Adder Encoder for Pre-trained Language Model

Jianbang Ding[1]*, **Suiyun Zhang**[1], **Linlin Li**[2]
[1]Huawei Technologies Co., Ltd
[2]Huawei Noah's Ark Lab
{dingjianbang1, zhangsuiyun, lynn.lilinlin}@huawei.com

Abstract

BERT, a pre-trained language model entirely based on attention, has proven to be highly performant for many natural language understanding tasks. However, pre-trained language models (PLMs) are often computationally expensive and can hardly be implemented with limited resources. To reduce energy burden, we introduce adder operations into the Transformer encoder and propose a novel AdderBERT with powerful representation capability. Moreover, we adopt mapping-based distillation to further improve its energy efficiency with an assured performance. Empirical results demonstrate that AddderBERT$_6$ achieves highly competitive performance against that of its teacher BERT$_{\text{BASE}}$ on the GLUE benchmark while obtaining a 4.9x reduction in energy consumption.

1 Introduction

The last five years have seen great success achieved by large-scale pre-trained language models, such as BERT (Devlin et al., 2019), ELECTRA (Clark et al., 2020), and GPT3 (Brown et al., 2020). By modeling long-distance dependencies based on self-attention, they can learn powerful language representations from the unlabeled corpus.

While these models lead to significant improvement on many downstream tasks (eg., the GLUE benchmark (Wang et al., 2019)), the growing computation costs have impaired their deployment, especially on limited-resource devices such as mobile phones, AR glasses, and smartwatch. Since attending to all tokens yields a complexity of $O(n^2)$ with respect to sequence length, prior works aim to investigate efficient Transformers with lower complexity. Kitaev et al. (2020) replaces dot-product attention with one using locality-sensitive hashing. Wang et al. (2020) decomposes the original attention into multiple smaller ones by linear projections. However, they can only solve the problem partway, for the consumption except self-attention has not been changed in the encoder.

Various attempts also focus on model compression techniques, including quantization (Gong et al., 2014), weights pruning (Han et al., 2015), and knowledge distillation (KD) (Romero et al., 2015). As one of the most popular methods, KD aims to transfer knowledge from a large teacher network to a small student network, employed by DistilBERT (Sanh et al., 2019), BERT-PKD (Sun et al., 2019), TinyBERT (Jiao et al., 2020), and FastBERT (Liu et al., 2020). Beyond these methods, Chen et al. (2020) proposed Adder Neural Network (AdderNet), which replaced massive multiplications with cheaper additions to reduce computation costs, and achieved better performance on the ImageNet dataset compared to CNNs. Then researchers attempt to build efficient deep-learning models based on AdderNet for computer vision tasks (Xu et al., 2020; Shu et al., 2021). Inspired by this, it is an interesting idea to investigate the feasibility of replacing multiplications with additions in pre-trained language models like BERT.

In this paper, we first present AdderBERT, a pre-trained model consisting of several adder encoders, in which key modules including multi-head attention and feed-forward network are implemented with cheaper adder operations. As shown in Figure 1, it also has a unique mapping-based distillation that could make it to be more energy-efficient with an assured performance. Finally, we conduct full experi-

