BitNet: Scaling 1-bit Transformers for Large Language Models

2023

HongyuWang*†‡, Shuming Ma*†, Li Dong†, Shaohan Huang†, Huaijie Wang§,
Lingxiao Ma†, Fan Yang†, Ruiping Wang‡, Yi Wu§, Furu Wei†⋄

† Microsoft Research § Tsinghua University

‡ University of Chinese Academy of Sciences

Part 1.

Background

- **Large Language Model(LLM)**
  - Significant improvements in various tasks
    - Large language models (LMs) elicit strong reasoning capabilities
    - High inference costs and energy consumption
  - Model quantization
    - Reduce the memory footprint and computational cost of large-scale models while maintaining competitive performance
    - Applicable to deploying these models on distributed systems or multi-device platforms, the inter-device communication

Part 1.

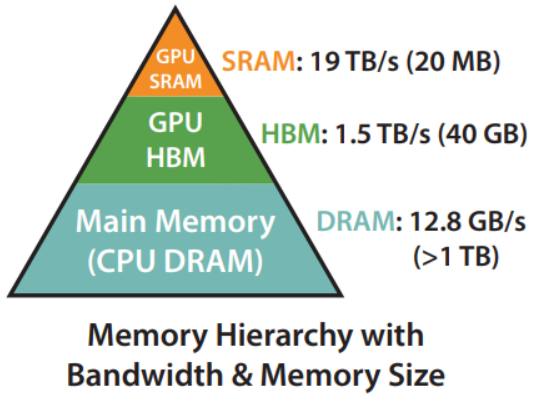
Background

- **Existing quantization approaches**
  - Post-training
    - Simple and easy to apply since it does not require any changes to the training pipeline
    - Result in a more significant loss of accuracy especially when the precision goes lower
  - Quantization-aware training
    - Model is trained to account for the reduced precision from the beginning
    - Typically results in better accuracy
    - Model becomes more difficult to converge as the precision goes lower
    - It is unknown whether quantization-aware training follows the scaling law of neural language models

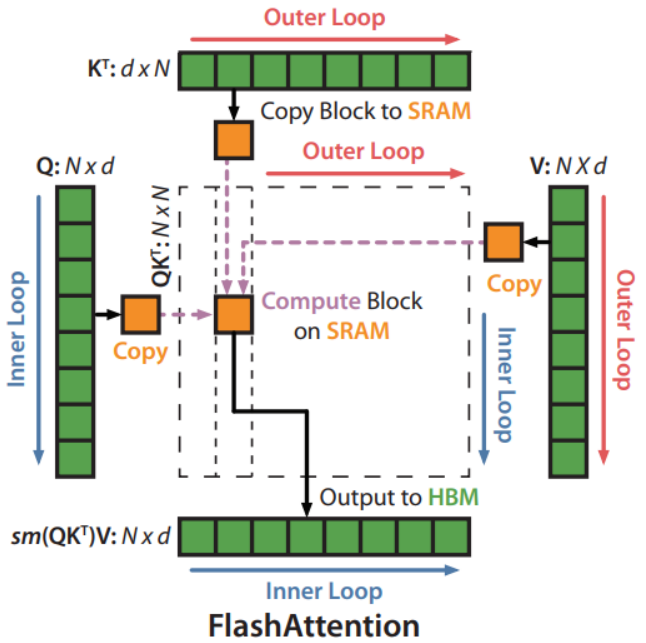
Part 2.

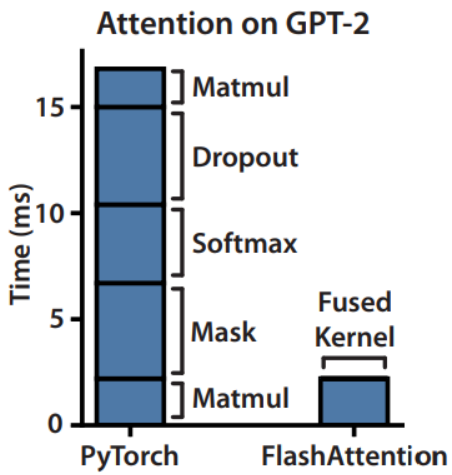
Introduction

- **BitNet**
  - 1-bit Transformer architecture for large language model
    - Low-precision binary weights and quantized activations
    - Maintain high precision for the optimizer states and gradients during training
  - Implementation of the BitNet architecture
    - Quite simple, requiring only the replacement of linear projections in the Transformer
    - it complements other acceleration methods, such as FlashAttention [DFE+22, Dao23]



