Correctness workshop at NCAR, 11/9/23

Correctness Concerns in HPC and ML

Ganesh Gopalakrishnan and Harvey Dam

THE UNIVERSITY OF UTAH

U.S. DEPARTMENT OF ENERGY | Office of Science

NSF

KAHLERT SCHOOL OF COMPUTING
THE UNIVERSITY OF UTAH

“Correctness Concerns in HPC and ML” is not a presumptuous title
Let us explain what we mean…

- “HPC”
  - ~~“It is all about PDE solving”~~
  - It is one of these pursuits:
    - Climate simulation
    - Finite Elements
  - Correctness
    - ~~“It is correct by definition”~~
    - Converges and obeys all conservation laws
      - A better characterization offered shortly
- “ML”
  - ~~“Cats vs. Dogs”~~
  - It is one of these pursuits
    - ChatGPT
    - Airport screening
  - Correctness
    - Training and test accuracy are high
      - A better characterization offered shortly

What we plan to put forth in this talk

- (GG) That HPC-correctness is rapidly changing at the “bleeding edge”
  - Our discussion
    - HPC’s use of hardware designed for ML is a huge risk
      - That even experts seem to be blissfully unaware of
        - (they seldom discuss)
      - **What have we done about it?**
- (HD) That ML-correctness issues are becoming relevant for HPC also
  - In two ways
    - Use of ML methods in HPC
      - These contribute to “ML correctness issues”
        - And these have “HPC connotations”
      - **What have we done about it?**
    - Use of hardware designed for ML in ML is also a huge risk
      - This is something worth knowing
        - Future work planned based on others' work

What we plan to put forth in this talk

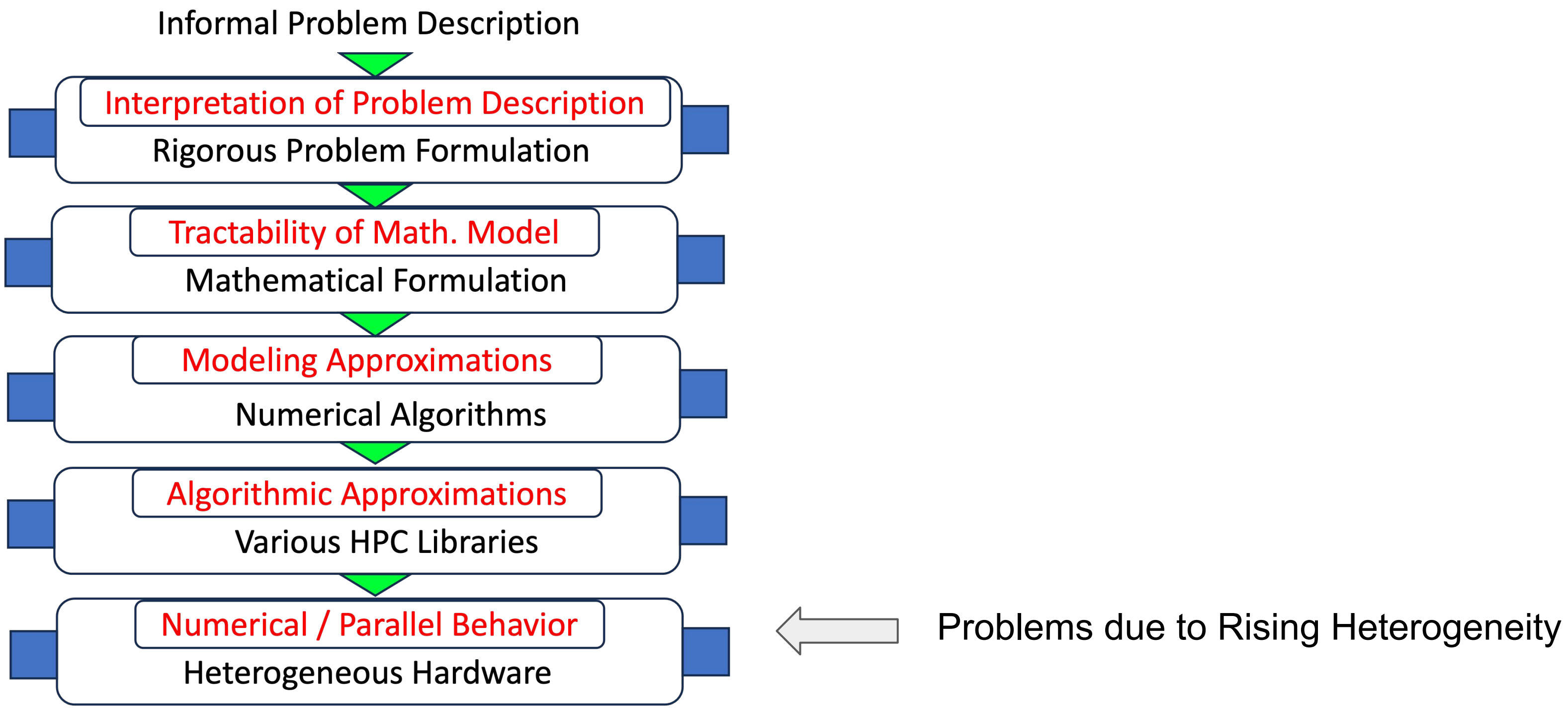
- (GG) That HPC-correctness is rapidly changing at the “bleeding edge”
    - Our discussion
        - HPC’s use of hardware designed for ML is a huge risk
            - That even experts seem to be blissfully unaware of
                - (they seldom discuss)
            - **What have we done about it?** Nixing the Nasty NaNs !!
- (HD) That ML-correctness issues are becoming relevant for HPC also
    - In two ways
        - Use of ML methods in HPC
            - These contribute to “ML correctness issues”
                - And these have “HPC connotations”
            - **What have we done about it?** Fixing the Fairness Faux Pas !!
        - Use of hardware designed for ML in ML is also a huge risk
            - This is something worth knowing
                - Future work planned

