Towards Statistical Guarantees in Controlling Quality Tradeoffs for Approximate Acceleration

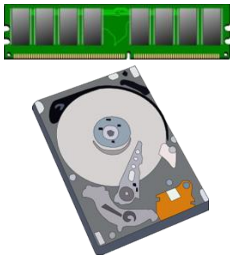
Divya Mahajan
Amir Yazdanbakhsh
Jongse Park
Bradley Thwaites
Hadi Esmaeilzadeh

Alternative Computing Technologies (ACT) Lab
Georgia Institute of Technology

Approximate Computing

Relax the abstraction of ***"near perfect"*** **accuracy in**

Data Processing



Storage

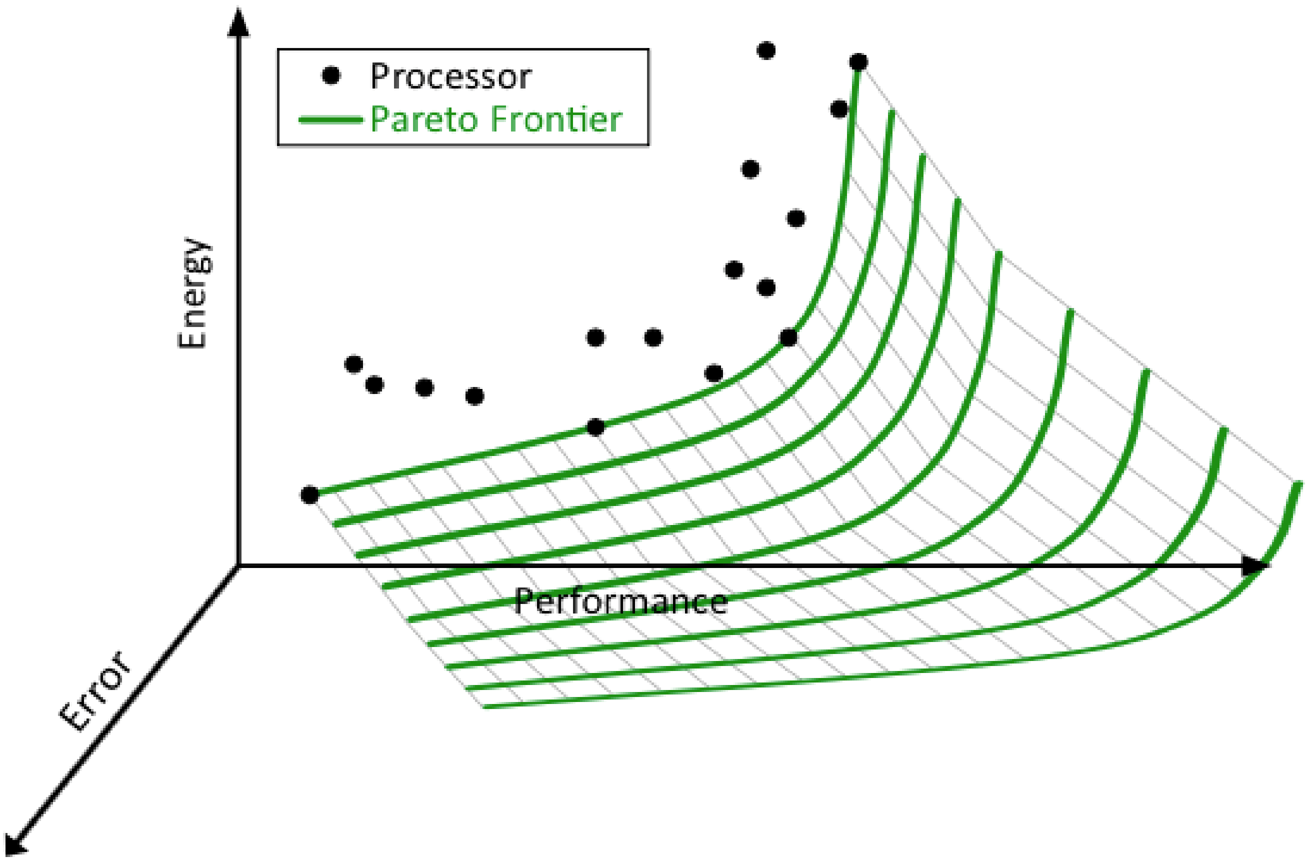
Communication

Accept **imprecision** to improve

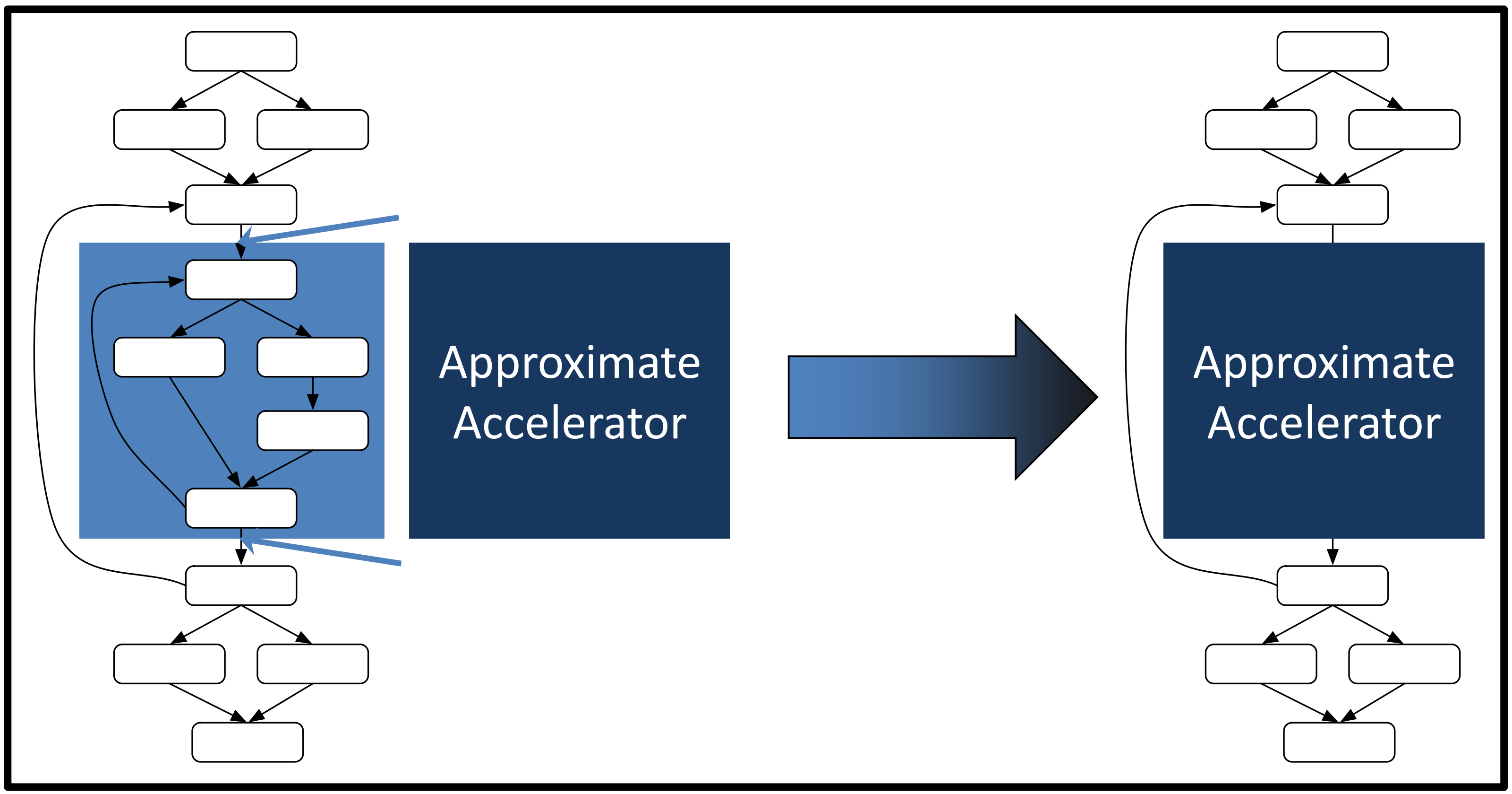
performance

energy efficiency

Approximate Acceleration



Approximate Acceleration



Application

Fixed Error

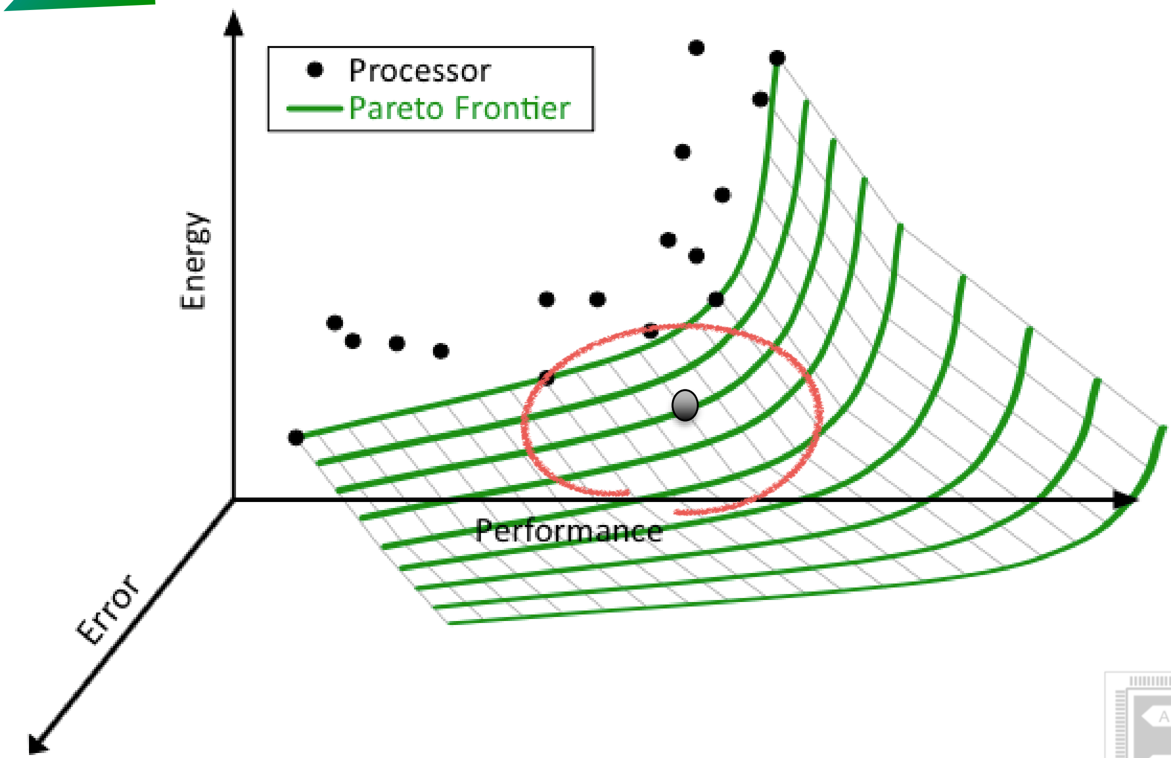
Lack of Error Control

Lack of Guarantees

Fixed Error

Lack of Error Control

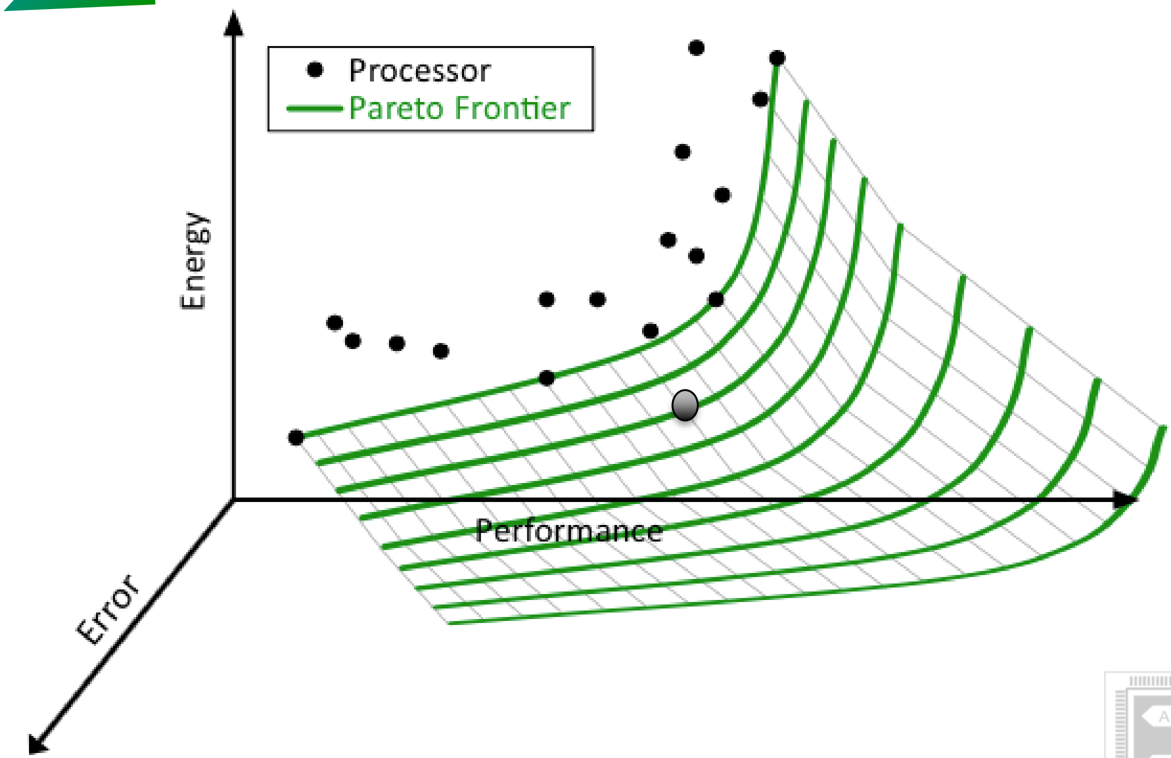
Lack of Guarantees



Fixed Error

Lack of Error Control

Lack of Guarantees

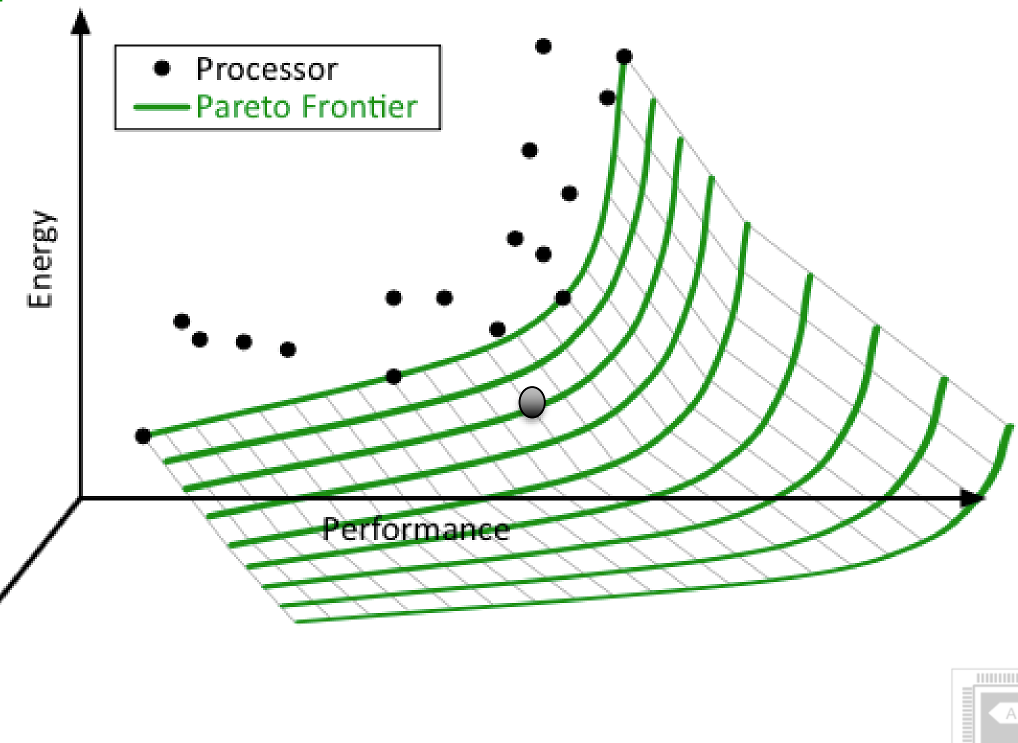


Fixed Error

Lack of Error Control

Lack of Guarantees

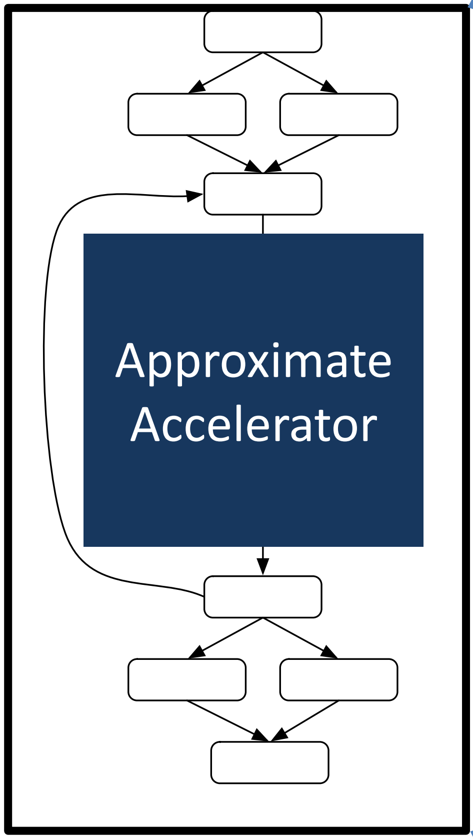
To tackle these shortcomings we devise **MITHRA**



