

# An Automatic and Efficient BERT Pruning for Edge AI Systems

Shaoyi Huang<sup>[1]</sup>, Ning Liu<sup>[2]</sup>, Yueying Liang<sup>[1]</sup>, Hongwu Peng<sup>[1]</sup>,  
Hongjia Li<sup>[2]</sup>, Dongkuan Xu<sup>[3]</sup>, Mimi Xie<sup>[4]</sup>, Caiwen Ding<sup>[1]</sup>

<sup>[1]</sup>University of Connecticut, <sup>[2]</sup>Northeastern University,  
<sup>[3]</sup>The Pennsylvania State University, <sup>[4]</sup>University of Texas at San Antonio

**Abstract**—With the yearning for deep learning democratization, there are increasing demands to implement Transformer-based natural language processing (NLP) models on resource-constrained devices for low-latency and high accuracy. Existing BERT pruning methods require domain experts to heuristically handcraft hyperparameters to strike a balance among model size, latency, and accuracy. In this work, we propose AE-BERT, an automatic and efficient BERT pruning framework with efficient evaluation to select a “good” sub-network candidate (with high accuracy) given the overall pruning ratio constraints. Our proposed method requires no human experts experience and achieves a better accuracy performance on many NLP tasks. Our experimental results on General Language Understanding Evaluation (GLUE) benchmark show that AE-BERT outperforms the state-of-the-art (SOTA) hand-crafted pruning methods on BERT<sub>BASE</sub>. On QNLI and RTE, we obtain 75% and 42.8% more overall pruning ratio while achieving higher accuracy. On MRPC, we obtain a 4.6 higher score than the SOTA at the same overall pruning ratio of 0.5. On STS-B, we can achieve a 40% higher pruning ratio with a very small loss in Spearman correlation compared to SOTA hand-crafted pruning methods. Experimental results also show that after model compression, the inference time of a single BERT<sub>BASE</sub> encoder on Xilinx Alveo U200 FPGA board has a  $1.83\times$  speedup compared to Intel(R) Xeon(R) Gold 5218 (2.30GHz) CPU, which shows the reasonableness of deploying the proposed method generated subnets of BERT<sub>BASE</sub> model on computation restricted devices.

**Index Terms**—Transformer, deep learning, pruning, acceleration

## I. INTRODUCTION

The Transformer-based models have witnessed eye-catching success in various fields, especially in natural language processing (NLP), such as sentiment classification [1], question answering [2], text summarization [3], and language modeling [4]. As one of the representative Transformer models, BERT [5], showed significant improvement on the General Language Understanding Evaluation (GLUE) benchmark, a well-known collection of tasks for analyzing natural language understanding systems [6].

With the yearning for deep learning democratization [7], there are increasing demands to implement BERT on resource-constrained devices with low latency and high accuracy. Several works were proposed to find a sub-network on BERT<sub>BASE</sub> model with desirable pruning ratio and accuracy [8]–[10]. However, these methods require domain experts to heuristically handcraft hyperparameters to strike a balance among

model size, latency, and accuracy, and therefore are often time-consuming and fail to achieve a globally optimal solution [11].

On the other hand, in computer vision, automatic weight pruning has been developed [12], [13] to identify the sub-networks for deep neural networks to achieve similarity as human experts. However, existing works are empirically costly. For each time of sampling sub-networks, it takes several training epochs to recover the accuracy of the sampled sub-networks from estimating the performance of the sub-networks roughly. Nevertheless, the large amount of sampled sub-networks may take more than thousands of training epochs during the sampling and evaluation phase [12], [13].

In addition, the syntax and semantics information of Transformer in the language/text-domain is more sensitive than computer vision tasks. As a result, it necessities an investigation on the automatic weight pruning on state-of-the-art (SOTA) Transformer-based pre-trained language models.

To this end, we develop an automatic pruning framework on the BERT model for various NLP tasks. The proposed framework has three stages, i.e., sample the sub-networks; evaluate the sampled sub-networks without fine-tuning; fine-tune the winning sub-network from the former stage.

