Bits on Bits: Showcasing Next-Gen Arithmetic through Information Theory

Max Hawkins and Rich Vuduc

Computing Export Controls

"Who controls the ~~spice~~ compute, controls the world"

REPORTS *Mar 24, 2025* | Hudson Institute

AI, National Security, and the Global Technology Race: How US Export Controls Define the Future of Innovation

Nury Turkel

Exclusive: Nvidia modifies H20 chip for China to overcome US export controls, sources say

By **Liam Mo** and **Brenda Goh**

May 9, 2025 5:21 AM EDT · Updated May 9, 2025

Definitions of computing performance impact
billion-dollar decisions,
national security,
and the future of computing.

How to do this accurately, fairly, and generally?

Which computer is more performant?

By how much?

Computer A

- 1.44 Exaflop/s*
- Nvidia's GB200 NVL72
  - "The NVIDIA GB200 NVL72 is an exascale computer in a single rack." - https://www.nvidia.com/en-us/data-center/gb200-nvl72/

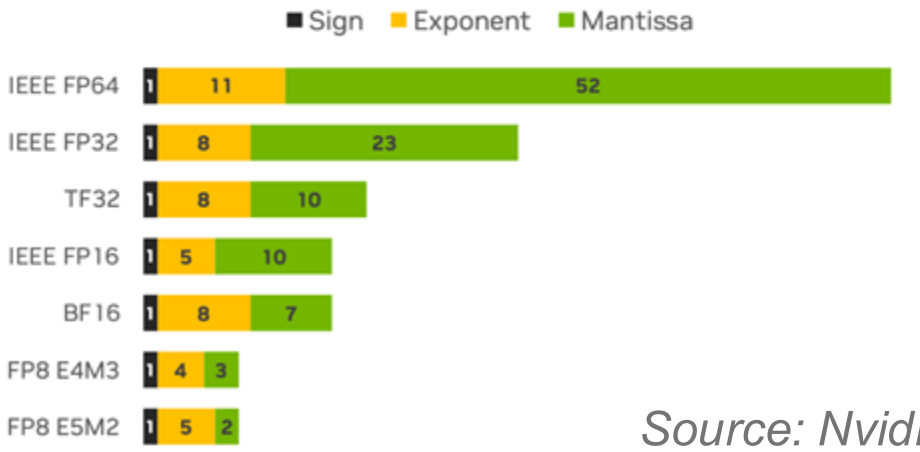
*** With sparse FP4 tensor cores**

Computer B

- 1.35 Exaflop/s**
- Frontier
  - #2 most performant public supercomputer

**** Dense FP64 with Linpack**

The Variety of Arithmetic



Source: Nvidia

- Data types
  - Bit widths
  - Bit allocation (e.g. mantissa and exponent bits)
  - Encoding schemes - integer vs floats vs posits…
  - Specifications (e.g. IEEE, OCP, vendor…)
- Operations
  - Add, subtract, negate, multiply, divide, compare, sqrt, tanh, …
  - Sparsity
  - Emulation (Ozaki and beyond)
  - Noise
  - Scalar vs vector vs matrix inputs

Hardware:

Quantum, analog, neuromorphic, reversible, …

How do we fairly measure and compare performance across this large design space?

What are we doing now?

What could we do now?

What can we do in the future?

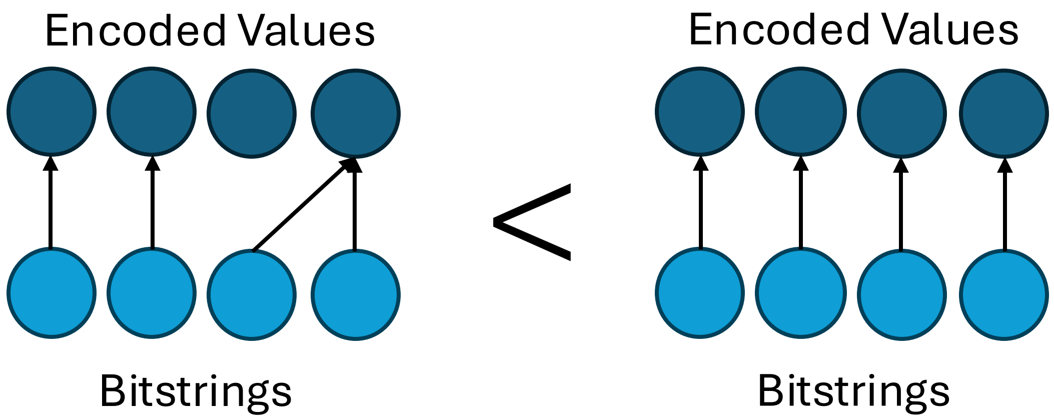
What are we doing now?

Weighting Data Types by Bit Widths

- FP4 vs FP64
  - ‘Exaflop’ computers: Frontier vs GB200 NVL72
  - 64 bits → 2^{64} possible states
  - 4 bits → 2^{4} possible states
- Linear comparisons?
  - $\frac{2^{64}}{2^{4}} = 1{,}152{,}921{,}504{,}606{,}846{,}976$
- Logarithmic comparisons?
  - $\frac{log2(2^{64})}{log2(2^{4})} = \frac{64}{4} = 16$
- We use logarithms of the state space to compare across bit widths
  - Bit width approximation
- U.S. Gov’t export controls use this approach

