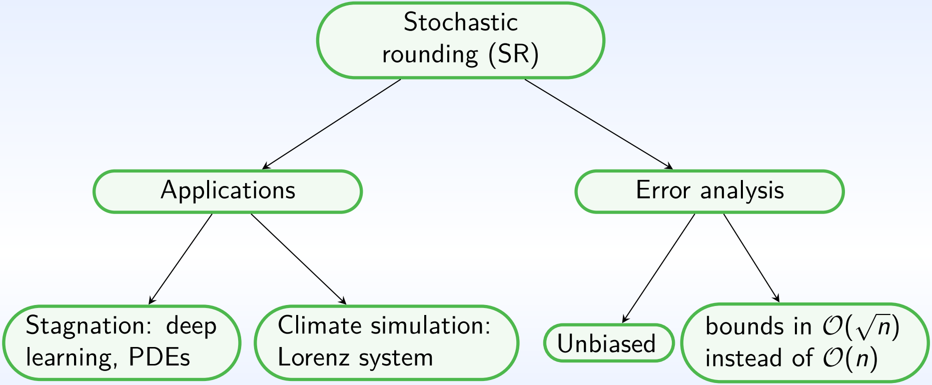
RAIM: Probabilistic error analysis of limited-precision stochastic rounding

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

Massimiliano Fasi, *Silviu-Ioan Filip* and *Mantas Mikaitis*

05/11/2024

Motivation


Stochastic rounding (SR)
Applications
Error analysis
Stagnation: deep learning, PDEs
Climate simulation: Lorenz system
Unbiased
bounds in $\mathcal{O}(\sqrt{n})$ instead of $\mathcal{O}(n)$

Stochastic rounding

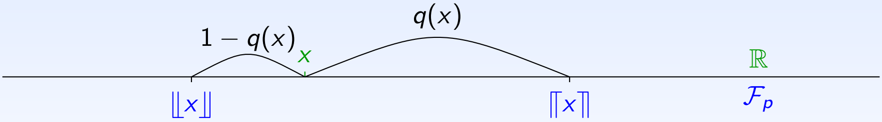


FIGURE. SR_p *with* $q(x) = \frac{x - \llbracket x \rrbracket}{\lceil\!\lceil x \rceil\!\rceil - \llbracket x \rrbracket}$

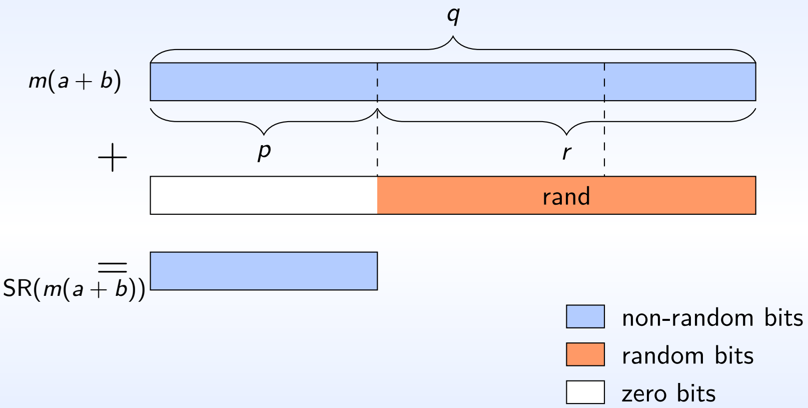
- $\mathrm{SR}_p(x) = x(1+\delta)$ such that $|\delta| \leqslant u_p$
- $\mathbb{E}\left(\mathrm{SR}_p(x)\right) = q(x)\lceil\!\lceil x \rceil\!\rceil + (1 - q(x))\llbracket x \rrbracket = x$, then $\mathbb{E}(\delta) = 0$
- SR satisfies the mean independence property

$$\mathbb{E}(\delta_k \mid \delta_1, \ldots, \delta_{k-1}) = \mathbb{E}(\delta_k) = 0$$

How can we implement this in hardware?