Our contributions are: (i) We develop AE-BERT, an auto weight pruning framework, to identify a “good” sub-net candidate (with high accuracy) without any expert experience given the overall pruning ratio constraints. (ii) We eliminate the need for thousands of training epochs on sub-networks sampling and evaluation on existing auto weight pruning methods [12], [13] while achieving a better accuracy performance. Experiments show that AE-BERT outperforms the SOTA hand-crafted pruning methods on BERT. On QNLI and RTE, we obtain 75% and 42.8% more overall pruning ratio while achieving higher accuracy. On MRPC, we obtain a 4.6 higher score than the SOTA at the same overall pruning ratio of 0.5. On STS-B, we can achieve a 40% higher pruning ratio with a very small loss in Spearman correlation compared to SOTA hand-crafted pruning methods [14].

## II. RELATED WORK

**Transformer-based Models.** Transformer-based models have great advantages of achieving leading results on major Natural Language Processing (NLP) tasks (i.e., question answering [15], machine translation [16], speech recognition [17]). Recently, ViT [18] and iGPT [19] have compet-

itive performance on Computer Vision related tasks with the state-of-the-art. The success of the Transformer-based models mainly benefits from the multi-head self-attention mechanism which computes the representation of a sequence by relating different positions of it multiple times in parallel [20]. Despite the success of Transformer-based models, their gigantic model size (for example, BERT<sub>BASE</sub> has 110M parameters, BERT<sub>LARGE</sub> has 340M parameters) results in massive computations and thus high latency which hurdles the deployment of them on space-intensive edge devices.

**BERT Pruning.** To alleviate the conflicts between model performance and model size and thus achieve model inference time speed up and space-saving, weight pruning techniques have been applied to the NLP field. For instance, irregular magnitude weight pruning (IMWP) has been evaluated on BERT, where 30%-40% weights with a magnitude close to zero are set to be zero [21]. Irregular reweighted proximal pruning (IRPP) [22] adopts iteratively reweighted  $l_1$  minimization with the proximal algorithm and achieves 59.3% more overall pruning ratio than irregular magnitude weight pruning without accuracy loss. [23] investigates the model general redundancy and task-specific redundancy on BERT and XLNet [24]. The Lottery Ticket Hypothesis (proposed by Frankle et al. [25] in computer vision), showing that a sub-network of the randomly-initialized network can replace the original network with the same performance, recently has been investigated on BERT [8]–[10] and achieves high sparsity with slight accuracy degradation. [26]–[30] explores pruning methods of transformer models to reduce the gigantic model size and increase the inference speed on edge devices.

**Knowledge Distillation.** Knowledge distillation (KD) is widely used to compress Transformer-based models while maintaining accuracy. As one of the most popular model compressing methods, the core idea of KD is transferring the knowledge from a large model (teacher) to the compressed model with fewer parameters (student). [31]–[33] utilized KD to transfer knowledge from full BERT<sub>BASE</sub> to shallow subnets with less number of encoder layers. [34], [35] adopted KD to transfer knowledge from full dense BERT<sub>BASE</sub> to the sparse subnets with the same number of encoder layers. Layer-wise knowledge distillation was shown to be effective in reducing the risk of overfitting and increasing network accuracy in [35].

### III. THE PROPOSED FRAMEWORK

#### A. Problem Formulation

Consider an  $N$ -layer model, where the weight and bias of  $i^{\text{th}}$  layer are denoted by  $\mathbf{W}_i, \mathbf{b}_i$  respectively. Let us denote the loss function of the  $N$ -layer model as  $f(\{\mathbf{W}_i\}_{i=1}^N, \{\mathbf{b}_i\}_{i=1}^N)$ . For the  $N$ -layer model, the pruning ratio of the  $i^{\text{th}}$  layer is controlled by  $\beta_i$  (for dense weight,  $\beta_i = 0$ ). Given a specific overall pruning ratio target, we randomly generate  $S$  pruning strategies containing varying pruning ratio strategies:  $\mathbf{B}_j = (\beta_0^j, \beta_1^j, \beta_2^j, \dots, \beta_N^j)$ ,  $j = 1, \dots, S$ . Our goal is to find the optimal pruned sub-network such that the loss function is minimized and the performance of the model achieves the best. Thus our problem is formulated as:

---

#### Algorithm 1: AE-BERT

---

**Input:** Model  $\mathbf{M}$  with a collection of  $N$  weights  $(\mathbf{W}_1, \mathbf{W}_2, \dots, \mathbf{W}_k)$ , overall pruning ratio  $p$ , number of candidates  $n$ ,  $best\_accuracy = 0$ ,  $best\_candidates = \text{none}$   
**Output:** the fine-tuned best candidate  $\mathbf{M}'$   
**Stage I:** Randomly generate  $n$  pruning strategies,  $\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_n$ , and each strategy resulting in an overall pruning ratio of  $p$   
**for**  $i = 1$  to  $n$  **do**  
    **for**  $j = 1$  to  $N$  **do**  
        **Stage I:**  $\mathbf{W}_{j'} = \text{magnitude pruning}(\mathbf{W}_j)$   
    **end for**  
     $\mathbf{M}_i = \text{model } \mathbf{M} \text{ with weights } (\mathbf{W}_{1'}, \mathbf{W}_{2'}, \dots, \mathbf{W}_{k'})$   
    **Stage II:**  $eval\_result = \text{evaluate subnet } \mathbf{M}_i$   
    **if**  $eval\_result \geq best\_accuracy$  **then**  
         $best\_candidate = \mathbf{M}_i$   
    **end if**  
**end for**  
**Stage III:**  $\mathbf{M}' = \text{fine-tuning}(best\_candidates)$

---

$$\mathbf{B}^* = \underset{\{\mathbf{W}_i\}, \{\mathbf{b}_i\}, \{\mathbf{B}_j\}}{\operatorname{argmin}} f(\{\mathbf{W}_i\}_{i=1}^N, \{\mathbf{b}_i\}_{i=1}^N, \{\mathbf{B}_j\}), 1 \leq j \leq S \quad (1)$$

where  $\mathbf{B}_j$  is the  $j^{\text{th}}$  pruning ratios combination or the  $j^{\text{th}}$  candidate and  $\mathbf{B}^*$  is the optimal one among the  $S$  generated strategies. The overall pruning ratio of the model forms the constraints for full model pruning. Under such constraints, we generate  $S$  pruning strategies which satisfy the pruning requirements or constraints and form a searching space. To find a "good" candidate (with high accuracy) when the given constraints of the overall pruning ratio, the searching space may be huge. The existing methods in [12], [13] execute several epochs of fine-tuning each candidate before selecting the best candidate or pruning strategy. As a result, an ample searching space forms a heavy workload in this step. Based on this, we propose an automatic BERT compression method which is fine-tuning free between the steps of pruning and the best candidate selection.

#### B. AE-BERT

To efficiently identify a good sub-network within the original network, we propose the efficient sub-net evaluation algorithm shown in Figure 1 with three stages, i.e., (i) Generate pruning strategies satisfying the constraints and apply the strategies to the original full model. (II) Evaluate the pruned sub-networks without fine-tuning. (III) Fine-tune the highest sub-net evaluation score sub-network.

In stage I, we generate  $n$  pruning strategies, and each strategy will result in the same overall pruning ratio of  $p$ . We adopt the common pruning method, which eliminates the least absolute value of weights [36] using the pruning strategies to the original dense model  $\mathbf{M}$  and produce  $n$  pruned candidates  $\mathbf{M}_1, \mathbf{M}_2, \dots, \mathbf{M}_n$ , as shown in Algorithm 1. In stage II, we evaluate the  $n$  candidates on the training set, and the one with the highest sub-net evaluation value will be the final  $best\_candidate$  after all the iterations. In stage III, we fine-tune the  $best\_candidate$  from the former stage and obtain the final "good" sub-network.