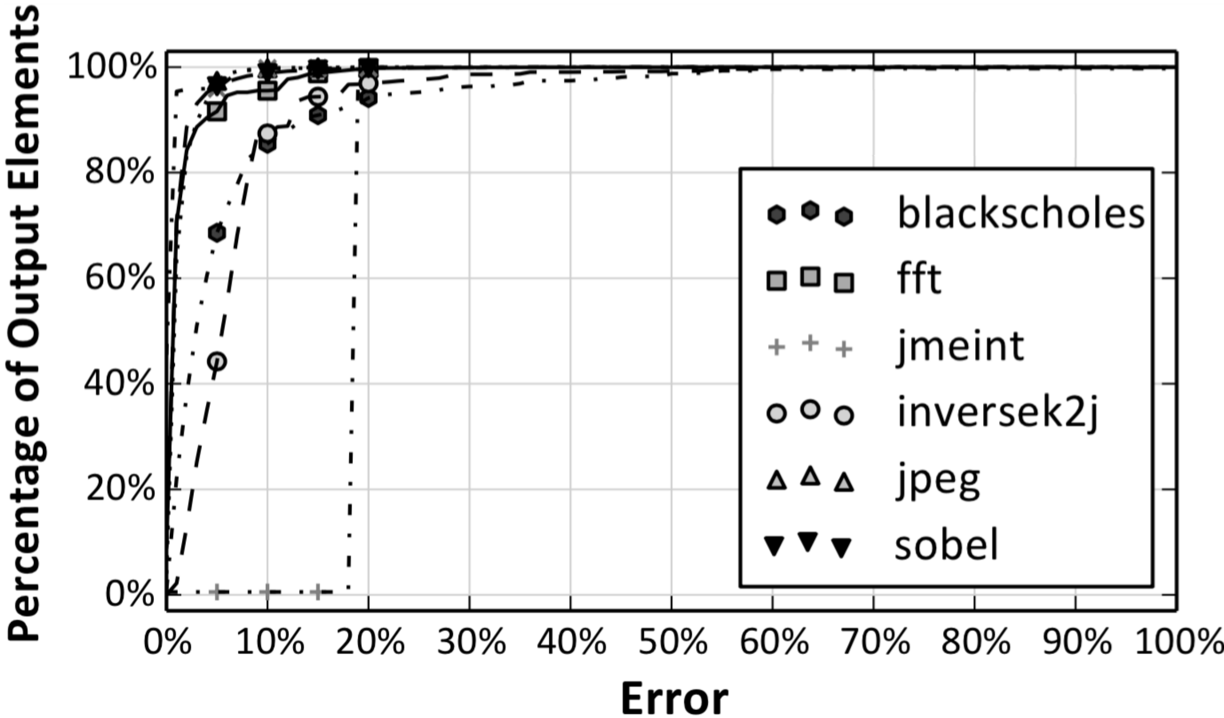
Overview

- Motivation
- Challenges in devising MITHRA
- A hardware software solution
- Detailing the components of MITHRA

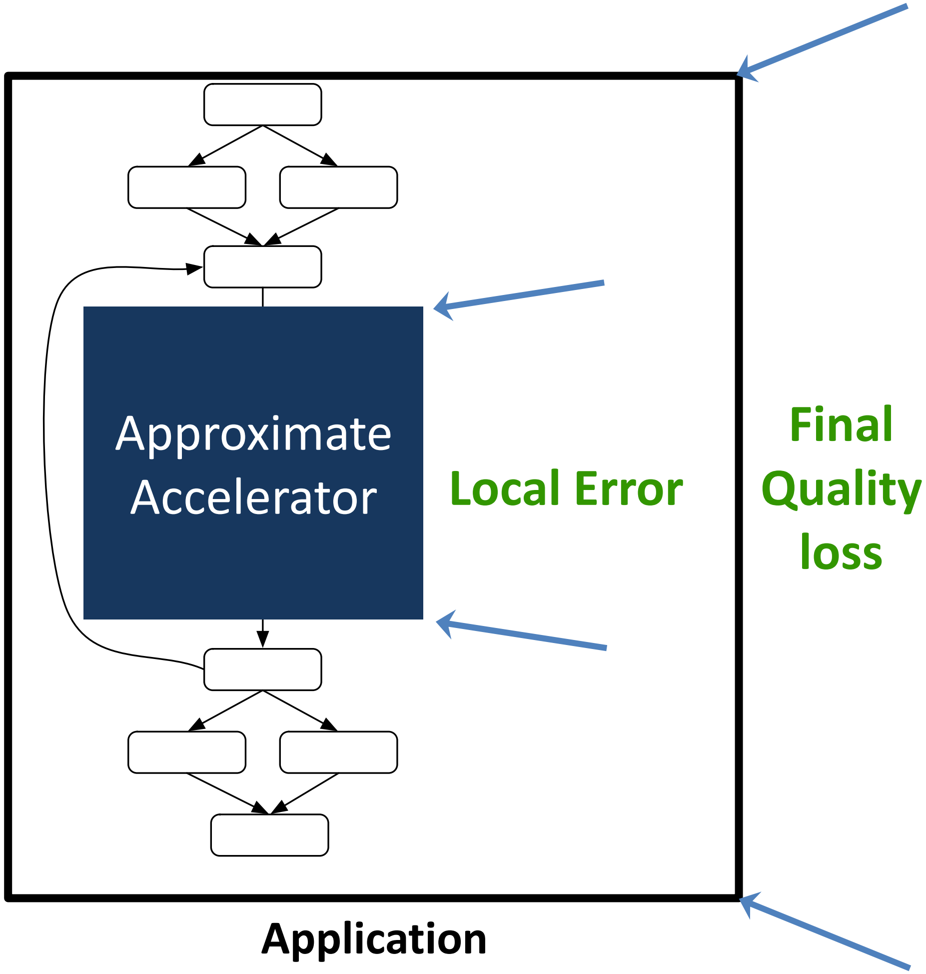
Exploiting Accelerator Characteristics



Application



Challenges



How to eliminate anomalous invocations?

Factors Influencing Error

Accelerator Error = $| \text{Output}_{accelerator} - \text{Output}_{original} |$

$\text{Output}_{accelerator} = f\,(\text{accelerator inputs, accelerator configuration})$

Factors Influencing Error

Accelerator Error = $|\ \text{Output}_{\text{accelerator}} - \text{Output}_{\text{original}}\ |$

$\text{Output}_{\text{accelerator}} = f\,(\text{accelerator inputs, accelerator configuration})$

Constant

Factors Influencing Error

Accelerator Inputs → **Accelerator Configuration** → $\text{Output}_{\text{accelerator}}$

$$\text{Accelerator Error} = |\,\text{Output}_{\text{accelerator}} - \text{Output}_{\text{original}}\,|$$

$$\text{Output}_{\text{accelerator}} = f\,(\text{accelerator inputs, accelerator configuration})$$

Constant

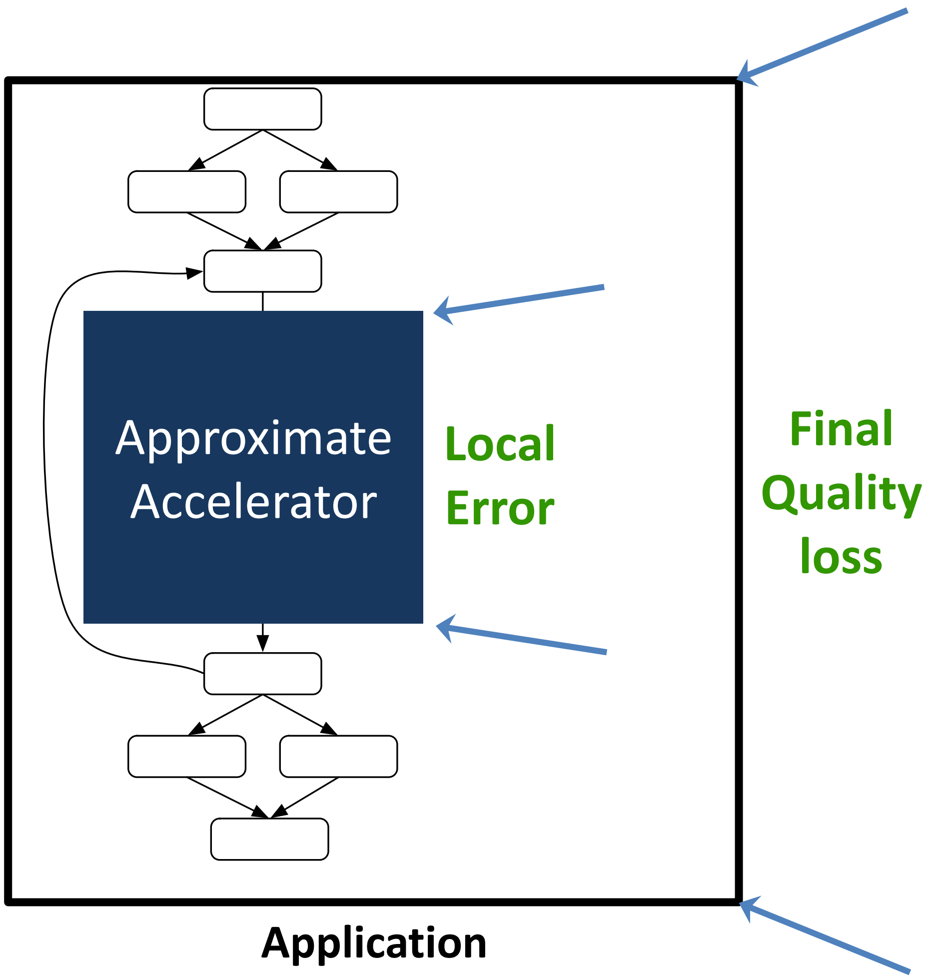
$$\text{Output}_{\text{accelerator}} = f\,(\text{accelerator inputs})$$

Challenges

How to eliminate anomalous invocations?

Look at the accelerator inputs

Challenges



Final Quality loss
→ Local Error ?

Challenges

How to eliminate anomalous invocations?

Look at the accelerator inputs

Final Quality loss → Local Error ?

Threshold the local error

Challenges

- How to eliminate anomalous invocations?
  - Look at the accelerator inputs
- Final Quality loss → Local Error ?
  - Threshold the local error
- What guarantees?

Challenges

- How to eliminate anomalous invocations?
- Final Quality loss → Local Error ?
- What guarantees?

- Look at the accelerator inputs
- Threshold the local error
- Statistical Guarantees

Challenges

How to eliminate anomalous invocations?

Look at the accelerator inputs

Final Quality loss → Local Error ?

Threshold the local error

What guarantees?

Statistical Guarantees

What algorithm at runtime?

Challenges

How to eliminate anomalous invocations?

Look at the accelerator inputs

Final Quality loss → Local Error ?

Threshold the local error

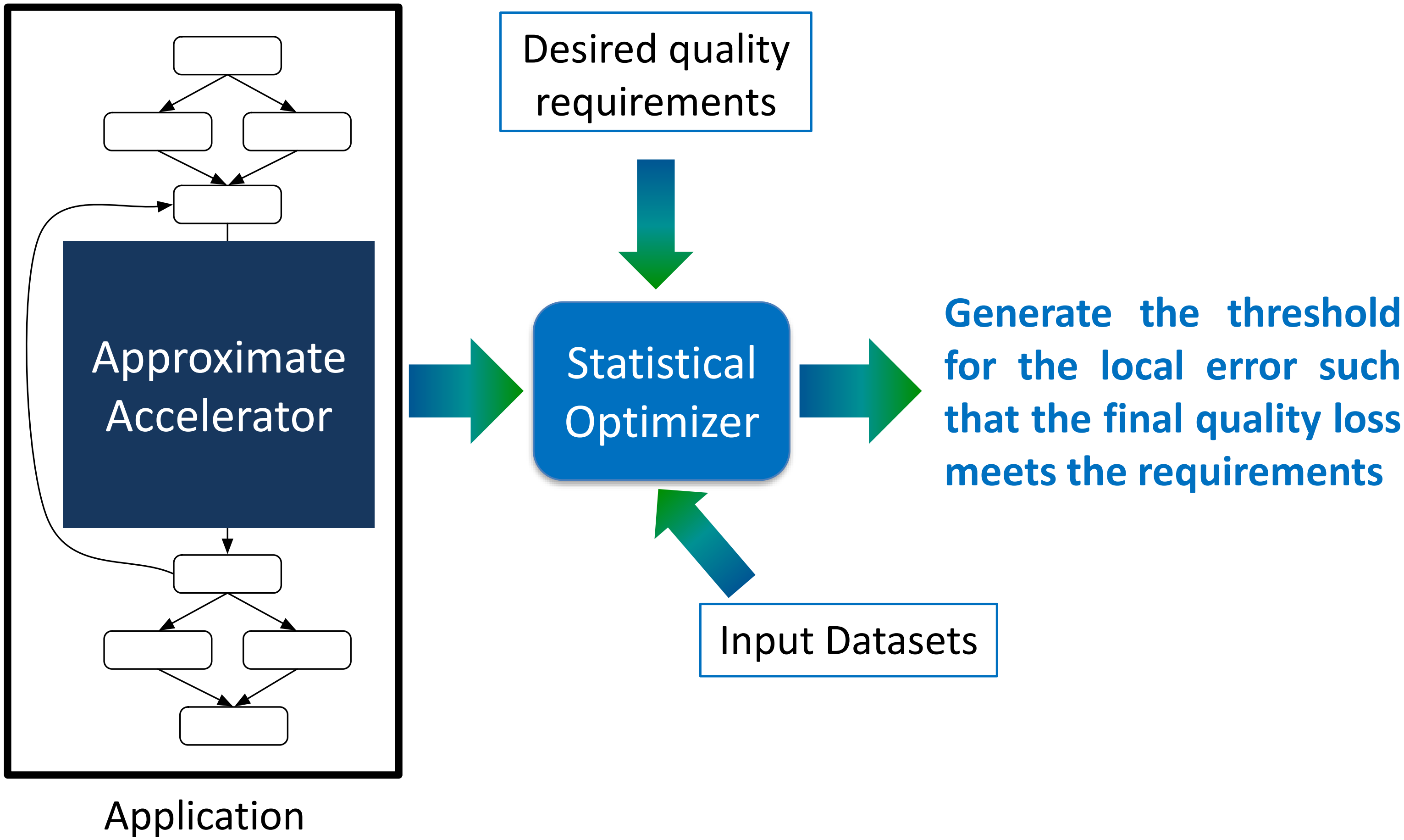
What guarantees?

