On Stochastic Rounding with Few Random Bits

Andrew Fitzgibbon, Stephen Felix
Graphcore

ENUMATH 2025

awf@fitzgibbon.ie

Context

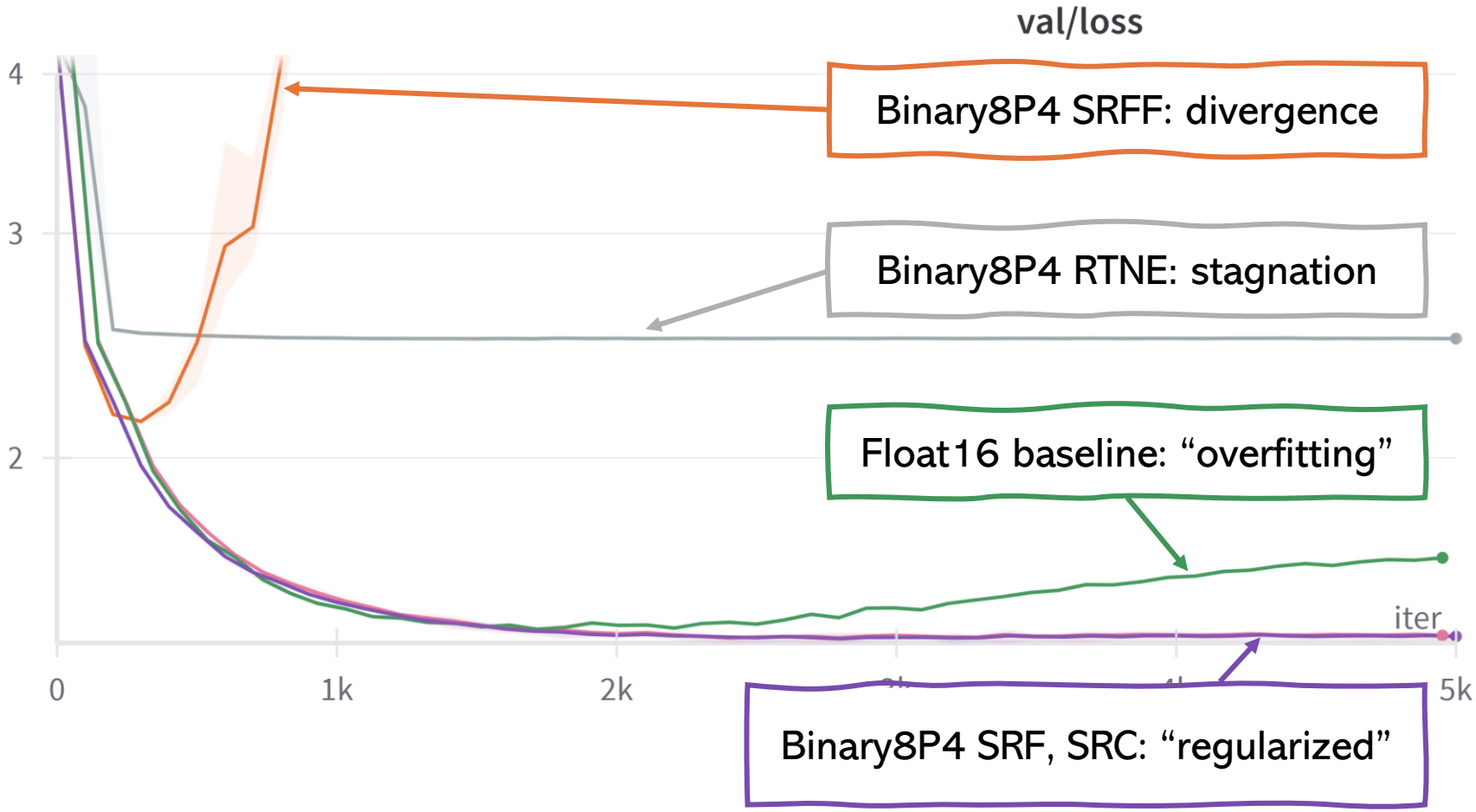
Memory, time, and power requirements of AI models mean moving to narrower and narrower (8, 6, or even 4-bit) floating point formats.

Stochastic rounding improves robustness of narrow-precision computations.

But… adds the cost of supplying random bits (rbits).

Recent work looks at supplying rbits more cheaply (nearby discards, re-use), or using fewer rbits.

Importance in practice to be explored…



Small transformer (10m params)

Training an 8 bit model with 16-bit gradients – important case for QAT

Master weights in P3109 binary8p4se

Using 3 rbits

W_8: F8 = *initialize*
while …
  W_{16} = To16(W_8)
  g_{16}: F16 = $\nabla loss(W_{16})$
  u_{16}: F16 = Adam update in F16
  W_8 = RoundTo8($W_{16} + u_{16}$)

[*] *Do not over index on “overfitting” vs “regularized” – this only applies to small models.*

This paper: quantity, not quality

Recent work looks at SR with few random bits, but optimal number of bits dependent on many factors

- Accumulator precision (precision of values before rounding, e.g. p=11 for Binary16, p=8 for BFloat16, other for specific h/w implementations
- Cost of random bits
- Accuracy requirements

Recent work [1] also looks at bias in SR with few random bits

[1]: El Arar, Fasi, Filip, Mikaitis. “Probabilistic error analysis of limited-precision stochastic rounding”, 2024

Defining SR

Real value X is to be rounded to floating point value set

$$\{m \times 2^e \mid m \in \mathbb{M}\}$$

Where range of integer significands $\mathbb{M} = \{2^P \leq m < 2^{P+1}, m \in \mathbb{N}\}$

Loosely, ignoring subnormals etc, we obtain real-valued significand using

$$E = \lfloor \log_2 X \rfloor$$

$$S = X \times 2^{-E+P-1}$$

and rounding S to integer $\lfloor S \rfloor$ or $\lfloor S \rfloor + 1$

Defining SR

Loosely, ignoring subnormals etc, we obtain real-valued significand using

$$E = \lfloor \log_2 X \rfloor$$

$$S = |X| \times 2^{-E+P-1}$$

and round to integer $\lfloor S \rceil$ or $\lfloor S \rceil + 1$

Using:

High-school rounding: $\hat{S} = \lfloor S + 0.5 \rfloor$

Stochastic rounding: $\hat{S} = \lfloor S + R \rfloor$ where $R \sim \text{Uniform}(0,1)$

Defining SR

Loosely, ignoring subnormals etc, we obtain real-valued significand using

$$E = \lfloor \log_2 X \rfloor$$

$$S = |X| \times 2^{-E+P-1}$$

and round to integer $\lfloor S \rfloor$ or $\lfloor S \rfloor + 1$

Using:

High-school rounding: $\hat{S} = \lfloor S + 0.5 \rfloor$

Stochastic rounding: $\hat{S} = \lfloor S + R \rfloor$ where $R \sim \text{Uniform}(0,1)$

"Round up with probability δ, down with probability $1 - \delta$, where $\delta = S - \lfloor S \rfloor$"

Our contribution

1. Identify bias in the low-r, low-p scenario ([1] did low-r only)

2. Show how to correct the bias in implementations
    1. Quick and reasonably accurate
    2. Slightly less quick and more accurate

3. With bias computations for all of the above

SR in practice

Concrete implementation in “gfloat” python package

Normal “round” function takes input value (and target format) only

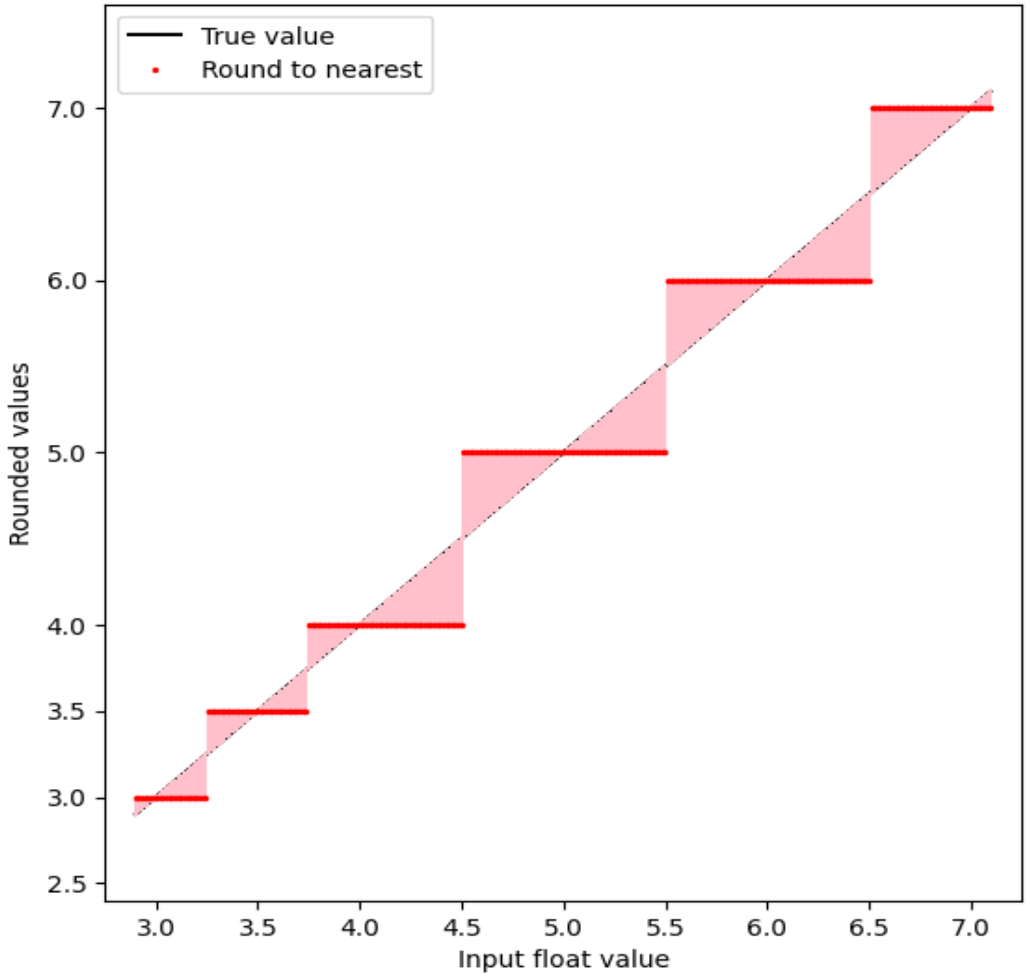
$$\text{round}(S\colon \mathbb{R}) \;= \lfloor S + 0.5 \rfloor$$

