The Era of 1-bit LLMs:
All Large Language Models are in 1.58 Bits

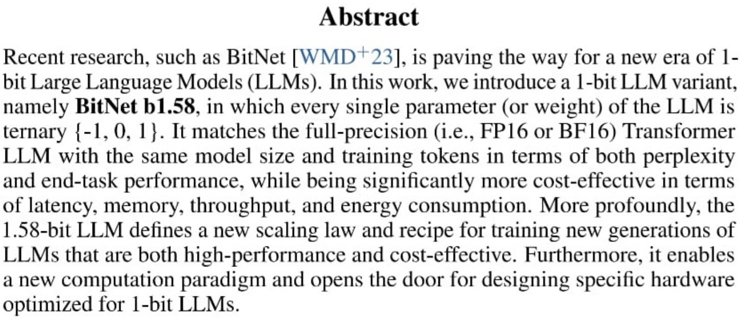
A Brief Overview

Adhilsha Ansad

SMLab Talks
National Institute of Science Education and Research, Bhubaneswar

22 April, 2024

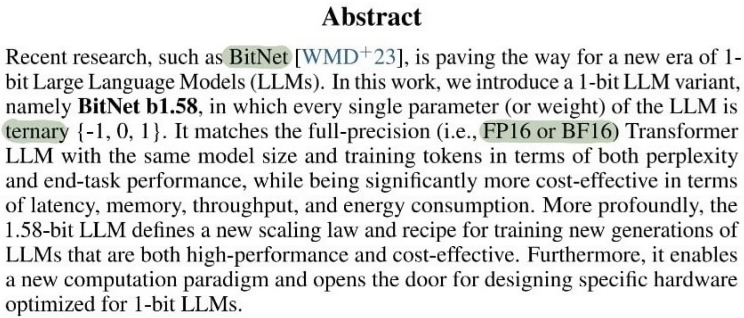
Abstract



Abstract

Recent research, such as BitNet [WMD+23], is paving the way for a new era of 1-bit Large Language Models (LLMs). In this work, we introduce a 1-bit LLM variant, namely **BitNet b1.58**, in which every single parameter (or weight) of the LLM is ternary {-1, 0, 1}. It matches the full-precision (i.e., FP16 or BF16) Transformer LLM with the same model size and training tokens in terms of both perplexity and end-task performance, while being significantly more cost-effective in terms of latency, memory, throughput, and energy consumption. More profoundly, the 1.58-bit LLM defines a new scaling law and recipe for training new generations of LLMs that are both high-performance and cost-effective. Furthermore, it enables a new computation paradigm and opens the door for designing specific hardware optimized for 1-bit LLMs.

Abstract



Abstract

Recent research, such as BitNet [WMD+23], is paving the way for a new era of 1-bit Large Language Models (LLMs). In this work, we introduce a 1-bit LLM variant, namely **BitNet b1.58**, in which every single parameter (or weight) of the LLM is ternary {-1, 0, 1}. It matches the full-precision (i.e., FP16 or BF16) Transformer LLM with the same model size and training tokens in terms of both perplexity and end-task performance, while being significantly more cost-effective in terms of latency, memory, throughput, and energy consumption. More profoundly, the 1.58-bit LLM defines a new scaling law and recipe for training new generations of LLMs that are both high-performance and cost-effective. Furthermore, it enables a new computation paradigm and opens the door for designing specific hardware optimized for 1-bit LLMs.