*Corresponding author

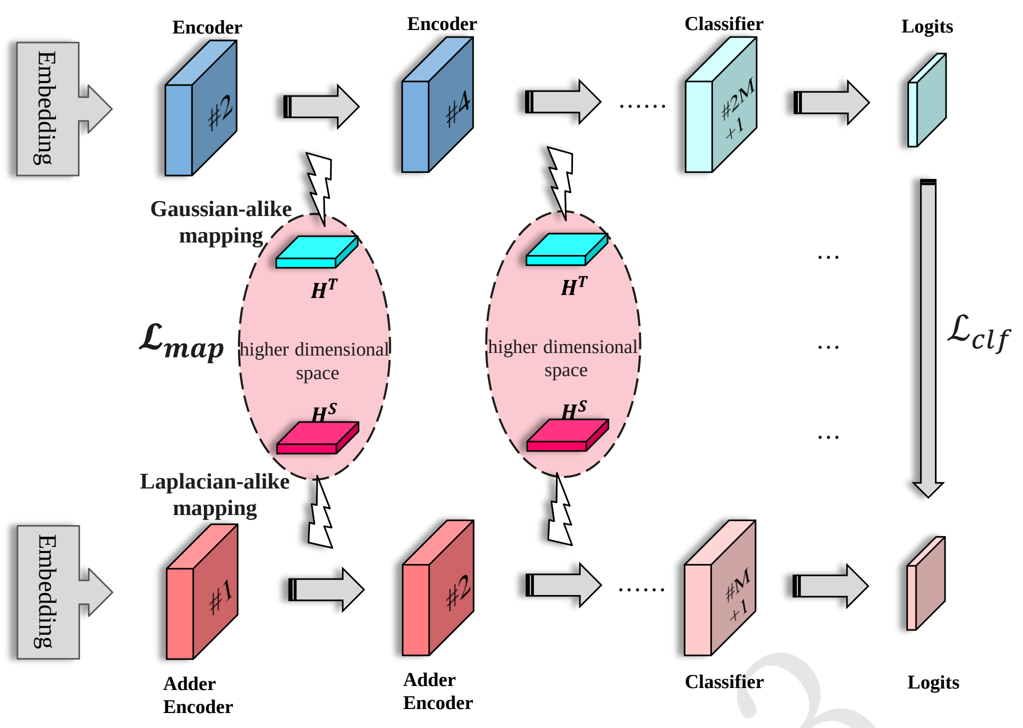

Figure 1: Depiction of AdderBERT learning. AdderBERT implements the encoder block with cheaper adder operations, and it has a unique mapping-based distillation. $\mathbf{H}^S$ and $\mathbf{H}^T$ are the hidden states of the student and teacher networks, respectively. M denotes the number of adder encoders.

ments on the GLUE benchmark. Empirical results demonstrate that our method can achieve comparable performance with the baselines in much lower energy consumption.

The contributions are summarized as follows:

- We propose AdderBERT, which introduces adder operations into the mechanism of self-attention and feed-forward network.
- We adopt a novel mapping-based distillation to encourage that linguistic knowledge can be adequately transferred from the teacher network to AdderBERT.
- Experimental results show that AddderBERT$_6$ can achieve highly competitive performance against that of its teacher BERT$_{\mathrm{BASE}}$ on the GLUE benchmark while obtaining a 4.9x reduction in energy consumption.

2 Preliminary

In this section, we revisit the related works including AdderNet and knowledge distillation. We are motivated by them to design AdderBERT.

2.1 Adder Neural Networks (AdderNet)

Denote the input feature as $\mathbf{X} \in \mathbb{R}^{h \times w \times c_{in}}$, in which h and w are the height and width of the feature map, respectively. Consider a filter $\mathbf{W} \in \mathbb{R}^{d \times d \times c_{in} \times c_{out}}$ in an arbitrary layer of AdderNet, where d is the kernel size, c_{in} and c_{out} are the number of input channels and output channels, respectively. The original adder operation is defined as:

$$\mathbf{Y}(m,n,v) = -\sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{u=1}^{c_{in}} \left|\mathbf{X}(m+i,n+j,u) - \mathbf{W}(i,j,u,v)\right|, \tag{1}$$

where $|\cdot|$ is the absolute value function. m and n are the spatial locations of features. v denotes the index of output channels. Given that Equation 1 has been proven to be used to replace the traditional convolution operation, it is an interesting idea to transport this success of CNNs to PTMs.

2.2 Knowledge Distillation (KD)

As one of the most popular compression techniques, KD was used to help a small student network S mimic the behavior of a large teacher network T for better performance. Given f^T and f^S represent the mapping functions of teacher and student networks, respectively. The student network can be optimized with the following objective function:

$$\mathcal{L}_{\text{KD}} = \sum_{x \in \Omega} \mathcal{L}(f^S(x), f^T(x)), \tag{2}$$

where $L(\cdot)$ is an arbitrary loss function, x is the input sequence and Ω denotes the training dataset, $f^S(x)$ and $f^T(x)$ are the outputs of student network and teacher network, respectively. Based on Equation 2, we adopt a unique kernel-based distillation to encourage that linguistic knowledge can be adequately transferred from the teacher network to the student AdderBERT.