Understanding and Mitigating Hardware Failures in Deep Learning Training Accelerator Systems

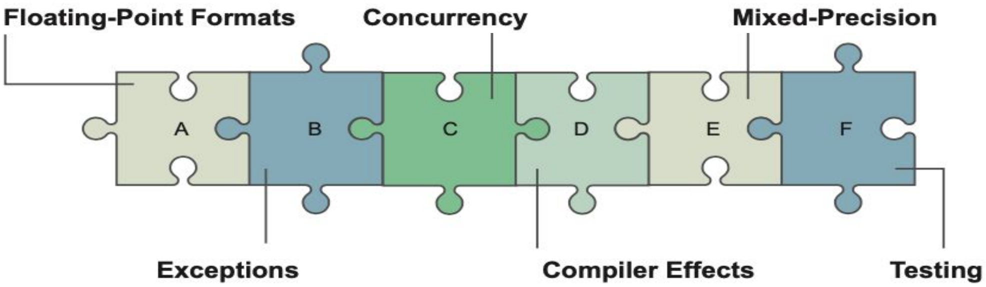
ISCA 2023 (study group Uchicago and Google)
HW faults turn into Nasty NaNs !!

HPC Correctness Stack : A Generic Portrayal

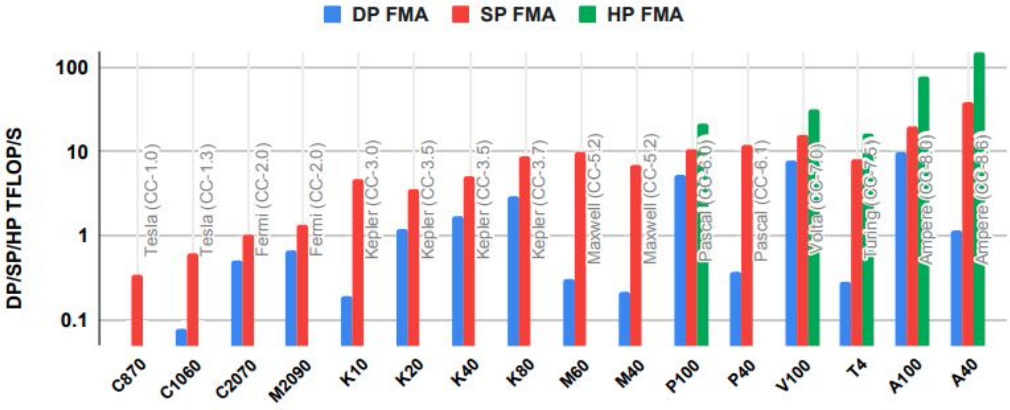
From Correctness in Scientific Computing (CSC 2023, a DOE/NSF Workshop, Orlando, FCRC)