Outline

1. Recap Essential Ideas
2. BitNet, the predecessor
3. BitNet b1.58
4. Key Takeaways

FP16 / BF16

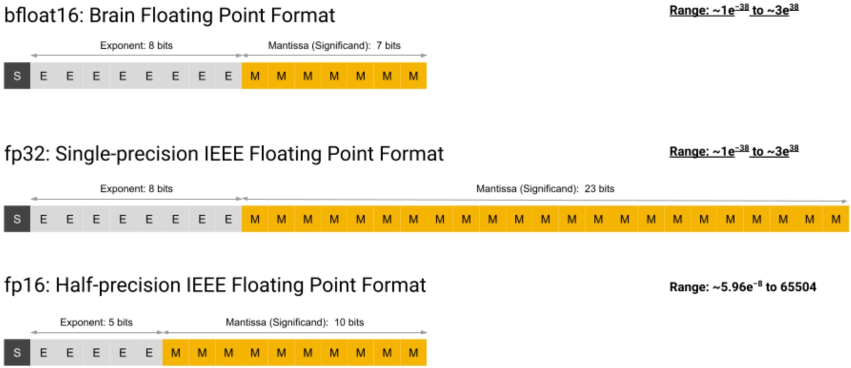
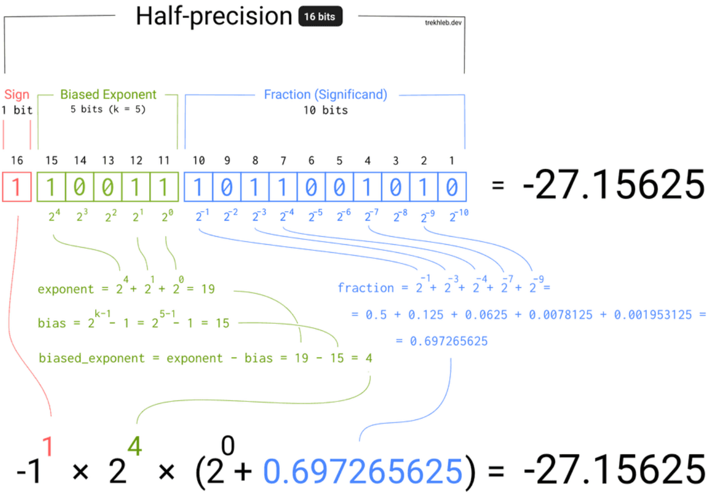

bfloat16: Brain Floating Point Format
Range: $\sim 1e^{-38}$ to $\sim 3e^{38}$
Exponent: 8 bits
Mantissa (Significand): 7 bits
S E E E E E E E E M M M M M M M
fp32: Single-precision IEEE Floating Point Format
Range: $\sim 1e^{-38}$ to $\sim 3e^{38}$
Exponent: 8 bits
Mantissa (Significand): 23 bits
S E E E E E E E E M M M M M M M M M M M M M M M M M M M M M M M
fp16: Half-precision IEEE Floating Point Format
Range: $\sim 5.96e^{-8}$ to 65504
Exponent: 5 bits
Mantissa (Significand): 10 bits
S E E E E E M M M M M M M M M M

Figure: Floating point formats[1]

[1] https://jaeyung1001.tistory.com/entry/bf16-fp16-fp32

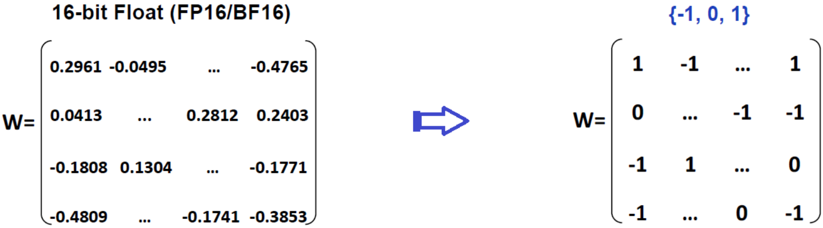
Floating Point format (IEEE 754[2,3])



[2] technical standard for floating-point arithmetic

[3] https://www.h-schmidt.net/FloatConverter/IEEE754.html

Ternary weights



16-bit Float (FP16/BF16)

$$W=\begin{pmatrix} 0.2961 & -0.0495 & \dots & -0.4765 \\ 0.0413 & \dots & 0.2812 & 0.2403 \\ -0.1808 & 0.1304 & \dots & -0.1771 \\ -0.4809 & \dots & -0.1741 & -0.3853 \end{pmatrix} \Rightarrow W=\begin{pmatrix} 1 & -1 & \dots & 1 \\ 0 & \dots & -1 & -1 \\ -1 & 1 & \dots & 0 \\ -1 & \dots & 0 & -1 \end{pmatrix}$$

