Implementing High Performance Zero-Knowledge ML Provers

Lilia Tang, Alluri Siddhartha Varma, Siheng Pan, Daniel Kang

UIUC

Implementing High Performance Zero-Knowledge ML Provers

Lilia Tang, Alluri Siddhartha Varma, Siheng Pan, Daniel Kang

UIUC

Increasing calls for ML transparency

CDRH Issues Guiding Principles for Transparency of Machine Learning-Enabled Medical Devices

FOR
June

A call for AI data transparency

April 11, 2024

By Wa

It's time to reveal all recommendation algorithms – by law if necessary

We should know why we see what we see, not be left in the dark

A necessary step: verify that ML model outputs are honestly computed

Increasing calls for ML transparency

CDRH Issues Guiding Principles for Transparency of Machine Learning-Enabled Medical Devices

FOR I

June

A call for AI data transparency

April 11, 2024

By Wa

It's time to reveal all recommendation algorithms – by law if necessary

We should know why we see what we see, not be left in the dark

A necessary step: verify that ML model outputs are honestly computed

Increasing calls for ML transparency

CDRH Issues Guiding Principles for Transparency of Machine Learning-Enabled Medical Devices

FOR
June

A call for AI data transparency

April 11, 2024

By Wa

It's time to reveal all recommendation algorithms – by law if necessary

We should know why we see what we see, not be left in the dark

A necessary step: verify that ML model outputs are honestly computed

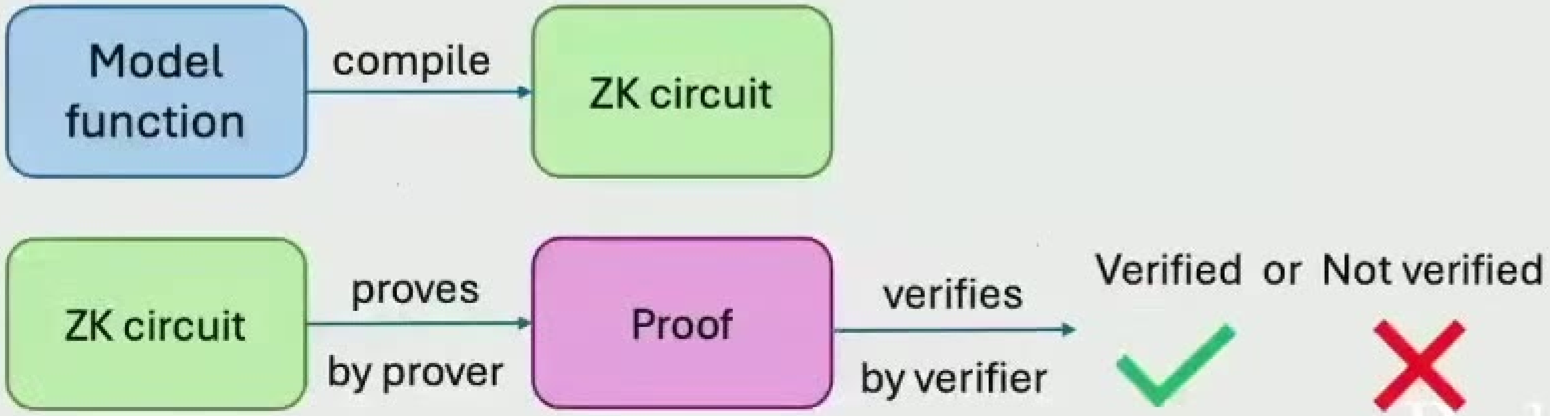
Zero-Knowledge Succinct Non-interactive Arguments of Knowledge (ZK-SNARK)

A ZK-SNARK is a cryptographic tool enabling a prover to prove **a statement true** to a verifier without disclosing additional information.

E.g., prove that a model is executed correctly without revealing weights and inputs

