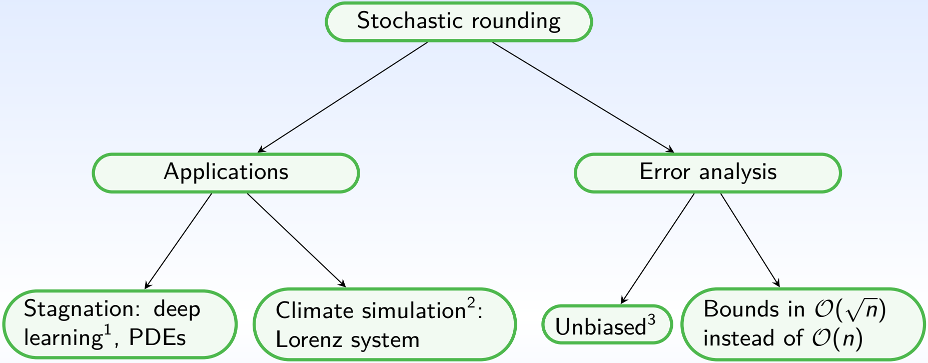
Error Analysis for Algorithms using Stochastic Rounding

El-Mehdi El ARAR
el-mehdi.el-arar@inria.fr

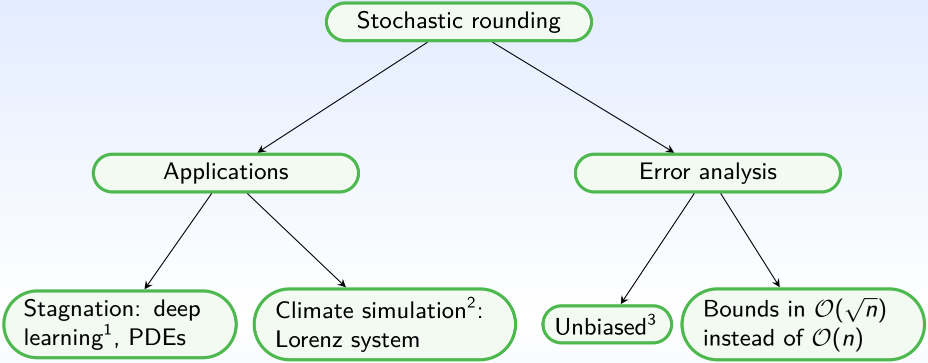
Journées InterFLOP/FPT-4

20/03/2025

Motivation



[1] Gupta et al: Deep Learning with Limited Numerical Precision
[2] Paxton et al: Climate Modeling in Low Precision: Effects of Both Deterministic and Stochastic Rounding
[3] Parker: Monte Carlo Arithmetic: Exploiting Randomness in Floating-Point Arithmetic



El arar: Stochastic models for the evaluation of numerical errors

[1] Gupta et al: Deep Learning with Limited Numerical Precision

[2] Paxton et al: Climate Modeling in Low Precision: Effects of Both Deterministic and Stochastic Rounding

[3] Parker: Monte Carlo Arithmetic: Exploiting Randomness in Floating-Point Arithmetic

1. Error Analysis of Algorithms with Stochastic Rounding

2. Error Analysis for the Implementation of Stochastic Rounding in Hardware

Stochastic rounding (SR)

- For $x, y \in \mathbb{R}$ and $\mathrm{op} \in \{+, -, \times, /\}$

$$\begin{aligned} \mathrm{SR}_p(x) &= x(1+\delta), & |\delta| \leqslant u_p \\ \mathrm{SR}_p(x \,\mathrm{op}\, y) &= (x \,\mathrm{op}\, y)(1+\beta), & |\beta| \leqslant u_p \end{aligned}$$

- Upward rounding $\lceil\!\lceil x \rceil\!\rceil$ and downward rounding $\lfloor\!\lfloor x \rfloor\!\rfloor$:

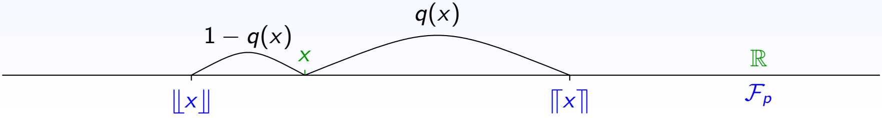


FIGURE. *SR_p with* $q(x) = \frac{x - \lfloor\!\lfloor x \rfloor\!\rfloor}{\lceil\!\lceil x \rceil\!\rceil - \lfloor\!\lfloor x \rfloor\!\rfloor}$

- $\mathbb{E}\left(\mathrm{SR}_p(x)\right) = q(x)\lceil\!\lceil x \rceil\!\rceil + (1 - q(x))\lfloor\!\lfloor x \rfloor\!\rfloor = x$, then $\mathbb{E}(\delta) = \mathbb{E}\left(\frac{\mathrm{SR}_p(x) - x}{x}\right) = 0$

SR and mean independence

- Using SR_p, for $x_1, x_2, x_3 \in \mathcal{F}_p$ and $\mathrm{op}_1, \mathrm{op}_2 \in \{+, -, *, /\}$

$$c = x_1 \,\mathrm{op}_1\, x_2 \,\mathrm{op}_2\, x_3 \quad \Longrightarrow \quad \mathrm{SR}_p(c) = ((x_1 \,\mathrm{op}_1\, x_2)(1+\delta_1) \,\mathrm{op}_2\, x_3)\,(1+\delta_2),$$

  with $\mathbb{E}(\delta_1) = \mathbb{E}(\delta_2) = 0$
- Mean independence: $\mathbb{E}[X_k / X_1, \ldots, X_{k-1}] = \mathbb{E}(X_k)$ for all k
- Independence $\Longrightarrow$ **Mean independence** $\Longrightarrow$ uncorrelatedness

Lemma 1 (M. P. Connolly et al.).

For some $\delta_1, \delta_2, \ldots$, in that order obtained from SR_p, the δ_k are random variables with mean zero such that $\mathbb{E}[\delta_k / \delta_1, \ldots, \delta_{k-1}] = \mathbb{E}(\delta_k) = 0$.

Bounds for sum-product DAGs

For z resulting of a multi-linear sum-product computation graph with n SR operations,

- $\Psi = \frac{\mathrm{SR}_p(z) - z}{z}$ is a martingale
- $E(\Psi) = 0$
- $|\Psi|$ is bounded by $\mathcal{O}(\sqrt{n}u_p)$ at fixed probability where n is the number of operations

- The paper gives tighter bounds depending on the operations combinations.

Preprint: `https://arxiv.org/abs/2411.13601`

Error Analysis of sum-product algorithms under stochastic rounding
de Oliveira Castro, El arar, Petit, Sohier

SR sum-product analysis gives error bounds for multi-linear algorithms:

- Inner product $\mathcal{O}(\sqrt{n}u_p)$
- Horner's polynomial evaluation $\mathcal{O}(\sqrt{n}u_p)$
- Pairwise summation $\mathcal{O}(\sqrt{\log_2 n}u_p)$
- Karatsuba multiplication $\mathcal{O}(\sqrt{\log_2 n}u_p)$

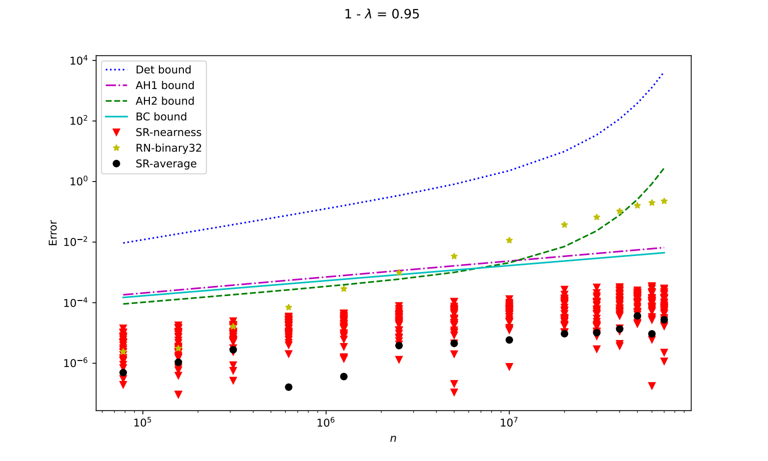


FIGURE. *Probabilistic bounds with probability* $1 - \lambda = 0.95$ *vs deterministic bound of the computed forward errors of the inner product for floating-point numbers chosen uniformly at random in* $[0;1]$ *with* $u_p = 2^{-23}$