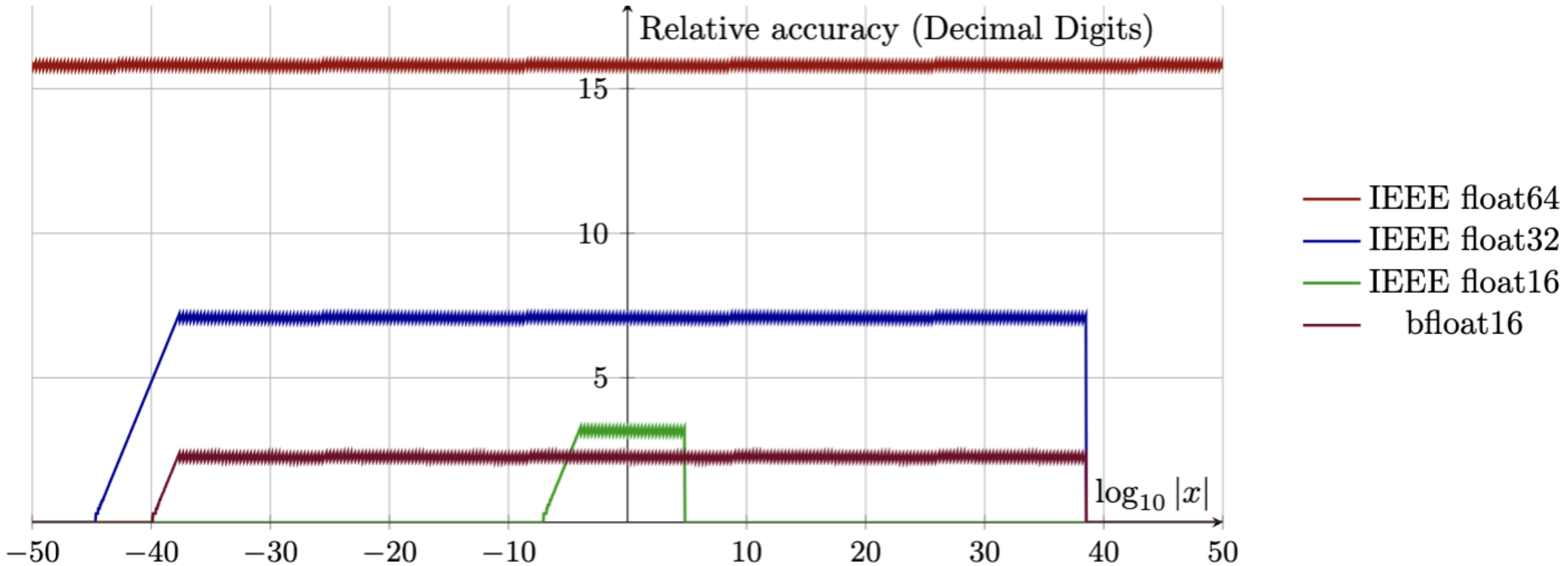
Reducing Redundant Encodings

- How many **distinct** states does a data type represent?
- Redundancy wastes bits/bitstrings
  - "There should be no redundant bit patterns to mean the same thing; every bit counts." – John Gustafson



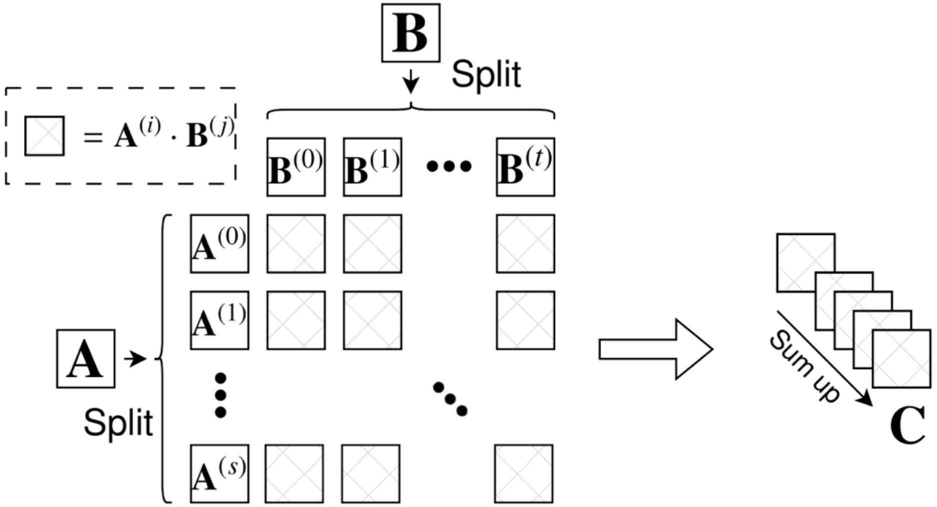
Optimizing Data Type Usage Efficiency

- "Does this kernel need FP64, or can I use FP16, Int8, FP2,...?"
  - Much existing bit inefficiency!
- Value range
  - Most data spans << 300 decades
- Accuracy
  - Many applications don't always need 53 bits of relative accuracy
  - Even HPL (see HPL-MxP)



Innovating in Data Types and Emulation

- Block-scaled encodings
- Posits, takums, ...
- Ozaki emulation
  - Performing floating-point matmul with lower-precision hardware
  - Useful when:
    - High-precision performance is low
    - Data spans a very small range and requires little accuracy



Source: https://arxiv.org/pdf/2306.11975

What are we doing now?

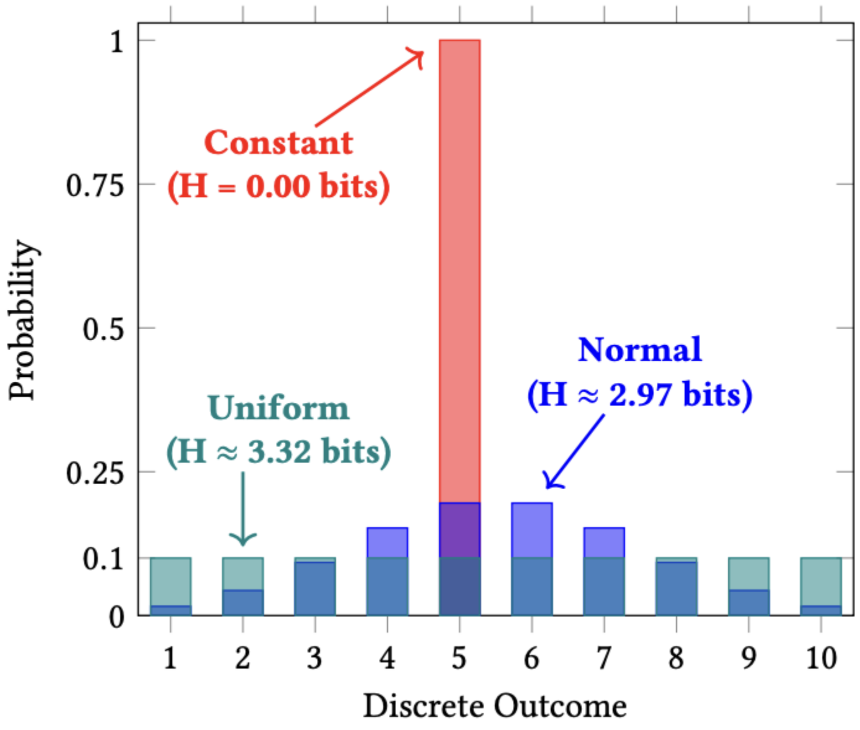
- Weighing data types by their bit widths
  - Log scaling of state space
- Reducing redundant value encodings
- Optimizing data type usage efficiency with smaller data types
  - "Does this kernel really need FP64?"
- Innovating in data types and emulation
  - Block-scaled FP, Posits, Takums, Ozaki emulation, and beyond

What could we do now?

Shannon Entropy in Brief

- Uncertainty
- Flipping a coin: Heads or Tails ➔ 1 bit
- Rolling a die with M faces ➔ $log_2(M)$

- Shannon entropy (H): A measure of uncertainty
  - Discrete random variable X with probability distribution p(x)
  - Measured in bits if b = 2



$$\mathrm{H}(X) = -\sum_{x \in \mathcal{X}} p(x) \log_b p(x)$$

Reframe Effects of Redundant Encodings with Entropy

- Redundancy reduces *information capturing potential*
  - Magnified by smaller number of bitstrings – low bit widths
- Quantified with encoding efficiency: $\eta = \frac{H_{encoding}}{bit\_width}$
  - Careful when mixing linear and log scaling
  - 50% bitstring redundant 4-bit encoding → 3 bits of entropy (not 2!)
- Example: IEEE-754 redundant NaN encodings

Every ~~physical~~ **informational** bit counts!

