Be Like Water: Adaptive Floating Point for Machine Learning

Thomas Y. Yeh[1], Maxwell R. Sterner[12], Zerlina Lai[2], Brandon Y. Chuang[3], Alexander Ihler[4]

[1] Pomona College [2] Occidental College [3] UC Santa Cruz [4] UC Irvine

Pomona College 1887

UCI

Motivation for AFP

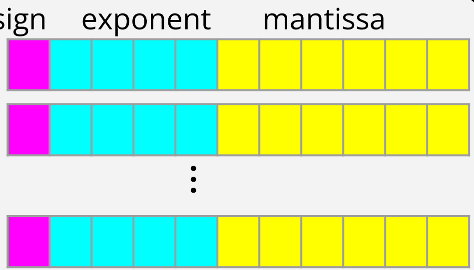
- DNNs continue to scale parameters and compute demands exponentially.

- Prior compute efficient data formats require either the use of scaling factors or selective application to certain layers.

- Can we design an efficient representation, applicable to all ML model data, which does not require the models to adapt to the representation?

Prior Quantization Formats

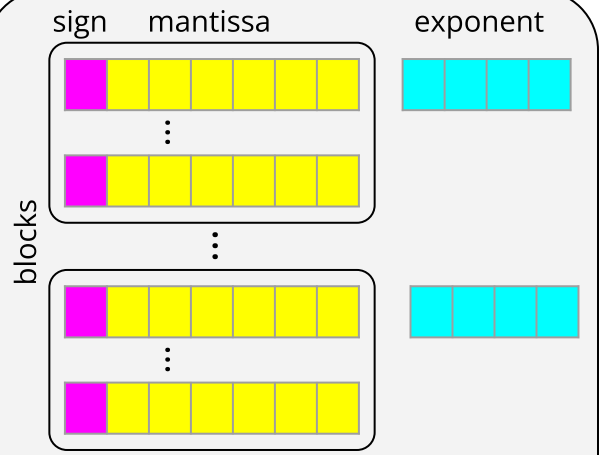
Floating Point (FP)



Each value stored in binary scientific notation

Ex: FP32 (1,8,23)
FP16 (1,5,10)
TF32 (1,8,10)
BFloat16 (1,8,7)

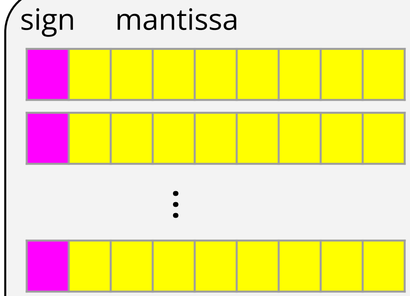
Block Floating Point (BFP)



"Per block" scaling factor

Ex: BFP
MSFP

Fixed Point



Scale values to integers (fixed exponent)

Ex: INT8
INT4
INT2

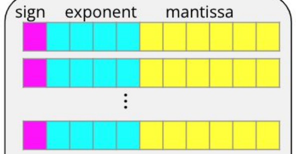
More **exponent** bits = more dynamic range among representable values
More **mantissa** bits = more precision (significant digits)
Square of **mantissa** width approximates multiplier area

Pomona College 1887

UCI

Adaptive Floating Point

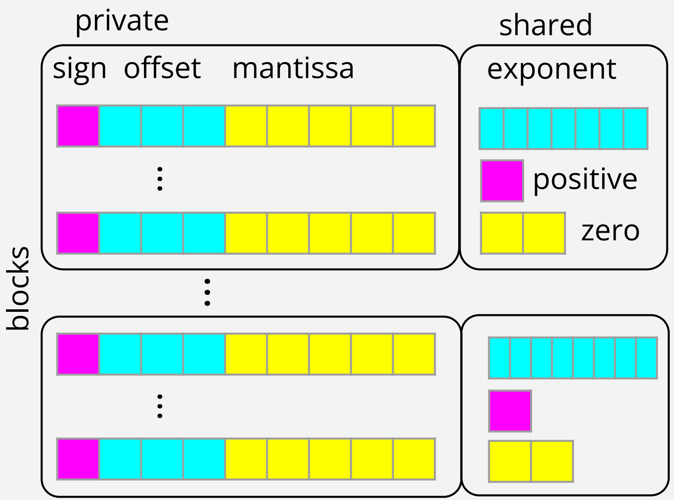
Floating Point (FP)