Plan

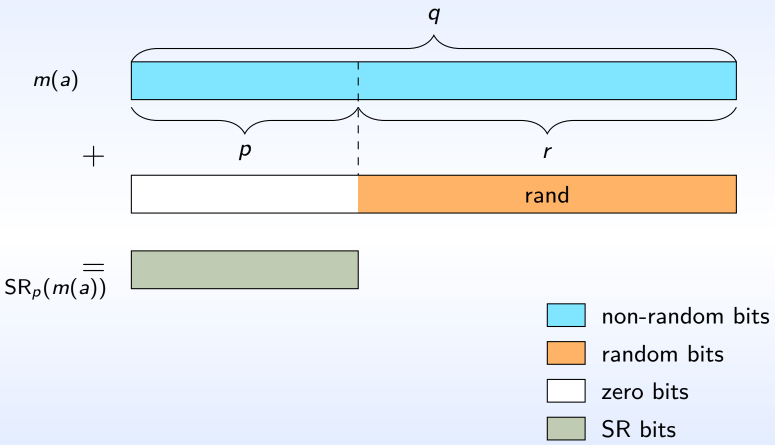
1. Error Analysis of Algorithms with Stochastic Rounding

2. Error Analysis for the Implementation of Stochastic Rounding in Hardware

How can we implement this in hardware?

Low precision

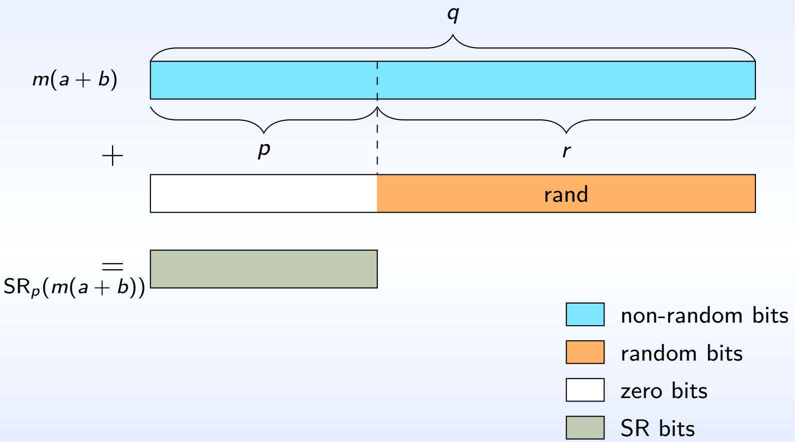
Let $a \in \mathcal{F}_q$, if we represent a in $\mathcal{F}_p$ such that $p < q$, we take $r = q - p$.



How can we implement this in hardware?

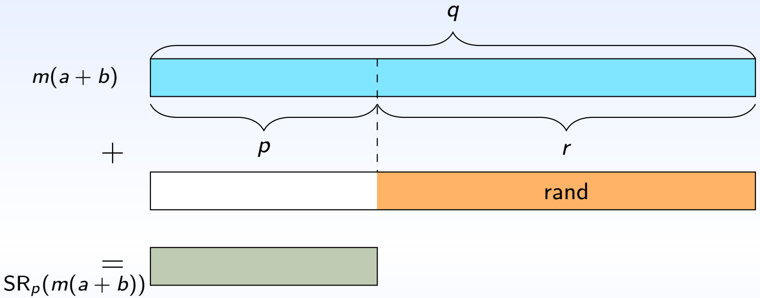
Addition

Let $a, b \in \mathcal{F}_p$, if we compute $a + b$ in $\mathcal{F}_q$ such that $p < q$, we take $r = q - p$.

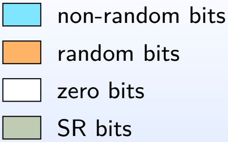


Addition

Let $a, b \in \mathcal{F}_p$, if we compute $a + b$ in $\mathcal{F}_q$ such that $p < q$, we take $r = q - p$.



