ApproxCaliper: A Programmable Framework for Application-aware Neural Network Optimization

Yifan Zhao* · Hashim Sharif* · Peter Pao-Huang · Vatsin Shah ·
Arun Narenthiran Sivakumar ·
Mateus Valverde Gasparino · Abdulrahman Mahmoud ·
Nathan Zhao · Sarita Adve · Girish Chowdhary ·
Sasa Misailovic · Vikram Adve

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

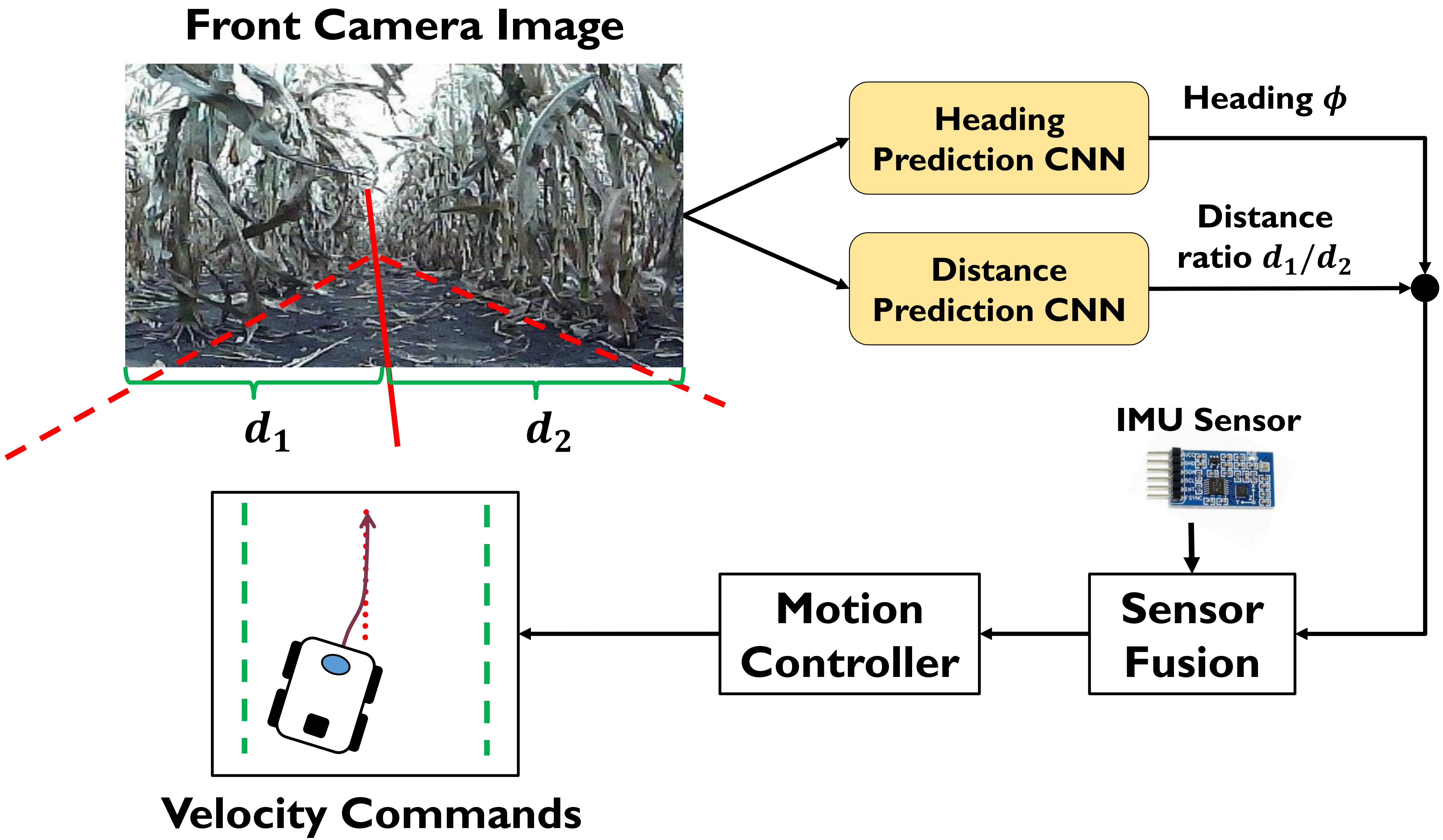
HARVARD UNIVERSITY

Agricultural Robot TerraSentia

EARTHSENSE
AGRICULTURAL INTELLIGENCE