Developers write

ZK framework then proves/verifies



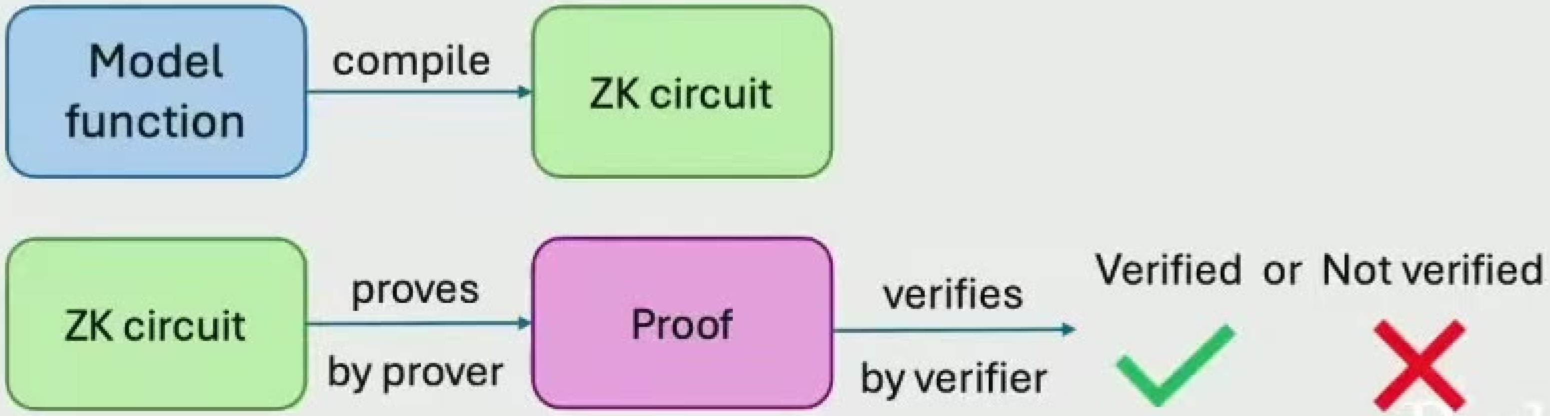
Zero-Knowledge Succinct Non-interactive Arguments of Knowledge (ZK-SNARK)

A ZK-SNARK is a cryptographic tool enabling a prover to prove **a statement true** to a verifier without disclosing additional information.

E.g., prove that a model is executed correctly without revealing weights and inputs

Developers write

ZK framework then proves/verifies



Many ZK-SNARK frameworks

gnark

Plonky2

==>circom
CIRCUIT COMPILER

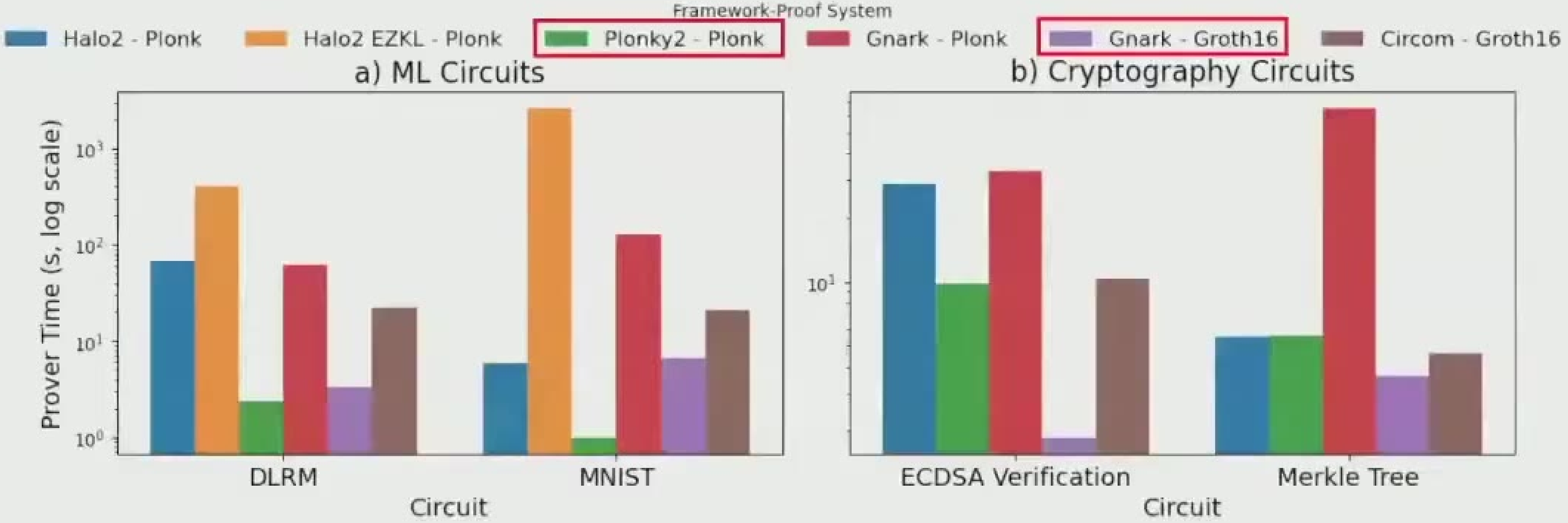
halo2

How do we pick which framework to use? And how do we use the framework effectively?

