Towards Reliable Machine Learning

Christoph Lauter[1], Anastasia Volkova[2]

[1]Assistant Professor at University of Alaska Anchorage
[1]Associate Professor at Sorbonne, Paris, France (on leave)
[2] Associate Professsor at University of Nantes, France

CERN Data Science seminar
March 31, 2021

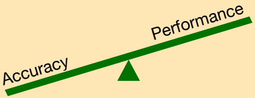
Research interests

linear algebra
interval arithmetic
floating-point
reliable computing
error analysis
fixed-point

Computer Arithmetic

static analysis
mathematical functions
code generation
IIR filters
FIR filters

Numerical kernels



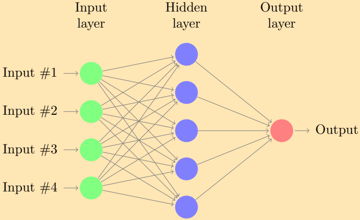
Targets

CPU
FPGA
libm
SIMD
GPU

Outline of the talk

1. Deep Neural Networks and Floating-Point
2. FP Error Analysis for DNNs
3. A Computer Arithmetic Look at DNNs
4. Making Things Concrete
5. Opening Up

Inference with Deep Neural Networks



"Cat"

- Input is fed flattened into the DNN
- The DNN's hidden layers mix the different input data
- An output layer yields the result of the DNN

☞ DNNs are huge:

- Imagenet has 27 million parameters
- BERT has 345 million parameters

High-Performance Computing to the Rescue

- DNNs are not a new invention

  ☞ Called Perceptrons in the 1960s

- DNNs are an extremely performance hungry application

  ☞ 345 million parameters for Language Processing
  Real-Time Language Processing ?

- High-Performance Computing (HPC) is the enabler of DNNs
  - Modern GPU: about $120 \cdot 10^{12}$ FLOPS per second
  - Exascale HPC Supercomputers: up to 10^{18} FLOPS per second

- HPC is centered around Floating-Point (FP) Arithmetic

  ☞ FP: a memory-efficient representation of the reals

Floating-Point Numbers

$$q_e = 10^{-19} \times 1.602176634 = \beta^E \times m$$

- Floating-Point (FP) numbers exist in different **radices** β
  ☞ $\beta = 2$ or $\beta = 10$ are common choices
- The exponent E gives the **order of magnitude**
- The **significand** $1 \leq |m| < \beta$ provides granularity
  ☞ m is a scaled integer $m = \beta^{k-1} M$, $M \in \mathbb{Z}$
- The number k of digits in M is called the **precision**
  ☞ Precision k is fixed for a FP format
  ☞ **Finite representation of real numbers**

Computing Right with Error

- Error is part of FP Arithmetic
  - Example: $x = 1.0 \in \mathbb{F}$, $y = 3.0 \in \mathbb{F}$, $x/y = 1/3 \notin \mathbb{F}$
  - **Any finite representation of the real numbers needs rounding.**
  - The maximum round-off error depends on the FP precision k.

Computing Right with Error

- Error is part of FP Arithmetic
  - Example: $x = 1.0 \in \mathbb{F}$, $y = 3.0 \in \mathbb{F}$, $x/y = 1/3 \notin \mathbb{F}$
  - **Any finite representation of the real numbers needs rounding.**
  - The maximum round-off error depends on the FP precision k.
- Round-off Error is what makes FP Arithmetic Weird
  - Associativity: $x \oplus (y \oplus z) \neq (x \oplus y) \oplus z$
  - Absorption: $\left(1 \oplus 10^{17}\right) \ominus 10^{17} = 0$
  - Cancellation: x, y have both some accuracy, $x \ominus y$ may be plain wrong

Computing Right with Error

- Error is part of FP Arithmetic
  - Example: $x = 1.0 \in \mathbb{F}$, $y = 3.0 \in \mathbb{F}$, $x/y = 1/3 \notin \mathbb{F}$
  - **Any finite representation of the real numbers needs rounding.**
  - The maximum round-off error depends on the FP precision k.
- Round-off Error is what makes FP Arithmetic Weird
  - Associativity: $x \oplus (y \oplus z) \neq (x \oplus y) \oplus z$
  - Absorption: $\left(1 \oplus 10^{17}\right) \ominus 10^{17} = 0$
  - Cancellation: x, y have both some accuracy, $x \ominus y$ may be plain wrong
- **A FP-based HPC machine executing 10^{15} FLOPS per second...**

  ...commits 10^{15} errors per second!

Computing Right with Error

- Error is part of FP Arithmetic
  - Example: $x = 1.0 \in \mathbb{F}$, $y = 3.0 \in \mathbb{F}$, $x/y = 1/3 \notin \mathbb{F}$
  - **Any finite representation of the real numbers needs rounding.**
  - The maximum round-off error depends on the FP precision k.
- Round-off Error is what makes FP Arithmetic Weird
  - Associativity: $x \oplus (y \oplus z) \neq (x \oplus y) \oplus z$
  - Absorption: $\left(1 \oplus 10^{17}\right) \ominus 10^{17} = 0$
  - Cancellation: x, y have both some accuracy, $x \ominus y$ may be plain wrong
- **A FP-based HPC machine executing 10^{15} FLOPS per second...**

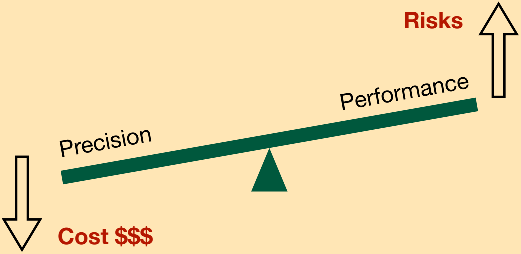
  ...commits 10^{15} errors per second!
- **These trillions of errors combine to a global error...**

  ...in non-linear, intricate ways.