{-1, 0, 1}

Outline

1. Recap Essential Ideas

2. BitNet, the predecessor

3. BitNet b1.58

4. Key Takeaways

BitNet

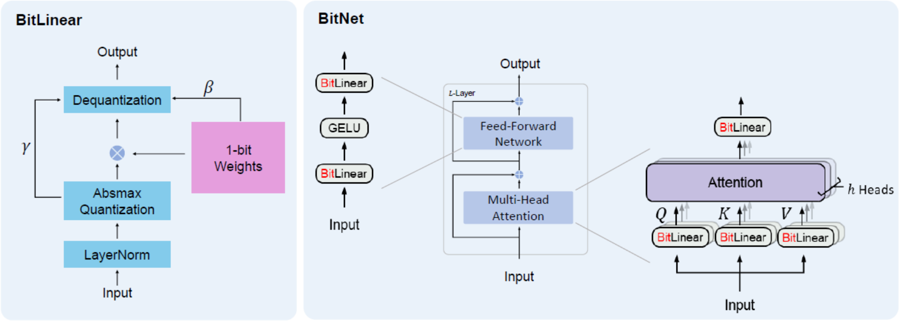


Figure: (a) The computation flow of `BitLinear` (b) The architecture of BitNet[4]

[4] BitNet: Scaling 1-bit Transformers for Large Language Models [Wang et al., 2023a]

BitNet summary

- binarized (i.e., 1-bit) model weights[5]: {-1,1}
  For a weight matrix $W_{m\times n}$,

$$\tilde{W} = \mathrm{Sign}(W - \alpha), \tag{1}$$

$$\mathrm{Sign}(W_{ij}) = \begin{cases} +1 & \text{if } W_{ij} > 0, \\ -1 & \text{if } W_{ij} \leq 0, \end{cases} \tag{2}$$

$$\alpha = \frac{1}{nm} \sum_{ij} W_{ij} \tag{3}$$

- b-bit ***absmax*** quantization: activations range $[-Q_b, Q_b]$, $\left(Q_b = 2^{b-1}\right)$

[5] Components other than matrix multiplication (residual connections, layer normalization etc.) retained precision (8-bit)

BitNet summary

- Quantization steps:

$$\tilde{x} = \mathrm{Quant}(x) = \mathrm{Clip}\left(x \times \frac{Q_b}{\gamma}, -Q_b + \epsilon, Q_b - \epsilon\right), \tag{4}$$

$$\mathrm{Clip}(x, a, b) = \max(a, \min(b, x)), \quad \gamma = \|x\|_\infty, \tag{5}$$

- Activations before non-linear functions: activations range $[0, Q_b]$

$$\tilde{x} = \mathrm{Quant}(x) = \mathrm{Clip}\left((x - \eta) \times \frac{Q_b}{\gamma}, \epsilon, Q_b - \epsilon\right), \ \eta = \min_{ij} x_{ij} \tag{6}$$

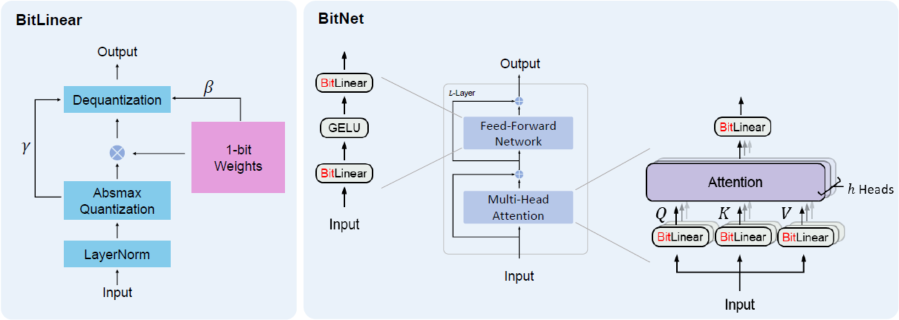
- Now, Matrix multiplication: $y = \tilde{W}\tilde{x}$

