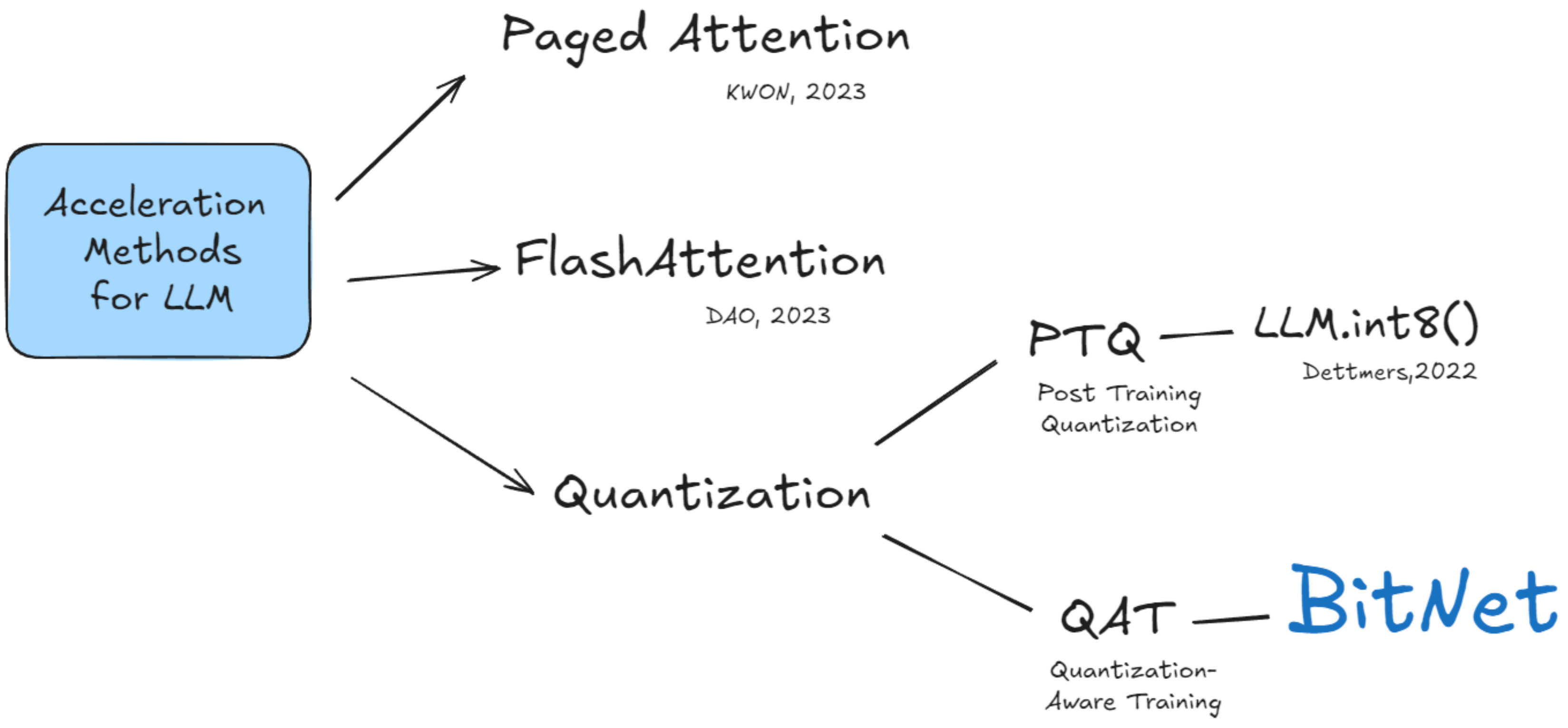
Bitnet 톺아보기

박택현

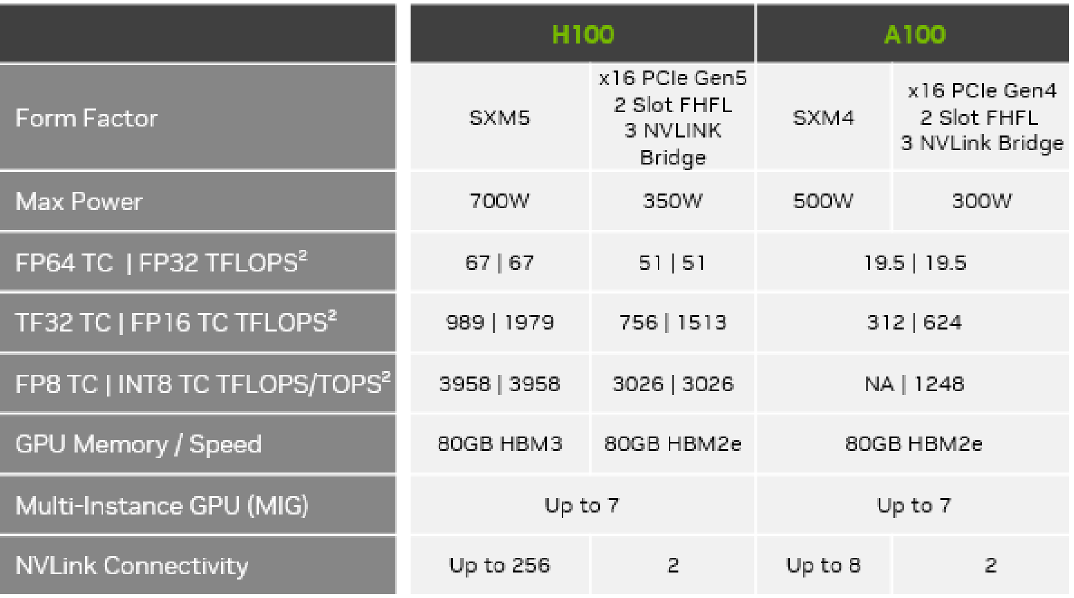
Graduate School of Data Science, PNU, Busan

부산대학교
PUSAN NATIONAL UNIVERSITY

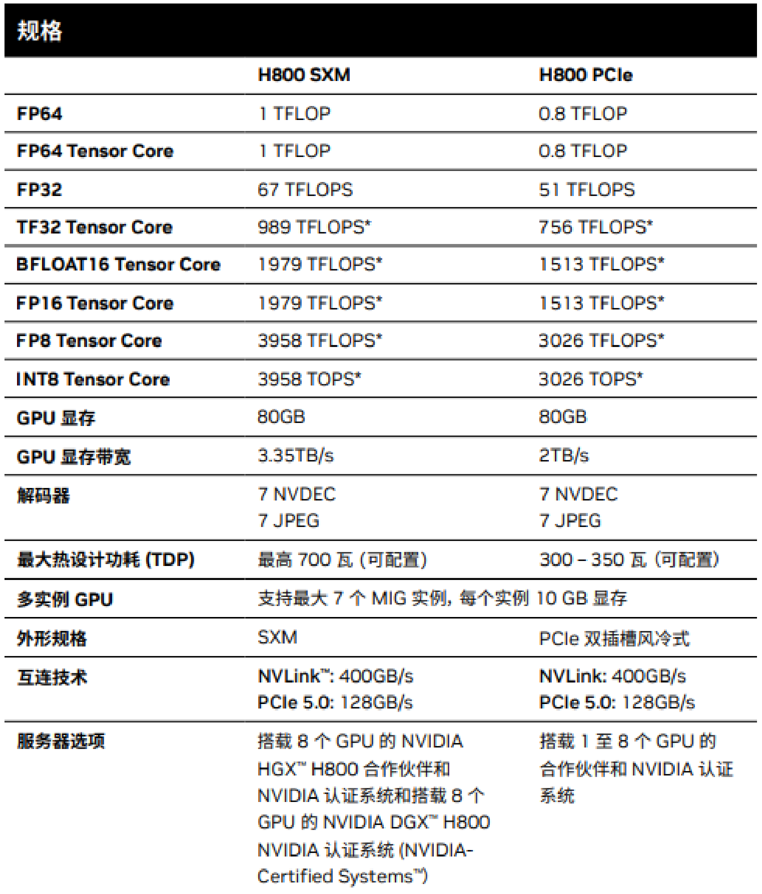
LLM 추론 속도 가속화 연구 동향



LLM 추론 속도 가속화 연구 동향



	H100		A100	
Form Factor	SXM5	x16 PCIe Gen5 2 Slot FHFL 3 NVLINK Bridge	SXM4	x16 PCIe Gen4 2 Slot FHFL 3 NVLink Bridge
Max Power	700W	350W	500W	300W
FP64 TC | FP32 TFLOPS[2]	67 | 67	51 | 51	19.5 | 19.5	
TF32 TC | FP16 TC TFLOPS[2]	989 | 1979	756 | 1513	312 | 624	
FP8 TC | INT8 TC TFLOPS/TOPS[2]	3958 | 3958	3026 | 3026	NA | 1248	
GPU Memory / Speed	80GB HBM3	80GB HBM2e	80GB HBM2e	
Multi-Instance GPU (MIG)	Up to 7		Up to 7	
NVLink Connectivity	Up to 256	2	Up to 8	2



规格

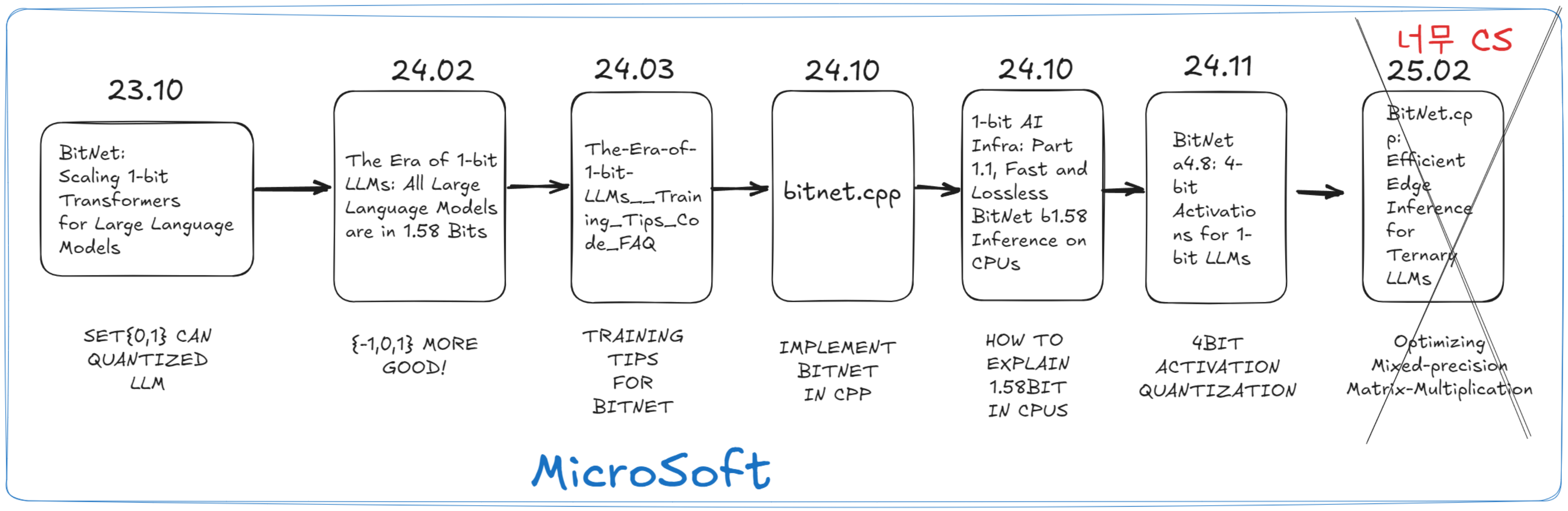
	H800 SXM	H800 PCIe
FP64	1 TFLOP	0.8 TFLOP
FP64 Tensor Core	1 TFLOP	0.8 TFLOP
FP32	67 TFLOPS	51 TFLOPS
TF32 Tensor Core	989 TFLOPS*	756 TFLOPS*
BFLOAT16 Tensor Core	1979 TFLOPS*	1513 TFLOPS*
FP16 Tensor Core	1979 TFLOPS*	1513 TFLOPS*
FP8 Tensor Core	3958 TFLOPS*	3026 TFLOPS*
INT8 Tensor Core	3958 TOPS*	3026 TOPS*
GPU 显存	80GB	80GB
GPU 显存带宽	3.35TB/s	2TB/s
解码器	7 NVDEC 7 JPEG	7 NVDEC 7 JPEG
最大热设计功耗 (TDP)	最高 700 瓦 (可配置)	300 – 350 瓦 (可配置)
多实例 GPU	支持最大 7 个 MIG 实例，每个实例 10 GB 显存	
外形规格	SXM	PCIe 双插槽风冷式
互连技术	NVLink™: 400GB/s PCIe 5.0: 128GB/s	NVLink: 400GB/s PCIe 5.0: 128GB/s
服务器选项	搭载 8 个 GPU 的 NVIDIA HGX™ H800 合作伙伴和 NVIDIA 认证系统和搭载 8 个 GPU 的 NVIDIA DGX™ H800 NVIDIA 认证系统 (NVIDIA-Certified Systems™)	搭载 1 至 8 个 GPU 的合作伙伴和 NVIDIA 认证系统