Statistical Guarantees

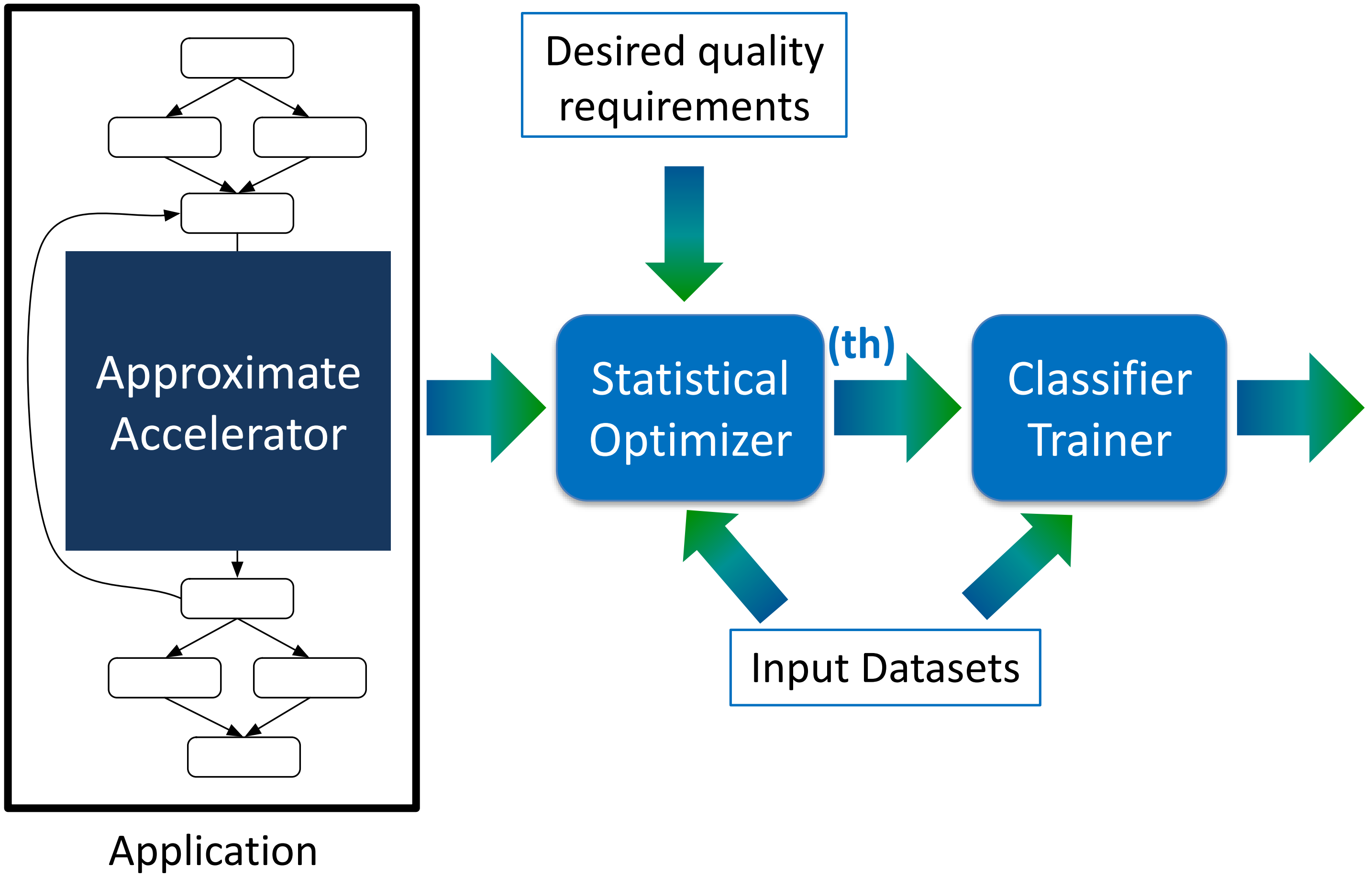
What algorithm at runtime?

Classification in Hardware

MITHRA: A Hardware/Software Solution

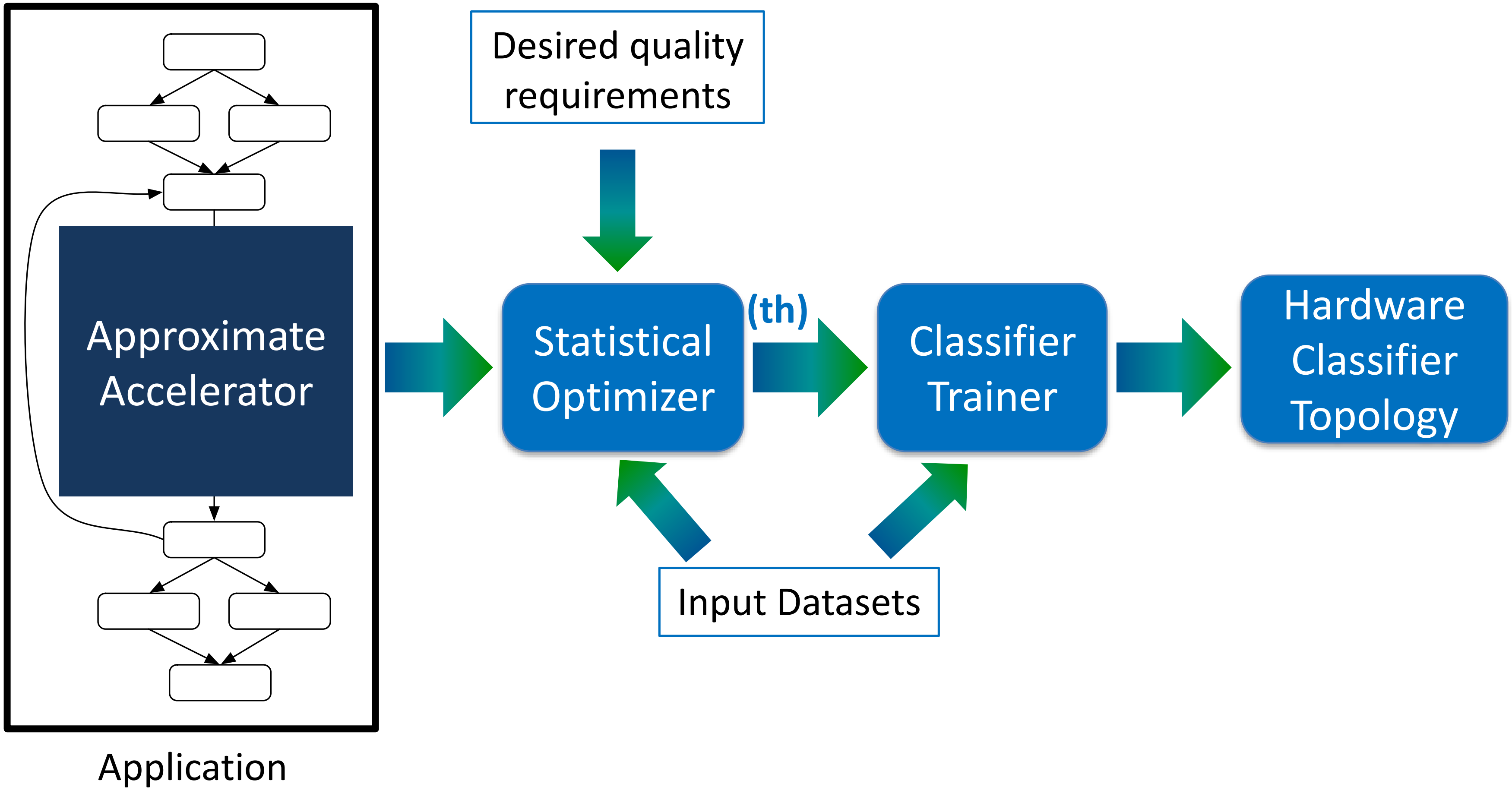


MITHRA: A Hardware/Software Solution

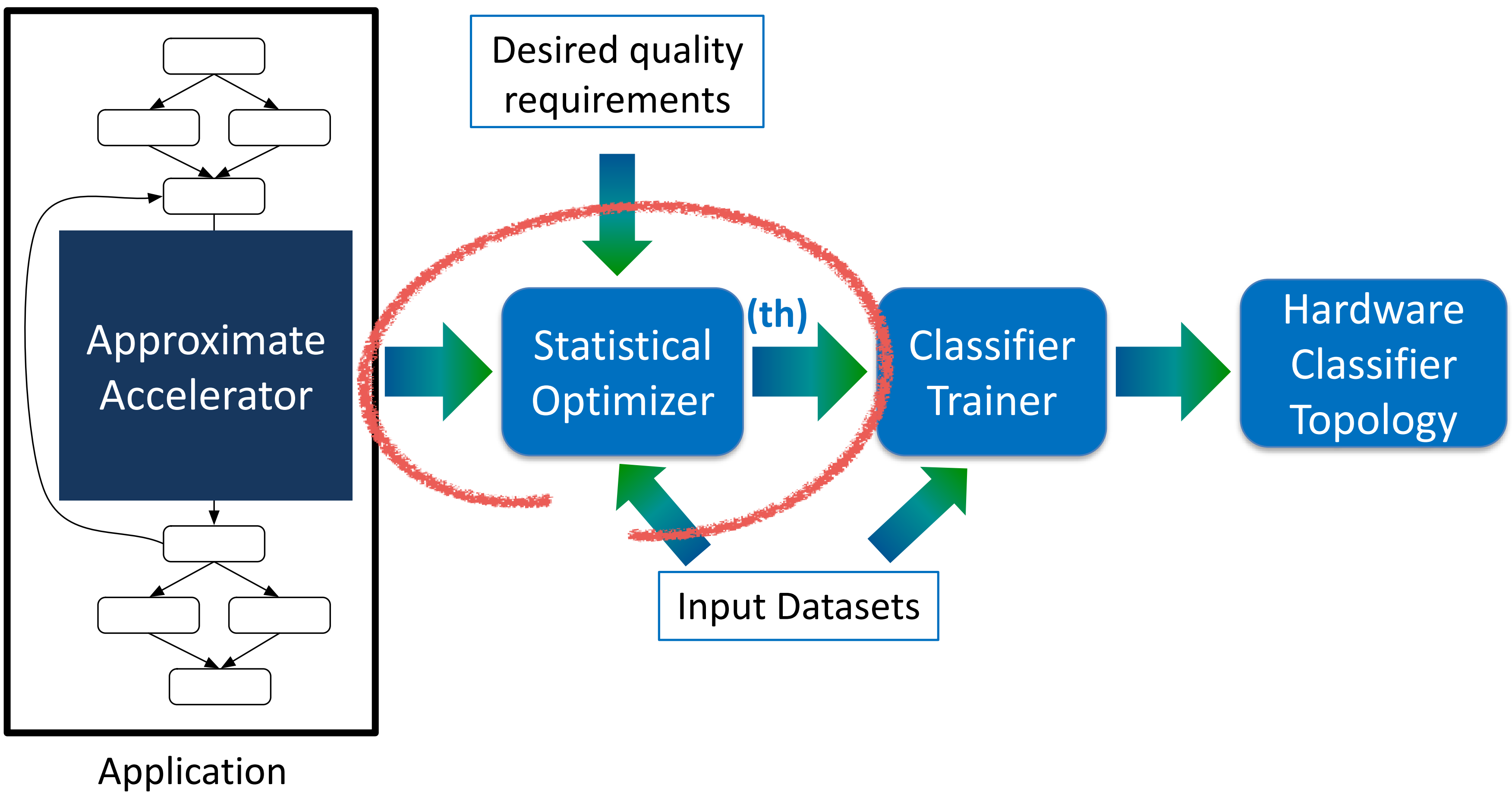


Generate training data that segregates inputs that give > and < (th)

