2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) | 978-1-7281-8643-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISPASS51385.2021.00025

Reducing BERT Computation by Padding Removal and Curriculum Learning

Wei Zhang, Wei Wei, Wen Wang, Lingling Jin and Zheng Cao
Alibaba Group
{wz.ww, w.wei, w.wang, l.jin, zhengzhi.cz}@alibaba-inc.com

I. Introduction

BERT [1] is very computationally expensive, which is a hurdle for its training and deployment. This work focuses on removing the unnecessary computation due to input padding in BERT. The input of BERT consists of two concatenated sentences. If the length of the two concatenated sentences is shorter than the maximum sequence length, padding must be added to the end of the sentences to fill the empty slots in the input. Because the lengths of sentences vary greatly, there can be a large amount of padding in input. For the English Wikipedia & BooksCorpus dataset, the percentage of padding among all the input tokens is 17% and 48%, respectively, when the max sequence length is set to 128 and 512. For the Chinese Wikipedia dataset, this percentage is 35% and 79%, respectively, when the max sequence length is 128 and 512. For SQuAD-v1.1 [2], padding accounts for 54% of the total input tokens when the max sequence length is 384. Thus, there is a lot of wasted computation on padding, which significantly increases the training and inference time.

To address this problem, this work proposes a new method that significantly reduces the wasted padding computation in BERT and its training and inference time. The proposed method groups inputs whose lengths are close or identical to each other into a single batch. Unlike the traditional approach which uses the same fixed sequence length for all the batches, the proposed method uses the same max sequence length for all sentences in a single batch, though different batches can have different max sequence lengths. The inputs are sorted by length and grouped into buckets. Each bucket contains inputs with a narrow range of lengths or with the same length. If a bucket only contains inputs with the same length, padding can be completely removed. To train BERT with many input buckets, we explore curriculum learning (CL) [3], [4] to control the training orders of the input buckets. By combining bucketing and CL, the computation time of BERT can be reduced significantly while the performance of the pretrained model can be improved slightly.

II. Removing Padding Computation

The padding removal method proposed in this work uses bucketing. The inputs are sorted by length and put into a number of buckets. Each bucket only holds inputs with a small range of lengths. For example, if the sequence length is 128, the inputs can be grouped into eight buckets which hold inputs with the following length (l) ranges respectively: $1 \leq l \leq 16$, $17 \leq l \leq 32$, $33 \leq l \leq 48$, $49 \leq l \leq 64$, $65 \leq l \leq 80$, $81 \leq l \leq 96$, $97 \leq l \leq 112$, $113 \leq l \leq 128$. The inputs in each bucket are padded to the maximum length of that bucket.

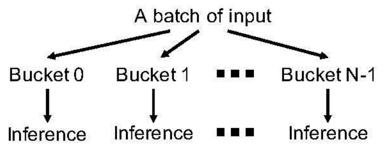

Fig. 1: Inference with padding removal.

A batch can only contain inputs from a single bucket and is not allowed to contain inputs from multiple buckets. During inference, instead of bucketing the entire dataset, only a single batch is bucketed at a time, shown in Figure 1. The inputs in a batch are first grouped into a number of buckets, called sub-batches, and then inference is performed on each sub-batch if that bucket is not empty. Because padding is largely removed in the sub-batches, the total inference time can still be less than a single big batch. This approach removes more padding than the existing optimization approaches [5]–[14], which simply pad a batch of random inputs to the longest one.

III. Curriculum Learning with Padding Removal

In the traditional BERT training, the input samples are shuffled and trained in a random order. This is not possible with the padding removal method. To address this issue, this work explores CL [3], [4] to train the model with the many buckets. Four predefined curricula are designed to train BERT in this work. An intuitive strategy is to train the buckets in a specific order. The orders investigated in this work include the increasing order, the decreasing order, and the random order. A more sophisticated curriculum orders the buckets based on the loss, shown in Algorithm 1. The predefined CL requires the users to define the curriculum beforehand, which may require some experiences or experiments and is not always easy. To avoid designing a curriculum beforehand, automated CL [4] is also explored to control the training of BERT with bucketed inputs. The automated CL explored in this work, shown in Algorithm 2, is based on the idea proposed in [4]. It uses the loss reduction as a reward to instruct the model to select the next syllabus. The automated CL uses the following adversarial bandits algorithm, denoted Exp3.S algorithm [4]:

$$\pi_t(i) = (1-\varepsilon)\frac{e^{w_{t,i}}}{\sum_{j=1}^{M} e^{w_{t,j}}} + \frac{\varepsilon}{M} \tag{1}$$

$$\begin{aligned} w_{t,i} =& \log[(1-t^{-1})\exp\{w_{t-1,i} + \eta\tilde{r}_{t-1,i}\} \\ &+ \frac{t^{-1}}{M-1}\sum_{j\neq i}\exp\{w_{t-1,j} + \eta\tilde{r}_{t-1,j}\}] \end{aligned} \tag{2}$$

$$\tilde{r}_{s,i} = \frac{r_s \mathbb{1}(a_s = i)}{\pi_s(i)} \tag{3}$$

TABLE I: Results for pretraining the BERT base model using the English Wikipedia & BooksCorpus dataset.

	SQuAD-v1 Exact match	SQuAD-v1 F1	MRPC Accuracy	MRPC F1	Phase 1 time	Phase 2 time	Total time	Training time reduction
Baseline	81.22	88.25	83.42	87.84	15h 4m	8h 58m	24h 2m	0%
Algorithm 1 (Increasing order)	81.13	88.31	83.30	88.08	12h 39m	6h 23m	19h 2m	21%
Algorithm 2 (Decreasing order)	81.45	88.28	84.17	88.61	13h 7m	6h 22m	19h 29m	19%
Algorithm 3 (Random order)	81.27	88.48	82.49	87.25	12h 46m	4h 44m	17h 30m	27%
Algorithm 4 (Loss control)	81.00	88.36	83.30	87.83	11h 5m	5h 4m	16h 9m	33%
Algorithm 5 (Automated CL)	80.59	88.07	83.13	87.56	13h 3m	4h 53m	17h 56m	25%

TABLE II: Results for pretraining the BERT base model using the Chinese Wikipedia dataset.

	XNLI Accuracy	Phase 1 time	Phase 2 time	Total time	Training time reduction
Baseline	0.7325	14h 7m	8h 1m	22h 8m	0%
Algorithm 1 (Increasing order)	0.7457	9h 59m	1h 23m	11h 22m	49%
Algorithm 2 (Decreasing order)	0.7375	10h 18m	2h 2m	12h 20m	44%
Algorithm 3 (Random order)	0.7303	10h 7m	1h 41m	11h 48m	47%
Algorithm 4 (Loss control)	0.7441	9h 16m	1h 51m	11h 7m	50%
Algorithm 5 (Automated CL)	0.7359	10h 26m	1h 52m	12h 18m	44%

TABLE III: Inference speedup relative to the BERT base baseline with fixed sequence length of 384 for SQuAD-v1.1.

Batch size	1024	512	256	128	64	32	16	8
3 buckets	1.97	1.83	1.79	1.69	1.58	1.47	1.37	1.17
6 buckets	2.31	2.11	2.02	1.87	1.72	1.47	1.30	1.01
12 buckets	2.43	2.18	2.05	1.83	1.58	1.25	1.03	0.83
24 buckets	2.42	2.10	1.88	1.53	1.21	0.93	0.76	0.68

$$r_t = \begin{cases} -1 & \text{if } \hat{r}_t < q_t^{lo} \\ 1 & \text{if } \hat{r}_t > q_t^{hi} \\ \dfrac{2(\hat{r}_t - q_t^{lo})}{q_t^{hi} - q_t^{lo}} - 1 & \text{otherwise} \end{cases} \tag{4}$$

$$\mathbb{1}(a_s = i) = \begin{cases} 1 & \text{if } a_s = i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Algorithm 1: Curriculum with Loss Control

```
Divide dataset into M buckets
Set loss target L for each bucket
for $i \leftarrow 1$ to $M$ do
    while $loss > L[i]$ do
        for each iteration do
            Draw batch from buckets[i]
            Train batch
        end
    end
end
```

IV. Results

The evaluation in this work uses the BERT code from the NVIDIA deep learning examples [15]. We modify this code to implement the proposed padding removal method and the CL algorithms. The experiments are performed on an NVIDIA HGX-2 machine with 16 V100 GPUs with 32GB memory for each GPU. Training uses all 16 GPUs. The framework used in the experiments is PyTorch.