Computing Right with Error

- Error is part of FP Arithmetic
  - Example: $x = 1.0 \in \mathbb{F}$, $y = 3.0 \in \mathbb{F}$, $x/y = 1/3 \notin \mathbb{F}$
  - **Any finite representation of the real numbers needs rounding.**
  - The maximum round-off error depends on the FP precision k.
- Round-off Error is what makes FP Arithmetic Weird
  - Associativity: $x \oplus (y \oplus z) \neq (x \oplus y) \oplus z$
  - Absorption: $\left(1 \oplus 10^{17}\right) \ominus 10^{17} = 0$
  - Cancellation: x, y have both some accuracy, $x \ominus y$ may be plain wrong
- **A FP-based HPC machine executing 10^{15} FLOPS per second...**
  ...commits 10^{15} errors per second!
- **These trillions of errors combine to a global error...**
  ...in non-linear, intricate ways.
- **How to Compute Right with Error ?**
  - "Crossing fingers"
  - Manual error analysis
  - Static analysis

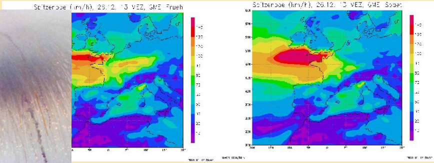
FP tradeoff



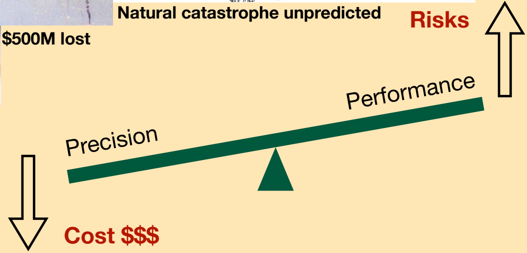
FP tradeoff

28 soldiers dead

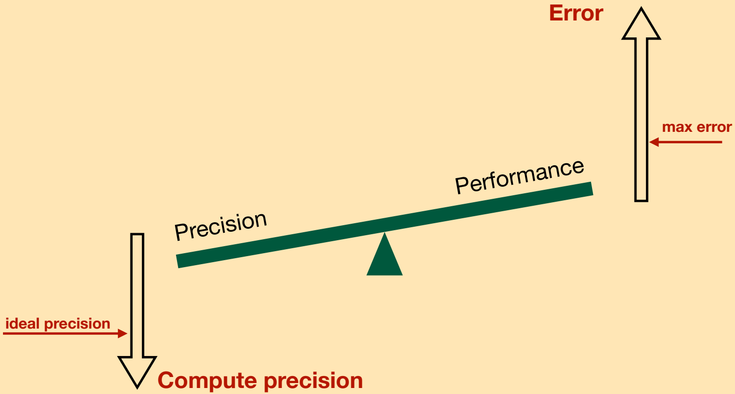
$500M lost



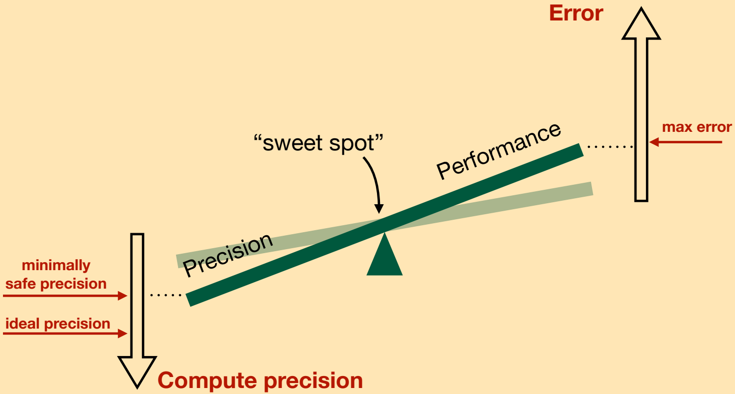
Natural catastrophe unpredicted



FP tradeoff


Error
max error
Performance
Precision
ideal precision
Compute precision

FP tradeoff


Error
max error
"sweet spot"
Performance
Precision
minimally
safe precision
ideal precision
Compute precision

DNNs and Precision

Literature Survey on DNNs and Precision

- Micikevicius, Narang et al. "Mixed precision training":
  $k = 16$ bits is enough

DNNs and Precision

Literature Survey on DNNs and Precision

- Micikevicius, Narang et al. “Mixed precision training”:
  $k = 16$ bits is enough
- Microsoft Brainwave Project:
  $k = 11$ bits is enough

DNNs and Precision

Literature Survey on DNNs and Precision

- Micikevicius, Narang et al. "Mixed precision training":
  $k = 16$ bits is enough
- Microsoft Brainwave Project:
  $k = 11$ bits is enough
- Agrawal, Mueller et al. "DLfloat: a 16-b floating-point...":
  $k = 9$ bits is enough

