

# Log<sub>b</sub>Quant: Quantizing Language Models in Logarithmic Space

Jeremias Bohn, Tizian Dippold, Mahdi Koubaa, Elias R. Wahl, and Georg Groh

School of Computation, Information and Technology, Technical University of Munich,  
Munich, Germany  
`{firstname.lastname}@tum.de`

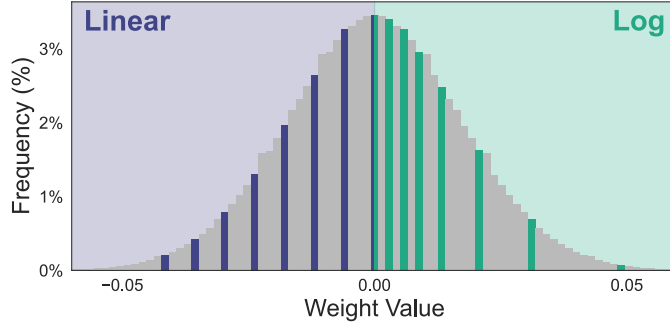
**Abstract.** Quantization has become an invaluable tool to reduce memory requirements and inference speed of modern language models, in particular to make them available for consumer setups and edge devices. While previous work has primarily focused on uniform quantization codebooks, such approaches are prone to suboptimal representations due to low-frequency high-magnitude weights. We introduce Log<sub>b</sub>Quant, a novel logarithmic quantization approach with adjustable bases, to adapt to common parameter distributions. We show that our method exhibits superior performance at 4-bit precision on several performance benchmarks compared to asymmetric linear quantization at tensor-wise granularity, while achieving moderate speedup and high memory savings, making it suitable for private use on consumer-grade GPUs.

**Keywords:** Quantization · Language Models · Resource-Constrained NLP.

## 1 Introduction

While large language models and their applications have reached the mainstream over the last few years, most models are closed source and only accessible via API and web interfaces. However, even open source models, such as Meta’s Llama [17] and Qwen’s [43] models, come with hardware requirements that often exceed common private setups, which usually comprise a single consumer-grade GPU or edge devices like smartphones. A significant bottleneck in such setups is the memory size, acting as a hard entrance barrier for local model inference. This is further exacerbated by the memory’s speed, which acts as a primary latency source for small batch sizes (which are common in a local scenario), also known as the *memory wall* [15].

While several approaches exist to reduce parameter count or improve caching, which also reduce memory load and bandwidth utilization, our work focuses on weight quantization, which provides a quick and performance-efficient measure for end users to tackle these issues. Though most quantization approaches rely on asymmetric linear codebooks, we show that quantization in logarithmic space retains performance better across several benchmarks, in particular in the low-bitwidth regime.



**Fig. 1.** Weight distribution of Llama-3.1-1B’s first layer up-projection. Left side: Negative half of 4-bit linear codebook marked in blue. Right side: Positive half of 4-bit  $\text{Log}_b\text{Quant}$  codebook marked in green.

## 2 Related Work

Approaches to reduce compute and memory requirements for machine learning models were already discussed early on [22, 19], since accelerating neural networks has been and still is a significant issue. While several approaches consider pruning [38, 28, 26] or model compression [4, 21, 24], quantization approaches, which reduce representation precision of weights and/or activations, faced widespread adoption due to their easy application and low impact on model performance.

One commonly distinguishes between *uniform quantization* [14], i.e., using equidistant codebook entries, and *non-uniform quantization*, such as our own approach,  $\text{Log}_b\text{Quant}$ . Uniform quantization can be described as a linear function  $\text{Quant}(s, z, x) = \lfloor \frac{x}{s} \rfloor - z$  with scaling factor  $s$  and zero-point  $z$  (where  $\lfloor \cdot \rfloor$  is a rounding function). Such approaches are favorable due to their arithmetic properties, but may spend too much precision to accommodate for low-frequency values (see Figure 1, left side). On the other hand, non-uniform quantization adapts better to the value distribution of modern neural networks, which commonly are densely centered around zero with few outliers [32, 34, 14, 39, 13]. Previous work which, as our own method, applies a logarithmic scale codebook mostly focused on predefined bases [32, 2, 7, 34, 35], mostly base 2 or powers thereof, which may not optimally fit distributions. On the other hand, we consider bases (and ranges) as optimizable parameters for each layer to fit our codebook to the weight distribution.

Quantization methods further differ in scope: *Weight-only quantization* reduces model parameter precision while activations are kept at full precision to reduce memory footprint and bandwidth usage [14, 11]. *Activation quantization* quantizes hidden representations [42], which becomes particularly relevant for the now common GLU-based MLP [37] models, as they project into a high dimensional space. *Weight-and-activation quantization* applies precision reduction

to both, which offers superior compression and speedups, but also introduces larger algorithmic and numerical challenges [5]. As our focus lies on consumer and edge devices where decoding of low batch sizes is memory-bound, we focus on a weight-only approach which comes with significant memory savings and preserves performance more reliably than more aggressive weight-and-activation quantization [15, 25, 11].

Lastly, to recover from quantization errors, two major calibration techniques are applied. In Quantization-Aware Training (QAT), the model is trained to unlearn the precision loss using gradient-based optimization. QAT approaches, such as Differentiable Soft Quantization [16], often re-enable superior performance but require extensive training and data and face issues with non-differentiable rounding operations in quantization functions [44]. Due to these issues, Post-Training Quantization (PTQ) methods like VPTQ [27], GPTQ [11], and GPTAQ [23] are often preferred as those can quantize models with a small calibration dataset or completely data-free [33] quickly. In our work, we use GPTQ due to its widespread use and easy adaptability to new quantization functions.

### 3 Method

We introduce Log<sub>b</sub>Quant, a novel symmetric quantization scheme in logarithmic space. Our approach is based on the observation that weights in language models are approximately normal distributed (see Figure 1), therefore we shift precision towards distributionally dense areas. Log<sub>b</sub>Quant reduces the need to operate on higher granularity levels, and instead allows tensor-level quantization with minimal performance loss.