Extreme Heterogeneity and its Correctness Consequences



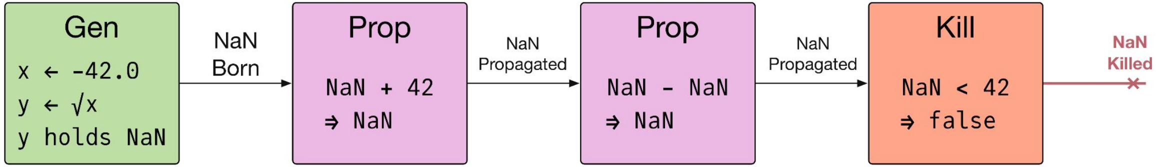
"Guarding Numerics Against Rising Heterogeneity", SC Correctness Workshop 2021 by the PIs



- Heterogeneity (CPUs/GPUs) the norm
  - Rapidly changing in features
  - AMD GPUs also on the rise
    - Unknown repro when porting
- Mostly undocumented building-blocks
  - Libraries are binary-only (in undocumented assembly-level ISA)
  - Compilers differ, especially across optimization levels
- Various Precision Choices
  - FP16, FP8
- Built-in Acceleration for matrix operations
  - Tensor cores (NVIDIA)
    - Not IEEE compatible
  - Matrix cores (AMD)
- No hardware trapping of exceptions in NVIDIA
  - AMD allegedly can trap
    - We have been unable to activate

Reasons to focus on FP Exceptions

- Floating-Point NaN and INF exceptions indicate “arithmetic gone wrong”
    - It is important to understand their origins, how they flow, and how they disappear



    - Given these nasty realities, innovative binary instrumentation methods are essential:
        - GPU ISAs are not documented (well, or at all)
            - Need to reverse-engineer GPU semantics experimentally
        - Many important GPU libraries are closed-source
            - Binary instrumentation essential to observe exception flows
                - (in many cases)
        - Once traced, understanding root-cause and fixing is a “black art”
            - Generic techniques desired
                - E.g., “diagonal boosting” is suggested by some library APIs

HPC Correctness is Seriously Threatened by Floating-Point Exceptions

X. Li, I. Laguna, B. Fang, K. Swirydowicz, A. Li and G. Gopalakrishnan, ***"Design and Evaluation of GPU-FPX: A Low-Overhead tool for Floating-Point Exception Detection in NVIDIA GPUs,"*** HPDC '23: Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing, August 2023, Pages 59–71, https://doi.org/10.1145/3588195.3592991

GPU-FPX: Our tool for Detecting Exceptions in NVIDIA GPU Binaries

- HPDC 2023 paper published on **new tool GPU-FPX** released at https://github.com/LLNL/GPU-FPX
- Found 27 previously unknown exceptions detected across 151 programs on their own data sets
  - Some repairs also identified based on tool feedback

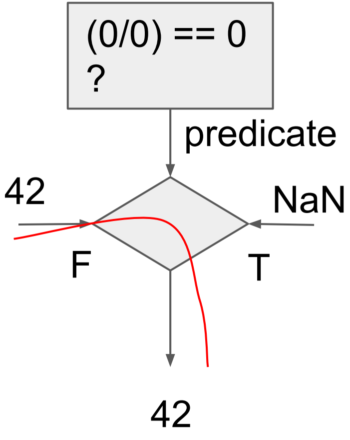


Table 7. Overview of Exception Diagnoses and Repairs using *Analyzer* for Programs with Severe Exceptions

Program	Source available?	Diagnose?	Exceptions Matter?	Fixed?	How Fixed?
GRAMSCHM	yes	yes	yes	yes	Remove 0 from input
LU	yes	yes	yes	yes	Remove 0 from input
myocyte	yes	no	N.A.	N.A.	N.A.
S3D	yes	yes	no	N.A.	N.A.
Interval	yes	yes	no	N.A.	N.A.
Laghos	yes	no	N.A.	N.A.	N.A.
Sw4lite	yes	no	N.A.	N.A.	N.A.
HPCG	no	no	N.A.	N.A.	N.A.
CuMF-Movielens	yes	yes	yes	yes	Enforce variable consistency
cuML-HousePrice	partial	yes	yes	partial	N.A.
CUDA GMRES	partial	yes	yes	partial	Diagonal boosting
SRU-Example	yes	yes	yes	yes	Change input generator