비트넷 시리즈 논문 대표 저자

Hongyu Wang (王鸿钰 왕홍옥)
중국과기대 18학번,
동대 박사과정 3학년
Citation(649)
GeneralAI(MS 리서치랩) 소속

Shuming Ma(马树铭 마수명)
북경대 12학번, 동대 석사졸
마이크로소프트 연구원
Citation(6801)
GeneralAI(MS 리서치랩) 소속

비트넷 연구 흐름


23.10
BitNet: Scaling 1-bit Transformers for Large Language Models
SET{0,1} CAN QUANTIZED LLM
24.02
The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits
{-1,0,1} MORE GOOD!
24.03
The-Era-of-1-bit-LLMs__Training_Tips_Code_FAQ
TRAINING TIPS FOR BITNET
24.10
bitnet.cpp
IMPLEMENT BITNET IN CPP
24.10
1-bit AI Infra: Part 1.1, Fast and Lossless BitNet b1.58 Inference on CPUs
HOW TO EXPLAIN 1.58BIT IN CPUS
24.11
BitNet a4.8: 4-bit Activations for 1-bit LLMs
4BIT ACTIVATION QUANTIZATION
너무 CS
25.02
BitNet.cpp: Efficient Edge Inference for Ternary LLMs
Optimizing Mixed-precision Matrix-Multiplication
MicroSoft

비트넷이란 무엇인가?

학습 때 부터 **1bit 양자화**를 시켜,
양자화 **손실** 없이
추론이 가능하게 하자!

BitNet: Scaling 1-bit Transformers for Large Language Models

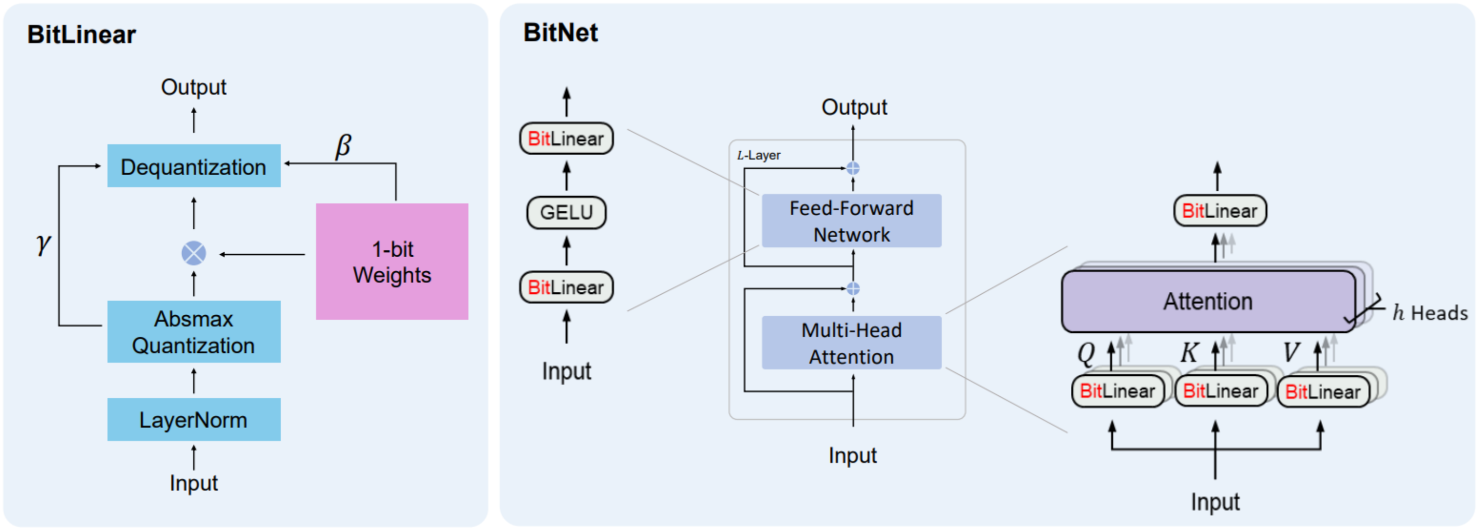


Figure 2: (a) The computation flow of `BitLinear`. (b) The architecture of BitNet, consisting of the stacks of attentions and FFNs, where matrix multiplication is implemented as `BitLinear`.

1. $y = xA^T + b$ 연산의 가중치를 1비트(1, -1)로 만 양자화해서 속도를 가속화 시키자.
2. 입력은 int 8Bit로 양자화 해서 가속화 시키자 (LLM.int8())

BitNet: Scaling 1-bit Transformers for Large Language Models

$$\widetilde{W} = \mathrm{Sign}(W - \alpha),$$

이진화 함수 Sign로 W를 양자화한다.

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

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

$$\widetilde{x} = \mathrm{Quant}(x) = \mathrm{Clip}(x \times \frac{Q_b}{\gamma}, -Q_b + \epsilon, Q_b - \epsilon),$$

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

$[Q_b, -Q_b]$ 범위로 x를 양자화 한다.
본 모델에서는 int8로 양자화 한다.
(llm.int8()와 동일한 구조)

BitNet: Scaling 1-bit Transformers for Large Language Models

$$y = \widetilde{W}\widetilde{x}$$

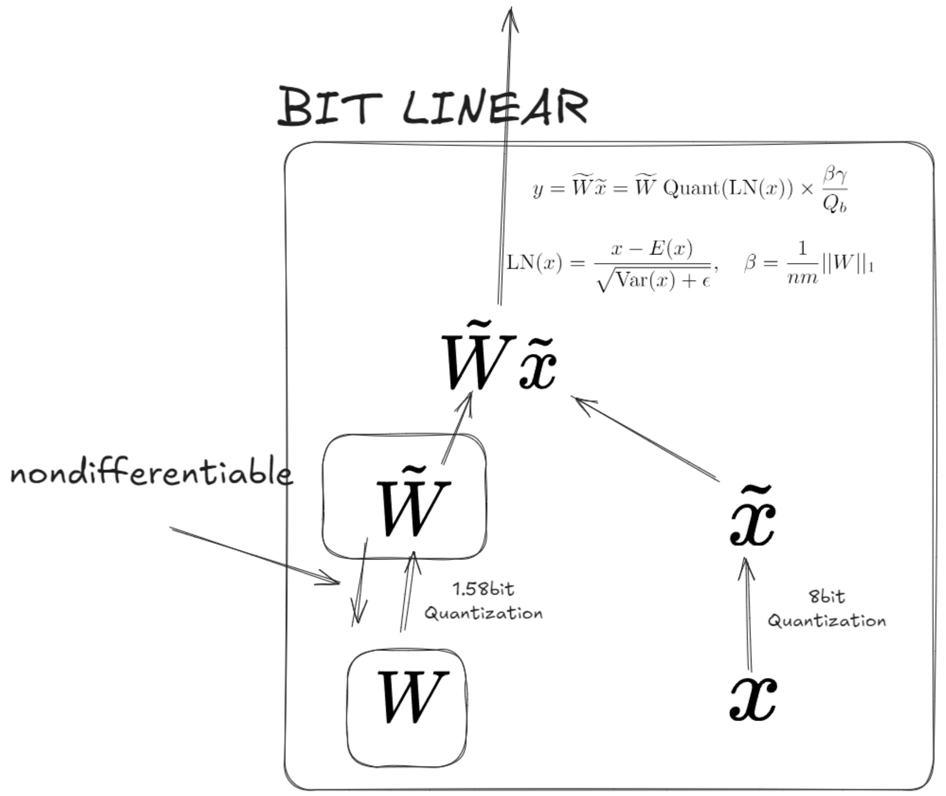
$$\begin{aligned} \mathrm{Var}(y) &= n\mathrm{Var}(\widetilde{w}\widetilde{x}) \\ &= nE[\widetilde{w}^2]E[\widetilde{x}^2] \\ &= n\beta^2 E[\widetilde{x}^2] \approx E[\widetilde{x}^2] \end{aligned}$$

$$y = \widetilde{W}\widetilde{x} = \widetilde{W}\ \mathrm{Quant}(\mathrm{LN}(x)) \times \frac{\beta\gamma}{Q_b}$$

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

Nomarlize 되지 않은 Bitlinear의 Var(y) 를 보게 되면 분산과 평균이 1,0을 만족하지 못하게 되므로 (Kaiming initialization or Xavier initialization)의 조건을 충족할 수 없기 때문에 최종적으로 Layer Normalization 을 추가하여 Bitlinear의 결과값은 다음과 같다.

BitNet: Scaling 1-bit Transformers for Large Language Models


BIT LINEAR
$y = \widetilde{W}\tilde{x} = \widetilde{W}\,\mathrm{Quant}(\mathrm{LN}(x)) \times \frac{\beta\gamma}{Q_b}$
$\mathrm{LN}(x) = \frac{x - E(x)}{\sqrt{\mathrm{Var}(x) + \epsilon}}, \quad \beta = \frac{1}{nm}\|W\|_1$
$\tilde{W}\tilde{x}$
nondifferentiable
$\tilde{W}$
1.58bit Quantization
W
$\tilde{x}$
8bit Quantization
x

Straight-through estimator

역전파를 수행할 때, Sign 함수의 경우 미분 불가능하기 때문에 제약사항이 생기는데 STE(BLC13)에 의해, 미분 불가능한 함수를 그냥 1로 두고 역전파를 수행한다.