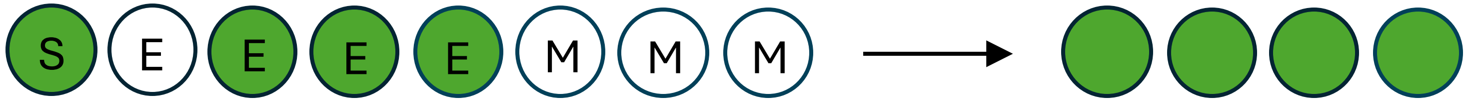
Encoding Efficiency in Practice (NaNs only)

Float Type	Bit Width	Encoding Entropy	Encoding Efficiency (%)
IEEE-754	64	63.974	99.960
IEEE-754	32	31.906	99.707
TF32	19	18.957	99.774
IEEE-754	16	15.657	97.854
BF16	16	15.969	99.806
OCP (E4M3)	8	7.992	99.902
IEEE-754 (E4M3)*	8	7.792	97.397
OCP (E2M3)	6	6	100
IEEE-754 (E2M3)*	6	5.167	86.119
IEEE-P3109	Any	Ideal	100
Posits	Any	Ideal	100
Takums	Any	Ideal	100

** Theoretical – Does not exist.*

Data Type Usage Efficiency and Info Theory

- Bits are your currency, and you allocate them as needed
  - "Do I really need FP64 for everything?"
- How to answer that analytically: Info theory
  - Bitstrings for values beyond needed range go unused → 0 entropy
  - Bits allocated towards excess precision are baggage
  - Constant values have 0 uncertainty → 0 entropy

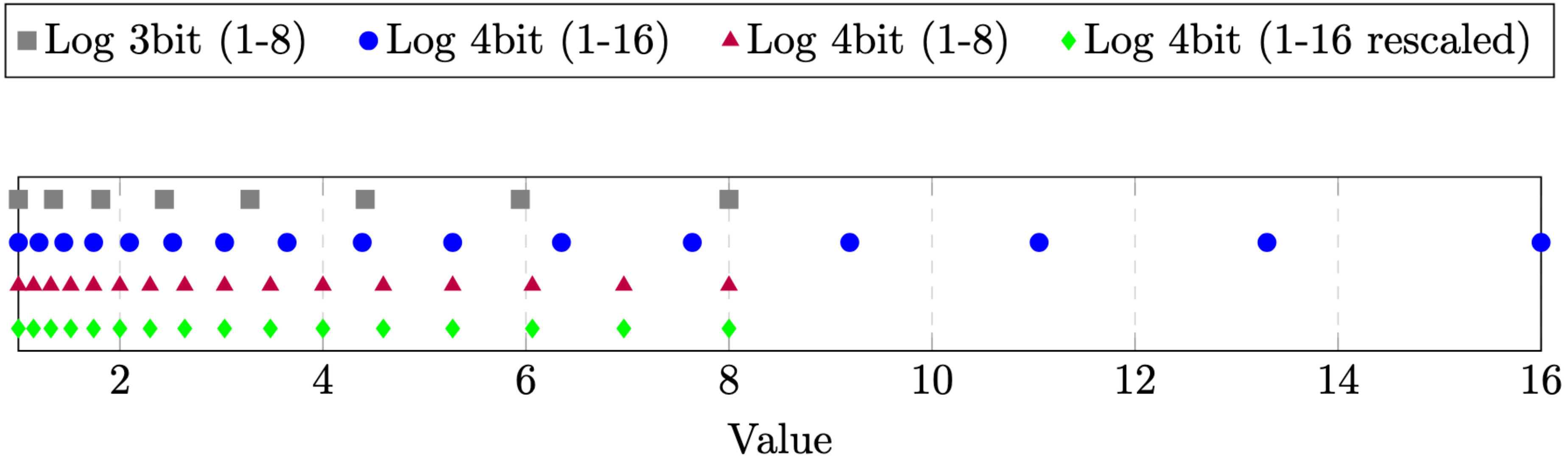


- Example: Using FP8 (E4M3) to encode 4-bit integer
  - Bit width approximation: 8 physical bits per op to 4 → 2x performance 'reduction'
  - Info Theory: 4 bits of info to 4 → No performance impact

Info theory doesn't 'punish' optimization (unlike bit width approx.)

Another Problem with *Mixed* Bit Field Mentality

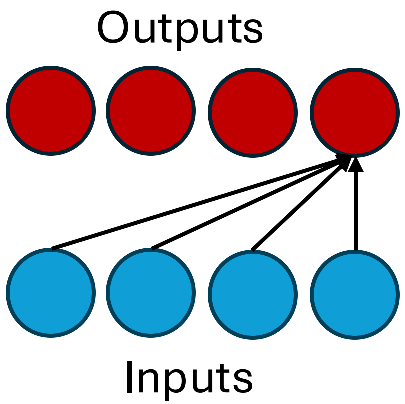
- Float encodings unnecessarily separate value range and precision!
  - Ex: BF16 vs FP16 or FP8 E4M3 vs E3M4
- In a purely log or purely integer format, these differences disappear
  - Can exactly tradeoff accuracy ←→ range
  - Simplifies reasoning about data types



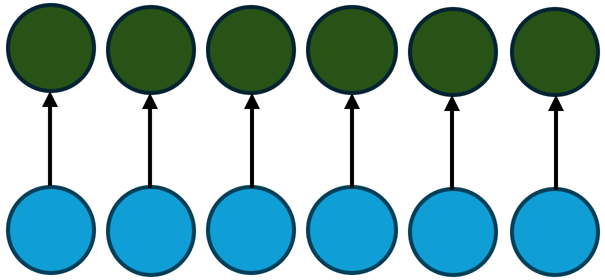
From Quantization/Communication to Operations

How much does a given operation *reduce uncertainty*?

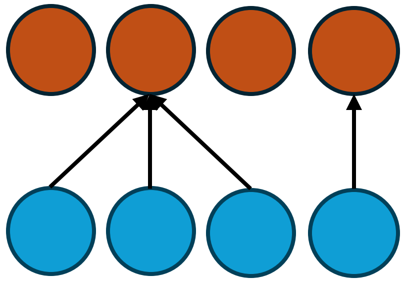
- Constant inputs or outputs → None!
  - Compile them away



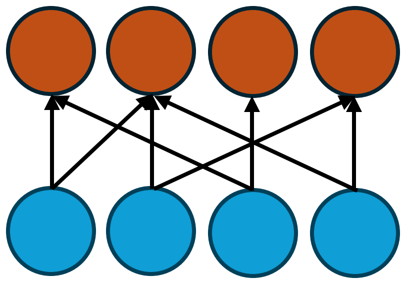
- More output states → More!