#### 3.1 Quantization Scheme

As explained above, balancing the number of weights mapped to each entry of our codebook  $C$  is central to our approach. We therefore define one codebook per weight matrix, parameterized by base  $b \in \mathbb{R}$  and shift  $s \in \mathbb{R}$ . Consider any weight  $x \in \mathbb{R}$ , any quantized weight  $q \in \mathbb{Z}$  and any codebook bitwidth  $n \in \mathbb{N}$ . Then we define our quantization and dequantization function as

$$\text{Quant}(b, s, x) := \begin{cases} 0 & x = 0 \\ \text{sign}(x) \lfloor \log_b(|x|) + s \rfloor_{-q_{\max}}^{q_{\max}} & x \neq 0 \end{cases} \quad (1)$$

$$\text{Dequant}(b, s, q) := \begin{cases} 0 & q = 0 \\ \text{sign}(q) \cdot b^{|q|-s} & q \neq 0 \end{cases} \quad (2)$$

where  $q_{\max} := 2^{n-1} - 1$  and  $\lfloor \cdot \rfloor_a^b$  is rounding to the nearest integer with clamping. Given an effective weight range  $[-w_{\min}, -w_{\max}] \cup [w_{\min}, w_{\max}]$  of a single matrix, we can define  $b$  as

$$b := \left( \frac{w_{\max}}{w_{\min}} \right)^{\frac{1}{q_{\max}}} \quad (3)$$

which ensures  $q_{\max}$  logarithmic steps from  $w_{\min}$  to  $w_{\max}$ . To align the value  $w_{\max}$  with  $q_{\max}$  in the codebook, we eventually define  $s$  as

$$s := q_{\max} - \lfloor \log_b w_{\max} \rfloor \quad (4)$$

While these values are not necessarily optimal in terms of codebook balancing, they act as a heuristic to approximate the dynamic range of the given matrix without the need of a non-convex optimization of Equation 1.

### 3.2 Energy-Based Pruning

As seen in Figure 1, weight distribution density is generally highest around 0. However, previous works on model pruning [19, 18, 8] already established that weights of low magnitude generally contribute less to a model’s calculations and can be set to 0 with low impact on predictions. Even though exploiting unstructured sparsity is usually difficult, we can use such an approach to shift our codebook density to those parts where it actually matters. We follow a signal-energy-based pruning strategy and use this to define the effective weight range in Subsection 3.1. While we can define the maximum in a straight-forward manner as  $w_{\max} := \max_i |w_i|$ , we prune weights by defining our minimum as:

$$w_{\min} := \min \left\{ t \mid \frac{\sum_{|w_i| \leq t} w_i^2}{\sum_i w_i^2} \geq \varepsilon \right\} \quad (5)$$

This allows us to discard a fraction  $\varepsilon$  of the total energy. The approach is self-calibrating across layers: In distributions with low standard deviation, the weights close to 0 hold a significant part of the total energy, thus reducing  $w_{\min}$ , while matrices with flatter weight distribution remove larger weights. We conducted a hyperparameter search which revealed that generally  $\varepsilon = 4 \times 10^{-3}$  and  $\varepsilon = 1 \times 10^{-6}$  work best for 4-bit and 8-bit quantization, respectively.

### 3.3 Implementation

Since we use a weight-only quantization approach, we have to dequantize the weights at inference to calculate linear layer outputs. However, as the exponentiation of floating points in Equation 2 for every weight is computationally expensive, we instead rely on a lookup table (LUT) to retrieve dequantized weights directly. To this end we employ the FLUTE kernel [6], which also directly allows us to apply weight packing for 4-bit quantization. As FLUTE is designed for blockwise quantization approaches, we simulate our tensor-wise method using a group size of 64, which we chose after a hyperparameter search optimizing for speed benchmarks.

## 4 Experiments

To assess quality and efficiency of our quantization scheme, we evaluate our approach across 8 different text-only decoder-based language models from two

model families: Qwen3 and Llama-3.1/3.2. We compare the models’ performance in terms of downstream task accuracy and perplexity and its efficiency in terms of tokens throughput and peak memory usage against a standard asymmetric linear quantization scheme following the methodology of [10]. Additionally, we provide an ablation study for fixed base 2, which can be found in Section 4.4.

#### 4.1 Setup

Throughout all our experiments we use the post-training quantization approach GPTQ [11] for model calibration, implemented as a Hugging Face Transformers [41] pipeline. For our calibration data, we follow the methodology described in GPTQ by randomly (using seed 0) sampling 128 samples of 2048 tokens from WikiText-2’s [30] train split.

We apply a tensor-based weight-only quantization approach both for the asymmetric linear and our own logarithmic scheme, which we apply to the models Llama-3.1-8B, Llama-3.2-1B and -3B, as well as the Qwen3 models of size 0.6B, 1.7B, 4B, 8B, and 14B. In all cases, the embedding layer as well as the language modeling head are kept in full 16-bit precision (bfloat16), while the remaining weights were reduced to 4- or 8-bit (int8).

For performance benchmarking, we use Eleuther AI’s LM Evaluation Harness [12]. Efficiency benchmarking is conducted using PyTorch’s [1] benchmark framework.

All experiments were carried out on a single Nvidia A40 GPU with 48GB VRAM, alongside an AMD Epyc 7413 24-core CPU @2.65GHz and 512 GB DDR4 RAM.

#### 4.2 Performance Benchmarks

Downstream performance is measured in terms of accuracy on MMLU [20] (5-shot) and normalized accuracy on PIQA [3] and ARC-C [9] (both zero shot), respectively. Performance of the unquantized models can be found in Table 1. The results of the quantized models can be found in Table 2.

Language modeling performance is measured in terms of perplexity on the validation splits of WikiText-2 [30], Penn Treebank (PTB) [29] and 1000 samples

**Table 1.** Accuracy (↑) on MMLU and normalized accuracy (↑) on ARC-C and PIQA, respectively. Reported values are for the base models in BF16.

| Model        | MMLU  | ARC-C | PIQA  | Avg.  |
|--------------|-------|-------|-------|-------|
| Llama-3.2-1B | 37.84 | 37.03 | 74.65 | 49.84 |
| Llama-3.2-3B | 55.01 | 46.42 | 77.80 | 59.74 |
| Llama-3.1-8B | 63.87 | 54.86 | 81.01 | 66.58 |
| Qwen3-0.6B   | 40.09 | 33.96 | 67.63 | 47.23 |
| Qwen3-1.7B   | 55.55 | 43.09 | 72.20 | 56.95 |
| Qwen3-4B     | 68.37 | 54.01 | 75.19 | 65.86 |
| Qwen3-8B     | 72.90 | 56.48 | 77.69 | 69.11 |
| Qwen3-14B    | 77.15 | 60.15 | 79.76 | 72.35 |

**Table 2.** Accuracy ( $\uparrow$ ) on MMLU and normalized accuracy ( $\uparrow$ ) on ARC-C and PIQA, respectively. We compare asymmetric linear quantization vs. our  $\text{Log}_b\text{Quant}$  approach.