BitLinear formulation

$$y = \tilde{W}\tilde{x} = \tilde{W}\,\mathrm{Quant}(\mathrm{LN}(x)) \times \frac{\beta\gamma}{Q_b}, \tag{7}$$

$$\mathrm{LN}(x) = \frac{x - E(x)}{\sqrt{\mathrm{Var}(x) + \epsilon}}, \quad \beta = \frac{1}{nm}\|W\|_1, \tag{8}$$

BitNet


BitLinear
Output
Dequantization
β
γ
1-bit Weights
Absmax Quantization
LayerNorm
Input
BitNet
BitLinear
GELU
BitLinear
Input
Output
L-Layer
Feed-Forward Network
Multi-Head Attention
Input
BitLinear
Attention
h Heads
Q
K
V
BitLinear
BitLinear
BitLinear
Input

BitNet Parallelism

- Model parallelism with Group Quantization and Normalization
  Why?
  - essential technique to scale up large language models
  - prerequisite of independence along partition dimension
  - parameters α, β, γ, and η needs whole tensor.
  - slows the forward pass as model becomes deeper

  What they do? ***Group Quantization***
  - weights and activations into groups
  - independently estimate each group's parameters

  For weight matrix $W \epsilon \mathbb{R}^{n \times m}$ divided into G groups (size $\frac{n}{G} \times m$),

$$\alpha_g = \frac{G}{nm} \sum_{ij} W_{ij}^{(g)}, \quad \beta_g = \frac{G}{nm} \|W^{(g)}\|_1, \tag{13}$$

$$\gamma_g = \|x^{(g)}\|_\infty, \quad \eta_g = \min_{ij} x_{ij}^{(g)}, \quad \mathrm{LN}(x^{(g)}) = \frac{x^{(g)} - E(x^{(g)})}{\sqrt{\mathrm{Var}(x^{(g)}) + \epsilon}} \tag{14}$$

BitNet training

- straight-through estimator (STE) for gradients during BP
- Mixed precision training
    - gradients and the optimizer states in high precision
    - latent weight in a high-precision
- Large learning rate
- autoregressive LMs ranging from 125M to 30B
- trained on an English-language corpus[6]
- Sentencepiece tokenizer and vocabulary size is 16K

[6] Pile dataset, Common Crawl snapshots, RealNews, and CC-Stories datasets

BitNet Results

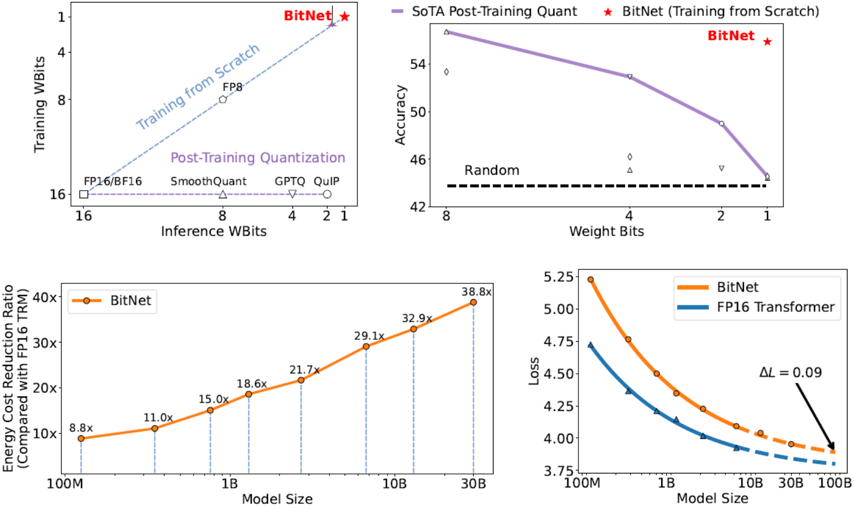
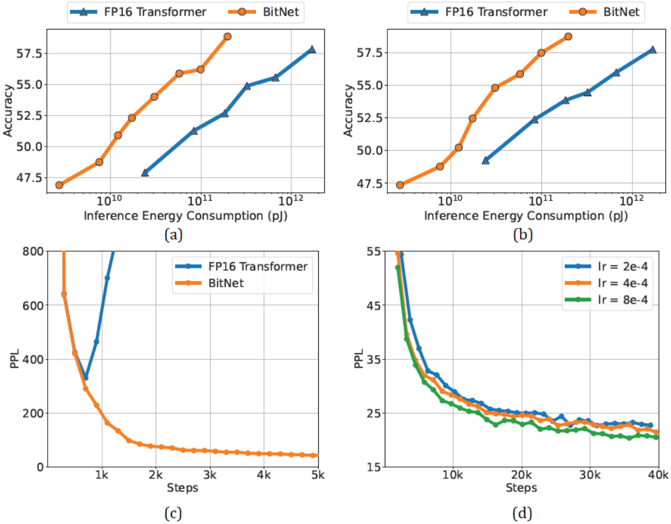