- Some states more likely → Less!
  - NaNs, overflow, underflow, etc



- Noisy/Error-prone HW → Less!
  - Determinism matters



From Quantization/Communication to Operations

"The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point”

– A Mathematical Theory of Communication (Shannon 1948)

- Generalized the concept of communication performance
    - Allowed for fair and generalized performance evaluation

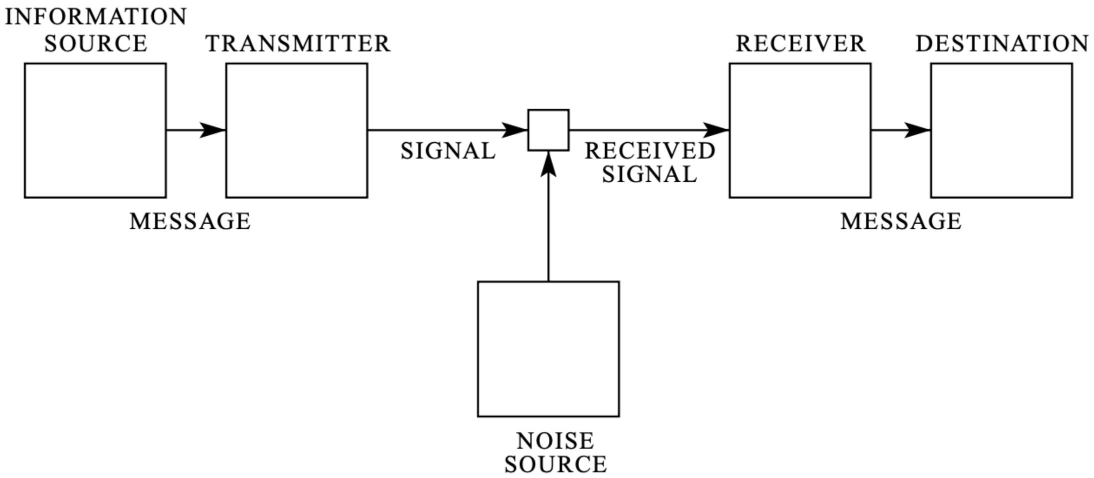
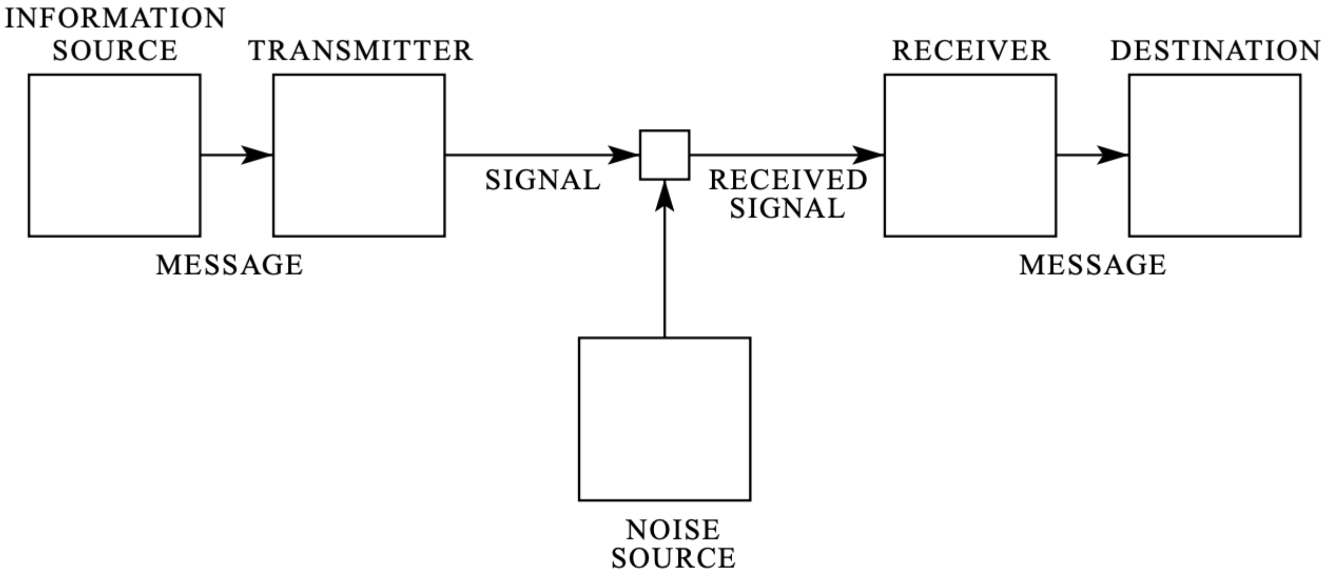
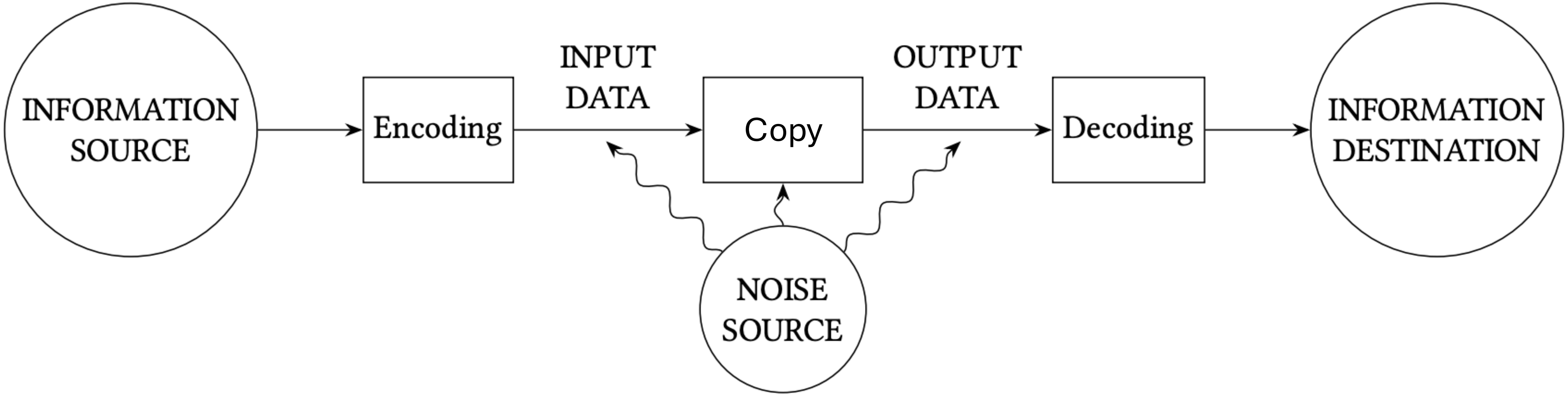