| Model |              | MMLU         |              | ARC-C        |              | PIQA         |              | Avg.         |              |
|-------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
|       |              | lin          | log          | lin          | log          | lin          | log          | lin          | log          |
| 8-bit | Llama-3.2-1B | <b>28.79</b> | 26.32        | <b>37.12</b> | 33.96        | <b>73.94</b> | 73.56        | <b>46.62</b> | 44.61        |
|       | Llama-3.2-3B | <b>54.55</b> | 48.21        | <b>45.82</b> | 42.83        | 76.33        | <b>76.50</b> | <b>58.90</b> | 55.85        |
|       | Llama-3.1-8B | <b>64.03</b> | 63.43        | <b>52.56</b> | 52.30        | 79.38        | <b>79.65</b> | <b>65.32</b> | 65.13        |
|       | Qwen3-0.6B   | <b>46.98</b> | 45.22        | <b>33.02</b> | 31.74        | <b>67.63</b> | 65.89        | <b>49.21</b> | 47.62        |
|       | Qwen3-1.7B   | <b>59.30</b> | 56.96        | <b>42.15</b> | 41.21        | <b>71.76</b> | 71.11        | <b>57.74</b> | 56.43        |
|       | Qwen3-4B     | <b>69.25</b> | 68.99        | <b>53.58</b> | 53.33        | 74.54        | <b>74.70</b> | <b>65.79</b> | 65.67        |
|       | Qwen3-8B     | 74.46        | <b>74.67</b> | <b>57.51</b> | 56.40        | 77.26        | <b>77.80</b> | <b>69.74</b> | 69.62        |
|       | Qwen3-14B    | <b>78.69</b> | 78.54        | <b>62.12</b> | 60.07        | 79.54        | <b>80.03</b> | <b>73.45</b> | 72.88        |
| 4-bit | Llama-3.2-1B | 24.12        | <b>24.78</b> | 26.79        | <b>30.97</b> | 50.60        | <b>66.70</b> | 33.84        | <b>40.82</b> |
|       | Llama-3.2-3B | 23.38        | <b>29.72</b> | 26.02        | <b>35.58</b> | 50.87        | <b>71.44</b> | 33.42        | <b>45.58</b> |
|       | Llama-3.1-8B | 25.22        | <b>45.42</b> | 25.43        | <b>39.42</b> | 51.52        | <b>74.59</b> | 34.06        | <b>53.14</b> |
|       | Qwen3-0.6B   | 24.65        | <b>34.02</b> | 26.19        | <b>27.56</b> | 52.18        | <b>62.19</b> | 34.34        | <b>41.26</b> |
|       | Qwen3-1.7B   | 24.33        | <b>47.83</b> | 26.19        | <b>32.42</b> | 50.76        | <b>66.05</b> | 33.76        | <b>48.77</b> |
|       | Qwen3-4B     | 24.70        | <b>60.09</b> | 28.07        | <b>42.83</b> | 50.22        | <b>72.80</b> | 34.33        | <b>58.57</b> |
|       | Qwen3-8B     | 25.06        | <b>66.02</b> | 27.05        | <b>44.88</b> | 50.27        | <b>74.86</b> | 34.13        | <b>61.92</b> |
|       | Qwen3-14B    | 24.68        | <b>72.97</b> | 24.83        | <b>49.15</b> | 49.29        | <b>77.75</b> | 32.93        | <b>66.62</b> |

of the C4 dataset [36]. Perplexity is calculated as an average of blocks of length 2048 without sliding window.

Performance of the unquantized models can be found in Table 3. The results of the quantized models can be found in Table 4.

For 8-bit quantization, the linear quantization approach consistently outperforms our own approach in terms of perplexity, even though the differences between the two approaches become marginal with increasing model size. For our downstream evaluation, linear still performs better for ARC-C and most of MMLU, however, differences between both approaches are low again. Both approaches tie on the PIQA dataset. However, it is worth noting that both approaches almost saturate the original 16-bit models’ performance. Further, we conducted additional experiments using an equally sized sample of the RedPajama [40] dataset for calibration on the Llama models. In those runs, our logarithmic approach outperformed asymmetric linear quantization across almost benchmarks (albeit also marginally) and average scores for both approaches were consistently slightly better (see Tables 5 and 6). For ensuring a fair comparison

**Table 3.** Perplexity ( $\downarrow$ ) on Wikitext-2, PennTreebank, and C4. Reported values are for the base models in BF16.

| Model        | WT2   | PTB   | C4    | Avg.  |
|--------------|-------|-------|-------|-------|
| Llama-3.2-1B | 10.05 | 16.78 | 13.87 | 13.57 |
| Llama-3.2-3B | 7.97  | 12.81 | 11.25 | 10.68 |
| Llama-3.1-8B | 6.52  | 10.74 | 9.52  | 8.93  |
| Qwen3-0.6B   | 22.44 | 43.08 | 30.25 | 31.92 |
| Qwen3-1.7B   | 18.00 | 32.48 | 22.83 | 24.44 |
| Qwen3-4B     | 14.64 | 24.52 | 20.32 | 19.83 |
| Qwen3-8B     | 10.32 | 16.97 | 15.56 | 14.28 |
| Qwen3-14B    | 9.16  | 14.93 | 13.92 | 12.67 |

**Table 4.** Perplexity ( $\downarrow$ ) on Wikitext-2, PennTreebank, and C4. We compare asymmetric linear quantization vs. our Log<sub>b</sub>Quant approach. Models L refer to Llama-3.1/3.2, models Q to Qwen3.

