1 Bit is All You Need: Large Classification Models For Small Computers

Maxwell Bernstein
High Definition Automatic Differentiation,
Palo Alto, CA, USA

F.D.C. Willard
Michigan State University, East Lansing, MI,
USA

Abstract

Recent research, such as BitNet and BitNet 1.58b, is paving the way for a new era of 1-bit Large Language Models (LLMs). Previous work introduces so-called “1-bit” models, which end up still being quite large. In this work, we introduce a true 1-bit machine learning model variant, namely BitNet b1.00, in which every single parameter (or weight) of the model is binary {0, 1} and there is precisely one bit. It only slightly underperforms the full-precision (i.e., FP16 or BF16) Transformer LLM and BitNet 1.58b with the same training tokens in end-task classification performance, while being significantly more cost-effective in terms of latency, memory, throughput, and energy consumption. More profoundly, the 1.00-bit model defines a new scaling law and recipe for training new generations of ML models 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 models.

1 Introduction

The current evolution of machine learning, “deep learning”, produces incredible results with only a handful of $100,000 USD datacenter GPUs. The problem with this arm of research is that as the model grows in size, so too do the hardware needs: large language models are limited both by raw floating-point compute performance and also by memory (and memory bandwidth). Recent research in large language models (LLMs) indicate that this may be due to the way such models compress information [1]. Large language models even beat the state of the art in compression algorithms such as PNG and FLAC ([2]) by 58.5% and 30.3%, respectively [1].

This compression ratio is enormous: considering the massive amounts of data pumped through a model in the training process, the fact that the weights are “only” 340 MB large in memory is, frankly, incredible. In this paper, we introduce a radical improvement to the compression ratio. We accomplish this by reducing the weights down to one bit and quantizing input data to one bit. This allows us to run these classification machines on commodity hardware, such as a consumer laptop, a single board computer such as a Raspberry Pi, or even microcontrollers, such as an ESP32. Now every device in the house can answer image and text classification questions near instantly.

2 The Era of 1-bit ML models

2.1 BitNet

The original BitNet paper [3] quantizes a model to use only a restricted range of values for weights. The follow-on BitNet paper [4] introduces the additional limitation of binary weights: {0, 1}, or 1 bit per weight. This is a significant leap forward in storage space and compute over traditional FP64, FP32, and other full floating point weights while still maintaining decent inference.

2.2 BitNet b1.58

BitNet b1.58 [5] has $\log_2(3)$~1.58 bits **per weight** and still aims to create a 7B, 14B, and bigger models. Their improvement over the original BitNet is adding a -1 weight, which improves evaluation performance while taking "only" one more bit per weight (since fractional bits do not exist). They also train quantized instead of quantizing ex post facto.

While the reduction in the constant factor for storage and forward pass computation is impressive, it is still $O(n)$ time.

2.3 BitNet b1.00 (ours)

BitLinear? More like BitConstant. In this paper, we optimize the hell out of traditional "linear" layers, which (coincidentally) operate in linear time complexity. Previous work ([4], [5]) brought the constant factor down while still staying at $O(n)$ time complexity. In this work, we bend the curve to $O(1)$ (well, really it's $O(\text{input})$, as we still need to quantize input at inference time).

3 BitNet b1.00 Model Training

Most model training requires hundreds of millions, if not billions, of pieces of sample data to converge. Some training does not even converge at all. Our model improves on both: we can detect convergence very early in the training process and stop much sooner. This cuts down on waste and also allows us to iterate faster. As for the divergence: because we only use one bit, the model is guaranteed to converge by definition. The binarization $\widetilde{W}$ of the weights $W \in \mathbb{R}^{\{n \times m\}}$ can be formulated as:

$$\widetilde{W} = \begin{cases} 0 \text{ if } W_{\text{ij}} \leq 0 \\ 1 \text{ if } W_{\text{ij}} > 0 \end{cases} \tag{1}$$

This is different from [4], which uses a more complex binarization but results in more bits. Compared to previous work, we can implement this binarization in a single instruction on most modern CPUs. We do not even need to use GPU acceleration, which is a significant improvement over previous work. For example, on Intel x86_64, assuming a 64-bit floating point number,:

$$\widetilde{W} = \text{x86.ucomisd}\left(W_{\text{ij}}, \text{zero reg}\right) \tag{2}$$

Where zero reg is a reserved floating-point register. We manage to reduce the number of bits to 1. These improvements are previously unheard of and were not thought possible until now. Training can be done trivially on any desktop computer as long as it implements either a multiplication operation (for example, `fmul` on Intel x86) or a bitwise and operation (for example, `and` on x86).