3 Method

This section describes our proposed AdderBERT as well as its training method. Concisely, AdderBERT implements the encoder block with cheaper adder operations, and it takes advantage of mapping-based distillation to be better in performance and efficient in energy.

3.1 Adder Encoder

Given that linear transformation is equivalent to 1×1 convolution with fixed input size in mathematical, in this paper, the adder operation can be redefined as:

$$\mathbf{Y}(l, v) = -\sum_{u=1}^{d_{embd}} |\mathbf{X}(l, u) - \mathbf{W}(u, v)| = \mathbf{X} \oplus \mathbf{W}, \tag{3}$$

where l is the sequence length, d_{embd} is the dimension of embedding, and $\oplus$ denotes the adder operation between matrices.

Following the original Transformer (Vaswani et al., 2017), We first consider creating output queries $\mathbf{Q} \in \mathbb{R}^{l \times d_q}$, keys $\mathbf{K} \in \mathbb{R}^{l \times d_k}$, and values $\mathbf{V} \in \mathbb{R}^{l \times d_v}$ by weight matrices $\mathbf{W}_Q \in \mathbb{R}^{d_{embd} \times d_q}$, $\mathbf{W}_K \in \mathbb{R}^{d_{embd} \times d_k}$, $\mathbf{W}_V \in \mathbb{R}^{d_{embd} \times d_v}$ in the projection layer of a single-head self-attention. d_q, d_k, d_v is the dimension of queries, keys, and values, respectively. We employ Equation 3 to measure the ℓ_1-distance between embedding and the weight matrices as:

$$\mathbf{Q} = \mathcal{LN}(\mathbf{X} \oplus \mathbf{W}_Q), \quad \mathbf{K} = \mathcal{LN}(\mathbf{X} \oplus \mathbf{W}_K), \quad \mathbf{V} = \mathcal{LN}(\mathbf{X} \oplus \mathbf{W}_V), \tag{4}$$

where $\mathbf{X} \in \mathbb{R}^{l \times d_{embd}}$ is the input embedding and $\mathcal{LN}(\cdot)$ denotes layer normalization (Ba et al., 2016). Chen et al. (2020) first indicated that the output values of the adder operation should be followed by batch normalization. We also apply layer normalization to stabilize the hidden state dynamics for better learning. Similarly, Equation 3 can be easily modified for a batch matrix-matrix product to realize self-attention:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q} \oplus \mathbf{K}^T}{\sqrt{d_k}}) \oplus \mathbf{V}, \tag{5}$$

where softmax$(\cdot)$ is the normalized exponential function and d_k is used for scaling. The attention matrix is calculated from the similarity of $\mathbf{Q}$ and $\mathbf{K}$ by adder operation and acts as the weighted sum factor to $\mathbf{V}$ to get the final output. Multi-head self-attention concatenated different heads from different representing subspaces as follows:

$$\text{MHA}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathcal{LN}(\text{Concat}(\text{head}_1, ..., \text{head}_n) \oplus \mathbf{W}_O), \tag{6}$$

where head_i is the i-th attention head obtained by Equation 5 and n is the number of heads. Then we use $\mathbf{W}_O$ to realize an adder projection for dimensional transformation followed by layer normalization. Finally, the feed-forward network can also be reformulated as:

$$\text{FFN}(X) = \mathcal{LN}(\text{ReLU}(\mathbf{X} \oplus \mathbf{W}_1) \oplus \mathbf{W}_2). \tag{7}$$

The new FFN consists of two adder linear transformations, one ReLU activation, and one layer normalization.

3.2 Mapping-based Distillation

Since the basic calculation paradigm of AdderBERT is completely different from that of the original BERT, we adopt a novel mapping-based distillation to adequately transport linguistic knowledge from Teacher BERT to student AdderBERT.