ZKPerf: a ZK-SNARK proving benchmark

- **ML circuits**
  - **MNIST Convolutional Neural Network**: convolutions
  - **Deep Learning Recommendation Model**: multilayer perceptrons
- Cryptographic circuits
  - Merkle Tree Membership Verification: hashing
  - ECDSA: signatures over elliptic curves
- Frameworks: circom (rapidsnark), Halo2, gnark, Plonky2

Proving time for all tasks and frameworks



- Gnark-Groth16 fastest on crypto circuits but slower on ML
- Plonky2 fastest on ML circuits

Lookups are key to scalable ML circuits

- Lookups can store input-output mappings for non-linearities
  - Use when expensive to compute with arithmetic constraints
- E.g., ReLU can be proven with bit-decomposition or lookups

$\mathrm{ReLU}(b) = \max(0, b)$

$\mathrm{ReLU}(b)$ bit decomposition **each operation:**

1. Prove

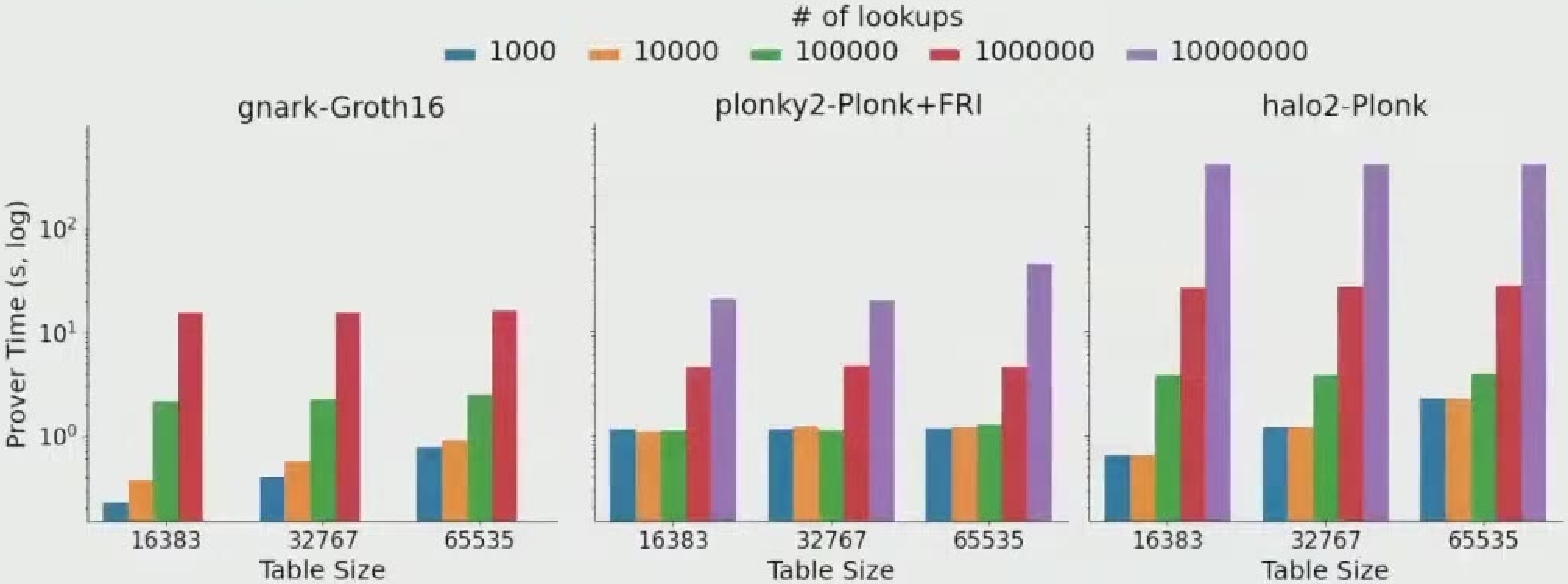
$$b = \sum_i b_i 2^{l-i} \qquad b_i(1 - b_i) = 0$$