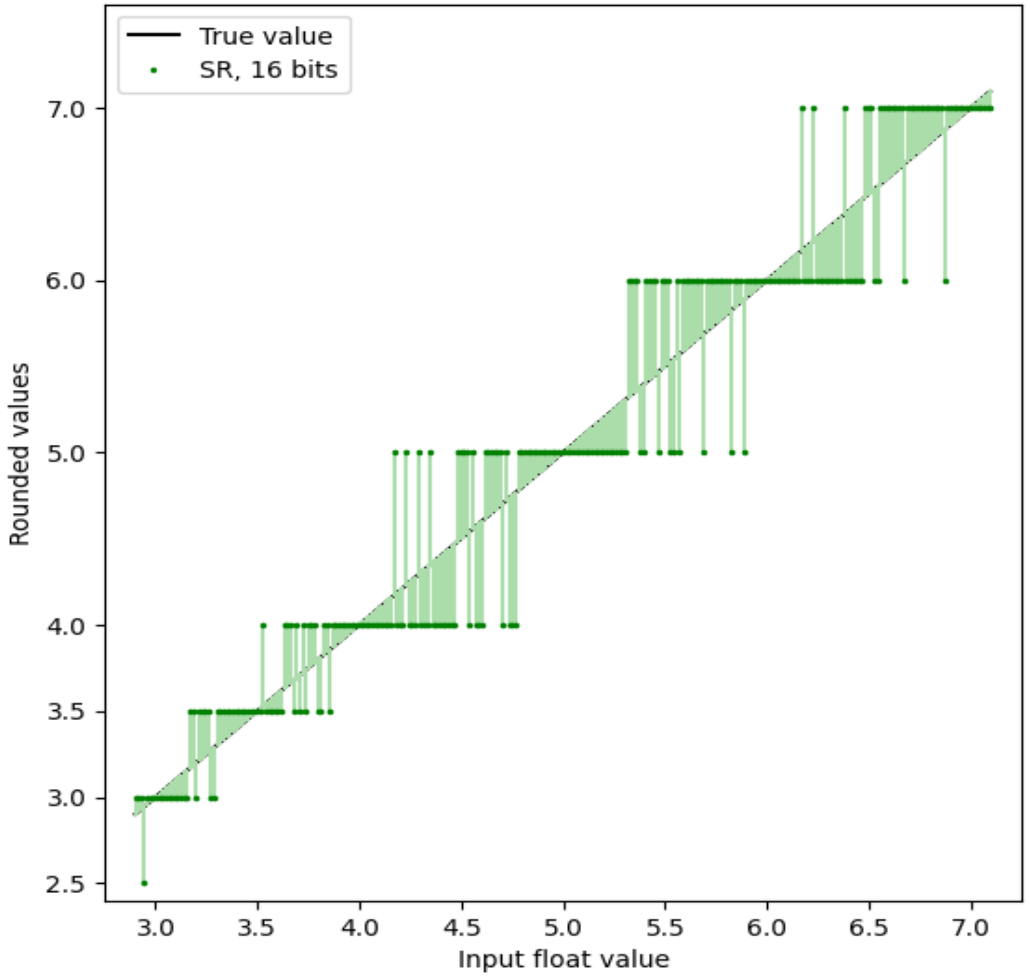
Stochastic “round” function also takes random bits, an integer $0\,..\,2^{\#R}$.

$$\text{round}(S\colon \mathbb{R}, R\colon \mathbb{N}) \;= \left\lfloor S + R \times 2^{-\#R} \right\rfloor$$

Note: Not “the number of random bits”, the actual bits themselves.

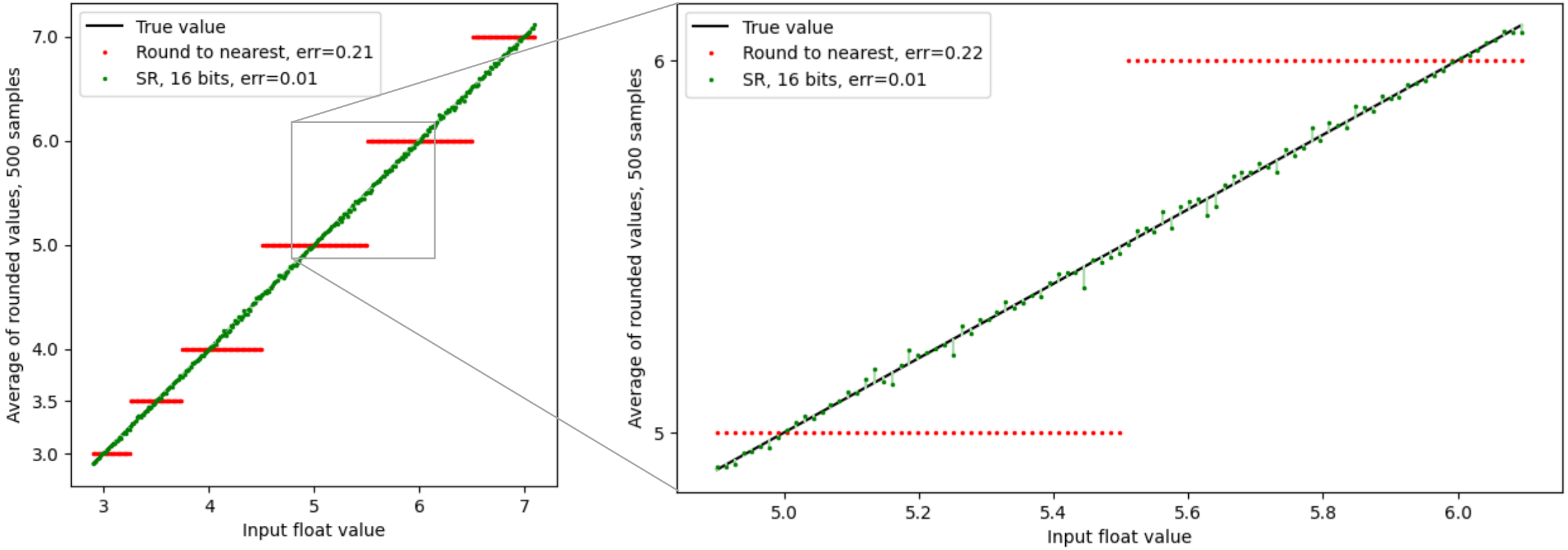
SR in practice



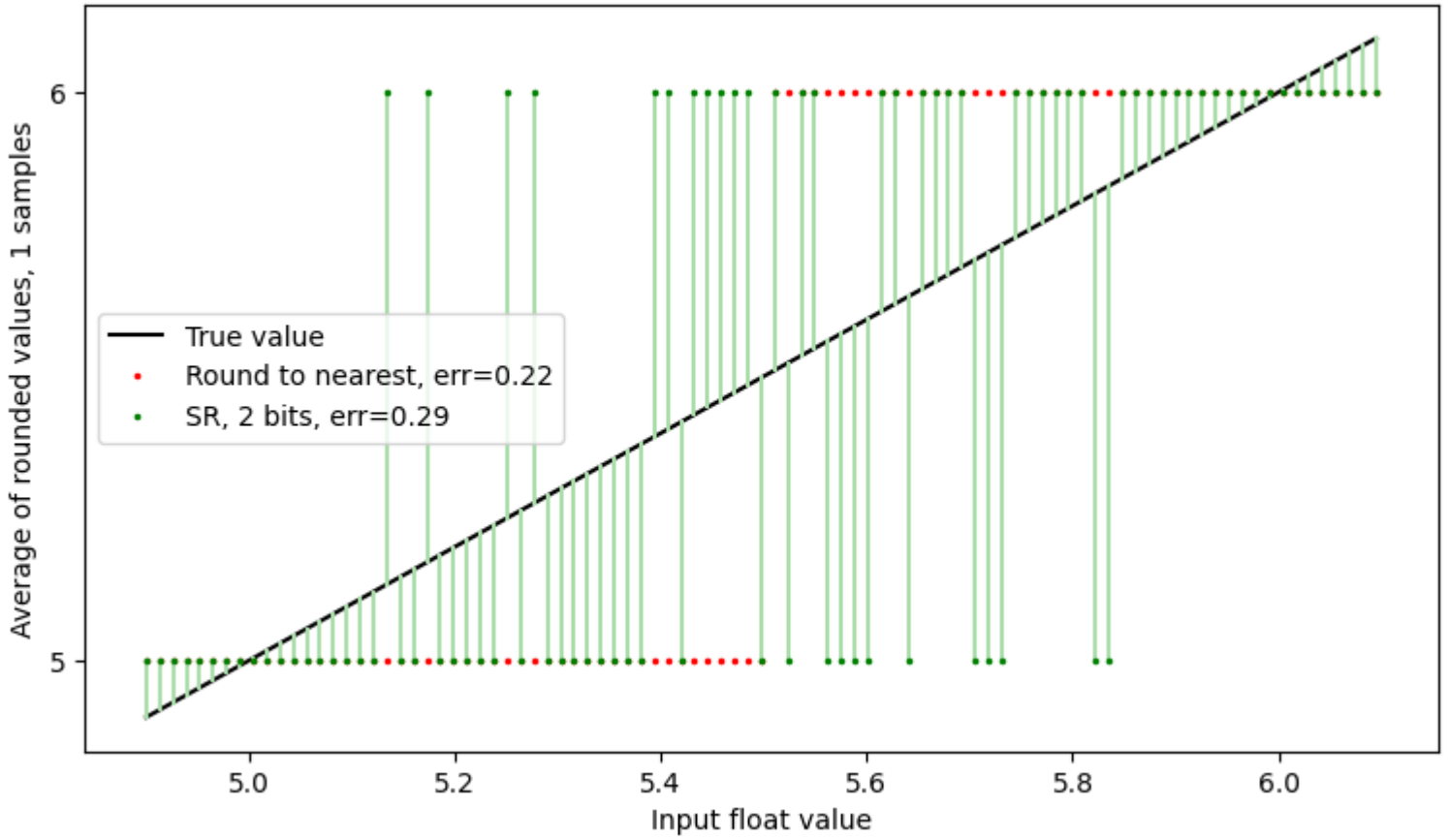


Rounding of $X \in$ "$\mathbb{R}$" to $Y \in \{3.0, 3.5, 4.0, 5.0, 6.0, 7.0\} \subset \mathbb{F}$ for a format $\mathbb{F}$

Average over a few runs...

