N-TORC: NATIVE TENSOR OPTIMIZER FOR REAL-TIME CONSTRAINTS

UNIVERSITY OF ARKANSAS

David Andrews

Miaoqing Huang

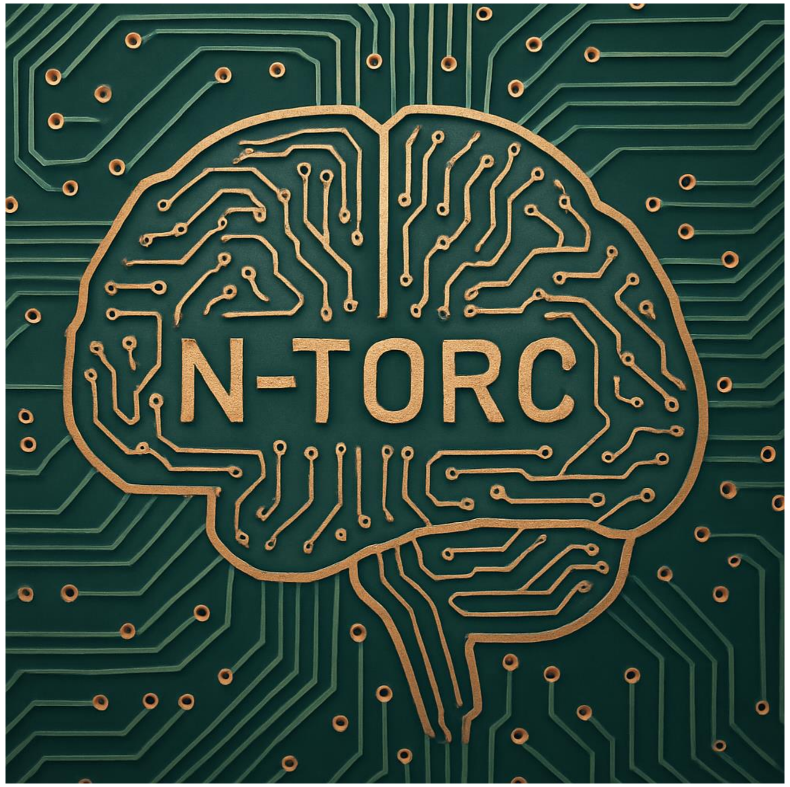
UNIVERSITY OF South Carolina

Suyash Vardhan Singh*

Iftakhar Ahmad*

Austin R.J. Downey

Jason D. Bakos

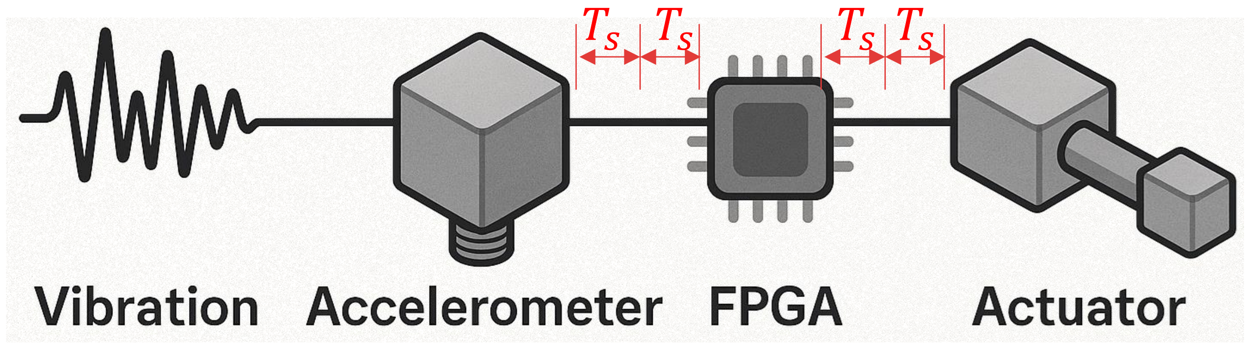


This material is based upon work supported by the National Science Foundation under Grant No. 1956071.

*student authors

OBJECTIVE

- HW/SW stack for High-Rate Machine Learning (HRML) applications:
  - Associated with a cyber physical system
  - Inference at KHz/MHz rates
  - Have real-time latency constraint of $\frac{1}{rate}$ (μs/ns)
  - Embedded platform: minimal resources needed for a particular accuracy