| Model |        | WT2                |              | PTB                |              | C4                 |              | Avg.               |              |
|-------|--------|--------------------|--------------|--------------------|--------------|--------------------|--------------|--------------------|--------------|
|       |        | lin                | log          | lin                | log          | lin                | log          | lin                | log          |
| 8-bit | L-1B   | <b>10.49</b>       | 15.13        | <b>17.72</b>       | 22.00        | <b>14.95</b>       | 24.11        | <b>14.39</b>       | 20.41        |
|       | L-3B   | <b>8.32</b>        | 9.46         | <b>13.35</b>       | 15.70        | <b>12.07</b>       | 14.38        | <b>11.25</b>       | 13.18        |
|       | L-8B   | <b>6.88</b>        | 6.97         | <b>11.13</b>       | 11.40        | <b>10.40</b>       | 10.60        | <b>9.47</b>        | 9.66         |
|       | Q-0.6B | <b>22.60</b>       | 24.10        | <b>43.83</b>       | 47.53        | <b>30.69</b>       | 32.25        | <b>32.37</b>       | 34.63        |
|       | Q-1.7B | <b>18.09</b>       | 20.31        | <b>32.30</b>       | 36.55        | <b>23.04</b>       | 25.11        | <b>24.48</b>       | 27.32        |
|       | Q-4B   | <b>14.69</b>       | 15.21        | <b>24.49</b>       | 25.77        | <b>20.27</b>       | 21.10        | <b>19.82</b>       | 20.69        |
|       | Q-8B   | <b>10.41</b>       | 10.57        | <b>17.17</b>       | 17.50        | <b>15.76</b>       | 15.87        | <b>14.45</b>       | 14.65        |
|       | Q-14B  | <b>9.26</b>        | 9.27         | <b>15.40</b>       | 15.60        | <b>14.20</b>       | 14.10        | <b>12.95</b>       | 12.99        |
| 4-bit | L-1B   | $1.88 \times 10^4$ | <b>18.26</b> | $4.09 \times 10^4$ | <b>32.81</b> | $3.88 \times 10^4$ | <b>30.78</b> | $3.28 \times 10^4$ | <b>27.28</b> |
|       | L-3B   | $9.63 \times 10^4$ | <b>13.33</b> | $1.38 \times 10^5$ | <b>24.61</b> | $7.46 \times 10^4$ | <b>21.46</b> | $1.03 \times 10^5$ | <b>19.80</b> |
|       | L-8B   | $1.53 \times 10^5$ | <b>10.30</b> | $1.60 \times 10^5$ | <b>18.76</b> | $1.17 \times 10^5$ | <b>17.98</b> | $1.44 \times 10^5$ | <b>15.68</b> |
|       | Q-0.6B | $5.37 \times 10^3$ | <b>31.85</b> | $3.86 \times 10^4$ | <b>65.23</b> | $1.47 \times 10^4$ | <b>45.31</b> | $1.95 \times 10^4$ | <b>47.46</b> |
|       | Q-1.7B | $2.76 \times 10^4$ | <b>22.10</b> | $8.12 \times 10^4$ | <b>43.46</b> | $3.16 \times 10^4$ | <b>29.23</b> | $4.68 \times 10^4$ | <b>31.60</b> |
|       | Q-4B   | $1.63 \times 10^3$ | <b>16.94</b> | $4.29 \times 10^3$ | <b>28.95</b> | $6.72 \times 10^3$ | <b>23.90</b> | $4.21 \times 10^3$ | <b>23.26</b> |
|       | Q-8B   | 325.18             | <b>11.82</b> | $1.84 \times 10^3$ | <b>19.63</b> | $2.11 \times 10^3$ | <b>18.00</b> | $1.42 \times 10^3$ | <b>16.48</b> |
|       | Q-14B  | 241.30             | <b>10.65</b> | $1.34 \times 10^3$ | <b>18.94</b> | $1.44 \times 10^3$ | <b>16.28</b> | $1.01 \times 10^3$ | <b>15.29</b> |

and aligning our methodology with the literature, we decided to report the scores after calibrating on WikiText-2 instead.

The more interesting effects become visible when quantizing to 4-bit representations. In our downstream benchmarks, linear quantization completely breaks the models’ reasoning capabilities, with results almost identical to random guessing. On the other hand, our own approach still performs well compared to the baseline even though performance slightly deteriorates. The perplexity results further emphasize this pattern: The linearly quantized models can no longer properly predict next tokens accurately. Performance is slightly recovered with increasing model size, but perplexity is still significantly higher than log-quantized models.

The experiments also show that our values of  $\varepsilon$  for the energy-based pruning prove robust and generalize well over different models. While more lenient pruning at 8-bit precision is sufficient, weights of low magnitude need to be removed more aggressively for 4-bit quantization as a tighter effective weight range improves approximation for small codebooks.

**Table 5.** Accuracy ( $\uparrow$ ) on MMLU and normalized accuracy ( $\uparrow$ ) on ARC-C and PIQA, respectively, after calibrating on RedPajama for 8-bit quantization. We compare asymmetric linear quantization vs. our Log<sub>b</sub>Quant approach.

| Model        | MMLU  |              | ARC-C        |              | PIQA         |              | Avg.  |              |
|--------------|-------|--------------|--------------|--------------|--------------|--------------|-------|--------------|
|              | lin   | log          | lin          | log          | lin          | log          | lin   | log          |
| Llama-3.2-1B | 34.80 | <b>37.67</b> | 36.01        | <b>37.12</b> | <b>74.86</b> | 74.54        | 48.56 | <b>49.78</b> |
| Llama-3.2-3B | 54.46 | <b>54.94</b> | <b>45.99</b> | 45.73        | 77.37        | <b>77.91</b> | 59.27 | <b>59.53</b> |
| Llama-3.1-8B | 62.29 | <b>64.11</b> | 53.24        | <b>54.52</b> | 80.52        | <b>80.79</b> | 65.35 | <b>66.47</b> |

**Table 6.** Perplexity ( $\downarrow$ ) on Wikitext-2, PennTreebank, and C4 after calibrating on RedPajama for 8-bit quantization. We compare asymmetric linear quantization vs. our  $\text{Log}_b\text{Quant}$  approach. L refers to Llama-3.1/3.2.

| Model        | WT2   |              | PTB   |              | C4    |              | Avg.  |              |
|--------------|-------|--------------|-------|--------------|-------|--------------|-------|--------------|
|              | lin   | log          | lin   | log          | lin   | log          | lin   | log          |
| Llama-3.2-1B | 10.33 | <b>10.12</b> | 17.14 | <b>16.90</b> | 14.31 | <b>13.95</b> | 13.93 | <b>13.66</b> |
| Llama-3.2-3B | 8.08  | <b>8.00</b>  | 12.97 | <b>12.86</b> | 11.45 | <b>11.29</b> | 10.83 | <b>10.72</b> |
| Llama-3.2-8B | 6.74  | <b>6.55</b>  | 10.97 | <b>10.78</b> | 9.98  | <b>9.58</b>  | 9.23  | <b>8.97</b>  |

### 4.3 Efficiency Benchmarks

To measure the efficiency of our approach, we compare the model throughput (see Table 7) and peak memory usage (see Table 8) of our approach at 4-bit compared to the baseline models (unquantized, bfloat16). In all experiments in this section, we produce exactly 128 new tokens per input prompt (input prompts being identical in all batch samples) with greedy search. Runs are repeated until wallclock times statistically stabilize, and are eventually averaged.

