Tensor precision / Data types

Quarter, Half and Mixed Precision

fp16

bf16

mixed fp16

mixed bf16

fp8

General OPs

`LayerNorm`-like operations must not do their work in half-precision, or they may lose a lot of data. Therefore when these operations are implemented correctly they do efficient internal work in fp32 and then their outputs are downcast to half-precision. Very often it's just the accumulation that is done in fp32, since adding up half-precision numbers is very lossy.

example:

Reduction collectives

fp16: ok to do in fp16 if loss scaling is in place

bf16: only ok in fp32

Gradient accumulation

best done in fp32 for both, but definitely for bf16

Optimizer step / Vanishing gradients

when adding a tiny gradient to a large number, that addition is often nullified

fp32 master weights and fp32 optim states

bf16 master weights and optim states can be done when using Kahan Summation and/or Stochastic rounding

Using fp16-pretrained model in bf16 regime

usually fails

Using bf16-pretrained model in fp16 regime

will lose some performance on conversion, but should work - best to finetune a bit

FP8

Main paper: FP8 Formats for Deep Learning