Memory Hierarchy with Bandwidth & Memory Size

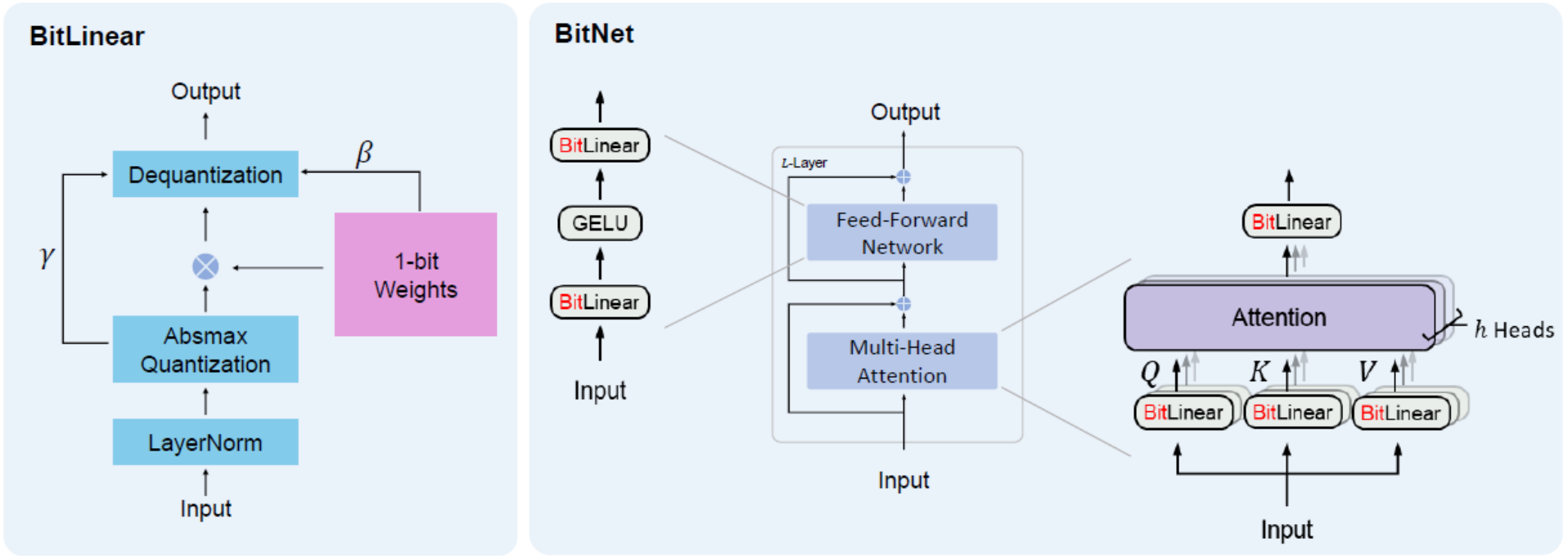




Tri Dao et al. FlashAttention: fast and memory-efficient exact attention with io-awareness. NeurIPS. 2022.
Tri Dao. FlashAttention-2: faster attention with better parallelism and work partitioning. CoRR, abs/2307.08691. 2023.

Method

- **BitNet Architecture**
  - Leave the other components high-precision without BitLinear (e.g., 8-bit)
    - Residual connections & layer normalization contribute negligible computation costs to large language models
    - Computation cost of QKV transformation is much smaller than the parametric projection
    - Preserve the precision for input/output embedding with using high-precision probabilities to perform sampling



Part 3.

Method

- **BitLinear**
  - Weight Quantization
    - Centralize the weights to be zero-mean before binarization
    - To increase the capacity within a limited numerical range

$$W \in \mathcal{R}^{n \times m}$$

$$\widetilde{W} = \text{Sign}(W - \alpha) \qquad \text{Sign}(W_{ij}) = \begin{cases} +1, & \text{if } W_{ij} > 0 \\ -1, & \text{if } W_{ij} \leq 0 \end{cases} \qquad \alpha = \frac{1}{nm} \sum_{ij} W_{ij}$$

Zechun Liu et al. BiT: robustly binarized multi-distilled transformer. NeurIPS. 2022.
Tim Dettmers et al. LLM.int8(): 8-bit matrix multiplication for transformers at scale. CoRR, 2022.

Part 3.

Method

- **BitLinear**
  - Activation Quantization (Absmax Quantization)
    - Scales activations into the range $[-Q_b, Q_b]$ $(Q_b = 2^{b-1})$ by multiplying with Q_b and dividing by the absolute maximum of the input matrix

$$\widetilde{x} = \text{Quant}(x) = \text{Clip}(x \times \frac{Q_b}{\gamma}, -Q_b + \epsilon, Q_b - \epsilon) \quad \text{Clip}(x, a, b) = \max(a, \min(b, x)) \quad \gamma = ||x||_\infty$$

  - Activation Quantization (Absmax Quantization)
    - Before the non-linear functions (e.g., ReLU)
    - Scale them into the range by subtracting the minimum of the inputs so that all values are non-negative
    - Quantize the activation to 8-bit and leave lower precision

$$\widetilde{x} = \text{Quant}(x) = \text{Clip}((x - \eta) \times \frac{Q_b}{\gamma}, \epsilon, Q_b - \epsilon) \quad \eta = \min_{ij} x_{ij}$$

Zechun Liu et al. BiT: robustly binarized multi-distilled transformer. NeurIPS. 2022.
Tim Dettmers et al. LLM.int8(): 8-bit matrix multiplication for transformers at scale. CoRR, 2022.