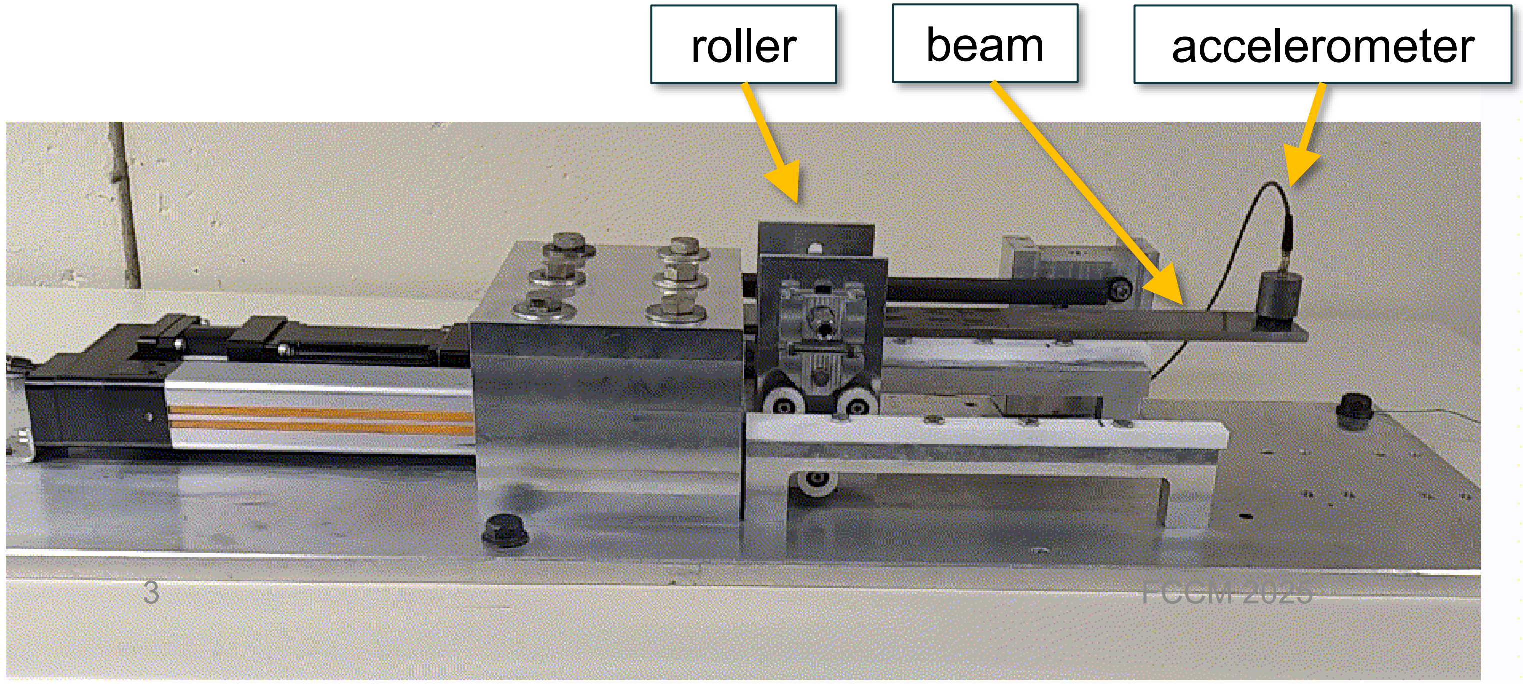


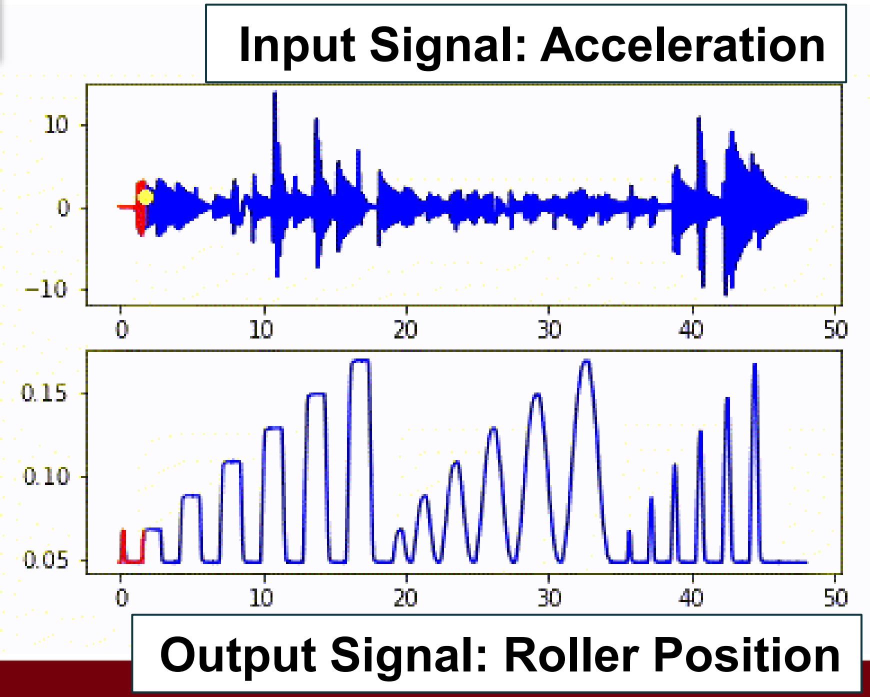
Applications:

Intelligent airbags, blast mitigation, active vibration dampening, etc.

DROPBEAR

- Dynamic Reproduction of Projectiles in Ballistic Environments for Advanced Research
- Developed by AFRL at Eglin AFB





DROPBEAR DATASET

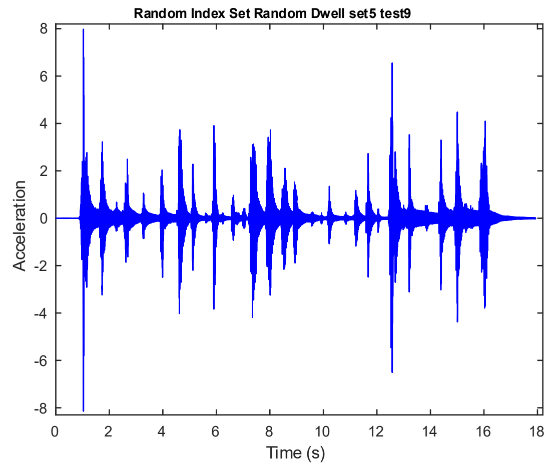
- Sample rate: 5 KHz
  - $T_s = 200\ \mu s$
- 150 experimental runs
- 3 categories of roller behavior

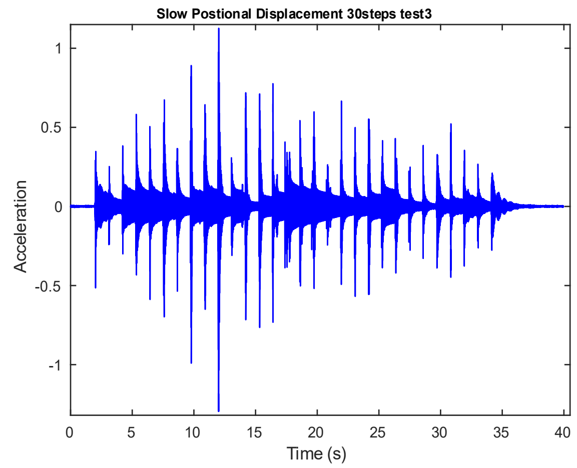
Repo:

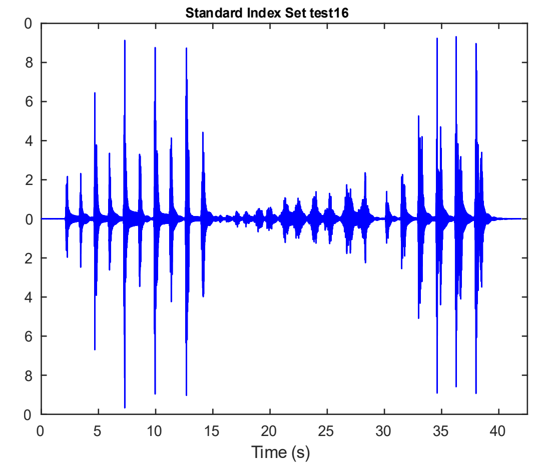
random index sets

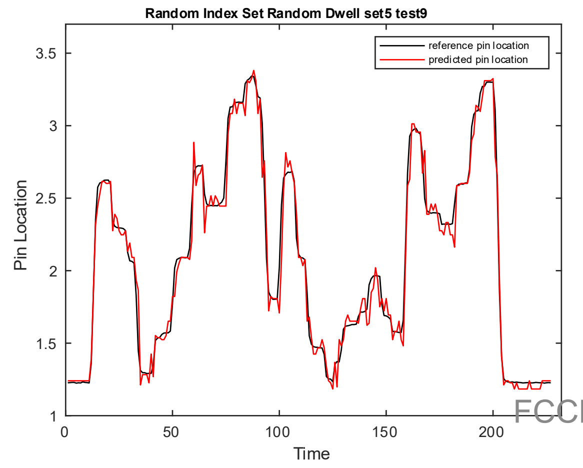
slow positional index sets

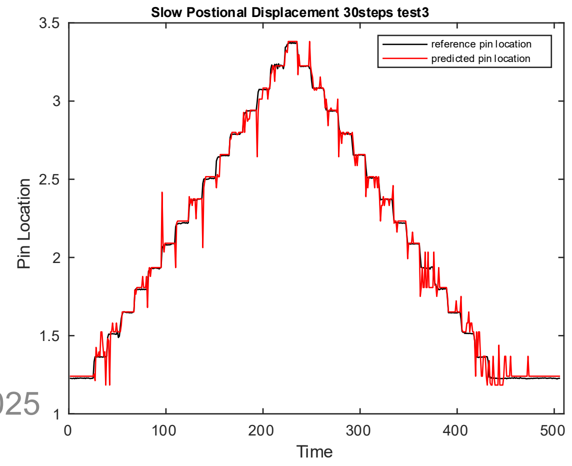
standard index sets

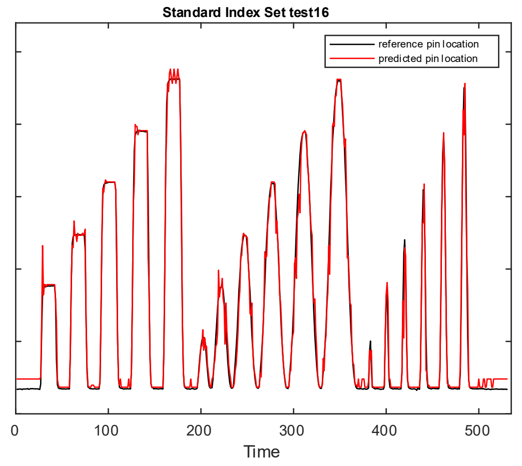




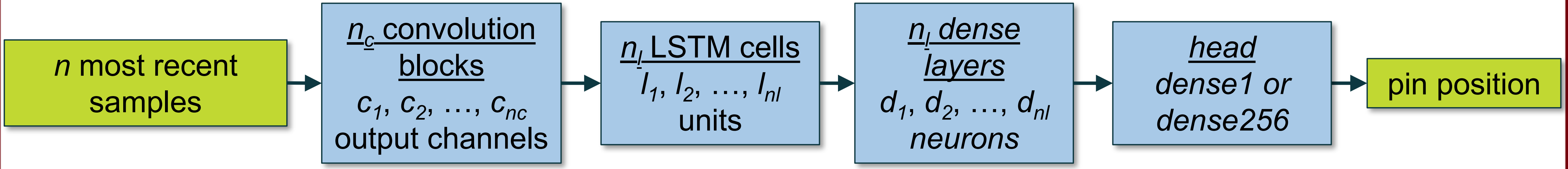






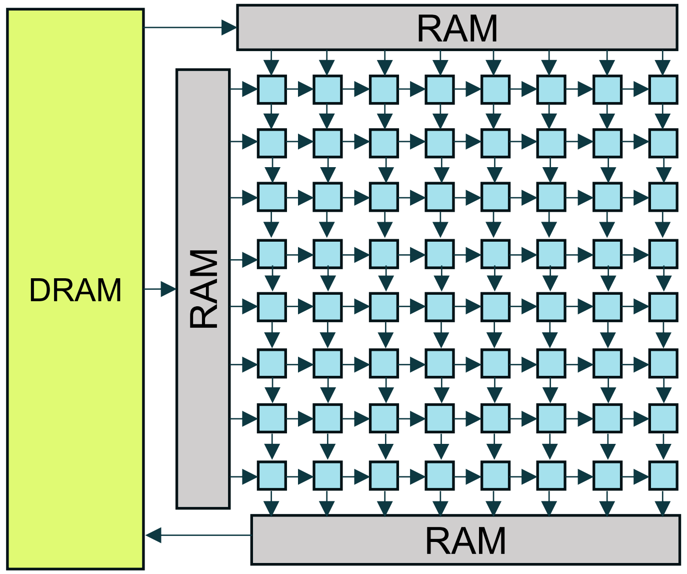


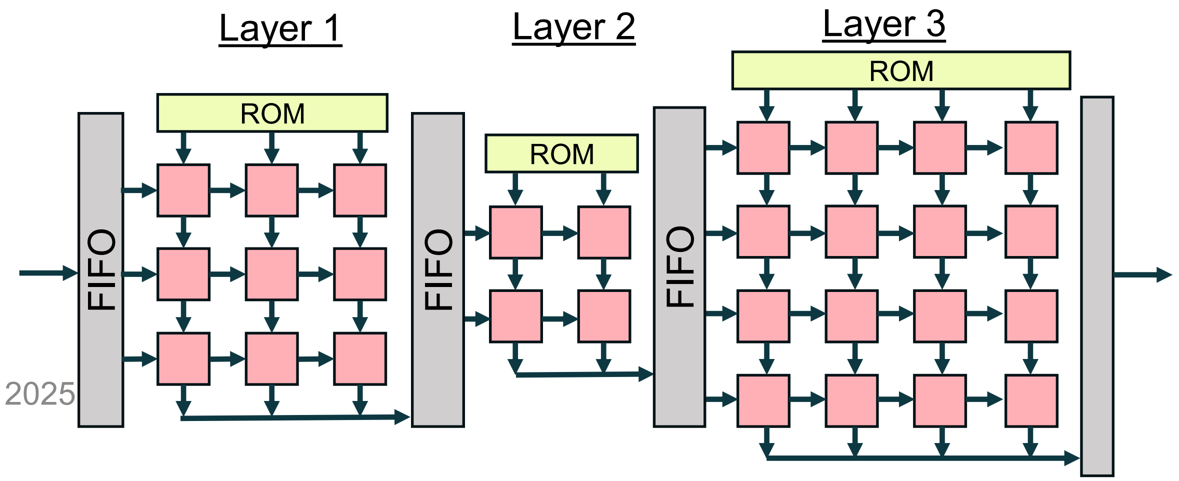
MODEL DEPLOYMENT



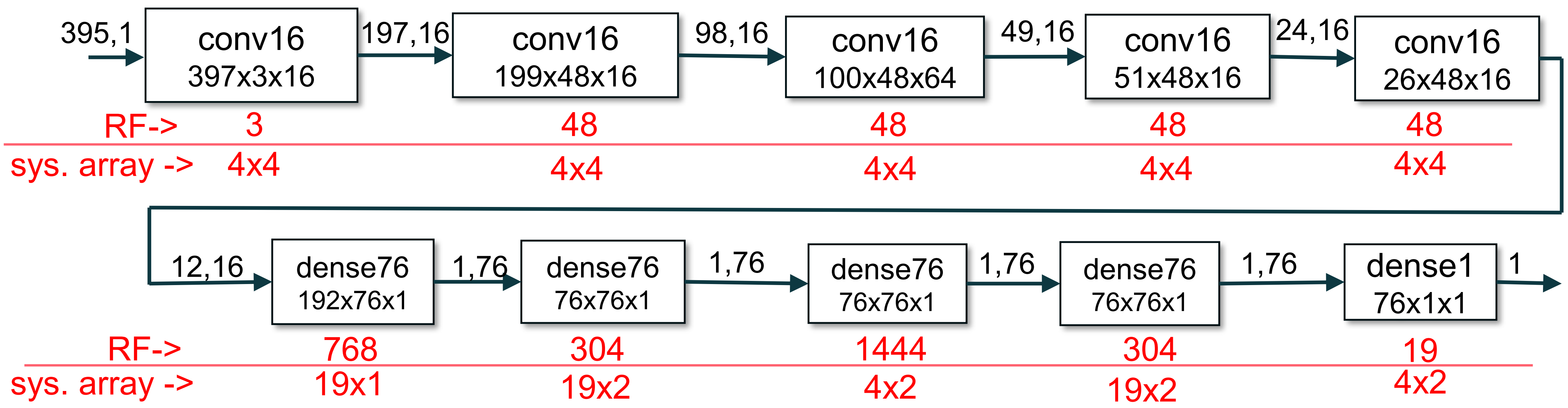
- "Traditional" overlay approach (TPU, VTA, Gemmini):
  - One systolic array shared by all layers
  - Weights, inputs, and outputs exchanged with off-chip memory

- Dataflow approach (hls4ml/FINN):
  - Allocate dedicated systolic array for each layer
  - # multipliers = $block\ factor = \frac{MVM\ size}{reuse\ factor}$
  - All weight tensors stored in on-chip ROMs
  - Outputs transferred via FIFOs





EXAMPLE MODEL

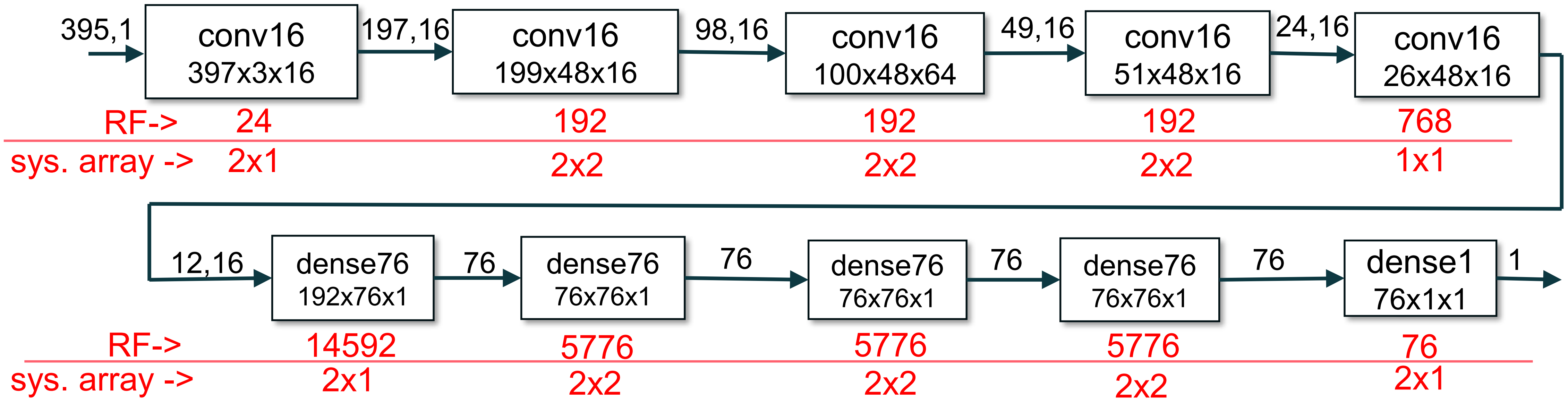


- hls4ml:
  - 1.4×10^{10} valid reuse factor permutations

With RFs shown:

- 177 total multipliers
- Latency = 12250 cycles (49 µs @ 250 MHz)
- 230K LUT (94%), 298 BRAM (47%) (on ZCU104)

EXAMPLE MODEL



- hls4ml:
  - 31 total multipliers
  - Latency = 54K cycles (216 μs @ 250 MHz)
  - 174K (-56K) LUTs, 279 (-19) BRAMs

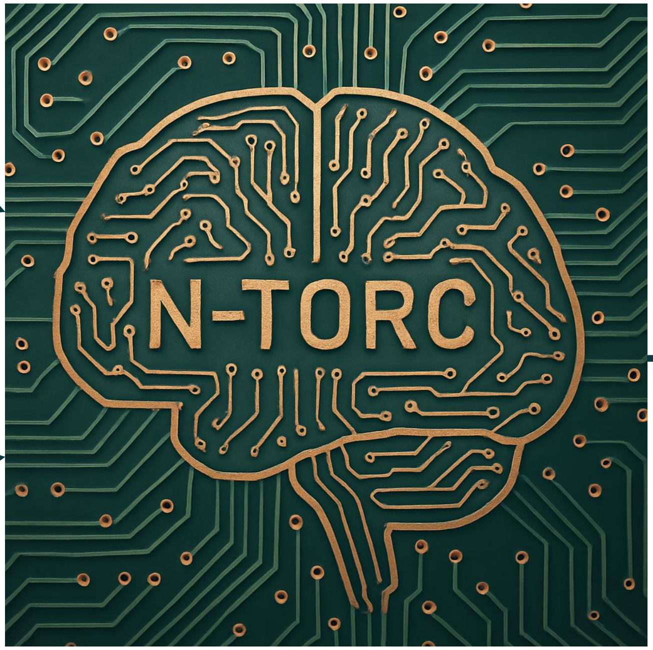
- MAESTRO systolic array overlay:
  - 16x16 systolic array
  - 256 KB weight buffer/128 KB output buffer
  - 4 word/cycle off-chip memory bandwidth

  - Latency = 221K cycles (884 μs @ 250 MHz)

N-TORC: AUTOMATIC DESIGN DEPLOYMENT

Training data
i.e. DROPBEAR

Target
sample rate

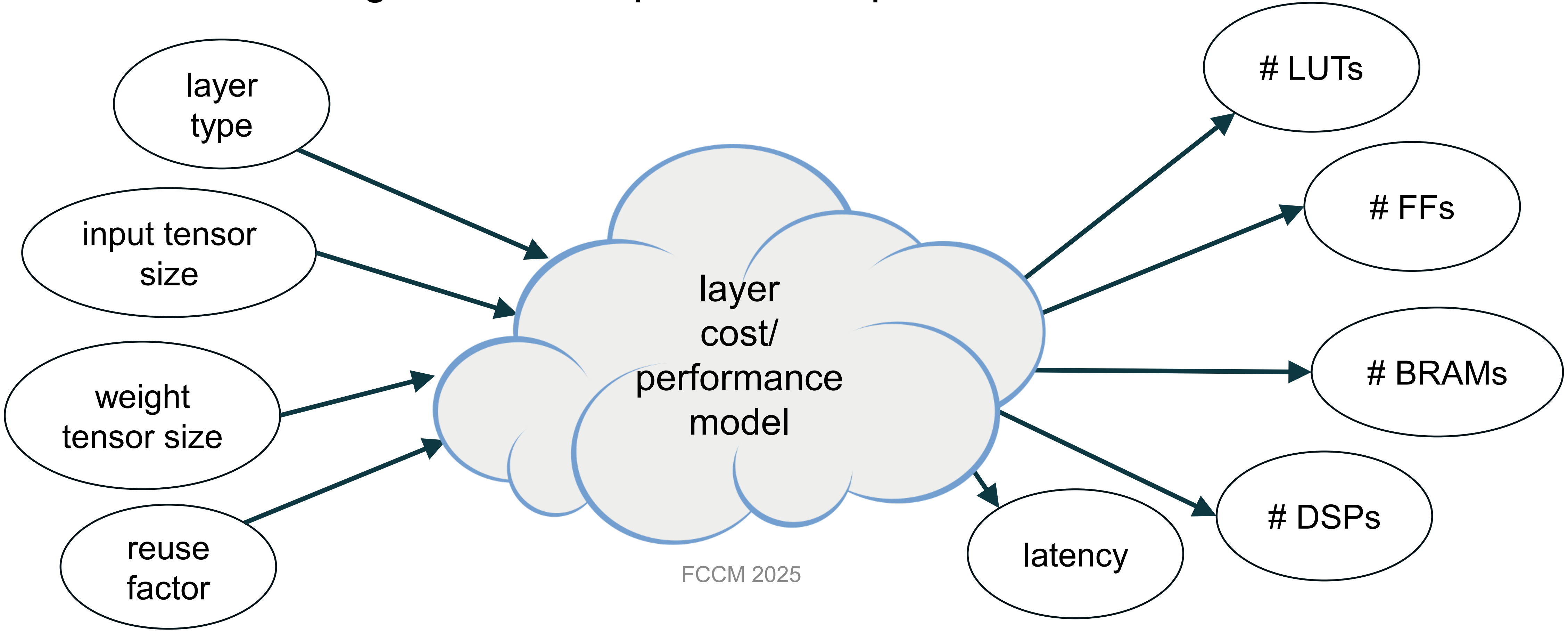


Set of HW deployed models:
1. Meets latency constraint
2. Minimal cost for its accuracy (Pareto optimal)

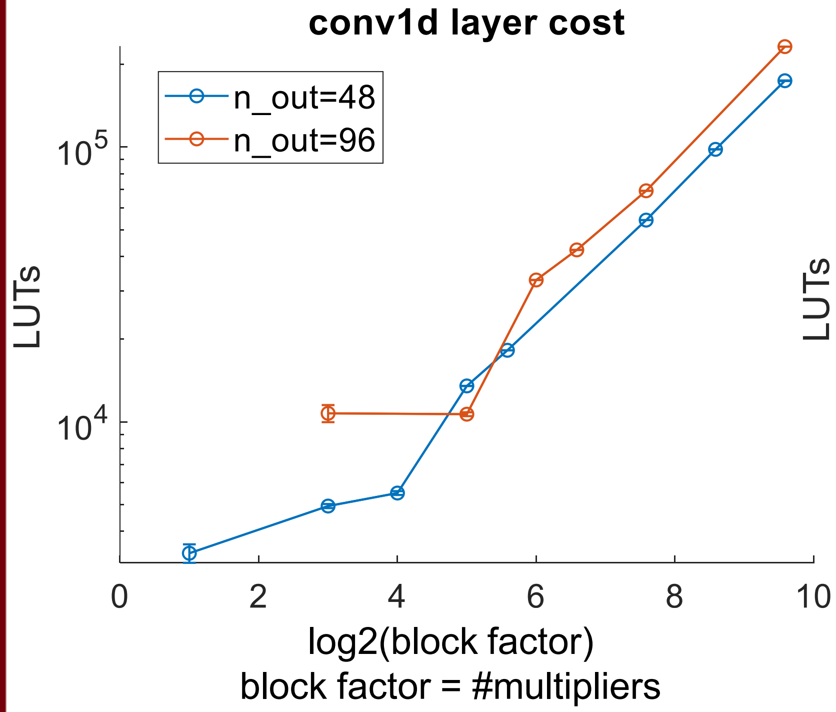
NEED:
1. Cost/performance models for individual hls4ml layers
2. Method to optimize the reuse factor of each layer to meet constraint and minimize cost
3. Method to generate a set of optimal DROPBEAR models w.r.t. accuracy and cost