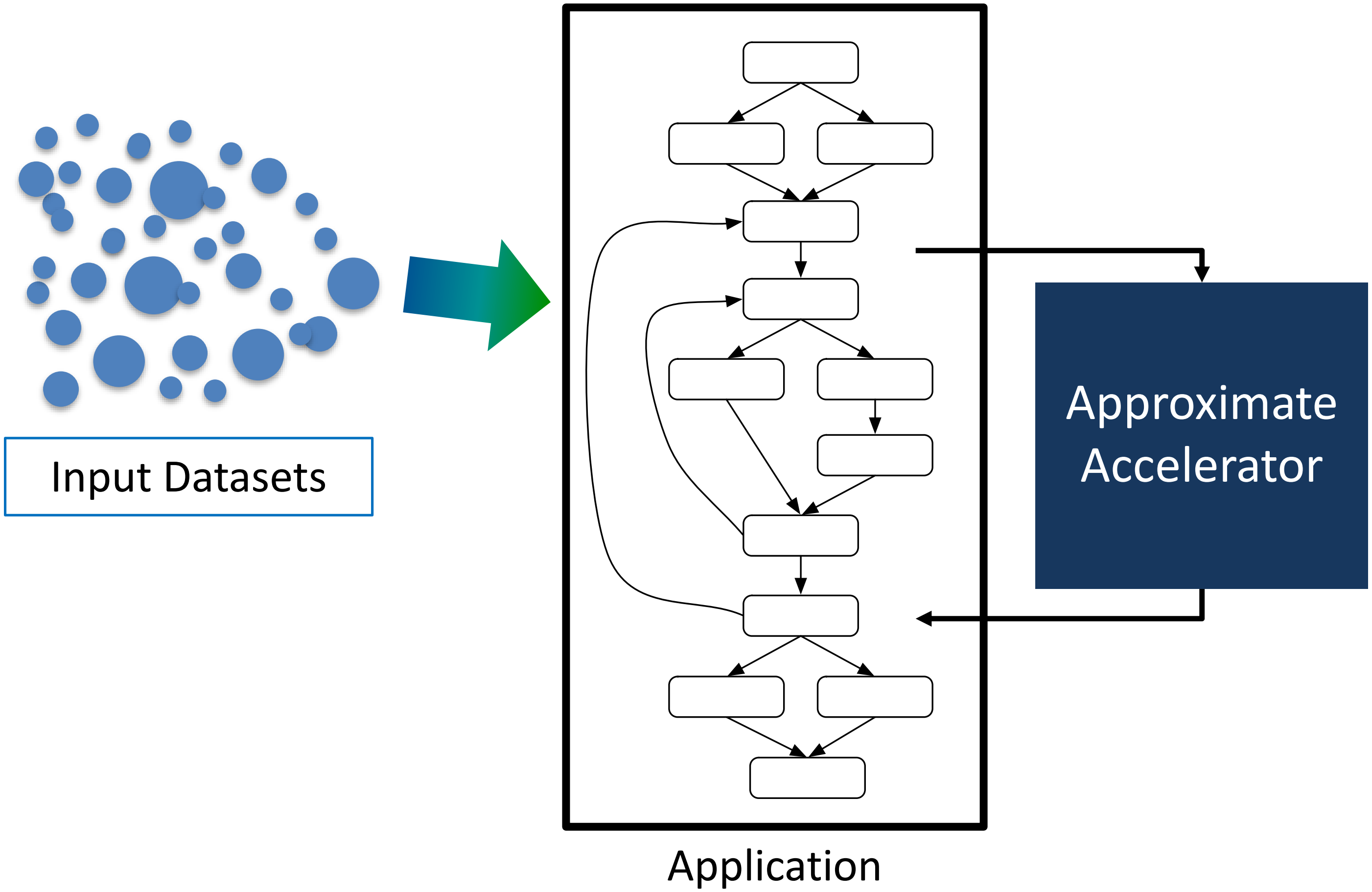
MITHRA: A Hardware/Software Solution



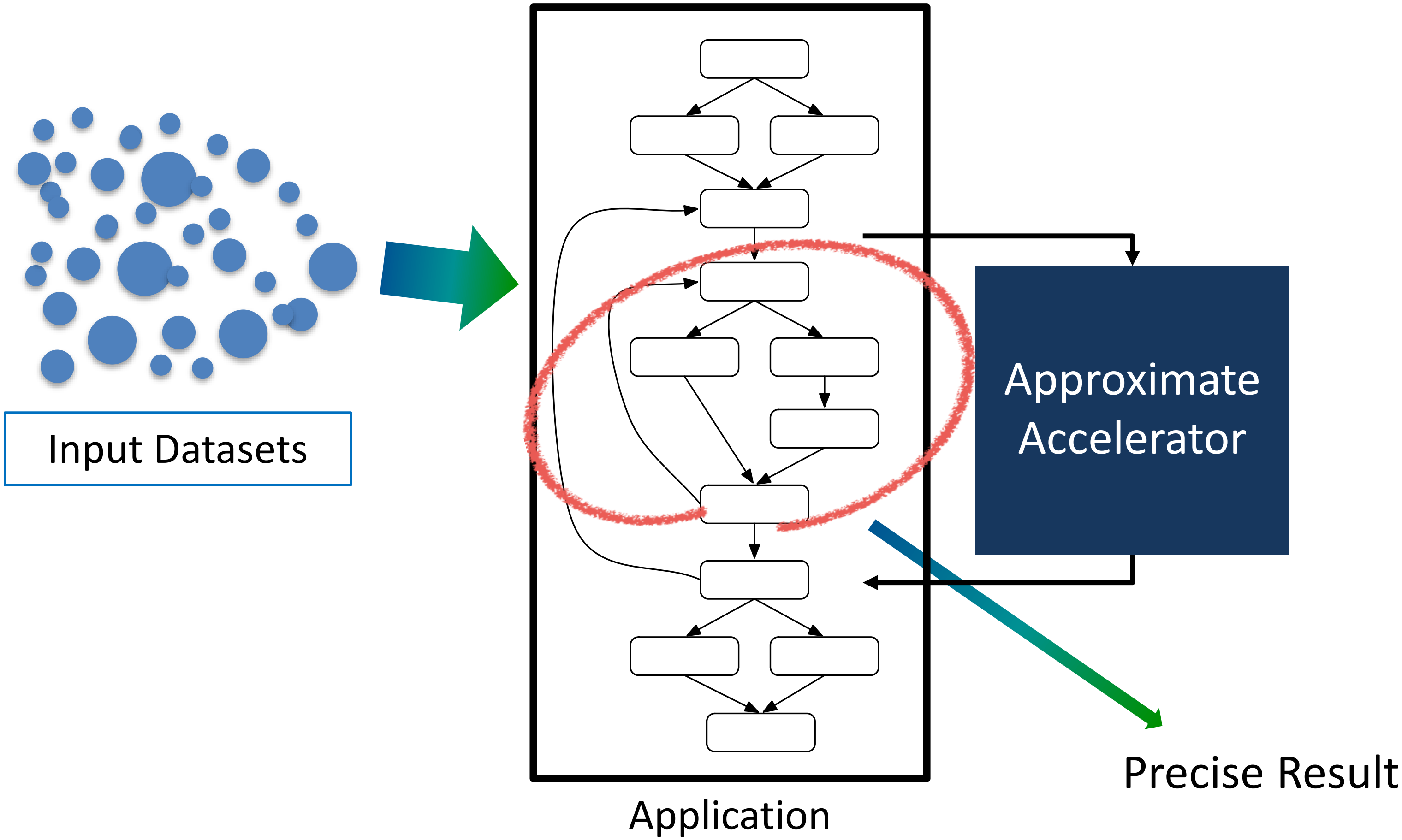
MITHRA: A Hardware/Software Solution



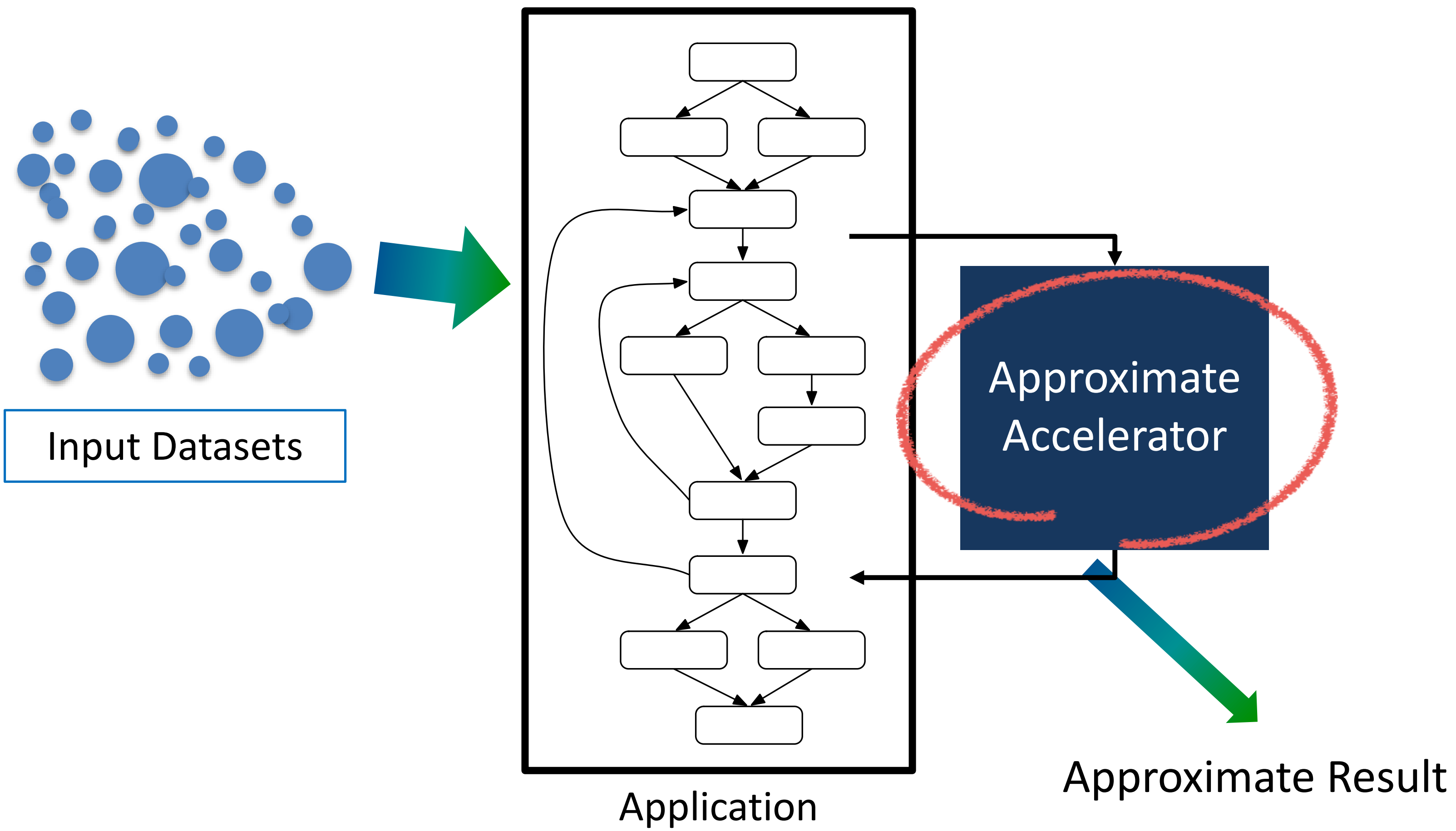
Statistical Optimizer



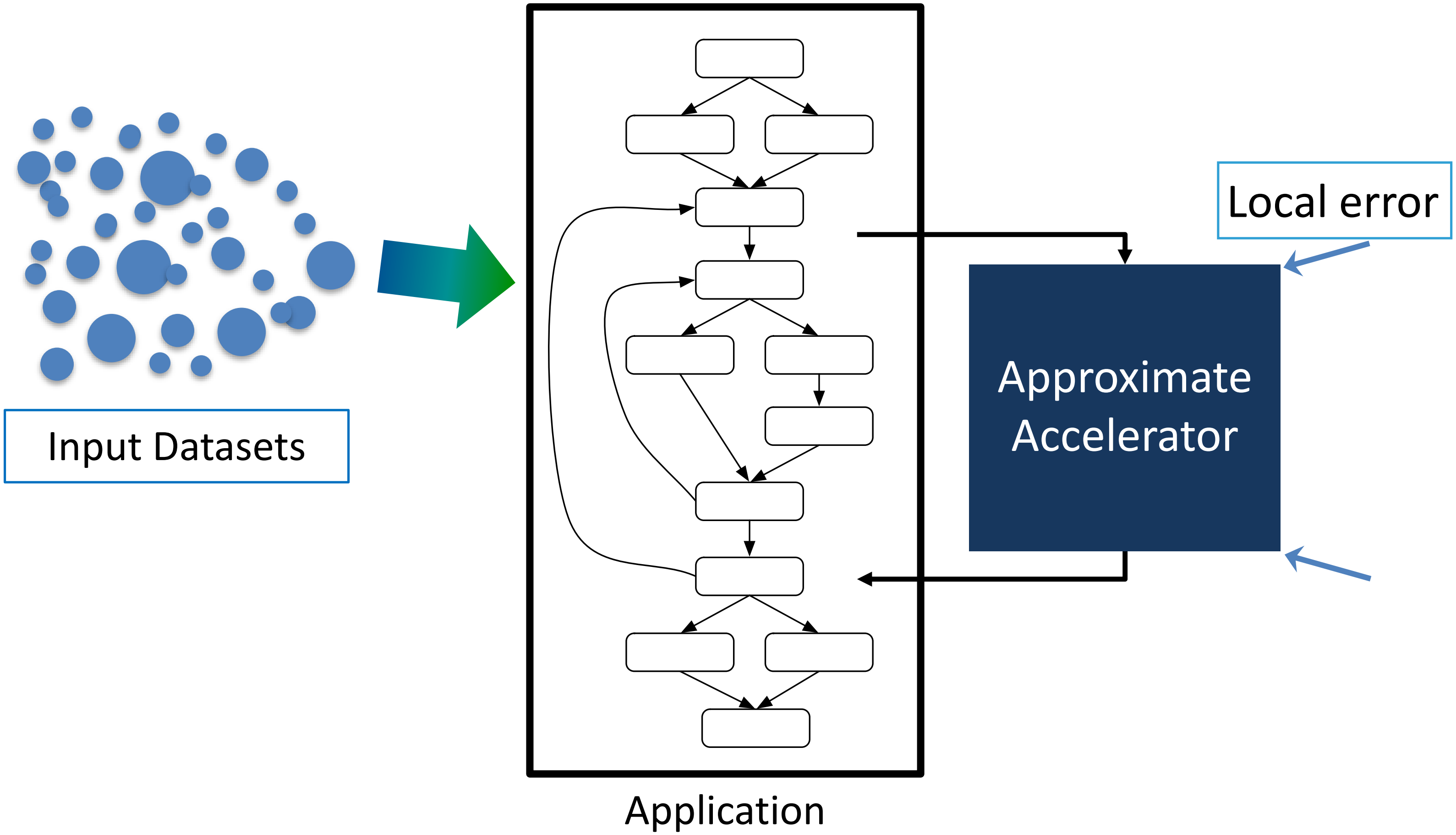
Statistical Optimizer



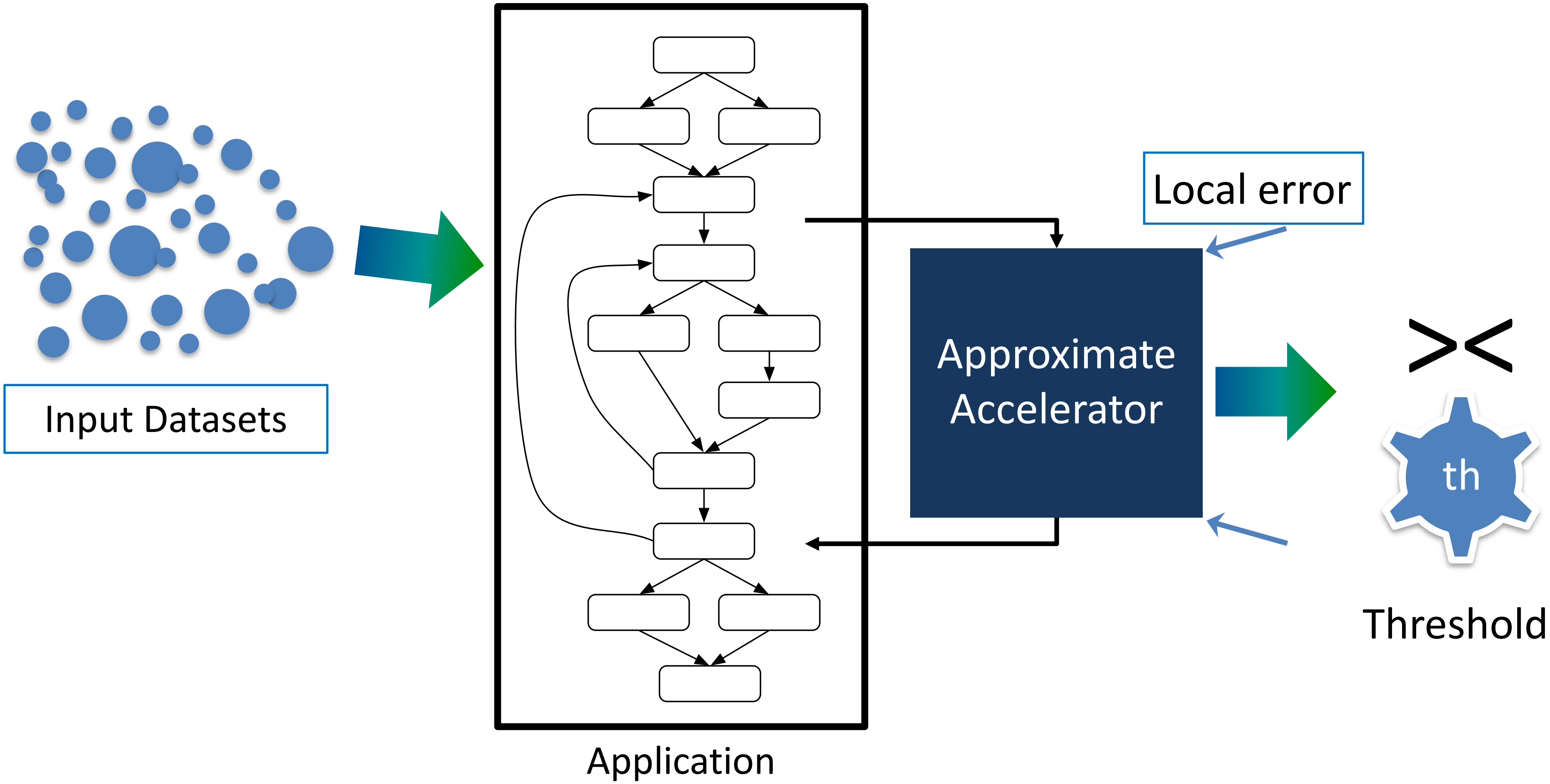
Statistical Optimizer

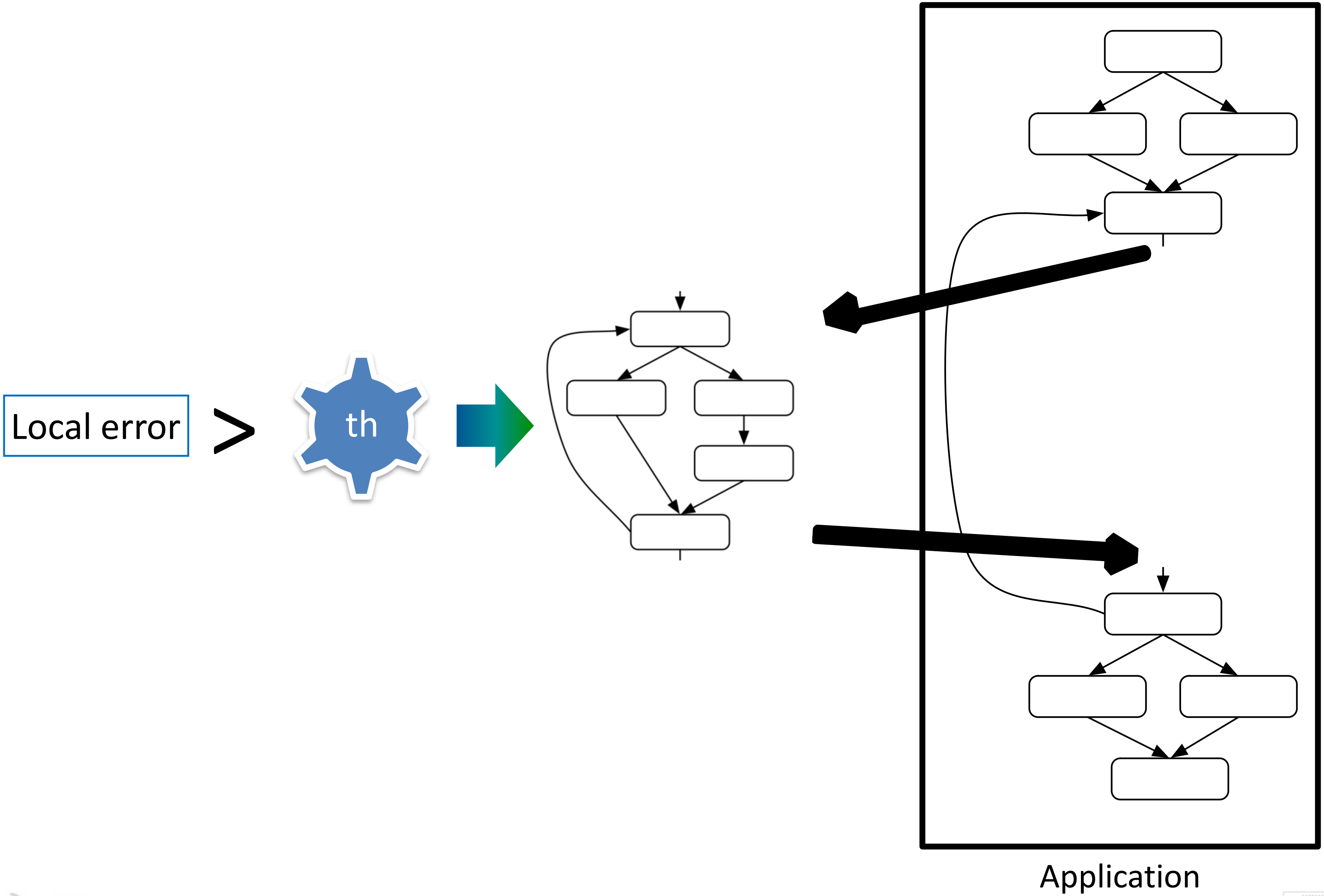


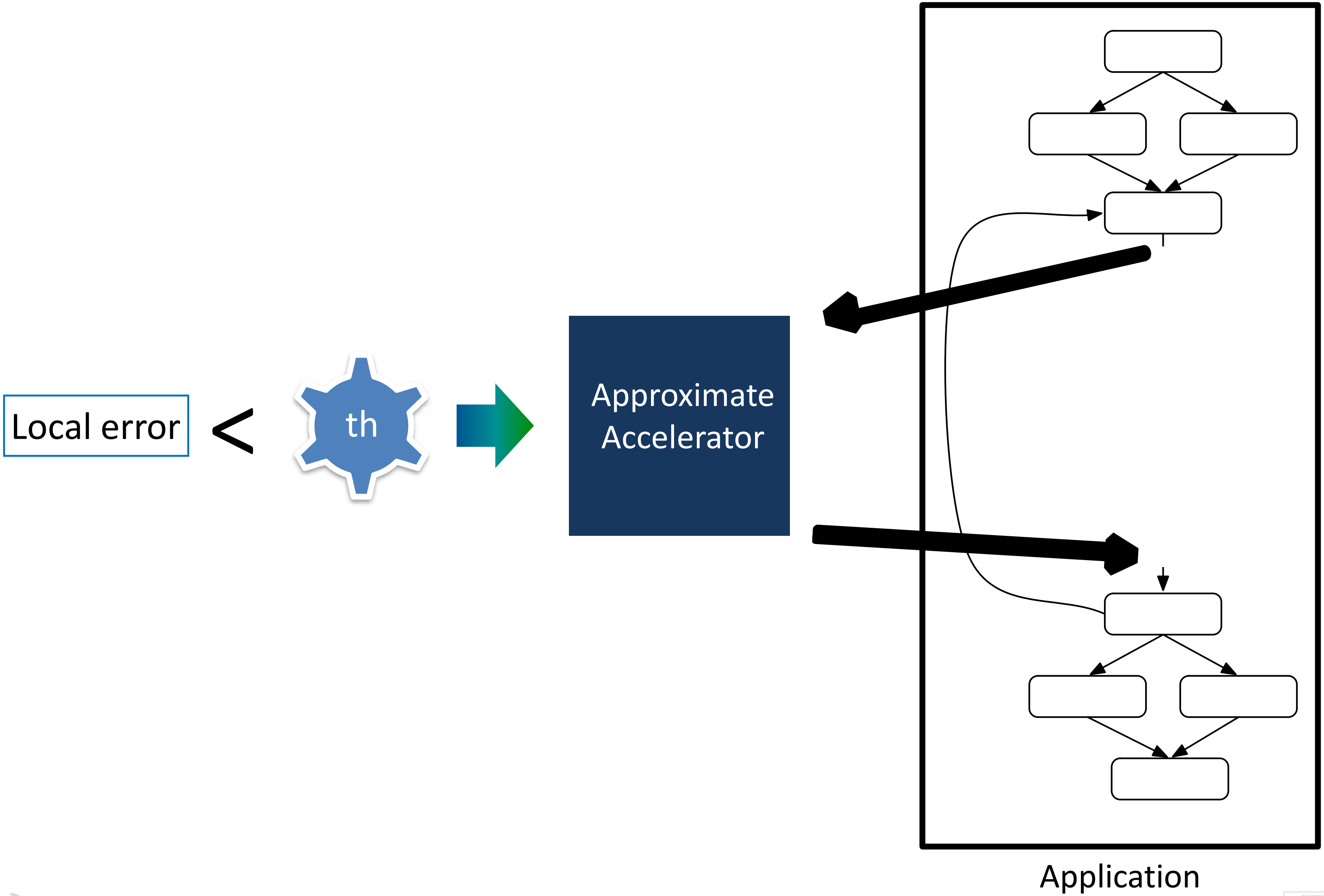
Statistical Optimizer



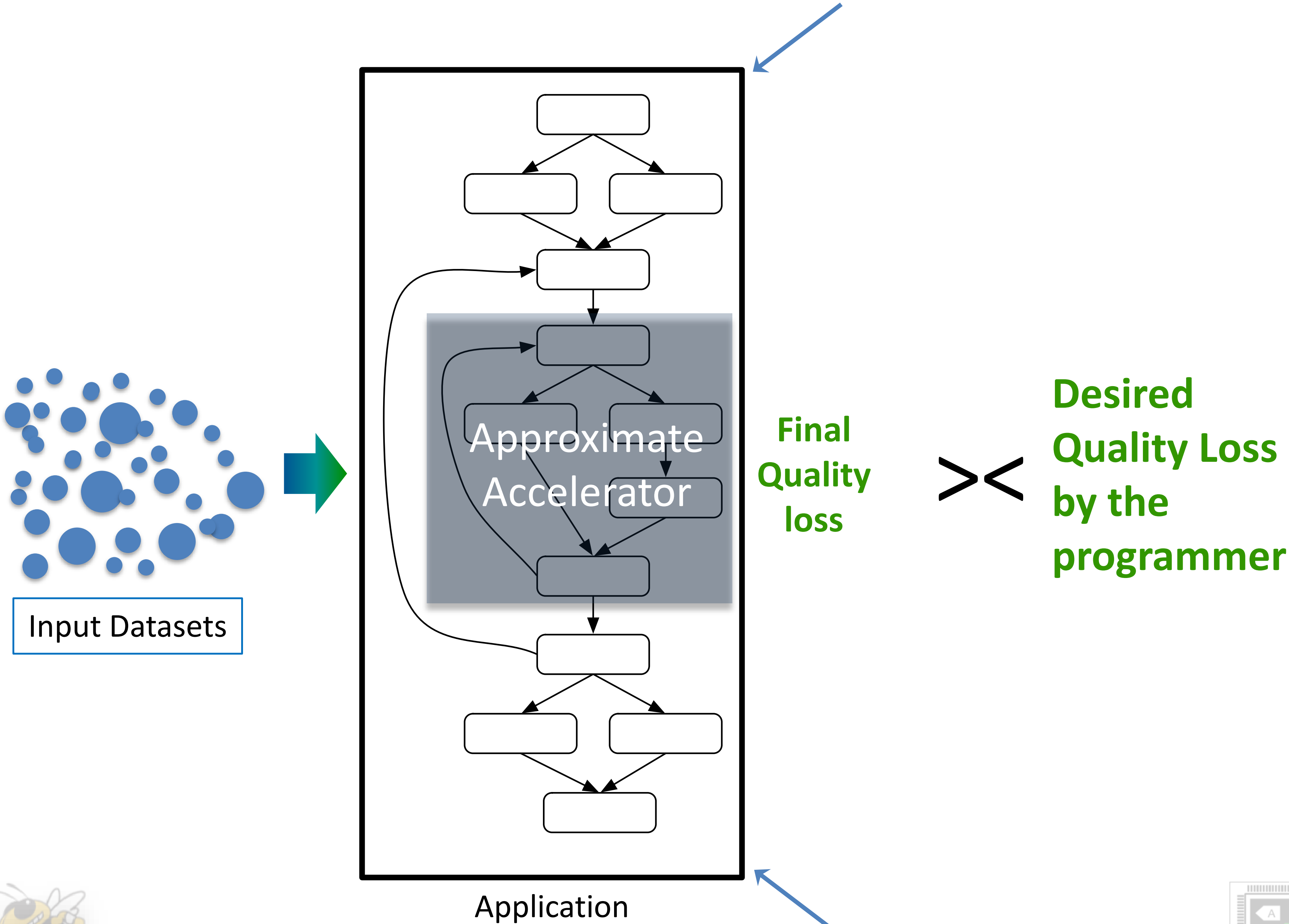
Statistical Optimizer



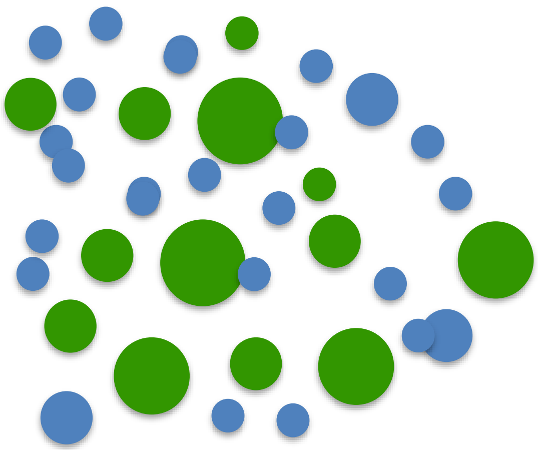

Local error
>
th
Application



Local error
<
th
Approximate Accelerator
Application


Input Datasets
Approximate
Accelerator
Application
Final
Quality
loss
><
Desired
Quality Loss
by the
programmer

Binomial Proportion Confidence Interval

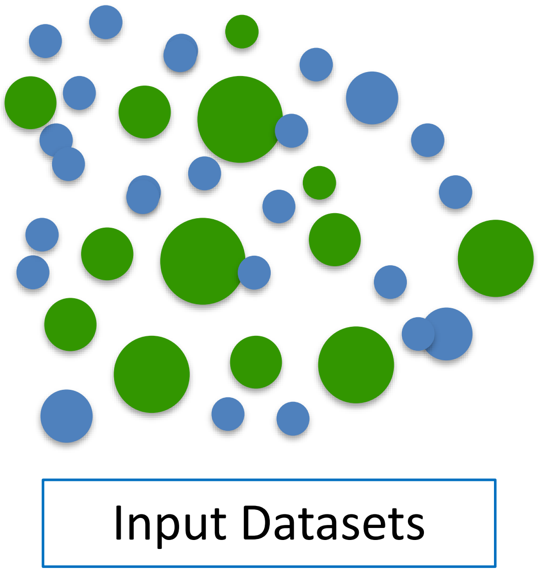


Input Datasets

- Less than the desired programmer quality loss
  → $n_{success}$

- Greater than the desired programmer quality loss

Binomial Proportion Confidence Interval



- Less than the desired programmer quality loss
  ➔ $n_{success}$
- Greater than the desired programmer quality loss

$$\frac{1}{1+\frac{(n_{trials}-n_{success}+1)}{n_{success}\times F[1-\alpha;2n_{success},2(n_{trials}-n_{success}+1)]}} < SuccessRate$$

with a confidence level

Example

$(n_{trials}, n_{success})$ ➡ $r < SuccessRate$
$with\ a\ confidence\ level$

E.g., (100, 80) ➡ 72% < SuccessRate

with 95% confidence level

Example

$(n_{trials}, n_{success})$ ➡ $r < SuccessRate$

$with\ a\ confidence\ level$

E.g., (100, 80) ➡ 72% < SuccessRate

with 95% confidence level

Final Quality Level, Success Rate and **Confidence Interval** programmer specified

Statistical Optimization

if desired metrics are not met:

$$th_{t+1} = th_t - \Delta$$

else if desired metrics are met:

$$th_{t+1} = th_t + \Delta$$

Reiterate; till the th_t meets the metrics but th_{t+1} doesn't

Statistical Optimization

if desired metrics are not met:

$th_{t+1} = th_t - \Delta$

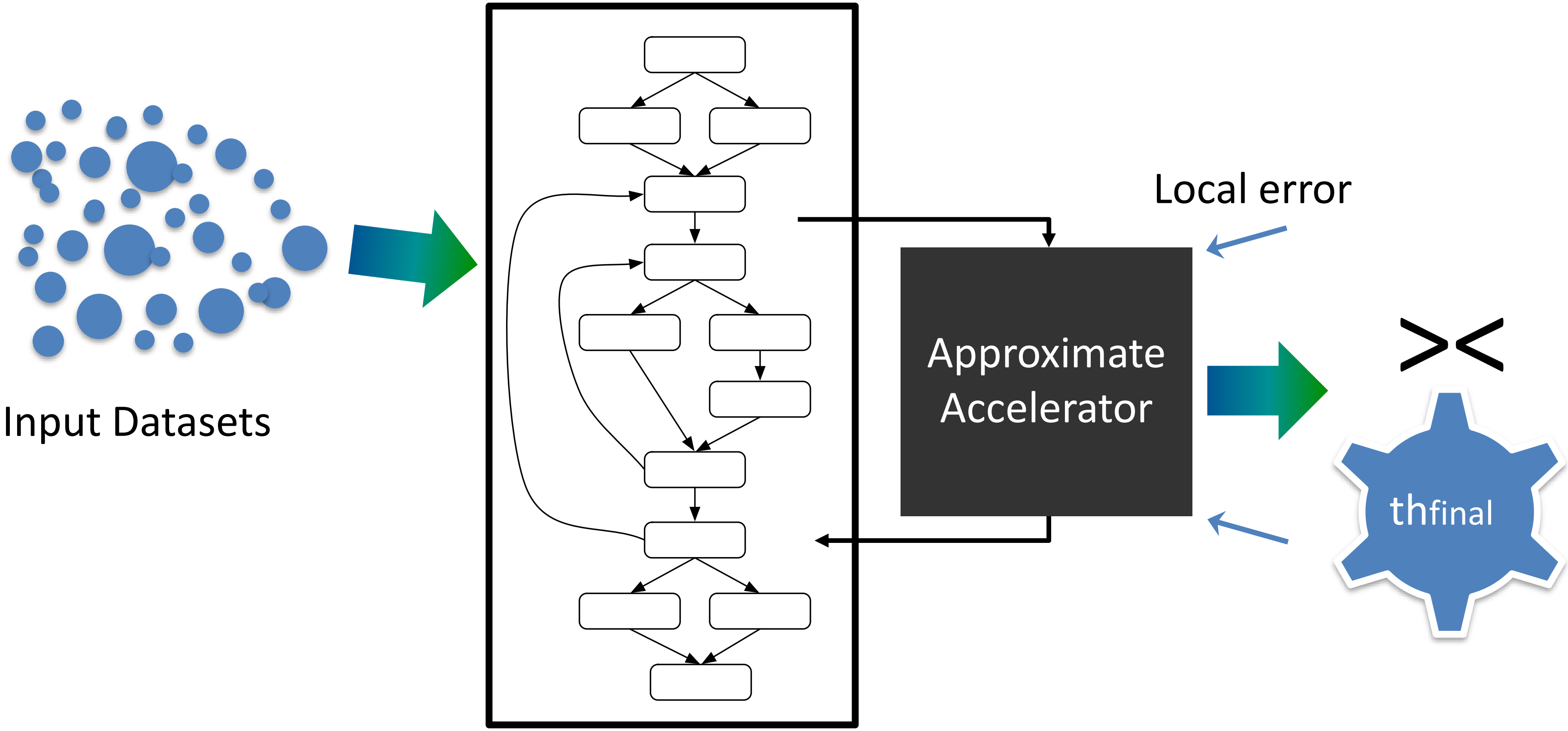
else if desired metrics are met:

$th_{t+1} = th_t + \Delta$

Reiterate; till the th_t meets the metrics but th_{t+1} doesn't

Tighter threshold better **Final Quality Loss Level, Success Rate** and **Confidence Interval** but lower benefits from approximation

Training the Classifiers



The training data used to generate classifier topology

Hardware Classifiers

Simple algorithm that can be easily implemented in hardware.

We use two techniques for this work:
1. Table Based
2. Neural Network Based

Table-based Classifiers

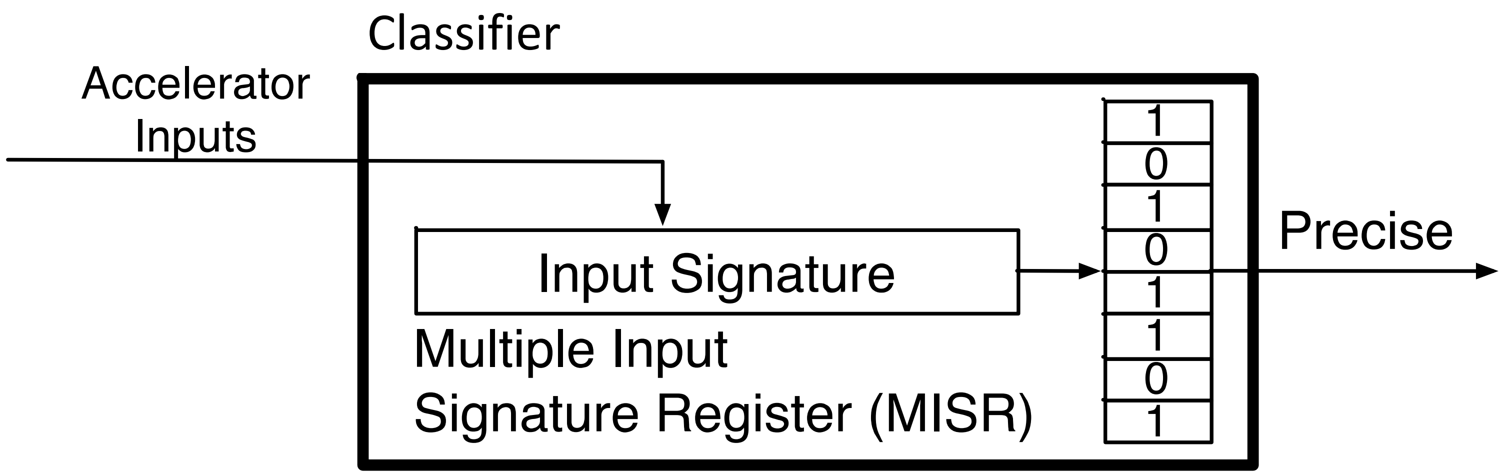
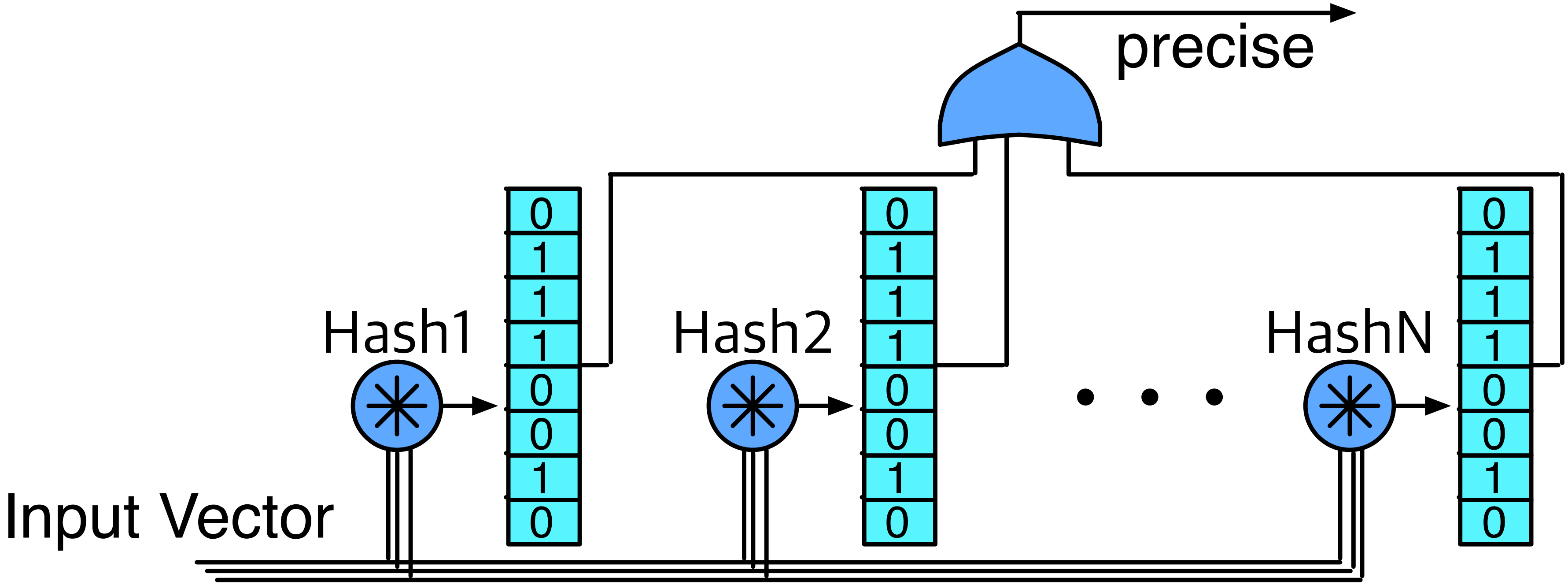
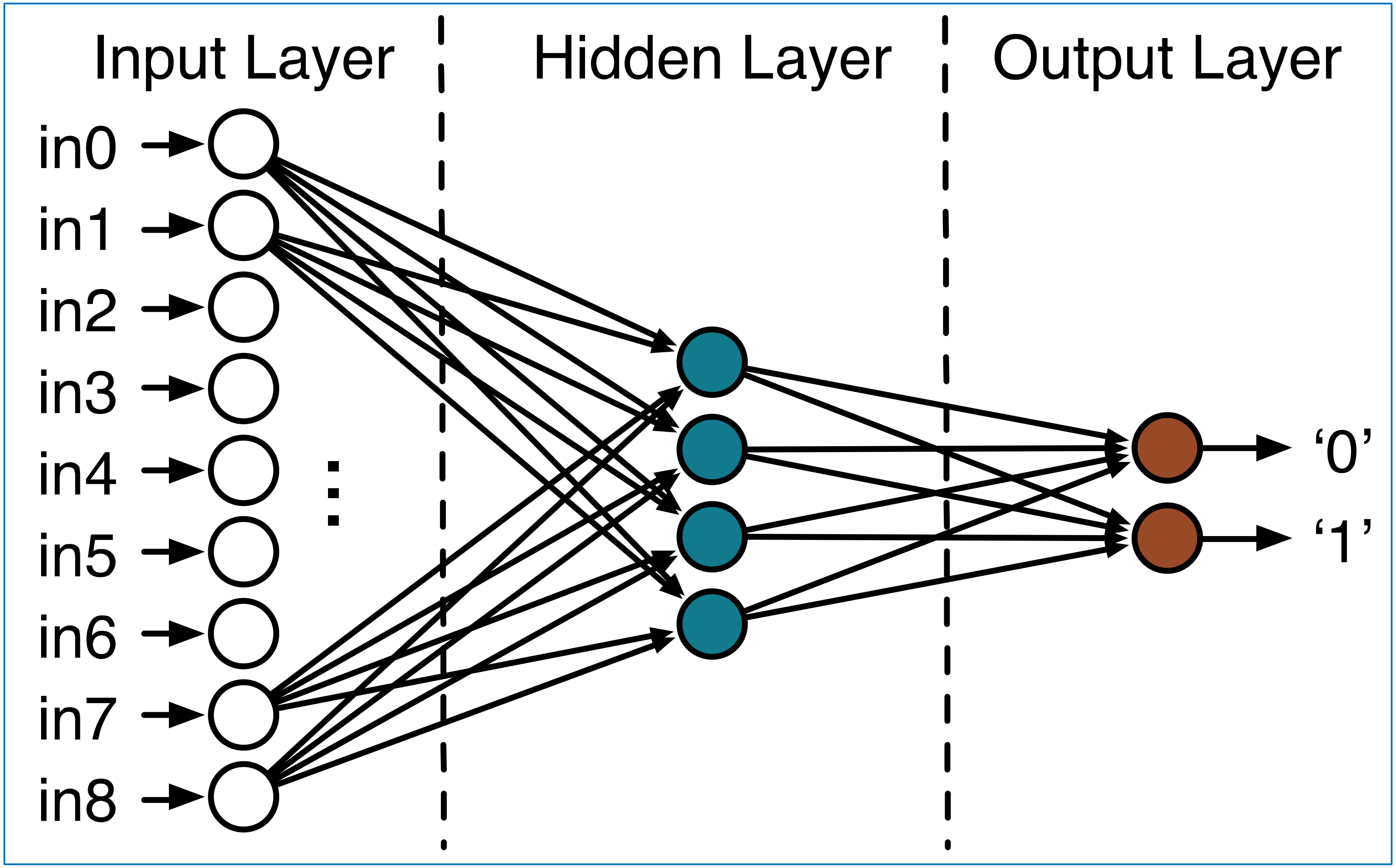


Table-based Classifiers



A small **ensemble** of table-based classifiers achieve better accuracy and performance