Task: autonomous row-following for various agricultural applications

TerraSentia Navigation Pipeline



Front Camera Image

d_1

d_2

Heading Prediction CNN

Distance Prediction CNN

Heading ϕ

Distance ratio d_1/d_2

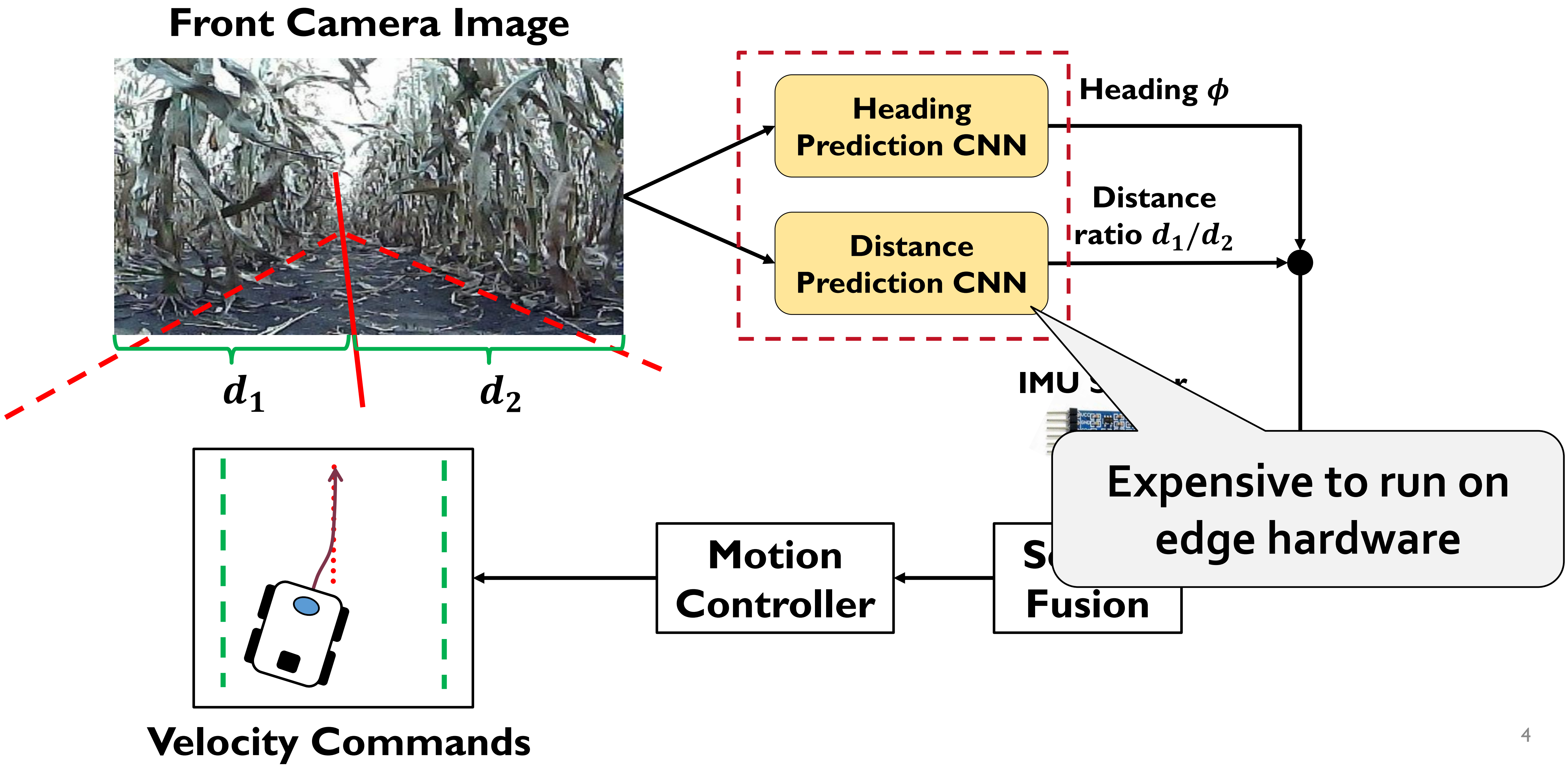
IMU Sensor

Sensor Fusion

Motion Controller

Velocity Commands

TerraSentia Navigation Pipeline



Expensive to run on edge hardware

Optimizing NN-based Edge Applications

Quality Requirement:

Collisions == 0

🟩 Valid

Collisions > 0

🟥 Invalid

Optimize For:

High performance

Good battery life

Low cost of hardware

Lightweight

…

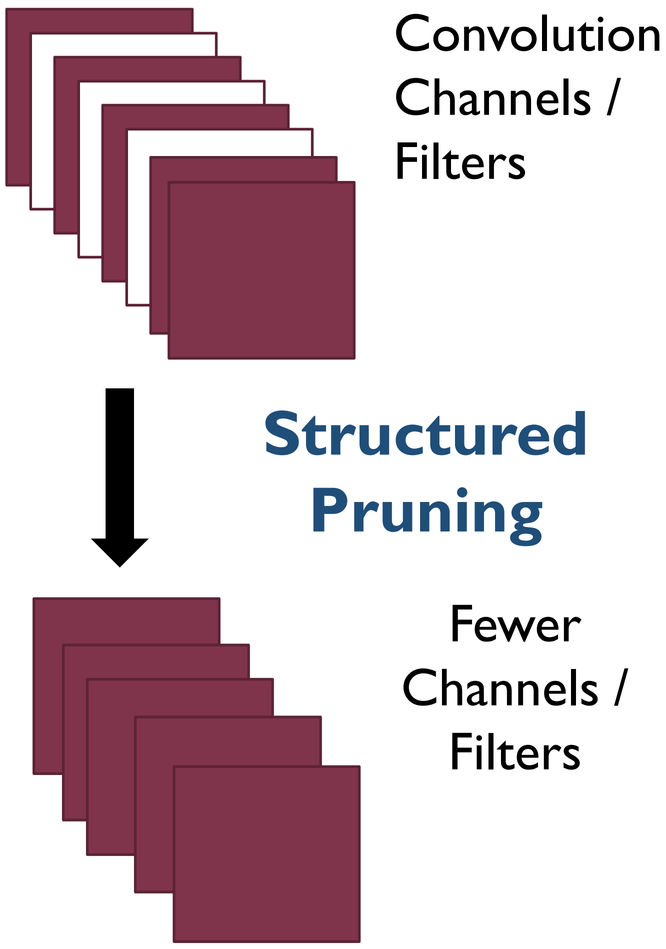
ApproxCaliper: Key Contributions

ApproxCaliper optimizes NNs while meeting **application-specific** goals & delivers **higher benefits** than application-agnostic tuning

Automates the optimization & **minimizes the search time** for approximations when application QoS checking is expensive

Neural Network Approximations

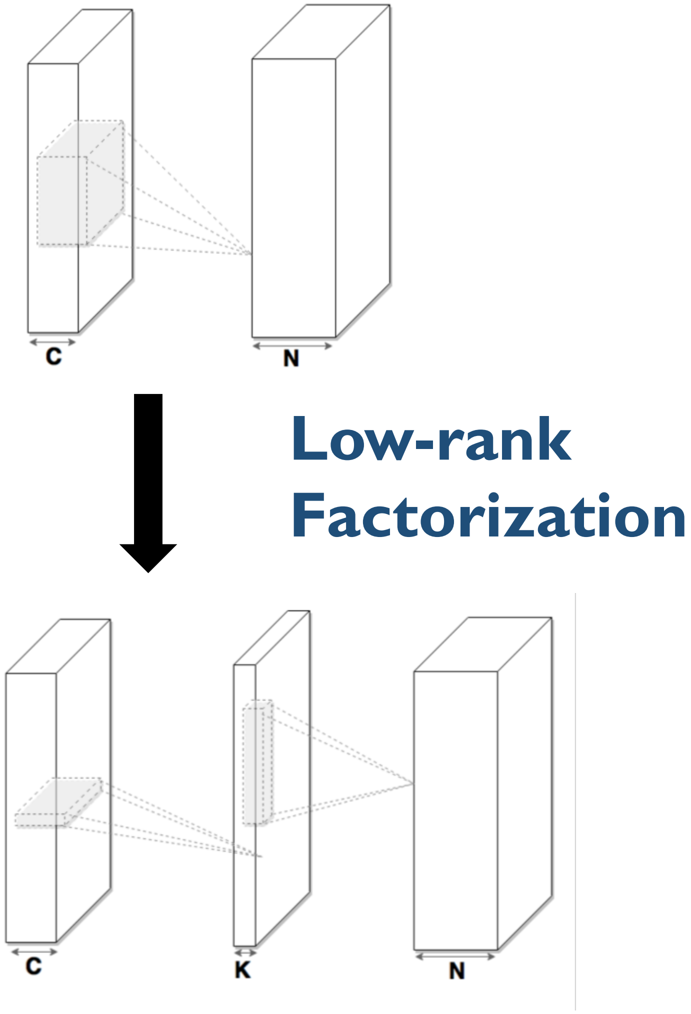
Lower latency, smaller model size, etc. at the cost of NN accuracy