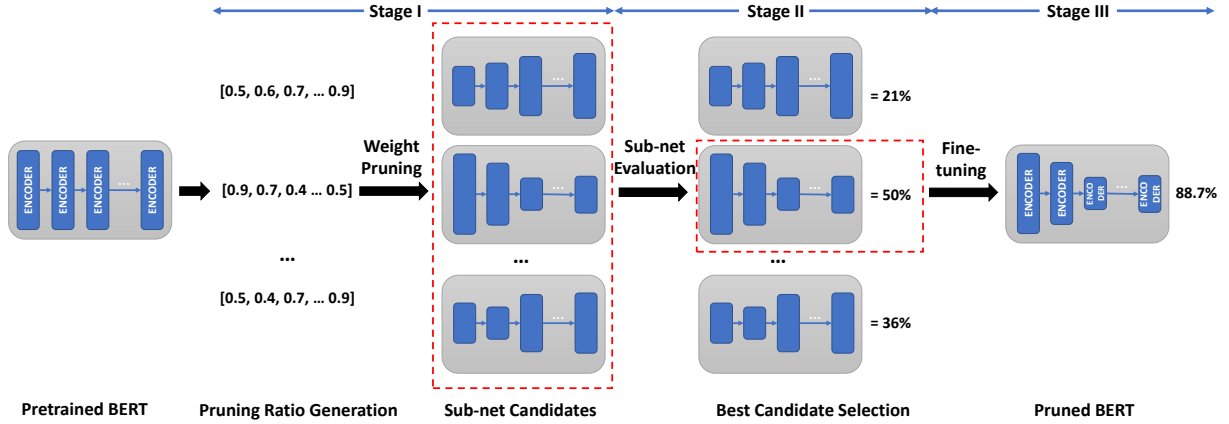


Fig. 1: The three-stage AE-BERT framework.

For stage II, one mainstream of evaluation is fine-tuning the candidate sub-networks with a small amount of training epochs [12], [13], [28], [30]. However, even a small amount of training epoch can be costly since the total number of sampled candidates are extensive because the outer for-loop will amplify the time. To mitigate this cost, we empirically study the correlation between pruning without fine-tuning manner and pruning with fine-tuning manner. We discuss the observation that there exists a high correlation between these two manners in section IV-C. This high correlation helps efficiently identify the high-quality sub-networks without any training epochs.

#### IV. EXPERIMENTS

##### A. Datasets

We adopt GLUE benchmark [6] as our dataset, which consists of three tasks (single sentence tasks, similarity matching tasks, and natural language inference tasks) according to the purpose of tasks and difficulty level of datasets. We test our method on two paraphrase similarity matching tasks (MRPC [37], STS-B [38]); and two natural language inference tasks: QNLI [6], RTE [6].

##### B. Experiment Setup

**Baseline Models.** The baseline model is our own fine-tuned unpruned BERT<sub>BASE</sub> [5]. We report our results (from the official bert-base-uncased) as Full BERT<sub>BASE</sub>. We use the Huggingface Transformer toolkit [39] to conduct our experiment. There are 12 layers ( $L=12$ ; hidden size  $H=768$ ; self-attention heads  $A=12$ ), with 110 million parameters. Moreover, we compare our proposed framework with two hand-crafted model compression methods on BERT, BERT<sub>BASE</sub> irregular [40] and BERT<sub>BASE</sub> LTH [14].

**Metrics.** We apply the same metrics of the tasks as the GLUE paper [6], i.e., accuracy scores are reported for RTE and QNLI; F1 scores are reported for MRPC; Spearman correlations are reported for STS-B.

**Platforms.** We use Python 3.6.10 with PyTorch 1.4.0 and CUDA 11.1 on Quadro RTX6000 GPU and Intel(R) Xeon(R)

Gold 5128 (2.30GHz) CPU for software experiments. We use the Xilinx Alveo U200 board (FPGA) as the hardware platform, which has 4,320 of 18k BRAM, 6,840 DSPs, and 1,882.2k logic cells (LUT) and benefits from the high-level synthesis tool (C/C++). To evaluate the hardware inference performance, we further compare it with Intel i5-5257U (2.7 GHz) CPU in terms of latency and throughput (frame/sequence per second, i.e., FPS).

**Hyperparameter Selection.** In the fine-tuning stage, we run four epochs for each of the tasks with a batch size of 32 and a learning rate of  $3e^{-5}$ – $5e^{-5}$  (learning rate that gives the best performance is selected for each task). For the generation of pruning ratio candidates, we randomly generate a list of pruning ratios that give the same overall pruning ratio. For the pruned weight selection, we consider the pruning for encoders. For QNLI, STS-B, MRPC, RTE, we target an overall pruning ratio of 0.7, 0.7, 0.5, and 0.6 as our constraints, respectively.