Example

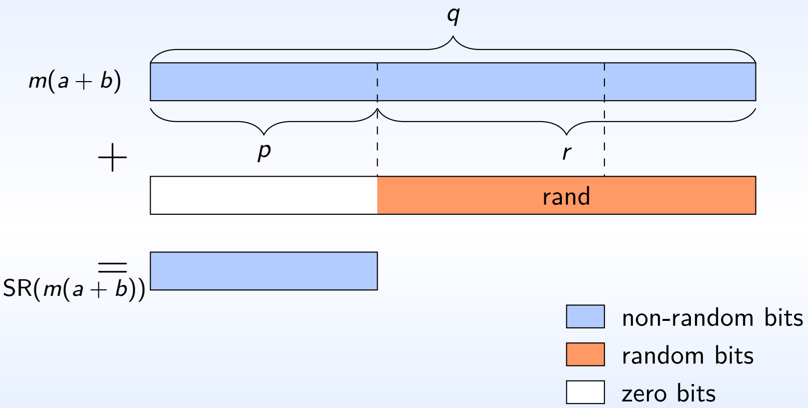
Let $a, b \in \mathcal{F}_p$, if we compute $a + b$ in $\mathcal{F}_q$ such that $q > p$, it suffices to take $r = q - p$.



How can we implement this in hardware?

Example

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

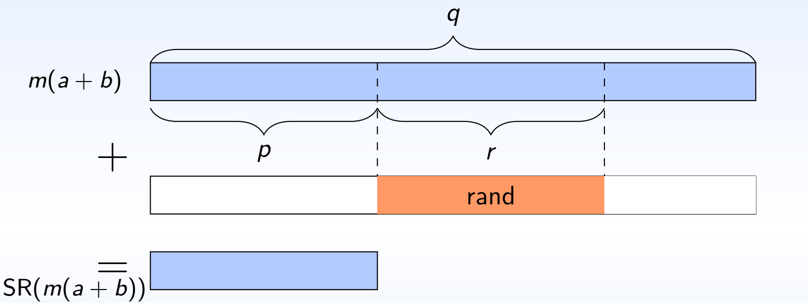


Expensive!!

How can we implement this in hardware?

Example

Let $a, b \in \mathcal{F}_p$, if we compute $a + b$ in $\mathcal{F}_q$ such that $q > p$, it suffices to 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"

Expensive!!

Limited-precision stochastic rounding

Main goal

How the behavior of SR_p changes when an infinitely precise x is not available?

Main goal

How the behavior of SR_p changes when an infinitely precise x is not available?

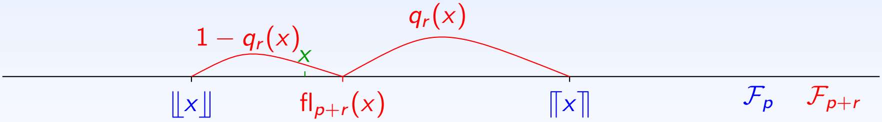


FIGURE. *$SR_{p,r}$ with* $q_r(x) = \frac{\mathrm{fl}_{p+r}(x) - \llbracket x \rrbracket}{\llbracket x \rrbracket - \llbracket x \rrbracket}$

Limited-precision stochastic rounding

Main goal

How the behavior of SR_p changes when an infinitely precise x is not available?

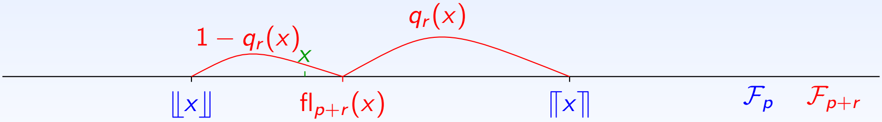


FIGURE. *$SR_{p,r}$ with $q_r(x) = \frac{\mathrm{fl}_{p+r}(x) - \llbracket x \rrbracket}{\llbracket x \rrbracket - \llbracket x \rrbracket}$*

- $\mathrm{fl}_{p+r}(x) = x(1+\beta)$ such that $|\beta| \leqslant u_{p+r} \quad \neq \quad \mathrm{SR}_{p,r}(x) = x(1+\delta)$
- $E\left(\mathrm{SR}_{p,r}(x)\right) = q_r(x)\llbracket x \rrbracket + (1 - q_r(x))\llbracket x \rrbracket = \mathrm{fl}_{p+r}(x)$
- 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 $\mathbb{E}(\beta_k) = \mathbb{E}(\delta_k)$

Main result

Lemma 1.

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

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

and $\gamma_m(x) = (1+x)^m - 1$

Error analysis of algorithms with limited-precision SR

Theorem 2.