Convolution Channels / Filters

Structured Pruning

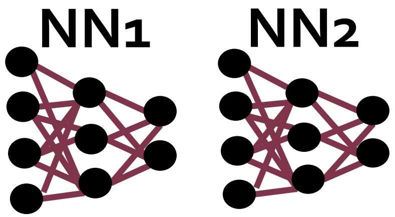
Fewer Channels / Filters



Low-rank Factorization

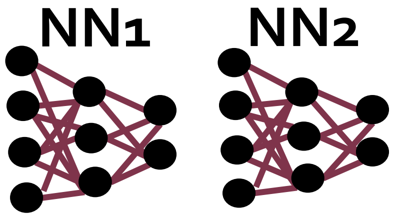
…

Manual Tuning is Too Expensive



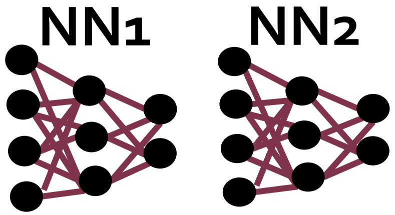
10% **Pr**uning

10% **L**ow-**R**ank **F**actorization



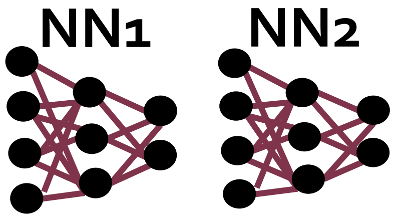
20% **Pr**

20% **LRF**



20% **Pr**

