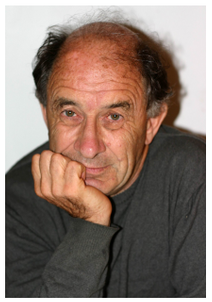
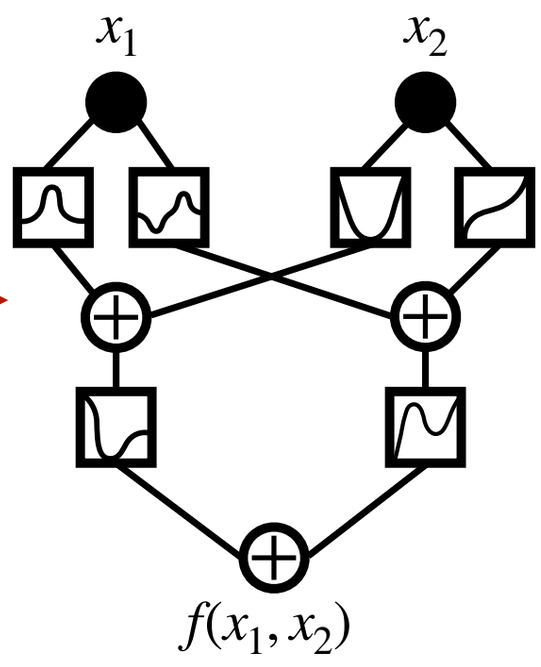
KANELÉ: Kolmogorov–Arnold Networks for Efficient LUT-based Evaluation

Kolmogorov-Arnold Representation Theorem

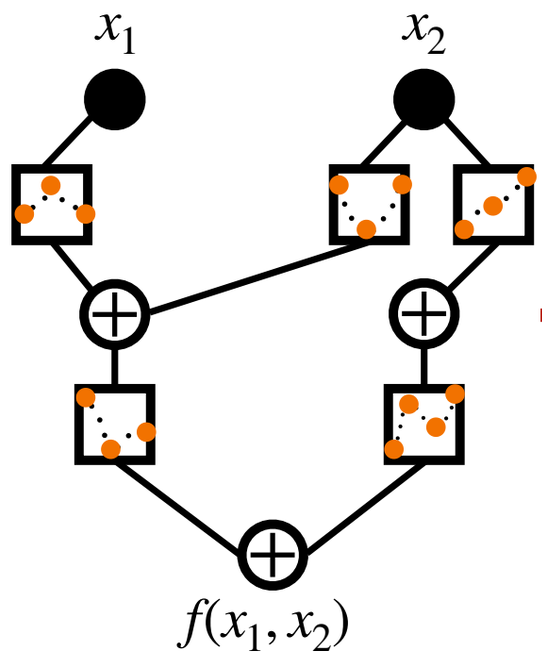



Multivariate $f \approx$ clever sum of 1D curves.

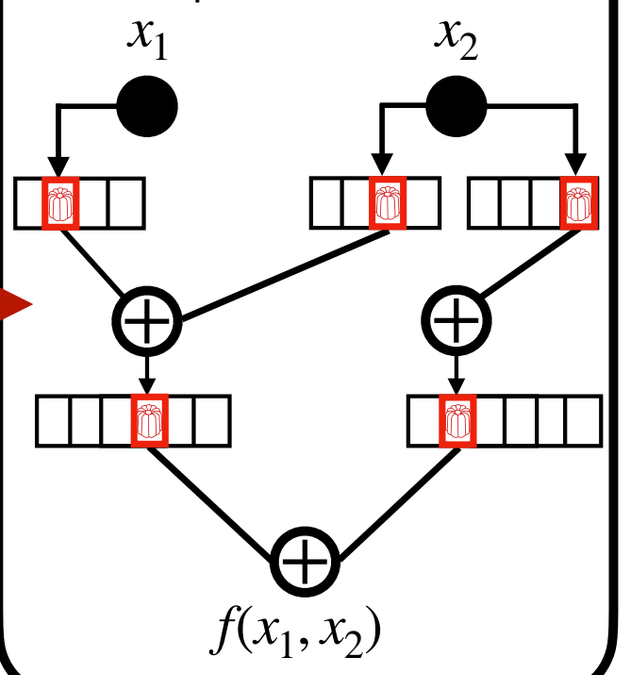
Kolmogorov-Arnold Network (KAN)



Quantization & Pruning

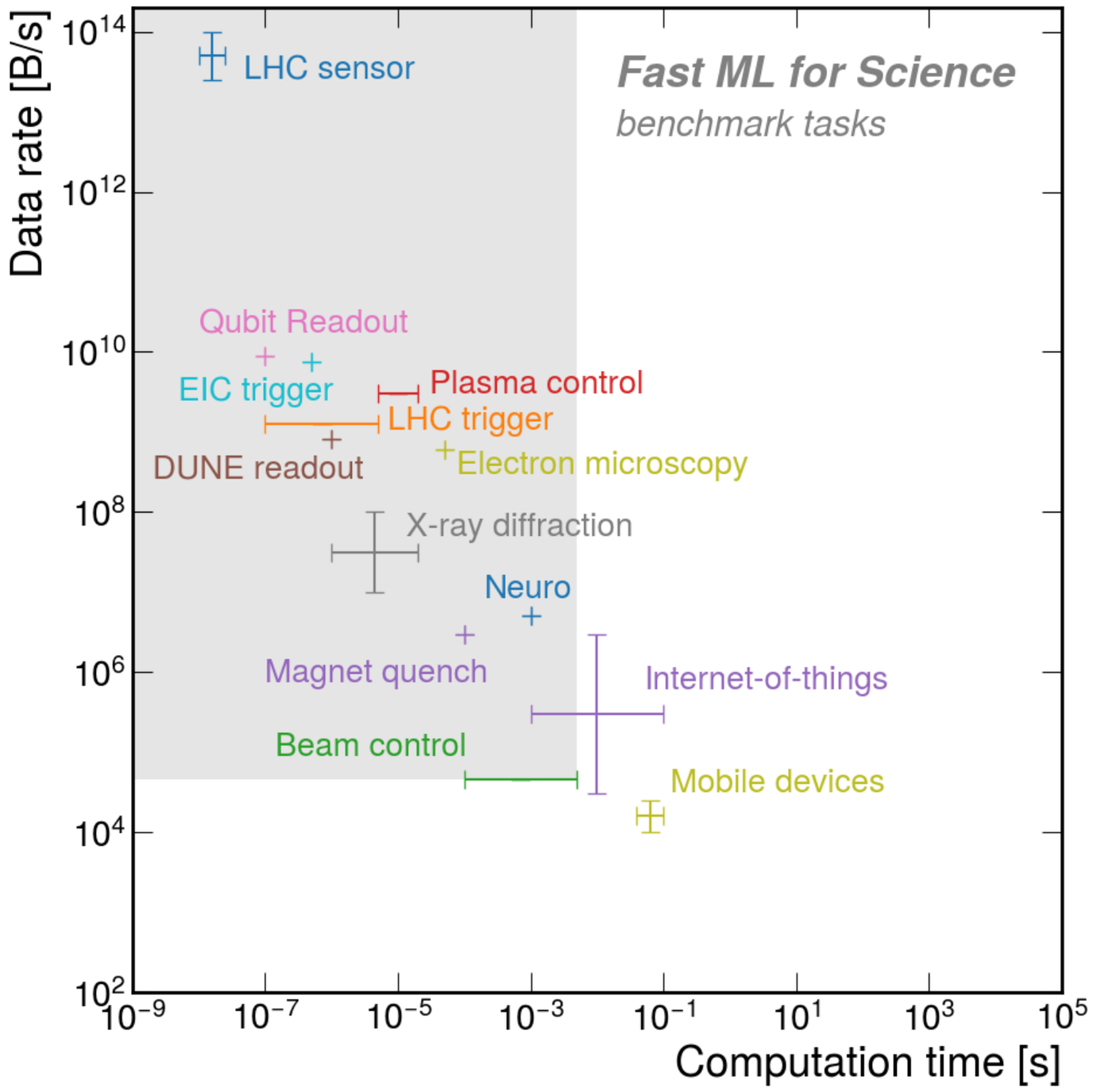


Efficient LUT-based implementation



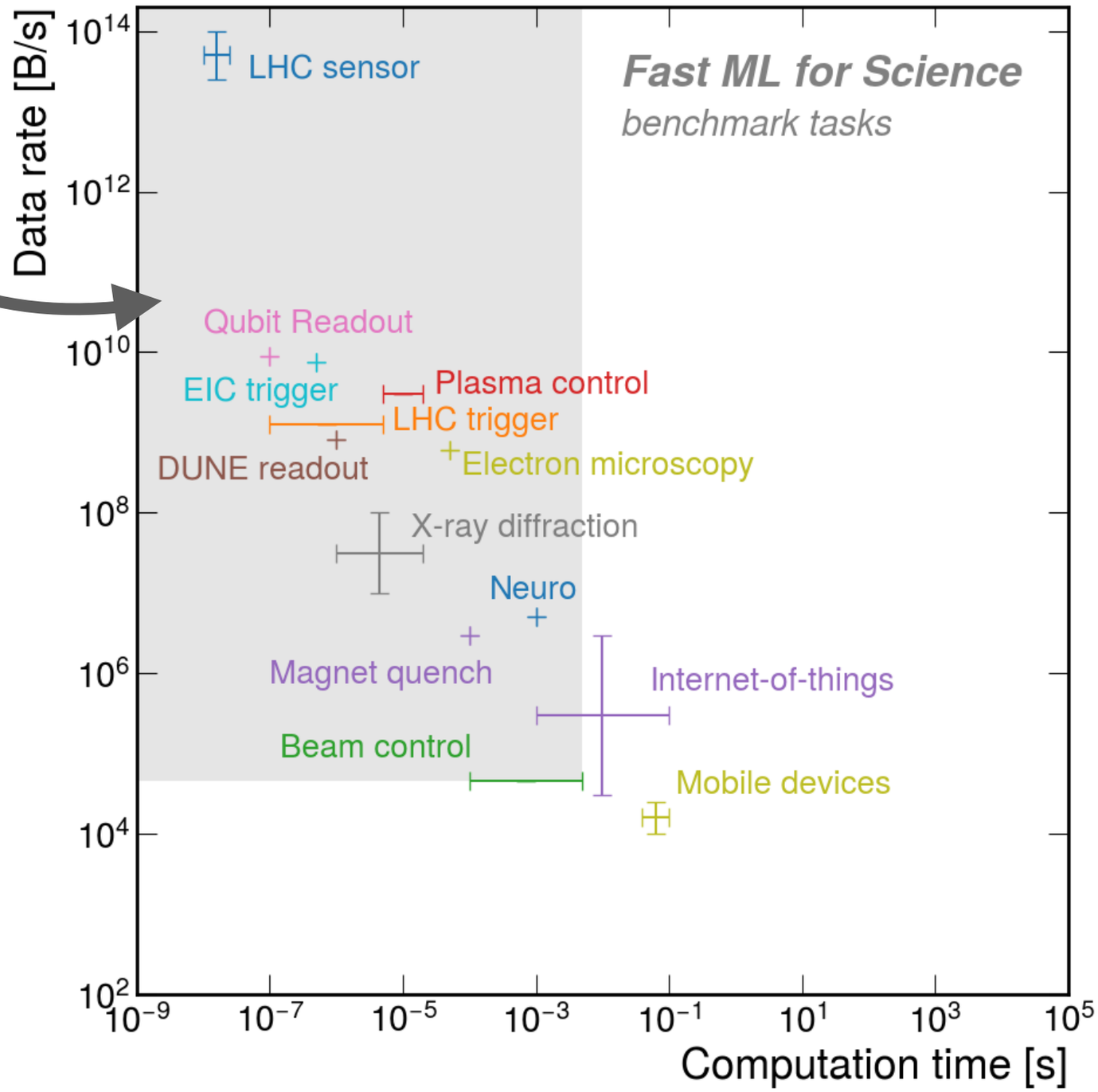
Duc Hoang, Aarush Gupta, Philip Harris.
dhoang@mit.edu

The need for ultrafast and efficient inference



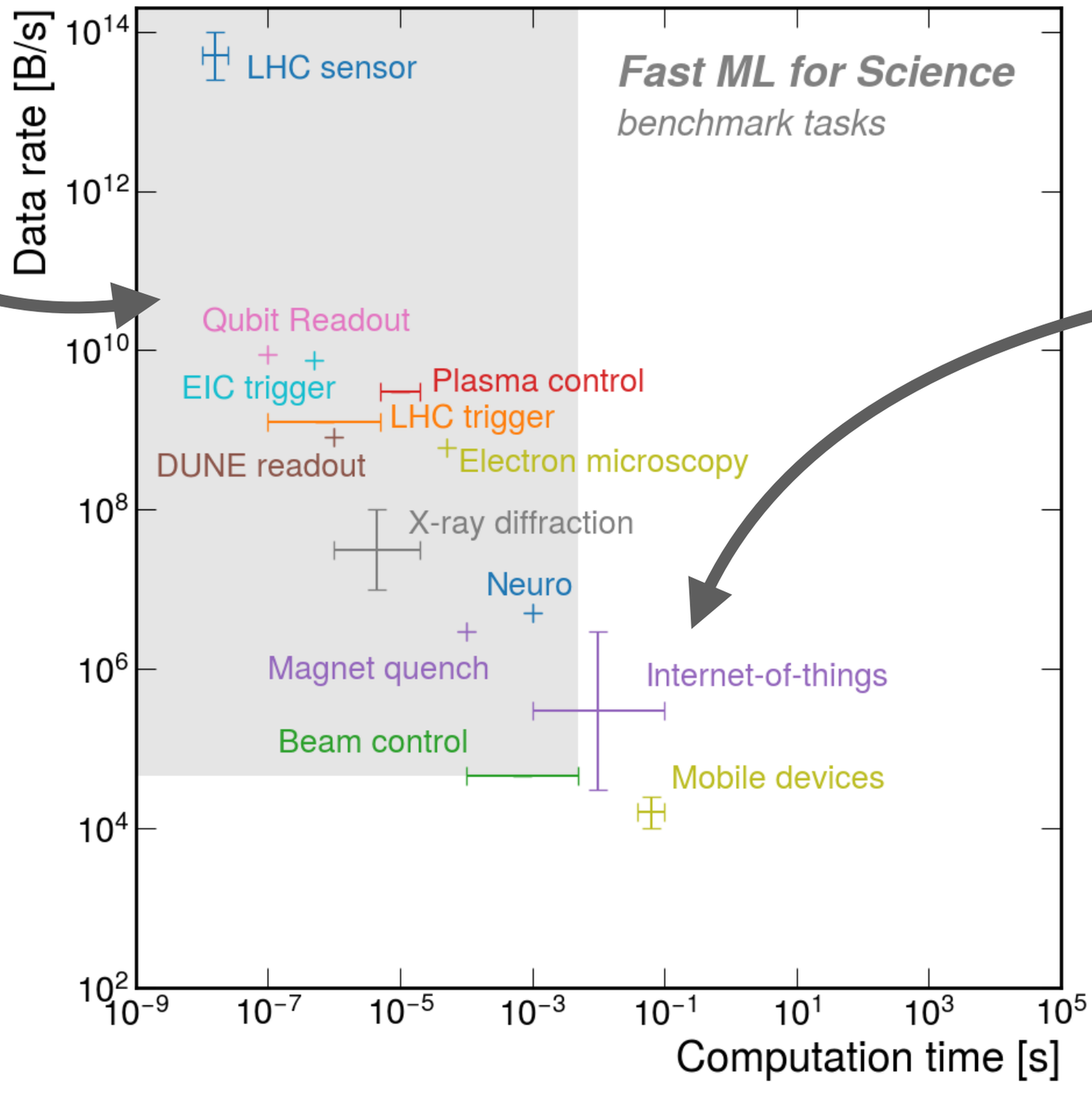
The need for ultrafast and efficient inference