DNNs and Precision

Literature Survey on DNNs and Precision

- Micikevicius, Narang et al. "Mixed precision training":
  $k = 16$ bits is enough
- Microsoft Brainwave Project:
  $k = 11$ bits is enough
- Agrawal, Mueller et al. "DIfloat: a 16-b floating-point..":
  $k = 9$ bits is enough
- Henry, Tang, Heinecke. "Leveraging the bfloat16 artificial...":
  $k = 8$ bits is enough

DNNs and Precision

Literature Survey on DNNs and Precision

- Micikevicius, Narang et al. "Mixed precision training":
  $k = 16$ bits is enough
- Microsoft Brainwave Project:
  $k = 11$ bits is enough
- Agrawal, Mueller et al. "DIfloat: a 16-b floating-point..."":
  $k = 9$ bits is enough
- Henry, Tang, Heinecke. "Leveraging the bfloat16 artificial...":
  $k = 8$ bits is enough
- Banner, Nahshan, Soudry. "Post training 4-bit quantization...":
  $k = 4$ bits is enough

DNNs and Precision

Literature Survey on DNNs and Precision

- Micikevicius, Narang et al. "Mixed precision training":
  $k = 16$ bits is enough
- Microsoft Brainwave Project:
  $k = 11$ bits is enough
- Agrawal, Mueller et al. "DLfloat: a 16-b floating-point…":
  $k = 9$ bits is enough
- Henry, Tang, Heinecke. "Leveraging the bfloat16 artificial…":
  $k = 8$ bits is enough
- Banner, Nahshan, Soudry. "Post training 4-bit quantization…":
  $k = 4$ bits is enough
- Hubara, Courbiaux et al. "Binarized neural networks":
  $k = 1$ bit is enough

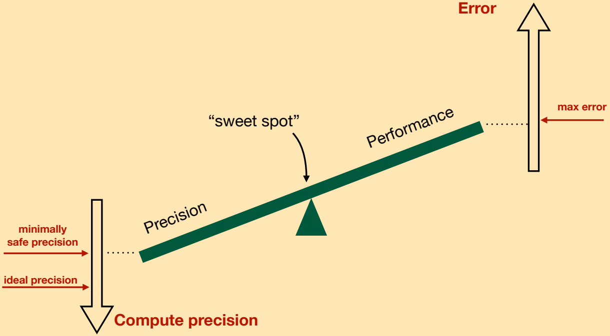
DNNs and Precision

Literature Survey on DNNs and Precision

- Micikevicius, Narang et al. "Mixed precision training":
  $k = 16$ bits is enough
- Microsoft Brainwave Project:
  $k = 11$ bits is enough
- Agrawal, Mueller et al. "DIfloat: a 16-b floating-point. . . ":
  $k = 9$ bits is enough
- Henry, Tang, Heinecke. "Leveraging the bfloat16 artificial. . . ":
  $k = 8$ bits is enough
- Banner, Nahshan, Soudry. "Post training 4-bit quantization. . . ":
  $k = 4$ bits is enough
- Hubara, Courbiaux et al. "Binarized neural networks":
  $k = 1$ bit is enough

Question: Why? How? How can $k = 4$ be enough?
Answer: We do not know. It just works.

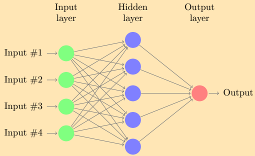
Our objective



Finding the sweet spot thru identifying the relation *precision*↔*accuracy*

- ☞ reliably
- ☞ automatically
- ☞ for large DNNs

Ingredients to DNNs



- Basic computational units of DNNs: **neurons**
- Each neuron computes $y = g(w \cdot x + b)$
  - Input $x \in \mathbb{R}$, Output $y \in \mathbb{R}$
  - Weight $w \in \mathbb{R}$, Bias $b \in \mathbb{R}$
  - Activation function $g : \mathbb{R} \to \mathbb{R}$
- Generalization: $w \cdot x + b$ becomes a matrix or tensor operation
- Computational Layers:
  - Essentially tensor operations with dot-products
  - May change the dimension of the output vector y w.r.t x
  - Maximum/minimum operations used as well
  - Extensive theory on FP for linear algebra exists
- Activation Layers:
  - Activation Functions must be non-linear
  - Activation Functions are differentiable
  - Several common choices

Activation Layers

- Activation Layers take input $x \in \mathbb{R}^m$ and produce output $y \in \mathbb{R}^m$
- Sigmoid function

$$y_i = \sigma(x_i) = \frac{1}{1 + e^{-x_i}} \quad \in [0; 1]$$

- Hyperbolic tangent

$$y_i = \tanh(x_i) \quad \in [-1; 1]$$

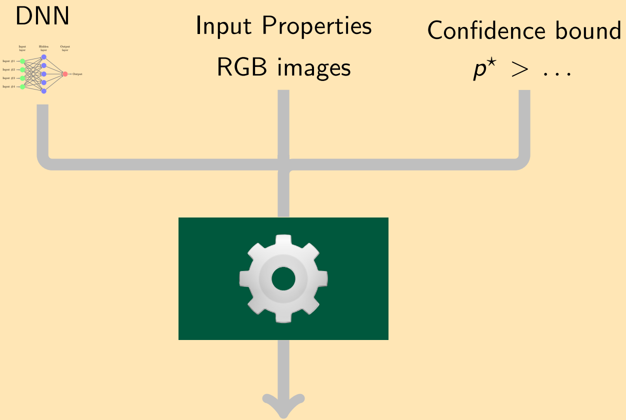
- Softmax

$$y_i = \text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{m} e^{x_j}} \quad \in [0; 1]$$