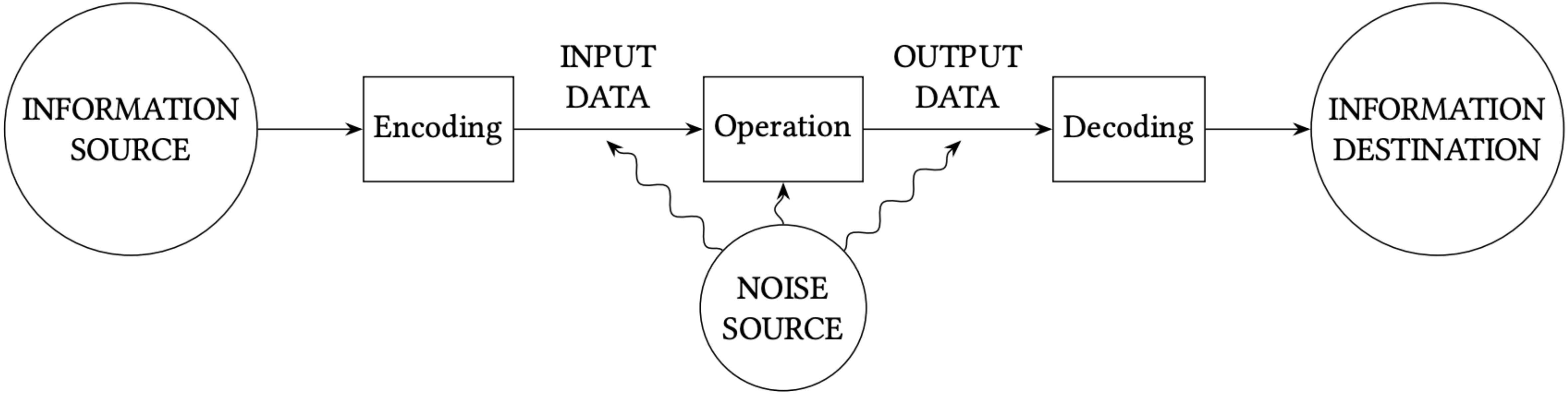
Fig. 1—Schematic diagram of a general communication system.

Communication: Copies inputs to outputs (identity operation)





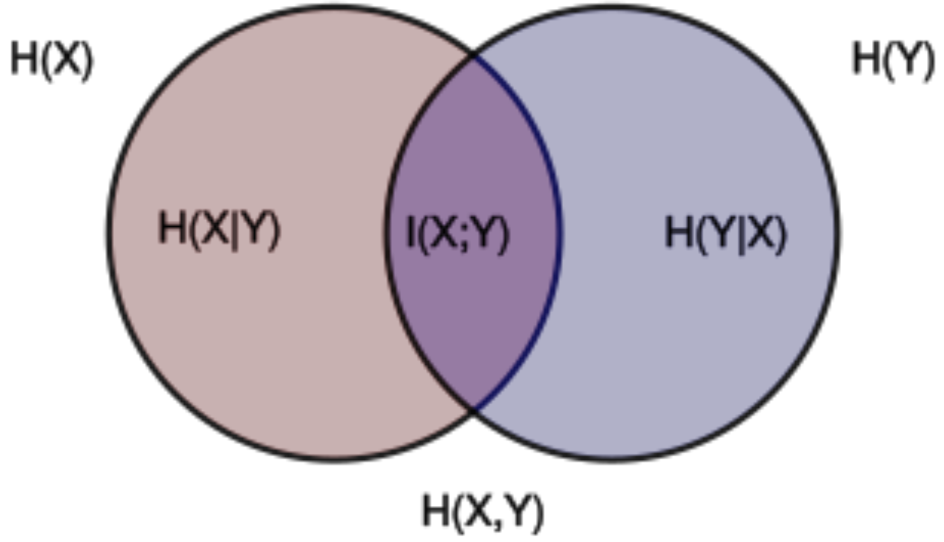
Expand beyond the identity: Computation



Mutual Information (I)

- Measures the 'shared' information of random variables
  - Considers properties of the operation and noise
- 'Operational' quantity
- Application and runtime-dependent

$$I(X;Y) = H(X) - H(X|Y)$$



Channel Capacity (C)

- Maximum MI over possible distributions
- Upper bound/ideal quantity
- Application-agnostic

$$C = \max_{p(x)} I(X;Y)$$

Info Theory and Computation

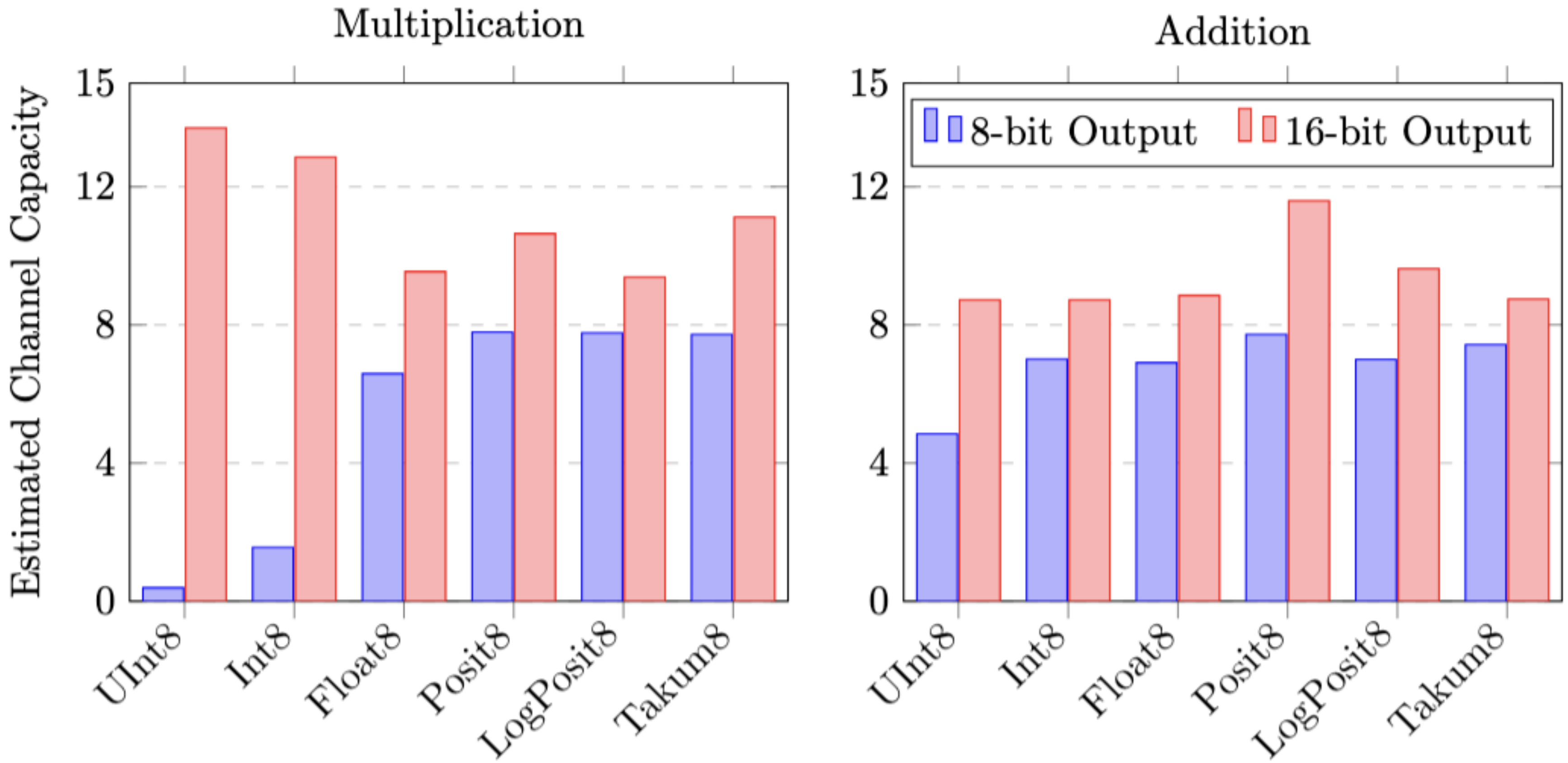
- Expand Shannon's communication performance model
  - Identity operator (communication) → Arbitrary operator (computation)
  - Mutual info and channel capacity naturally handle this
- Flop/s → Bit/s
  - Base Measure: Uncertainty reduction
  - Operational: Mutual Information
  - Ideal/Peak: Channel capacity
- Enable generalized and fair performance evaluation
  - Just like communication has had for >70 years
- Aligns with existing performance metrics