Ultrafast



The need for ultrafast and efficient inference

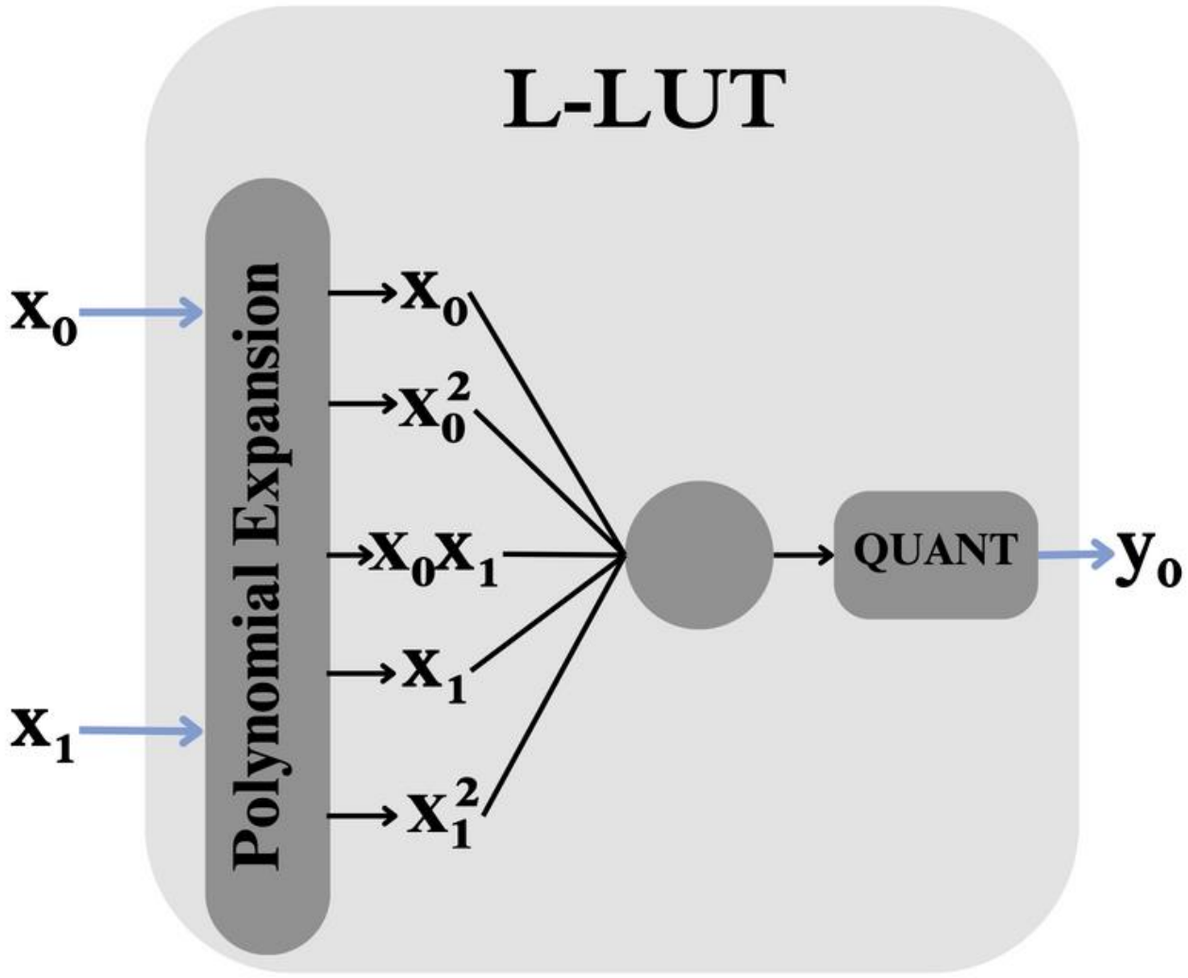
Ultrafast



Efficient

LUT based NNs offer a solution

[Andronic et al., PolyLUT.](#) **Best paper at FPT'23**

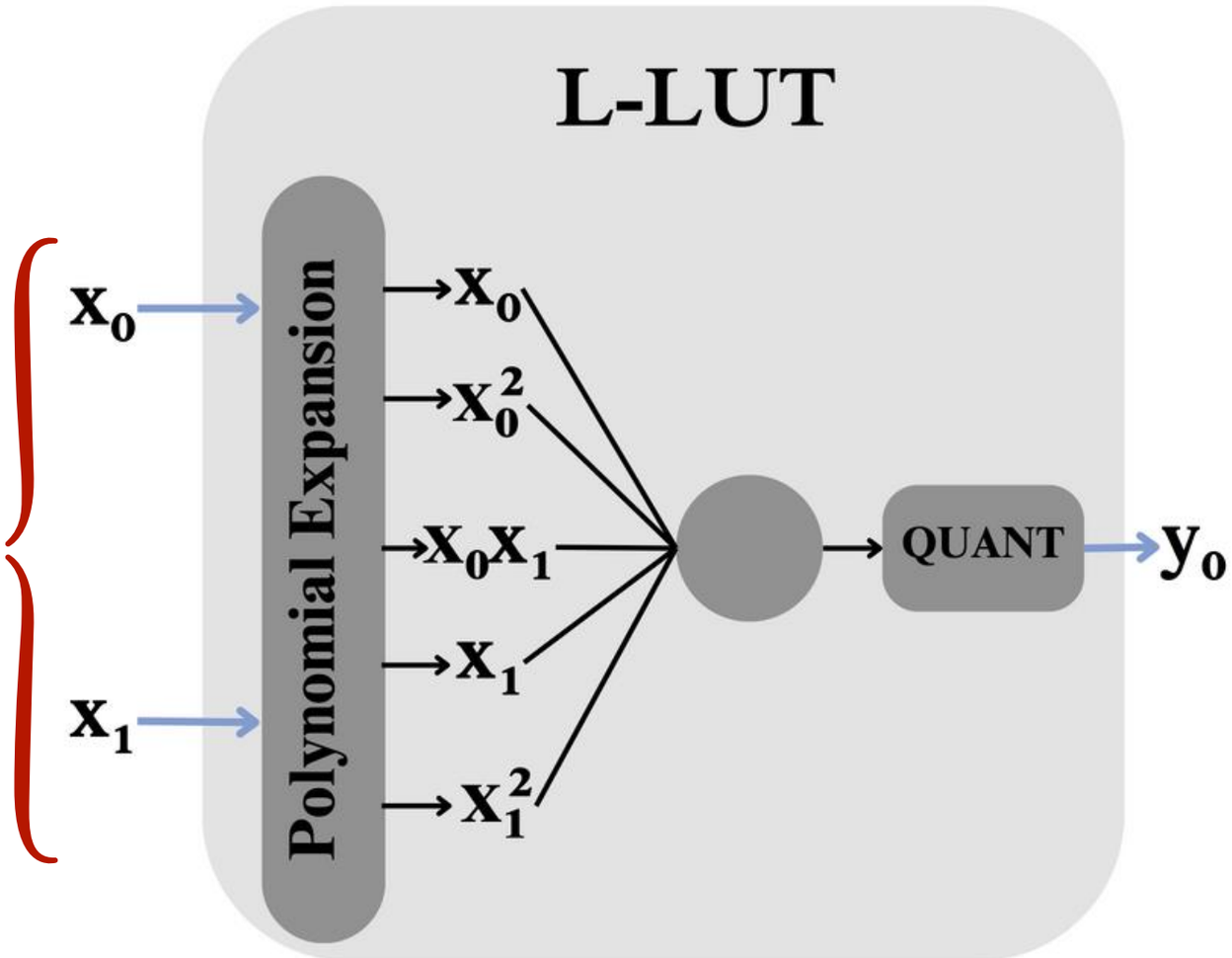


L-LUT hides a multivariate polynomial.

But multivariate LUTs scale *exponentially*

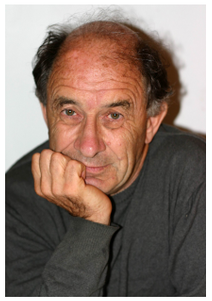
[Andronic et al., PolyLUT.](#) **Best paper at FPT'23**

LUT resources scale ***exponentially*** w.r.t to number of input features.



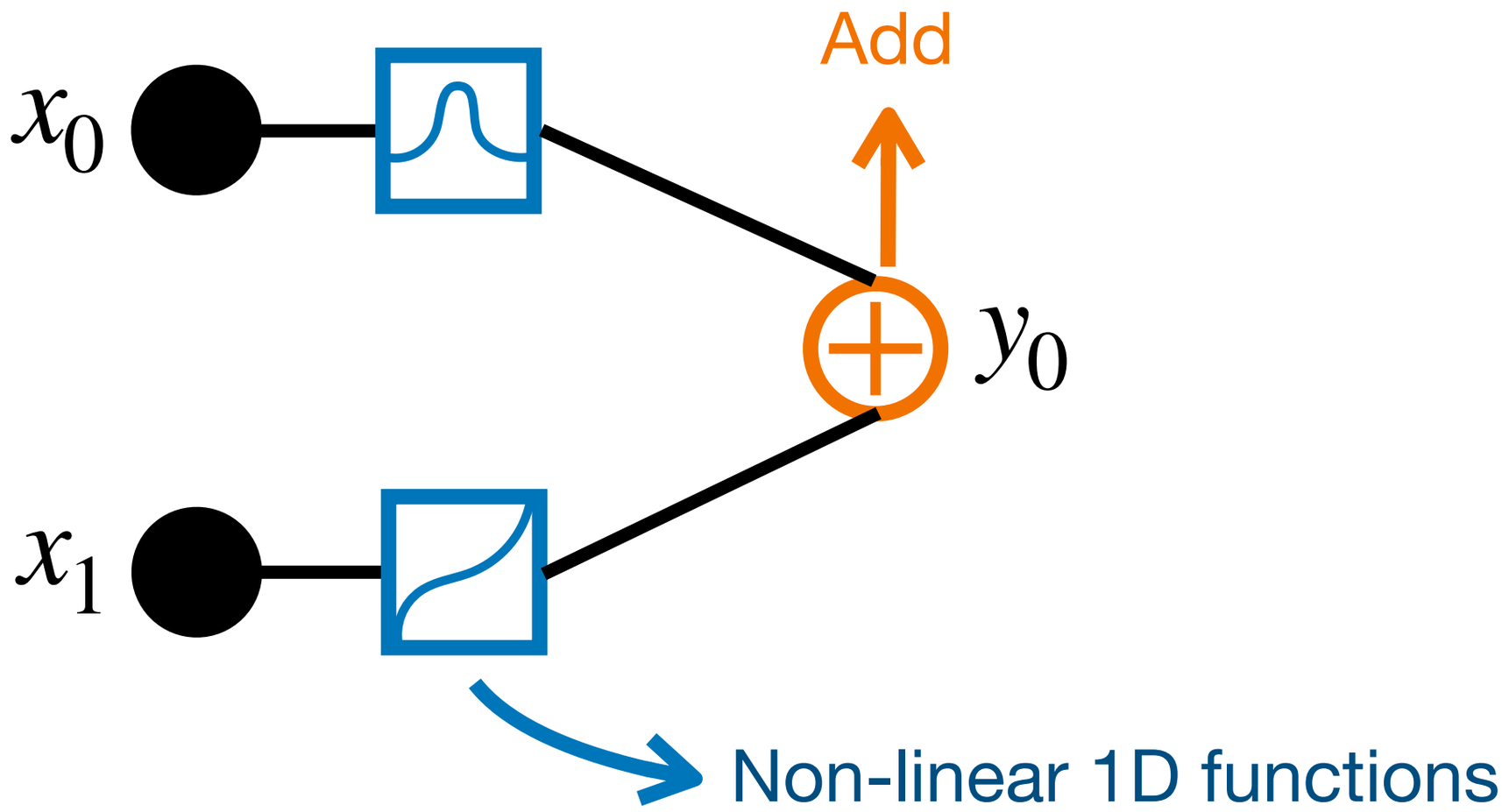
L-LUT hides a multivariate polynomial.

Kolmogorov-Arnold Representation Theorem



Multivariate $f \approx$ clever sum of 1D curves.