- Rectified Linear Unit

$$y_i = \text{ReLU}(x_i) = \max\left\{x_i, 0\right\}$$

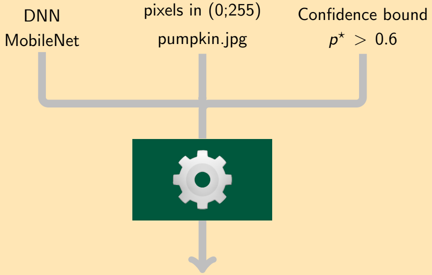
Towards Automatic Error Analysis of DNNs



- Maximum roundoff error is $\overline{\varepsilon} = \dots$
- **FP precision $k = \dots$ is enough to avoid misclassification**
- . . . (other properties)

Spoiler Alert

- FP data type is replaced with our custom type
- Error quantities are computed with interval arithmetic
- Output error is a multiple of u, set the one you like
- A whole class is analyzed, not just a point-image



- Maximum relative error is $\overline{\varepsilon} = 11.5\,u$
- FP precision $k = 8$ is enough to avoid misclassification
- . . . (other properties)

What will need to go into the heavy machinery?

What does the automatic analysis tool need to do?

- Find out how much round-off error is generated by each operation
  ☞ How does round-off behave as a function of precision k?
- Accumulate the round-off errors
  ☞ Not every round-off error is equally important eventually
- Rigorously enclose the ranges of each variable in the code
  ☞ This will allow errors be weighted correctly

How can we relate round-off error to misclassification?

- Pretty straightforward, we'll see later, bear with me

How can we make sure the results of the automatic tool are sane?

- Manual analysis of certain DNN layers required

Error of One FP Operation

- FP Operations behave as if computed exactly and then rounded:

$$x \oplus y = \diamond(x + y), \quad x \ominus y = \diamond(x - y), \quad x \otimes y = \diamond(x \times y), \dots$$

(Disclaimer: nothing on this slide is new science, see Wilkinson, Higham, Muller, Rump)

Error of One FP Operation

- FP Operations behave as if computed exactly and then rounded:

$$x \oplus y = \diamond(x + y), \quad x \ominus y = \diamond(x - y), \quad x \otimes y = \diamond(x \times y), \ldots$$

- The unit roundoff for precision k is $\mathrm{u} = 2^{-k+1}$

(Disclaimer: nothing on this slide is new science, see Wilkinson, Higham, Muller, Rump)

Error of One FP Operation

- FP Operations behave as if computed exactly and then rounded:

$$x \oplus y = \diamond(x + y), \quad x \ominus y = \diamond(x - y), \quad x \otimes y = \diamond(x \times y), \ldots$$

- The unit roundoff for precision k is $\mathrm{u} = 2^{-k+1}$
- The relative error bound due to the rounding

$$\varepsilon = \left| \frac{\diamond(x)}{x} - 1 \right| u^{-1} \leq 1/2$$

(Disclaimer: nothing on this slide is new science, see Wilkinson, Higham, Muller, Rump)

Error of One FP Operation

- FP Operations behave as if computed exactly and then rounded:

$$x \oplus y = \diamond(x + y), \quad x \ominus y = \diamond(x - y), \quad x \otimes y = \diamond(x \times y), \ldots$$

- The unit roundoff for precision k is $\mathrm{u} = 2^{-k+1}$
- The relative error bound due to the rounding

$$\varepsilon = \left| \frac{\diamond(x)}{x} - 1 \right| u^{-1} \leq 1/2$$

- Hence $\exists\, \varepsilon \in [-1/2, 1/2]$ s.t.

$$x \odot y = (x \circ y) \cdot (1 + \varepsilon\, \mathrm{u}), \quad \text{for } \circ \in \{+, -, \times, /\}$$

(Disclaimer: nothing on this slide is new science, see Wilkinson, Higham, Muller, Rump)

Error of One FP Operation

- FP Operations behave as if computed exactly and then rounded:

$$x \oplus y = \diamond(x + y), \quad x \ominus y = \diamond(x - y), \quad x \otimes y = \diamond(x \times y), \ldots$$

- The unit roundoff for precision k is $\mathrm{u} = 2^{-k+1}$
- The relative error bound due to the rounding

$$\varepsilon = \left| \frac{\diamond(x)}{x} - 1 \right| u^{-1} \leq {}^{1}\!/_{2}$$

- Hence $\exists\, \varepsilon \in [-{}^{1}\!/_{2}, {}^{1}\!/_{2}]$ s.t.

$$x \odot y = (x \circ y) \cdot (1 + \varepsilon\, \mathrm{u})\,, \quad \text{for } \circ \in \{+, -, \times, /\}$$