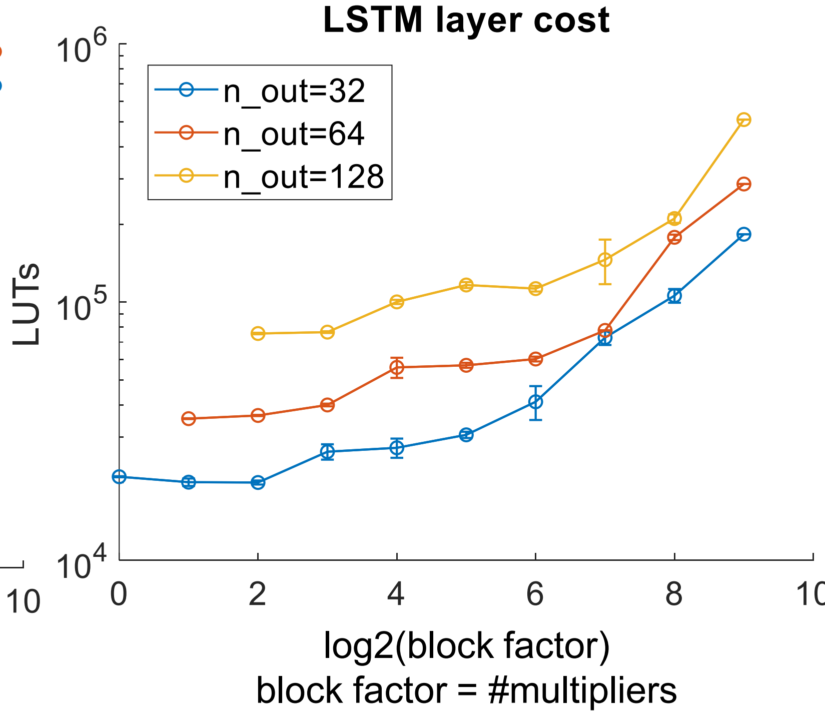
COST/PERFORMANCE MODEL

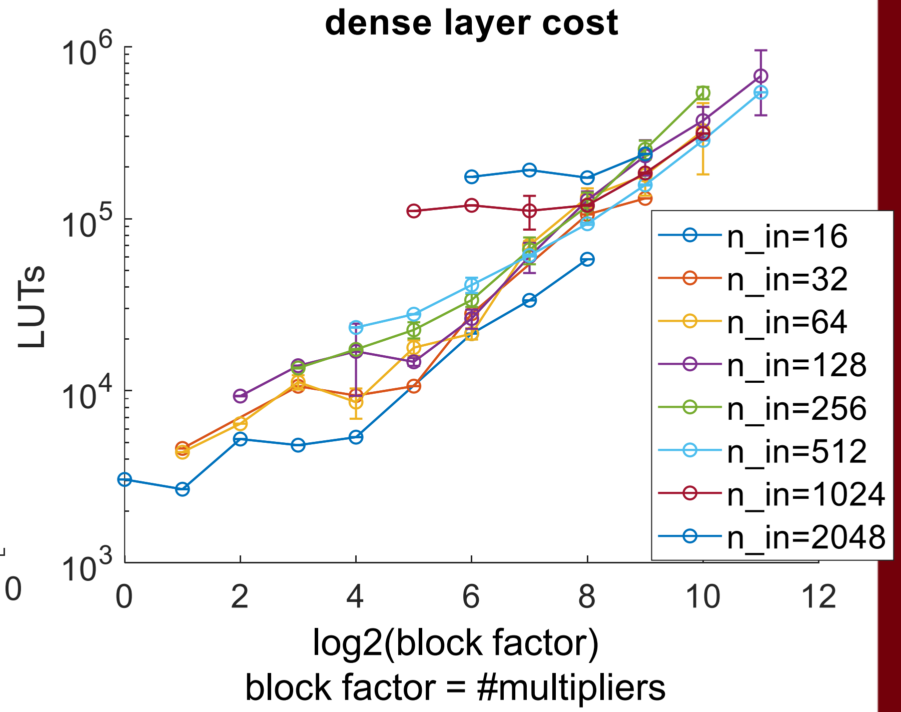
- Cost/performance prediction for HLS is an open problem
- N-TORC advantage: restricted parameter space