Insight 1: Multivariate → sum of 1D functions for *linear scaling*



Kolmogorov-Arnold Representation Theorem

Kolmogorov-Anorld Networks (KAN)

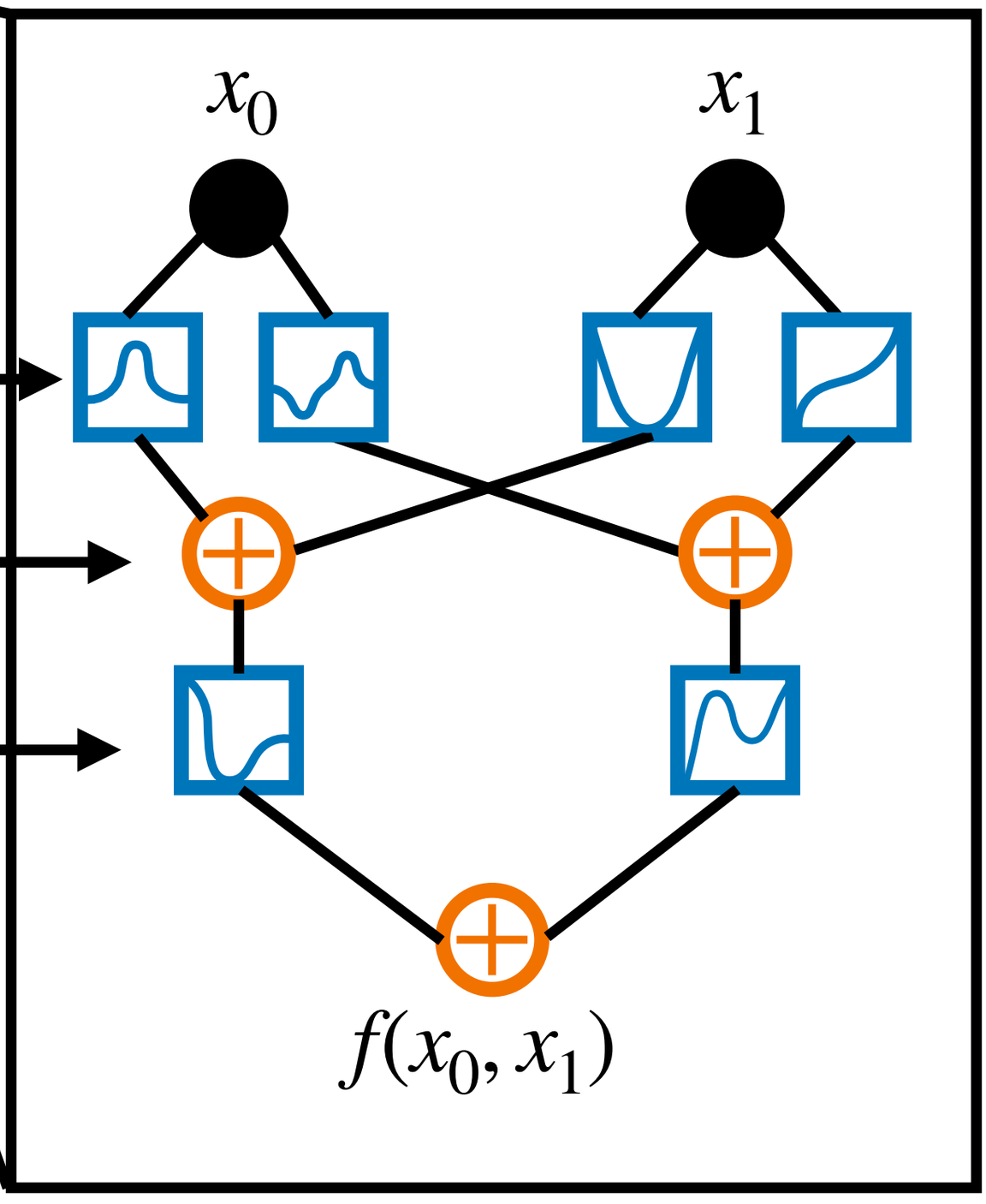
$f \approx$ clever sum of 1D curves.