- Spoiler: we do not have 1 operation, but n –lots of– operations

$$n = 2 :\ x \otimes (y \otimes z) = (x \times y \times z) \cdot \left(1 + \varepsilon_1\, \mathrm{u} + \varepsilon_2\, \mathrm{u} + \varepsilon_1\, \varepsilon_2\, \mathrm{u}^2\right)$$

(Disclaimer: nothing on this slide is new science, see Wilkinson, Higham, Muller, Rump)

Combined Errors

Solution: use Affine Arithmetic (AA)

- Annotate each FP quantity $\hat{q}$ in a code with a bound $\overline{\varepsilon}$ on its error:

$$\hat{q} = q \cdot (1 + \varepsilon \, \mathrm{u}) \,, \quad \text{with } |\varepsilon| \leq \overline{\varepsilon}$$

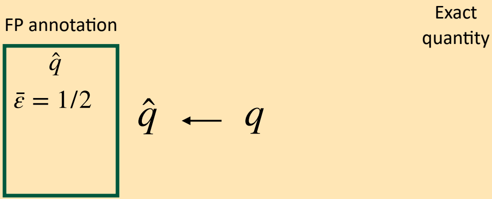
Combined Errors

Solution: use Affine Arithmetic (AA)

- Annotate each FP quantity $\hat{q}$ in a code with a bound $\overline{\varepsilon}$ on its error:

$$\hat{q} = q \cdot (1 + \varepsilon\, \mathrm{u}), \quad \text{with } |\varepsilon| \leq \overline{\varepsilon}$$

- If $\hat{q}$ stems from an exact quantity q, then simply set $\overline{\varepsilon} = 1/2$



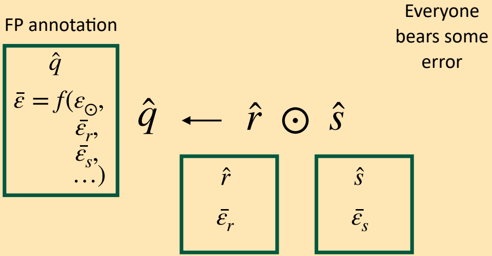
Combined Errors

Solution: use Affine Arithmetic (AA)

- Annotate each FP quantity $\hat{q}$ in a code with a bound $\overline{\varepsilon}$ on its error:

$$\hat{q} = q \cdot (1 + \varepsilon\, \mathrm{u}) , \quad \text{with } |\varepsilon| \leq \overline{\varepsilon}$$

- If $\hat{q}$ stems from an exact quantity q, then simply set $\overline{\varepsilon} = 1/2$
- If $\hat{q}$ is the result of an FP operation on quantities $\hat{r}$ and $\hat{s}$
  ☞ both $\hat{r}$ and $\hat{s}$ are annotated with $\overline{\varepsilon}_r$ and $\overline{\varepsilon}_s$
  ☞ propagate their bounds and integrate the operation's error bound



Error analysis example

For example, for an addition operation, which has an error $\varepsilon_\odot$, we obtain

$$\begin{aligned}\hat{q} &= \hat{r} \oplus \hat{s} = (\hat{r} + \hat{s}) \cdot (1 + \varepsilon_\odot \, \mathsf{u}) \\ &= (r \cdot (1 + \varepsilon_r \, \mathsf{u}) + s \cdot (1 + \varepsilon_s \, \mathsf{u})) \cdot (1 + \varepsilon_\odot \, \mathsf{u}) \\ &= (r + s) \cdot \left(1 + \varepsilon_r \, \frac{r}{r+s} \, \mathsf{u} + \varepsilon_s \, \frac{s}{r+s} \, \mathsf{u}\right) \cdot (1 + \varepsilon_\odot \, \mathsf{u}) \\ &= q \cdot (1 + \varepsilon \, \mathsf{u}) \, ,\end{aligned}$$

with

$$\varepsilon = f(\varepsilon_r, \varepsilon_s, \varepsilon_\odot, r, s) = \frac{\left(1 + \varepsilon_r \, \frac{r}{r+s} \, \mathsf{u} + \varepsilon_s \, \frac{s}{r+s} \, \mathsf{u}\right) \cdot (1 + \varepsilon_\odot \mathsf{u}) - 1}{\mathsf{u}}$$

Error analysis example

$$\varepsilon = f(\varepsilon_r, \varepsilon_s, \varepsilon_\odot, r, s) = \frac{\left(1 + \varepsilon_r \, \frac{r}{r+s} \, \mathsf{u} + \varepsilon_s \, \frac{s}{r+s} \, \mathsf{u}\right) \cdot (1 + \varepsilon_\odot \mathsf{u}) - 1}{\mathsf{u}}$$

To compute it we need:

- The bounds $\overline{\varepsilon}_r$ and $\overline{\varepsilon}_s$ for $\hat{r}$ and $\hat{s}$
- The bound $\overline{\varepsilon}_\odot$ for the operation
- Bounds on the amplification terms $\frac{r}{r+s}$ and $\frac{s}{r+s}$
- What if $r + s \to 0$?
  - ☞ Yes, of course, there is no finite bound
  - ☞ But that's precisely what FP theory predicts (cancellation)

Absolute vs. Relative Errors

Relative Error

$\hat{q} = q \cdot (1 + \varepsilon\,\mathsf{u}), \quad$ with $|\varepsilon| \leq \bar{\varepsilon}$