Neural Network Based Classifiers

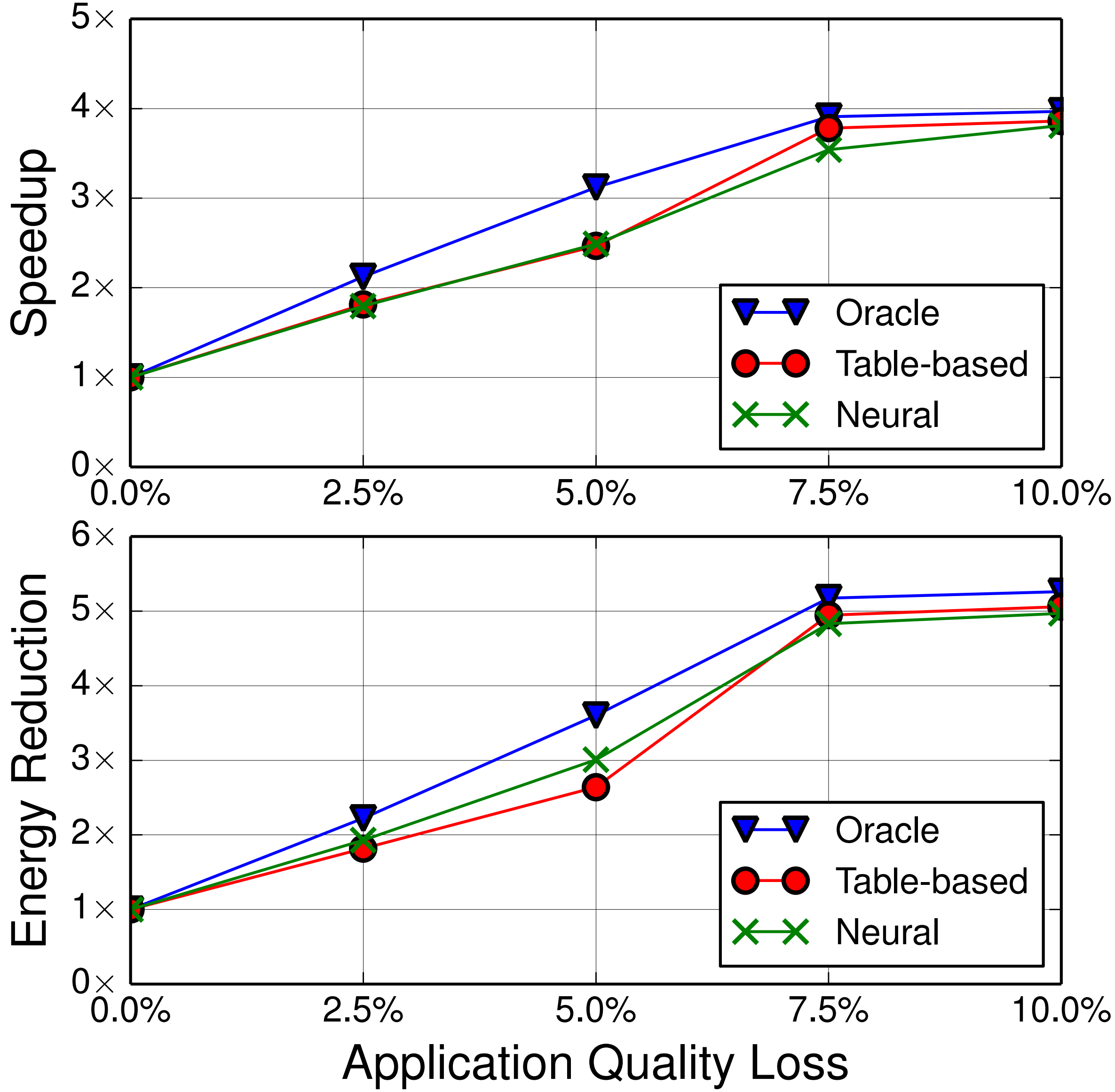


Benchmarks

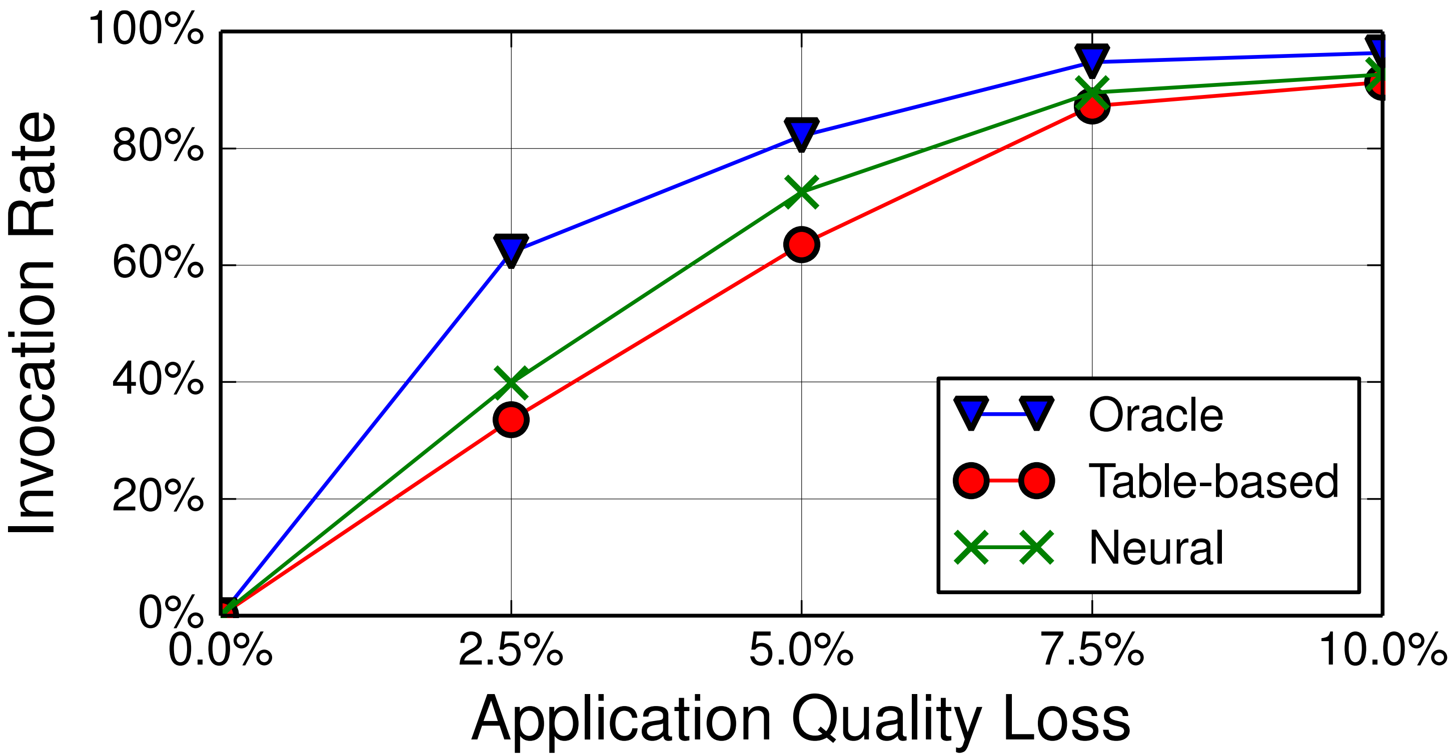
	Accelerator Topology	Baseline Error	Neural Classifier Topology
Blackscholes	6 → 8 → 8→1	6.02%	6 → 4 → 2
FFT	1→ 4 → 4→2	7.22%	1→ 4 → 2
Inversek2j	2 → 8 → 2	7.50%	2 → 4 → 2
JMEINT	18 → 32 → 8→2	17.69%	18 → 16 →2
JPEG Encoding	64 → 16 → 64	7.00%	64 → 2 → 2
Sobel	9 → 8 →1	9.96%	9 → 4 →2

Table Classifier Topology	8 tables each of size 0.5 KB

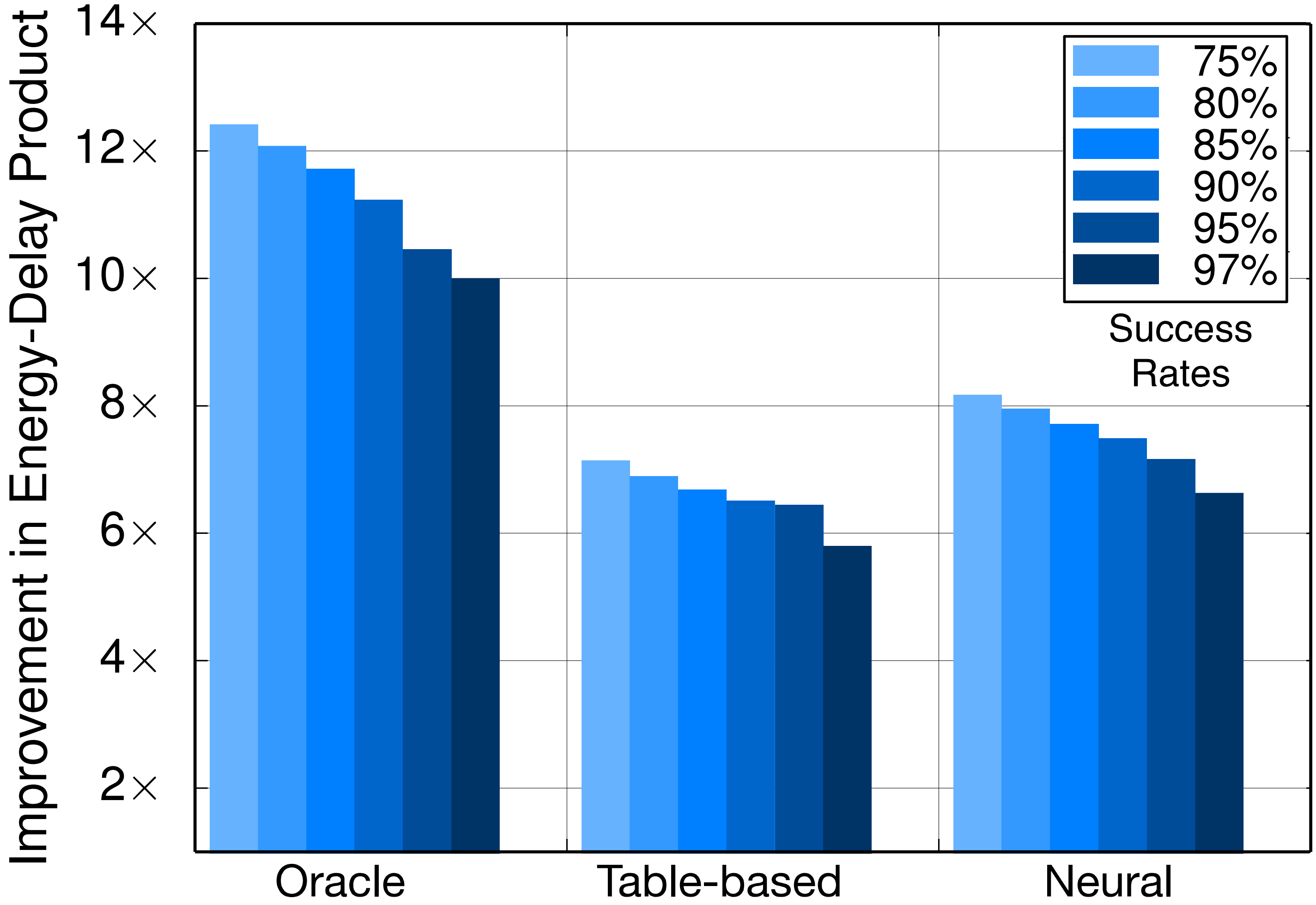
Energy and Performance Benefits



Invocation Rate



Varying Success Rate



Conclusion

MITHRA

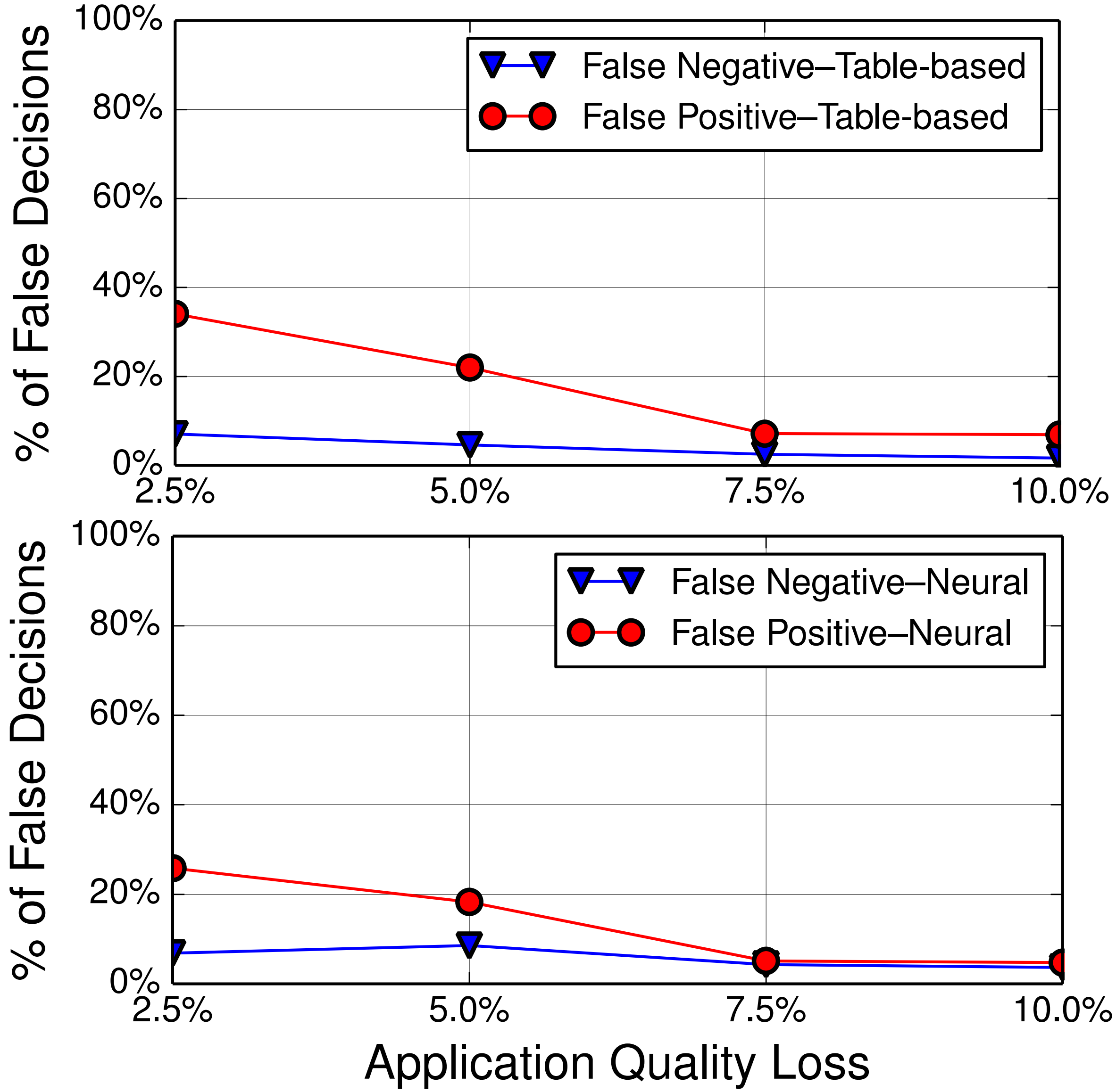
Hardware software co-design works well

Aims to make statistical guarantees a norm

Thank you

Thank you

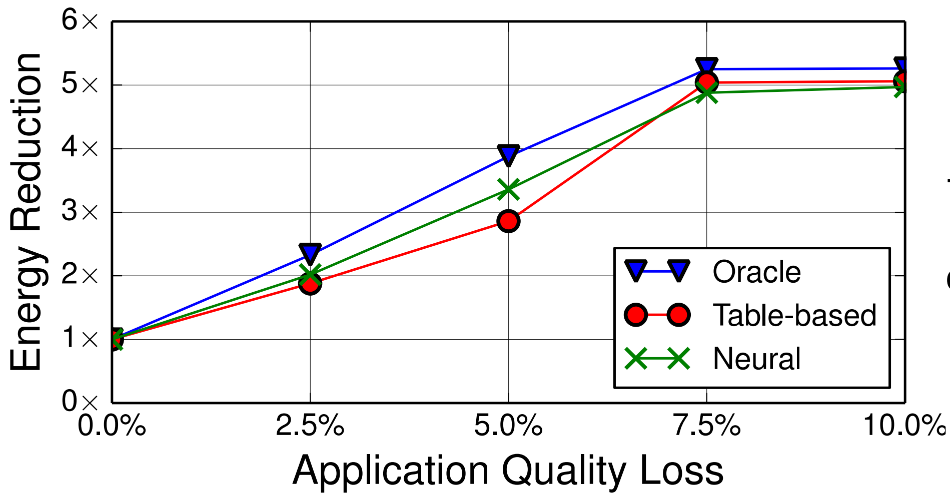
False Positive and False Negative Results