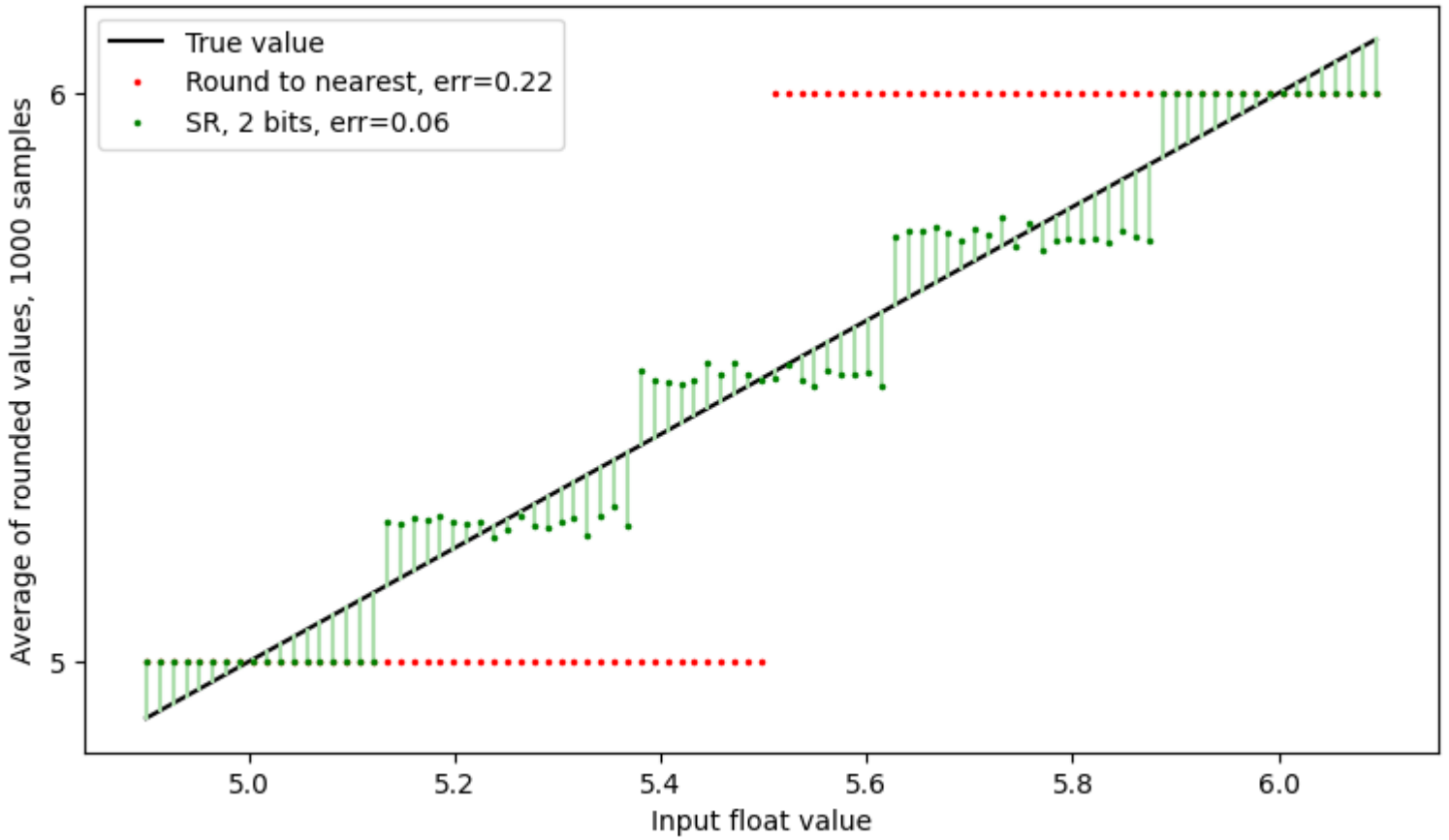


But that was 16 rbits on float64 inputs...



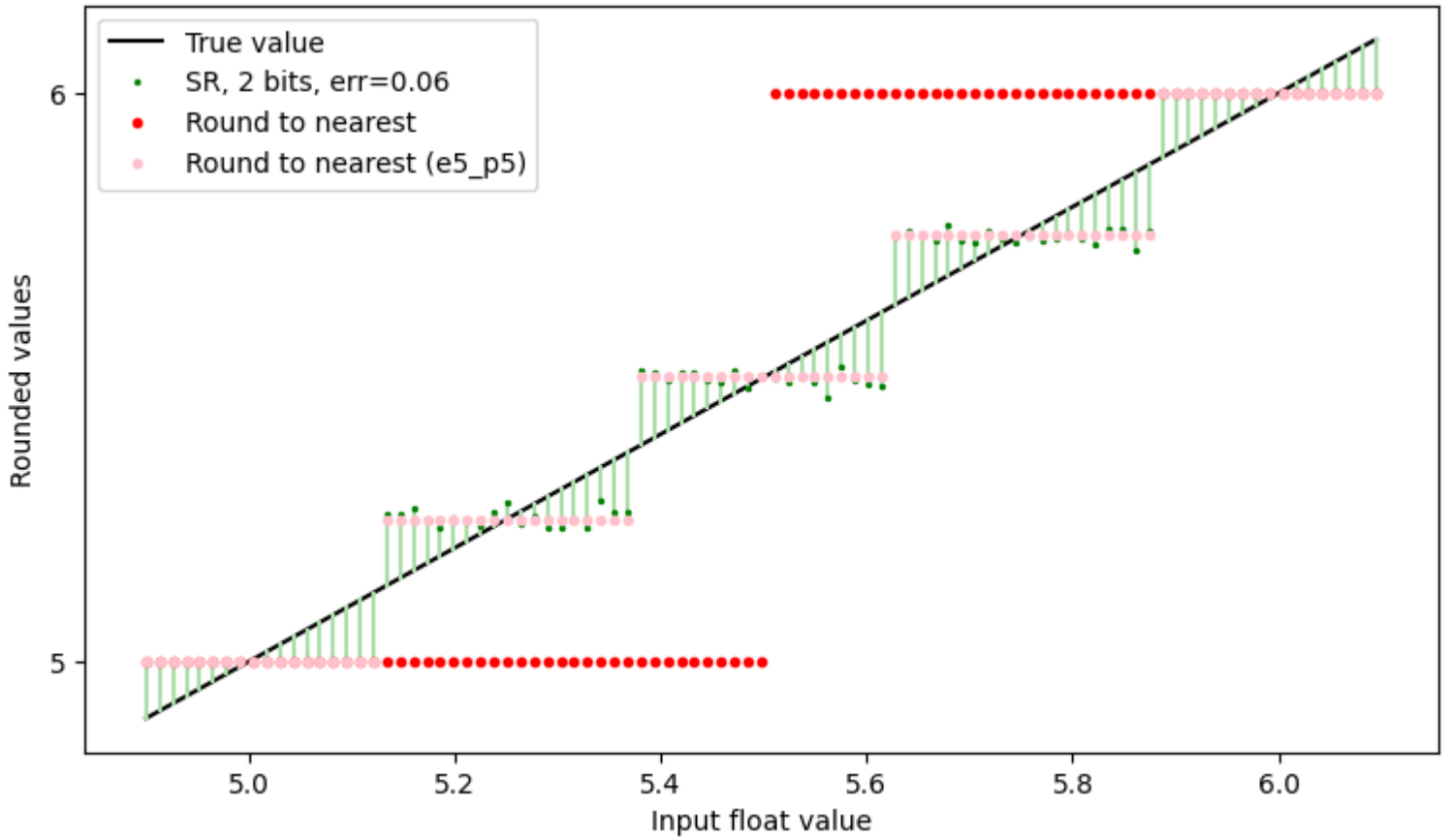
Rounding with 2 bits on float64 inputs – looks fine, right?