HLS4ML PERFORMANCE MODEL





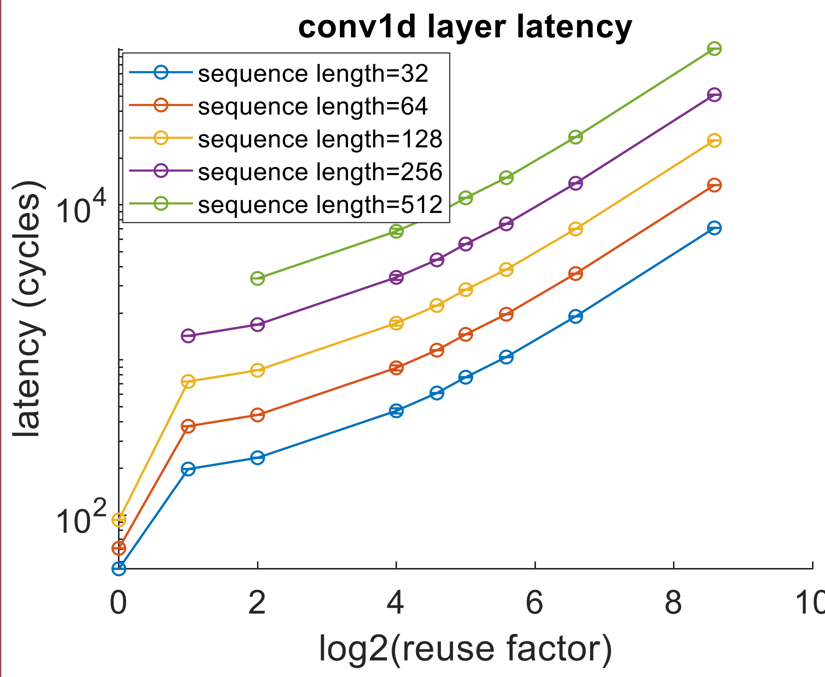


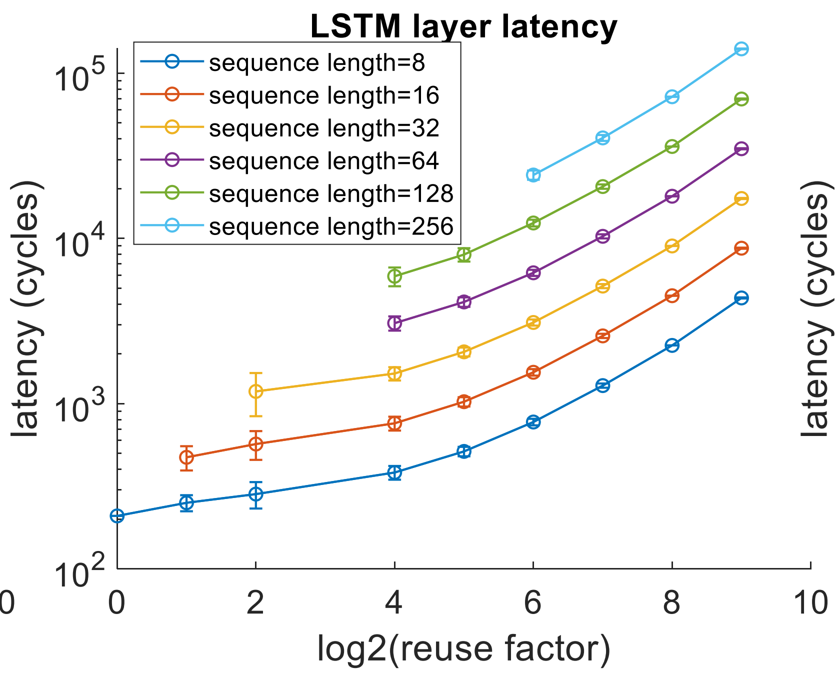
n_out = # output channels

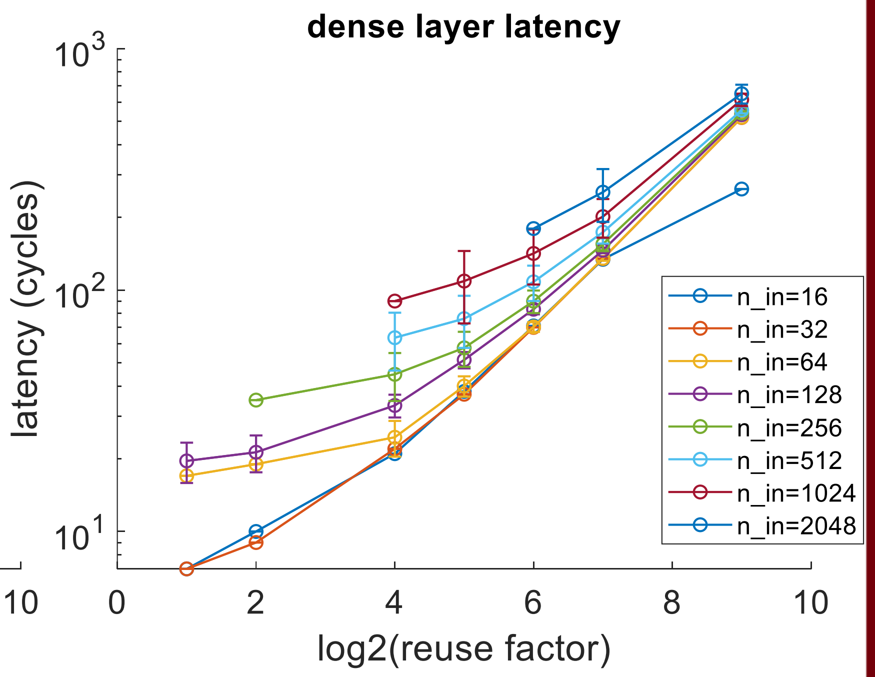
n_out = # units x 4

n_in = # inputs

HLS4ML PERFORMANCE MODEL

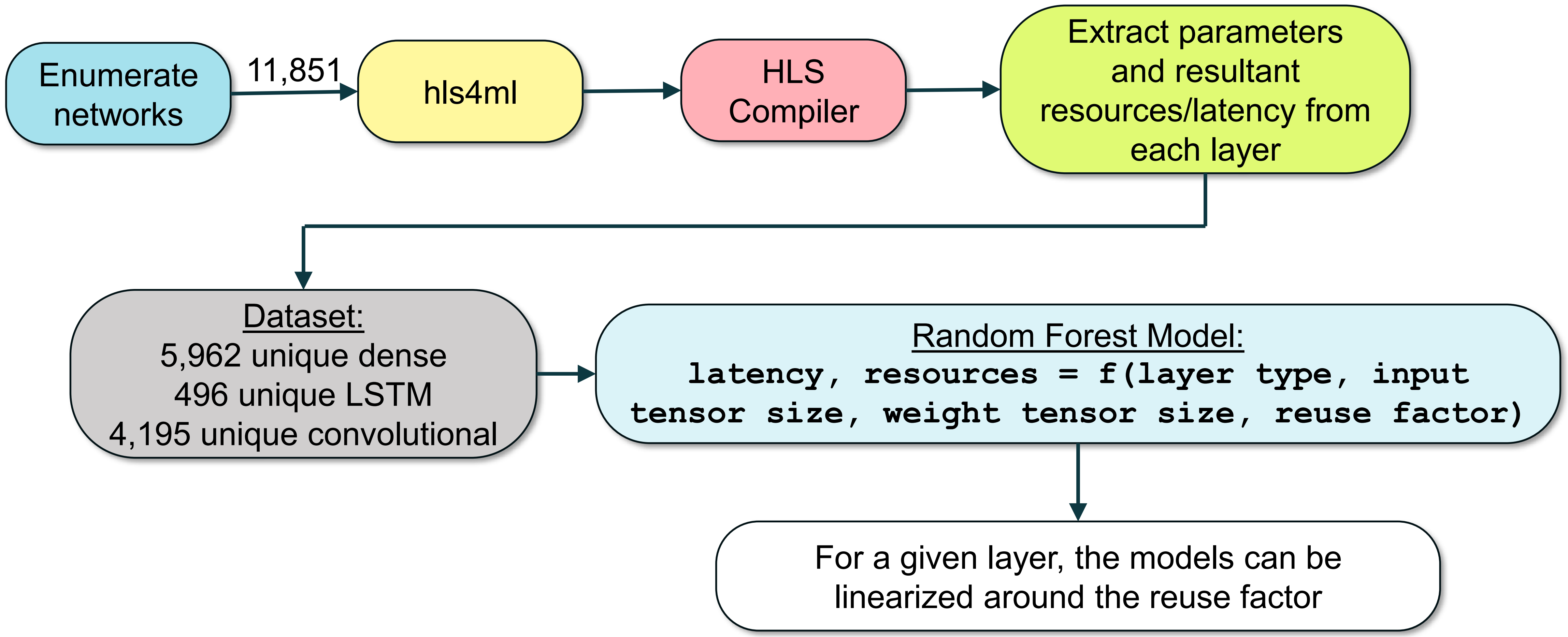






n_in = # inputs

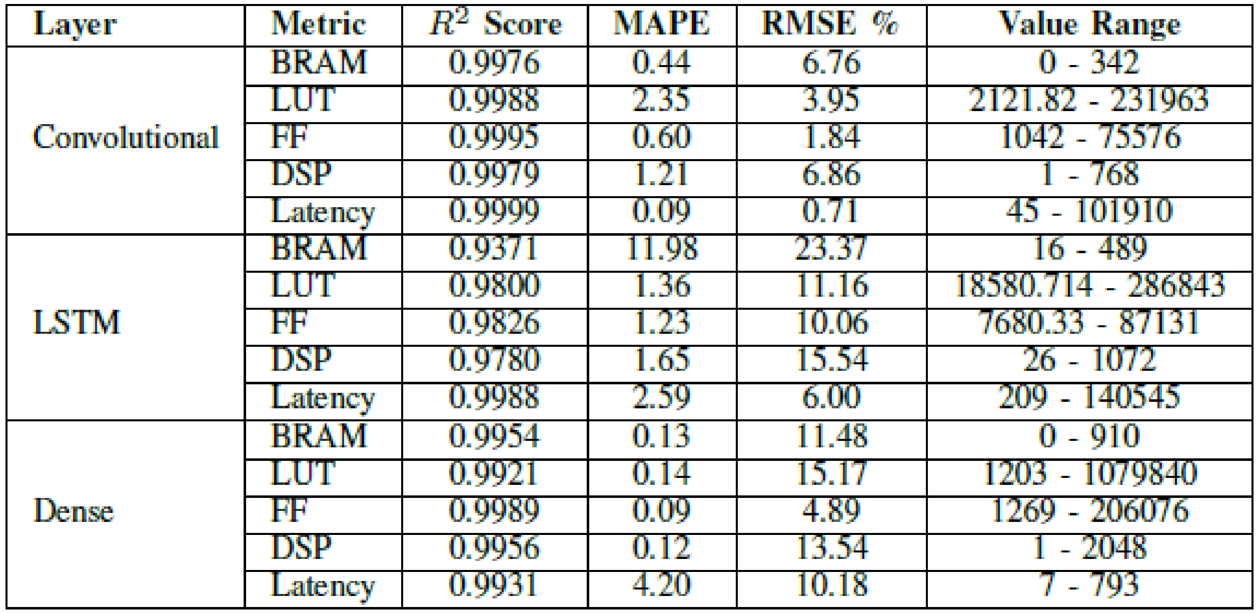
HLS4ML COST/PERFORMANCE MODELING


Enumerate networks
11,851
hls4ml
HLS Compiler
Extract parameters and resultant resources/latency from each layer
Dataset:
5,962 unique dense
496 unique LSTM
4,195 unique convolutional
Random Forest Model:
latency, resources = f(layer type, input tensor size, weight tensor size, reuse factor)
For a given layer, the models can be linearized around the reuse factor

* Amenable to ILP

COST/PERFORMANCE MODEL TEST ACCURACY

- Data-driven HLS p/c models in the literature achieve [1]:
  - DSP: 9% to 15% MAPE
  - LUT: 4% to 26% MAPE
  - FF: 6% to 26% MAPE
  - Latency: 4% MAPE



Layer	Metric	R^2 Score	MAPE	RMSE %	Value Range
Convolutional	BRAM	0.9976	0.44	6.76	0 - 342
	LUT	0.9988	2.35	3.95	2121.82 - 231963
	FF	0.9995	0.60	1.84	1042 - 75576
	DSP	0.9979	1.21	6.86	1 - 768