Part 3.

Method

- **BitLinear** $y = \widetilde{W}\widetilde{x}$
  - Elements in W & x are mutually independent and share the same distribution, and W and x are independent of each other
  - Variance of the output $\mathrm{Var}(y)$
    - With the standard initialization methods (e.g., Kaiming initialization or Xavier initialization)
    - Great benefit to the training stability

$$\mathrm{Var}(y) = n\mathrm{Var}(\widetilde{w}\widetilde{x}) = nE[\widetilde{w}^2]E[\widetilde{x}^2] = n\beta^2 E[\widetilde{x}^2] \approx E[\widetilde{x}^2]$$

  - Layernorm function before the activation quantization
    - To preserve the variance after quantization $\mathrm{Var}(y) \approx E[\mathrm{LN}(\widetilde{x})^2] = 1$
    - Exact implementation as SubLN [WMH+22]

$$y = \widetilde{W}\widetilde{x} = \widetilde{W}\ \mathrm{Quant}(\mathrm{LN}(x)) \times \frac{\beta\gamma}{Q_b} \quad \mathrm{LN}(x) = \frac{x - E(x)}{\sqrt{\mathrm{Var}(x) + \epsilon}} \quad \beta = \frac{1}{nm}||W||_1$$

Hongyu Wang et al. Foundation transformers. CoRR, 2022.

Method

- **Model parallelism with Group Quantization & Normalization**
  - Technique to scale up large language models
  - Partitions the matrix multiplication on multiple devices

  - Prerequisite for the existing model parallelism approaches
    - Tensors are independent along the partition dimension
    - However, all of the parameters $\alpha, \beta, \gamma, \eta$ are calculated from the whole tensors, breaking the independent prerequisite
    - Existing solution: all-reduce operation

  - The problem also exists in SubLN
    - The amount of synchronization is growing as the model becomes deeper
    - It significantly slows the forward pass

Part 3.

Method

- **Model parallelism with Group Quantization & Normalization**
  - Group Quantization
    - Divide weights & activations into groups
    - Then independently estimate each group's parameters
    - Parameters can be calculated locally without requiring additional communication
  - Group Quantization Process
    - Divde weight matrix $W \in \mathcal{R}^{n \times m}$ into G groups along the partition dimension

$$\gamma_g = ||x^{(g)}||_\infty \qquad \eta_g = \min_{ij} x_{ij}^{(g)}$$

    - Each group has a size of $\frac{n}{G} \times m$
    - Estimate the parameters for each group independently (e.g., Group normalization [WH20])

$$\alpha_g = \frac{G}{nm} \sum_{ij} W_{ij}^{(g)} \qquad \beta_g = \frac{G}{nm} ||W^{(g)}||_1 \qquad \mathrm{LN}(x^{(g)}) = \frac{x^{(g)} - E(x^{(g)})}{\sqrt{\mathrm{Var}(x^{(g)}) + \epsilon}}$$

Yuxin Wu and Kaiming He. Group normalization. Int. J. Comput. Vis., 128(3):742–755, 2020.

Method

- **Model Training**
  - Straight-through estimator (STE) [BLC13]
    - Approximate the gradient during backpropagation for non-differentiable functions
    - Make it possible to train our quantized model.
  - Mixed precision training
    - Maintain a latent weight in a high-precision format for the learnable parameters to accumulate the parameter updates
    - Latent weights are binarized on the fly during the forward pass and never used for the inference process
  - Large learning rate
    - Small update on the latent weights often makes no difference in the 1-bit weights
    - This problem is even worse at the beginning of the training, where the models are supposed to converge as fast as possible
    - Increasing the learning rate is the simplest and best way to accelerate the optimization

Yoshua Bengio et al. Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432. 2013

Method

- **Computational Efficiency: Arithmetic operations energy**
  - ADD[Hor14], MUL[ZZL22] energy consumption at 45nm and 7nm process nodes

Bits	ADD Energy $\hat{E}_{add}$ (pJ) 45nm	7nm	MUL Energy $\hat{E}_{mul}$ (pJ) 45nm	7nm
FP32	0.9	0.38	3.7	1.31
FP16	0.4	0.16	1.1	0.34
INT8	0.03	0.007	0.2	0.07

  - Vanilla Transformers
    - Energy consumption can be calculated for matrix multiplication ($m \times n$, $n \times p$)

$$E_{add} = m \times (n-1) \times p \times \hat{E}_{add} \quad E_{mul} = m \times n \times p \times \hat{E}_{mul}$$

  - BitNet
    - Multiplication operations are only applied to scale the output with the scalars β and $\frac{\gamma}{Q_b}$

$$E_{mul} = (m \times p + m \times n) \times \hat{E}_{mul}$$

Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). ISSCC. 2014.
Yichi Zhang, Zhiru Zhang, and Lukasz Lew. PokeBNN: A binary pursuit of lightweight accuracy. In IEEE/CVF.2022.