Few-bit SR, high-precision inputs



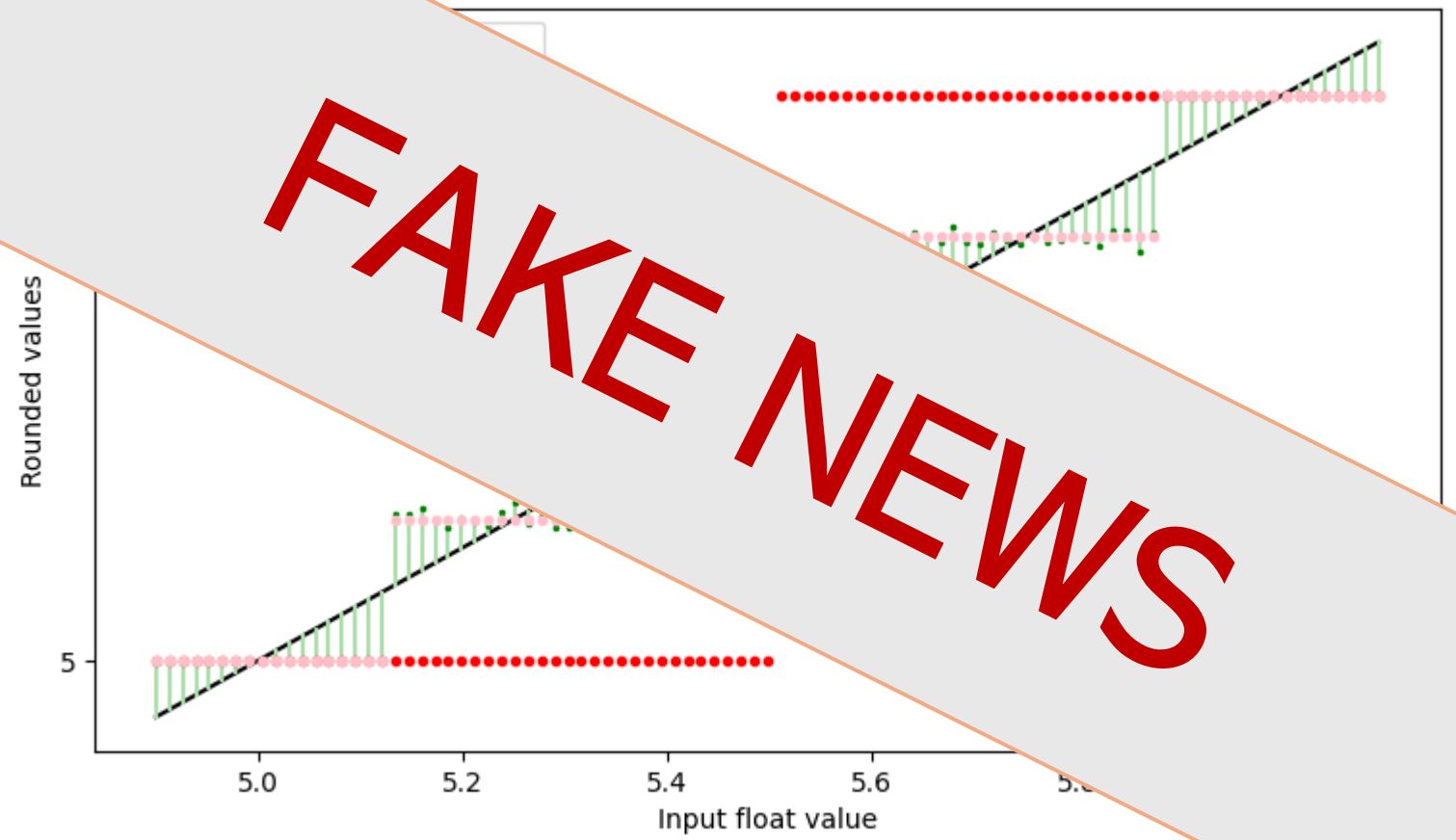
Averaged: looks like 2 rbits gives 2 precision bits on average

Few-bit SR, high-precision inputs



Averaged: looks like 2 rbits gives 2 precision bits on average
Yes, confirmed by RTNE to a format with 2 more precision bits

Few-bit SR, high-precision inputs

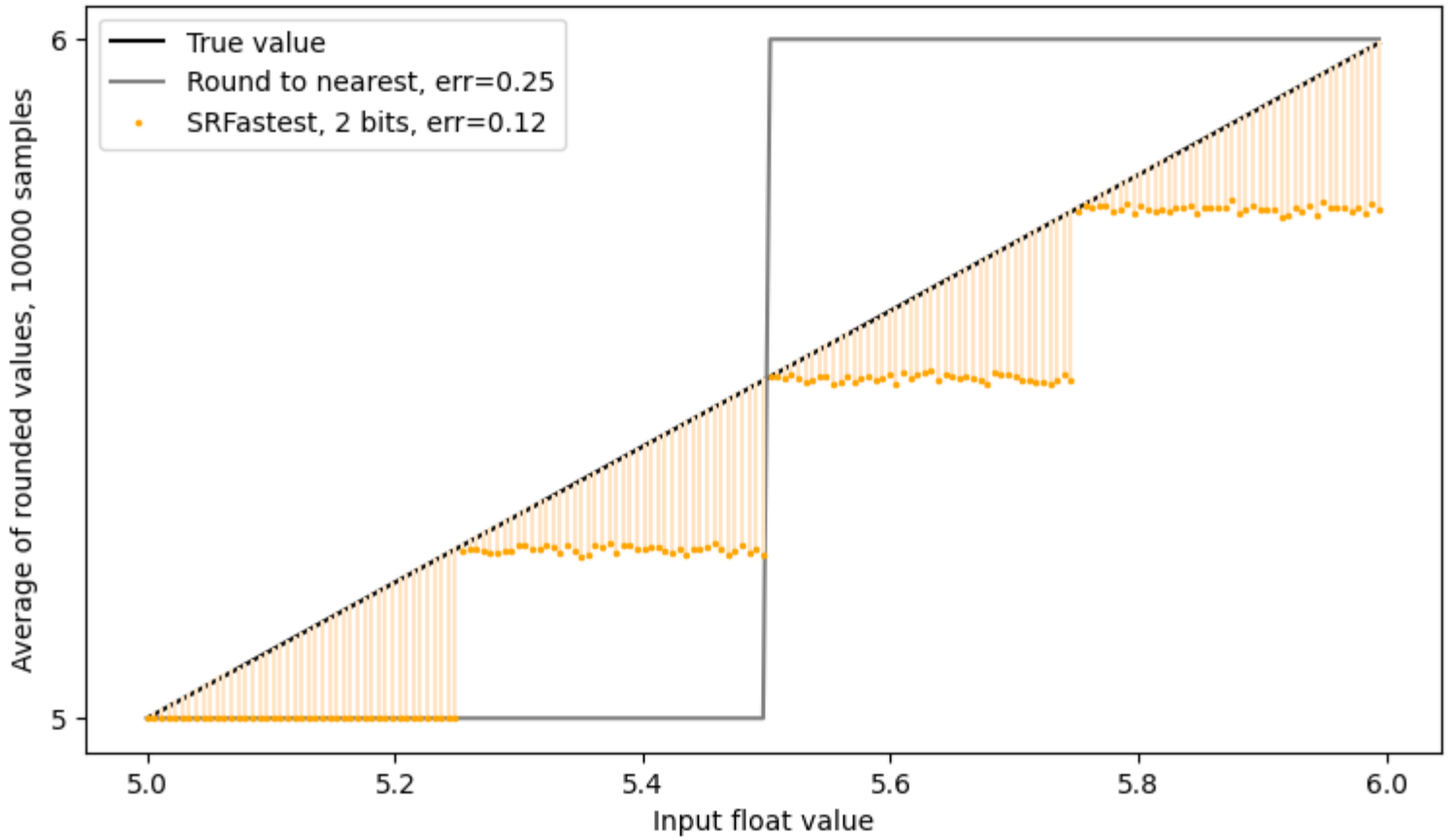


FAKE NEWS

Averaged: looks like 2 rbits gives 2 precision bits on average
Yes, confirmed by RTNE to a format with 2 more precision bits

“High school” SR is biased

$$\mathrm{round}(S:\mathbb{R}, R:\mathbb{N}) = \left\lfloor S + R \times 2^{-\#R} \right\rfloor$$



Aside: how do we measure bias?

Expected value of error

$$\text{Bias} = \mathbb{E}[X - \text{round}(X)\ ]$$

But what is the expectation over?

Need to pick a probability distribution $p(X)$ to write, more precisely:

$$\text{Bias} = \mathbb{E}_{X \sim p(X)}[X - \text{round}(X)\ ]$$

What probability distribution? It can't be uniform on $[-MAX, MAX]$, as then always rounding toward zero is "unbiased".

We really want a family of test distributions, one per float-pair.

Much of the paper… formulae for bias

$$
\begin{aligned}
2^{-N} \sum_{n=0}^{2^N-1} \left(\int_0^1 \mathbb{1}[\mathsf{R}_{\mathsf{SRFF}}(x,n)]\,\mathrm{d}x - \frac{1}{2} \right) &= \\
2^{-N} \sum_{n=0}^{2^N-1} \int_0^1 \mathbb{1}\left[x + n \times 2^{-N} \geq 1\right] \mathrm{d}x - \frac{1}{2} &= \\
2^{-N} \sum_{n=0}^{2^N-1} \int_{1-n\times 2^{-N}}^1 \mathrm{d}x - \frac{1}{2} &= \\
-2^{-(N+1)} &\neq 0
\end{aligned}
$$

$$Bias_{SRFF} = -2^{-(N+1)}$$

Much of the paper…

C. Bias of SRFF, finite-precision inputs

The bias at a single x value is

$$\begin{aligned} & bias_{SRFF}(x) \\ &= 2^{-N} \sum_{n=0}^{2^N-1} \mathbb{1}[\mathsf{R}_{\mathsf{SRFF}}(x,n)] - x \\ &= 2^{-N} \sum_{n=0}^{2^N-1} \mathbb{1}\left[x + 2^{-N}n \geq 1\right] - x \end{aligned}$$

and its expectation over the 2^D input values is

$$\begin{aligned} & bias_{SRFF,D} \\ &= 2^{-D} \sum_{i=0}^{2^D-1} bias_{SRFF}(x := 2^{-D}i) \\ &= 2^{-D} \sum_{i=0}^{2^D-1} \left(2^{-N} \sum_{n=0}^{2^N-1} \mathbb{1}\left[2^{-D}i + 2^{-N}n \geq 1\right] - 2^{-D}i \right) \\ &= 2^{-D} \sum_{i=0}^{2^D-1} \left(2^{-N} \sum_{n=0}^{2^N-1} \mathbb{1}\left[2^{-D}i + 2^{-N}n \geq 1\right] - 2^{-D}i \right) \\ &= \left(2^{-D} \sum_{i=0}^{2^D-1} 2^{-N} \sum_{n=0}^{2^N-1} \mathbb{1}\left[2^{-D}i + 2^{-N}n \geq 1\right] \right) - 2^{-D} \sum_{i=0}^{2^D-1} 2^{-D}i \\ &= \left(2^{-N} \sum_{n=0}^{2^N-1} 2^{-D} \sum_{i=0}^{2^D-1} \mathbb{1}\left[i \geq 2^D - 2^{D-N}n\right] \right) - \frac{1-2^{-D}}{2} \\ &= \left(2^{-N} \sum_{n=0}^{2^N-1} 2^{-D} \sum_{i=\lceil 2^D - 2^{D-N}n \rceil}^{2^D-1} 1 \right) - \frac{1-2^{-D}}{2} \\ &= 2^{-N} \sum_{n=0}^{2^N-1} 2^{-D} \left((2^D-1) - \lceil 2^D - 2^{D-N}n \rceil + 1 \right) - \frac{1-2^{-D}}{2} \\ &= 2^{-N} \sum_{n=0}^{2^N-1} 2^{-D} \lfloor 2^{D-N}n \rfloor - \frac{1-2^{-D}}{2} \end{aligned}$$