A Stochastic Rounding-Enabled Low-Precision Floating-Point MAC for DNN Training
"Ali, Sami Ben and Filip, Silviu-Ioan and Sentieys, Olivier"



Toy example: $a + b$

$$a = 1.111_{(2)} \cdot 2^{8} \qquad \xrightarrow{\text{significand alignment}} \qquad a = 1.111000_{(2)} \cdot 2^{8}$$

$$b = 1.101_{(2)} \cdot 2^{5} \qquad \qquad \qquad \qquad \qquad \qquad b = \underline{0.001101_{(2)} \cdot 2^{8}}$$

significand addition

$$a + b = 10.000101_{(2)} \cdot 2^{8}$$

$$\Big\downarrow \text{ normalization}$$

$$a + b = 1.0000101_{(2)} \cdot 2^{9}$$

$$\Big\downarrow \text{ add random bits}$$

$$a + b = 1.0000101_{(2)} \cdot 2^{9}$$

$$\text{rand} = \underline{0.0001101_{(2)} \cdot 2^{9}} \quad \text{round}$$

$$\mathrm{SR}(a + b) = 1.001_{(2)} \cdot 2^{9}$$

Toy example: $a+b$

$$a = 1.111_{(2)} \cdot 2^8 \qquad \xrightarrow{\text{significand alignment}} \qquad a = 1.111000_{(2)} \cdot 2^8$$
$$b = 1.101_{(2)} \cdot 2^5 \qquad \qquad b = \underline{0.001101_{(2)} \cdot 2^8}$$

significand addition $\quad a+b = 10.000101_{(2)} \cdot 2^8$

normalization $\downarrow$

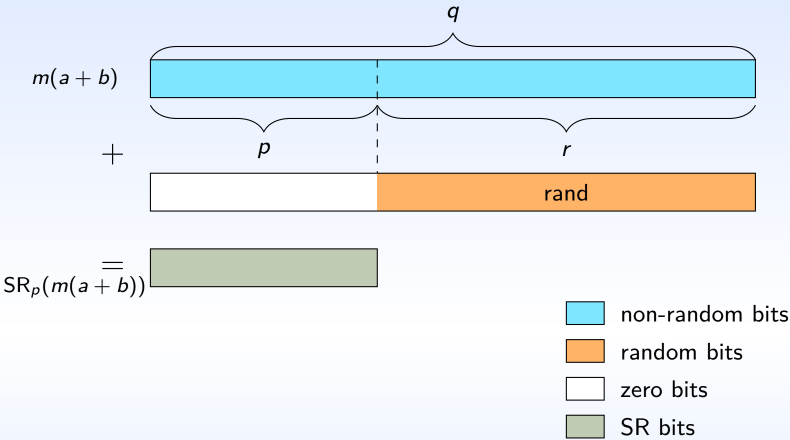
$$a+b = 1.0000101_{(2)} \cdot 2^9$$

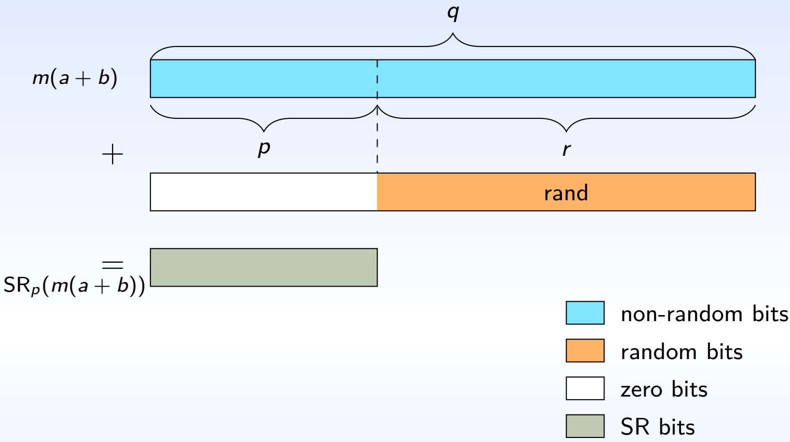
add random bits $\downarrow$

$$a+b = 1.0000101_{(2)} \cdot 2^9$$
$$\text{rand} = \underline{0.0000110_{(2)} \cdot 2^9} \quad \text{round}$$

