective regularization to finetune large-scale pretrained language models. *arXiv preprint arXiv:1909.11299*.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Proc. of KR*.

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

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. *arXiv preprint arXiv:2006.04884*.

Hao Peng, Jungo Kasai, Nikolaos Pappas, Dani Yogatama, Zhaofeng Wu, Lingpeng Kong, Roy Schwartz, and Noah A. Smith. 2022. ABC: Attention with bounded-memory control. In *Proc. of ACL*.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. 2021. Random feature attention. In *Proc. of ICLR*.

Ofir Press, Noah A. Smith, and Omer Levy. 2020. Improving transformer models by reordering their sublayers. In *Proc. of ACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proc. of EMNLP*.

Lakshay Sharma, Laura Graesser, Nikita Nangia, and Utku Evci. 2019. Natural language understanding with the quora question pairs dataset.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proc. of CoNLL*.

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 *Proc. of NeurIPS*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Proc. of NeurIPS*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proc. of BlackboxNLP*.

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

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proc. of NAACL*.