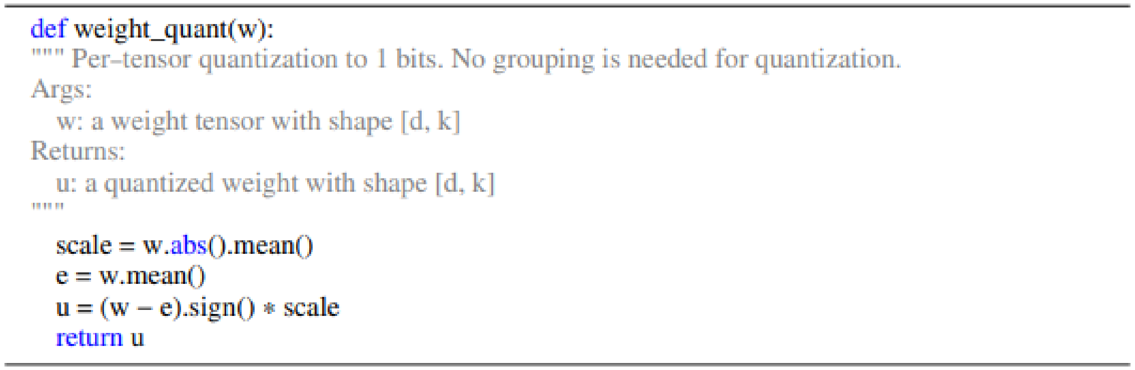
BitNet: Scaling 1-bit Transformers for Large Language Models

```
def activation_quant(x):
""" Per−token quantization to 8 bits. No grouping is needed for quantization.
Args:
    x: an activation tensor with shape [n, d]
Returns:
    y: a quantized activation tensor with shape [n, d]
"""
    scale = 127.0 / x.abs().max(dim=−1, keepdim=True).values.clamp_(min=1e−5)
    y = (x ∗ scale).round().clamp_(−128, 127) / scale
    return y
```

$$\widetilde{x} = \mathrm{Quant}(x) = \mathrm{Clip}(x \times \frac{Q_b}{\gamma}, -Q_b + \epsilon, Q_b - \epsilon),$$

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

BitNet: Scaling 1-bit Transformers for Large Language Models



```
def weight_quant(w):
""" Per−tensor quantization to 1 bits. No grouping is needed for quantization.
Args:
  w: a weight tensor with shape [d, k]
Returns:
  u: a quantized weight with shape [d, k]
"""
  scale = w.abs().mean()
  e = w.mean()
  u = (w − e).sign() ∗ scale
  return u
```

Figure 5: Pytorch code for the BitLinear component in BitNet b1.

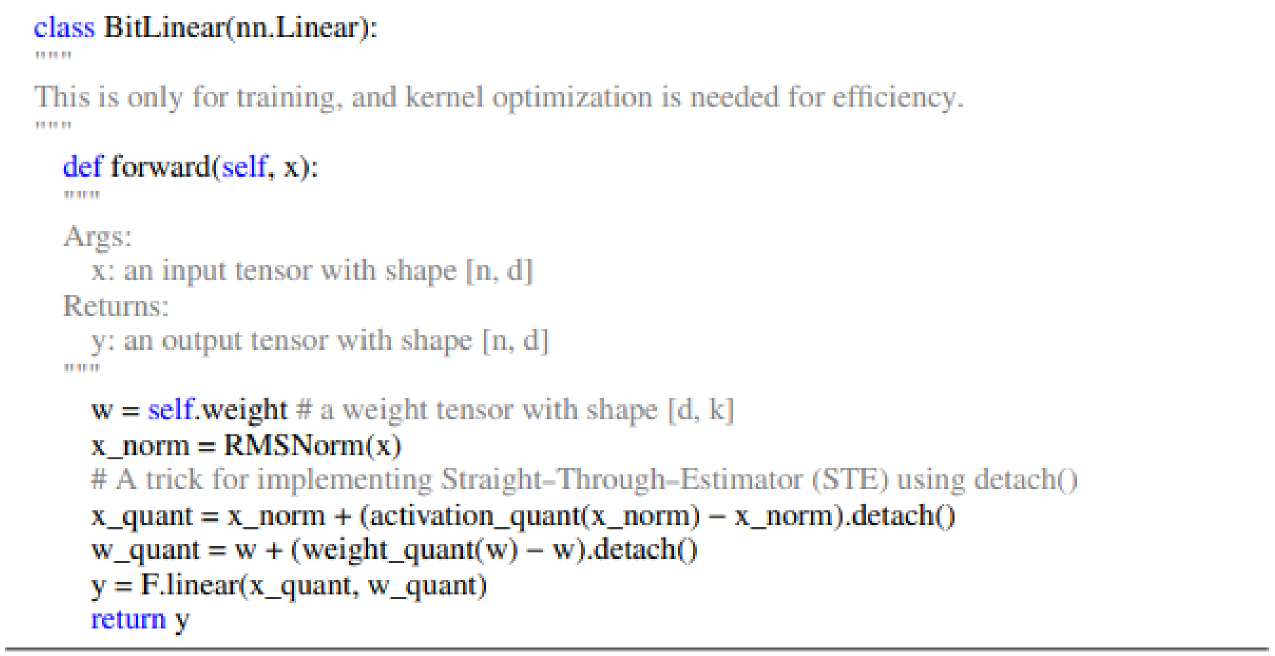
$$\widetilde{W} = \mathrm{Sign}(W - \alpha),$$

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

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

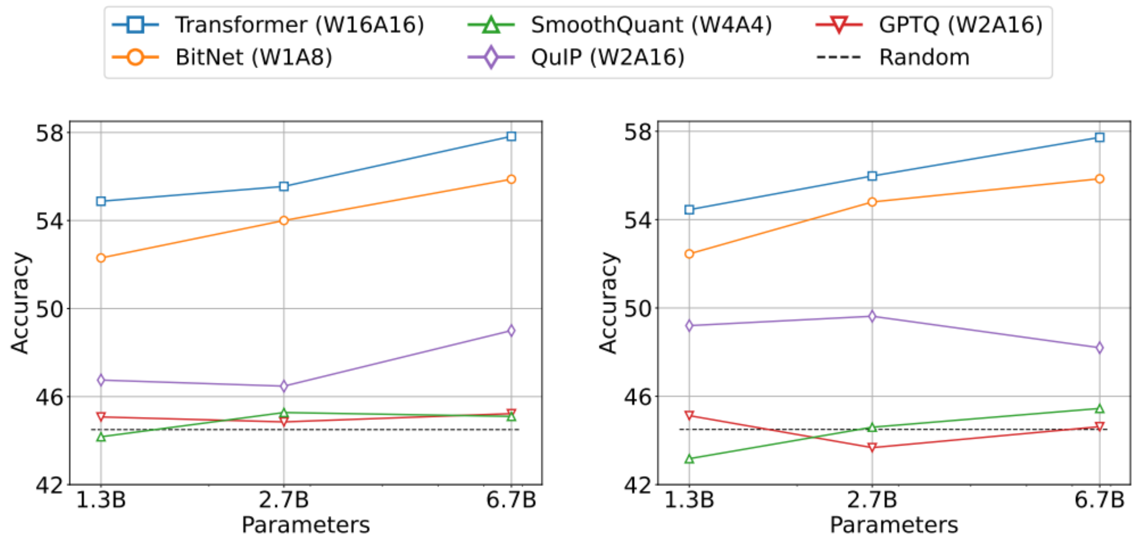
BitNet: Scaling 1-bit Transformers for Large Language Models

- Detach()로 STE를 구현. VQ-VAE에서 Encoding에서 Decoding으로 넘어 갈 때 쓰던 테크닉.
- 이렇게 작성하면 output은 양자화된 입력 역전파가 차단됨.



```
class BitLinear(nn.Linear):
""""
This is only for training, and kernel optimization is needed for efficiency.
""""
  def forward(self, x):
  """"
  Args:
    x: an input tensor with shape [n, d]
  Returns:
    y: an output tensor with shape [n, d]
  """"
    w = self.weight # a weight tensor with shape [d, k]
    x_norm = RMSNorm(x)
    # A trick for implementing Straight-Through-Estimator (STE) using detach()
    x_quant = x_norm + (activation_quant(x_norm) – x_norm).detach()
    w_quant = w + (weight_quant(w) – w).detach()
    y = F.linear(x_quant, w_quant)
    return y
```

BitNet: Scaling 1-bit Transformers for Large Language Models



(a) (b)

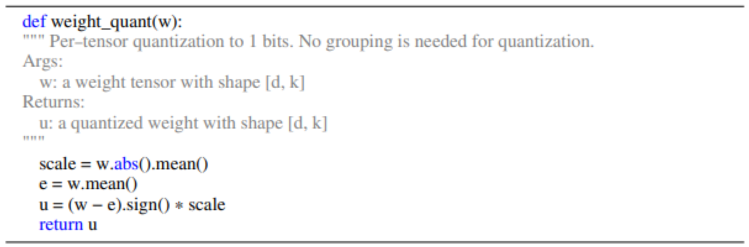
Figure 6: Zero-shot (Left) and few-shot (Right) results for BitNet and the post-training quantization baselines on downstream tasks.

다른 QAT 기반 모델들에 비해서 큰 Accuracy 손실 없이 잘 압축된 것을 확인 할 수 있다.

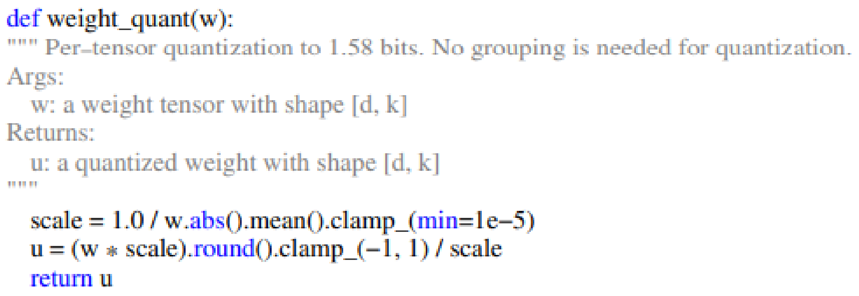
The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits

{-1, 1} 공간에서
{-1,0,1} 공간으로 확장해
양자화를 수행하자!

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



```
def weight_quant(w):
""" Per−tensor quantization to 1 bits. No grouping is needed for quantization.
Args:
  w: a weight tensor with shape [d, k]
Returns:
  u: a quantized weight with shape [d, k]
"""
  scale = w.abs().mean()
  e = w.mean()
  u = (w − e).sign() ∗ scale
  return u
```



```
def weight_quant(w):
""" Per−tensor quantization to 1.58 bits. No grouping is needed for quantization.
Args:
  w: a weight tensor with shape [d, k]
Returns:
  u: a quantized weight with shape [d, k]
"""
  scale = 1.0 / w.abs().mean().clamp_(min=1e−5)
  u = (w ∗ scale).round().clamp_(−1, 1) / scale
  return u
```