Figure: BitNet Results[7]

BitNet Results



(a)

(b)

(c)

(d)

Figure: (a) Zero-shot and (b) few-shot performance of BitNet. (c) BitNet comparative performance at large learning rate and (d) various learning rates

Outline

1. Recap Essential Ideas
2. BitNet, the predecessor
3. BitNet b1.58
4. Key Takeaways

Motivation

- Increase in size ⟶ Increase in performance
- Increase in size ⟶ challenges for deployment
- Increase in size ⟶ environmental and economic concerns

Solutions?

- Post-training quantization
- attempts to enlarge SRAM
- 1-bit LLMs

Why 1 bit LLMs?

- lower memory footprint
- Efficient bandwidth
- Energy efficient LLMs
- Pareto improvement from existing LLMs

Contributions

- 1-bit LLM variant with ternary weights
- retains all the benefits of BitNet
- More efficient in memory consumption, throughput and latency
- explicit support for ”feature filtering”
- match FP16 baselines in terms of PPL and end-task performance

Architecture

- 1.58-bit weights: -1, 0, 1
- 8-bit activations
- ***absmean*** quantization:

$$\tilde{W} = \text{RoundClip}\left(\frac{W}{\gamma + \epsilon}, -1, 1\right), \tag{1}$$

$$\text{RoundClip}(x, a, b) = \max(a, \min(b, \text{round}(x))), \tag{2}$$

$$\gamma = \frac{1}{nm}\sum_{ij} |W_{ij}|. \tag{3}$$

Architecture

- Same range for activations: $[-Q_b, Q_b]$
  i.e. no *zero-point quantization*
  Why?
  - simple for implementation
  - convenient for system-level optimization
  - negligible effects to the performance[8]
- LLaMA-alike Components
  - RMSNorm
  - SwiGLU[9]
  - rotary embedding