It is important to note that, in contrast to memory savings, speed is not the focus of our approach since dequantization at inference time introduces an overhead. However, we nevertheless achieve minor to moderate speedups, in particu-

**Table 7.** Model throughput in tokens per second ( $\uparrow$ ). We compare the original 16-bit bfloat model (full) against our 4-bit  $\text{Log}_b\text{Quant}$  model (log) for 3 different batch sizes. Models L refer to Llama-3.1/3.2, models Q to Qwen3.

| Model  | batch size 1 |       |         | batch size 32 |         |         | batch size 128 |         |         |
|--------|--------------|-------|---------|---------------|---------|---------|----------------|---------|---------|
|        | full         | log   | speedup | full          | log     | speedup | full           | log     | speedup |
| L-1B   | 78.06        | 81.74 | 1.05×   | 2435.39       | 2685.20 | 1.10×   | 9743.38        | 8225.03 | 0.84×   |
| L-3B   | 45.51        | 49.18 | 1.08×   | 1430.64       | 1565.21 | 1.09×   | 4673.76        | 3659.98 | 0.78×   |
| L-8B   | 31.23        | 40.88 | 1.31×   | 875.01        | 1231.72 | 1.41×   | 2751.18        | 1704.08 | 0.62×   |
| Q-0.6B | 39.59        | 39.83 | 1.01×   | 1225.51       | 1279.95 | 1.04×   | 4408.56        | 4965.88 | 1.13×   |
| Q-1.7B | 36.16        | 38.74 | 1.07×   | 1157.96       | 1218.55 | 1.05×   | 4521.08        | 4603.01 | 1.02×   |
| Q-4B   | 27.68        | 30.11 | 1.09×   | 904.29        | 967.06  | 1.07×   | 3364.21        | 2527.99 | 0.75×   |
| Q-8B   | 27.61        | 28.33 | 1.03×   | 785.01        | 885.33  | 1.13×   | 2478.66        | 1572.74 | 0.63×   |
| Q-14B  | 17.42        | 26.24 | 1.51×   | 494.13        | 723.44  | 1.46×   | 1614.11        | 909.88  | 0.56×   |

lar with growing model size. Similar results are observed by [10], whose approach we followed for the linear implementation. We attribute the throughput increase to the aforementioned memory wall: As weights are streamed in a compressed representation, we are no longer bound by memory bandwidth. Speedups diminish or turn into slowdowns when running at a batch size of 128. We assume this is due to excessive reads of the shared memory banks for dequantizing using LUTs. This assumption is further supported by comparing Qwen3-0.6B/1.7B against Llama-3.2-1B: While the Llama model has only 16 hidden layers, both Qwen3



models have 28 instead, implying a lower number of weights per layer and thus fewer simultaneous reads.

**Table 8.** Model peak memory usage in GB (↓). We compare the original 16-bit bfloat model (full) against our 4-bit Log<sub>b</sub>Quant model (log) for 3 different batch sizes. Models L refer to Llama-3.1/3.2, models Q to Qwen3.

| Model  | batch size 1 |       |         | batch size 32 |       |         | batch size 128 |       |         |
|--------|--------------|-------|---------|---------------|-------|---------|----------------|-------|---------|
|        | full         | log   | savings | full          | log   | savings | full           | log   | savings |
| L-1B   | 2.43         | 1.20  | 50.40%  | 2.60          | 1.37  | 47.09%  | 3.13           | 1.90  | 39.12%  |
| L-3B   | 6.30         | 2.43  | 61.44%  | 6.80          | 2.93  | 56.92%  | 8.35           | 4.48  | 46.35%  |
| L-8B   | 15.71        | 5.87  | 62.62%  | 16.28         | 6.44  | 60.44%  | 18.03          | 8.19  | 54.56%  |
| Q-0.6B | 1.19         | 0.73  | 38.64%  | 1.69          | 1.23  | 27.13%  | 3.26           | 2.80  | 14.11%  |
| Q-1.7B | 3.39         | 1.54  | 54.62%  | 3.89          | 2.04  | 47.54%  | 5.45           | 3.60  | 33.92%  |
| Q-4B   | 7.89         | 2.85  | 63.90%  | 8.52          | 3.48  | 59.14%  | 10.49          | 5.45  | 48.04%  |
| Q-8B   | 16.03        | 6.24  | 61.09%  | 16.66         | 6.87  | 58.76%  | 16.66          | 8.84  | 52.55%  |
| Q-14B  | 28.88        | 10.10 | 65.03%  | 29.58         | 10.80 | 63.49%  | 31.75          | 12.97 | 59.15%  |

Our main focus is memory efficiency, at which Log<sub>b</sub>Quant excels. Since we provide a weight-only quantization approach, memory savings decrease with batch size as more memory is required for activations and KV cache. As the unquantized embedding layers and language modeling heads contribute negatively to memory usage, savings deviate from the optimal 75% possible with full 4-bit quantization, with relatively higher values for increasing model sizes.

**Table 9.** Accuracy (↑) on MMLU and normalized accuracy (↑) on ARC-C and PIQA, respectively. We compare fixed base 2 vs. our Log<sub>b</sub>Quant approach.