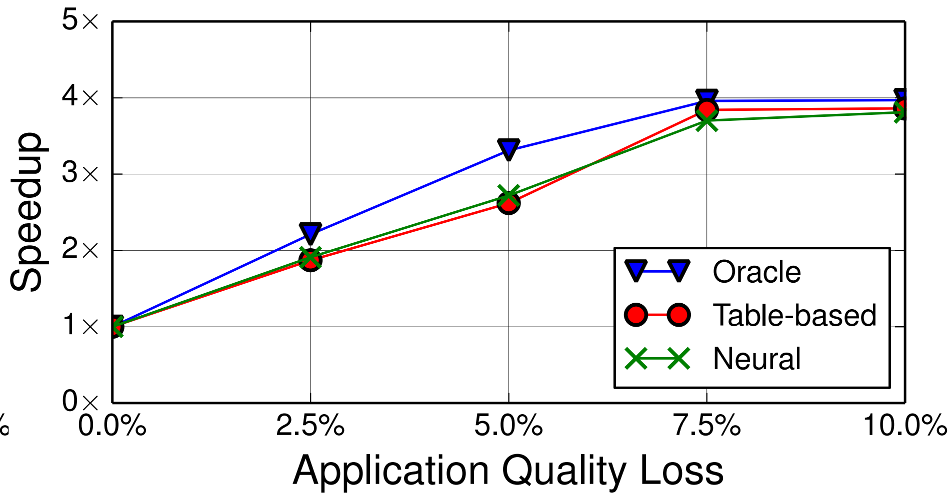


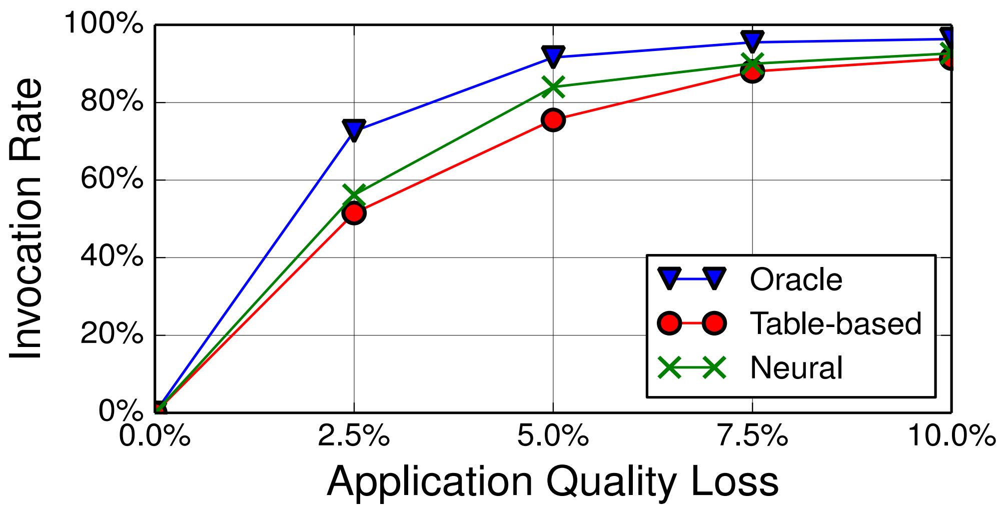
Binomial Proportion Confidence Interval

$(n_{trials}, n_{success})$ ➡ $r < SuccessRate$
$with\ a\ confidence\ level$

Evaluation of MITHRA

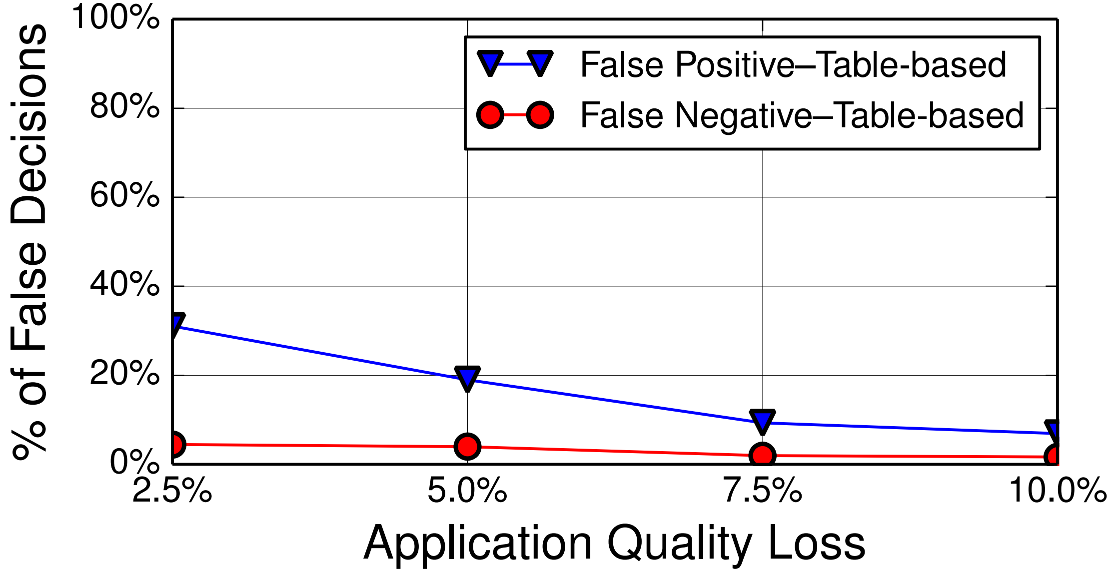


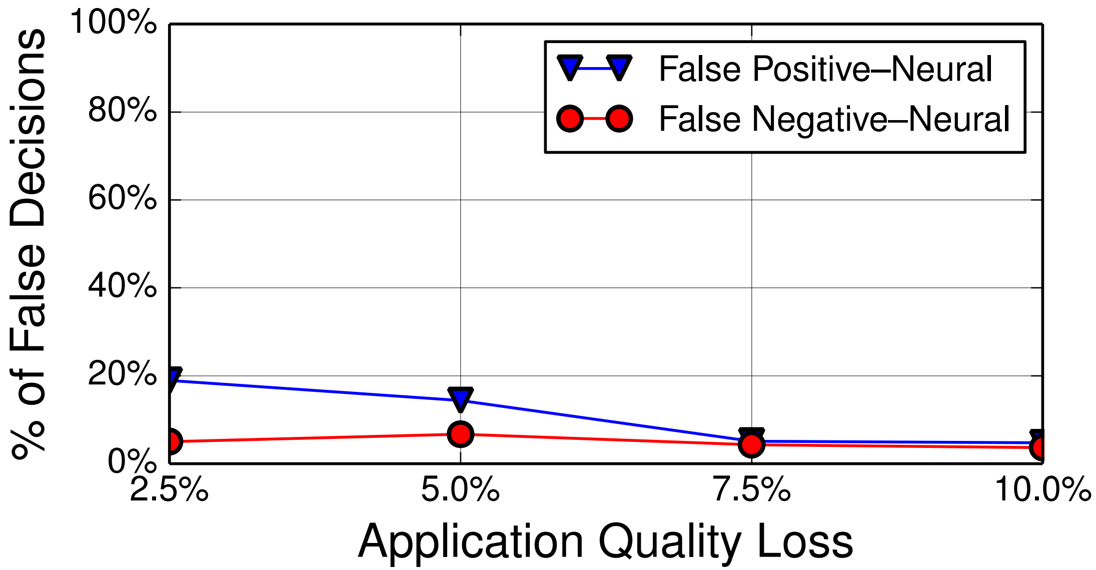




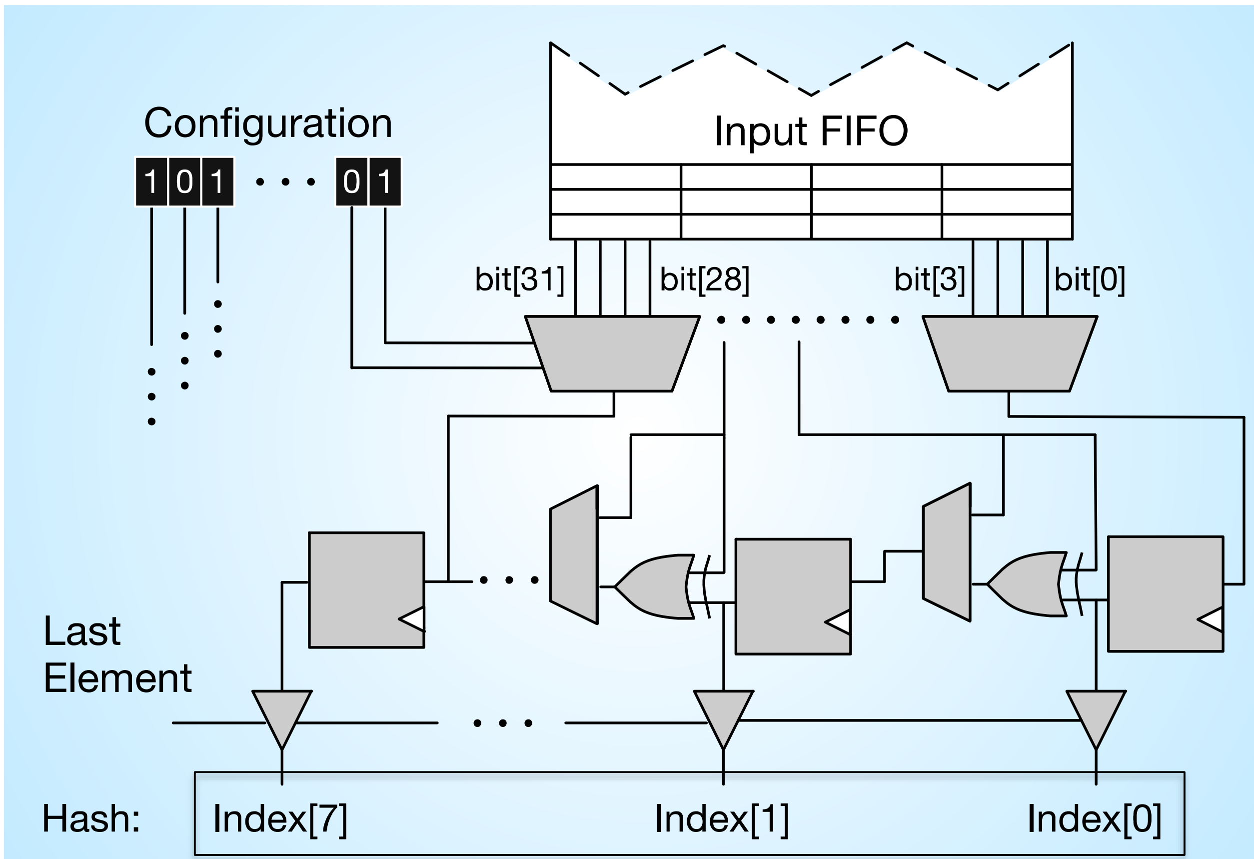
Comparing geometric mean - speedup, energy reduction and invocation rate across varied set of benchmarks.

False Positive and False Negatives

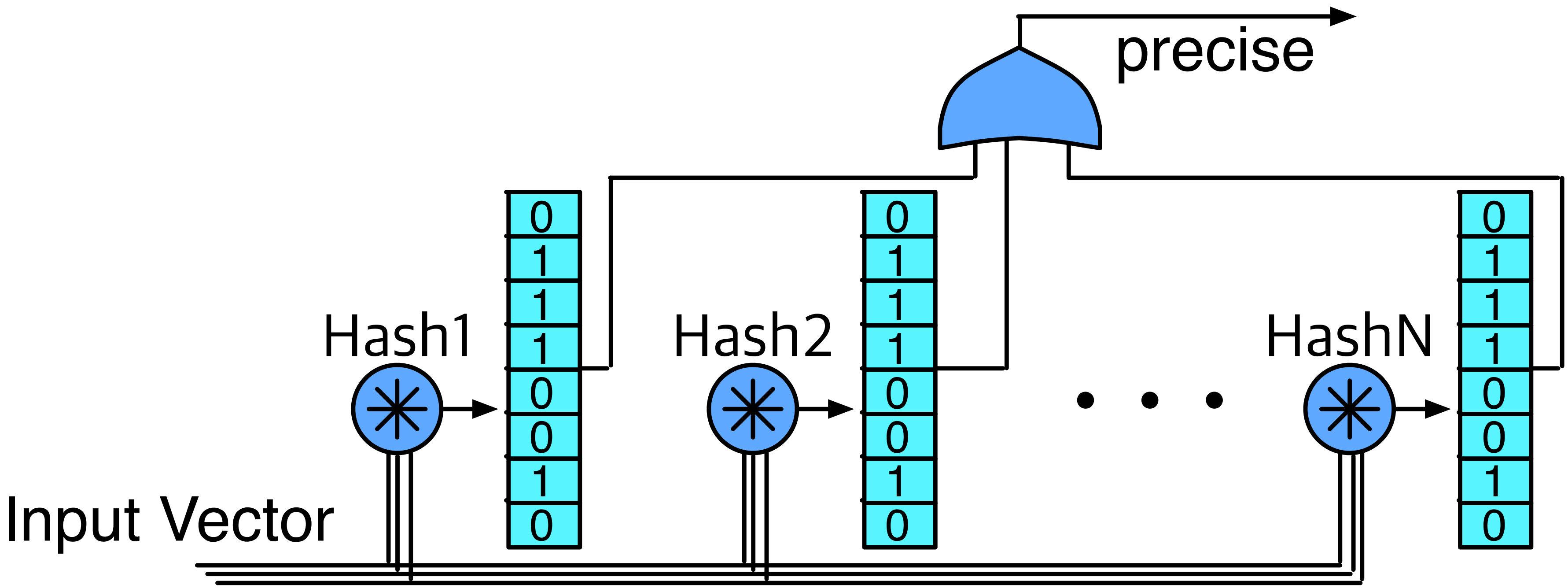




Multi-Input Hashing and Input Signature Generation

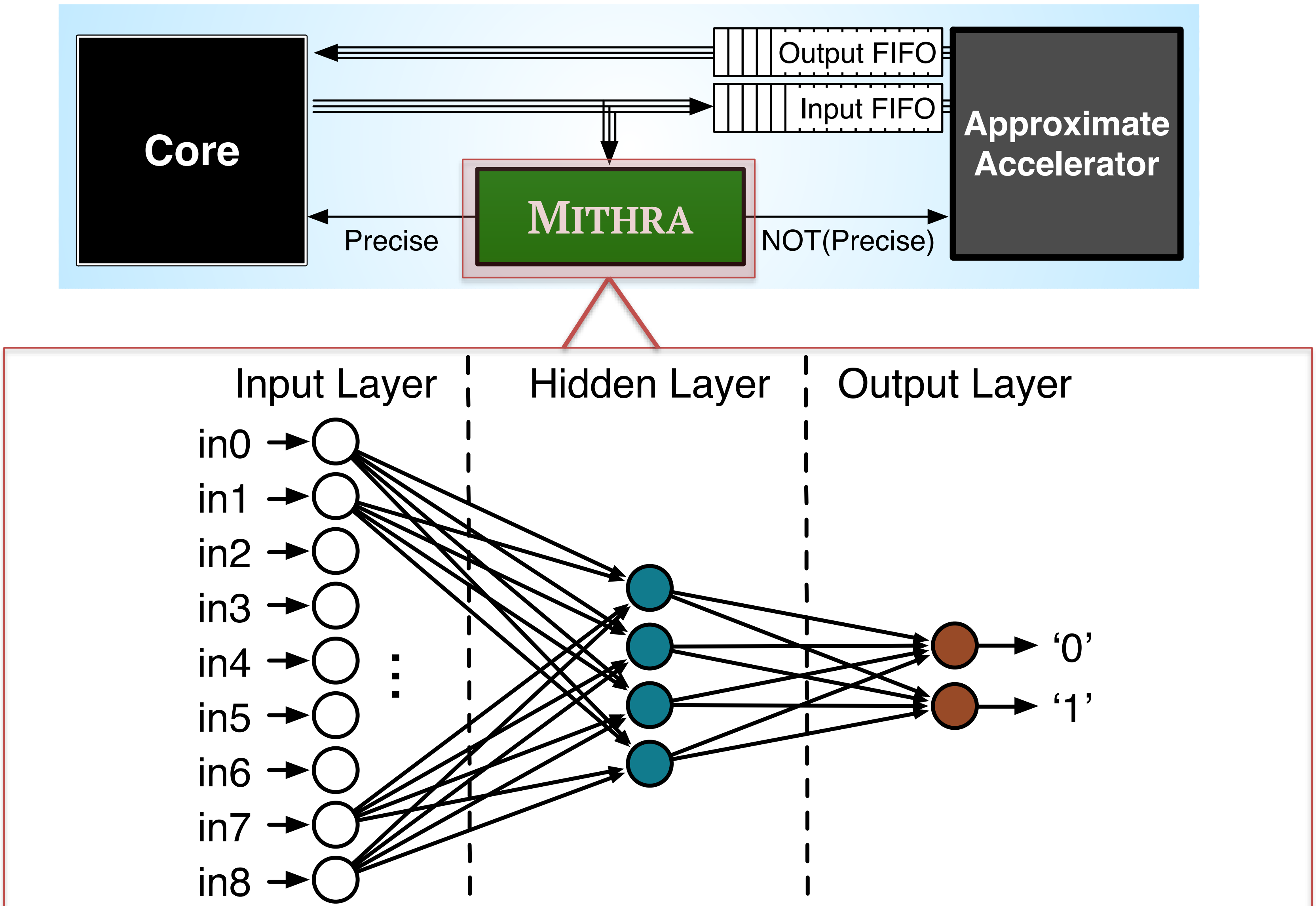


Multiple Tables for Improved Performance



A small ensemble of table-based Classifiers achieve better accuracy and performance

Neural Network Based Mechanism



Compiler Support for MITHRA

- Programmer provides:
  - Application specific quality requirement
  - Quality Metric
  - Set of representative application inputs

- Algorithm maps final output requirement to a threshold (**th**) on **accelerator error**.

Accelerator error < th : learn to invoke the accelerator
Accelerator error > th : learn to fall back to the original function

Quality vs Benefits Tradeoffs