Specifically, we distill the output of each encoder block, and the objective function is as follows:

$$\mathcal{L}_{map} = \text{MSE}(\mathbf{H}^S, \mathbf{H}^T), \tag{8}$$

where $\mathbf{H}^S$ and $\mathbf{H}^T$ are the hidden states of the student and teacher networks, respectively. MSE$(\cdot)$ denotes the mean square error loss function. As discussed in AdderNet (Chen et al., 2020), the weight distribution in a well-trained ANN is Laplacian distribution rather than Gaussian distribution. Thus we attempt to map the inputs and weights to a higher dimensional space to minimize the distribution gap between $\mathbf{H}^S$ and $\mathbf{H}^T$.

Given $\{\mathbf{X}^S, \mathbf{W}_1^S, \mathbf{W}_2^S\}$, $\{\mathbf{X}^T, \mathbf{W}_1^T, \mathbf{W}_2^T\}$ are the inputs and weights of the FFN of the student and teacher network, respectively. During the distillation process, we transform the hidden states by feature mapping as follows:

$$\begin{aligned} \mathbf{H}^S &= k_1 < \mathbf{X}^S, \mathbf{W}_1^S, \mathbf{W}_2^S >= e^{-\frac{\mathbf{x}^S \oplus \mathbf{w}_1^S \oplus \mathbf{w}_2^S}{\sigma_s}}, \\ \mathbf{H}^T &= k_2 < \mathbf{X}^T, \mathbf{W}_1^T, \mathbf{W}_2^T >= e^{-\frac{\mathbf{x}^T \mathbf{w}_1^T \mathbf{w}_2^T}{2\sigma_t^2}}, \end{aligned} \tag{9}$$

where σ_s and σ_t are two learnable smoothing factors. $k_1 < \cdot >$ is a designed Laplacian-alike kernel that takes the adder operation of two matrices, while $k_2 < \cdot >$ is a Gaussian-alike kernel. After applying Equation 9, the inputs and weights are mapped into a higher dimensional space, thus we can calculate the hidden states by the new smoothing representation.

We also use the cross-entropy loss for classifier distillation $\mathcal{L}_{clf}$ as in previous work (Hinton et al., 2015). Then the final loss function is defined as:

$$\mathcal{L}_{model} = \alpha \mathcal{L}_{map} + \mathcal{L}_{clf} = \sum_{x \in \Omega} (\sum_{m=1}^{M} \alpha \mathcal{L}_{map}^m(x) + \mathcal{L}_{clf}(x)), \tag{10}$$

where M is the number of encoder blocks of AdderBERT, and m denotes the m-th block. α is the hyper-parameter for seeking the balance between $\mathcal{L}_{map}$ and $\mathcal{L}_{clf}$.

4 Experiment

In this section, we verify the effectiveness of AdderBERT on three tasks with different model settings.

4.1 Datasets

We evaluate our method on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019). For Sentiment classification, we test on CoLA (Warstadt et al., 2019), SST-2 (Socher et al., 2013). For similarity matching, we conduct on QQP [1], MRPC (Dolan and Brockett, 2005), and STS-B (Cer et al., 2017). For language inference, we use MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), WNLI (Levesque et al., 2012) and RTE (Bentivogli et al., 2009).

4.2 AdderBERT Settings

For a fair comparison, We build AdderBERT$_{12}$ with the same configuration as the original BERT$_{\text{BASE}}$ (the number of layers is 12, the hidden size is 768, the feed-forward size is 3072, the number of heads is 12). BERT$_{\text{BASE}}$ uses pre-trained parameters released by Google, and we train AdderBERT$_{12}$ from scratch with the same pre-training settings. We fine-tune them both using the AdaMod (Ding et al., 2019) optimizer for better performance, the learning rate is set to 2e-5, and the batch size is set to 32. We then select the model with the best accuracy in 3 epochs.

We also use the fine-tuned BERT$_{\text{BASE}}$ as the teacher model and use 6 and 3 layers of AdderBERT as the student models (i.e. AdderBERT$_6$ and AdderBERT$_3$). The student models learn from every 2 and 4 layers of the teacher model, respectively. We increase the learning rate to 5e-5 and distill them for 5 epochs. All the experiments are conducted on NVIDIA Tesla-V100 GPUs.