GPU-FPX Components

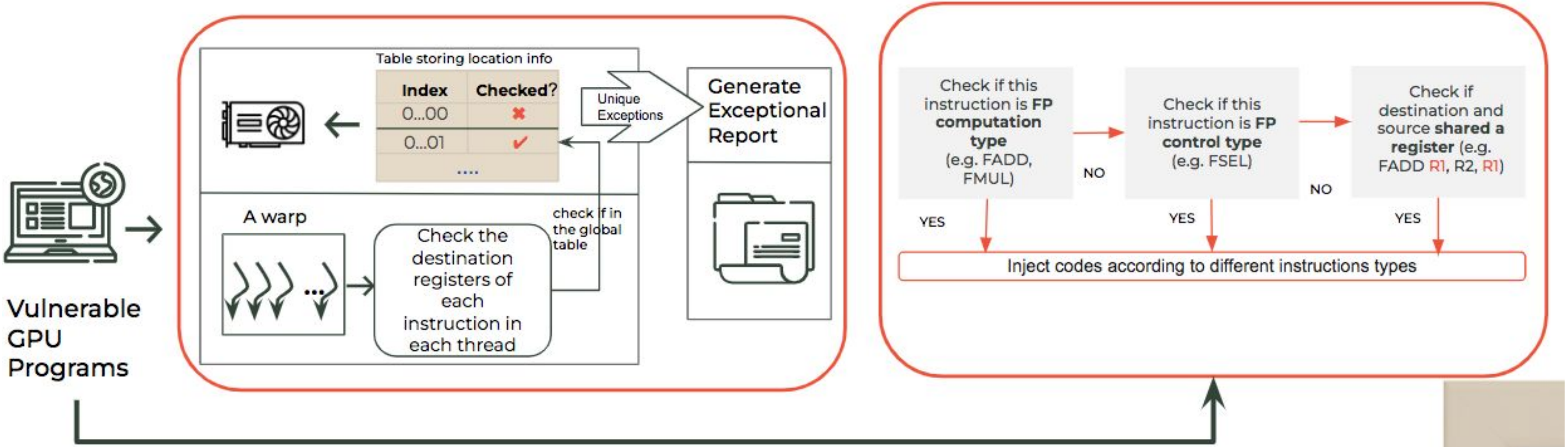
https://github.com/LLNL/GPU-FPX

DETECTOR

Pinpoints exception-generating locations across all kernels

ANALYZER

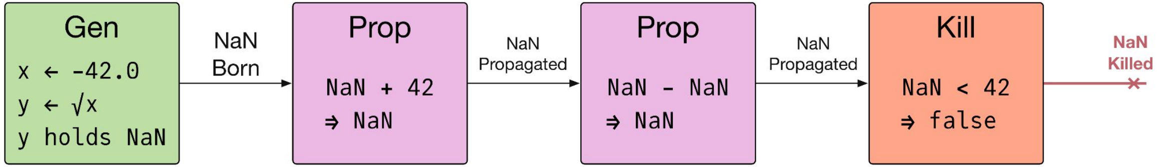
Reports how exceptions flow within one instruction



Life gets easier with source-tracking FloatTracker (Allred, Li, Wiersdorf, Gopalakrishnan)