Each value stored in binary scientific notation

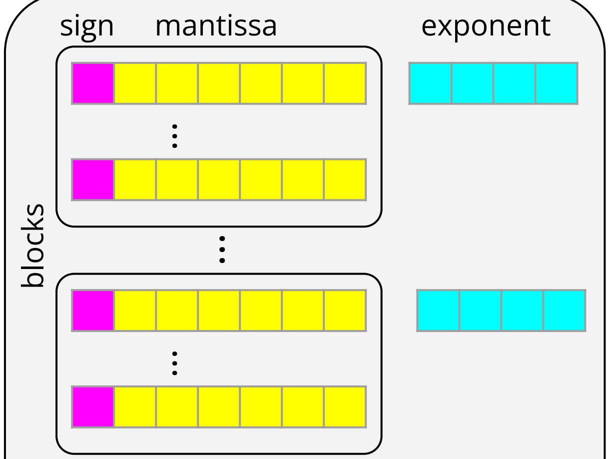
Ex: FP32 (1,8,23)
FP16 (1,5,10)
TF32 (1,8,10)
BFloat16 (1,8,7)

Adaptive Floating Point (AFP)



Adds **private exponent offsets**
(More range + effective precision)
Additional shared fields can recover some precision

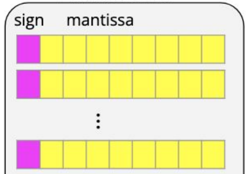
Block Floating Point (BFP)



"Per block" scaling factor

Ex: MSFP

Fixed Point



Scale values to integers (fixed exponent)