Every *informational* bit counts

(for communication, quantization, and computation)

Data Type Channel Capacity Estimation

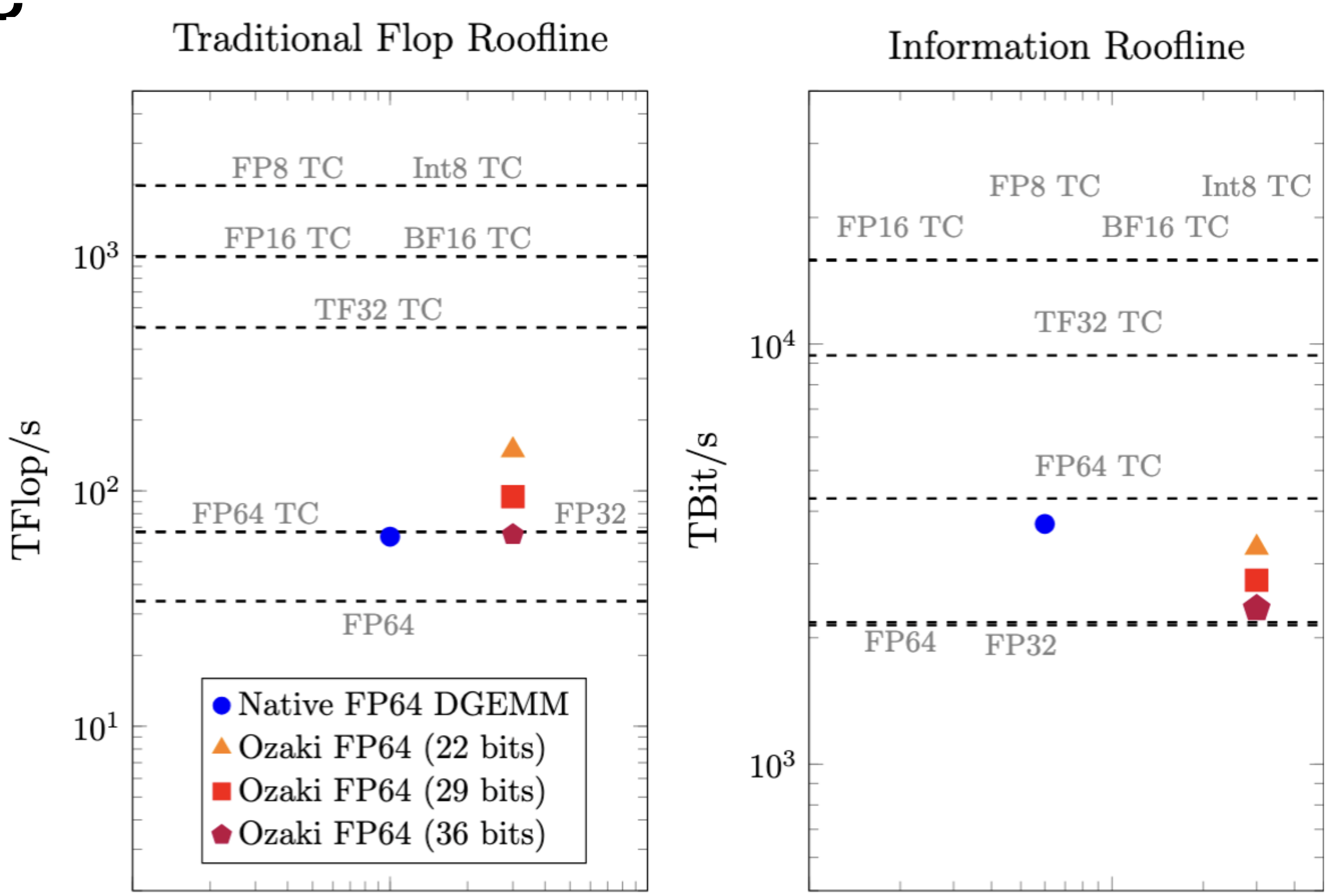
- Estimate the channel capacity of operations
- Inputs: Every possible combination of 8-bit values
- Outputs: Add/Mult in 8 or 16-bit formats



Incorporate Data Type Utilization Into Tools

Information Roofline

- Extends traditional roofline
- Adds data type utilization
  - Needed with today's innovation
  - Every bit counts!
- X-Axis: Arithmetic Intensity
  - Unitless $\left(\frac{bits_{comp}}{bits_{comm}}\right)$
- Y-Axis: Information Throughput
  - Bits of information per second