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

Error analysis of algorithms with limited-precision SR

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

Martingale

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

$$\mathcal{O}(\sqrt{n}u_p)$$

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

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

Rule of thumb

Lemma 3.

Theorem 2 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

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:

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

Numerical experiments: Rosenbrock function

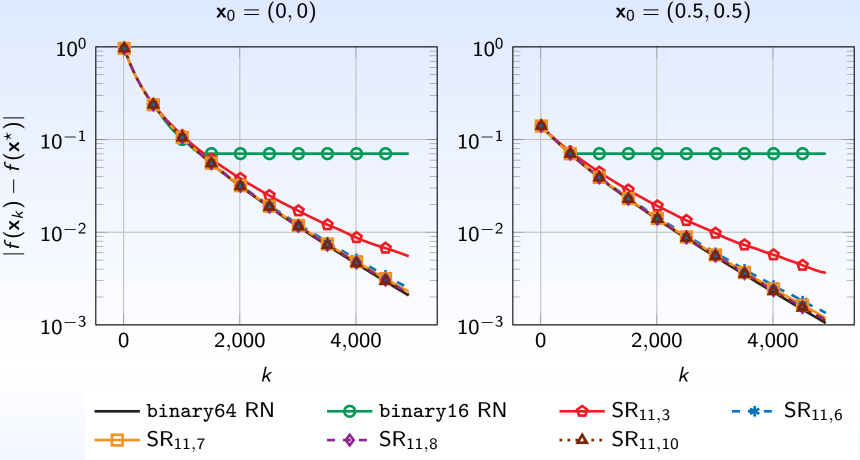


FIGURE. *Convergence profiles for 6,000 iterations of gradient descent on the Rosenbrock function. For both experiments, we average each $SR_{11,r}$ error over* 500 *different runs, and the learning rate is* $t_k = 0.001$.

Numerical experiments: Rosenbrock function

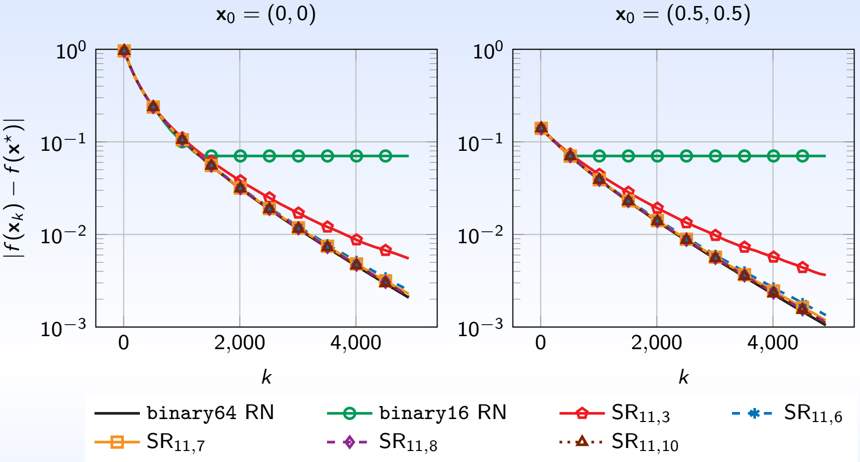


FIGURE. *Convergence profiles for 6,000 iterations of gradient descent on the Rosenbrock function. For both experiments, we average each $SR_{11,r}$ error over 500 different runs, and the learning rate is $t_k = 0.001$.*