$$\mathrm{SR}(a+b) = 1.000_{(2)} \cdot 2^9$$

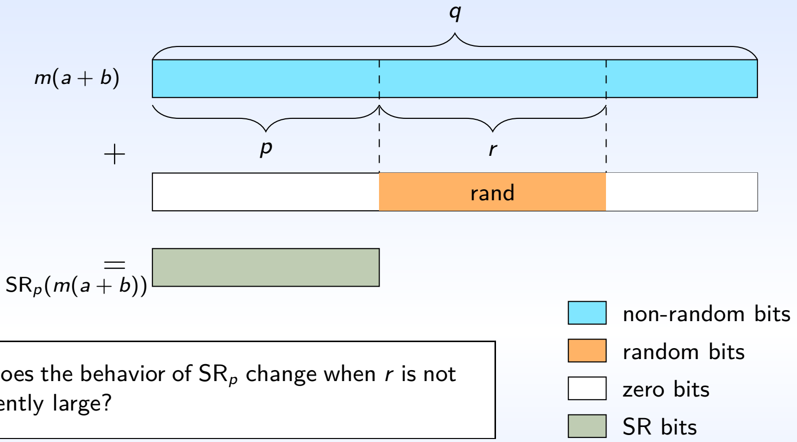
How can we implement this in hardware?




q
$m(a+b)$
p
r
$+$
rand
$=$
$\mathrm{SR}_p(m(a+b))$
non-random bits
random bits
zero bits
SR bits

Expensive!!

How can we implement this in hardware?



How does the behavior of SR_p change when r is not sufficiently large?

Expensive!!

Limited-precision stochastic rounding

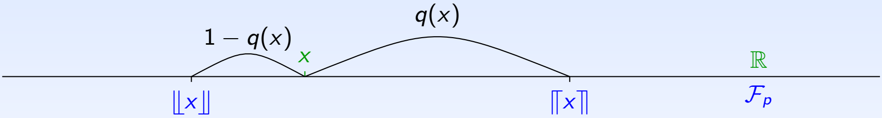


FIGURE. SR_p *with* $q(x) = \frac{x - \llbracket x \rrbracket}{\llbracket x \rrbracket - \llbracket x \rrbracket}$

Limited-precision stochastic rounding

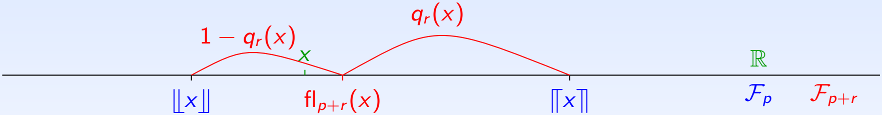


FIGURE. $SR_{p,r}$ *with* $q_r(x) = \frac{\mathrm{fl}_{p+r}(x) - \llfloor x \rrfloor}{\llceil x \rrceil - \llfloor x \rrfloor}$

Limited-precision stochastic rounding

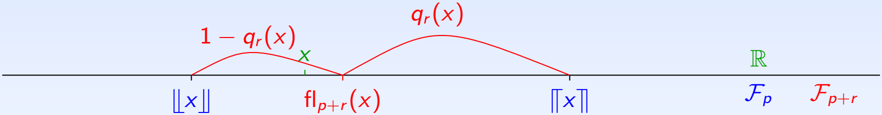


FIGURE. *$SR_{p,r}$ with* $q_r(x) = \frac{\mathrm{fl}_{p+r}(x) - \lfloor\!\lfloor x \rfloor\!\rfloor}{\lceil\!\lceil x \rceil\!\rceil - \lfloor\!\lfloor x \rfloor\!\rfloor}$

- $\mathrm{SR}_{p,r}(x) = x(1+\delta),\ |\delta| \leqslant u_p \quad \neq \quad \mathrm{fl}_{p+r}(x) = x(1+\beta),\ |\beta| \leqslant u_{p+r}$
- $\mathbb{E}\left(\mathrm{SR}_{p,r}(x)\right) = q_r(x)\lceil\!\lceil x \rceil\!\rceil + (1-q_r(x))\lfloor\!\lfloor x \rfloor\!\rfloor = \mathrm{fl}_{p+r}(x)$, then

$$\mathbb{E}(\delta) = \mathbb{E}\left(\frac{\mathrm{SR}_{p,r}(x) - x}{x}\right) = \beta$$