Method

- **Computational Efficiency: Arithmetic operations energy**
  - Energy savings of W1A8 BitNet compared to a full-precision (32-32) and half-precision (16-16) Transformer
    - Provides significant energy savings, especially for the multiplication operations
    - Use the major component of the matrix multiplication energy consumption

Models	Size	WBits	7nm Energy (J)		45nm Energy (J)	
			MUL	ADD	MUL	ADD
Transformer	6.7B	32	4.41	1.28	12.46	3.03
		16	1.14	0.54	3.70	1.35
BitNet		1	**0.02**	**0.04**	**0.08**	**0.13**
Transformer	13B	32	8.58	2.49	24.23	5.89
		16	2.23	1.05	7.20	2.62
BitNet		1	**0.04**	**0.06**	**0.12**	**0.24**
Transformer	30B	32	20.09	5.83	56.73	13.80
		16	5.21	2.45	16.87	6.13
BitNet		1	**0.06**	**0.14**	**0.20**	**0.53**

Part 4.

Experiment

- **Setup**
  - Comparison with FP16 Transformers
    - Model: BitNet of various scales, ranging from 125M to 30B
    - Dataset: Pile dataset, Common Crawl snapshots, RealNews, CC-Stories datas
    - Tokenizer: Sentencpiece, vocabulary 16K
  - Comparison with Post-training Quantization
    - Model: BitNet, Absmax, SmoothQuant, GPTQ, and QuIP
    - The same Dataset and Tokenizer

Model configuration for BitNet

Params	# Hidden	# Layers	# Heads	Learning Rate
125M	768	12	12	2.4e-3
350M	1024	24	16	1.2e-3
760M	1536	24	16	1e-3
1.3B	2048	24	32	8e-4
2.7B	2560	32	32	6.4e-4
6.7B	4096	32	32	4.8e-4
13B	5120	40	40	4e-4
30B	7168	48	56	4e-4

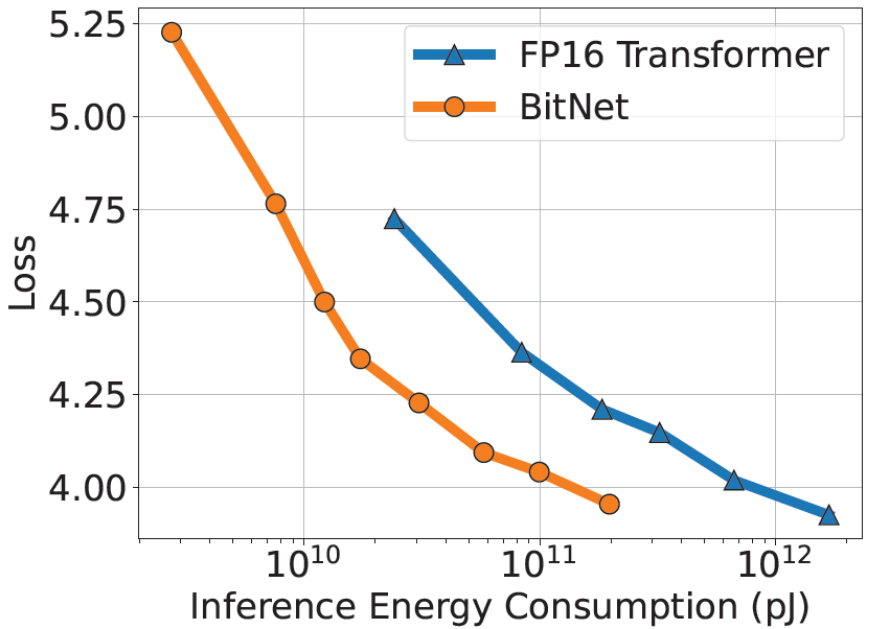
Hyperparameters for BitNet and the FP16 Transformers