Combined Julia / GPU tracing is being planned

2:43:50 mark at
https://www.youtube.com/live/rMrHCM1Etng?feature=share

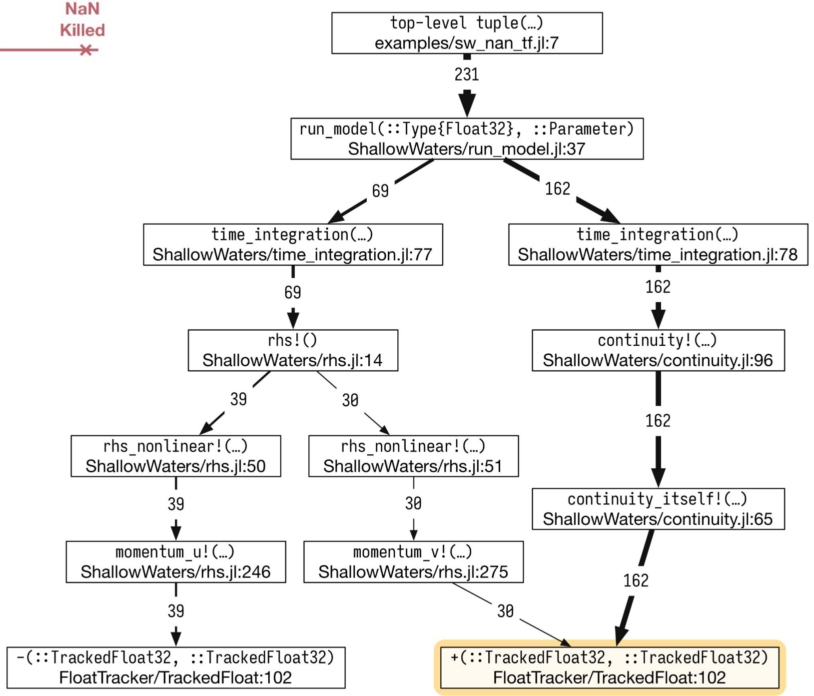



```julia
# Pull in FloatTracker
using FloatTracker

# Wrap inputs in a TrackedFloat* type
num = TrackedFloat64(-42.0)

# Watch as a NaN gets born
should_be_nan = sqrt(num)

# Flush FloatTracker's logs
ft_flush_logs()
```



How ML-correctness has become Relevant to HPC

- HPC will increasingly rely on ML-surrogates such as PINNs
- Errors in regression and classification can impact Science

AI Benchmarking for Science: Efforts from the MLCommons Science Working Group

Jeyan Thiyagalingam[1♠], Gregor von Laszewski[2], Junqi Yin[3], Murali Emani[4], Juri Papay[1], Gregg Barrett[5], Piotr Luszczek[6], Aristeidis Tsaris[3], Christine Kirkpatrick[7], Feiyi Wang[3], Tom Gibbs[8], Venkatram Vishwanath[4], Mallikarjun Shankar[3], Geoffrey Fox[2♣], Tony Hey[1]

Table 7: Summary of the Evaluation.

Benchmark	**Platforms /(Architectures)**	**Science Metric(s)**	**Performance Metric(s)**
`cloud-mask`	Pearl (V100) Summit (V100)	Accuracy	Scalability
`stemdl`	Summit (V100)	Accuracy, F1	-
`candle-uno`	Theta (A100)	-	Throughput
`tevelop`	K80, P100 V100, A100 RTX3080, RTX3090	NNSE	Training Time

Correctness in machine learning, and Science Impacts

Not just loss functions.

- Generalization
- Robustness
- Fairness
    - And all the HPC correctness impacts due to these “ML defects”

Common loss functions

Classification

- Cross-entropy loss
- F score

Regression

- Mean squared error
- Mean absolute error

Reward functions

- Maybe you observe it; maybe you learn it (RLHF)

Robustness

Margins in classifiers
- (Softmax) logit margin
- Parameter margin
- Input margin

Regression
- Coefficient of determination (R^2)
- Gradient magnitude
- Downstream classification?

Adversarial and out-of-distribution robustness
- Input attacks, parameter attacks, hardware attacks
- Mismatch between training and deployed data

What we can do
- Augmentation, adversarial training, human-in-the-loop

Fairness

A few ways of measuring fairness

- Demographic parity: decision is independent of certain features
- Predictive parity: equal precision among subgroups.
- Equal opportunity: equal true positives among subgroups.

Some bias mitigation techniques

- Reweighing (2012)
- Learning fair representations (2013)
- Adversarial debiasing (2018)

You’ll probably sacrifice performance on traditional accuracy measures.

Model complexity and other considerations

No free lunch.

Balancing complexity

- Pruning and sparsification (compression)
  - What Do Compressed Deep Neural Networks Forget?
    - Sara Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, Andrea Frome. https://arxiv.org/abs/1911.05248
  - Understanding the Effect of the Long Tail on Neural Network Compression
    - Harvey Dam, Vinu Joseph, Aditya Bhaskara, Ganesh Gopalakrishnan, Saurav Muralidharan, Michael Garland. https://arxiv.org/abs/2306.06238

Interpretability

- Facilitates validation and error analysis
- Alignment with domain expertise or fairness goals

Bias in data

- A generative model may encounter its own output

Understanding the Effect of the Long Tail on Neural Network Compression

Influence of a training example: the expected accuracy gain from training on a dataset that includes that example vs training on a dataset without it. There is a way to estimate it in reasonable time via sharding,

(from Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation, 2020.)

