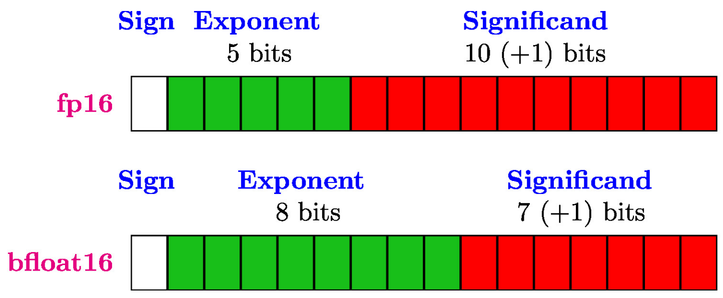
What Is Bfloat16 Arithmetic?

Nicholas J. Higham[*]

June 2, 2020

Bfloat16 is a floating-point number format proposed by Google. The name stands for "Brain Floating Point Format" and it originates from the Google Brain artificial intelligence research group at Google.

Bfloat16 is a 16-bit, base 2 storage format that allocates 8 bits for the significand and 8 bits for the exponent. It contrasts with the IEEE fp16 (half precision) format, which allocates 11 bits for the significand but only 5 bits for the exponent. In both cases the implicit leading bit of the significand is not stored, hence the "+1" in this diagram:



The motivation for bfloat16, with its large exponent range, was that "neural networks are far more sensitive to the size of the exponent than that of the mantissa" (Wang and Kanwar, 2019).

Bfloat16 uses the same number of bits for the exponent as the IEEE fp32 (single precision) format. This makes conversion between fp32 and bfloat16 easy (the exponent is kept unchanged and the significand is rounded or truncated from 24 bits to 8) and the possibility of overflow in the conversion is largely avoided. Overflow can still happen, though (depending on the rounding mode): the significand of fp32 is longer, so the largest fp32 number exceeds the largest bfloat16 number, as can be seen in the following table. Here, the precision of the arithmetic is measured by the unit roundoff, which is 2^{-t}, where t is the number of bits in the significand.

	Bits (significand, exponent)	Unit roundoff	Min positive subnormal	Min positive normalized	Max finite
bfloat16	(8, 8)	3.91×10^{-3}	9.18×10^{-41}	1.18×10^{-38}	3.39×10^{38}
fp16	(11, 5)	4.88×10^{-4}	5.96×10^{-8}	6.10×10^{-5}	6.55×10^{4}
fp32	(24, 8)	5.96×10^{-8}	1.40×10^{-45}	1.18×10^{-38}	3.40×10^{38}
fp64	(53, 11)	1.11×10^{-16}	4.94×10^{-324}	2.22×10^{-308}	1.80×10^{308}

[*]Department of Mathematics, University of Manchester, Manchester, M13 9PL, UK (nick.higham@manchester.ac.uk).

Note that although the table shows the minimum positive subnormal number for bfloat16, current implementations of bfloat16 do not appear to support subnormal numbers (this is not always clear from the documentation).

As the unit roundoff values in the table show, bfloat16 numbers have the equivalent of about three decimal digits of precision, which is very low compared with the eight and sixteen digits, respectively, of fp32 and fp64 (double precision).

The next table gives the number of numbers in the bfloat16, fp16, and fp32 systems. It shows that the bfloat16 number system is very small compared with fp32, containing only about 65,000 numbers.

	Normalized	Subnormal
bfloat16	6.5×10^4	2.5×10^2
fp16	6.1×10^4	2.0×10^3
fp32	4.3×10^9	1.7×10^7

The spacing of the bfloat16 numbers is large far from 1. For example, 65280, 65536, and 66048 are three consecutive bfloat16 numbers.

At the time of writing, bfloat16 is available, or announced, on four platforms or architectures.

- The Google Tensor Processing Units (TPUs, versions 2 and 3) use bfloat16 within the matrix multiplication units. In version 3 of the TPU the matrix multiplication units carry out the multiplication of 128-by-128 matrices.
- The NVIDIA A100 GPU, based on the NVIDIA Ampere architecture, supports bfloat16 in its tensor cores through block fused multiply-adds (FMAs) $C + A * B$ with 8-by-8 A and 8-by-4 B.
- Intel has published a specification for bfloat16 and how it intends to implement it in hardware. The specification includes an FMA unit that takes as input two bfloat16 numbers a and b and an fp32 number c and computes $c + a * b$ at fp32 precision, returning an fp32 number.
- The Arm A64 instruction set supports bfloat16. In particular, it includes a block FMA $C + A * B$ with 2-by-4 A and 4-by-2 B.

The pros and cons of bfloat16 arithmetic versus IEEE fp16 arithmetic are

- bfloat16 has about one less (roughly three versus four) digit of equivalent decimal precision than fp16,
- bfloat16 has a *much* wider range than fp16, and
- current bfloat16 implementations do not support subnormal numbers, while fp16 does.

If you wish to experiment with bfloat16 but do not have access to hardware that supports it you will need to simulate it. In MATLAB this can be done with the chop function written by me and Srikara Pranesh.

References

This is a minimal set of references, which contain further useful references within.

- Arm A64 Instruction Set Architecture Armv8, for Armv8-A Architecture Profile, ARM Limited, 2019.
- Intel Corporation, BFLOAT16—Hardware Numerics Definition, 2018.

- IEEE Standard for Floating-Point Arithmetic, IEEE Std 754-2019 (Revision of IEEE 754-2008), The Institute of Electrical and Electronics Engineers, New York, 2019.
- NVIDIA Corporation, NVIDIA A100 Tensor Core GPU Architecture, 2020.

Related Blog Posts

- BFloat16 Processing for Neural Networks on Armv8-A by Nigel Stephens (2019)
- BFloat16: The Secret To High Performance on Cloud TPUs by Shibo Wang and Pankaj Kanwar (2019)
- A Multiprecision World (2017)
- Half Precision Arithmetic: fp16 Versus bfloat16 (2018)
- The Rise of Mixed Precision Arithmetic (2015)
- Simulating Low Precision Floating-Point Arithmetics in MATLAB (2020)
- What Is Floating-Point Arithmetic? (2020)
- What Is IEEE Standard Arithmetic? (2020)

This article is part of the "What Is" series, available from `https://nhigham.com/category/what-is` and in PDF form from the GitHub repository `https://github.com/higham/what-is`.