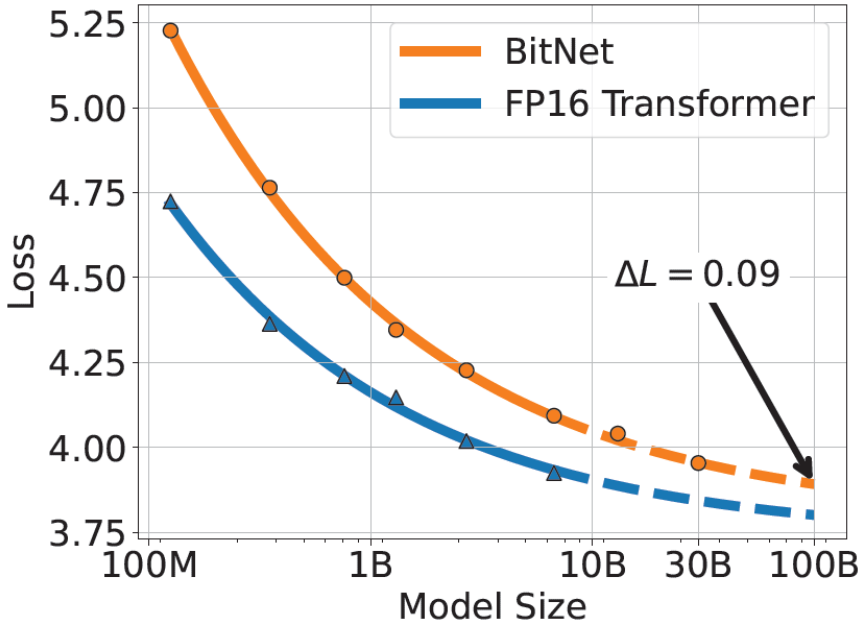
Hyperparameters	Value
Training updates	40K
Tokens per sample	256K
Adam β	(0.9, 0.98)
Learning rate schedule	Polynomial decay
Warmup updates	750
Gradient clipping	✗
Dropout	✗
Attention dropout	✗
Weight decay	0.01

Experiment

- **Inference-Optimal Scaling Law (with FP16 Transformers)**
  - Scale predictably [KMH+20] to determine optimal allocation of a computation budget
  - Predict the loss against the energy consumption
  - Given a fixed computation budget, BitNet achieves a significantly better loss
  - Inference cost is much smaller to get the same performance as the FP16 models

Scaling curves of BitNet and FP16 Transformers





Scaling Law

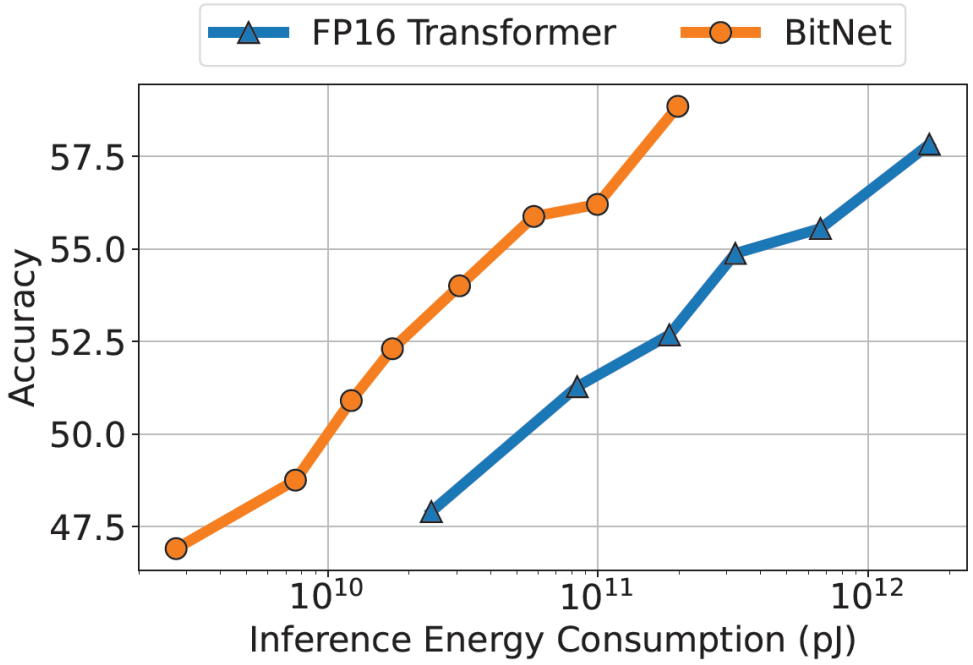
$$L(N) = aN^b + c$$

Jared Kaplan et al. Scaling laws for neural language models. CoRR, abs/2001.08361, 2020.