Ex: INT8
INT4
INT2

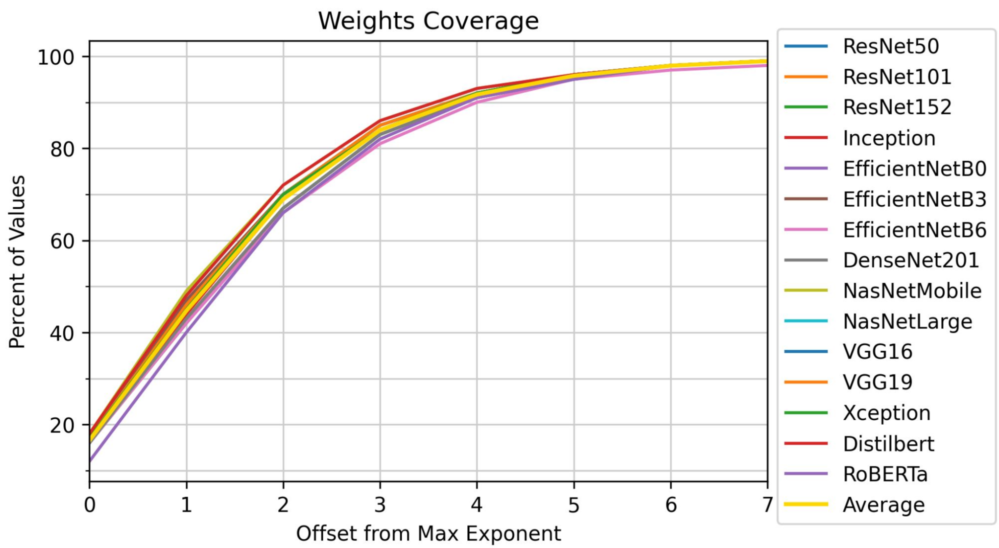
Pomona College 1887

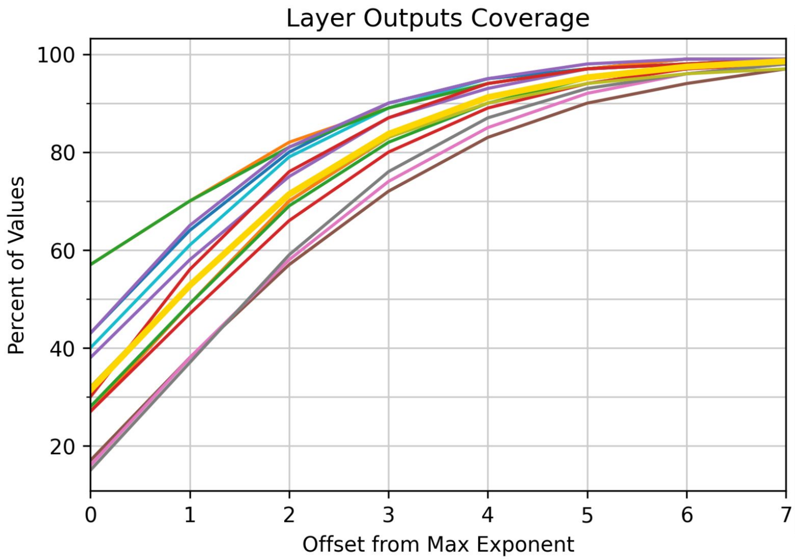
Nearest rounding provides best inference performance.
Minimum mantissa with private exponents is 5 bits.

UCI

Representing ML Data

In actual ML data, how much do the values vary within blocks?





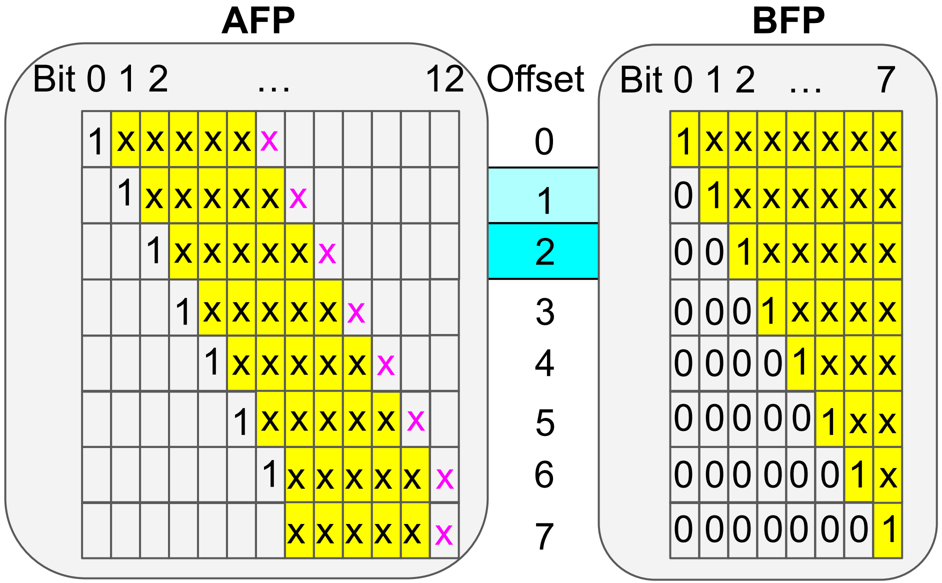
Very consistent across models!

3-bit private offset covers 99% of all weights and outputs!