[8] in their experiments

[9] `https://www.ai-contentlab.com/2023/03/swishglu-activation-function.html`

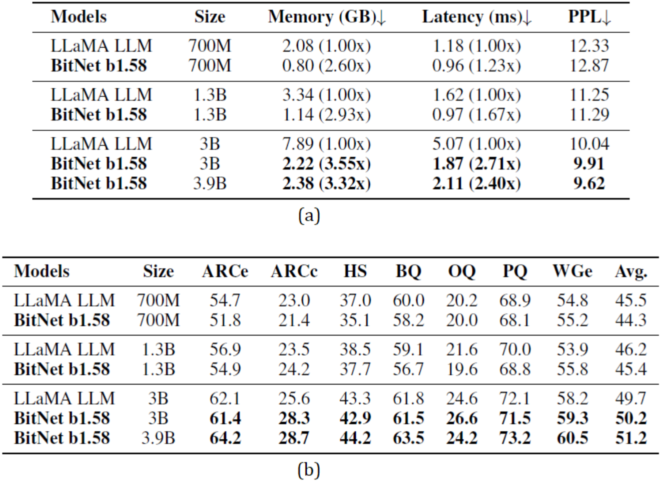
Training

- **BASELINES**: FP16 LLaMA LLM in various sizes.
- **PRE-TRAINING**: RedPajama dataset (100 billion tokens)
- **ZERO-SHOT TASKS**:
  - ARC-Easy and ARC-Challenge
  - Hellaswag
  - Winogrande
  - PIQA
  - OpenbookQA
  - BoolQ
  - WikiText2 and C4 (validation PPL)
- Compares runtime GPU memory, latency and throughput
- used FasterTransformer[10] codebase
- 2-bit kernel from Ladder[11] integrated for BitNet b1.58

[10] https://github.com/NVIDIA/FasterTransformer

[11] [Wang et al., 2023b]

Results



Models	Size	Memory (GB)↓	Latency (ms)↓	PPL↓
LLaMA LLM	700M	2.08 (1.00x)	1.18 (1.00x)	12.33
BitNet b1.58	700M	0.80 (2.60x)	0.96 (1.23x)	12.87
LLaMA LLM	1.3B	3.34 (1.00x)	1.62 (1.00x)	11.25
BitNet b1.58	1.3B	1.14 (2.93x)	0.97 (1.67x)	11.29
LLaMA LLM	3B	7.89 (1.00x)	5.07 (1.00x)	10.04
BitNet b1.58	3B	**2.22 (3.55x)**	**1.87 (2.71x)**	**9.91**
BitNet b1.58	3.9B	**2.38 (3.32x)**	**2.11 (2.40x)**	**9.62**

(a)

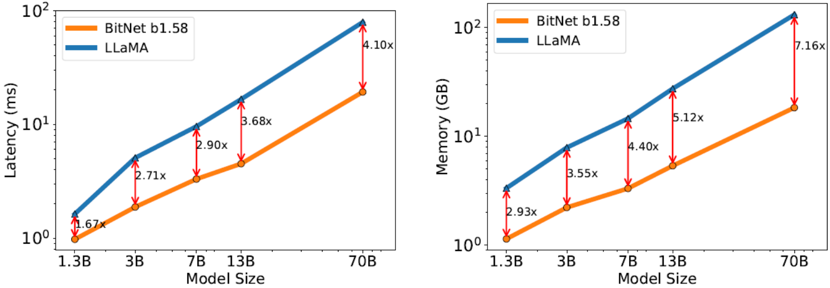
Models	Size	ARCe	ARCc	HS	BQ	OQ	PQ	WGe	Avg.
LLaMA LLM	700M	54.7	23.0	37.0	60.0	20.2	68.9	54.8	45.5
BitNet b1.58	700M	51.8	21.4	35.1	58.2	20.0	68.1	55.2	44.3
LLaMA LLM	1.3B	56.9	23.5	38.5	59.1	21.6	70.0	53.9	46.2
BitNet b1.58	1.3B	54.9	24.2	37.7	56.7	19.6	68.8	55.8	45.4
LLaMA LLM	3B	62.1	25.6	43.3	61.8	24.6	72.1	58.2	49.7
BitNet b1.58	3B	**61.4**	**28.3**	**42.9**	**61.5**	**26.6**	**71.5**	**59.3**	**50.2**
BitNet b1.58	3.9B	**64.2**	**28.7**	**44.2**	**63.5**	**24.2**	**73.2**	**60.5**	**51.2**

(b)

Figure: (a) BitNet b1.58 and LLaMA LLM comparisons and (b) Zero-shot accuracy on the end tasks[12]

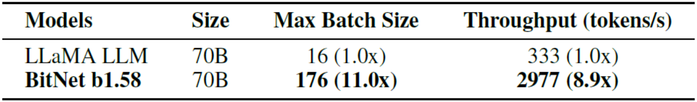
[12] [Ma et al., 2024]