	Latency	0.9999	0.09	0.71	45 - 101910
LSTM	BRAM	0.9371	11.98	23.37	16 - 489
	LUT	0.9800	1.36	11.16	18580.714 - 286843
	FF	0.9826	1.23	10.06	7680.33 - 87131
	DSP	0.9780	1.65	15.54	26 - 1072
	Latency	0.9988	2.59	6.00	209 - 140545
Dense	BRAM	0.9954	0.13	11.48	0 - 910
	LUT	0.9921	0.14	15.17	1203 - 1079840
	FF	0.9989	0.09	4.89	1269 - 206076
	DSP	0.9956	0.12	13.54	1 - 2048
	Latency	0.9931	4.20	10.18	7 - 793

[1] C. Hao et al, "High-level Synthesis Performance Prediction using GNNs: Benchmarking, modeling, and advancing," DAC22.

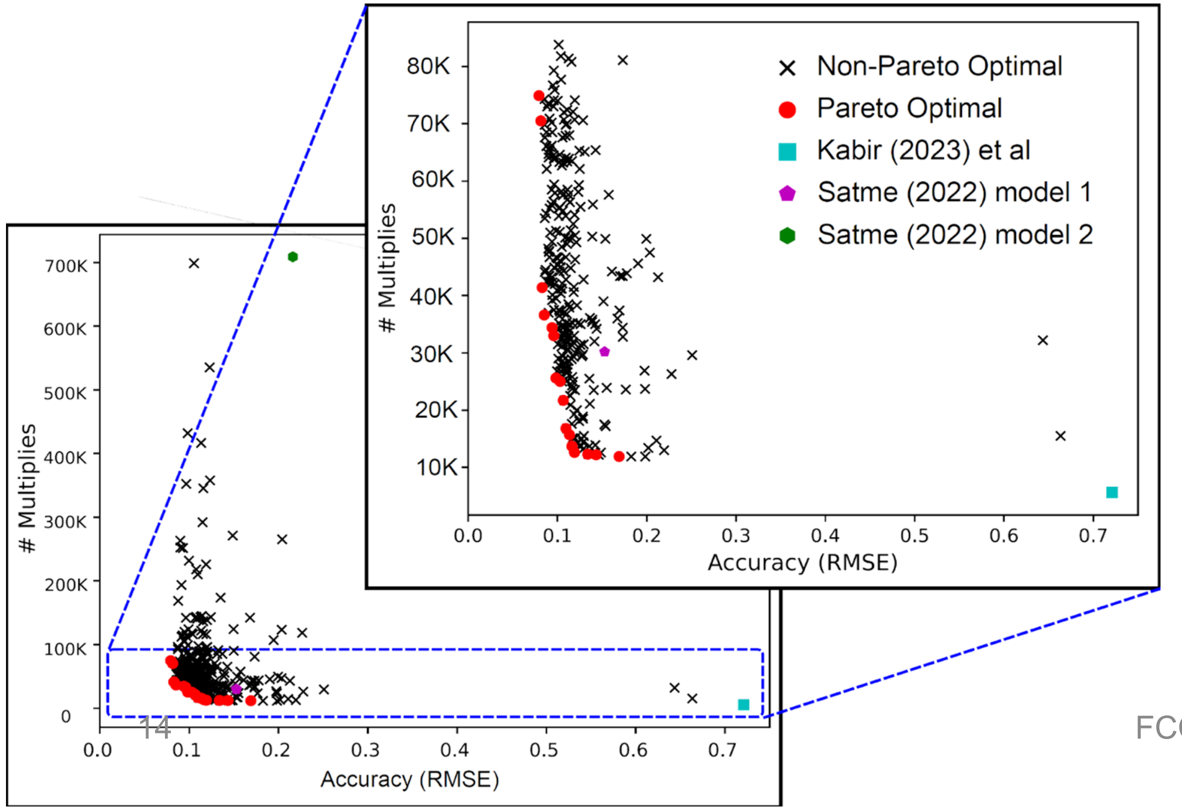
$$\mathrm{R}^2 = 1 - \frac{\sum(y_i - \widehat{y_i})^2}{\sum(y_i - \mu)^2}$$

N-TORC DESIGN FLOW

- **Input:** dataset (e.g. DROPBEAR)
- **Output:** set of latency constrained, Pareto optimal model deployments (accuracy/cost)

Step 1: Train DROPBEAR Models
Multi-objective Bayesian optimization
Pareto optimal models

Step 2:
For each use integer linear solver (Gurobi) to solve reuse factor for each layer to constrain latency to 200 μs and minimize resources



$$\begin{tikzcd} \text{conv} \arrow[r] & \text{conv} \arrow[r] & \text{LSTM} \arrow[r] & \text{dense} \arrow[r] & \text{dense} \end{tikzcd}$$