1Bit

$$\widetilde{W} = \mathrm{Sign}(W - \alpha),$$

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

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

1.58Bit

$$\widetilde{W} = \mathrm{RoundClip}(\frac{W}{\gamma + \epsilon}, -1, 1),$$

$$\mathrm{RoundClip}(x, a, b) = \max(a, \min(b, \mathrm{round}(x))),$$

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

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

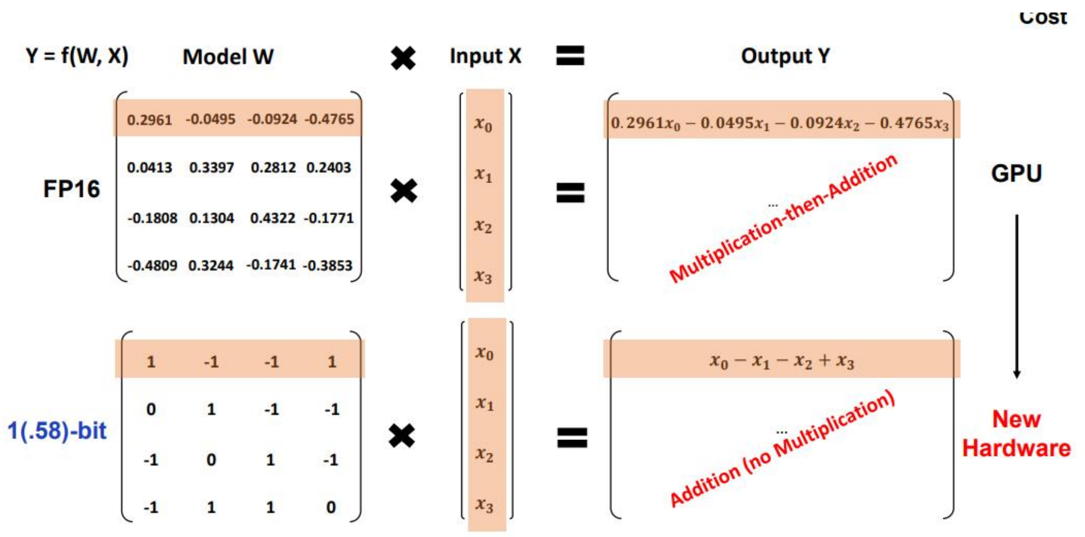
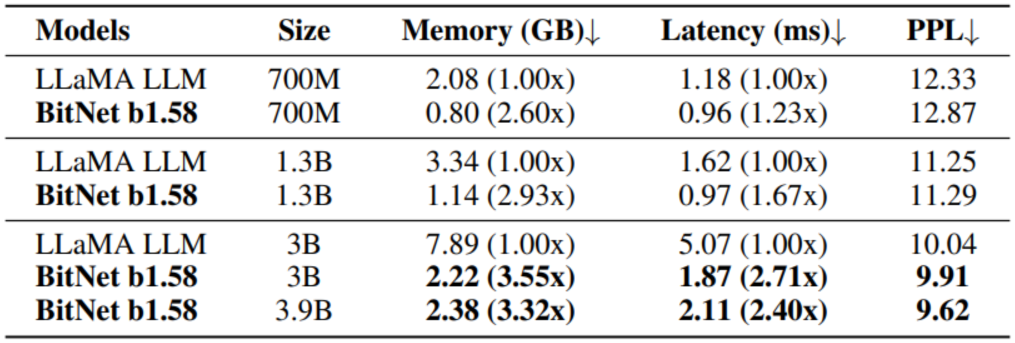


Figure 1: 1-bit LLMs (e.g., BitNet b1.58) provide a Pareto solution to reduce inference cost (latency, throughput, and energy) of LLMs while maintaining model performance. The new computation paradigm of BitNet b1.58 calls for actions to design new hardware optimized for 1-bit LLMs.

FP16: Mul ->Add
1.58Bit: Switch -> Add
(하지만 아키텍처 이슈로 실제로 GPU(TensorCore 이슈) 구현은 X)

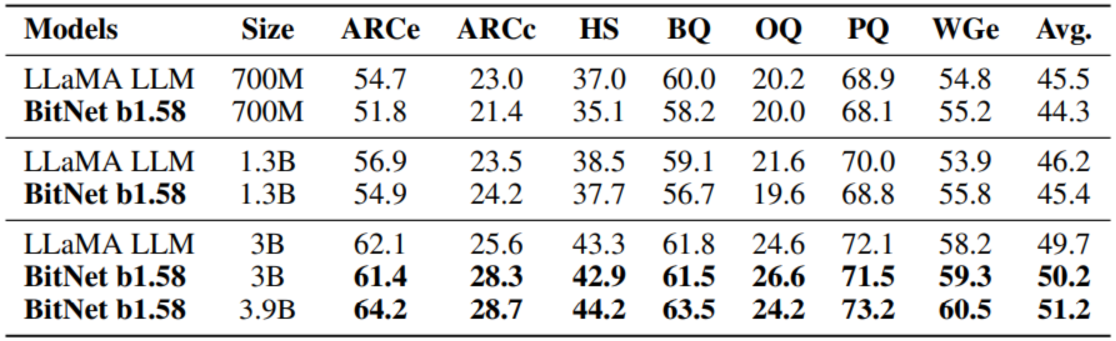
The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits



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**

Table 1: Perplexity as well as the cost of BitNet b1.58 and LLaMA LLM.

성능상승



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**

Table 2: Zero-shot accuracy of BitNet b1.58 and LLaMA LLM on the end tasks.

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

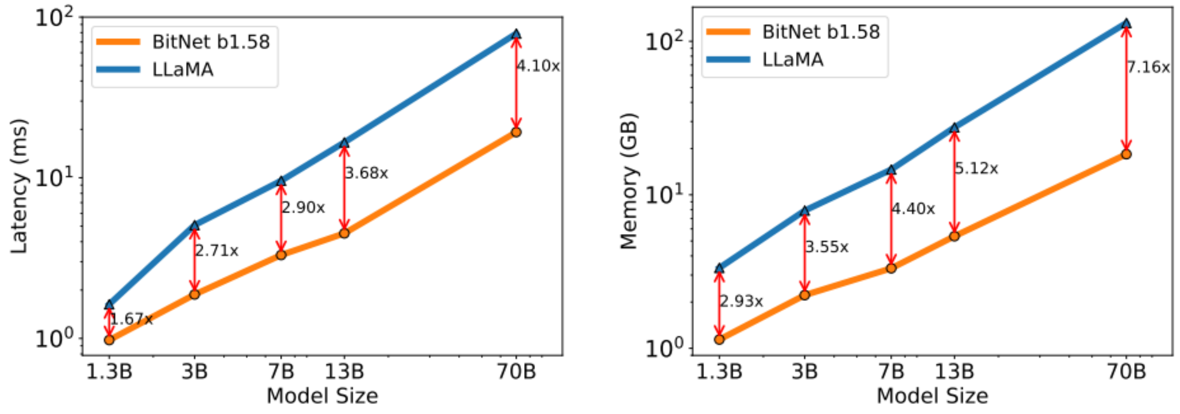


Figure 2: Decoding latency (Left) and memory consumption (Right) of BitNet b1.58 varying the model size.

메모리 효율
연산속도
상승

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)**

Table 3: Comparison of the throughput between BitNet b1.58 70B and LLaMA LLM 70B.

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

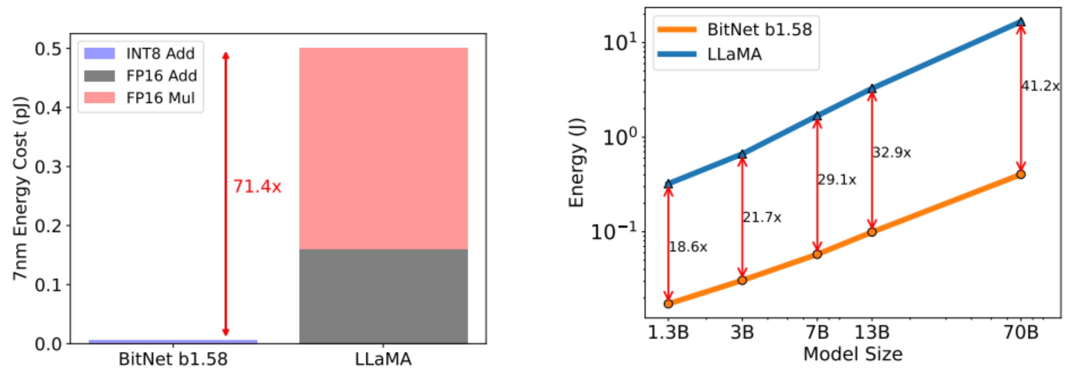


Figure 3: Energy consumption of BitNet b1.58 compared to LLaMA LLM at 7nm process nodes. On the left is the components of arithmetic operations energy. On the right is the end-to-end energy cost across different model sizes.

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**

Table 4: Comparison of BitNet b1.58 with StableLM-3B with 2T tokens.

에너지 효율 상승

1-bit AI Infra: Part 1.1, Fast and Lossless BitNet b1.58 Inference on CPUs

- 일반적으로 CPU에서 LLM을 돌릴려면 LLAMA.cpp를 제일 많이 씀.
- 그런데 LLAMA.cpp는 int8 or float16연산에 최적화됨.(GGUF)
- 이걸 최적화 위해서 llama.cpp 에서

1. 연산 최적화
2. Kernel 최적화

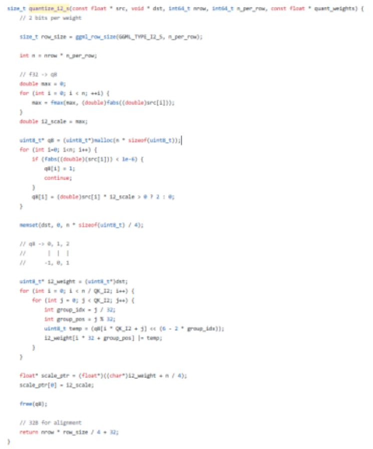
를 통해서 연산 속도를 최적화함.

Backend	Target devices
Metal	Apple Silicon
BLAS	All
BLIS	All
SYCL	Intel and Nvidia GPU
MUSA	Moore Threads MTT GPU
CUDA	Nvidia GPU
HIP	AMD GPU
Vulkan	GPU
CANN	Ascend NPU
OpenCL	Adreno GPU

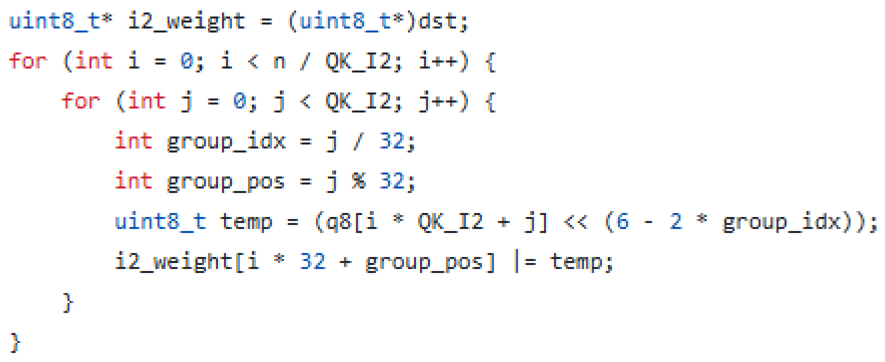
1-bit AI Infra: Part 1.1, Fast and Lossless BitNet b1.58 Inference on CPUs

1. **연산 최적화**

- SIMD 기반 병렬 연산 최적화.
- int8(입력) 와 int2(가중치) 간의 연산을 비트시프트 기반으로 Load 최적화. (가중치를 빠르게 load함.)
- 이부분은 C로 구현되어 있어서 상술하지 않겠음.



```
size_t quantize_i2_s(const float * src, void * dst, int64_t nrow, int64_t n_per_row, const float * quant_weights) {
    // 2 bits per weight

    size_t row_size = ggml_row_size(GGML_TYPE_I2_S, n_per_row);

    int n = nrow * n_per_row;

    // f32 -> q8
    double max = 0;
    for (int i = 0; i < n; ++i) {
        max = fmax(max, (double)fabs((double)src[i]));
    }
    double i2_scale = max;

    uint8_t* q8 = (uint8_t*)malloc(n * sizeof(uint8_t));
    for (int i=0; i<n; i++) {
        if (fabs((double)(src[i])) < 1e-6) {
            q8[i] = 1;
            continue;
        }
        q8[i] = (double)src[i] * i2_scale > 0 ? 2 : 0;
    }

    memset(dst, 0, n * sizeof(uint8_t) / 4);

    // q8 -> 0, 1, 2
    //       |  |  |
    //      -1, 0, 1

    uint8_t* i2_weight = (uint8_t*)dst;
    for (int i = 0; i < n / QK_I2; i++) {
        for (int j = 0; j < QK_I2; j++) {
            int group_idx = j / 32;
            int group_pos = j % 32;
            uint8_t temp = (q8[i * QK_I2 + j] << (6 - 2 * group_idx));
            i2_weight[i * 32 + group_pos] |= temp;
        }
    }

    float* scale_ptr = (float*)((char*)i2_weight + n / 4);
    scale_ptr[0] = i2_scale;

    free(q8);

    // 32B for alignment
    return nrow * row_size / 4 + 32;
}
```



```
uint8_t* i2_weight = (uint8_t*)dst;
for (int i = 0; i < n / QK_I2; i++) {
    for (int j = 0; j < QK_I2; j++) {
        int group_idx = j / 32;
        int group_pos = j % 32;
        uint8_t temp = (q8[i * QK_I2 + j] << (6 - 2 * group_idx));
        i2_weight[i * 32 + group_pos] |= temp;
    }
}
```

https://github.com/microsoft/BitNet/blob/main/src/ggml-bitnet-mad.cpp

1-bit AI Infra: Part 1.1, Fast and Lossless BitNet b1.58 Inference on CPUs

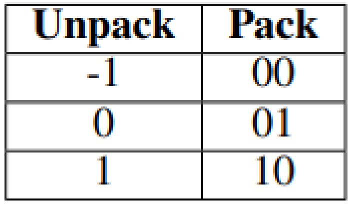
2. Kernel 최적화

- 이론적으로 BitNet은 $\log 3 \cong 1.58$ 비트에 저장이 가능.
- 하지만 일반적으로 한 가중치를 2비트에 저장함.