Results



(a)

(b)

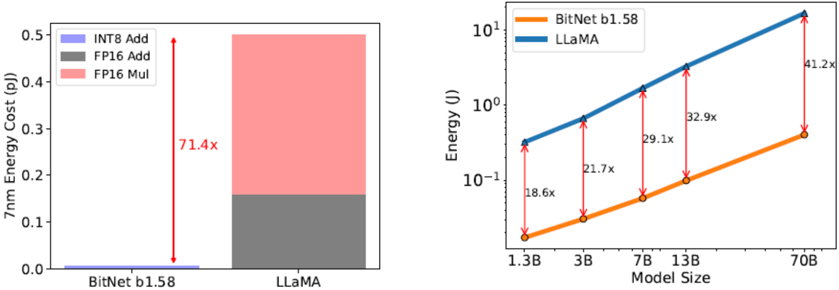


Models	Size	Max Batch Size	Throughput (tokens/s)
LLaMA LLM	70B	16 (1.0x)	333 (1.0x)
BitNet b1.58	70B	**176 (11.0x)**	**2977 (8.9x)**

(c)

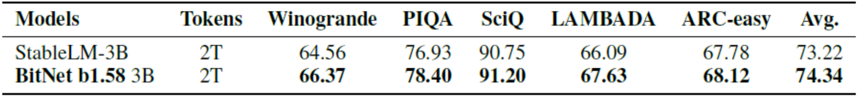
Figure: (a) Decoding latency and (b) memory consumption of BitNet b1.58. (c) Comparison of the throughput between BitNet b1.58 70B and LLaMA LLM 70B

Results



(a)

(b)



Models	Tokens	Winogrande	PIQA	SciQ	LAMBADA	ARC-easy	Avg.
StableLM-3B	2T	64.56	76.93	90.75	66.09	67.78	73.22
BitNet b1.58 3B	2T	**66.37**	**78.40**	**91.20**	**67.63**	**68.12**	**74.34**

(c)

Figure: Comparison of Energy consumption (a) of arithmetic operations energy and (b) end-to-end energy cost. (c) Comparison of BitNet b1.58 with StableLM-3B with 2T tokens.

Results

- Following equivalence are drawn between 1.58-bit and 16-bit models based on efficiency in terms of latency, memory usage and energy consumption:
  - 13B BitNet b1.58 > 3B FP16 LLM
  - 30B BitNet b1.58 > 7B FP16 LLM
  - 70B BitNet b1.58 > 13B FP16 LLM

Outline

1 Recap Essential Ideas

2 BitNet, the predecessor

3 BitNet b1.58

4 Key Takeaways

Key Takeaways

For 1.58 bit LLM,

- For models of similar size, the performance gap narrows as the model size increases.
- Efficient in terms of latency, memory usage and energy consumption.
- Speed-up increases as the model size scales.
- Compared to models of similar size, the throughput is higher.
- Strong generalization capabilities when trained with large number of tokens. i.e. Pareto improvement over the state-of-the-art LLM models.

Possible future works

- Native Support of Long Sequence in LLMs
  - activations from 16 bits to 8 bits, double context length given the same resources/
  - Experiments with lower compressions
- New Hardware for 1-bit LLMs

References

Ma, S., Wang, H., Ma, L., Wang, L., Wang, W., Huang, S., Dong, L., Wang, R., Xue, J., and Wei, F. (2024).
The era of 1-bit llms: All large language models are in 1.58 bits.

Wang, H., Ma, S., Dong, L., Huang, S., Wang, H., Ma, L., Yang, F., Wang, R., Wu, Y., and Wei, F. (2023a).
Bitnet: Scaling 1-bit transformers for large language models.
CoRR, abs/2310.11453.

Wang, L., Ma, L., Cao, S., Zheng, N., Zhang, Q., Xue, J., Miao, Z., Cao, T., and Yang, Y. (2023b).
Ladder: Efficient tensor compilation on customized data format.
In *Proceedings of the 17th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*.