1D functions on ***edges***

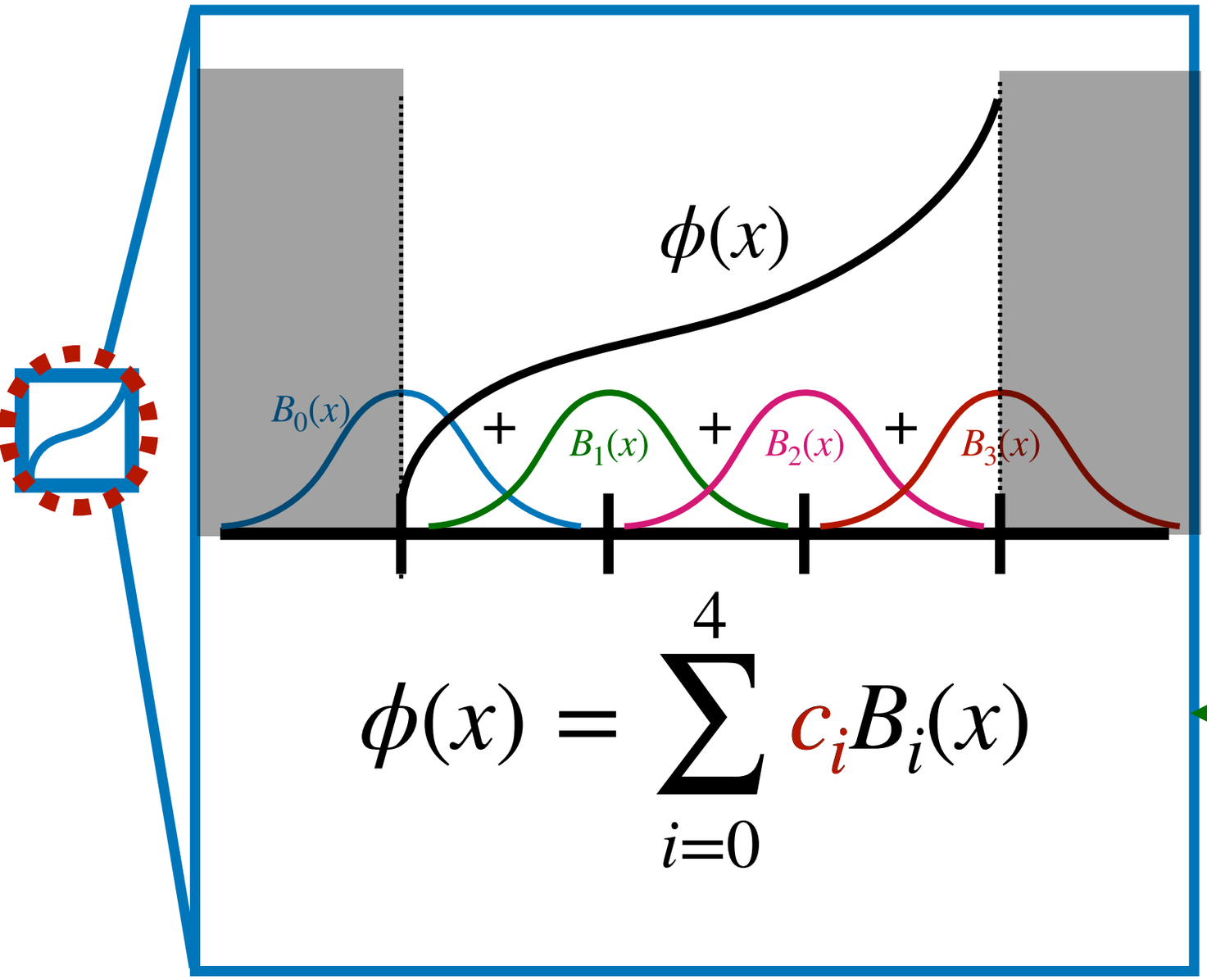
Sum on ***nodes***

1D function on ***edges***

Liu et al. ArXiv: 2404.19756



KAN inference is notoriously hard



GPU latency **15-100x slower than MLPs.**

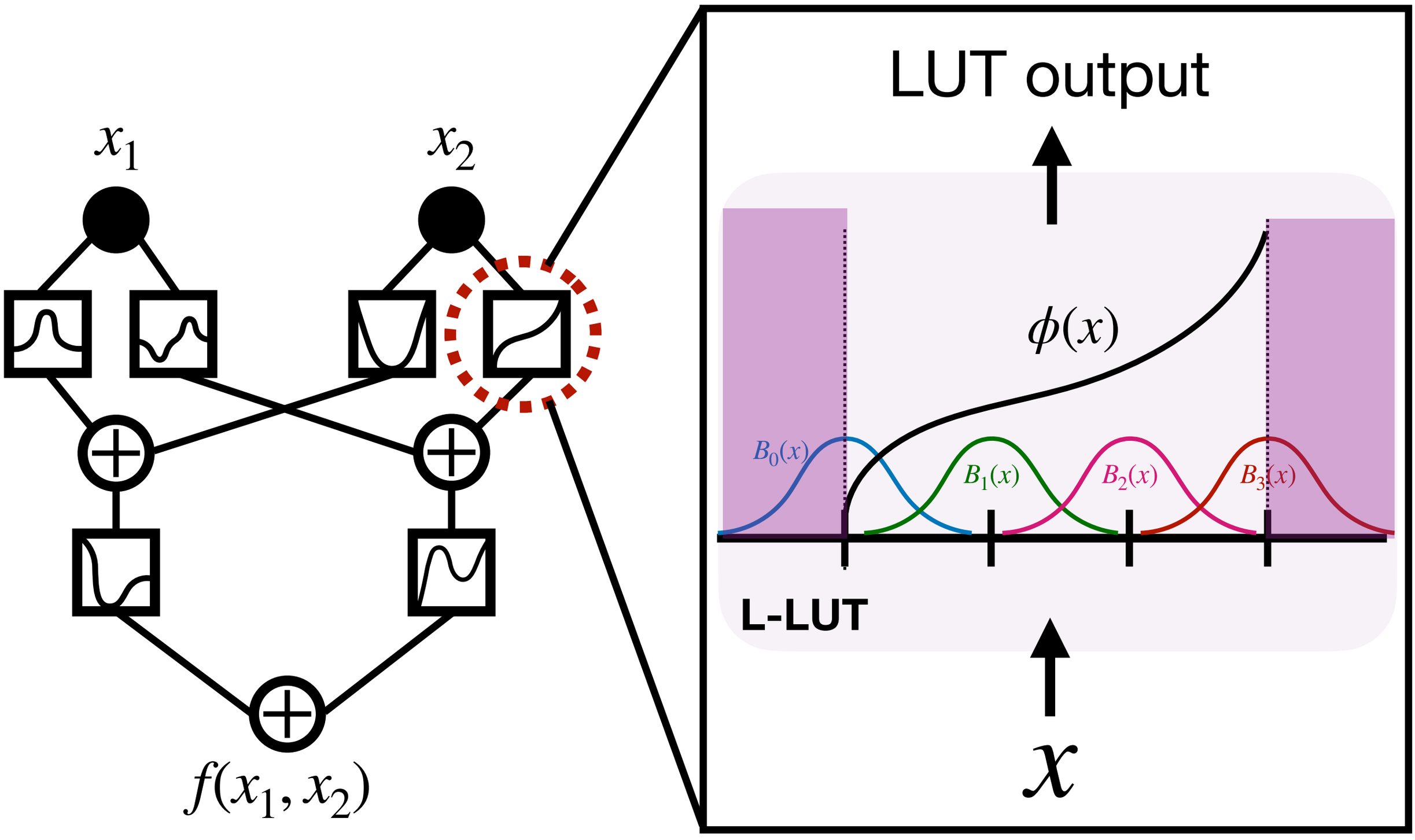
ArXiv: 2407.11075

FPGA is **"impractical"**

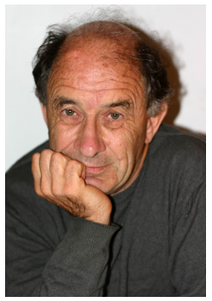
ArXiv: 2407.17790

Prior works ignore **abstraction,** which makes KAN **native on FPGA.**

Insight 2: 1D functions map directly to LUT primitives

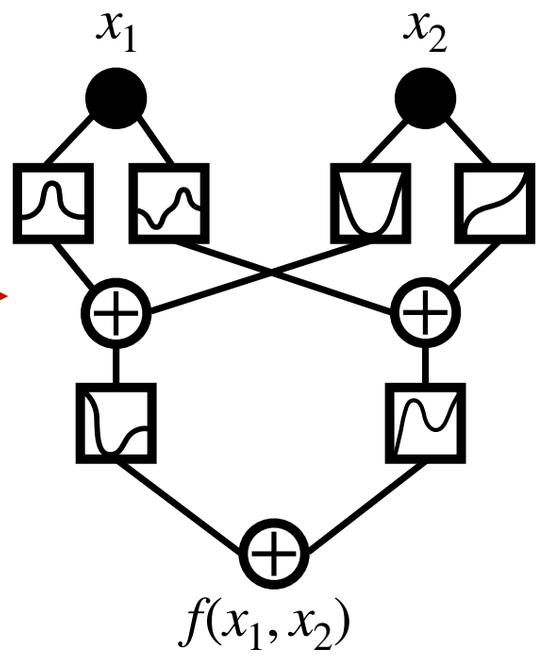


Kolmogorov-Arnold Representation Theorem

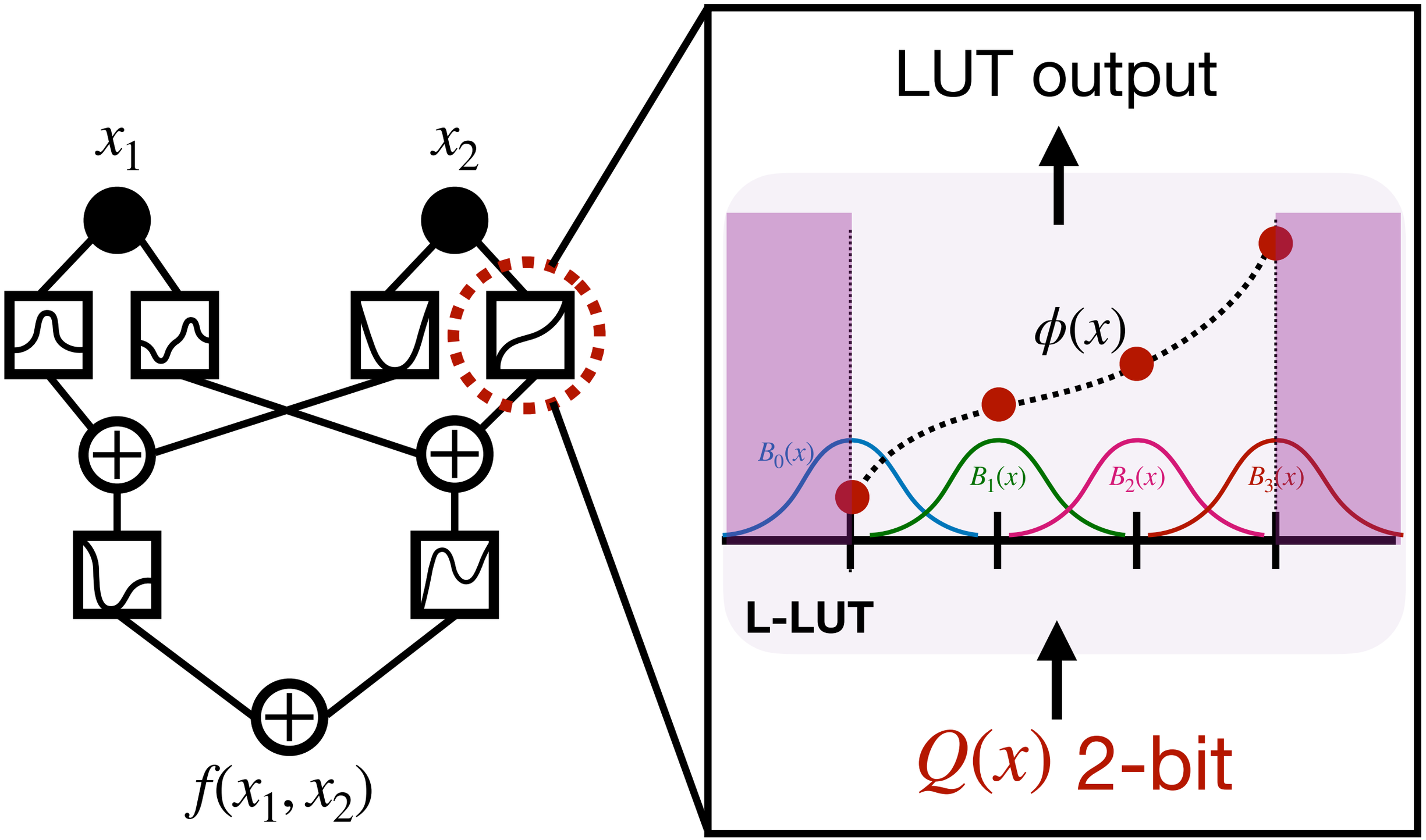


Multivariate $f \approx$ clever sum of 1D curves.

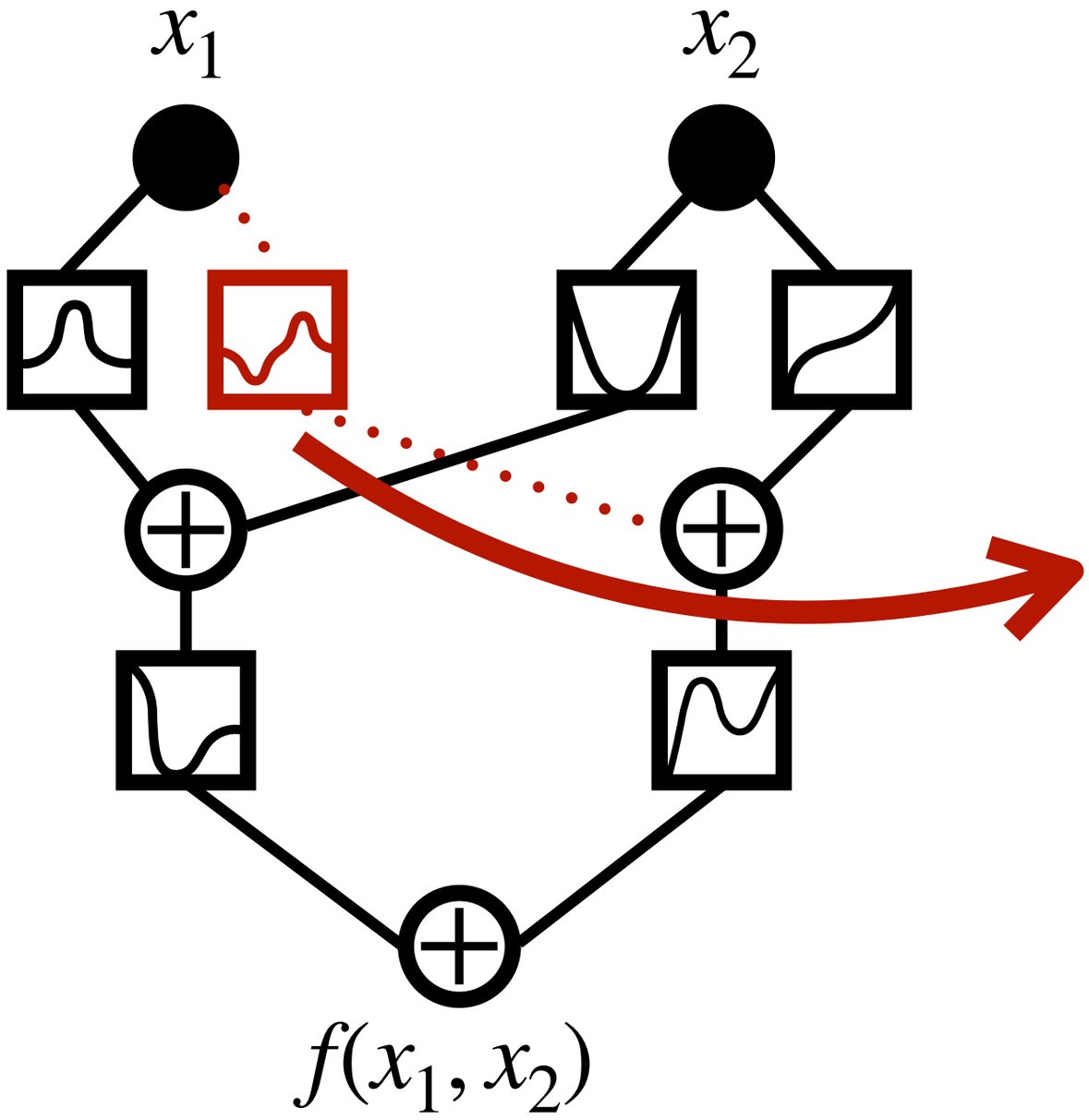
Kolmogorov-Arnold Network (KAN)



Insight 3: Learned functions are extremely natural to be quantized.



Insight 4: Additive edges → Pruning is clean

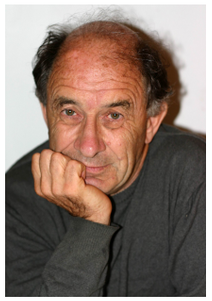


Magnitude-based pruning

Pre-set universal threshold

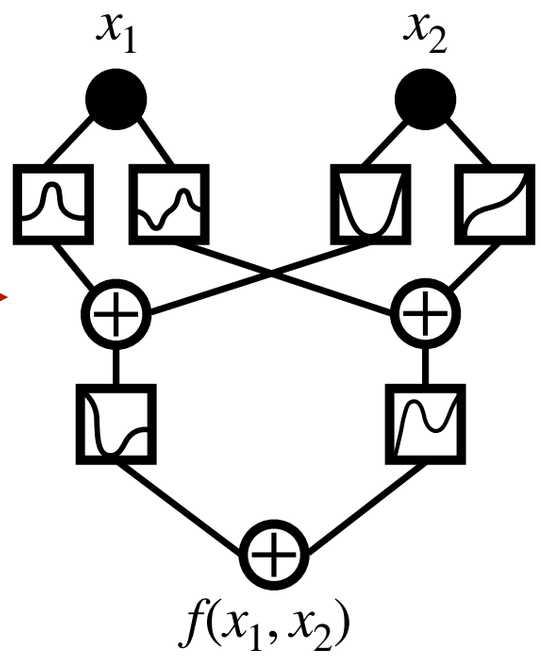
If $\|\phi\|_2$ **small**, prune it

Kolmogorov-Arnold Representation Theorem

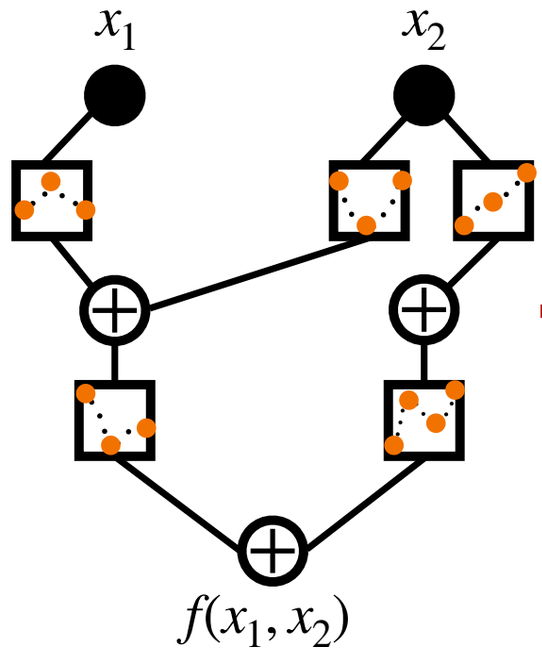


Multivariate $f \approx$ clever sum of 1D curves.

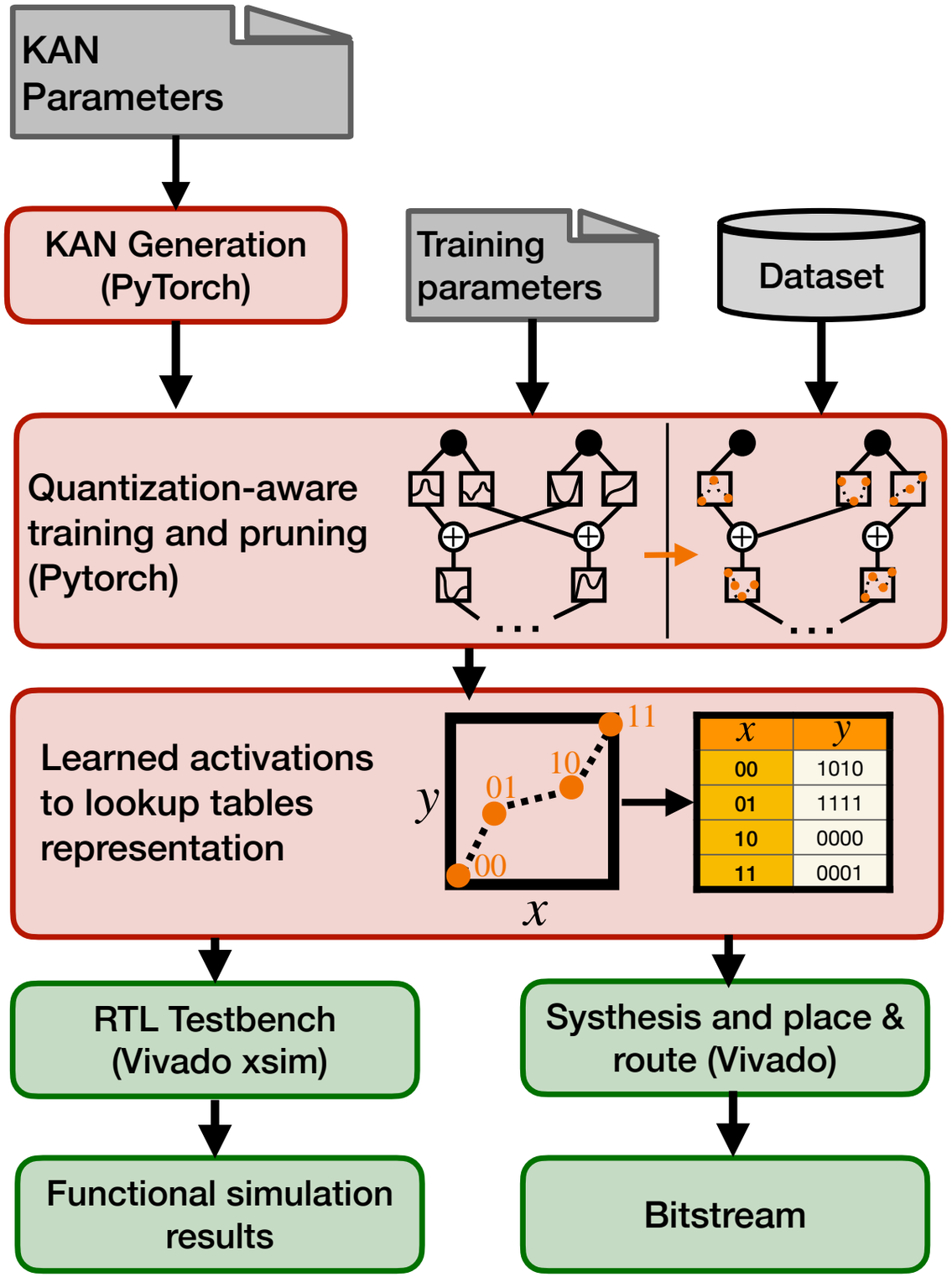
Kolmogorov-Arnold Network (KAN)



Quantization & Pruning

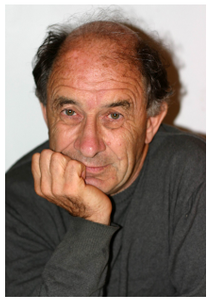


Toolflow



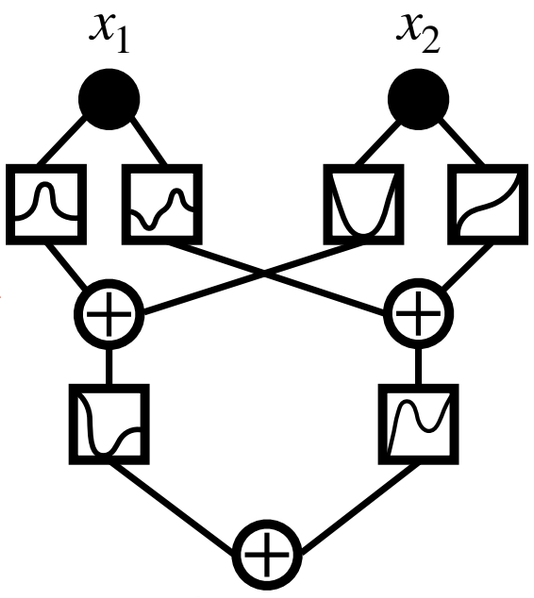
Takes ~***seconds*** to go from a trained model to RTL codes.
Many previous works take ***hours***.