2. Output based on selector on sign bit

$\mathrm{ReLU}(b)$ lookup: commit table and add lookup constraints

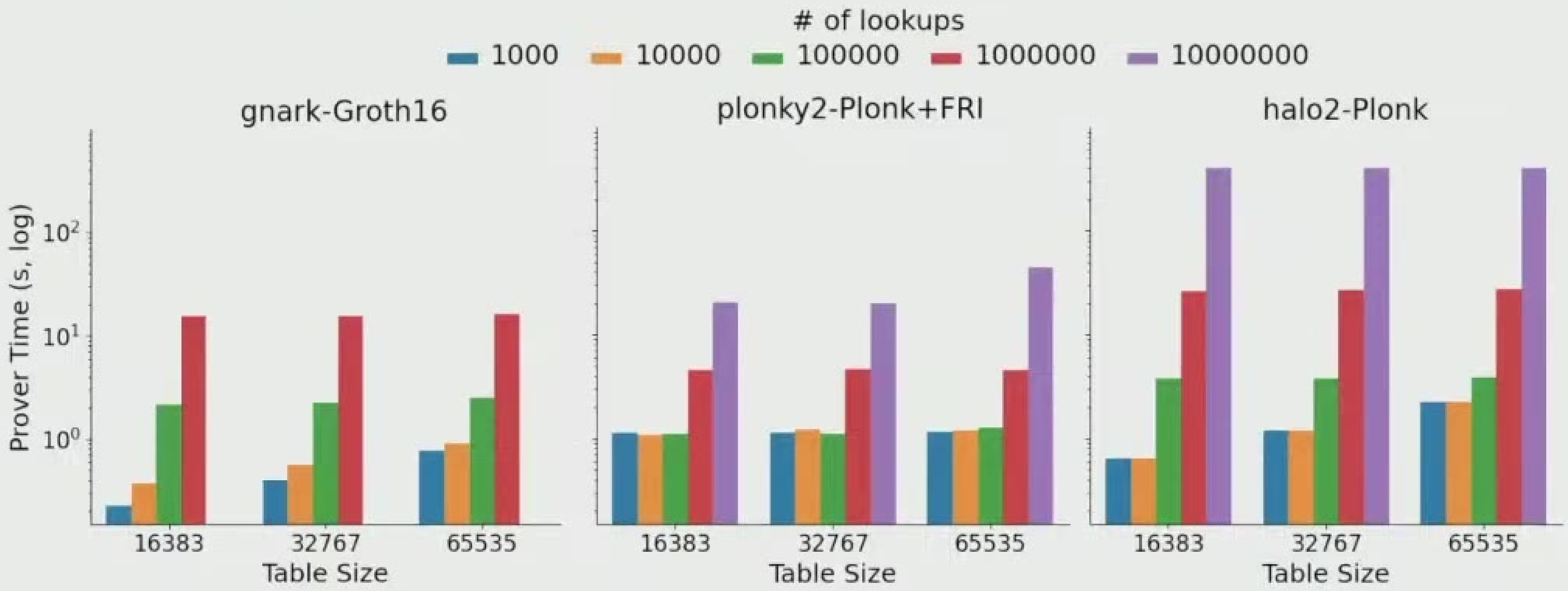
b	ReLU(b)
-1	0
0	0
1	1
2	2

Lookup costs



- Gnark's lookups are slow enough that bit-decomposition wins out
- Plonky2's lookups are most scalable

Lookup costs

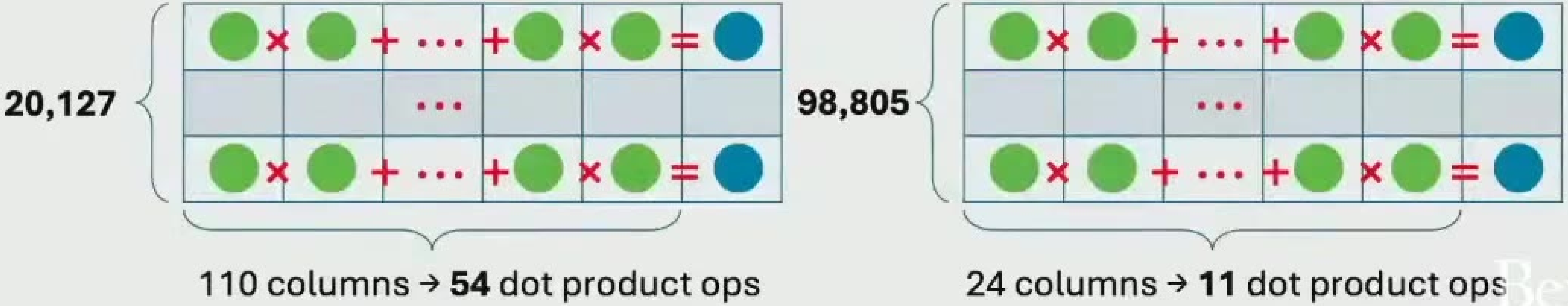


- Gnark's lookups are slow enough that bit-decomposition wins out
- Plonky2's lookups are most scalable