- The mean independence is lost

$$\mathbb{E}(\delta_k \mid \delta_1, \ldots, \delta_{k-1}) = \beta_k \neq \mathbb{E}(\delta_k)$$

- β_k is a random variable and $\mathbb{E}(\beta_k) = \mathbb{E}(\delta_k)$

Doob–Meyer decomposition

Lemma 2.

Let $\delta_1, \delta_2, \ldots, \delta_n$ be random errors produced by a sequence of elementary operations using $\mathrm{SR}_{p,r}$*, and let $\beta_1, \beta_2, \ldots, \beta_n$ be their corresponding errors incurred by* fl_{p+r}*. Then, the random variables $\alpha_k = \delta_k - \beta_k$ for $1 \leqslant k \leqslant n$ are mean independent*

$$\mathbb{E}(\alpha_k \mid \alpha_1, \ldots, \alpha_{k-1}) = \mathbb{E}(\alpha_k) = 0.$$

Moreover, for all $1 \leqslant i \leqslant n$

$$\prod_{k=i}^{n}(1+\delta_k) = \prod_{k=i}^{n}(1+\alpha_k) + \mathcal{B}_i,$$

with

$$|\mathcal{B}_i| \leqslant \gamma_{n-i+1}(u_p + u_{p+r}) - \gamma_{n-i+1}(u_p)$$

Theorem 3.

For $y = \sum_{i=1}^{n} a_i b_i$ *and* $0 < \lambda < 1$*, the quantity* $SR_{p,r}(y)$ *satisfies*

$$\frac{|SR_{p,r}(y) - y|}{|y|} \leqslant \kappa(a \circ b) \left(\sqrt{u_p \gamma_{2n}(u_p)} \sqrt{\ln(2/\lambda)} + \gamma_n(u_p + u_{p+r}) - \gamma_n(u_p) \right)$$

$$= \kappa(a \circ b) \left(\sqrt{2n} \sqrt{\ln(2/\lambda)} u_p + n u_{p+r} \right) + \mathcal{O}(\|(u_p, u_{p+r})\|_2)$$

with probability at least $1 - \lambda$*.*

- It can be applied to all previous algorithms studied with SR_p

$$
\begin{tikzcd}
 & \prod_{k=i}^{n}(1+\delta_k) = \prod_{k=i}^{n}(1+\alpha_k) + \mathcal{B}_i & \\
\text{Martingale} \arrow[ur] \arrow[d] & & |\mathcal{B}_i| \leqslant \gamma_{n-i+1}(u_p + u_{p+r}) - \gamma_{n-i+1}(u_p) \arrow[ul] \arrow[d] \\
\mathcal{O}(\sqrt{n}u_p) \arrow[dr] & & \mathcal{O}(nu_{p+r}) \arrow[dl] \\
 & \mathcal{O}(\sqrt{n}u_p + nu_{p+r}) &
\end{tikzcd}
$$

- We have the same result with BC method

Rule of thumb

Lemma 4.

Theorem 3 leads to

$$\mathcal{O}(\sqrt{n}u_p + nu_{p+r})$$

we want

$$\sqrt{n}u_p > nu_{p+r}$$

we thus have this good rule of thumb

$$r \geqslant \lceil (\log_2 n)/2 \rceil$$

Numerical experiments

- Rosenbrock function **(srfloat)**
- Parameter update in deep neural network training **(MPTorch)**

- `https://github.com/MehdiElArar/srfloat`
- `https://github.com/mptorch/mptorch`

The Rosenbrock function is a non-convex function defined by

$$f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$$

with a global minimum of 0, occurring at $\mathbf{x}^\star = (1, 1)$.