We used this to estimate the influence of CIFAR10 training examples, then compressed several image classifiers using Group Sparsity

Yawei Li, Shuhang Gu, Christoph Mayer, Luc Van Gool, and Radu Timofte. Group sparsity: The hinge between filter pruning and decomposition for network compression. 2020.

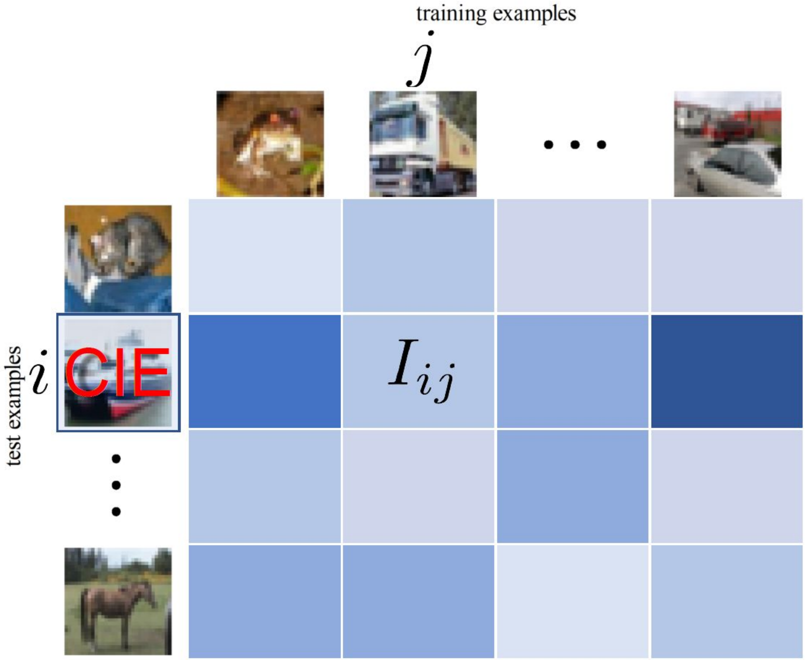
And our own parameterized loss function $L = \alpha L_{\mathrm{CE}} + \beta L_{\mathrm{MSE}} + \gamma L_{\mathrm{CEPred}}$