Kolmogorov-Arnold Representation Theorem

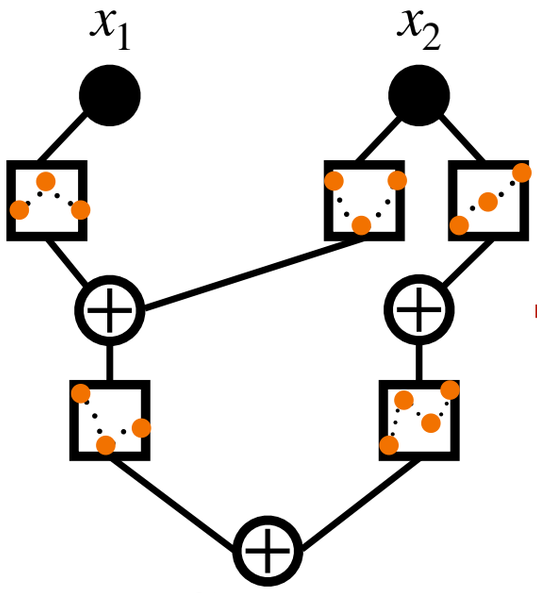


Multivariate $f \approx$ clever sum of 1D curves.

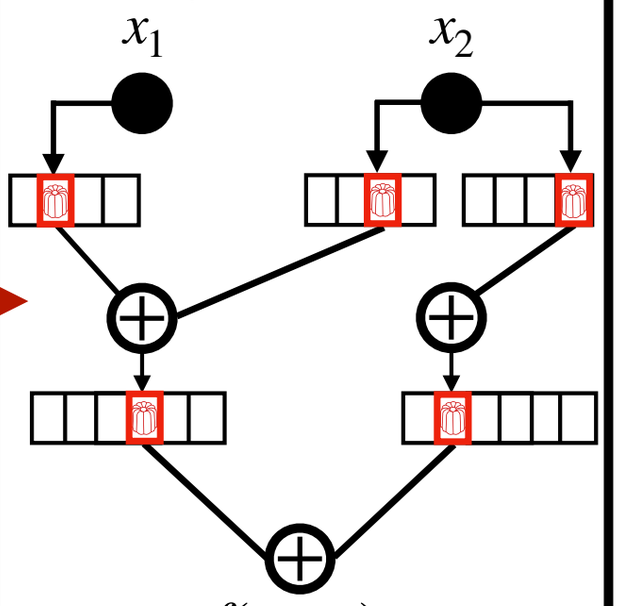
Kolmogorov-Arnold Network (KAN)