If $N \leq D$ then 2^{D-N} is an integer so the floor operation is a no-op. If $D < N$ then 2^{D-N} is a reciprocal power of two so the argument will be integral for some values of n. We may obtain a bound using $\lfloor x \rfloor \leq x$, which will be tight for $N \leq D$.

$$\begin{aligned} bias_{SRFF,D} &= 2^{-N} \sum_{n=0}^{2^N-1} 2^{-D} \lfloor 2^{D-N}n \rfloor - \frac{1-2^{-D}}{2} \\ &\leq 2^{-N} \sum_{n=0}^{2^N-1} 2^{-N}n - \frac{1-2^{-D}}{2} \\ &= \frac{1-2^{-N}}{2} - \frac{1-2^{-D}}{2} \\ &= -2^{-(N+1)} + 2^{-(D+1)} \\ &= \frac{2^{-D} - 2^{-N}}{2} \end{aligned}$$

$$Bias_{SRFF,D} \leq 2^{-(D+1)} - 2^{-(N+1)}$$

Bias formulae

Writing $N = \#R$, i.e. using N bits of randomness

$$Bias_{SRFF} = -2^{-(N+1)}$$

If incoming values are finite-precision (generally true), with precision D, then

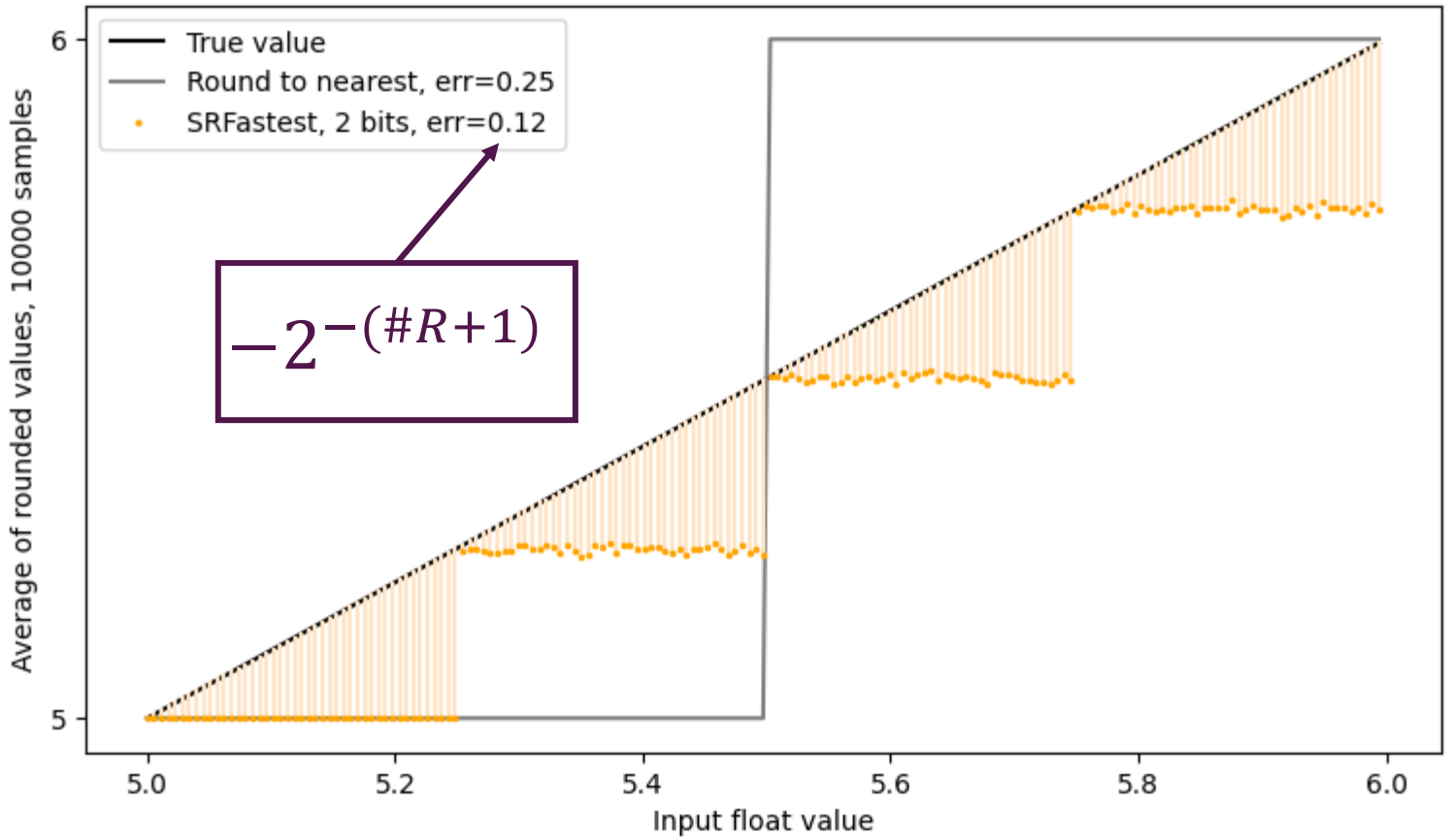
$$Bias_{SRFF,D} \leq 2^{-(D+1)} - 2^{-(N+1)}$$

Bound tight for $N < D$, note zero for $N = D$, i.e. number of rbits equal to difference in precisions.

This is the case in preceding work (the non-few-bit case). I.e. existing hardware is fine, as it uses a lot of rbits; any future hardware trying to save rbits will need to correct this bias.

“High school” SR is biased

$$\text{round}(S:\mathbb{R}, R:\mathbb{N}) = \left\lfloor S + R \times 2^{-\#R} \right\rfloor$$



Debugging: Rounding profiles

With 2 random bits, we can just treat SR as selecting from 4 deterministic schemes:

$$\text{round}(S, 0b00) = \lfloor S + 0.00 \rfloor = \lfloor S \rfloor, \quad \text{“Always floor”}$$

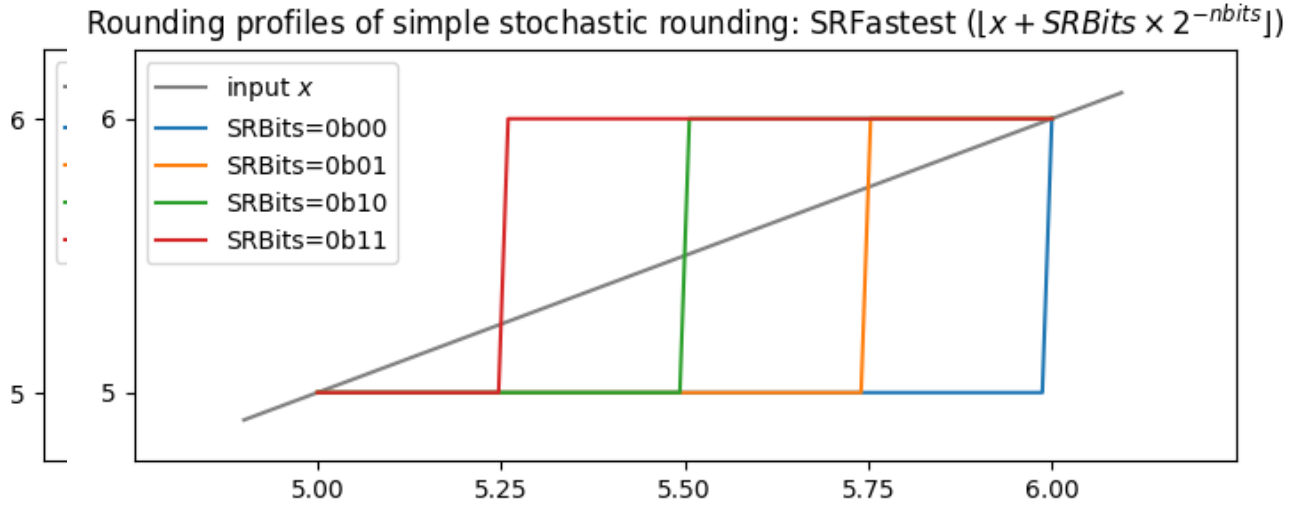
$$\text{round}(S, 0b01) = \lfloor S + 0.25 \rfloor \quad \text{“Ceil if } \delta \geq 3/4\text{”}$$

$$\text{round}(S, 0b10) = \lfloor S + 0.50 \rfloor \quad \text{“Ceil if } \delta \geq 2/4\text{”}$$

$$\text{round}(S, 0b11) = \lfloor S + 0.75 \rfloor \quad \text{“Ceil if } \delta \geq 1/4\text{”}$$

We can plot these…

… and see the asymmetry



Quick fix:

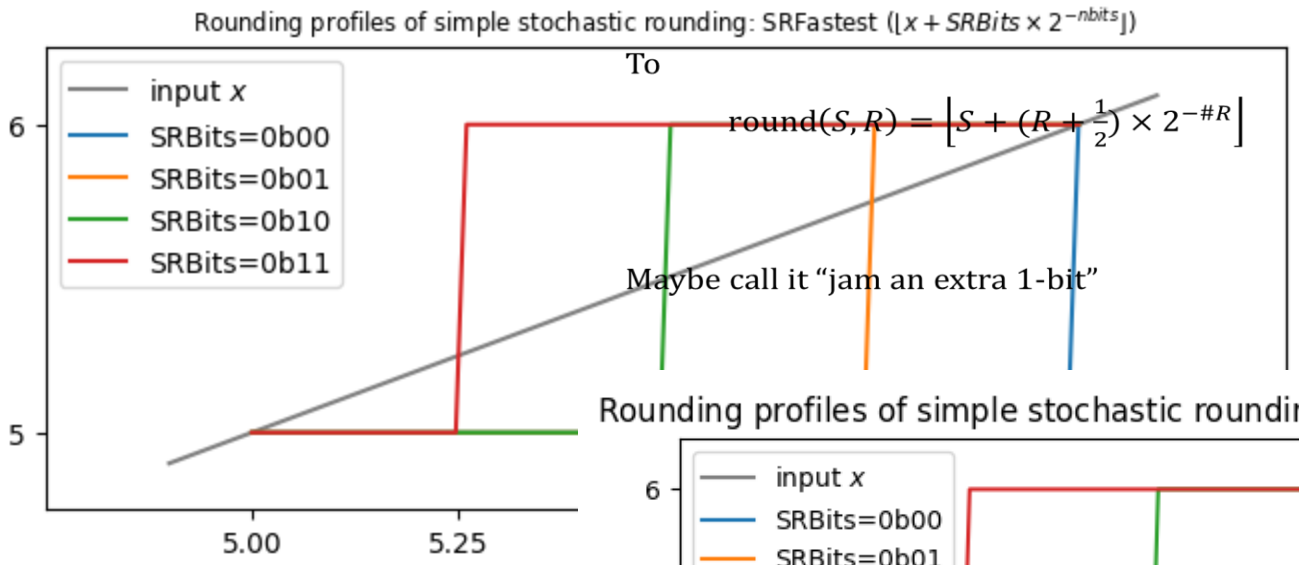
Move from

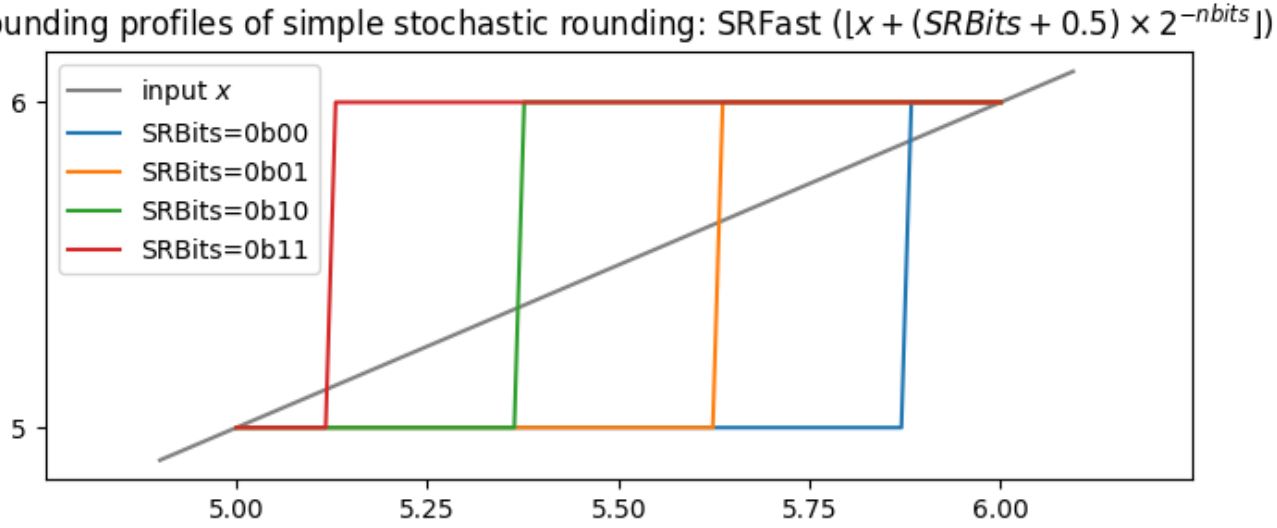
$$\text{round}(S, R) = \lfloor S + R \times 2^{-\#R} \rfloor$$

To

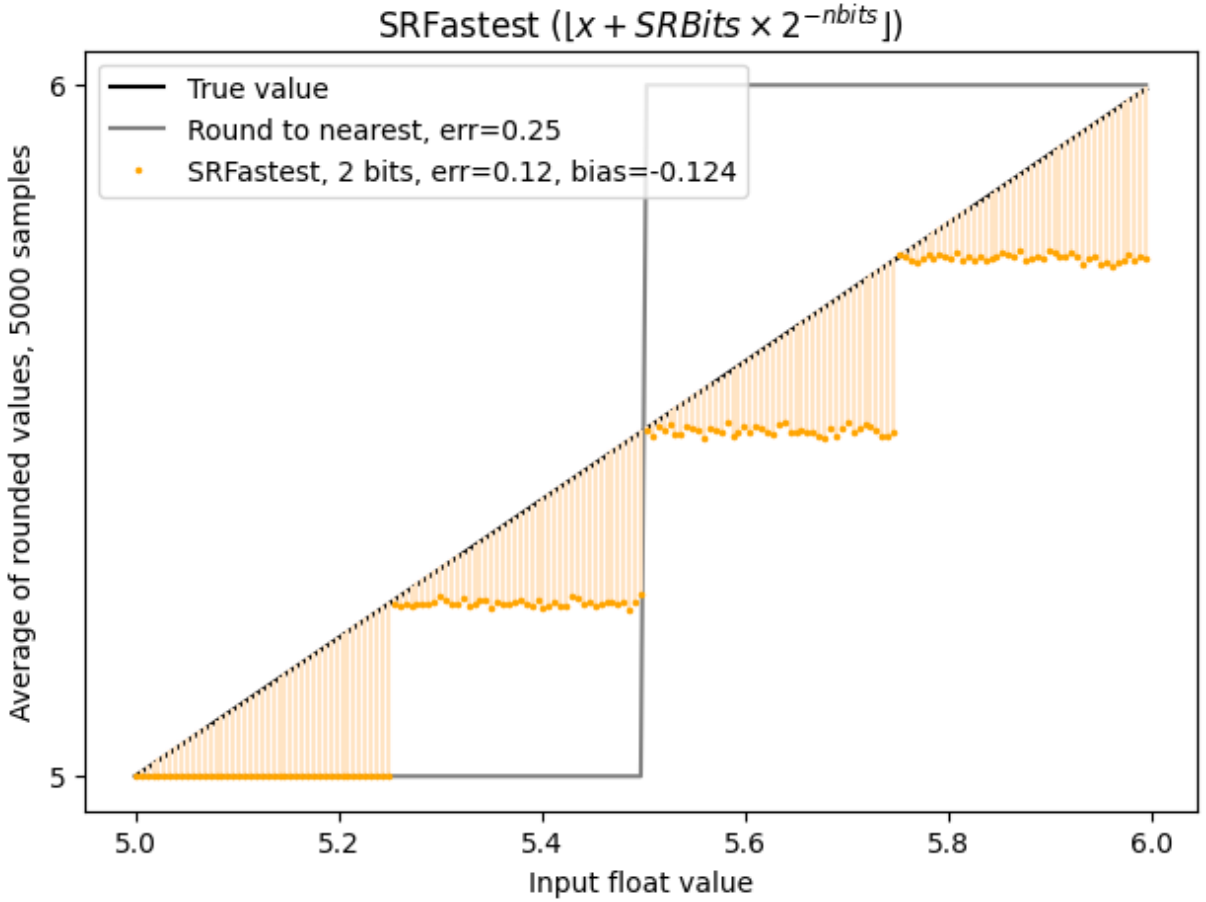
$$\text{round}(S, R) = \lfloor S + (R + \tfrac{1}{2}) \times 2^{-\#R} \rfloor$$

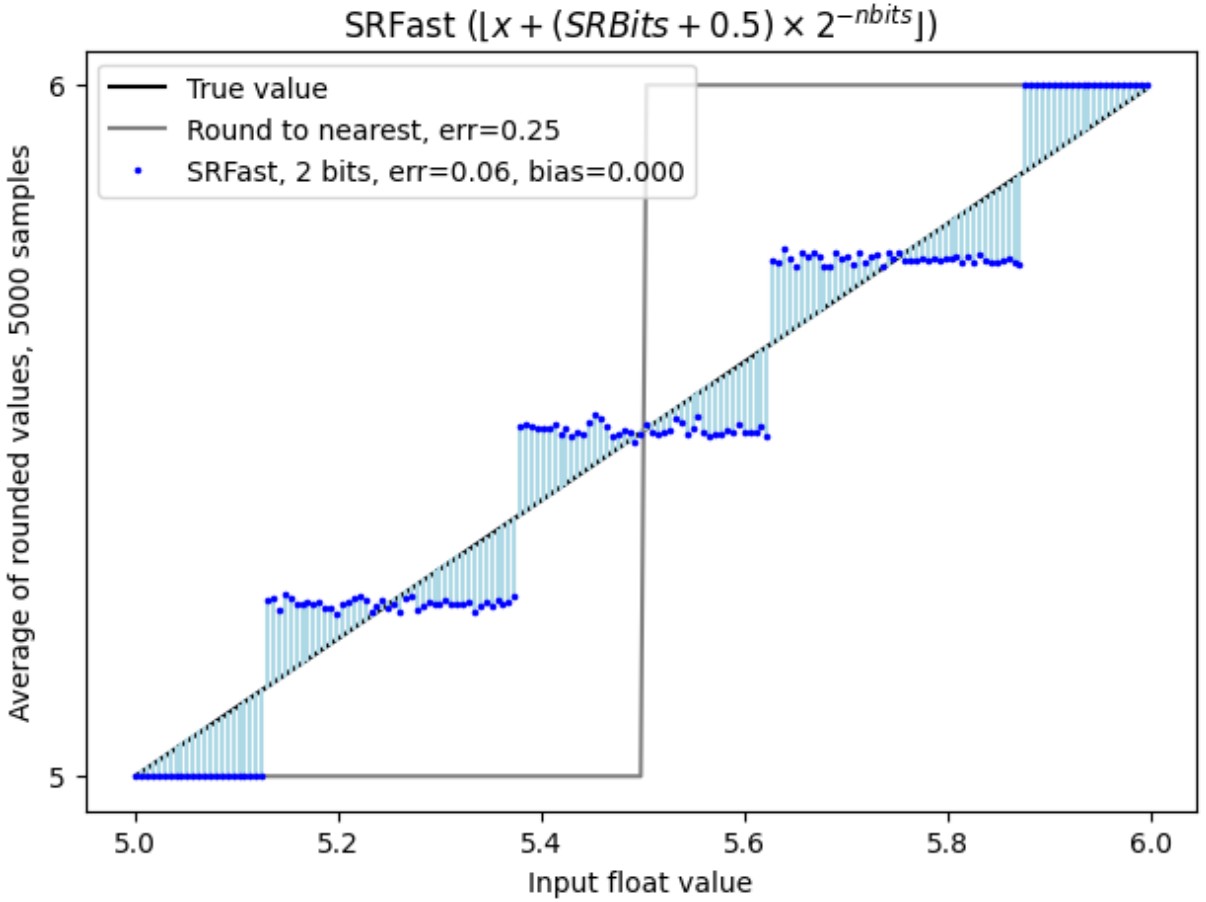
Maybe call it “jam an extra 1-bit”





Fixed. So are we done yet?