The gradient descent:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - t_k \nabla f(\mathbf{x}_k)$$

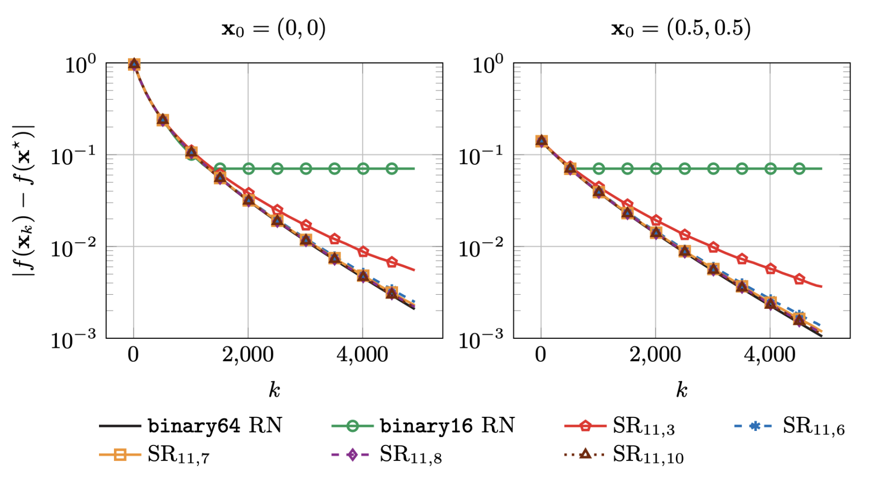


FIGURE. *Convergence profiles for 5,000 iterations of gradient descent on the Rosenbrock function. For both experiments, we average each* $SR_{11,r}$ *error over* 500 *different runs, and the learning rate is* $t_k = 0.001$.

Rosenbrock function

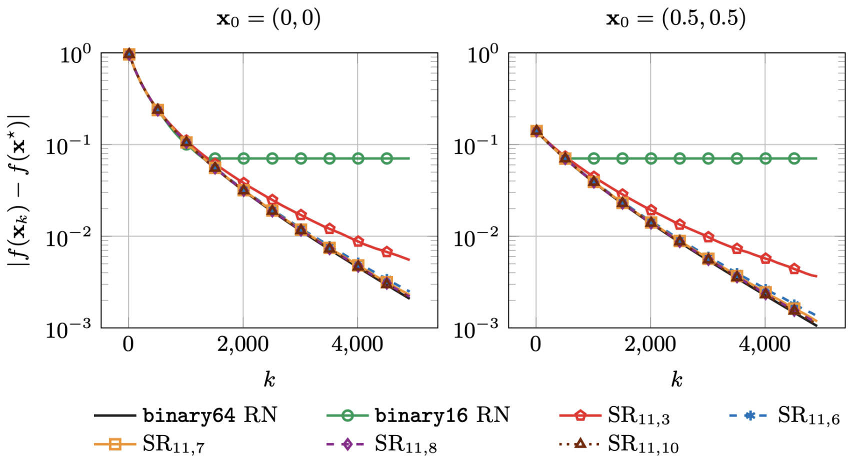


FIGURE. *Convergence profiles for 5,000 iterations of gradient descent on the Rosenbrock function. For both experiments, we average each* $SR_{11,r}$ *error over* 500 *different runs, and the learning rate is* $t_k = 0.001$.

$\lceil \log_2(5{,}000)/2 \rceil = 7$

Parameter update in deep neural network training

Focus:
Training a ResNet32[1] model on the CIFAR-10 dataset

Training Setup:

- **Hyperparameters:**
  - **Batch Size:** 128, **Momentum:** $\mu = 0.9$
  - **Total Training:** 64,000 iterations (200 epochs)
  - **Learning Rate:** $t_k = 0.1$, reduced by 10 at 32,000 and 48,000 iterations

Numerical Precision:

- **Arithmetic:** `bfloat16` ($p = 8$)
- **Update Rule:**

$$
\begin{aligned}
\mathbf{v}_{k+1} &= \circ(\mu \mathbf{v}_k + \mathbf{g}_k), \\
\mathbf{x}_{k+1} &= \circ(\mathbf{x}_k - t_k \mathbf{v}_{k+1})
\end{aligned}
$$

- **Components:**
  - $\mathbf{v}_k$: Velocity vector
  - $\mathbf{g}_k$: Gradient of the loss function