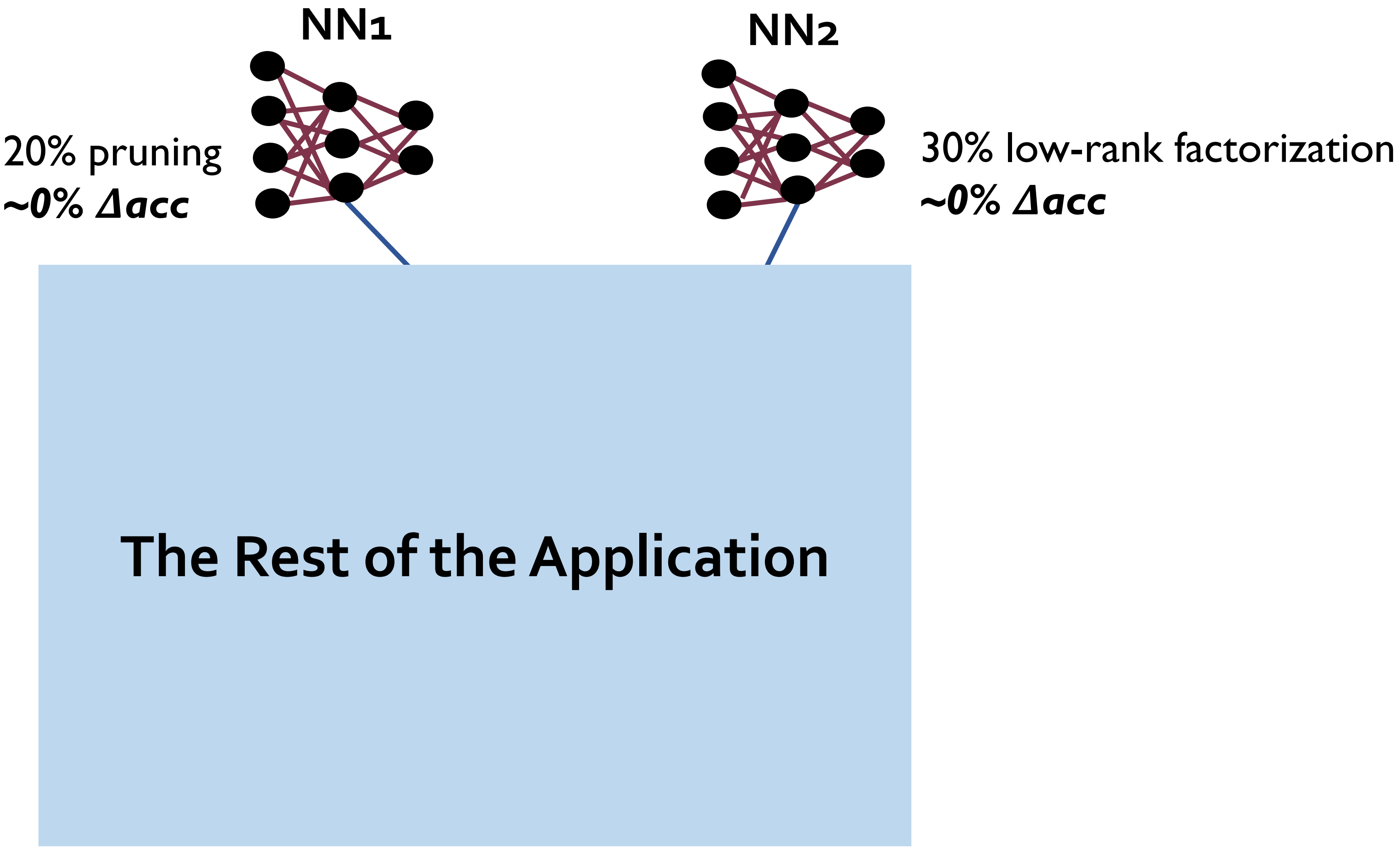
30% **Pr**



40% **LRF**

40% **Pr**

“Same NN Accuracy” is Too Conservative



NN1

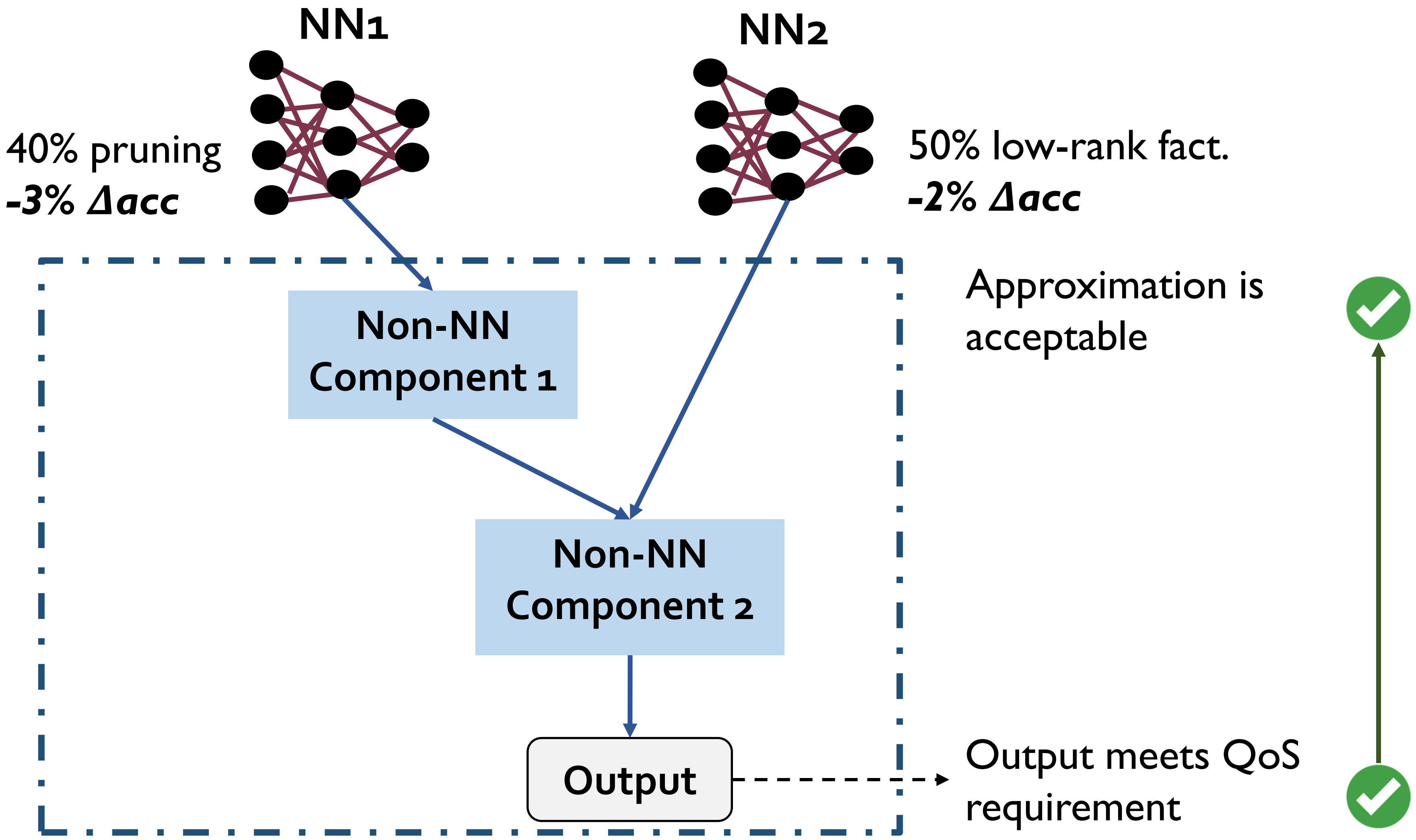
NN2

20% pruning
~0% Δacc

30% low-rank factorization
~0% Δacc

The Rest of the Application

Automatic Application-aware NN Optimization



NN1

NN2

40% pruning
-3% Δacc

50% low-rank fact.
-2% Δacc

Non-NN Component 1

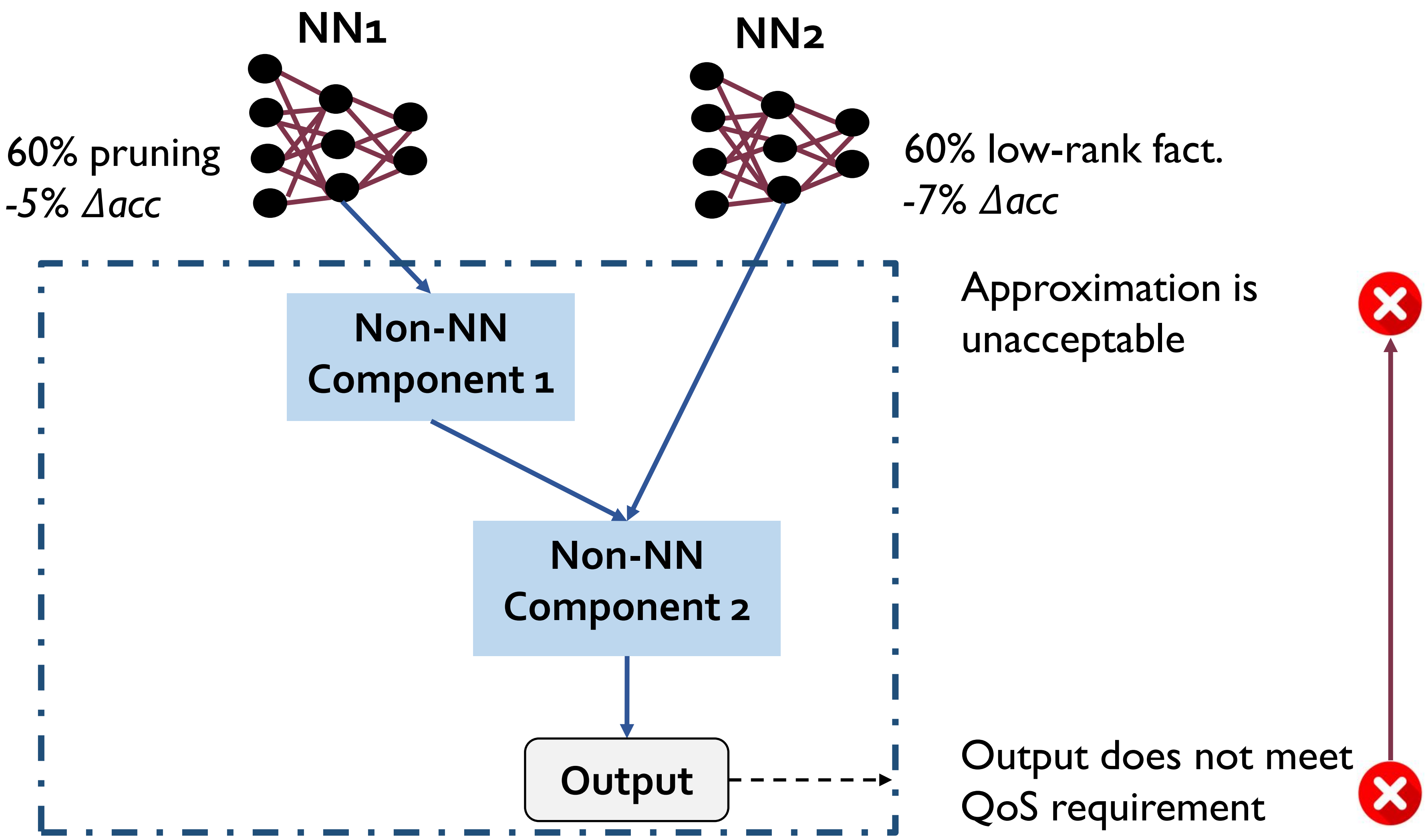