Given that hyperparameters can exert a great impact on the ultimate result, we report the full details about them. We follow the grid search method until the best-performing parameters are at one of the middle points in the grid. For fine-tuning, we tune over hyperparameters to work well across all tasks about batch size: {8, 16, 32, 64}, the initial learning rate of AdaMod:{2e-5, 3e-5, 5e-5, e-4}, and the number of epochs:{2, 3, 4, 5}.

4.3 Baselines

We compare AdderBERT against two strong baselines as follows:

- **BERT** The 12-layer BERT$_{\text{BASE}}$ model, which was pre-trained on Wiki corpus and released by Google (Devlin et al., 2019).
- **DistilBERT** The most famous distillation version of BERT with 6 layers, which was released by Huggingface (Sanh et al., 2019). In addition, we use the same method to distill the DistilBERT with 3 layers.

4.4 Experiments on GLUE

We submitted our model predictions to the official GLUE evaluation server to get results on the test data, as reported in Table 1. Note that values in both models are 32-bit floating numbers and the energy consumptions for a 32-bit multiplication and addition are 3.7pJ and 0.9pJ, respectively (Dally, 2015). The original BERT$_{\text{BASE}}$ achieves a 79.7 score on average with 11.27B multiplications and 11.27B additions, and AdderBERT$_{12}$ achieves a 79.0 score with 0.31B multiplications and 22.23B additions. By replacing massive multiplications with additions, our proposed model obtains about a 2.5x reduction in energy consumption from 51.8BpJ to 21.1BpJ at the cost of a little performance loss (0.7 drops relative to BERT$_{\text{BASE}}$ on the average score). This demonstrates that AdderBERT performs a powerful representation capacity like BERT even with few multiplications.

We then evaluate the distillation versions of our model against the strong KD baselines, respectively. For 6 layers, DistilBERT$_6$ achieves a 75.8 score on average with 5.64B multiplications and 5.64B additions while AdderBERT-6 achieves a higher 79.6 score with 0.16B multiplications and 11.12B additions. With mapping-based distillation, our proposed model AdderBERT-6 significantly outperforms the baseline DistilBERT$_6$ by a margin of 3.8 on average and obtains a 2.5x reduction in energy consumption as well. Compared to BERT$_{\text{BASE}}$, AdderBERT$_6$ is in much lower energy consumption (4.9x reduction) while maintaining competitive performance (79.7 vs 79.6). This indicates that our proposed KD method

[1]https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

Model	#Mul.	#Add.	Energy (pJ)	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg
$\text{BERT}_{\text{BASE}}$ (T)	11.27B	11.27B	51.8B	83.8/83.1	71.0	90.7	93.9	52.5	85.5	89.1	68.1	**79.7**
AdderBERT_{12}	0.31B	22.23B	**21.1B**	84.3/83.4	70.4	90.9	92.5	50.7	83.7	89.0	65.9	79.0
DistilBERT_{6}	5.64B	5.64B	25.9B	82.1/81.3	69.9	88.9	92.4	47.4	76.2	88.1	56.3	75.8
AdderBERT_{6}	0.16B	11.12B	**10.6B**	84.9/83.1	71.3	91.2	93.8	51.3	83.2	87.9	69.5	**79.6**
DistilBERT_{3}	2.82B	2.82B	13.0B	73.4/72.9	66.0	81.3	85.6	27.5	75.1	80.2	61.0	69.2
AdderBERT_{3}	0.08B	5.56B	**5.3B**	80.7/80.9	68.6	88.0	90.7	45.9	79.3	87.2	65.2	**76.3**

Table 1: Results from the GLUE test server. The best results for each group are in-bold. All models are learned in a single-task manner. The energy consumption is calculated from the number of multiplications and additions, respectively. Nothing that T denotes the teacher model, and all the 3-layers and 6-layer models are distilled from it while AdderBERT_{12} is undistilled.

can adequately transport linguistic knowledge from the teacher model to the student model. For 3 layers, AdderBERT_3 is consistently better than DistilBERT_3 (a large improvement of 8.1 on average), especially on the challenging CoLA dataset, and it only consumes less than one-tenth of the energy of the teacher model. In conclusion, empirical results validate our motivation that AdderBERT combines the advantage of both BERT and AdderNet, that is, it could obtain comparable results with the teacher model but substantially reduce the energy burden.