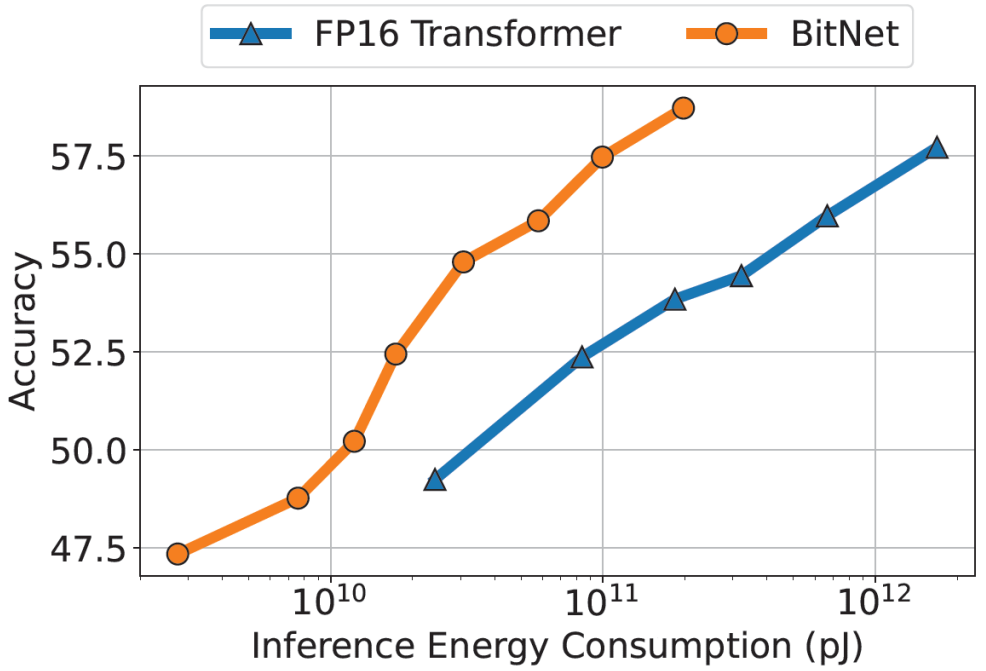
Experiment

- **Results on Downstream Tasks**
  - 0-shot and 4-shot performance of BitNet and FP16 Transformer on four downstream tasks, including Hellaswag,Winogrande, Winograd, Storycloze
  - Performance on the downstream tasks can scale as the computation budget grows
  - Scaling efficiency of capabilities is much higher than the FP16 Transformer baseline