$\lceil \log_2(6{,}000)/2 \rceil = 7$

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

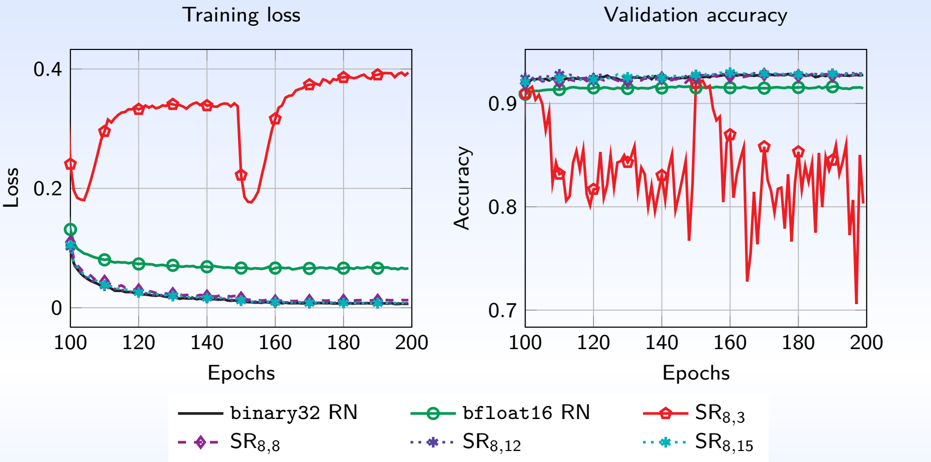


FIGURE. *In the baseline configuration,* `binary32` *arithmetic with RN is used for computing, and the same format is used for storage. For the low-precision configurations, parameters are stored and updated using* `bfloat16` *arithmetic with either RN or* $SR_{p,r}$.

Parameter update in deep neural network training

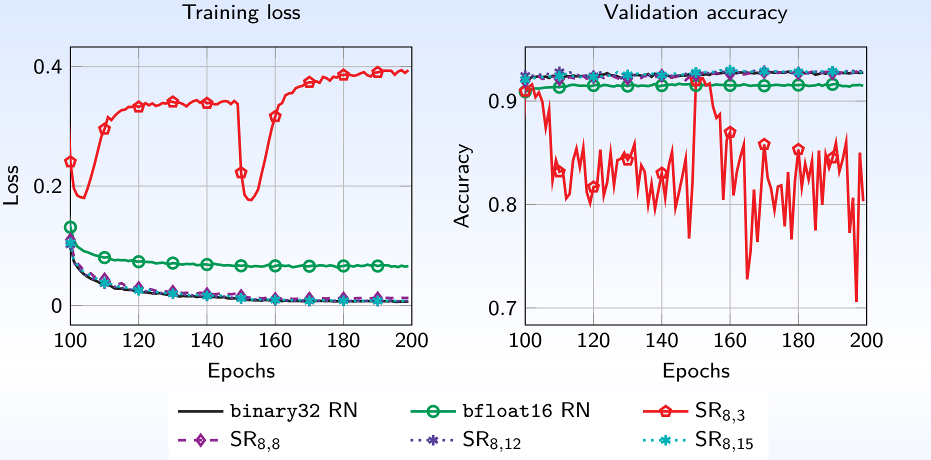


FIGURE. *In the baseline configuration,* `binary32` *arithmetic with RN is used for computing, and the same format is used for storage. For the low-precision configurations, parameters are stored and updated using* `bfloat16` *arithmetic with either RN or* $SR_{p,r}$.

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

Conclusion

	SR_p	$SR_{p,r}$
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 submitted for publication: `https://arxiv.org/abs/2408.03069`

Probabilistic error analysis of limited-precision stochastic rounding

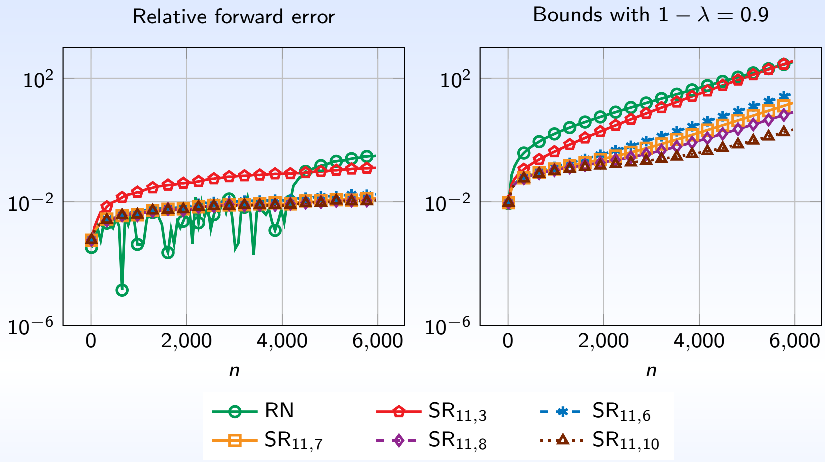


FIGURE. *The recursive summation of n floating-point numbers drawn from a uniform distribution between 0 and 1. For each value of n, the reported relative error for $SR_{11,r}$ is the average value over* 500 *runs.*

$$\lceil \log_2(6{,}000)/2 \rceil = 7$$