✘ addition

Absolute vs. Relative Errors

Relative Error

$\hat{q} = q \cdot (1 + \varepsilon\, \mathsf{u})$, with $|\varepsilon| \leq \overline{\varepsilon}$

✗ addition

Absolute Error

$\hat{q} = q + \delta\, \mathsf{u}$, with $|\delta| \leq \overline{\delta}$

✓ addition

Absolute vs. Relative Errors

Relative Error

$\hat{q} = q \cdot (1 + \varepsilon\,\mathsf{u}), \quad \text{with } |\varepsilon| \leq \overline{\varepsilon}$

- ✗ addition
- ✗ subtraction
- ✓ multiplication
- ✓ division

Absolute Error

$\hat{q} = q + \delta\,\mathsf{u}, \quad \text{with } |\delta| \leq \overline{\delta}$

- ✓ addition
- ✓ subtraction
- ✗ multiplication
- ✗ division

Absolute vs. Relative Errors

Relative Error		Absolute Error	
$\hat{q} = q \cdot (1 + \varepsilon\,\mathsf{u}), \quad \text{with } \lvert\varepsilon\rvert \leq \overline{\varepsilon}$		$\hat{q} = q + \delta\,\mathsf{u}, \quad \text{with } \lvert\delta\rvert \leq \overline{\delta}$	
✗	addition	✓	addition
✗	subtraction	✓	subtraction
✓	multiplication	✗	multiplication
✓	division	✗	division

Solution: use both bounds and propagate them whenever possible.

Using Errorprone FP to Analyze the Errors of FP?

- Can we use FP arithmetic –which prone to error– to
  - Compute bounds on amplification terms like $\frac{r}{r+s}$ or
  - Evaluate formulas as the one we saw for $\overline{\varepsilon}$ for $\oplus$?

☞ No, we need something **“error-free”**

- Exact, rational arithmetic is not usable either

☞ too costly and transcendental functions in the activation layers

- Solution: use Interval Arithmetic (IA)
  - Each quantity q is an interval $\left[\underline{q}, \overline{q}\right]$
  - Operations have interval inputs and interval outputs, inclusion property
  - IA can be implemented efficiently with FP and directed roundings

$$\text{e.g.}\quad [\underline{x}, \overline{x}] \mathbin{\overline{+}} [\underline{y}, \overline{y}] = \left[\triangle\left(\underline{x} + \underline{y}\right), \triangledown\left(\overline{x} + \overline{y}\right)\right]$$

Decorrelation Effect

- IA never lies but take e.g. $x \in [-1, 1]$ in the code snippet

```
y = x;
z = x - y;
```

☞ IA yields $z \in [-2, 2]$ even if FP z is surely 0

- This is the **Decorrelation Effect**
- Absolute or relative AA share the same issue
- Solution:
  - Annotate each quantity with a unique, fresh ID
  - Copy the ID at assignments
  - Modify IA and AA to produce 0 for subtractions with same ID (resp. 1 for divisions with same ID)

☞ We call our approach

Combined Enhanced Affine Arithmetic (CAA)

A CAA Class

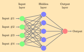
- **In a DNN, we replace the native FP type with a CAA class**

- The CAA class has the following members
  - The original FP quantity with its native FP type
  - A unique quantity ID
  - An IA bound for the quantities range
  - An absolute error bound $\overline{\delta}$ expressed as an IA interval
  - A relative error bound $\overline{\varepsilon}$ expressed as an IA interval

- The CAA class overloads all original FP operations
  - Constructors from integer types
  - Constructors from FP constants
  - Addition, Subtraction, Multiplication, Division, Square Root
  - exp, tanh, log...
  - max, min

Where are we at?

DNN

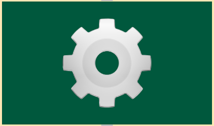


Input Properties

RGB images

Confidence bound

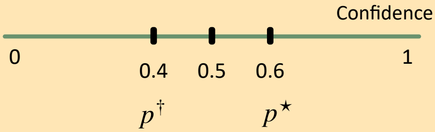
$p^\star > \dots$



- Maximum roundoff error is $\overline{\varepsilon} = \dots$
- **FP precision $k = \dots$ is enough to avoid misclassification**
- ... (other properties)

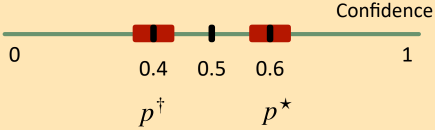
Classification Problems and Error

- Issue: Relate Error and Misclassification
- Example:
  - Simple classification DNN
  - Two classes: *Cat* and *Dog*
  - Let $p^\star$ be the maximum confidence



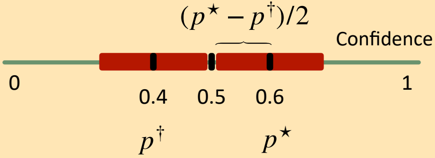
Classification Problems and Error

- Issue: Relate Error and Misclassification
- Example:
  - Simple classification DNN
  - Two classes: *Cat* and *Dog*
  - Let $p^\star$ be the maximum confidence