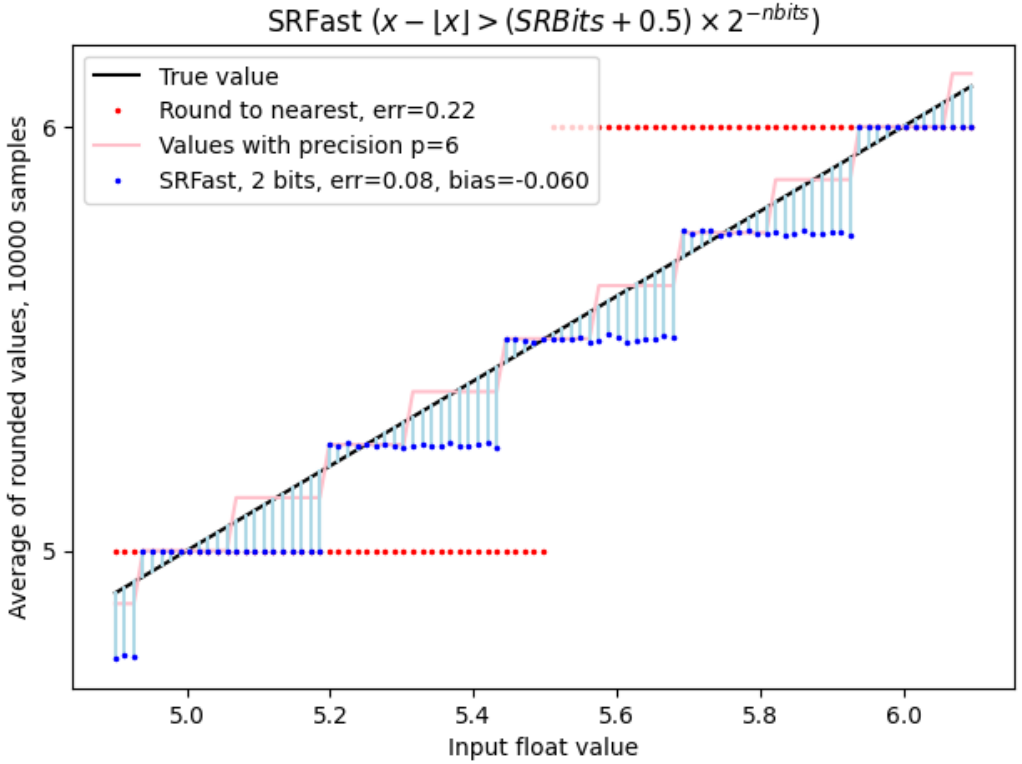




Finite-precision inputs

With limited precision inputs

Bias returns if input precision before SR is close to target precision (e.g. bfloat16, p=8 to binary8p4, precision difference is just 4)



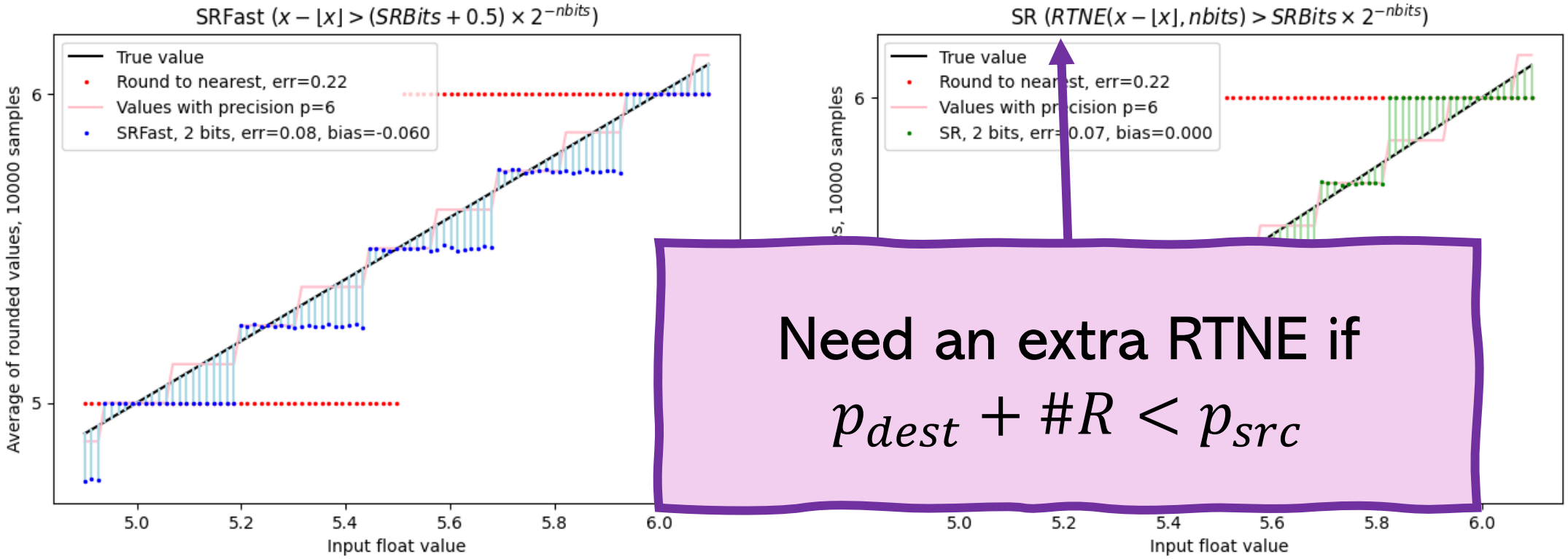
In this example:

- Input precision p=8
- Target precision p=4
- #R = 2

So SR should simulate p=6

With limited precision inputs

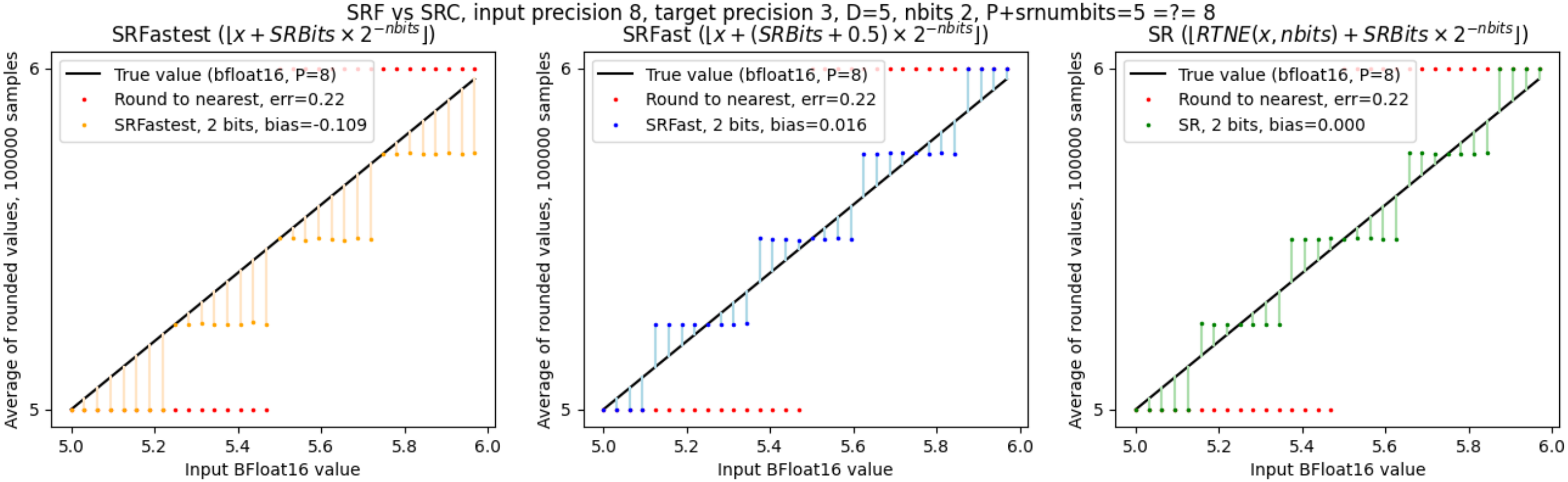
Bias returns if input precision before SR is close to target precision (e.g. bfloat16, p=8 to binary8p4, precision difference is just 4)