**Hardware Setting.** To improve the performance and better utilize the resource during hardware implementation, we adopt operation scheduling methods [41] for hardware design to allocate an appropriate amount of resources for functions. The optimization problem can be described as:

$$\begin{aligned} \min_{\{W_n\}, \{b_n\}} \quad & \min(O_1, O_2, \dots, O_k) \\ \text{subject to} \quad & R_t \geq M \sum_{i=1}^k R_i + R_m \end{aligned} \quad (2)$$

where we denote  $O_1, O_2, \dots, O_k$  as the latency of each individual operation,  $R_t = [R_{DSP}, R_{FF}, R_{LUT}, R_{BRAM}]$  represents the total available resources on an FPGA chip,  $R_i$  stands for the resource that is utilized by each function with an encoder/decoder, and  $R_m$  is denoted as the resource used by the DDR controller, PCIe controller, or other types of miscellaneous function within the FPGA system. For the first step, we design hardware without any parallelism. Then we optimize the performance by iteratively adding hardware parallelism in the slowest function or loop and checking the resource constraint until it is satisfied during the process. After optimizing the slowest function or loop, we move to the next slowest function or loop and optimize it with the same

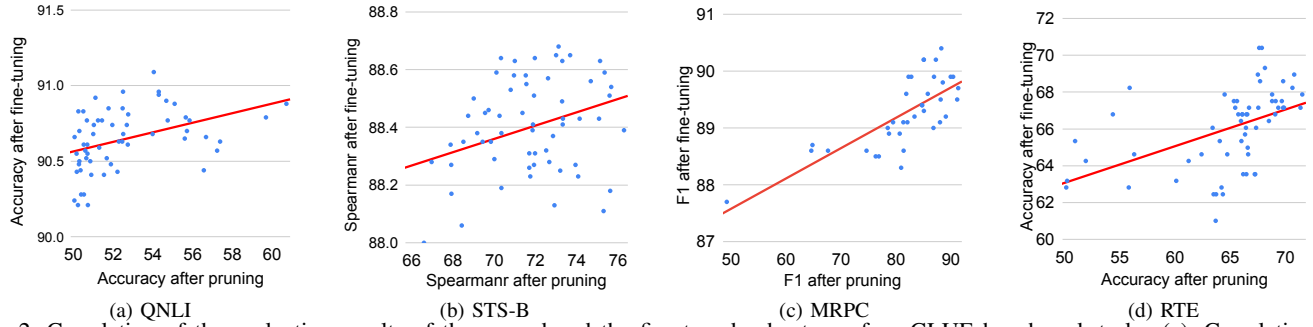


Fig. 2: Correlation of the evaluation results of the pruned and the fine-tuned subnets on four GLUE benchmark tasks. (a). Correlation of after pruning and after fine-tuning accuracy on QNLI; (b). Correlation of after pruning and after fine-tuning Spearman correlation on STS-B; (c). Correlation of after pruning and after fine-tuning F1 score on MRPC; (d) Correlation of after pruning and after fine-tuning accuracy on RTE. The red line in each sub-figure is the trend line.

procedure until the overall hardware resources and latency are optimal. The hardware scheduling results of each operation (e.g., Matrix Multiplication (MM), dot product attention, add normalization) in encoder and decoder are shown in Table III.

### C. Results

We demonstrate the correlation between pruning without fine-tuning manner and pruning with fine-tuning manner on selected GLUE benchmark tasks in Figure 2. Each blue dot in all sub-figures represents a candidate, while the values on the x-axis and y-axis stand for the accuracy/metrics values after pruning (pruning without fine-tuning) and the value after fine-tuning (pruning with fine-tuning), respectively. The red line in each subplot is the auto-fitting trend line. The positive slope of trend lines in four sub-figures indicate a linear mapping relation between the manner of pruning without fine-tuning and the manner of pruning with fine-tuning, which justifies our observation.

TABLE I: Comparison of AE-BERT with the state-of-the-arts.