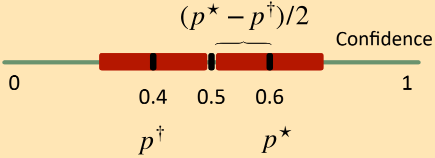
Classification Problems and Error

- Issue: Relate Error and Misclassification
- Example:
  - Simple classification DNN
  - Two classes: *Cat* and *Dog*
  - Let $p^\star$ be the maximum confidence



Classification Problems and Error

- Issue: Relate Error and Misclassification
- Example:
    - Simple classification DNN
    - Two classes: *Cat* and *Dog*
    - Let $p^\star$ be the maximum confidence



For absolute error: $$p^\star - \frac{1}{2} = \bar{\delta}\mathsf{u} = \bar{\delta}2^{-k+1}$$

For relative error: $$\frac{2p^\star - 1}{2p^\star + 1} = \bar{\delta}\mathsf{u} = \bar{\delta}2^{-k+1}$$

- **Solving for k uses the headroom for FP error while guaranteeing correct classification**

Analyzing a Softmax Layer

- Classification DNNs often end with a Softmax layer:

$$y_i = \mathrm{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{m} e^{x_j}}$$

- Let $\hat{x}_i = x_i + \delta_i$ be the FP representation of x_i.
- We get

$$\begin{aligned}
\hat{y}_i &= \frac{e^{x_i+\delta_i}}{\sum\limits_k e^{x_k+\delta_k}} \\
&= \frac{e^{x_i}}{\sum\limits_k e^{x_k} \cdot \left(1 + \frac{\sum\limits_k e^{x_k}\cdot\left(e^{\delta_k-\delta_i}-1\right)}{\sum\limits_k e^{x_k}}\right)} \\
&= y_i \cdot \left(1 + \frac{1}{1+\eta_i} - 1\right) = y_i \cdot (1+\varepsilon_i)
\end{aligned}$$

Analyzing a Softmax Layer (cont'd)

In that last term, we have

$$\eta_i = \sum_k \frac{e^{x_k}}{\sum_j e^{x_j}} \cdot \left(e^{\delta_k - \delta_i} - 1 \right).$$

This quantity is easily bounded with

$$\begin{aligned} |\eta_i| &\leq \sum_k \frac{e^{x_k}}{\sum_j e^{x_j}} \cdot \max_t \left| e^{\delta_t - \delta_i} - 1 \right| \\ &\leq \max_k \left| e^{\delta_k - \delta_i} - 1 \right|. \end{aligned}$$

Assume the δ_k and η_i mildly bounded, we get –with a Taylor development–

$$|\varepsilon_i| \leq {}^{11}\!/_{2} \max_k |\delta_k|$$

Softmax Layer: Lessons Learned

- $\hat{y}_i = \text{Softmax}(x_i + \delta_i) = \text{Softmax}(x_i) \cdot (1 + \varepsilon_i)$
  with $|\varepsilon_i| \leq {}^{11}\!/_2 \max_k |\delta_k|$
- **This is great news!**
  - Relative error is hard to achieve in FP Arithmetic
  - Absolute error is essentially trivial
  - In a DNN with a Softmax Layer, we get relative error...
  - ...for the **price of an absolute error bound**.

Softmax Layer: Lessons Learned

- $\hat{y}_i = \mathrm{Softmax}(x_i + \delta_i) = \mathrm{Softmax}(x_i) \cdot (1 + \varepsilon_i)$
  with $|\varepsilon_i| \leq {}^{11}\!/_2 \max_k |\delta_k|$

- **This is great news!**
  - Relative error is hard to achieve in FP Arithmetic
  - Absolute error is essentially trivial
  - In a DNN with a Softmax Layer, we get relative error...
  - ...for the **price of an absolute error bound**.

- Example:
  - Suppose we have $p^\star > 60\%$
  - Hence $\frac{2\,p^\star - 1}{2\,p^\star + 1} = 2^{-3.45}$
  - Equating $\varepsilon_i = 2^{-3.45}$, we get the requirement $|\delta_k| < {}^{2}\!/_{11} \cdot 2^{-3.45} \approx 2^{-6}$

  - **This means 6bit fixed-point arithmetic is enough.**
    FP can only do better.

Experimental Results

- DNNs used for Experiments
  - Digits: self-trained, handwritten digits recognition, NIST data
  - MobileNet: standard Keras Mobile Vision DNN
  - Pendulum: self-trained, DNN used as a controller for a dyn. system
- Environment
  - Intel Core i7-7500U CPU at 2.70GHz, 4 cores
  - Linux 4.19.0-13-amd64 Debian 4.19.160-2
  - gcc/g++ 8.3.0
  - MPFI 1.5.0 on MPFR 4.0.2 on GMP 6.1.2

	max absolute error in u	max relative error in u	analysis time
Digits	1.1u	3.4u	12s per class
MobileNet	22.4u	11.5u	4.2h per class
Pendulum	1.7u	–	100ms

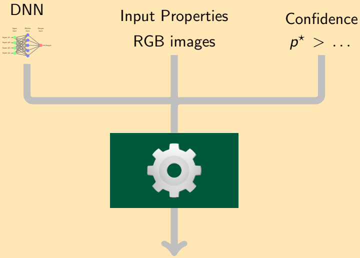
☞ $p^\star = 60\%$: precision $k = 8$ is enough for Digits

Towards Automatic Error Analysis of DNNs

What we have so far

- Front-end supports keras models
- CAA class replaces all FP computations
- Error bounds are guaranteed
- Templated arithmetics
- Computed error is a multiple of u
- A whole class is analyzed, not just a point-image

... and this also works for Fixed-Point !