가중치 2개 => 3.16비트를 4비트에 저장.=> 손해X
가중치 3개 => 4.74비트를 6비트에 저장.=> **1비트** 손해
가중치 4개 => 6.32비트를 8비트에 저장…

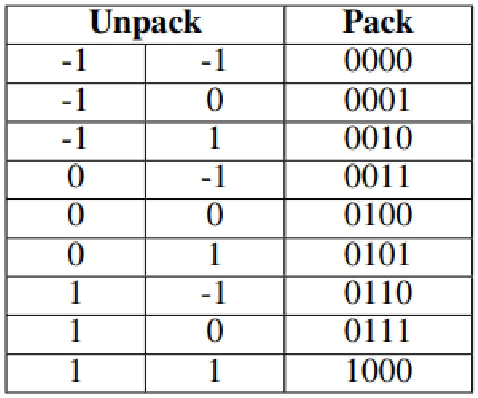
- 따라서 이러한 손해를 메꾸기 위해 **LookUpTable**형태로 비트를 저장해서 손해를 최소화

1-bit AI Infra: Part 1.1, Fast and Lossless BitNet b1.58 Inference on CPUs



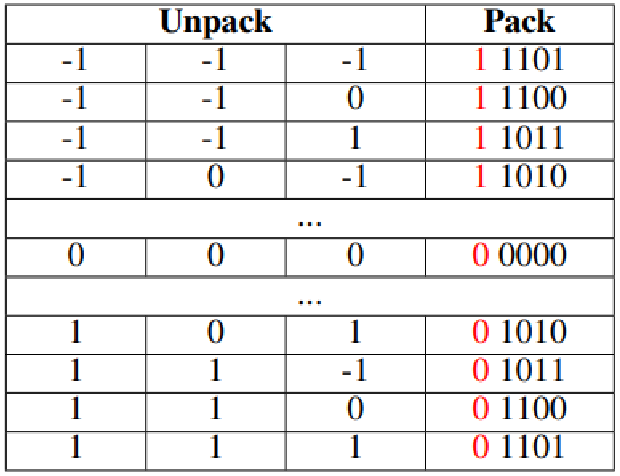
Unpack	Pack
-1	00
0	01
1	10

I2_S



Unpack		Pack
-1	-1	0000
-1	0	0001
-1	1	0010
0	-1	0011
0	0	0100
0	1	0101
1	-1	0110
1	0	0111
1	1	1000

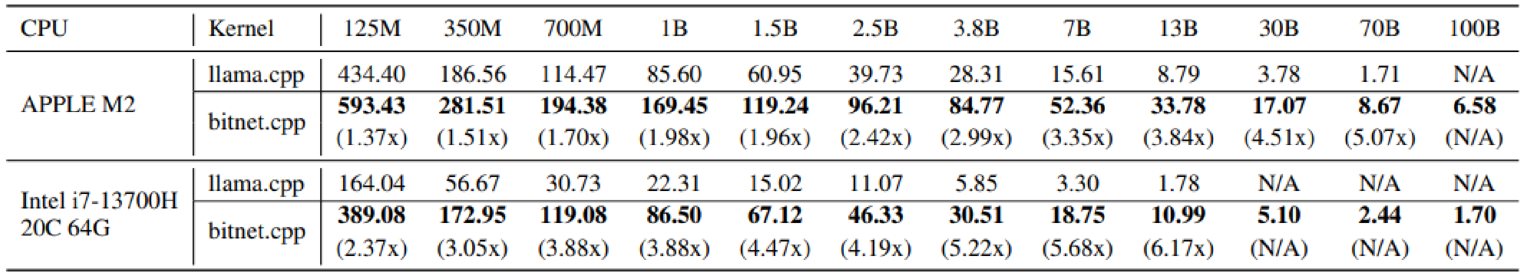
TL_1



Unpack			Pack
-1	-1	-1	1 1101
-1	-1	0	1 1100
-1	-1	1	1 1011
-1	0	-1	1 1010
...			
0	0	0	0 0000
...			
1	0	1	0 1010
1	1	-1	0 1011
1	1	0	0 1100
1	1	1	0 1101

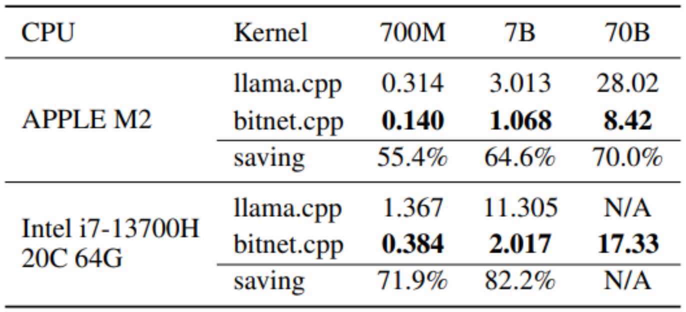
TL_2

1-bit AI Infra: Part 1.1, Fast and Lossless BitNet b1.58 Inference on CPUs



CPU	Kernel	125M	350M	700M	1B	1.5B	2.5B	3.8B	7B	13B	30B	70B	100B
APPLE M2	llama.cpp	434.40	186.56	114.47	85.60	60.95	39.73	28.31	15.61	8.79	3.78	1.71	N/A
	bitnet.cpp	**593.43** (1.37x)	**281.51** (1.51x)	**194.38** (1.70x)	**169.45** (1.98x)	**119.24** (1.96x)	**96.21** (2.42x)	**84.77** (2.99x)	**52.36** (3.35x)	**33.78** (3.84x)	**17.07** (4.51x)	**8.67** (5.07x)	**6.58** (N/A)
Intel i7-13700H 20C 64G	llama.cpp	164.04	56.67	30.73	22.31	15.02	11.07	5.85	3.30	1.78	N/A	N/A	N/A
	bitnet.cpp	**389.08** (2.37x)	**172.95** (3.05x)	**119.08** (3.88x)	**86.50** (3.88x)	**67.12** (4.47x)	**46.33** (4.19x)	**30.51** (5.22x)	**18.75** (5.68x)	**10.99** (6.17x)	**5.10** (N/A)	**2.44** (N/A)	**1.70** (N/A)

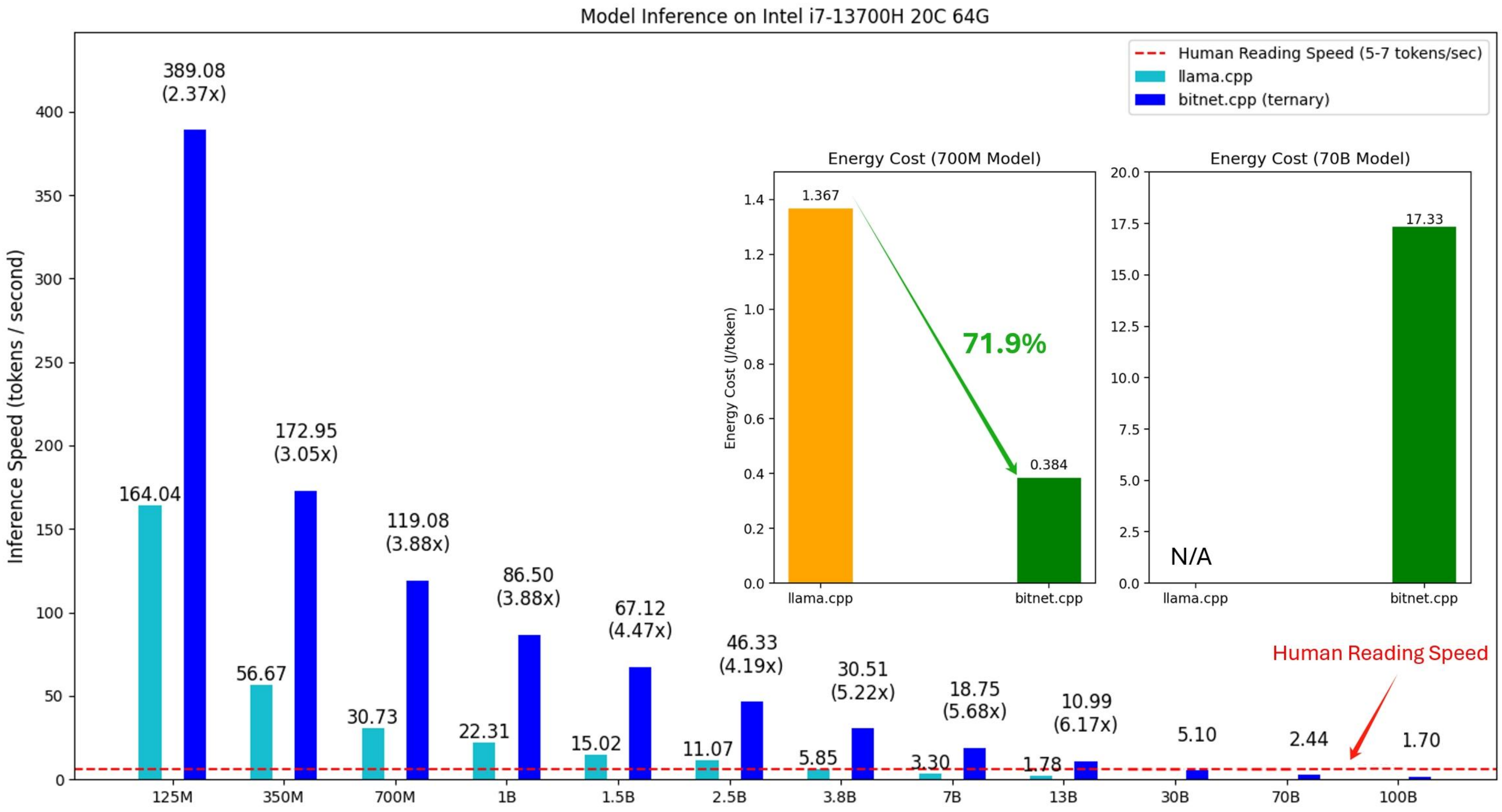
Table 4: Comparison of inference speed across different CPUs (Unit: Tokens/Second) in an unlimited thread setting. "N/A" indicates that the tested CPU cannot host the specified model size with the given kernel.