Zero-shot

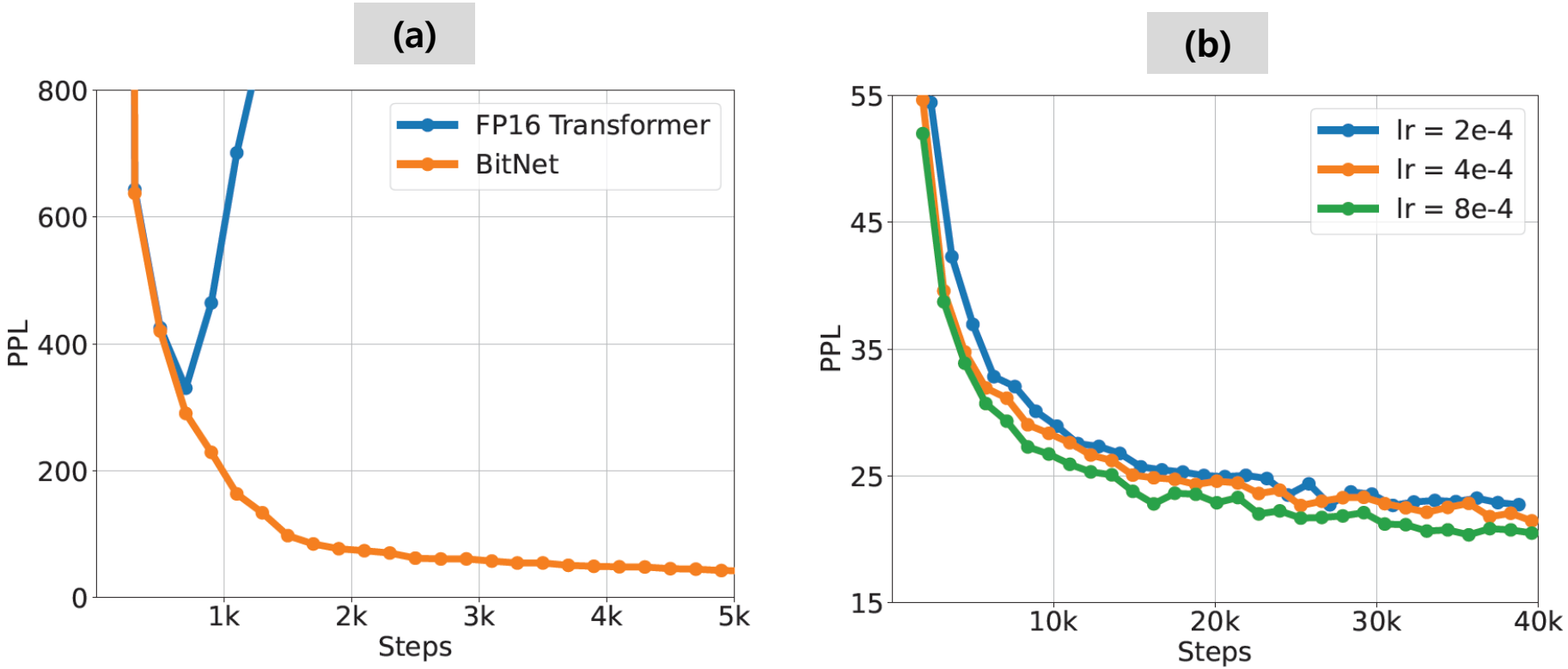


Few-shot



Experiment

- **Stability Test**
  - (a) BitNet can converge with a large learning rate while FP16 Transformer can not
  - (a) Better training stability of BitNet in optimization enables the training with larger learning rates
  - (b) The more large learning rate is, The better BitNet training convergence is



Experiment

- **Comparison with FP16 Transformers**
  - Setup
    - Model: BitNet of various scales, ranging from 125M to 30B
    - Dataset: Pile dataset, Common Crawl snapshots, RealNews, CC-Stories datas
    - Tokenizer: Sentencpiece, vocabulary 16K

Model configuration for BitNet

Params	# Hidden	# Layers	# Heads	Learning Rate
125M	768	12	12	2.4e-3
350M	1024	24	16	1.2e-3
760M	1536	24	16	1e-3
1.3B	2048	24	32	8e-4
2.7B	2560	32	32	6.4e-4
6.7B	4096	32	32	4.8e-4
13B	5120	40	40	4e-4
30B	7168	48	56	4e-4

Hyperparameters for BitNet and the FP16 Transformers

Hyperparameters	Value
Training updates	40K
Tokens per sample	256K
Adam β	(0.9, 0.98)
Learning rate schedule	Polynomial decay
Warmup updates	750
Gradient clipping	✗
Dropout	✗
Attention dropout	✗
Weight decay	0.01

Experiment

- **Comparison with Post-training Quantization**
  - All models have the model sizes of 6.7B for a fair comparison
  - Zero-shot scores of BitNet are comparable with the 8-bit models
  - BitNet Significantly achieves better results than both the weight-and-activation quantization methods and the weight-only methods

WBits	Methods	PTQ	PPL↓	WG↑	WGe↑	HS↑	SC↑	Avg↑
-	Random	✗	-	50.0	50.0	25.0	50.0	43.8
16	Transformer	✗	15.19	66.7	54.3	42.9	67.4	57.8
8	Absmax	✓	21.43	60.4	52.0	38.3	62.7	53.4
	SmoothQuant	✓	15.67	65.3	53.1	40.9	67.6	56.7
4	GPTQ	✓	16.05	57.2	51.2	39.9	63.4	52.9
	Absmax	✓	4.8e4	55.8	50.9	25.0	53.1	46.2
	SmoothQuant	✓	1.6e6	53.7	48.3	24.8	53.6	45.1
2	GPTQ	✓	1032	51.6	50.1	25.8	53.4	45.2
	QuIP	✓	70.43	56.1	51.2	30.3	58.4	49.0
1	Absmax	✓	3.5e23	49.8	50.0	24.8	53.6	44.6
	SmoothQuant	✓	3.3e21	50.5	49.5	24.6	53.1	44.4
1	**BitNet**	✗	17.07	66.3	51.4	38.9	66.9	55.9