Models	QNLI	STS-B	MRPC	RTE
<b>Full BERT<sub>BASE</sub></b>	91.4	89.1	89.4	71.5
<b>BERT<sub>BASE</sub> irregular [40]</b>	87.8	86.7	-	63.5
<b>Pruning ratio</b>	0.4	0.6	-	0.42
<b>BERT<sub>BASE</sub> LTH [14]</b>	88.9	88.2	84.9	66
<b>Pruning ratio</b>	0.7	0.5	0.5	0.6
<b>AE-BERT (ours-Top 1)</b>	88.7	86.1	89.5	69.7
<b>Pruning ratio</b>	0.7	0.7	0.5	0.6
<b>AE-BERT (ours-Top 10)</b>	89.7	86.7	89.6	70.4
<b>Pruning ratio</b>	0.7	0.7	0.5	0.6

The experimental results of are shown in Table I on GLUE benchmark tasks. We can achieve the aimed prune ratio with limited accuracy loss by applying our proposed AE-BERT framework. To be more specific, for the paraphrase similarity matching task on MRPC both the top 1 and top 10 F1 scores (89.5 and 89.6) of the sparse model at a pruning ratio of 0.5 even exceed our own fine-tuned baseline (89.4). Furthermore, for STS-B, the Spearman correlation loss for the top 1 result is 3.0, while the top 10 is 2.4 at a pruning ratio of 0.7. On the natural language inference tasks (QNLI and RTE), our

proposed framework can limit the accuracy loss up to 1.7 for the top 10 results.

Comparison results with BERT compression works [40] and [14] are shown in Table I, which indicating that our method has advantage on these tasks. (Since there is no result of irregular pruning on MRPC in [40], we do not compare the result of it.) Our AE-BERT framework exhibits the highest accuracy performance among various NLP tasks without the need for domain experts’ efforts. Under a more overall pruning ratio, our proposed framework achieves higher accuracy than the hand-crafted model compression methods for BERT. On QNLI and RTE, 75% and 42.8% more overall pruning ratio is obtained while higher accuracy is maintained. On MRPC, we obtain a 4.6 higher score at the same overall pruning ratio of 0.5. On STS-B, we can achieve a 40% higher pruning ratio with only up to 2.38% accuracy loss in Spearman correlation.

A comparison of AE-BERT and sparse BERT<sub>BASE</sub> at higher sparsity (0.9) is shown in Table II. The difference between the two compression methods is that the sparsity of different encoders of the former may be different from each other, while the latter is the same. The experimental results show that AE-BERT can find better subnets than irregular pruned BERT<sub>BASE</sub> on QNLI, STS-B, MRPC, and RTE. We also investigate the effects of knowledge distillation on the finetuning stage where we use full BERT<sub>BASE</sub> and the best candidates as teacher and student, respectively. We use layer-wise distillation loss instead of hard logits loss to update weights. The experimental results show that KD could further improve the performance of the best candidates. On QNLI, STS-B, MRPC, there are 3%, 5.8%, 1.9% performance gains using KD loss, respectively.

TABLE II: Comparison of AE-BERT (the sparsity of different encoders may vary from each other) with sparse BERT<sub>BASE</sub> (the sparsity of different encoders are the same) at same overall sparsity 0.9.

Models	QNLI	STS-B	MRPC	RTE
<b>Full BERT<sub>BASE</sub></b>	91.4	89.1	89.4	71.5
<b>Sparse BERT<sub>BASE</sub></b>	64.2	29.5	79.1	55.6
<b>AE-BERT</b>	72.3	41.2	80.8	56.8
<b>AE-BERT + KD</b>	74.5	43.6	82.3	57.1

**Hardware Evaluation and Analysis.** DSP, FF, and LUT are the three most important metrics while evaluating performance on hardware. In our experiments, we observe from the table that a single encoder occupies 2050, 558.4k, and 692.8k resources for DSP, FF, and LUT, respectively, as shown in Table III. As for the total hardware resources utilization, DSP, FF, and LUT require up to 6840, 2364.5k, and 1882.2k, and the percentages of the resource utilization on FPGA are 30.1%, 23.6% and 36.8%. The latency of an encoder is reduced to 17.23 ms after optimization, which satisfies the real-time constraints, thus making it possible for various NLP tasks while operating on resource-constrained devices.