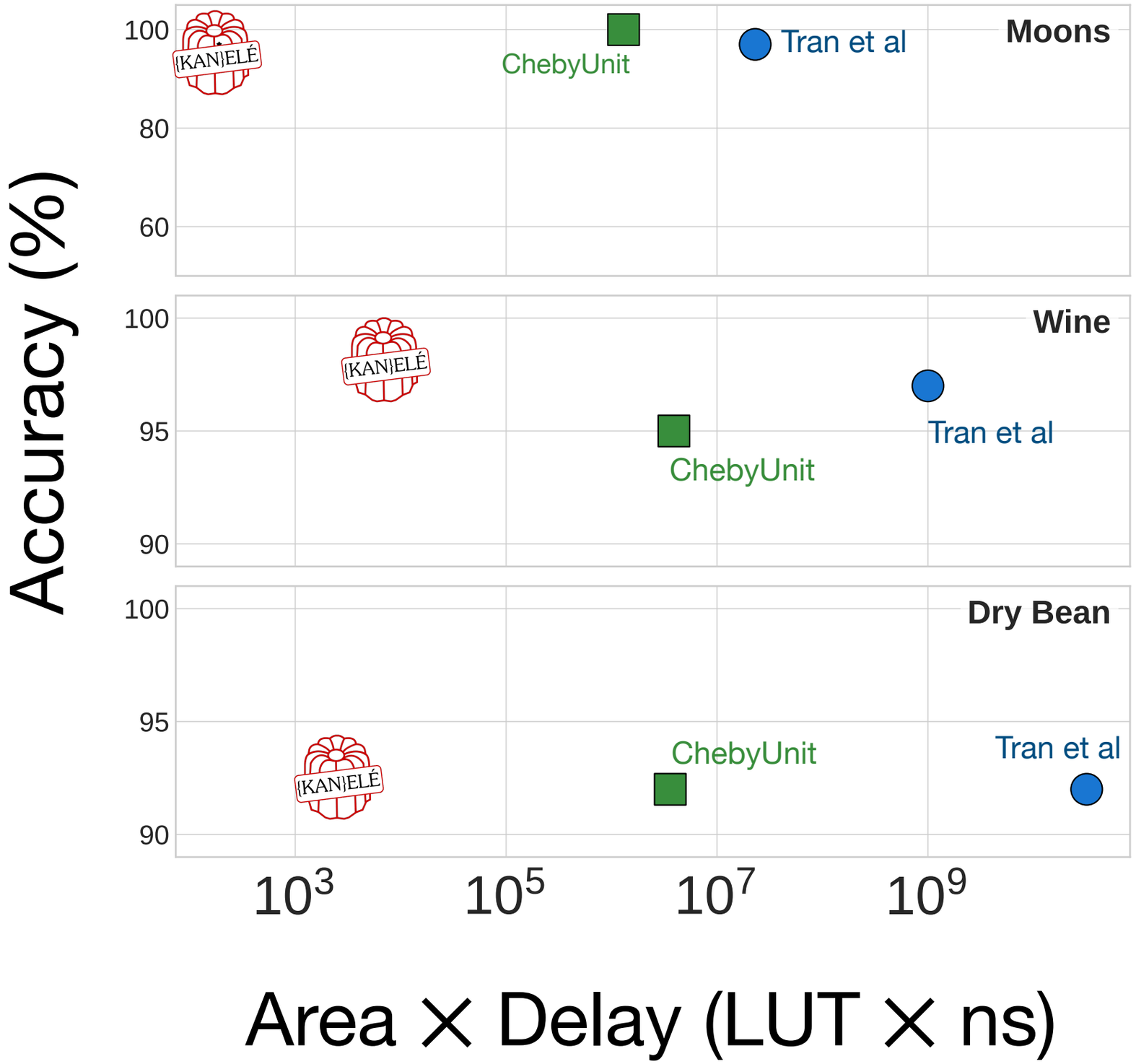
Quantization & Pruning



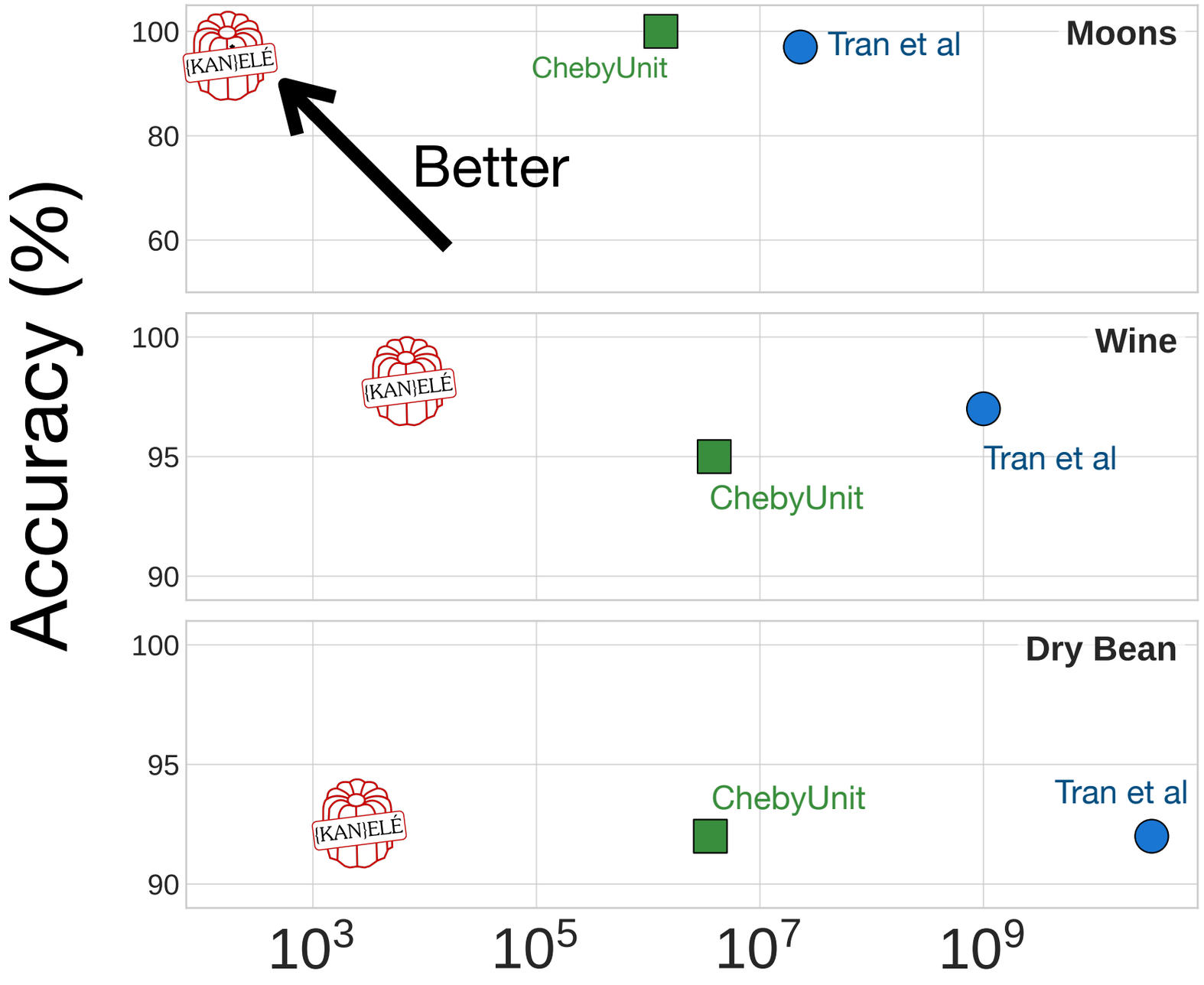
Efficient LUT-based implementation



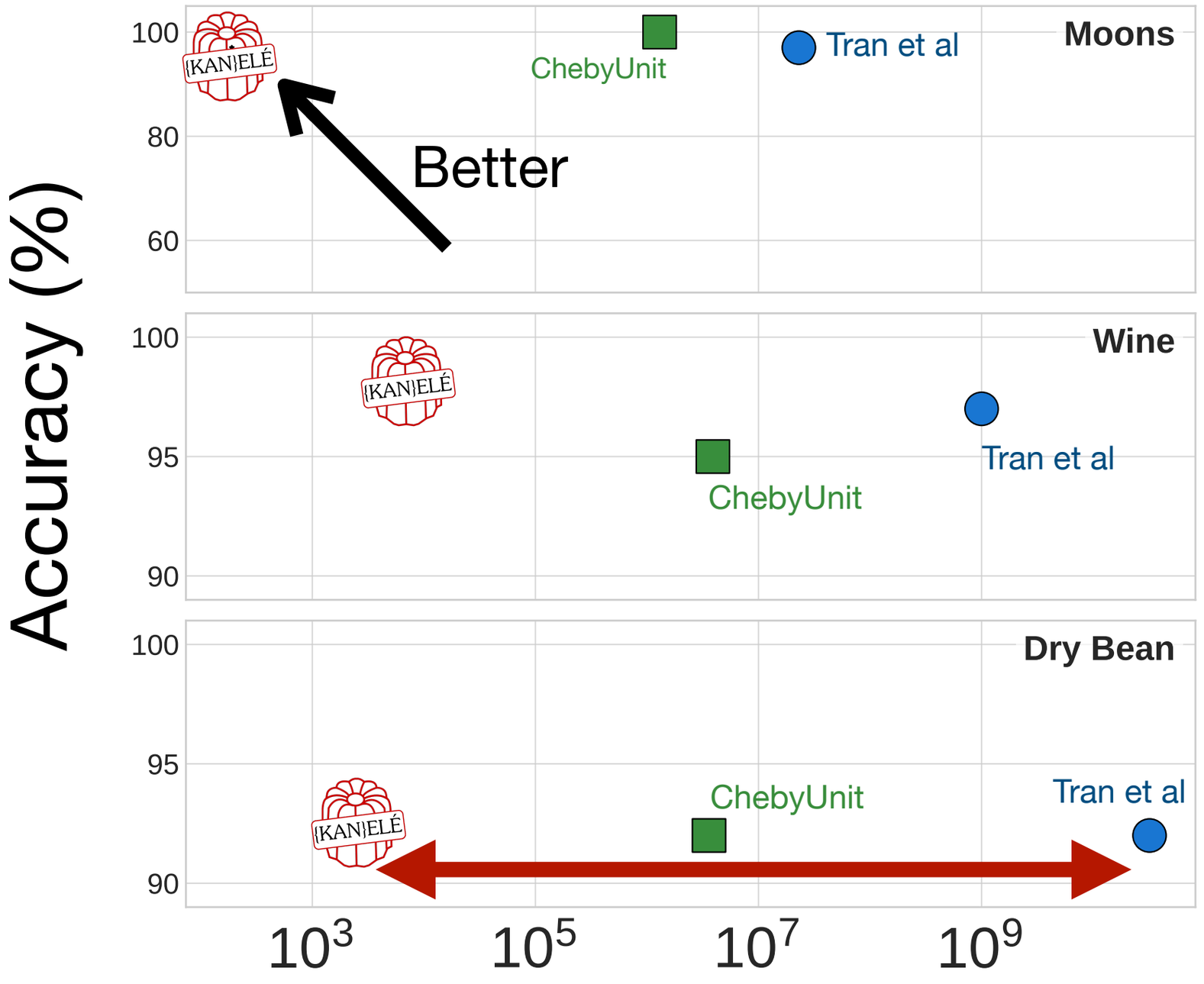
Benchmark - Previous KAN-FPGAs



Benchmark - Previous KAN-FPGAs

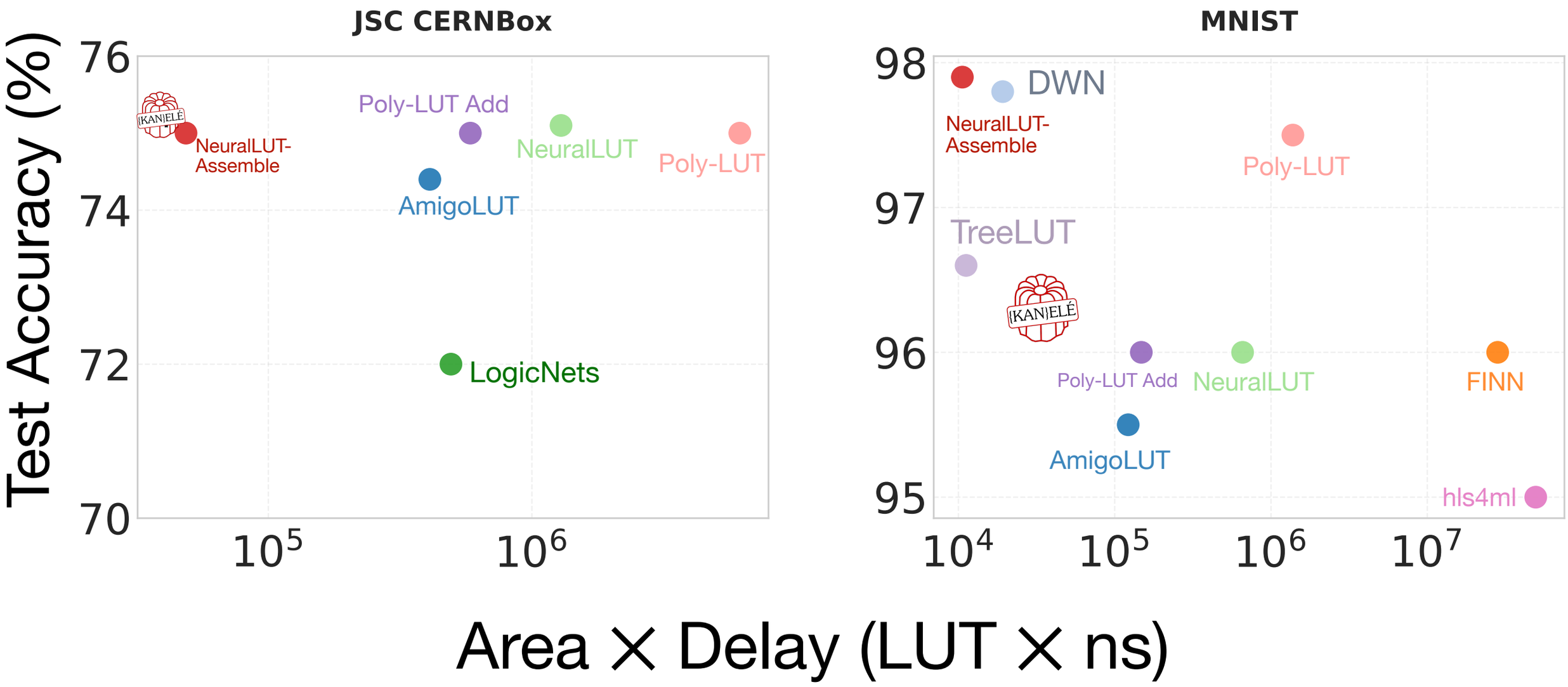


Benchmark - Previous KAN-FPGAs

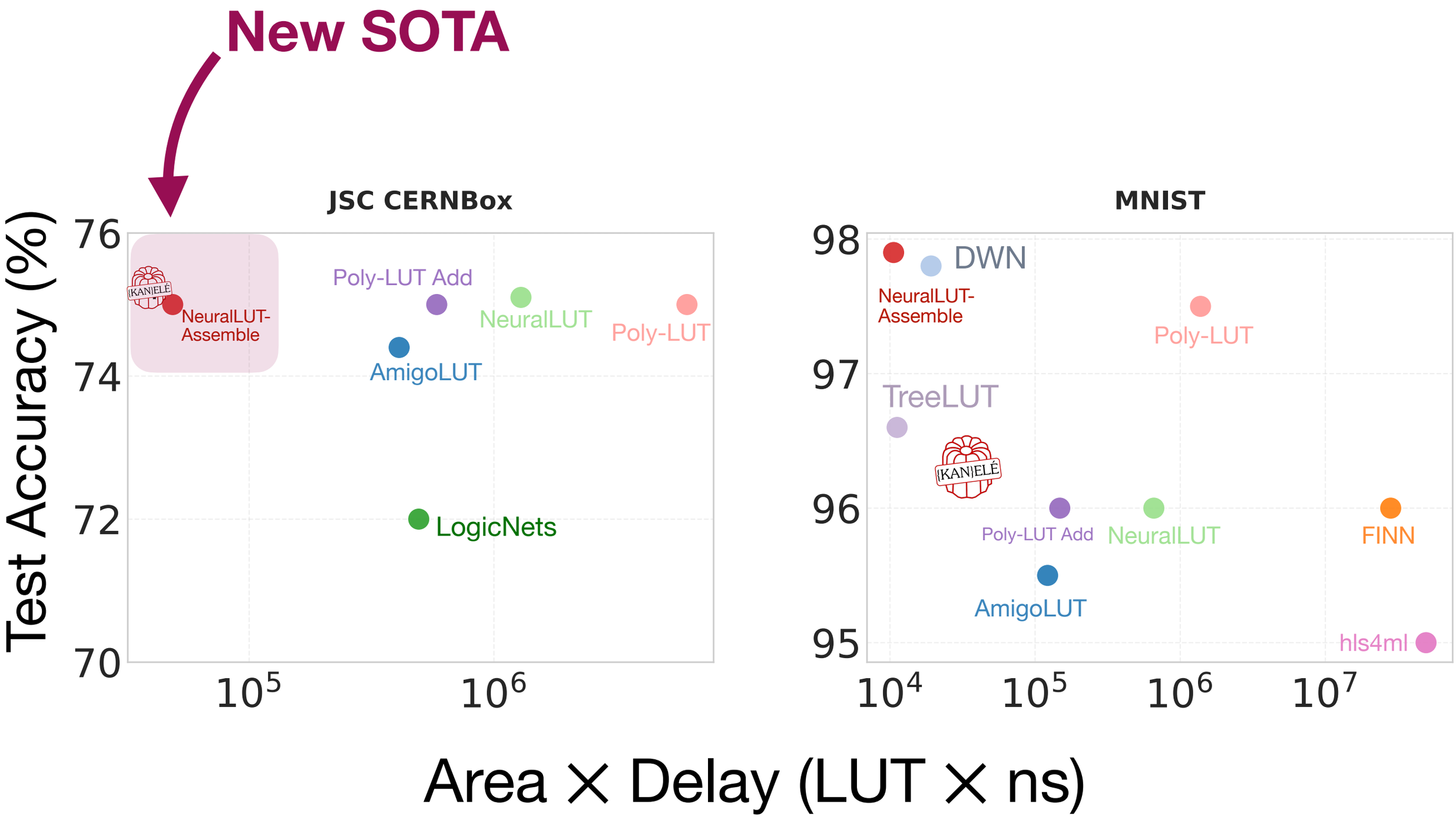


From $10^3 \rightarrow 10^7$**x** improvement compared to prior works.