4 Implementation and evaluation

We present Python implementations of the two possible classification models. These implementations very intentionally do not use common machine learning libraries such as PyTorch [6] or JAX [7]. While some level of popularity ensures that models using those tools are well understood, they are too heavyweight for this project. For that reason, we implement all of our own infrastructure.

We would also like to note that while Python makes for an excellent prototyping language and scientific computing *lingua franca*, it should **not** be used in performance-critical contexts; for those, we recommend a careful transcription to C or CUDA.

```
def linear(l, r):
  return l * r

def variant_a(inp):
  bit = 0
  inp = quantize(inp)
  return linear(inp, bit)

def variant_b(inp):
  bit = 1
  inp = quantize(inp)
  return linear(inp, bit)
```

Listing 1: Note the subtle distinction between the two model implementations: while they both use one bit, in variant A the bit is 0 while in variant B the bit is 1.

Last, note that we do not provide a quantization function in this paper; this is context-dependent and also left as an exercise to the reader. We do, however, provide one in our C implementation of the model. The only requirement is that the `quantize` function return one bit, `0` or `1`.

4.1 Results

In order to evaluate the performance of our model, we adapt the model from [8] to classify images as either hot dogs or legs. We implement the model for ARM and then link to it from an Android application for mobile use. When we point the phone camera at the legs of the authors, the phone clearly says "legs" (correct). When we point the phone camera at a hot dog, the phone still says "legs" (incorrect). We theorize this error is due to the hot dog being vegetarian. Unfortunately, our experimental setup requires that the food be vegetarian for personal reasons, so we cannot measure error on normal ("meat") hot dogs.

4.2 Inference latency

To give an idea for the kind of latency improvements this work achieves, we compare with BitNet b1.58 and Mistral 7B on Ollama on consumer hardware. As a benchmark, we classify Hot Dogs or Legs over 100,000 images from iStockPhoto. While Mistral took 12 minutes and 47 seconds, and BitNet b1.58 took 6 minutes and 23 seconds, our model takes only several milliseconds. Most of this time is spent iterating over the directory listing and not actually in inference.

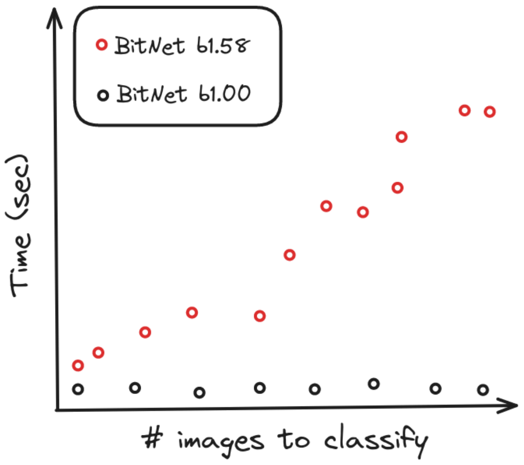

Figure 1: Lower is better. A comparison clearly showing that we can do inference faster. We lost the granular data for the Mistral comparison but trust us, it was the slowest.

5 Open source weights

Due to the lightweight nature of the model and ease of training, we are able to provide the weights with an open license. We provide the weights in GGUF. We also provide a `.c` file for use in embedded C code. To see the weights, please visit our website at bernsteinbear.com/bitnet-b100/weights.html.

6 Discussion and Future Work

We would like to continue our research on Hotdogs or Legs using BitNet b1.00 with actual hot dogs. This requires ethics board approval and also restraining Max, because he does not like the smell.

We would also like to experiment with introducing high latency into machine learning software development. Interested parties can contact the authors and pay us to Just Stop.

Acknowledgements

Thanks to Vassil Mladenov for trolling assistance.

References

[1] G. Delétang *et al.*, "Language Modeling Is Compression." 2024.

[2] M. van Beurden and A. Weaver, "Free Lossless Audio Codec," Internet Engineering Task Force, Jan. 2024. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-cellar-flac/14/

[3] A. Raghavan, M. Amer, S. Chai, and G. Taylor, "BitNet: Bit-Regularized Deep Neural Networks." 2018.

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

[5] S. Ma *et al.*, "The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits." 2024.

[6] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library." 2019.

[7] J. Bradbury *et al.*, "JAX: composable transformations of Python+NumPy programs." [Online]. Available: http://github.com/google/jax

[8] A. Fakhrou, J. Kunhoth, and S. Al Maadeed, "Smartphone-based food recognition system using multiple deep CNN models," *Multimed. Tools Appl.*, vol. 80, no. 21–23, pp. 33011–33032, Sep. 2021.

[9] "TinyGo: Go compiler for small places." [Online]. Available: https://tinygo.org/

[10] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952, doi: 10.1109/JRPROC.1952.273898.

[11] I. E. Richardson, *The H.264 Advanced Video Compression Standard*, 2nd ed. Wiley Publishing, 2010.