TABLE III: Encoder implementation (sparsity = 0.9, batch size = 1)

	DSP	FF	LUT	Latency
<b>Total hardware resources</b>	6,840	2,364.5k	1,882.2k	N/A
<b>Encoder</b>	DSP	FF	LUT	Latency
Sparse MM accelerator 1	662	300.8k	301.6k	5.760 ms
Dot product attention $\times$ 12	584	119.2k	203.4k	2.770 ms
Sparse MM accelerator 2	336	46.6k	62.4k	3.520 ms
Add normalization 1	124	35.8k	36.6k	1.605 ms
Sparse MM accelerator 3	344	56k	52.2k	1.965 ms
Add normalization 2	124	35.8k	36.6k	1.605 ms
<b>Resources for 1 encoder</b>	2050	558.4k	692.8k	17.23 ms
<b>Percentage</b>	30.1%	23.6%	36.8%	N/A

**Cross Platform Comparison.** As shown in Table IV, we compare the latency and throughput of a single encoder with a batch size of 1 on Intel(R) Xeon(R) Gold 5218 (2.30GHz) CPU and Xilinx Alveo U200 FPGA board, respectively. The latency/throughput of a single encoder for the CPU and the FPGA are 31.50/31.75, 17.23/58.04, individually. As a result, a  $1.83 \times$  inference speedup on a single encoder compared to the CPU. In conclusion, the proposed pruning method can generate well-performed subnets, and the hardware design on FPGA accelerates the inference time of BERT<sub>BASE</sub>.

TABLE IV: Comparison of inference performance of a single encoder on different hardware devices

Hardware	Latency (ms)	Throughput (FPS)
<b>Intel(R) Xeon(R) Gold 5218 (2.30GHz) CPU</b>	31.50	31.75
<b>Xilinx Alveo U200 FPGA board</b>	17.23	58.04

## V. CONCLUSION

In this work, we propose a three-stage AE-BERT method to achieve automatic pruning on BERT<sub>BASE</sub> without the experience of human experts. In stage I, we sample the sub-networks according to the strategy generated under pruning constraints. In stage II, we evaluate the sampled sub-networks and select the candidate with the best performance on GLUE. Then, we consider this sub-network with the highest potential to achieve the highest performance. In addition, we achieve fine-tuning free in this stage. We fine-tune the winning candidate from the previous stage and obtain the final sparse model in stage III. Experiments on four GLUE benchmark tasks show that our proposed method outperforms the SOTA hand-crafted pruning methods. Experiments on hardware platforms show that there

is a  $1.83 \times$  inference time speedup of a single encoder on FPGA compared to CPU.

## REFERENCES

- [1] Q. Xie *et al.*, “Unsupervised data augmentation for consistency training,” *Advances in Neural Information Processing Systems*, vol. 33, 2020.
- [2] I. Beltagy *et al.*, “Longformer: The long-document transformer,” *arXiv preprint arXiv:2004.05150*, 2020.
- [3] X. Zhang *et al.*, “Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 5059–5069.
- [4] N. Kitaev *et al.*, “Reformer: The efficient transformer,” *arXiv preprint arXiv:2001.04451*, 2020.
- [5] J. Devlin *et al.*, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in *NAACL-HLT (1)*, 2019.
- [6] A. Wang *et al.*, “Glue: A multi-task benchmark and analysis platform for natural language understanding,” *arXiv preprint arXiv:1804.07461*, 2018.
- [7] C. Garvey, “A framework for evaluating barriers to the democratization of artificial intelligence,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [8] T. Chen, J. Frankle, S. Chang, S. Liu, Y. Zhang, Z. Wang, and M. Carbin, “The lottery ticket hypothesis for pre-trained bert networks,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 15 834–15 846.
- [9] H. You, C. Li, P. Xu, Y. Fu, Y. Wang, X. Chen, R. G. Baraniuk, Z. Wang, and Y. Lin, “Drawing early-bird tickets: Toward more efficient training of deep networks,” in *International Conference on Learning Representations*, 2019.
- [10] S. Prasanna, A. Rogers, and A. Rumshisky, “When bert plays the lottery, all tickets are winning,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2020, pp. 3208–3229.
- [11] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Gutttag, “What is the state of neural network pruning?” *arXiv preprint arXiv:2003.03033*, 2020.
- [12] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, “Amc: Automl for model compression and acceleration on mobile devices,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 784–800.
- [13] N. Liu, X. Ma, Z. Xu, Y. Wang, J. Tang, and J. Ye, “Autocompress: An automatic dnn structured pruning framework for ultra-high compression rates,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4876–4883.
- [14] T. Chen, J. Frankle, S. Chang, S. Liu, Y. Zhang, Z. Wang, and M. Carbin, “The lottery ticket hypothesis for pre-trained bert networks,” *arXiv preprint arXiv:2007.12223*, 2020.
- [15] B. van Aken, B. Winter, A. Löser, and F. A. Gers, “How does bert answer questions? a layer-wise analysis of transformer representations,” in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. New York, NY, USA: ACM, 2019, pp. 1823–1832.
- [16] T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015, pp. 1412–1421.
- [17] L. Dong, S. Xu, and B. Xu, “Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition,” in *International Conference on Acoustics, Speech and Signal Processing*. Calgary, Alberta, Canada: IEEE, 2018, pp. 5884–5888.
- [18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” 2020.
- [19] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever, “Generative pretraining from pixels,” in *International Conference on Machine Learning*. Vienna, Austria: PMLR, 2020, pp. 1691–1703.
- [20] A. Vaswani *et al.*, “Attention is all you need,” in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