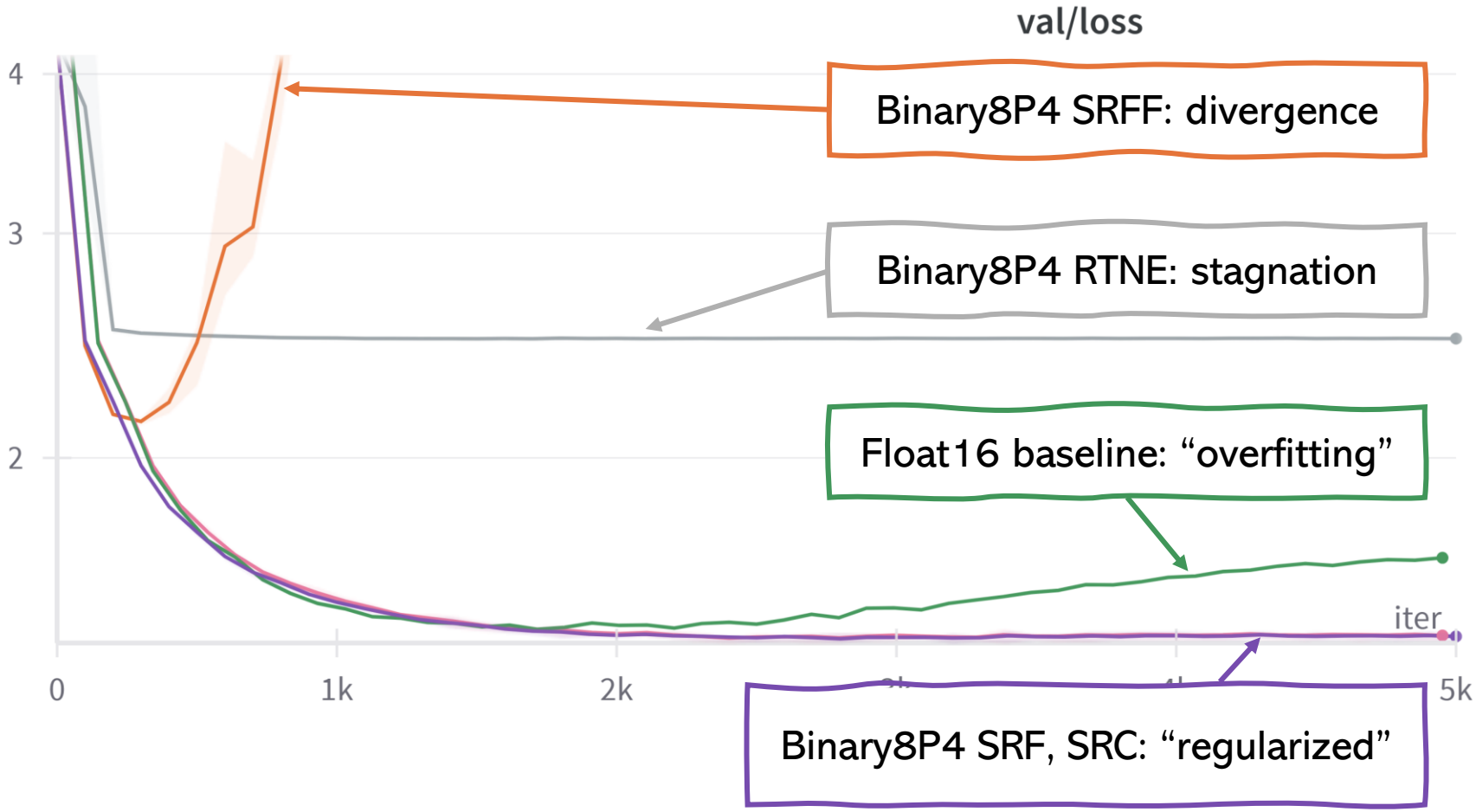
Need an extra RTNE if

$$p_{dest} + \#R < p_{src}$$

BFloat16 to target P3



Importance in practice to be explored…



Small transformer (10m params)

Training an 8 bit model with 16-bit gradients – important case for QAT

Master weights in Binary8P4

Using 3 rbits

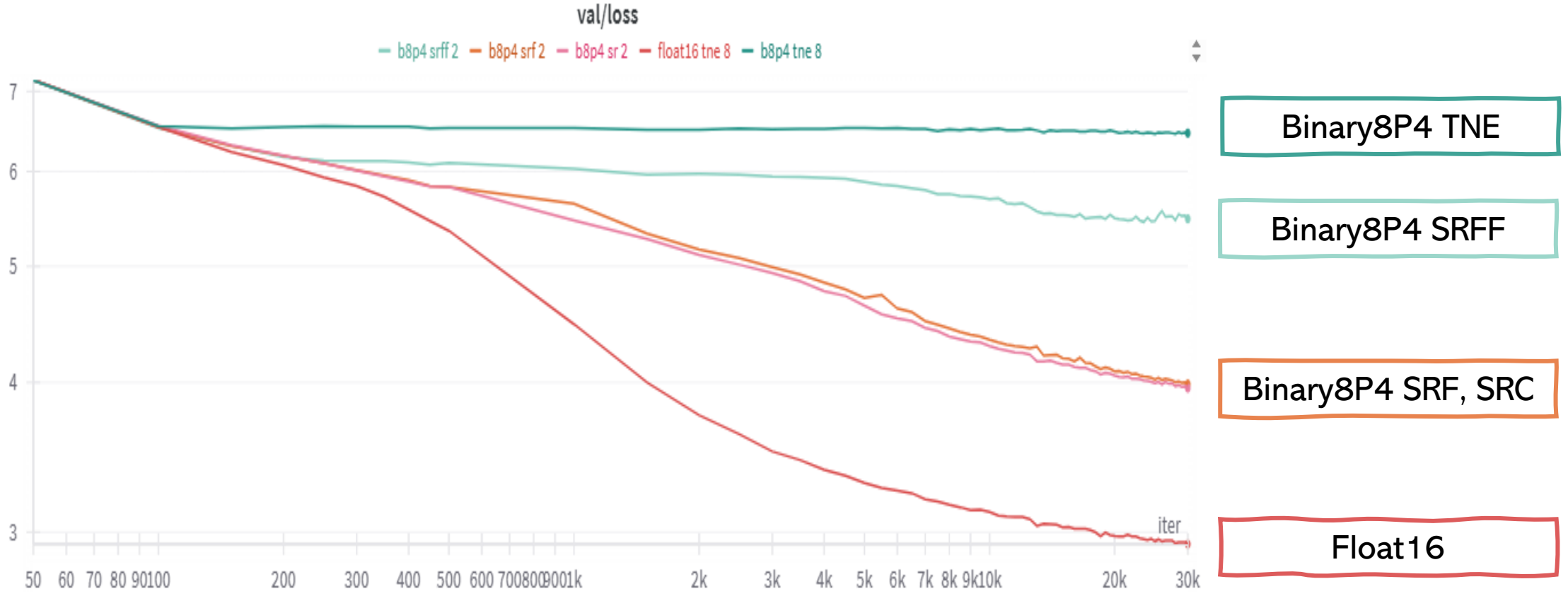
W_8: F8 = *initialize*
while …
  W_{16} = To16(W_8)
  g_{16}: F16 = $\nabla loss(W_{16})$
  u_{16}: F16 = Adam update in F16
  W_8 = RoundTo8($W_{16} + u_{16}$)

[*] *Do not over index on "overfitting" vs "regularized" – this only applies to small models.*

Medium scale (350M params, GPT-2 like)

Comparisons are more tricky – some suggestion that learning rate should be higher for SR

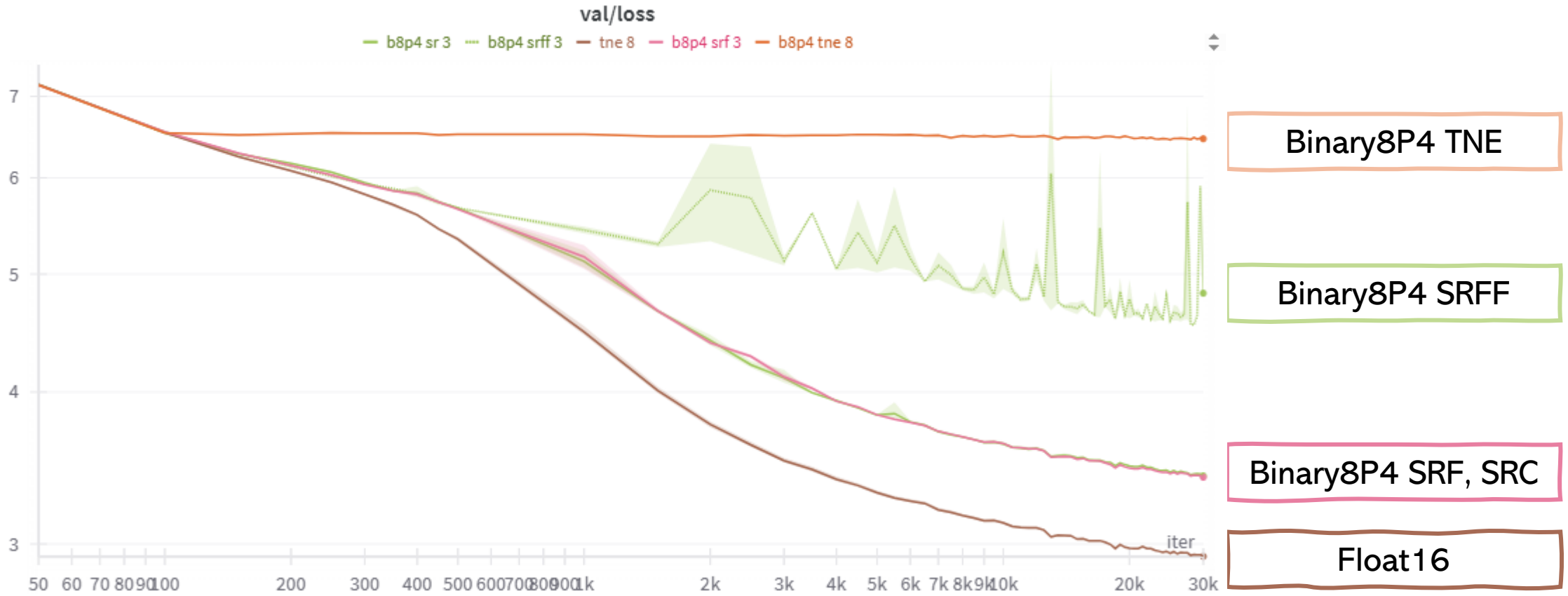
2-bit SR:



Medium scale (350M params, GPT-2 like)

Comparisons are more tricky – some suggestion that learning rate should be higher for SR

3-bit SR:



This whole paper:

Given S as binary real

0011011.10110101011110…

We add either 0.5

.10000000000000 “High school” rounding

Or random bits

.01011101000101 … “High school SR” with infinite precision – unbiased

Or a “few” random bits

.01100000000000 “High school SR” with fewer rbits than input precision – biased

.01110000000000 “Quick fix” implementation – still biased

Or first RTNE to the number of SR bits, and add the few bits

0011011.10110101011110…

0011011.11000000000000

.01100000000000 “Corrected” implementation – unbiased

This whole paper:

Given S as BF16

0011011.1011

We add either 0.5

.1000 “High school” rounding

Or 4 random bits

.0101 “High school SR” with infinite precision – unbiased

Or a “few” (<4) random bits

.0110 “High school SR” with fewer rbits than input precision – biased

.0111 “Quick fix” implementation – still biased

Or first RTNE to the number of SR bits, and add the few bits

0011011.1110

0011011.1100

.0110 “Corrected” implementation – unbiased

Conclusions

Few-bit SR can be effective, and as more FLOPs are issued per cycle, more SR bits are needed per cycle.

A simple implementation of bias correction can perform as well as the "optimal" implementation.

Existing hardware is fine – new hardware should take note.

Experiments continue… take a look at

https://github.com/graphcore-research/arith25-stochastic-rounding

Email: awf@fitzgibbon.ie