Pomona College

UCI

AFP Maintains Precision

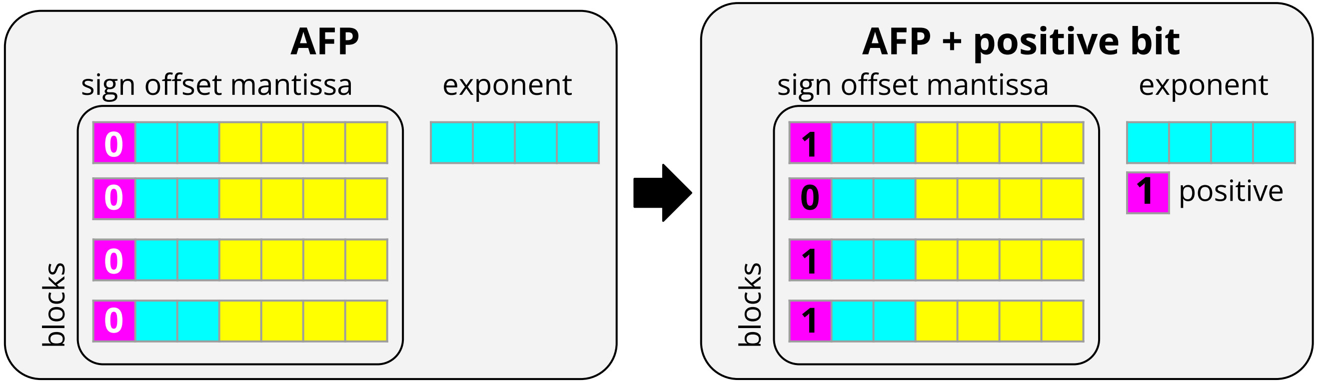


For weights and layer outputs, AFP reduces absolute error by 23% and 46% and relative error by 60% and 43% vs BFP.

Pomona College

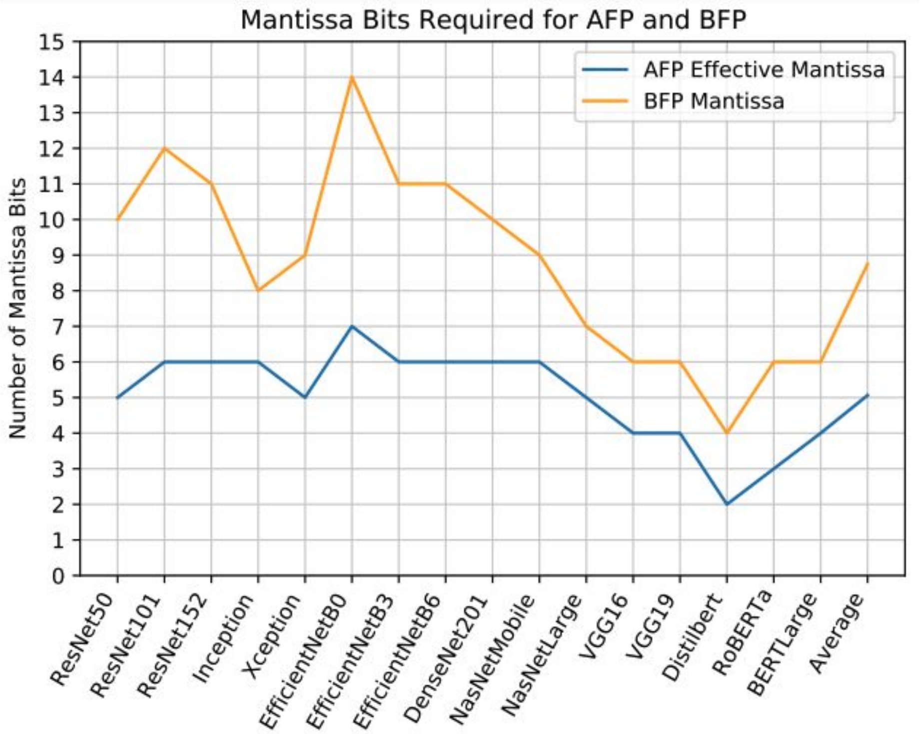
UCI

Block Characterization



Positive blocks common in layer outputs.
Lightweight compression of 1 mantissa bit.

Main Results



AFP vs	Compute Density	Memory Density
FP32	12x	3.2x
BFP	4x	1.6x

Performance improvement if compute-limited

or, if bandwidth-limited

Target performance: 99% of FP32's

Contact: tomyenhsiyeh@gmail.com

Pomona College 1887

UCI