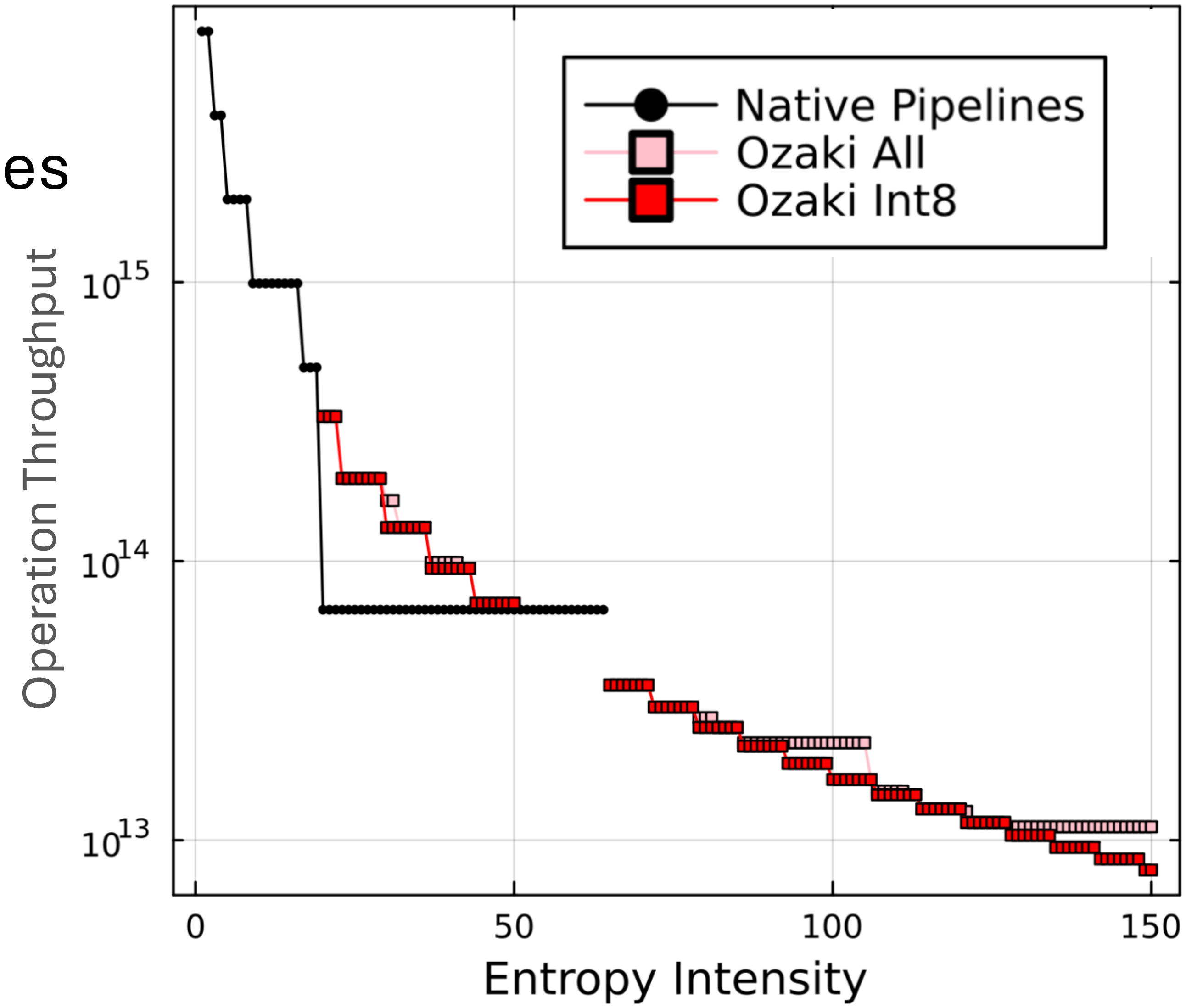
Either you're inefficiently using a data type, or you traded accuracy/range for op-throughput.

The information roofline makes this explicit.

Entropy Staircase

- Simplify complex execution choices
  - Native pipelines and emulation
- Quantify benefits of emulation
- Ease of ascending/descending the entropy staircase
  - Posits vs floats
  - Traditional 'mixed' precision

GEMM on Nvidia H200 GPU



What could we do now?

- Reframe redundant value encodings with entropy loss
- Measure data type and operation performance with info. theory
  - Unifies communication and computation
- Incorporate data type utilization into usable tools
  - Information roofline
  - Entropy staircase

What could we do in the future?

Uncorrectable Noise

This is real!
H100 GPU in space.

- Status quo: Digital computing is nearly lossless
  - Data type designers: “We don’t need to care about bit flips.”
  - Infs/NaNs/bitfields make bitflips disastrous
- Motivations for accepting error:
  - Undervolting: Energy consumption ←→ Error tradeoff
  - Space-based datacenters: Radiation causing bit flips
- Shannon capacity of graphs
  - Single bit flips: Hypercube
- Could digital encodings look more like neuromorphic spike trains?

[Image Source]()

What could we do next?

- Noise/error-tolerant formats and algorithms
- Variable-width encodings
- Dataflow optimization (hyper-localized data types/operations)
- Compare performance across hardware paradigms
    - Quantum
    - Neuromorphic
    - Analog
    - Reversible

There is still room for innovation.

When will the juice be worth the squeeze?

How will we fairly and generally measure computing performance in the future?

- Innovation will continue
  - Loss of Moore's law/Dennard scaling
- Data type and hardware variety will grow
- How many asterisks is too many?
  - *Flop/s in FP32, with sparsity, using tensor cores, emulated with Ozaki scheme 1 using two slices of Int8
- Export controls need a fundamental grounding
- HPC + Quantum/Neuromorphic/Analog/Reversible systems

We need to innovate in performance measurement and tooling to capture the evolving hardware and data types.

Information Theory Enables a Useful and General Framework for Computing Performance

Every *informational* bit counts!