Non-NN Component 2

Output

Approximation is acceptable ✓

Output meets QoS requirement ✓

Automatic Application-aware NN Optimization



NN1

60% pruning
-5% *Δacc*

NN2

60% low-rank fact.
-7% *Δacc*

Non-NN Component 1

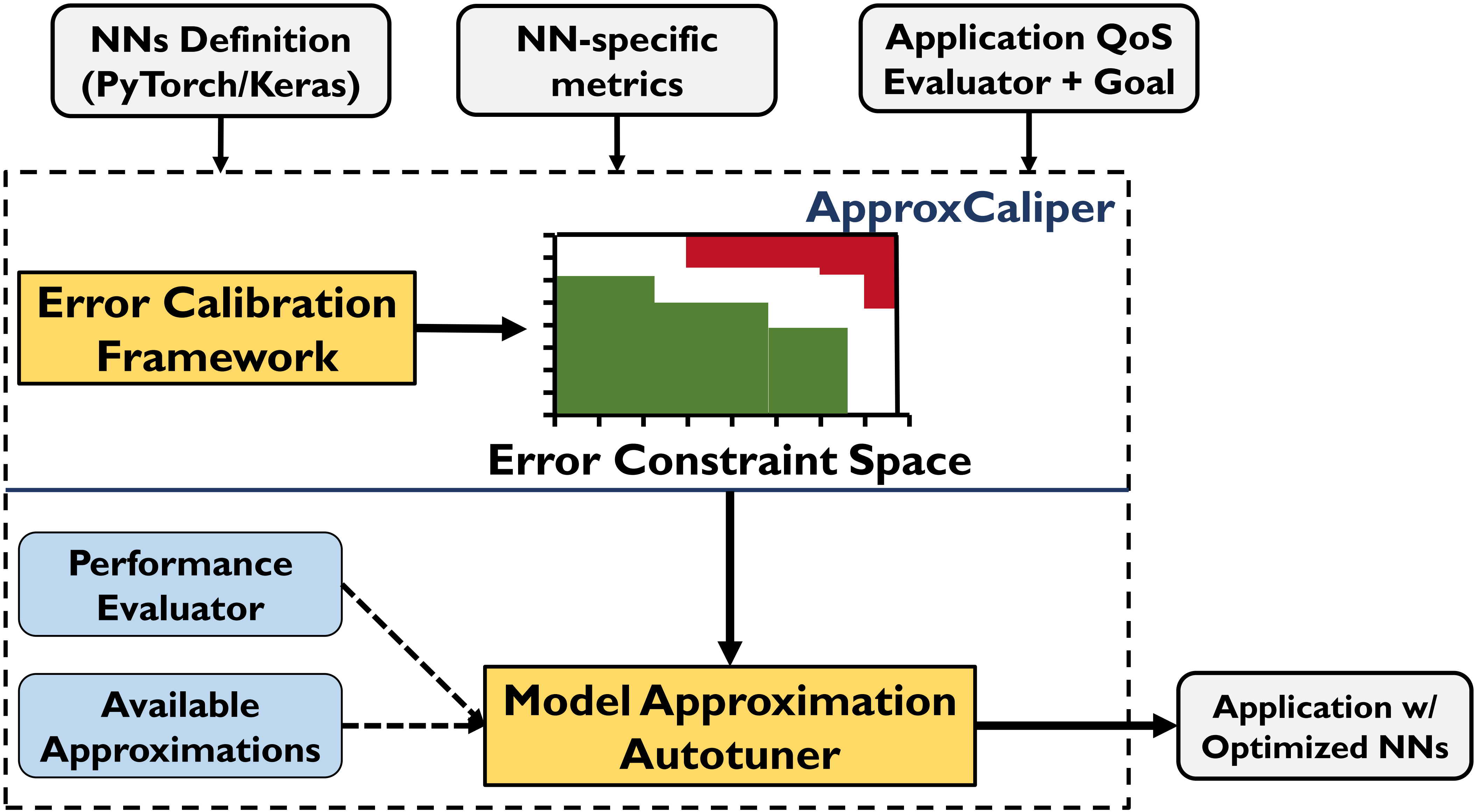
Non-NN Component 2

Output

Approximation is unacceptable ☒

Output does not meet QoS requirement ☒

ApproxCaliper Workflow



NNs Definition (PyTorch/Keras)

NN-specific metrics

Application QoS Evaluator + Goal

ApproxCaliper

Error Calibration Framework

Error Constraint Space

Performance Evaluator

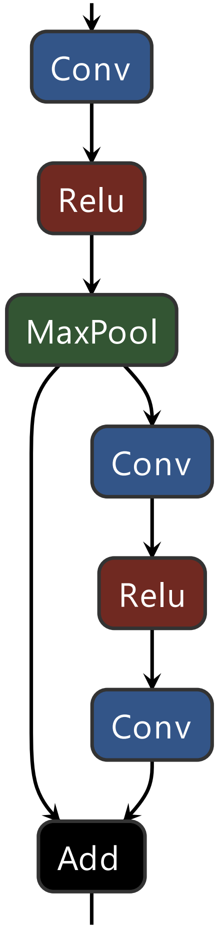
Available Approximations

Model Approximation Autotuner

Application w/ Optimized NNs

ApproxCaliper Workflow

NNs Definition (PyTorch/Keras)



NN-specific metrics

L1/L2 error

Regression tasks

Accuracy (%)