| Model |            | MMLU         |              | ARC-C        |              | PIQA  |              | Avg.         |              |
|-------|------------|--------------|--------------|--------------|--------------|-------|--------------|--------------|--------------|
|       |            | fixed        | flex         | fixed        | flex         | fixed | flex         | fixed        | flex         |
| 8-bit | Llama-1B   | <b>28.36</b> | 26.32        | 32.17        | <b>33.96</b> | 70.62 | <b>73.56</b> | 43.72        | <b>44.61</b> |
|       | Llama-3B   | <b>51.85</b> | 48.21        | <b>44.37</b> | 42.83        | 76.22 | <b>76.50</b> | <b>57.48</b> | 55.85        |
|       | Llama-8B   | 58.66        | <b>63.43</b> | 48.46        | <b>52.30</b> | 78.51 | <b>79.65</b> | 61.88        | <b>65.13</b> |
|       | Qwen3-0.6B | 38.41        | <b>45.22</b> | 28.33        | <b>31.74</b> | 63.44 | <b>65.89</b> | 43.39        | <b>47.62</b> |
|       | Qwen3-1.7B | 50.80        | <b>56.96</b> | 35.49        | <b>41.21</b> | 68.61 | <b>71.11</b> | 51.63        | <b>56.43</b> |
|       | Qwen3-4B   | 65.52        | <b>68.99</b> | 48.98        | <b>53.33</b> | 73.99 | <b>74.70</b> | 62.83        | <b>65.67</b> |
|       | Qwen3-8B   | 71.56        | <b>74.67</b> | 53.84        | <b>56.40</b> | 76.39 | <b>77.80</b> | 67.26        | <b>69.62</b> |
|       | Qwen3-14B  | 77.14        | <b>78.54</b> | 59.64        | <b>60.07</b> | 79.27 | <b>80.03</b> | 72.02        | <b>72.88</b> |
| 4-bit | Llama-1B   | 24.64        | <b>24.78</b> | 25.43        | <b>30.97</b> | 51.47 | <b>66.70</b> | 33.85        | <b>40.82</b> |
|       | Llama-3B   | 24.64        | <b>29.72</b> | 26.88        | <b>35.58</b> | 51.14 | <b>71.44</b> | 34.22        | <b>45.58</b> |
|       | Llama-8B   | 26.89        | <b>45.42</b> | 27.90        | <b>39.42</b> | 50.82 | <b>74.59</b> | 35.20        | <b>53.14</b> |
|       | Qwen3-0.6B | 23.84        | <b>34.02</b> | 25.00        | <b>27.56</b> | 50.05 | <b>62.19</b> | 32.96        | <b>41.26</b> |
|       | Qwen3-1.7B | 22.73        | <b>47.83</b> | 26.02        | <b>32.42</b> | 50.65 | <b>66.05</b> | 33.13        | <b>48.77</b> |
|       | Qwen3-4B   | 24.13        | <b>60.09</b> | 23.29        | <b>42.83</b> | 49.84 | <b>72.80</b> | 32.42        | <b>58.57</b> |
|       | Qwen3-8B   | 24.43        | <b>66.02</b> | 24.32        | <b>44.88</b> | 51.36 | <b>74.86</b> | 33.37        | <b>61.92</b> |
|       | Qwen3-14B  | 24.38        | <b>72.97</b> | 24.74        | <b>49.15</b> | 50.71 | <b>77.75</b> | 33.28        | <b>66.62</b> |

#### 4.4 Ablation: Base 2

We conduct an ablation study on the effect of the flexible base  $b$ . To this end, we apply the same method as described in Section 3, except for the calculation of base  $b$ , which we fix to base 2. The results in Tables 9 and 10 show that the flexible base is significant for our approach, outperforming the fixed base in almost all experiments. In particular when quantizing to 4-bit, the fixed base models collapse like the linear approach.

**Table 10.** Perplexity ( $\downarrow$ ) on Wikitext-2, PennTreebank, and C4. We compare fixed base 2 vs. our  $\text{Log}_b\text{Quant}$  approach. Models L refer to Llama-3.1/3.2, models Q to Qwen3.

| Model |        | WT2                |              | PTB                |              | C4                 |              | Avg.               |              |
|-------|--------|--------------------|--------------|--------------------|--------------|--------------------|--------------|--------------------|--------------|
|       |        | fixed              | flex         | fixed              | flex         | fixed              | flex         | fixed              | flex         |
| 8-bit | L-1B   | <b>13.10</b>       | 15.13        | <b>22.00</b>       | <b>22.00</b> | <b>19.62</b>       | 24.11        | <b>18.24</b>       | 20.41        |
|       | L-3B   | <b>9.16</b>        | 9.46         | <b>14.78</b>       | 15.70        | <b>13.63</b>       | 14.38        | <b>12.52</b>       | 13.18        |
|       | L-8B   | 7.82               | <b>6.97</b>  | 12.92              | <b>11.40</b> | 12.33              | <b>10.60</b> | 11.02              | <b>9.66</b>  |
|       | Q-0.6B | 32.57              | <b>24.10</b> | 66.79              | <b>47.53</b> | 43.67              | <b>32.25</b> | 47.68              | <b>34.63</b> |
|       | Q-1.7B | 21.65              | <b>20.31</b> | 40.76              | <b>36.55</b> | 27.30              | <b>25.11</b> | 29.90              | <b>27.32</b> |
|       | Q-4B   | 15.88              | <b>15.21</b> | 26.17              | <b>25.77</b> | 21.99              | <b>21.10</b> | 21.35              | <b>20.69</b> |
|       | Q-8B   | 10.81              | <b>10.57</b> | 17.76              | <b>17.50</b> | 16.30              | <b>15.87</b> | 14.96              | <b>14.65</b> |
|       | Q-14B  | 9.30               | <b>9.27</b>  | <b>15.15</b>       | 15.60        | 14.21              | <b>14.10</b> | <b>12.89</b>       | 12.99        |
| 4-bit | L-1B   | $9.08 \times 10^3$ | <b>18.26</b> | $1.38 \times 10^4$ | <b>32.81</b> | $3.23 \times 10^4$ | <b>30.78</b> | $1.84 \times 10^4$ | <b>27.28</b> |
|       | L-3B   | $1.07 \times 10^4$ | <b>13.33</b> | $3.10 \times 10^4$ | <b>24.61</b> | $7.02 \times 10^4$ | <b>21.46</b> | $3.73 \times 10^4$ | <b>19.80</b> |
|       | L-8B   | $8.47 \times 10^5$ | <b>10.30</b> | $1.13 \times 10^6$ | <b>18.76</b> | $1.03 \times 10^6$ | <b>17.98</b> | $1.00 \times 10^6$ | <b>15.68</b> |
|       | Q-0.6B | $7.56 \times 10^3$ | <b>31.85</b> | $5.91 \times 10^4$ | <b>65.23</b> | $3.07 \times 10^4$ | <b>45.31</b> | $3.25 \times 10^4$ | <b>47.46</b> |
|       | Q-1.7B | $4.25 \times 10^5$ | <b>22.10</b> | $3.18 \times 10^5$ | <b>43.46</b> | $3.48 \times 10^5$ | <b>29.23</b> | $3.64 \times 10^5$ | <b>31.60</b> |
|       | Q-4B   | $1.59 \times 10^3$ | <b>16.94</b> | $4.76 \times 10^3$ | <b>28.95</b> | $4.74 \times 10^3$ | <b>23.90</b> | $3.70 \times 10^3$ | <b>23.26</b> |
|       | Q-8B   | 710.71             | <b>11.82</b> | $3.60 \times 10^3$ | <b>19.63</b> | $2.83 \times 10^3$ | <b>18.00</b> | $2.38 \times 10^3$ | <b>16.48</b> |
|       | Q-14B  | 801.20             | <b>10.65</b> | $5.08 \times 10^3$ | <b>18.94</b> | $3.48 \times 10^3$ | <b>16.28</b> | $3.12 \times 10^3$ | <b>15.29</b> |

## 5 Discussion and Limitations

Our novel quantization scheme,  $\text{Log}_b\text{Quant}$ , shows superior benchmark performance against asymmetric linear quantization at 4-bit and comes with minor to moderate speedups and significant memory savings compared to the unquantized model. In particular, it is worth noting that Qwen3-14B, quantized with our method, shows an increase in token throughput of  $1.51\times$  with a 65% memory reduction to 10.10 GB with an average accuracy reduction of about 6 percent points, making it the best model (from linear, logarithmic and no quantization) in our experiments that can be run on common consumer-grade graphics cards with 12 GB VRAM, such as the Nvidia RTX 5070.  $\text{Log}_b\text{Quant}$  performs well for small codebooks since high magnitude outlier values can be appropriately represented without skewing precision in high-density regions of the weight distributions, unlike asymmetric linear approaches.