4.5 Ablation Study

We further investigate the effectiveness of different distillation objectives on AdderBERT learning. The baselines include without mapping-based distillation (w/o map) or classification distillation (w/o clf), respectively. The results are summarized in Table 2. We can find the performance without mapping-based distillation drops significantly from 76.9 to 71.8, which demonstrates that our proposed method plays the most important role of the two objectives. The reason for the significant drop lies in the distribution gap between H^S and H^T. Linguistic knowledge is hard to transport across completely different representations.

Model	MNLI-m	MNLI-mm	MRPC	CoLA	Avg
AdderBERT_6	84.3	83.4	89.0	50.7	76.9
w/o map	80.5	77.8	84.3	44.6	71.8
w/o clf	82.0	79.3	88.6	46.9	74.2

Table 2: Ablation studies of different distillation objectives in the AdderBERT learning. The results are validated on the dev set.

5 Conclusion

In this paper, we propose an energy-efficient version of BERT, called AdderBERT. Specifically, AdderBERT consists of several adder encoders implemented by cheap addition operations but has a powerful representation capacity. It adopts a unique mapping-based distillation method to narrow the gap in feature distribution between the teacher and student model. Empirical results on the GLUE benchmark demonstrate that our method can achieve highly competitive performance to the teacher $\text{BERT}_{\text{BASE}}$ while reducing energy consumption significantly.

Acknowledgements

We are grateful to Yunhe Wang, and Yixing Xu for their helpful discussions. This work is supported by Huawei Technologies Co., Ltd.

References

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. 2009. The fifth PASCAL recognizing textual entailment challenge. In *TAC*. NIST.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS*.

Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval@ACL*, pages 1–14. Association for Computational Linguistics.

Hanting Chen, Yunhe Wang, Chunjing Xu, Boxin Shi, Chao Xu, Qi Tian, and Chang Xu. 2020. Addernet: Do we really need multiplications in deep learning? In *CVPR*, pages 1465–1474. Computer Vision Foundation / IEEE.

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

B. Dally. 2015. High-performance hardware for machine learning.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics.

Jianbang Ding, Xuancheng Ren, Ruixuan Luo, and Xu Sun. 2019. An adaptive and momental bound method for stochastic learning. *CoRR*, abs/1910.12249.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *IWP@IJCNLP*. Asian Federation of Natural Language Processing.

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir D. Bourdev. 2014. Compressing deep convolutional networks using vector quantization. *CoRR*, abs/1412.6115.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both weights and connections for efficient neural network. In *NIPS*, pages 1135–1143.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling BERT for natural language understanding. In *EMNLP (Findings)*, volume EMNLP 2020 of *Findings of ACL*, pages 4163–4174. Association for Computational Linguistics.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *ICLR*. OpenReview.net.

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

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392. The Association for Computational Linguistics.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. Fitnets: Hints for thin deep nets. In *ICLR (Poster)*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Han Shu, Jiahao Wang, Hanting Chen, Lin Li, Yujiu Yang, and Yunhe Wang. 2021. Adder attention for vision transformer. In *NeurIPS*, pages 19899–19909.

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 *EMNLP*, pages 1631–1642. ACL.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *EMNLP/IJCNLP (1)*, pages 4322–4331. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR (Poster)*. OpenReview.net.

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

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Trans. Assoc. Comput. Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, pages 1112–1122. Association for Computational Linguistics.

Yixing Xu, Chang Xu, Xinghao Chen, Wei Zhang, Chunjing Xu, and Yunhe Wang. 2020. Kernel based progressive distillation for adder neural networks. In *NeurIPS*.