- [21] M. A. Gordon, K. Duh, and N. Andrews, "Compressing bert: Studying the effects of weight pruning on transfer learning," *arXiv preprint arXiv:2002.08307*, 2020.
- [22] F.-M. Guo, S. Liu, F. S. Mungall, X. Lin, and Y. Wang, "Reweighted proximal pruning for large-scale language representation," *arXiv preprint arXiv:1909.12486*, 2019.
- [23] F. Dalvi, H. Sajjad, N. Durrani, and Y. Belinkov, "Analyzing redundancy in pretrained transformer models," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 4908–4926.
- [24] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.
- [25] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.
- [26] P. Qi, Y. Song, H. Peng, S. Huang, Q. Zhuge, and E. H.-M. Sha, "Accommodating transformer onto fpga: Coupling the balanced model compression and fpga-implementation optimization," in *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, 2021, pp. 163–168.
- [27] P. Qi, E. H.-M. Sha, Q. Zhuge, H. Peng, S. Huang, Z. Kong, Y. Song, and B. Li, "Accelerating framework of transformer by hardware design and model compression co-optimization," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–9.
- [28] H. Peng, S. Huang, T. Geng, A. Li, W. Jiang, H. Liu, S. Wang, and C. Ding, "Accelerating transformer-based deep learning models on fpgas using column balanced block pruning," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2021, pp. 142–148.
- [29] S. Huang, S. Chen, H. Peng, D. Manu, Z. Kong, G. Yuan, L. Yang, S. Wang, H. Liu, and C. Ding, "Hmc-tran: A tensor-core inspired hierarchical model compression for transformer-based dnns on gpu," in *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, 2021, pp. 169–174.
- [30] S. Chen, S. Huang, S. Pandey, B. Li, G. R. Gao, L. Zheng, C. Ding, and H. Liu, "Et: re-thinking self-attention for transformer models on gpus," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–18.
- [31] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.
- [32] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "Tinybert: Distilling bert for natural language understanding," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020, pp. 4163–4174.
- [33] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," *Advances in Neural Information Processing Systems (NIPS)*, 2020.
- [34] D. Xu, I. E. Yen, J. Zhao, and Z. Xiao, "Rethinking network pruning—under the pre-train and fine-tune paradigm," 2021.
- [35] S. Huang, D. Xu, I. E. Yen, S.-e. Chang, B. Li, S. Chen, M. Xie, H. Liu, and C. Ding, "Sparse progressive distillation: Resolving overfitting under pretrain-and-finetune paradigm," *arXiv preprint arXiv:2110.08190*, 2021.
- [36] S. Han *et al.*, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143.
- [37] B. Dolan and C. Brockett, "Automatically constructing a corpus of sentential paraphrases," in *Third International Workshop on Paraphrasing (IWP2005)*. Asia Federation of Natural Language Processing, January 2005. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/automatically-constructing-a-corpus-of-sentential-paraphrases/>
- [38] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 1–14. [Online]. Available: <https://www.aclweb.org/anthology/S17-2001>
- [39] T. Wolf *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *ArXiv, abs/1910.03771*, 2019.
- [40] B. Li *et al.*, "Efficient transformer-based large scale language representations using hardware-friendly block structured pruning," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.
- [41] S. Wang, Z. Li, C. Ding, B. Yuan, Q. Qiu, Y. Wang, and Y. Liang, "C-lstm: Enabling efficient lstm using structured compression techniques on fpgas," in *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2018, pp. 11–20.