Our results show that coarse, tensor-level quantization does not need to imply significant loss of performance at low bitwidths, but instead can show similarly

good results as less strong compression. However, logarithmic quantization at 8-bit comes with no advantage over linear approaches, where both methods almost saturate or exceed base model performance. Since the overhead of direct dequantization or significantly larger LUTs imposes a stronger constraint than linear quantization to int8, the latter approach is preferable in this scenario. Further, quantization to the FP8 format, which becomes more broadly available on accelerators these days [31], may resolve issues of both linear and Log<sub>b</sub>Quant int8 approaches in this scenario.

The speedup of Log<sub>b</sub>Quant is further theoretically limited by the free choice of bases  $b$  in weight-and-activation quantization: Since bases differ between tensors and are real-valued, the approximations of multiplication and particularly of addition become unstable when done directly in int8. This stands in contrast to weight-and-activation linear quantization schemes, which can achieve high throughput by directly calculating in int8.

## 6 Conclusion and Future Work

We introduce Log<sub>b</sub>Quant, a novel quantization scheme, and show that it performs particularly well in several benchmarks at coarse granularity settings and low bitwidth precision against asymmetric quantization. Additionally to tangible speedups, which were not the main focus of this work, significant memory savings allow us to quantize models with 14B parameters to fit into 12 GB of VRAM at inference time with low impact on performance, which is crucial for acceleration on current consumer hardware.

While our results already showcase the benefit of our method for the tested settings, generalization to even lower bitwidth representations are of particular interest. As our primary goal was to reduce memory requirements for consumer and edge devices, activation quantization was out-of-scope as its memory savings are negligible for personal use. Nevertheless, extending our approach to activations and KV cache could still be of interest. Alternatively, testing Log<sub>b</sub>Quant in combination with other methods orthogonal to ours, such as TurboQuant [45], may improve memory efficiency even further.

## References

1. Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., Chauhan, G., Chourdia, A., Constable, W., Desmaison, A., DeVito, Z., Ellison, E., Feng, W., Gong, J., Gschwind, M., Hirsh, B., Huang, S., Kalambarkar, K., Kirsch, L., Lazos, M., Lezcano, M., Liang, Y., Liang, J., Lu, Y., Luk, C.K., Maher, B., Pan, Y., Puhersch, C., Reso, M., Saroufim, M., Siraichi, M.Y., Suk, H., Zhang, S., Suo, M., Tillet, P., Zhao, X., Wang, E., Zhou, K., Zou, R., Wang, X., Mathews, A., Wen, W., Chanan, G., Wu, P., Chintala, S.: Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In: Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating

- Systems, Volume 2. p. 929–947. ASPLOS '24, Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3620665.3640366>, <https://doi.org/10.1145/3620665.3640366>
2. Berger, C.: Linear and logarithmic quantization approaches for efficient inference with deep neural networks. Tech. rep., Technical University Munich, Chair of Data Processing, Munich, Germany (2022)
  3. Bisk, Y., Zellers, R., Bras, R.L., Gao, J., Choi, Y.: Piqa: Reasoning about physical commonsense in natural language (2019), <https://arxiv.org/abs/1911.11641>
  4. Bohn, J., Mrozinski, F., Groh, G.: Adaptive parameter compression for language models. In: Findings of the Association for Computational Linguistics: NAACL 2025. pp. 7269–7286 (2025)
  5. Bondarenko, Y., Nagel, M., Blankevoort, T.: Understanding and overcoming the challenges of efficient transformer quantization (2021), <https://arxiv.org/abs/2109.12948>
  6. Brüggemann, A., Hundt, R., Schneider, T., Suresh, A., Yalame, H.: FLUTE: Fast and secure lookup table evaluations (full version), <https://eprint.iacr.org/2023/499>, publication info: Published elsewhere. IEEE Security & Privacy 2023
  7. Cai, J., Takemoto, M., Nakajo, H.: A deep look into logarithmic quantization of model parameters in neural networks. In: Proceedings of the 10th International Conference on Advances in Information Technology (IAIT). pp. 1–8. ACM (2018)
  8. Cheng, H., Zhang, M., Shi, J.Q.: A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. IEEE Transactions on Pattern Analysis and Machine Intelligence **46**(12), 10558–10578 (2024)
  9. Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., Tafjord, O.: Think you have solved question answering? try arc, the ai2 reasoning challenge (2018), <https://arxiv.org/abs/1803.05457>
  10. Dettmers, T., Lewis, M., Belkada, Y., Zettlemoyer, L.: LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale (2022), <https://arxiv.org/abs/2208.07339>
  11. Frantar, E., Ashkboos, S., Hoefler, T., Alistarh, D.: Gptq: Accurate post-training quantization for generative pre-trained transformers (2023), <https://arxiv.org/abs/2210.17323>
  12. Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noach, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Suta-wika, L., Tang, E., Thite, A., Wang, B., Wang, K., Zou, A.: The language model evaluation harness (07 2024). <https://doi.org/10.5281/zenodo.12608602>, <https://zenodo.org/records/12608602>
  13. Geng, X., Liu, S., Wang, H., Han, J., Jiang, H.: Lookup table refactoring: Towards efficient logarithmic number system addition for large language models. In: 2025 Design, Automation & Test in Europe Conference (DATE). pp. 1–7 (2025). <https://doi.org/10.23919/DATE64628.2025.10993215>
  14. Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M.W., Keutzer, K.: A survey of quantization methods for efficient neural network inference. <https://doi.org/10.48550/arXiv.2103.13630>, <http://arxiv.org/abs/2103.13630>
  15. Gholami, A., Yao, Z., Kim, S., Hooper, C., Mahoney, M.W., Keutzer, K.: AI and memory wall. IEEE Micro **44**(3), 33–39 (2024). <https://doi.org/10.1109/MM.2024.3373763>
  16. Gong, R., Liu, X., Jiang, S., Li, T., Hu, P., Lin, J., Yu, F., Yan, J.: Differentiable soft quantization: Bridging full-precision and low-bit neural networks. <https://doi.org/10.48550/arXiv.1908.05033>, <http://arxiv.org/abs/1908.05033>