Benchmark - Other LUT based NNs



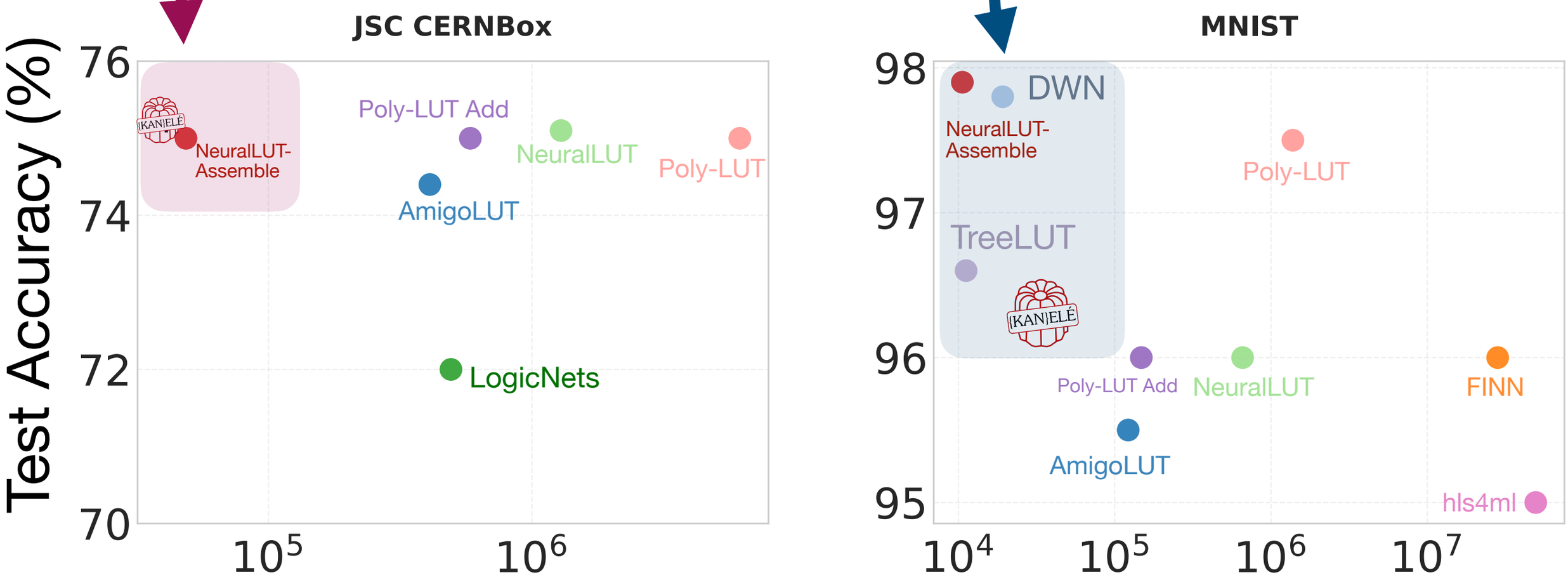
Benchmark - Other LUT based NNs



Benchmark - Other LUT based NNs

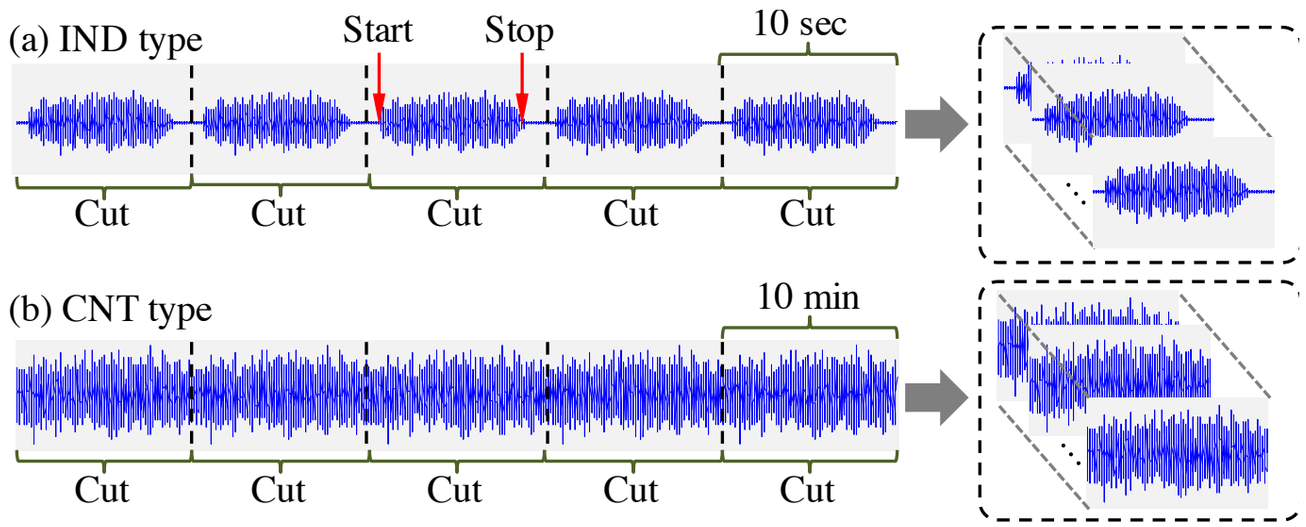
New SOTA

Competitive but not the best, **KAN CNN better?**

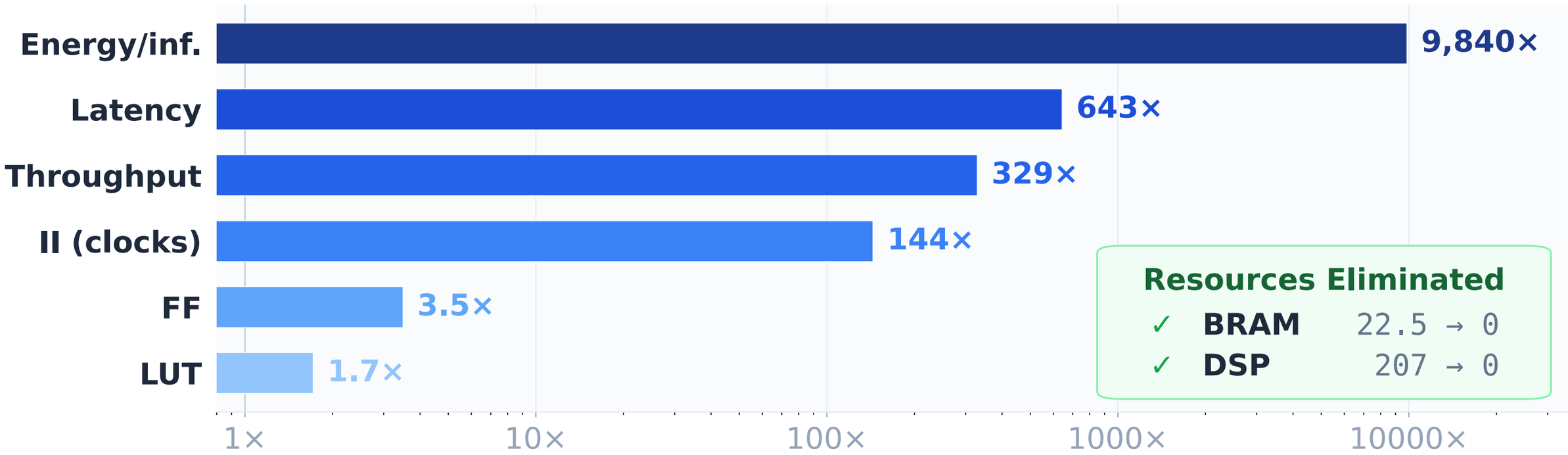


New Benchmark - MLPerf Tiny

ToyADMOS signal **autoencoder anomaly detection**

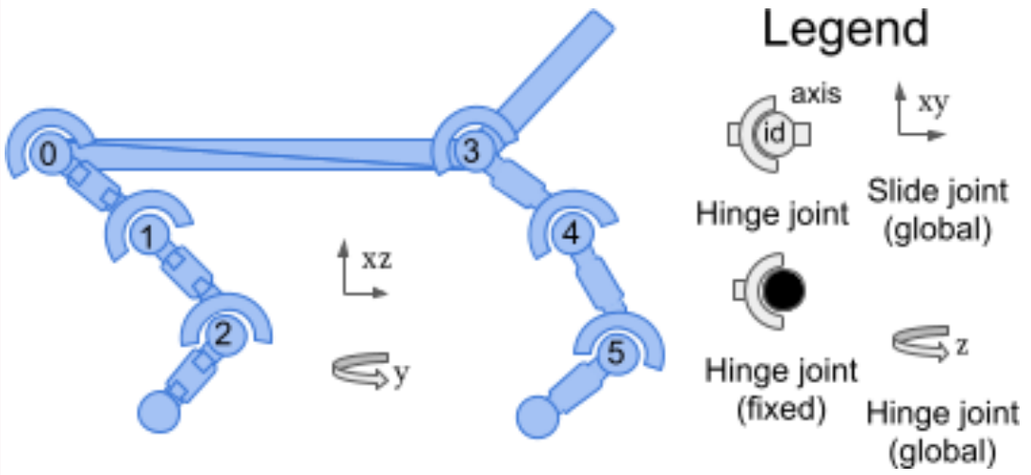


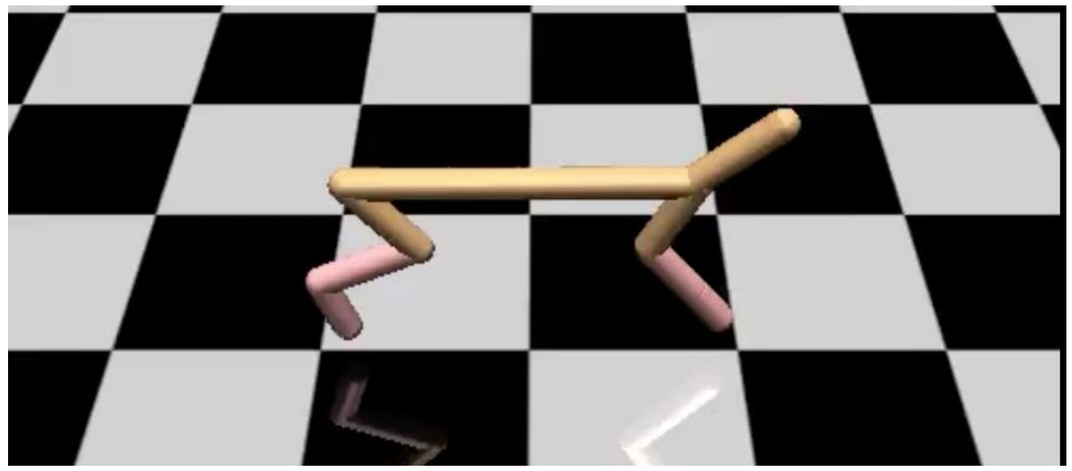
Improvement factor over hls4ml (MLPerf Tiny v0.7) - **same AUC = 0.83**