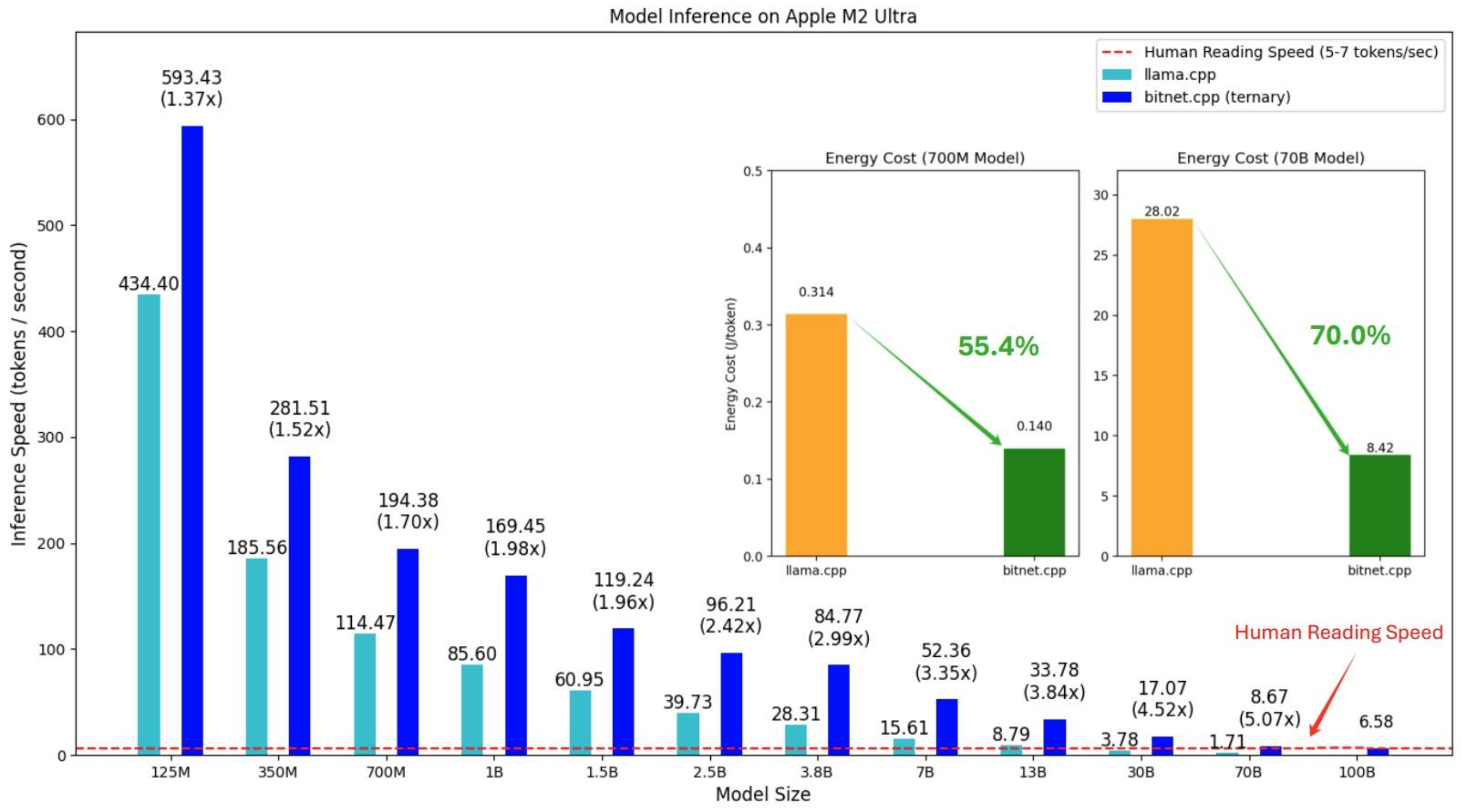
CPU	Kernel	700M	7B	70B
APPLE M2	llama.cpp	0.314	3.013	28.02
	bitnet.cpp	**0.140**	**1.068**	**8.42**
	saving	55.4%	64.6%	70.0%
Intel i7-13700H 20C 64G	llama.cpp	1.367	11.305	N/A
	bitnet.cpp	**0.384**	**2.017**	**17.33**
	saving	71.9%	82.2%	N/A

Table 6: Comparison of Energy Costs Across CPUs (Unit: J/Token). "N/A" indicates that the specific model size cannot be hosted on the tested CPU with the given kernel.

1-bit AI Infra: Part 1.1, Fast and Lossless BitNet b1.58 Inference on CPUs



1-bit AI Infra: Part 1.1, Fast and Lossless BitNet b1.58 Inference on CPUs



BitNet a4.8: 4-bit Activations for 1-bit LLMs

Activations을
Sparsification을 활용해서
4Bit로 압축해서
양자화하자 !

BitNet a4.8: 4-bit Activations for 1-bit LLMs

- Hybrid Quantization 과 Sparsification 전략을 활용.

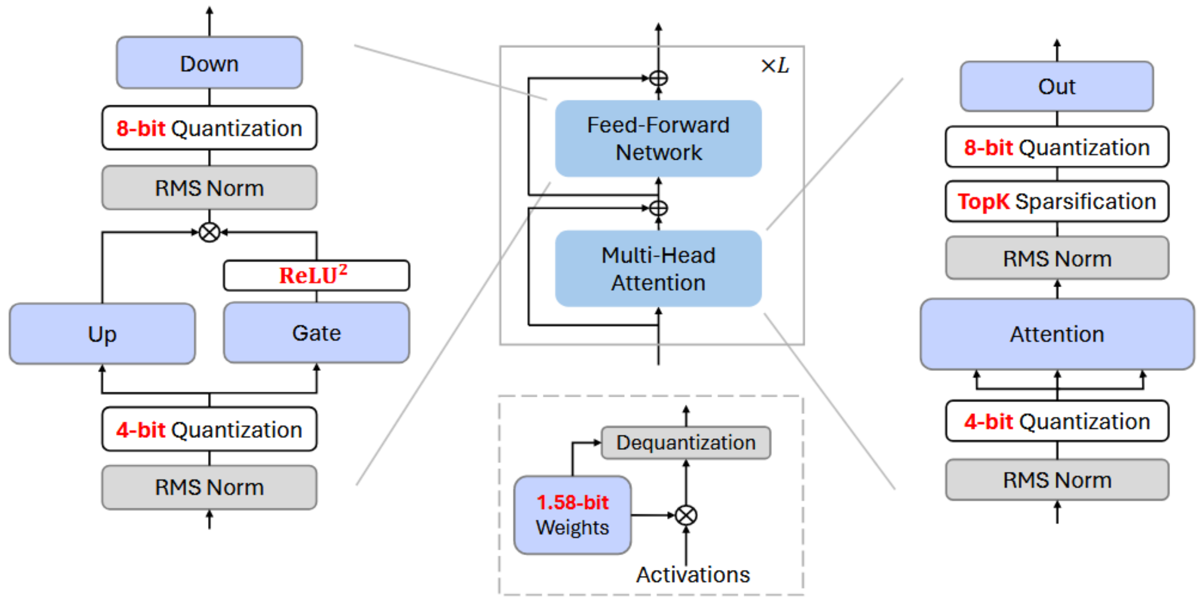


Figure 1: The overview of BitNet a4.8 with both weight and activation quantization. All the parameters are ternery (i.e., 1.58-bit as in BitNet b1.58 [MWM⁺24]). We use a hybrid quantization and sparsification strategy to deal with outlier activations in certain Transformer sub-layers.

BitNet a4.8: 4-bit Activations for 1-bit LLMs

- Sparsification은 Q-Sparse라는 이름으로 Wang과 ,Ma가 제안한 기법.(Submitted ICLR 2025) 활성화 된 Input중에서 TopK 만큼만 활성화 시키고, 역전파는 STE 쓰는 기법.

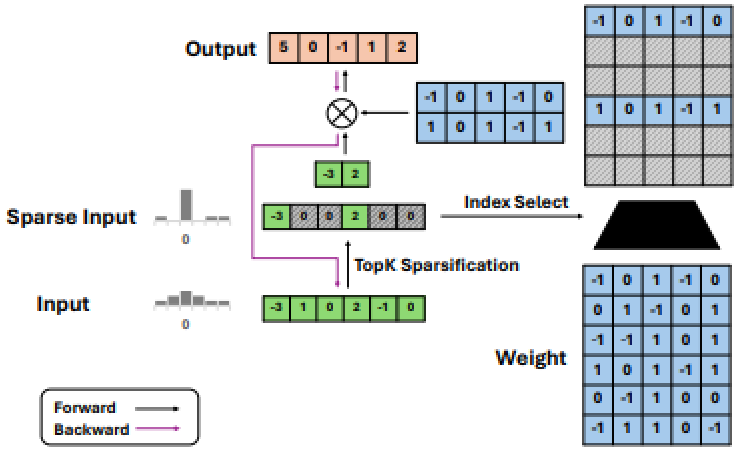


Figure 1: Q-Sparse achieves a superior inference-optimal scaling law than the dense models. It saves significant compute of matrix multiplication by top-K sparsification of the activations.

BitNet a4.8:
4-bit Activations for 1-bit LLMs

A4.8

$$\mathbf{Y} = (\mathrm{Q}_{\mathrm{INT8}}(\mathbf{X}) \odot \mathbf{M}) \cdot \mathrm{Q}_w(\mathbf{W})^T, \ \mathbf{M} = \mathrm{Top}_k(|\mathbf{X}|)$$

1.58

$$y = \widetilde{W}\widetilde{x} = \widetilde{W}\ \mathrm{Quant}(\mathrm{LN}(x)) \times \frac{\beta\gamma}{Q_b}$$

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

$$\widetilde{W} = \mathrm{RoundClip}(\frac{W}{\gamma + \epsilon}, -1, 1),$$

$$\mathrm{RoundClip}(x, a, b) = \max(a, \min(b, \mathrm{round}(x))),$$

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

BitNet a4.8: 4-bit Activations for 1-bit LLMs

- 1.58Bit Linear를 적용한 LLM에서 Projection전의 입력값의 분포는 가우시안 분포를 따르고 있음.
- 근데 특정 Activation이 0근처에 몰려 있고, 일부 극단적인 값들이 많이 관찰되고 있음.이는 full- precision에서도 마찬가지임.
- 이걸 그냥 Quantization을 수행하게 되면 오류가 커져 버리기 때문에 Quantization에 상당한 영향을 미침.
- 이에 따라서 각 공간의 분포의 특성에 맞게 다르게 처리해서 성능을 극대화 시킴.
  - 극단적인 부분은 TopK Sparsification 과 8bit Quantization을 수행.
  - 아닌 부분은 4-bit로 처리.