original loss

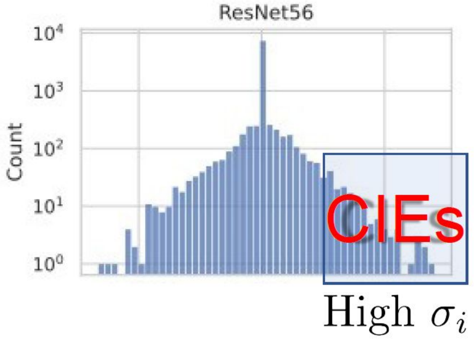
difference between logits

difference between predictions

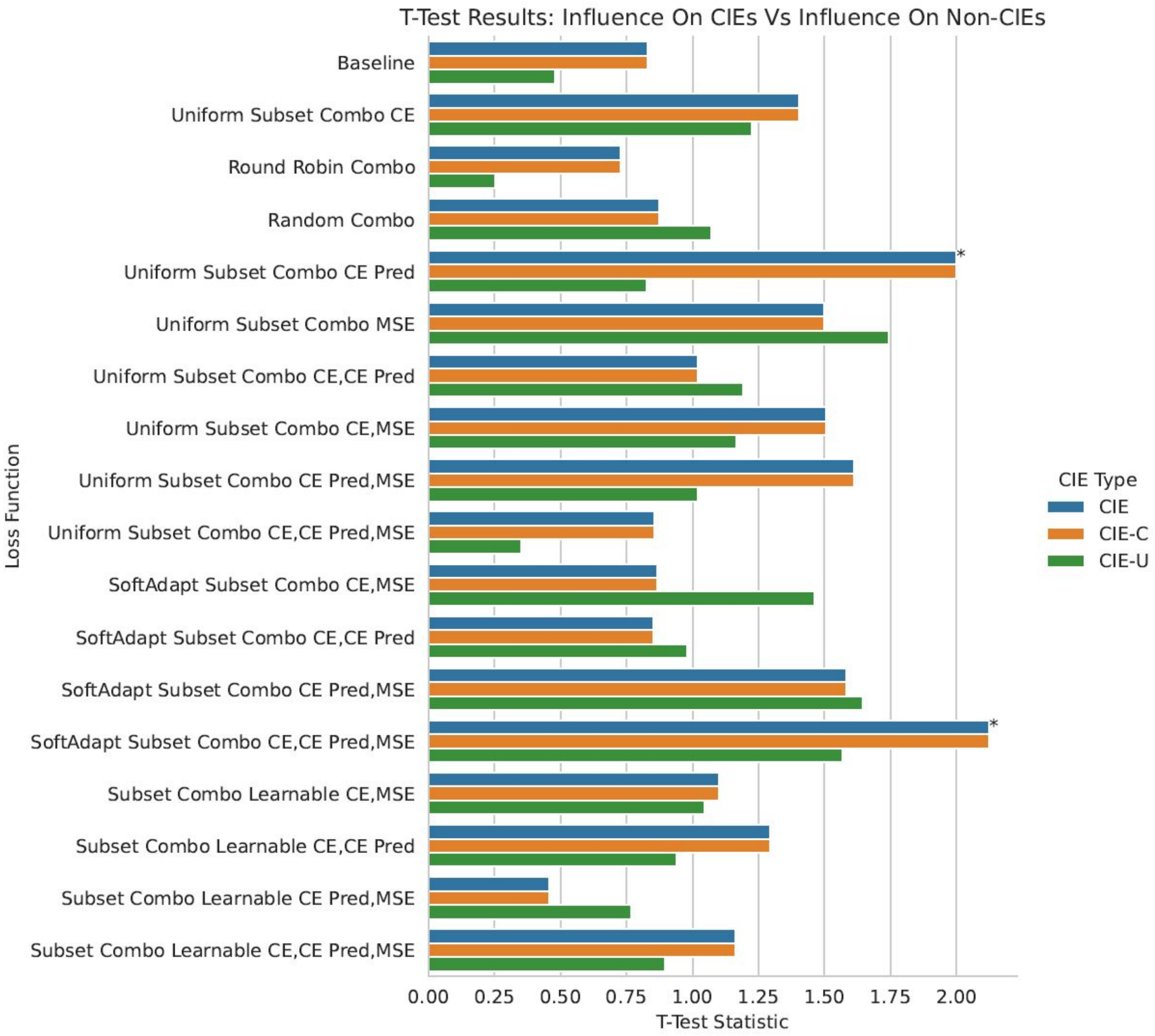
Are instances of disagreements also the most influenced?



$$\sigma_i = \sum_j I_{ij} \oslash |S|$$



Sometimes.


T-Test Results: Influence On CIEs Vs Influence On Non-CIEs
Baseline
Uniform Subset Combo CE
Round Robin Combo
Random Combo
Uniform Subset Combo CE Pred
Uniform Subset Combo MSE
Uniform Subset Combo CE,CE Pred
Uniform Subset Combo CE,MSE
Uniform Subset Combo CE Pred,MSE
Uniform Subset Combo CE,CE Pred,MSE
SoftAdapt Subset Combo CE,MSE
SoftAdapt Subset Combo CE,CE Pred
SoftAdapt Subset Combo CE Pred,MSE
SoftAdapt Subset Combo CE,CE Pred,MSE
Subset Combo Learnable CE,MSE
Subset Combo Learnable CE,CE Pred
Subset Combo Learnable CE Pred,MSE
Subset Combo Learnable CE,CE Pred,MSE
Loss Function
*
*
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
T-Test Statistic
CIE Type
CIE
CIE-C
CIE-U

Summary and key takeaways

- Correctness in ML includes accuracy, robustness, and fairness.
- No free lunch → choose a model.
- Different kinds of fairness, different ways to pursue it.
- Correctness changes when you perturb ML models.
- Explain your work and don't take forever.

Work in progress: solving the challenges of closed designs

- Better fault-location support based on binary instrumentation
  - Tracing executions, comparing traces
- Explaining faults, moving toward repair
  - Need to gather information from the execution context
    - This may again be partly cloistered in closed-source libraries
- Key Takeaways w.r.t. groups like us
  - Until the use of robust design practices are firmly in place, it is not in anyone's interests (esp. academic groups) to go after failures arising from
    - Poor practices
    - Legacy code issues