Index

PTQ: Post-training quantization

WGe: Winogrande

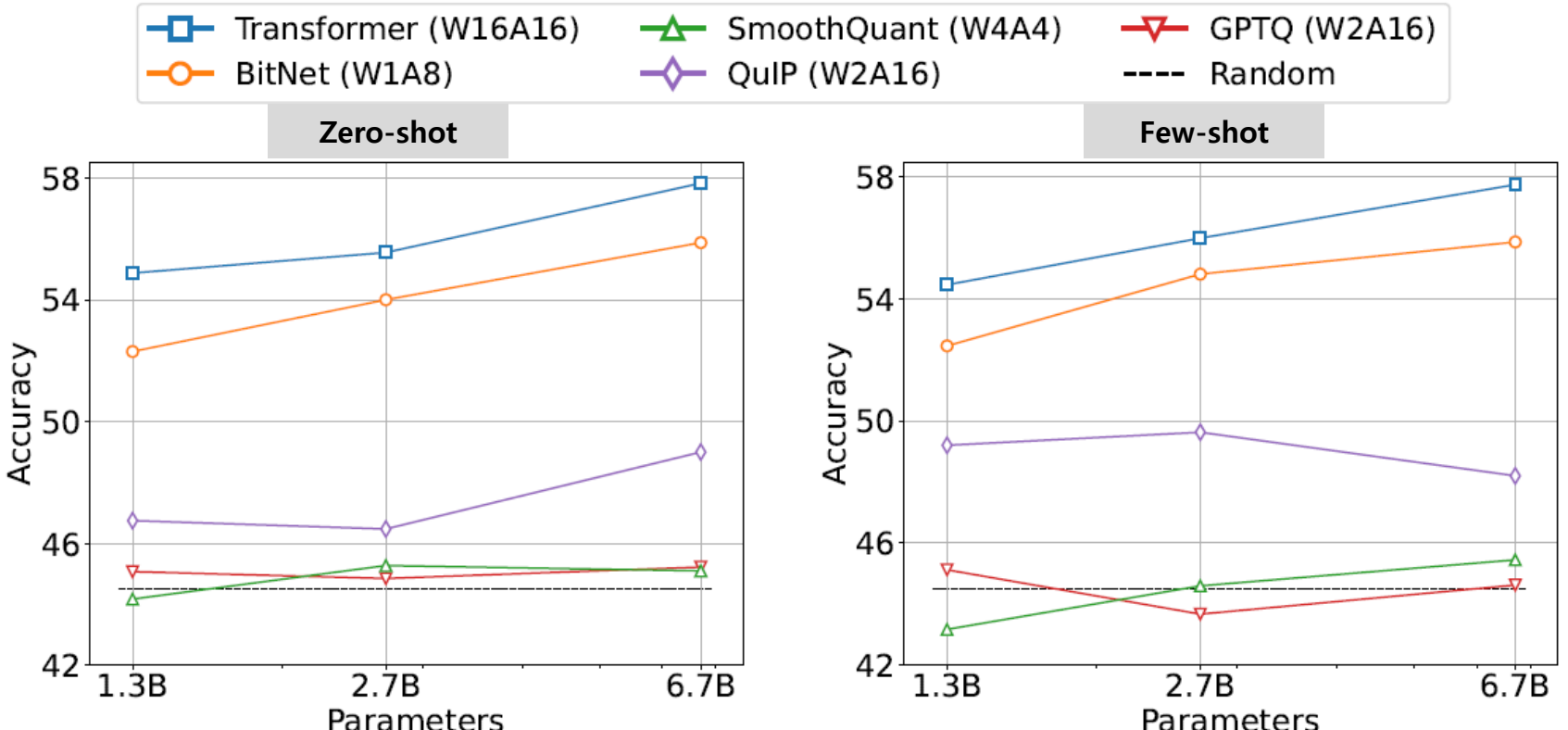
WG: Winograd

SC: Storycloze

HS: Hellaswag dataset

Experiment

- **Comparison with Post-training Quantization**
  - BitNet has consistently superior scores over all baselines
  - The advantages of the quantization-aware training approaches over the post-training quantization methods
  - Proves that the advantage is consistent across different scales



Experiment

- **Ablation Studies**
  - absmax
    - Better performance, more stable training
    - Enables a larger learning rate for BitNet
  - SubLN with the Pre-LN
    - Pre-LN is the default architecture for GPT pertaining
    - Improve the stability of binarized model

Methods	PPL↓	HS↑	WGe↑	WG↑	SC↑	Avg↑
Zero-Shot Learning						
BitNet	**20.34**	33.2	52.1	60.7	63.2	**52.3**
Elastic + Pre-LN	24.05	29.6	52.9	56.8	61.3	50.2
Absmax + Pre-LN	22.11	31.6	50.0	61.8	61.6	51.3
Absmax + BMT	22.98	31.2	52.1	60.4	62.7	51.6
Few-Shot Learning						
BitNet	**20.34**	33.5	50.4	62.1	63.8	**52.5**
Elastic + Pre-LN	24.05	29.9	51.7	57.5	61.1	50.1
Absmax + Pre-LN	22.11	31.4	51.9	63.9	61.6	52.2
Absmax + BMT	22.98	31.3	51.5	57.5	62.6	50.7

Part 5.

Conclusion

- **BitNet**
  - 1-bit Transformer architecture for large language models
  - Performance
    - Achieve competitive performance in terms of both perplexity and downstream task performance
    - Significantly reducing memory footprint and energy consumption compared to the baselines
  - Scaling Law
    - BitNet follows a scaling law similar to that of full-precision Transformers
    - It can be effectively scaled to even larger language models with potential benefits in terms of performance and efficiency
  - Further Work
    - Scale up BitNet in terms of model size and training steps
    - Apply BitNet in other architectures (e.g., RetNet) for training large language models