BitNet a4.8: 4-bit Activations for 1-bit LLMs

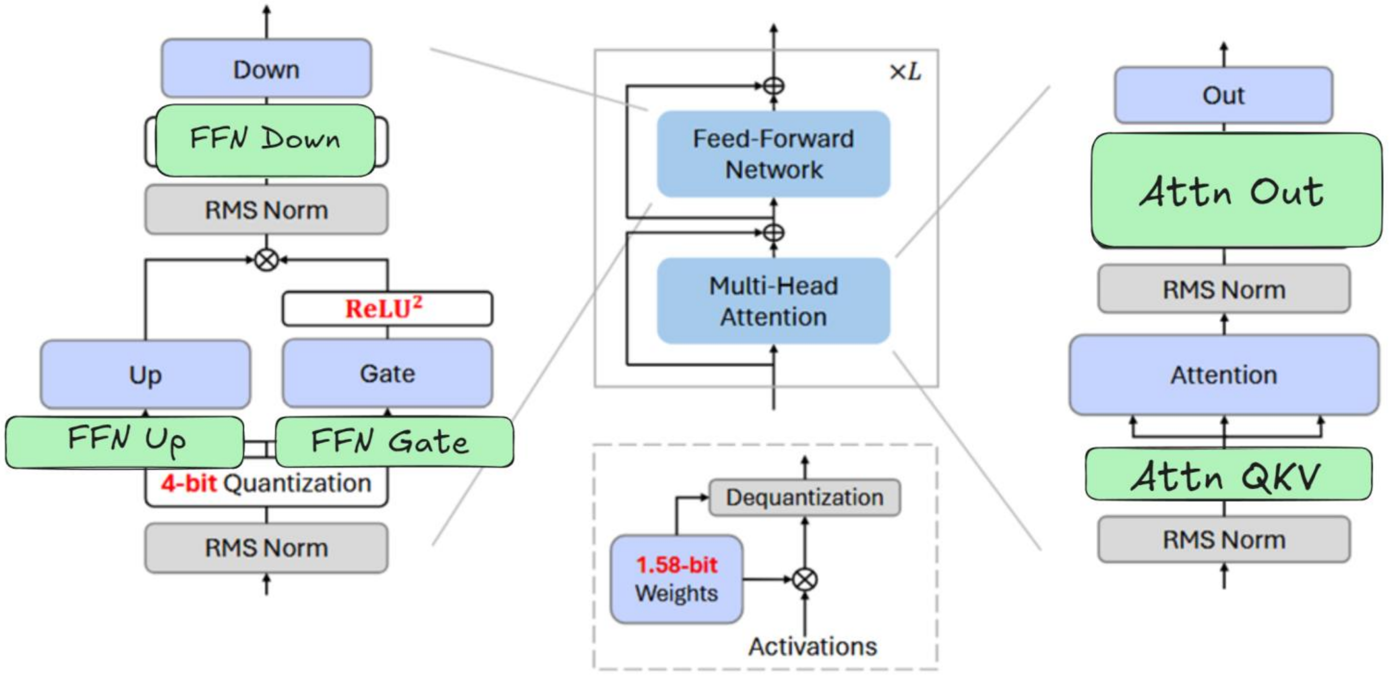


Figure 1: The overview of BitNet a4.8 with both weight and activation quantization. All the parameters are ternery (i.e., 1.58-bit as in BitNet b1.58 [MWM+24]). We use a hybrid quantization and sparsification strategy to deal with outlier activations in certain Transformer sub-layers.

BitNet a4.8: 4-bit Activations for 1-bit LLMs

양자화 전 , Activiation 분포

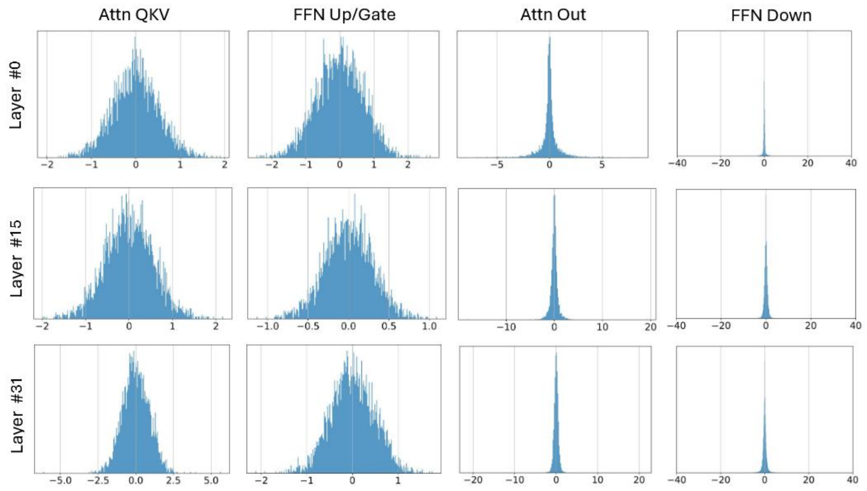


Figure 2: The distribution of the inputs to each projection. The visualization is conducted with a 7B BitNet b1.58 model on a subset of the valid set of C4. For the layers that exhibit Gaussian-like distributions, we employ 4-bit activation quantization. For the layers which distributions are sharp, we adopt Q-Sparse [WMWW24] to perform sparsification on the activations.

양자화 후 , Activiation 분포

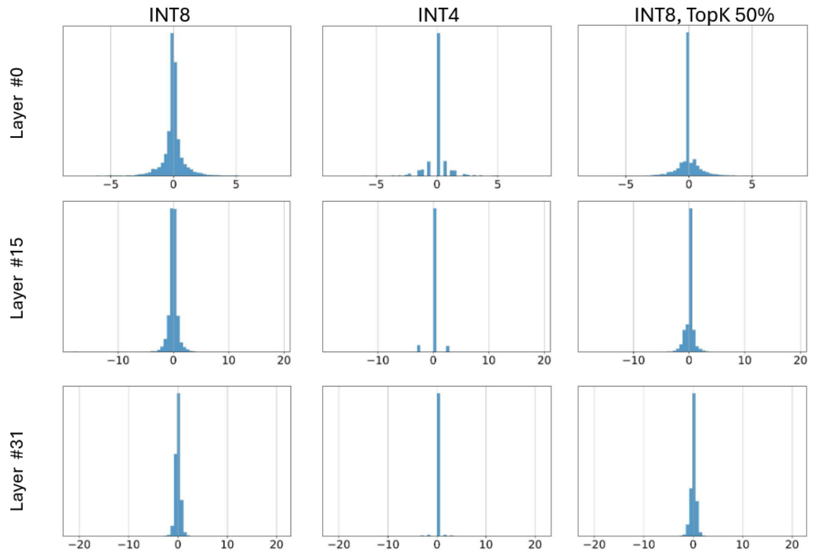


Figure 3: The distribution of the inputs to the output projection of attention with different quantization and sparsification. The visualization is conducted with a 7B BitNet b1.58 model on a subset of the valid set of C4.

BitNet a4.8: 4-bit Activations for 1-bit LLMs

Minmax Quantization

$$\mathrm{Q}_w(\mathbf{W}) = \alpha \mathrm{RoundClip}(\frac{\mathbf{W}}{\alpha + \epsilon}, -1, 1), \ \alpha = \mathrm{mean}(|\mathbf{W}|) \tag{2}$$

$$\mathrm{Q}_{\mathrm{INT8}}(\mathbf{X}) = \frac{\gamma}{127} \mathrm{RoundClip}(\frac{127}{\gamma + \epsilon}\mathbf{X}, -128, 127), \ \gamma = \max(|\mathbf{X}|) \tag{3}$$

$$\mathrm{RoundClip}(X, a, b) = \min(\max(\mathrm{round}(X), a), b) \tag{4}$$

Q-Sparse(ICLR2025) 논문(홍옥과 수명이 연구)에서 연구 되었던 것을 그대로 차용해서 가져옴

$$\mathrm{ReLU}^2\mathrm{GLU}(\mathbf{X}) = \mathbf{X}\mathbf{W}_{\mathrm{up}}^T \odot \mathrm{ReLU}^2(\mathbf{X}\mathbf{W}_{\mathrm{gate}}^T)$$

Absmean Quantization

$$\mathbf{Y} = \mathrm{Q}_{\mathrm{INT4}}(\mathbf{X}) \cdot \mathrm{Q}_w(\mathbf{W})^T \tag{6}$$

$$\mathrm{Q}_{\mathrm{INT4}}(\mathbf{X}) = \frac{\beta}{\sqrt{7}} \mathrm{RoundClip}(\frac{\sqrt{7}}{\beta + \epsilon}\mathbf{X}, -8, 7), \ \beta = \mathrm{mean}(|\mathbf{X}|) \tag{7}$$

BitNet a4.8: 4-bit Activations for 1-bit LLMs

Training

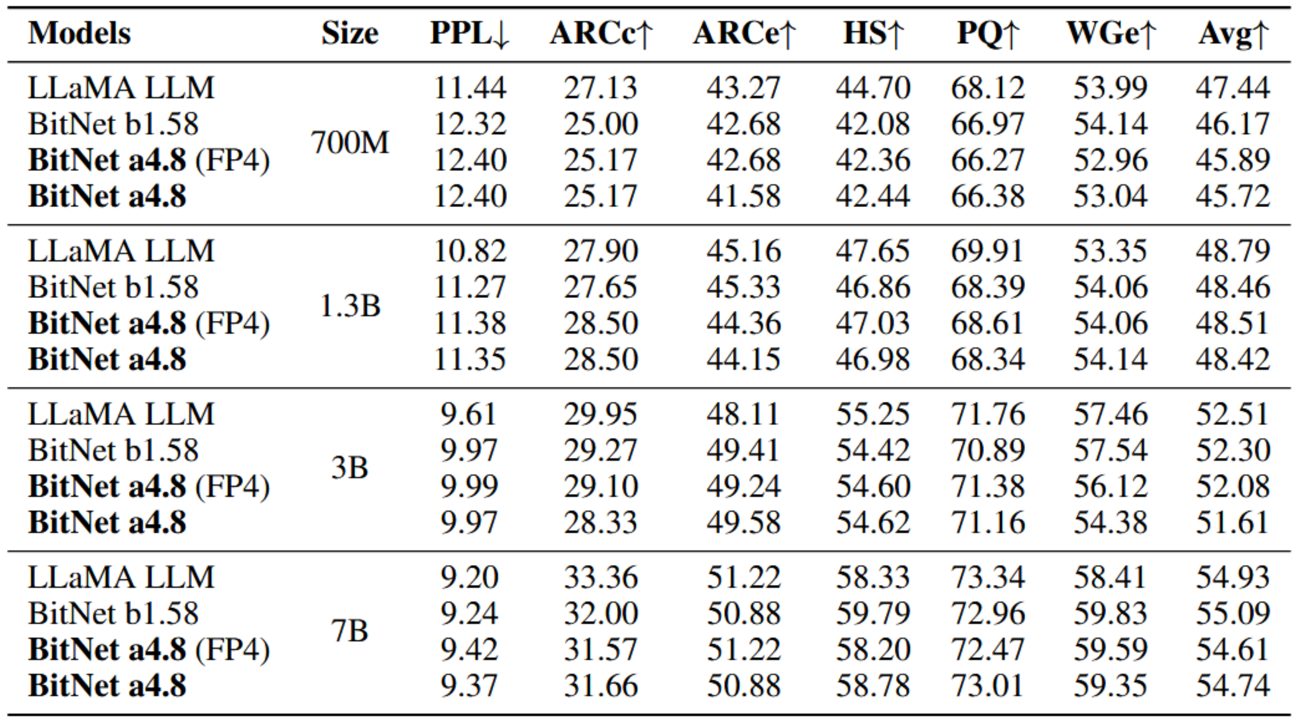
- Continue-training from BitNet b1.58
  - A4.8을 학습시에 일단 8-bit activation과 $RELU^2GLU$ 를 사용하여 학습을 하다가 hybrid quantization and sparsification를 중간에 도입하면 Loss가 빠르게 적응함.
- Floating-point quantization
  - FP4로 양자화를 최종 FFN에서 8-bit로 양자하는걸 제외하고 전부 도입해서 진행을 해봤음. (실험)
  - FP4 는 MinMax Quantizer를 사용

$$\mathrm{Q}_{\mathrm{FP4}}(\mathbf{X}) = \frac{\gamma}{2^{M+b}}\mathrm{Round}(\frac{2^{M+b}}{\gamma}\mathbf{X}),\ \gamma = 2^{\max(\lfloor\lfloor\log_2|\mathbf{X}|\rfloor+b\rfloor,1)} \tag{8}$$

$$b = \log_2(\frac{2-2^{-M}}{|\mathbf{X}|_{\max}}) + 2^E - 1 \tag{9}$$

BitNet a4.8: 4-bit Activations for 1-bit LLMs

종합적인 성능은 전부 크게 변화하기 않음을 알 수 있다.