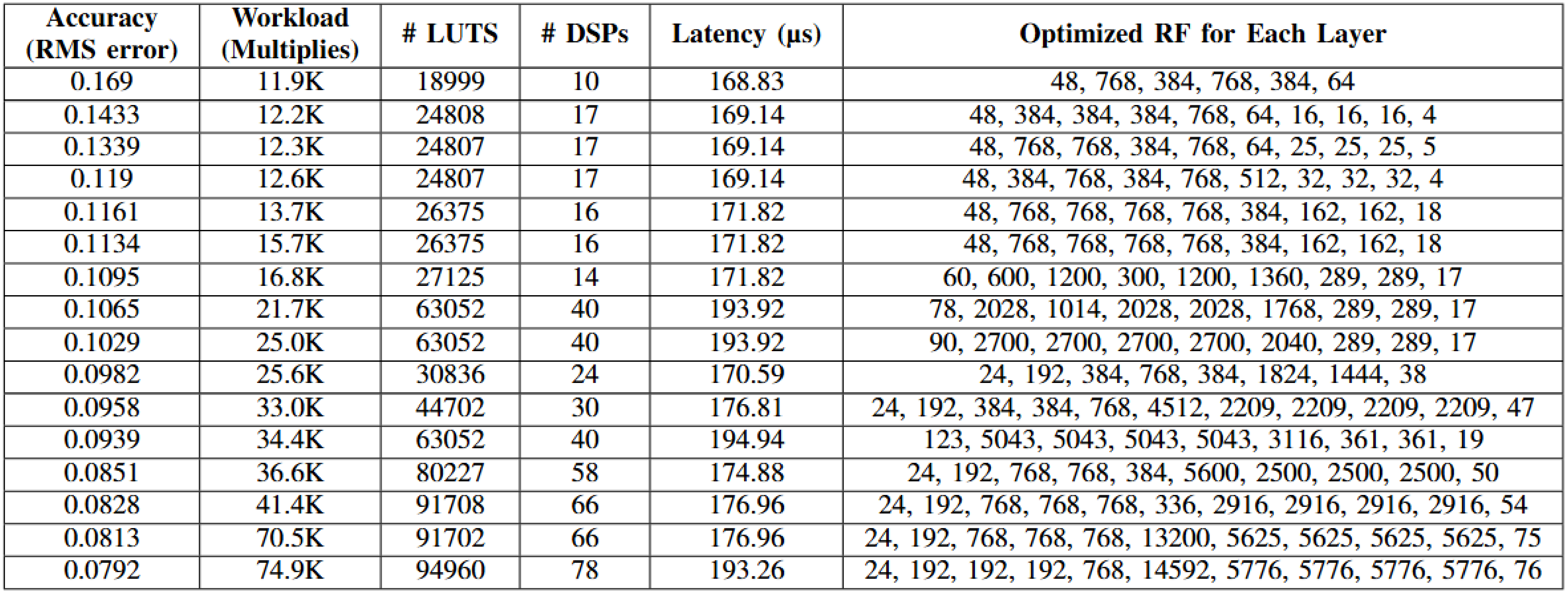
RF=? RF=? RF=? RF=? RF=?

$$\text{Minimize:} \quad \sum_{i \in \text{layers}} \left(\widehat{\text{LUTS}}_i + \widehat{\text{FF}}_i + \widehat{\text{BRAM}}_i + \widehat{\text{DSP}}_i \right)$$

$$\text{Subject to:} \quad \sum_{i \in \text{layers}} \widehat{\text{latency}}_i \le 50000$$

(based on linearized RF models)

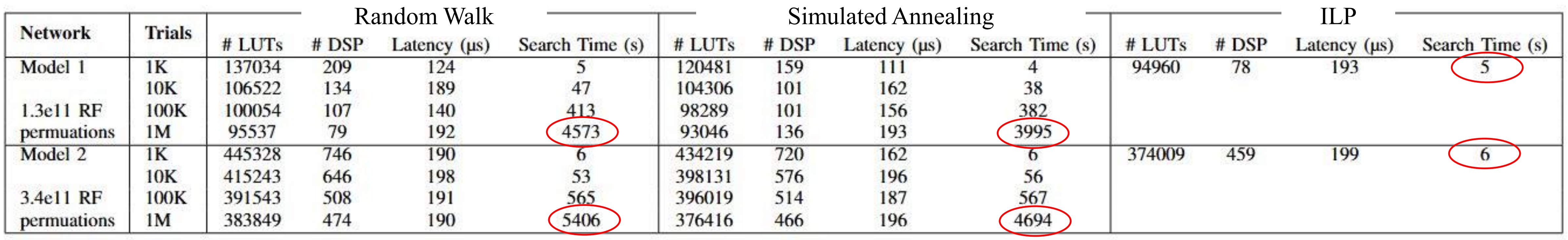
TRAINING AND DEPLOYMENT RESULTS FOR PARETO OPTIMAL NETWORKS



Accuracy (RMS error)	Workload (Multiplies)	# LUTS	# DSPs	Latency (µs)	Optimized RF for Each Layer
0.169	11.9K	18999	10	168.83	48, 768, 384, 768, 384, 64
0.1433	12.2K	24808	17	169.14	48, 384, 384, 384, 768, 64, 16, 16, 16, 4
0.1339	12.3K	24807	17	169.14	48, 768, 768, 384, 768, 64, 25, 25, 25, 5
0.119	12.6K	24807	17	169.14	48, 384, 768, 384, 768, 512, 32, 32, 32, 4
0.1161	13.7K	26375	16	171.82	48, 768, 768, 768, 768, 384, 162, 162, 18
0.1134	15.7K	26375	16	171.82	48, 768, 768, 768, 768, 384, 162, 162, 18
0.1095	16.8K	27125	14	171.82	60, 600, 1200, 300, 1200, 1360, 289, 289, 17
0.1065	21.7K	63052	40	193.92	78, 2028, 1014, 2028, 2028, 1768, 289, 289, 17
0.1029	25.0K	63052	40	193.92	90, 2700, 2700, 2700, 2700, 2040, 289, 289, 17
0.0982	25.6K	30836	24	170.59	24, 192, 384, 768, 384, 1824, 1444, 38
0.0958	33.0K	44702	30	176.81	24, 192, 384, 384, 768, 4512, 2209, 2209, 2209, 2209, 47
0.0939	34.4K	63052	40	194.94	123, 5043, 5043, 5043, 5043, 3116, 361, 361, 19
0.0851	36.6K	80227	58	174.88	24, 192, 768, 768, 384, 5600, 2500, 2500, 2500, 50
0.0828	41.4K	91708	66	176.96	24, 192, 768, 768, 768, 336, 2916, 2916, 2916, 2916, 54
0.0813	70.5K	91702	66	176.96	24, 192, 768, 768, 768, 13200, 5625, 5625, 5625, 5625, 75
0.0792	74.9K	94960	78	193.26	24, 192, 192, 192, 768, 14592, 5776, 5776, 5776, 5776, 76

ILP VS STOCHASTIC SEARCH

- Random Walk and Simulated Annealing:
  - Same linear cost and performance models
  - Same latency constraint and resource minimization
  - Two different DROPBEAR networks
  - 1K to 1M iterations



Network	Trials	Random Walk				Simulated Annealing				ILP			
		# LUTs	# DSP	Latency (µs)	Search Time (s)	# LUTs	# DSP	Latency (µs)	Search Time (s)	# LUTs	# DSP	Latency (µs)	Search Time (s)
Model 1	1K	137034	209	124	5	120481	159	111	4	94960	78	193	5
	10K	106522	134	189	47	104306	101	162	38				
1.3e11 RF	100K	100054	107	140	413	98289	101	156	382				
permuations	1M	95537	79	192	4573	93046	136	193	3995				
Model 2	1K	445328	746	190	6	434219	720	162	6	374009	459	199	6
	10K	415243	646	198	53	398131	576	196	56				
3.4e11 RF	100K	391543	508	191	565	396019	514	187	567				
permuations	1M	383849	474	190	5406	376416	466	196	4694				

CONCLUSIONS AND FUTURE WORK

- N-TORC combines hyperparameter search with architecture optimization
- Designed for high-rate (real-time) machine learning
- Limited to small models due to on-chip memory constraints
- Future work:
  - Move to alternative backend that supports dataflow with off-chip memory access (e.g. InTAR)
  - Incorporate quantization into optimizer and cost/performance models

THANK YOU!
Q&A

Molinaroli College of Engineering and Computing
UNIVERSITY OF SOUTH CAROLINA