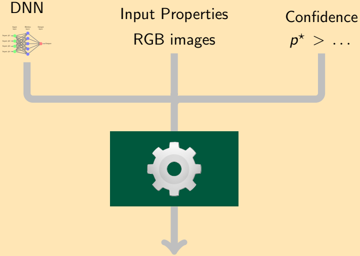
- Maximum roundoff error is $\bar{\varepsilon} = \dots$
- FP precision $k = \dots$ is enough to avoid misclassification
- . . . (other properties)

Towards Automatic Error Analysis of DNNs

What needs to be improved

- Analysis time too high
- FxP analysis needs to cope with different wordlengths
- Too much human interaction needed
- $p^\star$ value needs to be known
- Once precision is known, code still needs to be written manually

  ☞ Leverage the power of code generation



- Maximum roundoff error is $\overline{\varepsilon} = \dots$
- FP precision $k = \dots$ is enough to avoid misclassification
- ... (other properties)

Beyond this work

- Efficient and accurate deployment
  ☞ Low-precision activation functions
- Doing such analysis for training is much harder
  ☞ Towards probabilistic interval arithmetic ?
- Offending Input of a DNN:
  - e.g. DNN is fed a picture of a cat but says “Dog”
  - Dangerous for ML used for autonomous tasks like driving
- **Is there a way to rigorously determine offending inputs?**
- Polytopic approach
  - Output classes form polytopes in vector spaces
  - “Backpropagate” these polytopes to the input
  - Intersect input polytopes of two output classes
  - Issues:
    - Non-linear activation transform polytopes non-linearly
    - Explosion of dimensions and associated search-space

Beyond this work

- Industry uses Kalman Filters since the 1960ies
  ☞ Kalman Filters are a type of Machine Learning

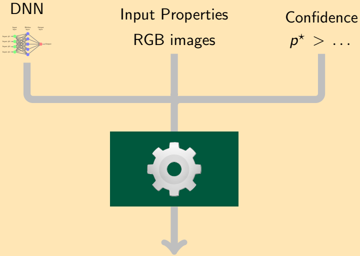
- Kalman Filter
  - Observe measurements with statistical noise over time
  - Use a model of the observed phenomenon
  - Remove part of the statistical noise of the measurments
  - Produce estimates of hidden, unobservable state variables

- Implementation of Kalman filters
  ☞ We work on code generation for LTI filters
  ☞ We analyze FxP for DNNs
  ☞ **Towards automatic FxP code generation for Kalman Filters**

Q/A

Thank you for your attention !
Questions?

arXiv preprint:
https://arxiv.org/abs/2002.03869
Tool available soon, follow
https://avolkova.org
Contact us:
anastasia.volkova@univ-nantes.fr
clauter@alaska.edu



- Maximum roundoff error is $\bar{\varepsilon} = \dots$
- FP precision $k = \dots$ is enough to avoid misclassification
- . . . (other properties)

Fixed-Point Arithmetic in Embedded Systems

- Embedded Systems often work with Fixed-Point Arithmetic (FxP)
  - FxP represents quantities as $2^E \cdot m$ as well
  - Only the **integer** significand m is stored
  - The exponent E stays **implicit**
  - The exponent E relates to the MSB position of the quantities
- FxP Operations lead to round-off error, too
  - Implicit exponents need to be unified before operations
  - Multiplications modify the implicit exponents in output
  - Integer shifts required for ajustment

    ☞ Right-shifts (>>) drop LSBs and provoke error
- FxP Arithmetic works correctly only if we can prevent overflow

☞ **Static analysis of MSB positions, error and ranges required**

Analyzing FxP

- FxP Error Analysis goes in several phases:
  1. Determine the "precision" of each variable
     ☞ Called **wordlength** *w* in FxP
  2. Determine the position of the MSB bit of each variable
     ☞ This piece of information comes **for free** as IA gives ranges
  3. Deduce the appropriate left- and right-shifts in the code
     ☞ Easy to implement in a class' methods
  4. Deduce the corresponding operation errors
  5. Propagate and accumulate the errors
     ☞ Essentially the same as for absolute error on FP

- FxP is right-aligned while FP is left-aligned
  - Error Analysis parameterized by a unit u is hard
  - **Future work** needs to address this drawback

Experimental Results for FxP

- DNNs used for Experiments
  - Digits: self-trained, handwritten digits recognition, NIST data
  - MobileNet: standard Keras Mobile Vision DNN
  - Pendulum: self-trained, DNN used as a controller for a dyn. system
- Using several FxP Arithmetic wordlengths w in the experiments

	max error for $w = 24$	max error for $w = 32$	analysis time
Digits	$6.0 \cdot 10^{-2}$	$7.3 \cdot 10^{-4}$	20s per class
MobileNet	–	–	(too long)
Pendulum	$4.5 \cdot 10^{-6}$	$1.8 \cdot 10^{-8}$	2s

☞ **Automatic** FxP MSB determination & overflow prevention with **rigorous** error bounds