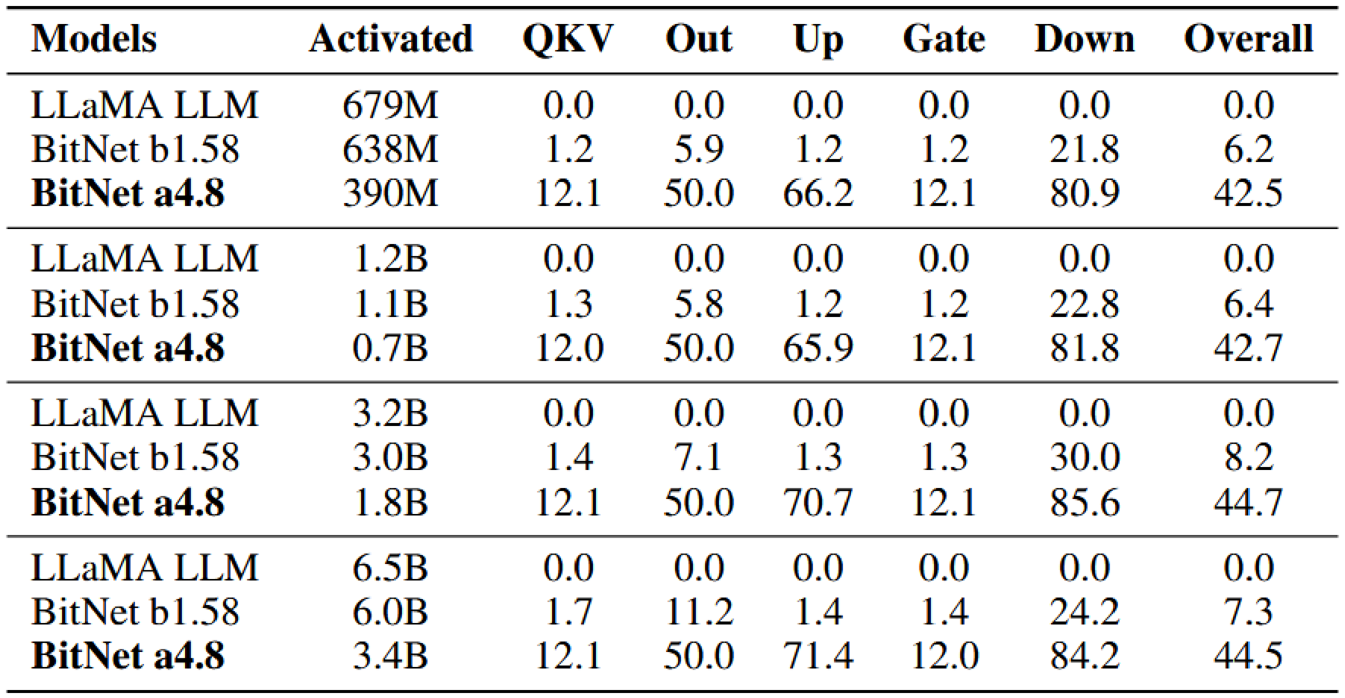


Models	Size	PPL↓	ARCc↑	ARCe↑	HS↑	PQ↑	WGe↑	Avg↑
LLaMA LLM	700M	11.44	27.13	43.27	44.70	68.12	53.99	47.44
BitNet b1.58		12.32	25.00	42.68	42.08	66.97	54.14	46.17
BitNet a4.8 (FP4)		12.40	25.17	42.68	42.36	66.27	52.96	45.89
BitNet a4.8		12.40	25.17	41.58	42.44	66.38	53.04	45.72
LLaMA LLM	1.3B	10.82	27.90	45.16	47.65	69.91	53.35	48.79
BitNet b1.58		11.27	27.65	45.33	46.86	68.39	54.06	48.46
BitNet a4.8 (FP4)		11.38	28.50	44.36	47.03	68.61	54.06	48.51
BitNet a4.8		11.35	28.50	44.15	46.98	68.34	54.14	48.42
LLaMA LLM	3B	9.61	29.95	48.11	55.25	71.76	57.46	52.51
BitNet b1.58		9.97	29.27	49.41	54.42	70.89	57.54	52.30
BitNet a4.8 (FP4)		9.99	29.10	49.24	54.60	71.38	56.12	52.08
BitNet a4.8		9.97	28.33	49.58	54.62	71.16	54.38	51.61
LLaMA LLM	7B	9.20	33.36	51.22	58.33	73.34	58.41	54.93
BitNet b1.58		9.24	32.00	50.88	59.79	72.96	59.83	55.09
BitNet a4.8 (FP4)		9.42	31.57	51.22	58.20	72.47	59.59	54.61
BitNet a4.8		9.37	31.66	50.88	58.78	73.01	59.35	54.74

Table 1: Perplexity and results of BitNet a4.8, BitNet b1.58 and LLaMA LLM on the end tasks. The standard variance of error for average scores is 1.06%.

BitNet a4.8: 4-bit Activations for 1-bit LLMs

희소화(0이 되서 연산 안해도 되는 비율)도 상당히 개선된 것을 알 수 있다.

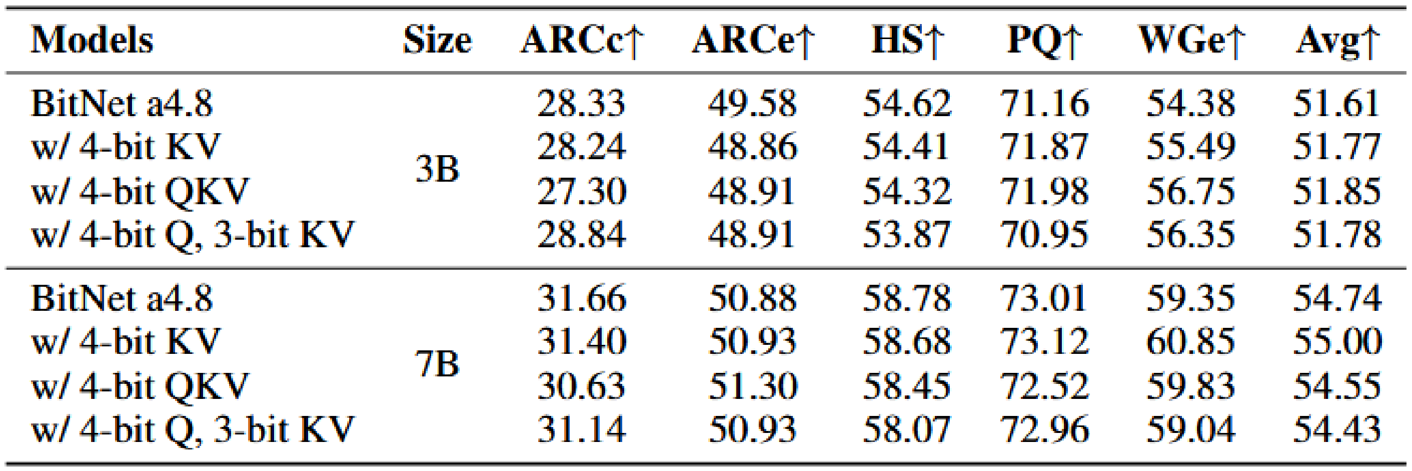


Models	Activated	QKV	Out	Up	Gate	Down	Overall
LLaMA LLM	679M	0.0	0.0	0.0	0.0	0.0	0.0
BitNet b1.58	638M	1.2	5.9	1.2	1.2	21.8	6.2
BitNet a4.8	390M	12.1	50.0	66.2	12.1	80.9	42.5
LLaMA LLM	1.2B	0.0	0.0	0.0	0.0	0.0	0.0
BitNet b1.58	1.1B	1.3	5.8	1.2	1.2	22.8	6.4
BitNet a4.8	0.7B	12.0	50.0	65.9	12.1	81.8	42.7
LLaMA LLM	3.2B	0.0	0.0	0.0	0.0	0.0	0.0
BitNet b1.58	3.0B	1.4	7.1	1.3	1.3	30.0	8.2
BitNet a4.8	1.8B	12.1	50.0	70.7	12.1	85.6	44.7
LLaMA LLM	6.5B	0.0	0.0	0.0	0.0	0.0	0.0
BitNet b1.58	6.0B	1.7	11.2	1.4	1.4	24.2	7.3
BitNet a4.8	3.4B	12.1	50.0	71.4	12.0	84.2	44.5

Table 2: Detailed sparsity of BitNet a4.8, BitNet b1.58 and LLaMA LLM on the valid set of C4.

BitNet a4.8: 4-bit Activations for 1-bit LLMs

양자화를 KV에서 3bit만 수행해도 성능이 좋은 것을 알 수 있다.



Models	Size	ARCc↑	ARCe↑	HS↑	PQ↑	WGe↑	Avg↑
BitNet a4.8	3B	28.33	49.58	54.62	71.16	54.38	51.61
w/ 4-bit KV		28.24	48.86	54.41	71.87	55.49	51.77
w/ 4-bit QKV		27.30	48.91	54.32	71.98	56.75	51.85
w/ 4-bit Q, 3-bit KV		28.84	48.91	53.87	70.95	56.35	51.78
BitNet a4.8	7B	31.66	50.88	58.78	73.01	59.35	54.74
w/ 4-bit KV		31.40	50.93	58.68	73.12	60.85	55.00
w/ 4-bit QKV		30.63	51.30	58.45	72.52	59.83	54.55
w/ 4-bit Q, 3-bit KV		31.14	50.93	58.07	72.96	59.04	54.43

Table 3: Detailed results of BitNet a4.8 with QKV states varying bit-widths on the end tasks. We reported the zero-shot accuracy of all models.

BitNet a4.8: 4-bit Activations for 1-bit LLMs

A4.8 전략을 도입하지 않고 INT4를 사용했을 경우 학습중 PPL이 터지는 것을 알 수 있다.

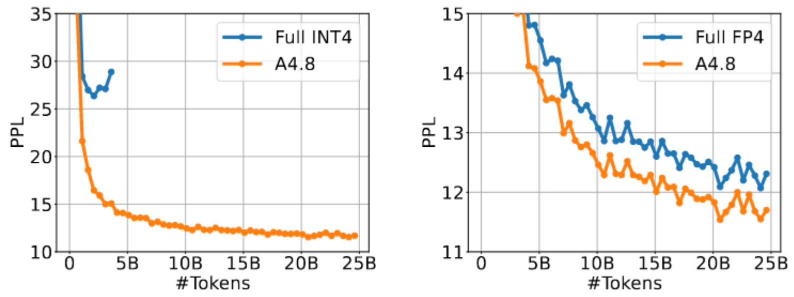


Figure 4: Ablation study on the hybrid quantization and sparsification.

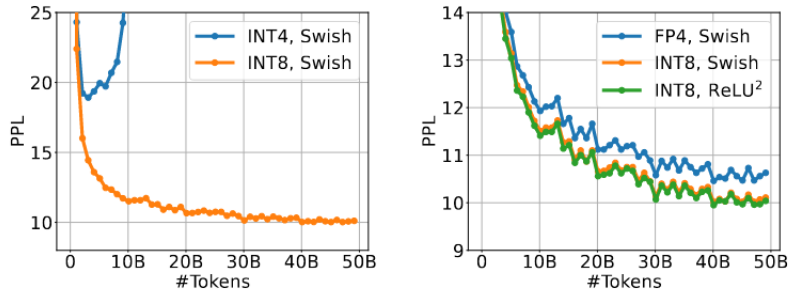


Figure 5: Ablation study on different quantization or activation function for the inputs to down projection of FFN.