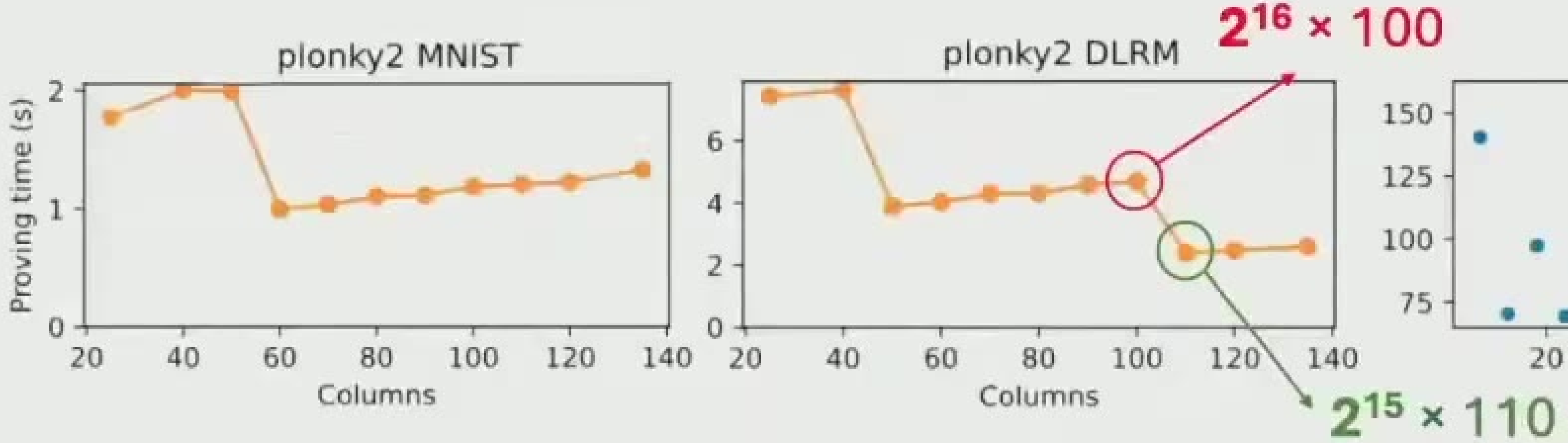
Optimizations with circuit structure

- Halo2 and Plonky2 have rows and columns
- Rows affect sizes of operations: FFTs and MSMs
  - Padded to a power of 2
- Columns affect number of operations

Dot product gate configurations in DLRM



Row and column microbenchmarks



Proving time drops when increasing columns decreases rows

Conclusion

- ZKPerf: a ZK-SNARK proving benchmark with crypto and ML tasks
- Most costly parts of ML circuits are nonlinearities (e.g., ReLU)
  - Need efficient lookups to prove larger models
- Circuit structure matters
  - One way: optimize row and column dimensions

@liliatangxy liliat2@Illinois.edu

Conclusion

- ZKPerf: a ZK-SNARK proving benchmark with crypto and ML tasks
- Most costly parts of ML circuits are nonlinearities (e.g., ReLU)
  - Need efficient lookups to prove larger models
- Circuit structure matters
  - One way: optimize row and column dimensions

@liliatangxy liliat2@Illinois.edu

Conclusion

- ZKPerf: a ZK-SNARK proving benchmark with crypto and ML tasks
- Most costly parts of ML circuits are nonlinearities (e.g., ReLU)
  - Need efficient lookups to prove larger models
- Circuit structure matters
  - One way: optimize row and column dimensions

uiuc-kang-lab/zkperf

@liliatangxy liliat2@illinois.edu

Conclusion

- ZKPerf: a ZK-SNARK proving benchmark with crypto and ML tasks
- Most costly parts of ML circuits are nonlinearities (e.g., ReLU)
  - Need efficient lookups to prove larger models
- Circuit structure matters
  - One way: optimize row and column dimensions

uiuc-kang-lab/zkperf

@liliatangxy

liliat2@illinois.edu