The proposed padding removal method dramatically reduces the pretraining time of the BERT model. Table I shows the pretraining time of the BERT base model using the English Wikipedia & BooksCorpus dataset. Algorithm 1 with loss control reduces the training time by 33%, cutting the BERT base model's training time from 24 hours to 16 hours. Table II shows the pretraining time of the BERT base model using the Chinese Wikipedia dataset. Algorithm 1 with loss control reduces the training time by 50%, halving the training time of the BERT base model. The inference speedup is evaluated using the SQuAD-v1.1 task, and the results are shown in Table III. A single batch is further bucketed into a number of sub-batches, 3, 6, 12, and 24, under different batch sizes. The results show that more buckets are better for larger batch sizes. For a maximum sequence length of 384, dividing the batch into six buckets gives the best overall performance. The performance improvement for inference using the padding removal method is dramatic, achieving 2.43x speedup when 12 buckets are used for the batch size of 1024.

Algorithm 2: Automated curriculum

```
Divide dataset into M buckets
Initially: $w_i = 0$ for $i \in [M]$
for $t = 1...T$ do
    $\pi_t(k) = (1-\varepsilon)\frac{e^{w_{t,k}}}{\sum_{j=1}^{M} e^{w_{t,j}}} + \frac{\varepsilon}{M}$
    Draw task index $k$ from $\pi_t$
    Randomly draw training batch $b$ from $D_k$
    Train network $p_\theta$ on $b$
    Compute loss reward $\hat{r}_t$ for $b$
    Map $\hat{r}_t$ to $r \in [-1, 1]$ using Equation 4
    Update $w$ with reward $r_t$ using Equation 2, 3, 5
end
```

To evaluate the quality of the pretrained model on the English dataset, the fine-tunings of the SQuAD-v1.1 task and a GLUE task [16], MRPC, are performed on the pretrained model. Table I shows the accuracy and F1 scores for all the proposed CL algorithms. To evaluate the pretrained model on the Chinese dataset, the fine-tuning of the XNLI task is performed on the pretrained the model. Table II shows the accuracy of XNLI for the proposed CL algorithms.

References

[1] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, 2018.

[2] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[3] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *ICML*, 2009.

[4] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu, "Automated curriculum learning for neural networks," in *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[5] [Online]. Available: https://www.kaggle.com/bminixhofer/speed-up-your-rnn-with-sequence-bucketing

[6] [Online]. Available: https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification/discussion/94779

[7] V. Khomenko, O. Shyshkov, O. Radyvonenko, and K. Bokhan, "Accelerating recurrent neural network training using sequence bucketing and multi-gpu data parallelization," in *2016 IEEE First International Conference on Data Stream Mining Processing (DSMP)*, 2016.

[8] P. Doetsch, P. Golik, and H. Ney, "A comprehensive study of batch construction strategies for recurrent neural networks in mxnet," *CoRR*, 2017.

[9] A. G. Baydin, L. Shao, W. Bhimji, L. Heinrich, L. Meadows, J. Liu, A. Munk, S. Naderiparizi, B. Gram-Hansen, G. Louppe, M. Ma, X. Zhao, P. Torr, V. Lee, K. Cranmer, Prabhat, and F. Wood, "Etalumis: Bringing probabilistic programming to scientific simulators at scale," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019.

[10] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *ICLR*, 2020.

[11] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *International Conference on Learning Representations*, 2020.

[12] K. Clark, M.-T. Luong, V. Q. Le, and D. C. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," *International Conference on Learning Representations*, 2020.

[13] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *International Conference on Learning Representations*, 2020.

[14] L. Gong, D. He, Z. Li, T. Qin, L. Wang, and T.-Y. Liu, "Efficient training of bert by progressively stacking," in *International Conference on Machine Learning*, 2019.

[15] [Online]. Available: https://github.com/NVIDIA/DeepLearningExamples

[16] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *International Conference on Learning Representations*, 2019.