17. Grattafiori, A., et al.: The Llama 3 Herd of Models (2024), <https://arxiv.org/abs/2407.21783>
18. Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural networks. In: Advances in Neural Information Processing Systems. vol. 28 (2015)
19. Hassibi, B., Stork, D.G.: Second order derivatives for network pruning: Optimal Brain Surgeon. In: Advances in Neural Information Processing Systems. vol. 5, pp. 164–171. Morgan Kaufmann (1992)
20. Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., Steinhardt, J.: Measuring massive multitask language understanding (2021), <https://arxiv.org/abs/2009.03300>
21. Hsu, Y.C., Hua, T., Chang, S., Lou, Q., Shen, Y., Jin, H.: Language model compression with weighted low-rank factorization. In: International Conference on Learning Representations (2022)
22. LeCun, Y., Denker, J., Solla, S.: Optimal brain damage. Advances in neural information processing systems **2** (1989)
23. Li, Y., Yin, R., Lee, D., Xiao, S., Panda, P.: GPTAQ: Efficient finetuning-free quantization for asymmetric calibration. In: International Conference on Machine Learning. pp. 36690–36706. PMLR (2025)
24. Lin, C.H., Gao, S., Smith, J., Patel, A., Tuli, S., Shen, Y., Jin, H., Hsu, Y.C.: MoDeGPT: Modular decomposition for large language model compression. In: International Conference on Learning Representations. vol. 2025, pp. 101355–101390 (2025)
25. Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.M., Wang, W.C., Xiao, G., Dang, X., Gan, C., Han, S.: Awq: Activation-aware weight quantization for llm compression and acceleration (2024), <https://arxiv.org/abs/2306.00978>
26. Ling, G., Wang, Z., Yan, Y., Liu, Q.: SlimGPT: Layer-wise structured pruning for large language models. Advances in Neural Information Processing Systems **37**, 107112–107137 (2024)
27. Liu, Y., Wen, J., Wang, Y., Ye, S., Zhang, L.L., Cao, T., Li, C., Yang, M.: VPTQ: Extreme low-bit vector post-training quantization for large language models. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. pp. 8181–8196 (2024)
28. Ma, X., Fang, G., Wang, X.: Llm-pruner: On the structural pruning of large language models. Advances in neural information processing systems **36**, 21702–21720 (2023)
29. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of english: The penn treebank. Computational Linguistics **19**(2), 313–330 (1993)
30. Merity, S., Xiong, C., Bradbury, J., Socher, R.: Pointer sentinel mixture models (2016), <https://arxiv.org/abs/1609.07843>
31. Micikevicius, P., Stosic, D., Burgess, N., Cornea, M., Dubey, P., Grisenthwaite, R., Ha, S., Heinecke, A., Judd, P., Kamalu, J., et al.: FP8 formats for deep learning. arXiv preprint arXiv:2209.05433 (2022)
32. Miyashita, D., Lee, E.H., Murmann, B.: Convolutional neural networks using logarithmic data representation. arXiv preprint arXiv:1603.01025 (2016)
33. Nagel, M., Baalen, M.v., Blankevoort, T., Welling, M.: Data-free quantization through weight equalization and bias correction. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 1325–1334 (2019)

34. Pospieszny, P., Mormul, W., Szyndler, K., Kumar, S.: ADALog: Adaptive unsupervised anomaly detection in logs with self-attention masked language model. <https://doi.org/10.48550/arXiv.2505.13496>, <http://arxiv.org/abs/2505.13496>
35. Przewlocka-Rus, D., Sarwar, S.S., Sumbul, H.E., Li, Y., Salvo, B.D.: Power-of-two quantization for low bitwidth and hardware compliant neural networks. <https://doi.org/10.48550/arXiv.2203.05025>, <http://arxiv.org/abs/2203.05025>
36. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research* **21**, 140:1–140:67 (2020)
37. Shazeer, N.: GLU variants improve transformer. arXiv preprint arXiv:2002.05202 (2020)
38. Sun, M., Liu, Z., Bair, A., Kolter, Z.: A simple and effective pruning approach for large language models. In: *International Conference on Learning Representations*. vol. 2024, pp. 4942–4964 (2024)
39. Vogel, S., Liang, M., Guntoro, A., Stechele, W., Ascheid, G.: Efficient hardware acceleration of CNNs using logarithmic data representation with arbitrary log-base. In: *Proceedings of the International Conference on Computer-Aided Design*. pp. 1–8. ACM (2018). <https://doi.org/10.1145/3240765.3240803>, <https://dl.acm.org/doi/10.1145/3240765.3240803>
40. Weber, M., Fu, D., Anthony, Q., Oren, Y., Adams, S., Alexandrov, A., Lyu, X., Nguyen, H., Yao, X., Adams, V., Athiwaratkun, B., Chalamala, R., Chen, K., Ryabinin, M., Dao, T., Liang, P., Ré, C., Rish, I., Zhang, C.: RedPajama: an open dataset for training large language models. <https://doi.org/10.48550/arXiv.2411.12372>, <http://arxiv.org/abs/2411.12372>
41. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., Rush, A.: Transformers: State-of-the-art natural language processing. In: Liu, Q., Schlangen, D. (eds.) *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. pp. 38–45. Association for Computational Linguistics, Online (Oct 2020). <https://doi.org/10.18653/v1/2020.emnlp-demos.6>, <https://aclanthology.org/2020.emnlp-demos.6/>
42. Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., Han, S.: Smoothquant: Accurate and efficient post-training quantization for large language models (2024), <https://arxiv.org/abs/2211.10438>
43. Yang, A., et al.: Qwen3 Technical Report (2025), <https://arxiv.org/abs/2505.09388>
44. Yin, P., Lyu, J., Zhang, S., Osher, S., Qi, Y., Xin, J.: Understanding straight-through estimator in training activation quantized neural nets. arXiv preprint arXiv:1903.05662 (2019)
45. Zandieh, A., Daliri, M., Hadian, M., Mirrokni, V.: TurboQuant: Online vector quantization with near-optimal distortion rate. arXiv preprint arXiv:2504.19874 (2025)