[1] Deep Residual Learning for Image Recognition

Parameter update in deep neural network training

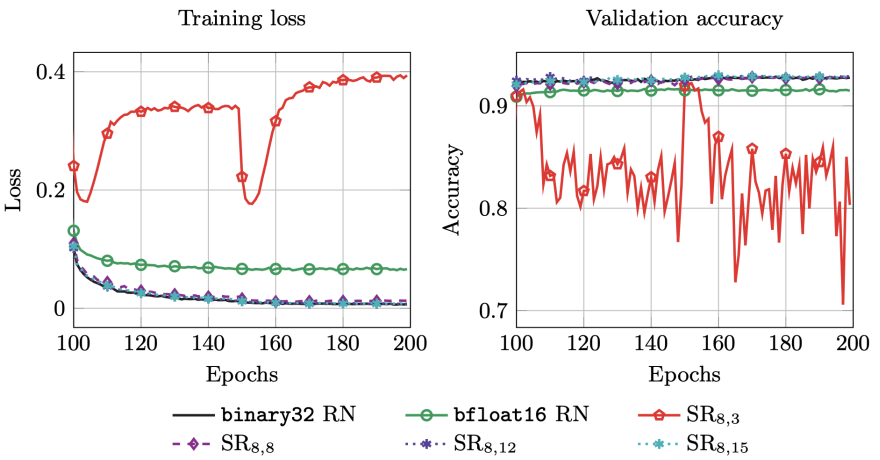


FIGURE. *In the baseline configuration,* `binary32` *arithmetic with RN is used for computing, and the same format is used for storage. For the low-precision configurations, parameters are stored and updated using* `bfloat16` *arithmetic with either RN or* $SR_{p,r}$.

Parameter update in deep neural network training

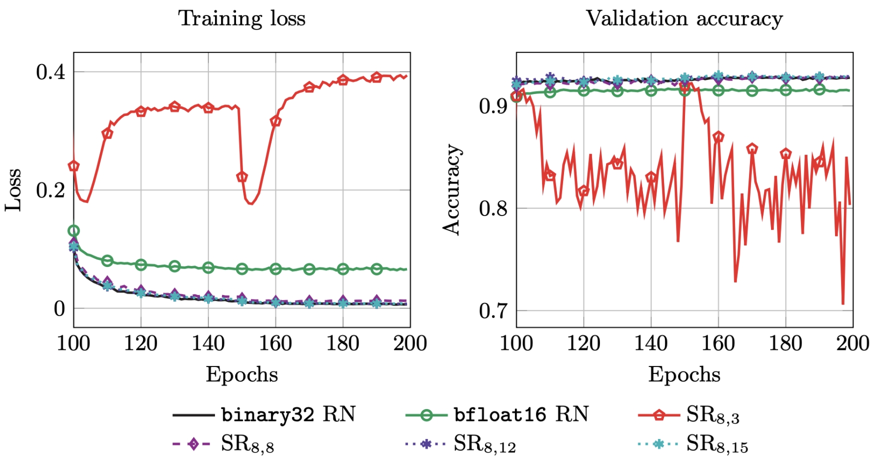


FIGURE. *In the baseline configuration,* `binary32` *arithmetic with RN is used for computing, and the same format is used for storage. For the low-precision configurations, parameters are stored and updated using* `bfloat16` *arithmetic with either RN or* $SR_{p,r}$.

$$\lceil \log_2(64{,}000)/2 \rceil = 8$$

Summary

	SR_p	$\mathrm{SR}_{p,r}$
Unbiased	✓	✗
Mean independence	✓	✗
Probabilistic bound	$\mathcal{O}(\sqrt{n}u_p)$	$\mathcal{O}(\sqrt{n}u_p + nu_{p+r})$
Rule of thumb		$r \geqslant \lceil(\log_2 n)/2\rceil$

TABLE. *Classic stochastic rounding versus limited-precision stochastic rounding*

Preprint: https://arxiv.org/abs/2408.03069

Probabilistic error analysis of limited-precision stochastic rounding
El-Mehdi El arar, Massimiliano Fasi, Silviu-Ioan Filip, Mantas Mikaitis