4-page preprint:
arxiv.org/pdf/2508.05621

mhawkins60@gatech.edu

richie@cc.gatech.edu

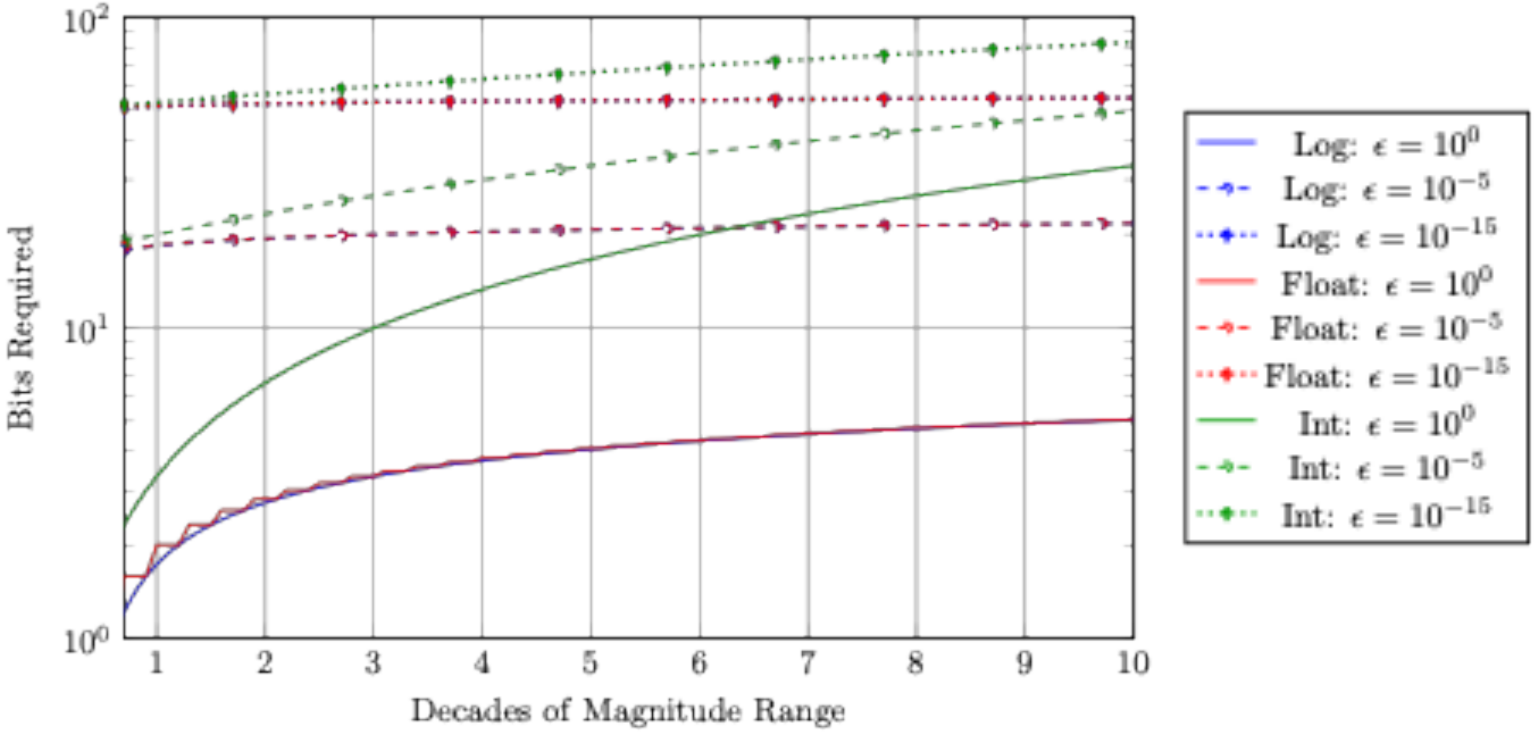
Bonus Slides!

Aspects Information Theory Explicitly Ignores

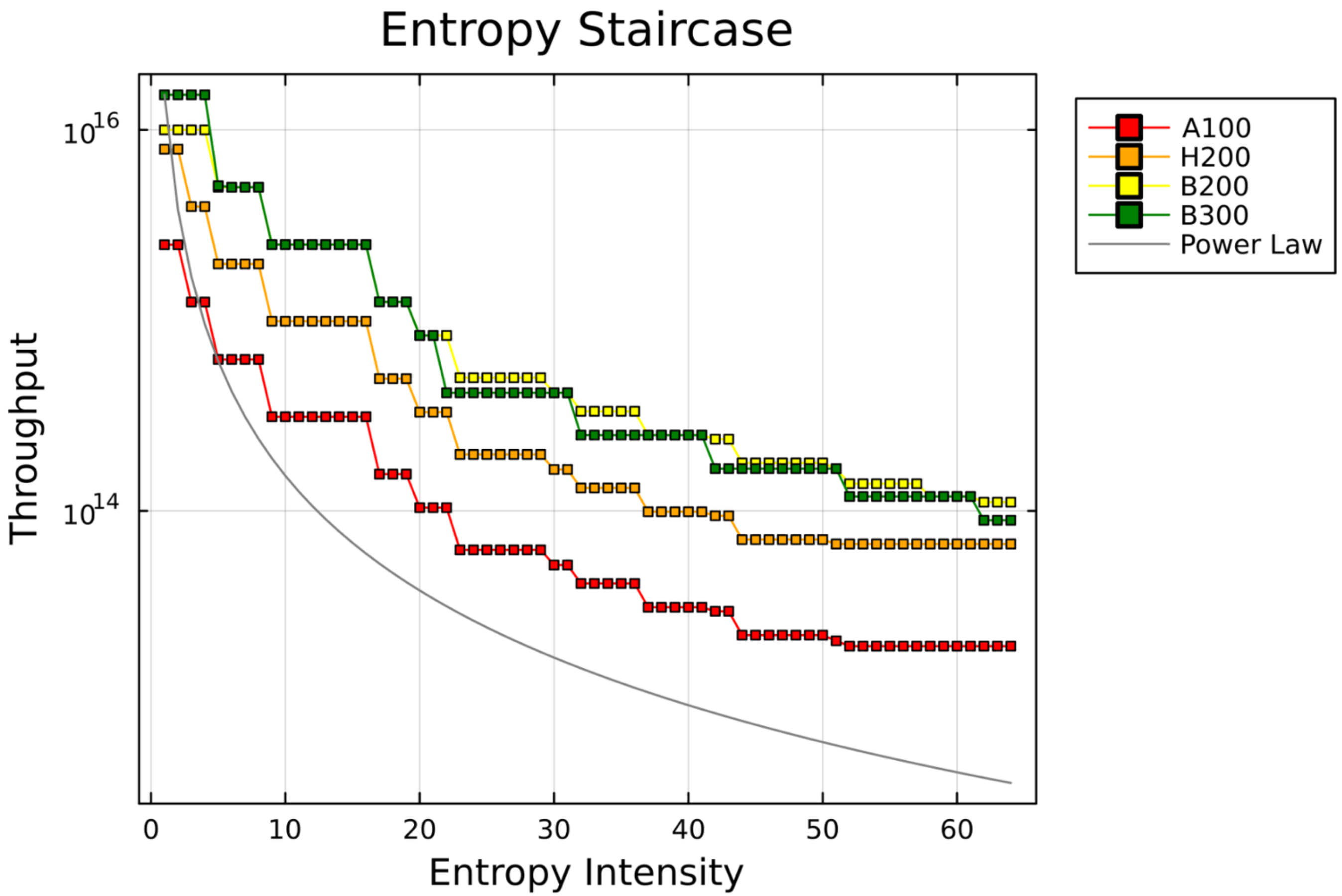
- Ease of physical implementation
- Semantics and error analysis
- Reproducibility

Max Relative Error and Logarithmic Encodings

- Logarithmic encodings are optimal*
  - ...if using max relative error on a bounded interval



["Weber-Fechner Law and the Optimality of the Logarithmic Scale" (2010)]



Entropy Staircase
Throughput
10^{16}
10^{14}
0
10
20
30
40
50
60
Entropy Intensity
A100
H200
B200
B300
Power Law

(Computer) Arithmetic

- What is the 'freedom' or set of potential values of variable x?
  - $x \in R$
- Mathematician: If $x \in R$, infinitely many values!
- Hardware designer: If stored in FP64, ~2^{64} states
- HPC practitioner: [-1000, 1000] but mostly close to zero

- "All models are wrong. Some are useful"