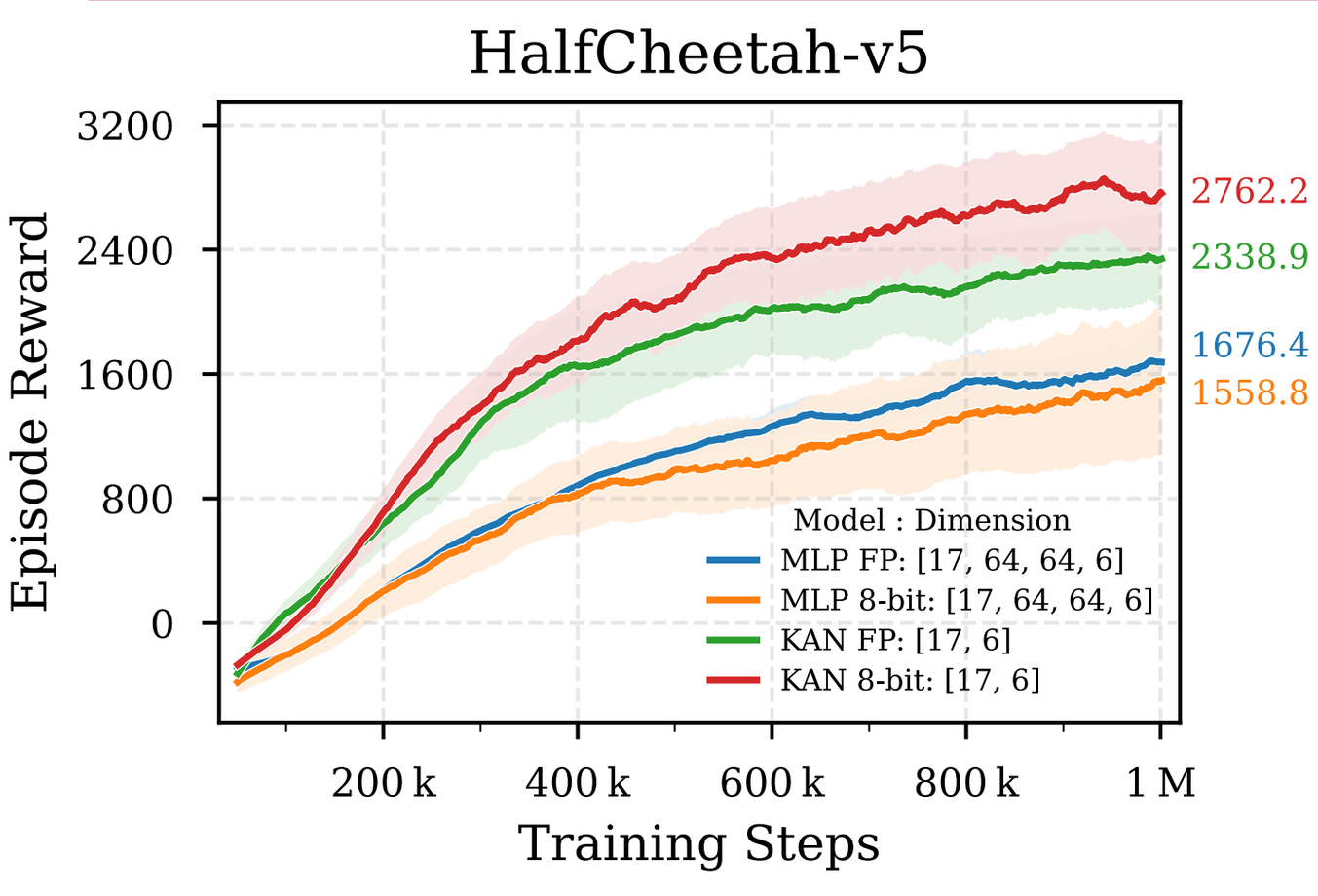


New Benchmark - Control System

OpenAI Gymnasium - HalfCheetah







KAN controller needs 5x less trainable parameters

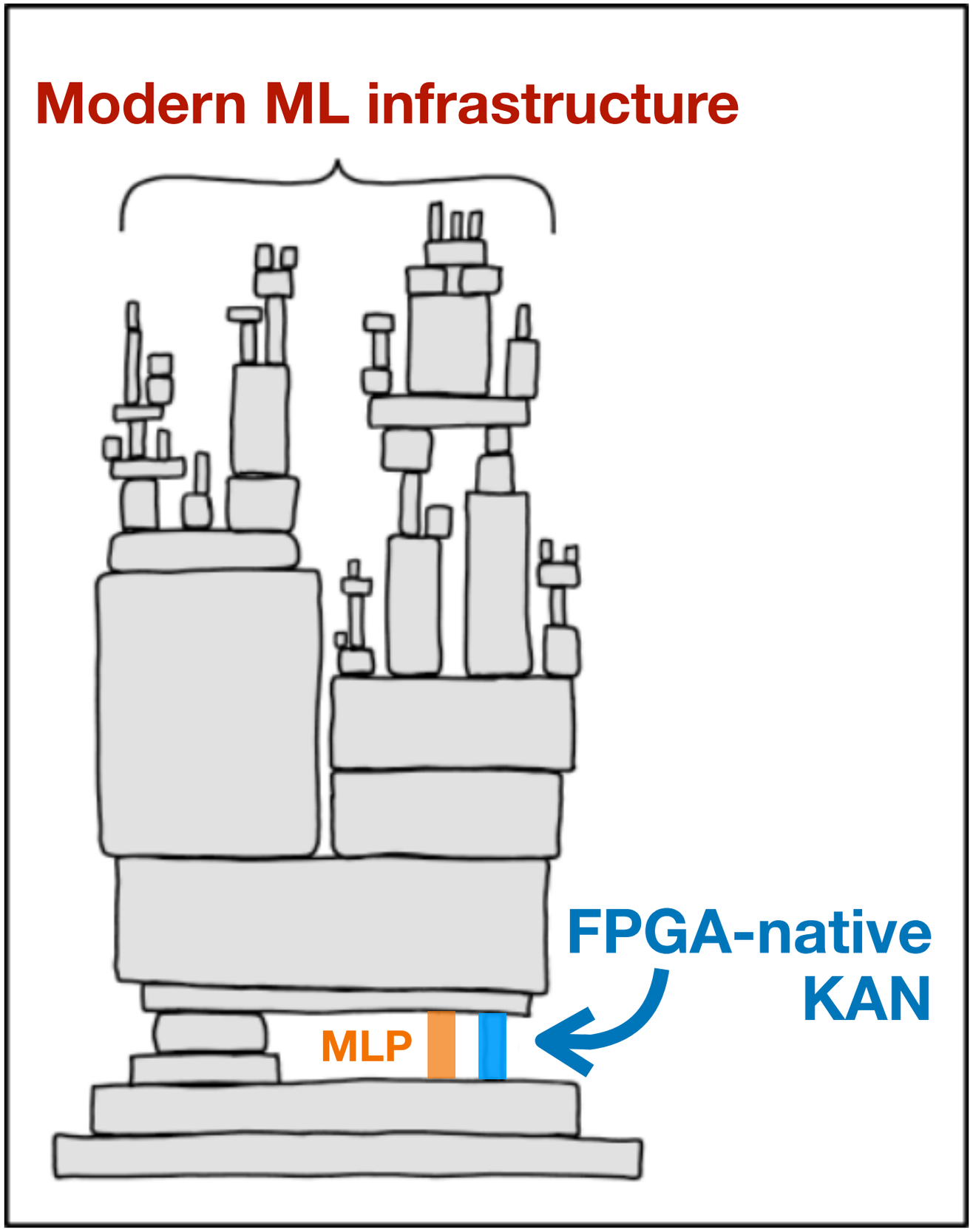
Better rewards under quantization

MLP controller does not fit on the same Zynq board

New paper 👉 *ArXiv:2602.02056*

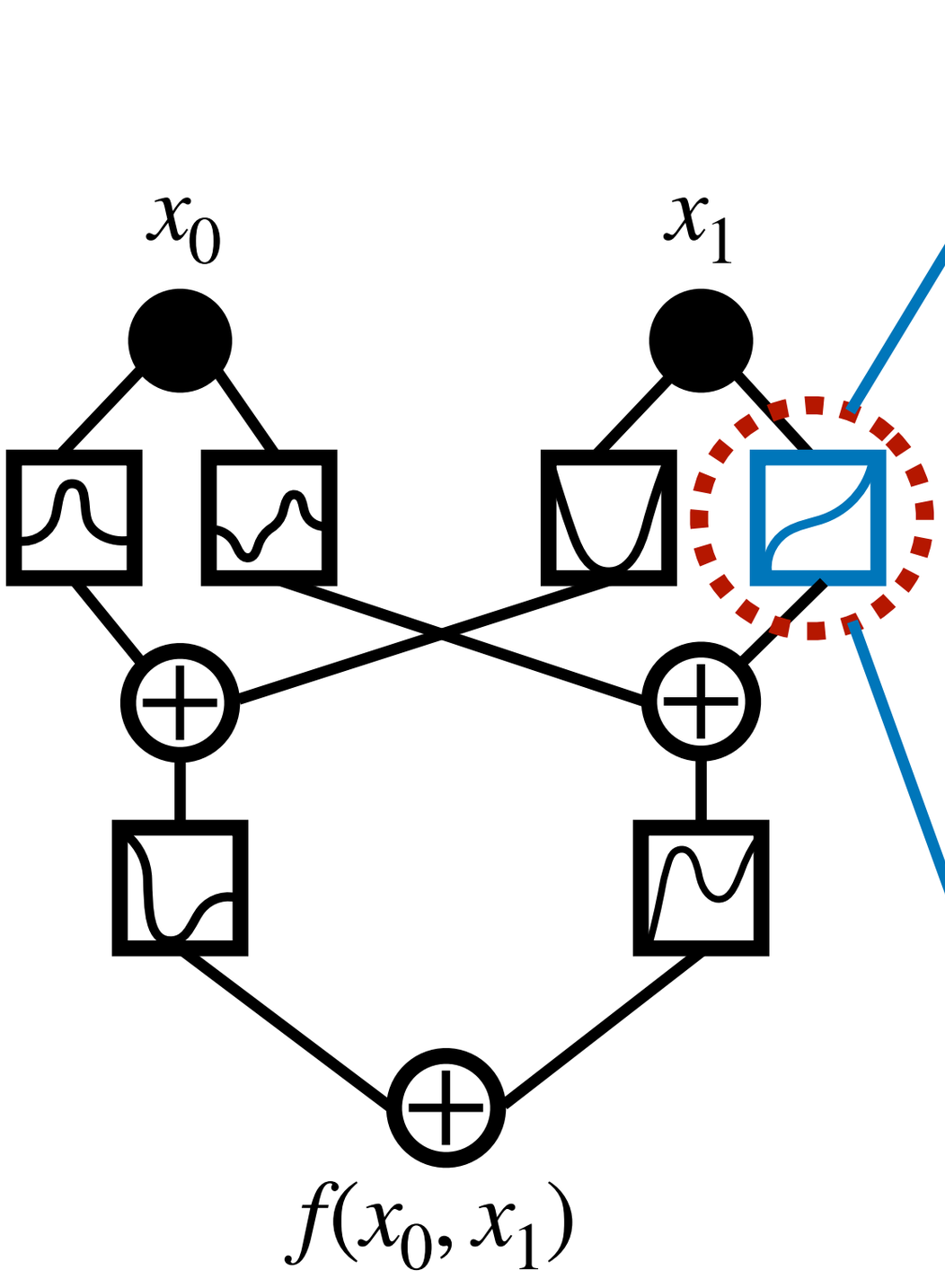
Conclusion

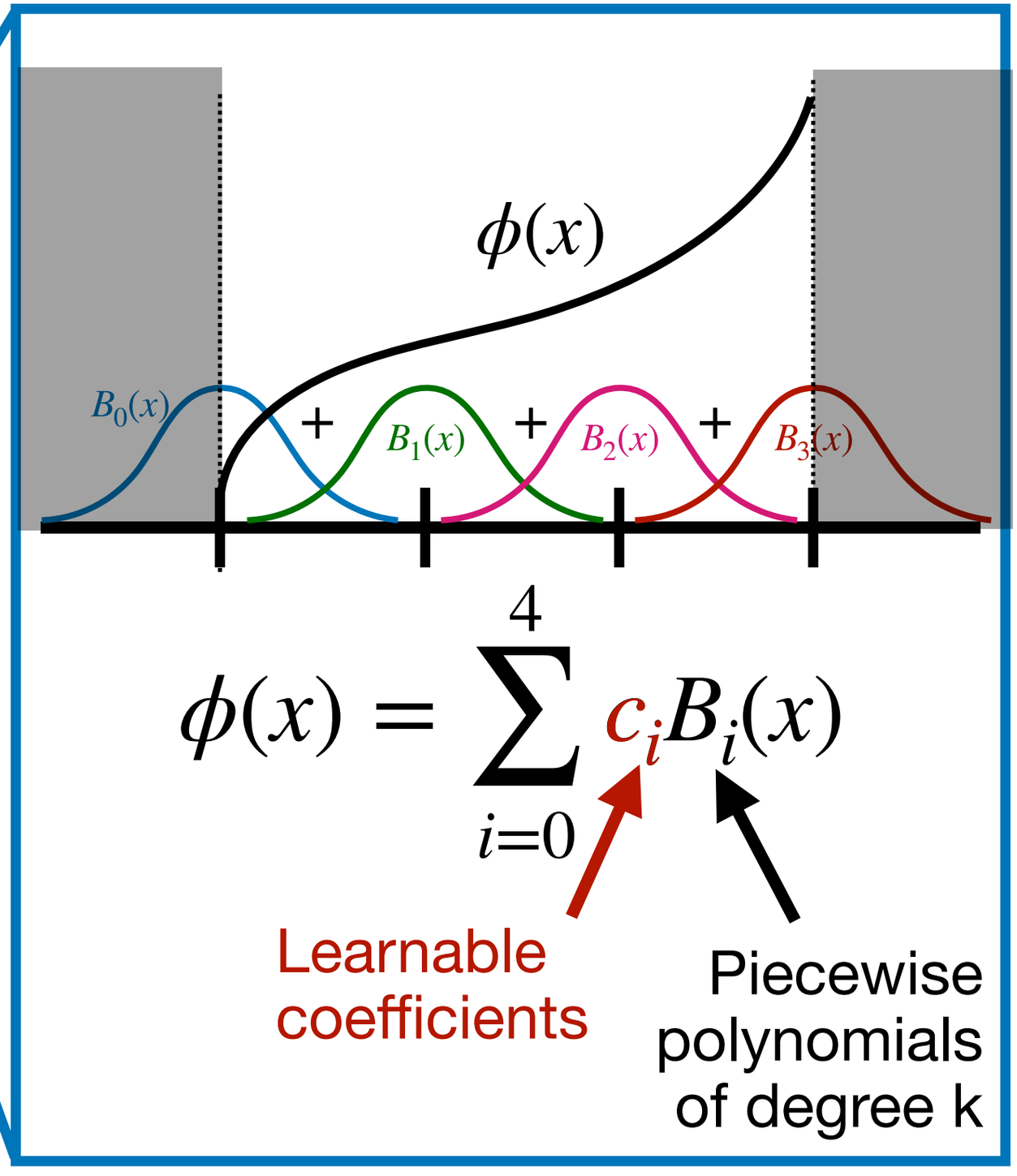
- KAN is a promising alternative to MLP.
- We enable **efficient and practical KAN inference for the first time.**
- Competive with or surpasses many SOTA LUT-based architectures.
- Introduce more complex benchmarks.
- **Unlock future design spaces: KAN-CNN, KAN-formers etc.**



Backup

B(asis)-splines



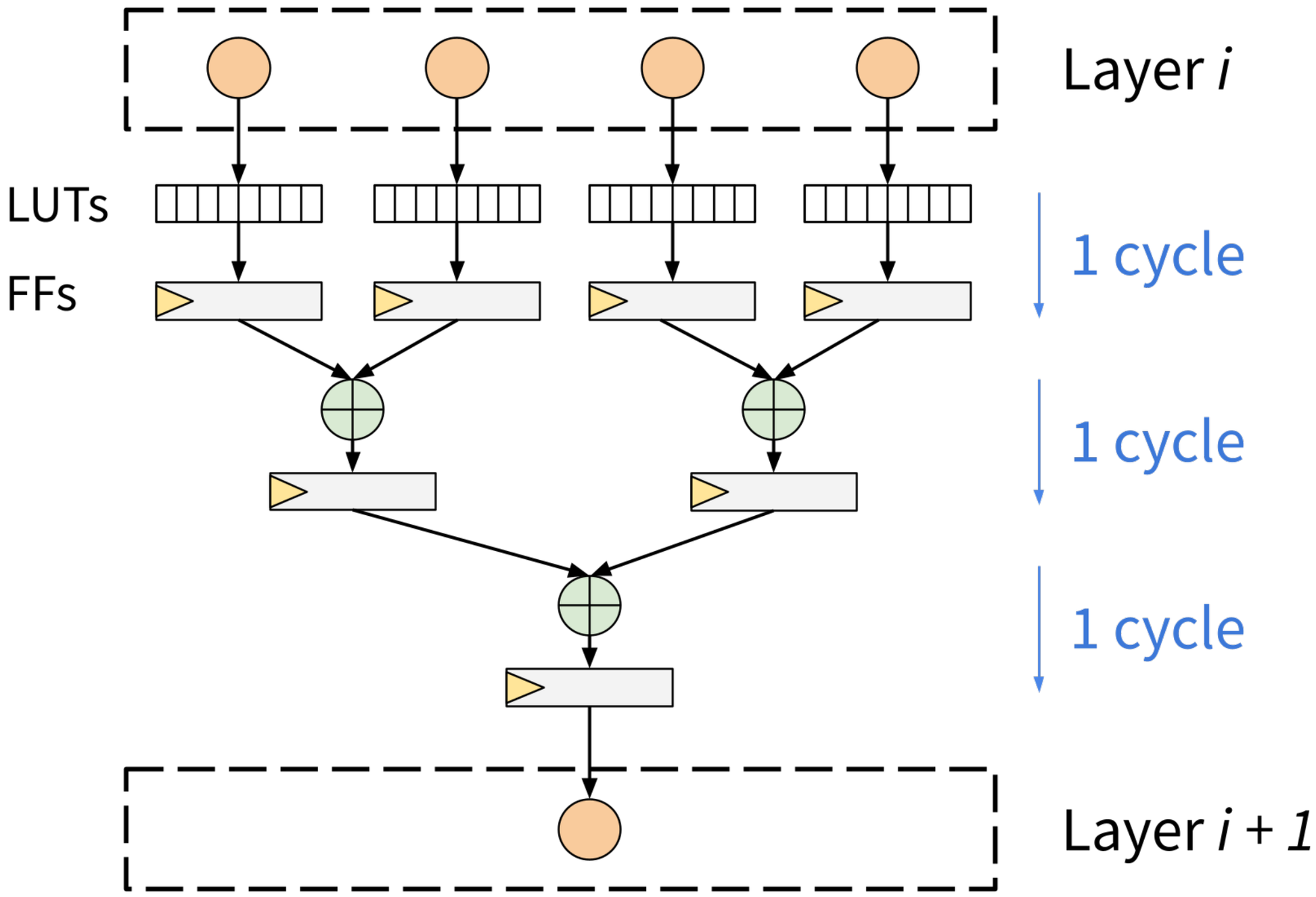


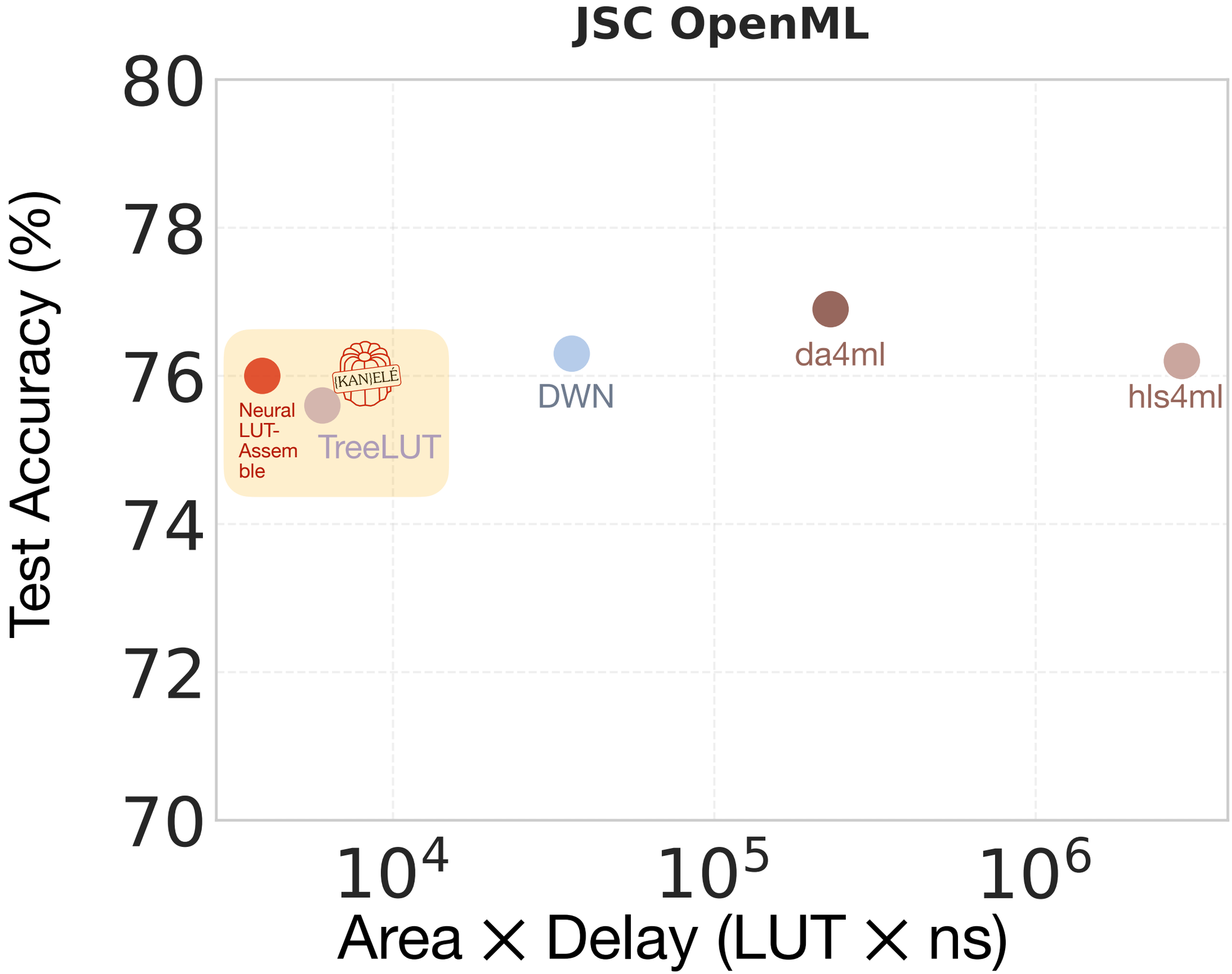
Floating point & QAT + prune training

Table 2: Accuracy comparison of MLP Floating Point and KAN Floating Point and Quantized models on benchmark datasets. All models have the same dimensions listed. The floating point versions only use the layer size (d_l) parameter.

Datasets	G	$[a, b]$	S	d_l	n_l	T	Accuracy (%)/AUC		
							MLP FP	KAN FP	KAN Quantized & Pruned
KAN FPGA Benchmarks [41]									
Moons	6	[-8, 8]	3	[2, 2, 1]	[6, 5, 8]	0.	87.2	**97.7**	97.4
Wine	6	[-8, 8]	3	[13, 4, 3]	[6, 7, 8]	0.	96.3	98.1	**98.2**
Dry Bean	6	[-8, 8]	3	[16, 2, 7]	[6, 6, 8]	0.	90.9	**92.2**	92.1
LUT-based neural network benchmarks									
MNIST	30	[-8,8]	3	[784, 62, 10]	[1, 6, 6]	1.	96.7	**97.9**	96.3
JSC CERNBox	30	[-2, 2]	10	[16, 12, 5]	[8, 8, 6]	0.14	73.0	**75.1**	**75.1**
JSC OpenML	40	[-2, 2]	10	[16, 8, 5]	[6, 7, 6]	0.9	76.5	**76.5**	76.0
MLPerf Tiny Benchmark									
ToyADMOS	30	[-2, 2]	10	[64, 16, 8, 16, 64]	[7, 8, 8, 7, 8]	0.9	0.80	**0.83**	**0.83**

Adder tree




JSC OpenML
80
78
76
74
72
70
Test Accuracy (%)
Neural LUT-Assemble
TreeLUT
DWN
da4ml
hls4ml
10^4
10^5
10^6
Area × Delay (LUT × ns)