Classification tasks

⋮

Application QoS Evaluator & Goal

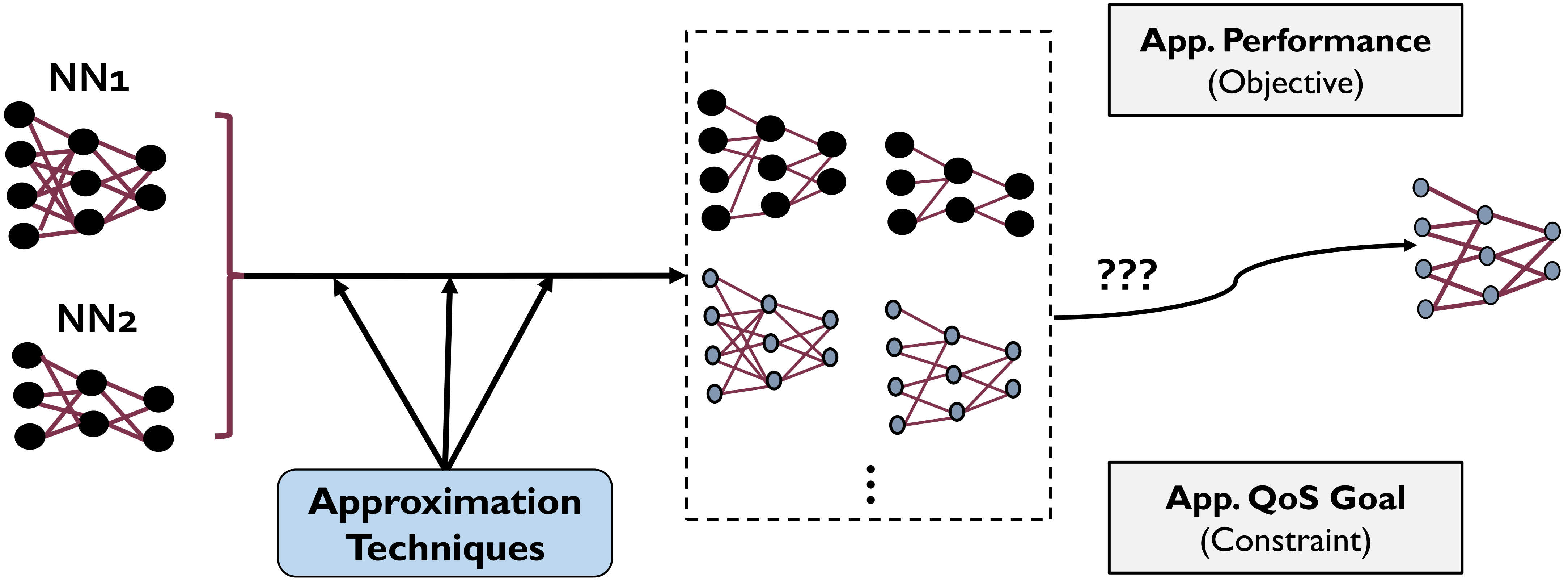
Collisions == 0

🟩 **Valid**

Collisions > 0

🟥 **Invalid**

Approximation Tuning in ApproxCaliper

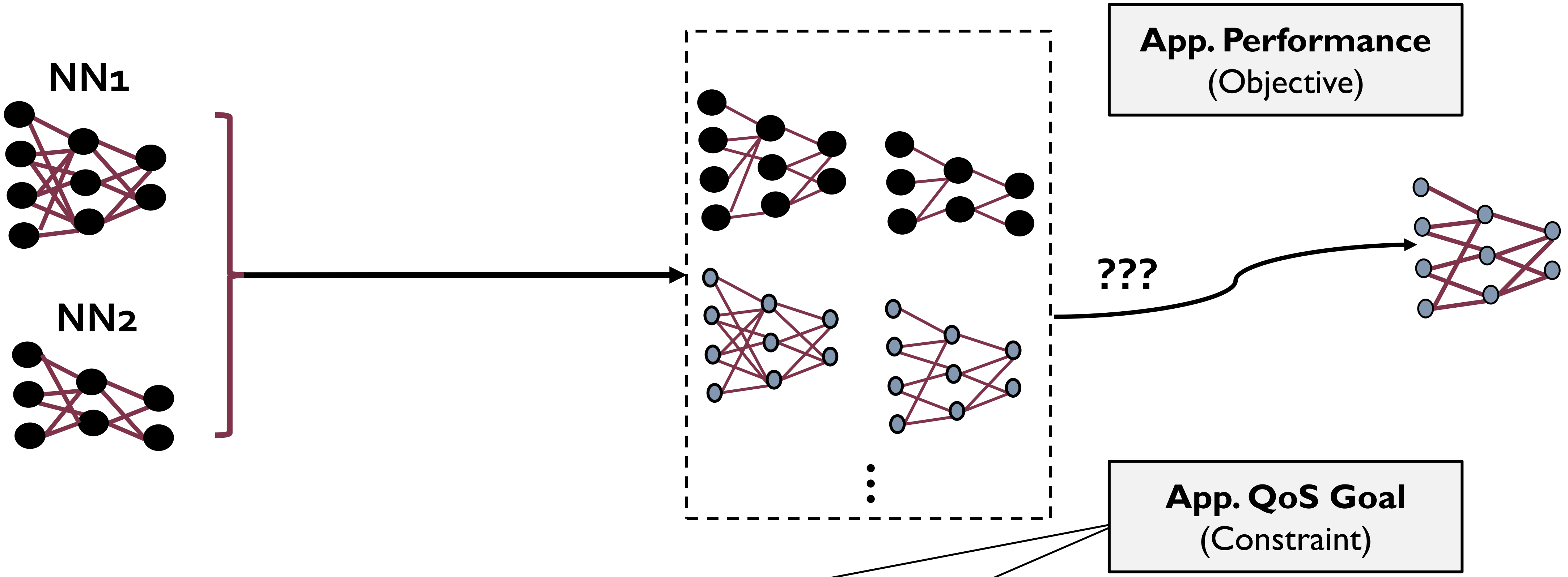


Initial NN Variants

Space of Approximated NNs

Optimized NN Selection

Approximation Tuning in ApproxCaliper



NN1

NN2

???

App. Performance
(Objective)

App. QoS Goal
(Constraint)

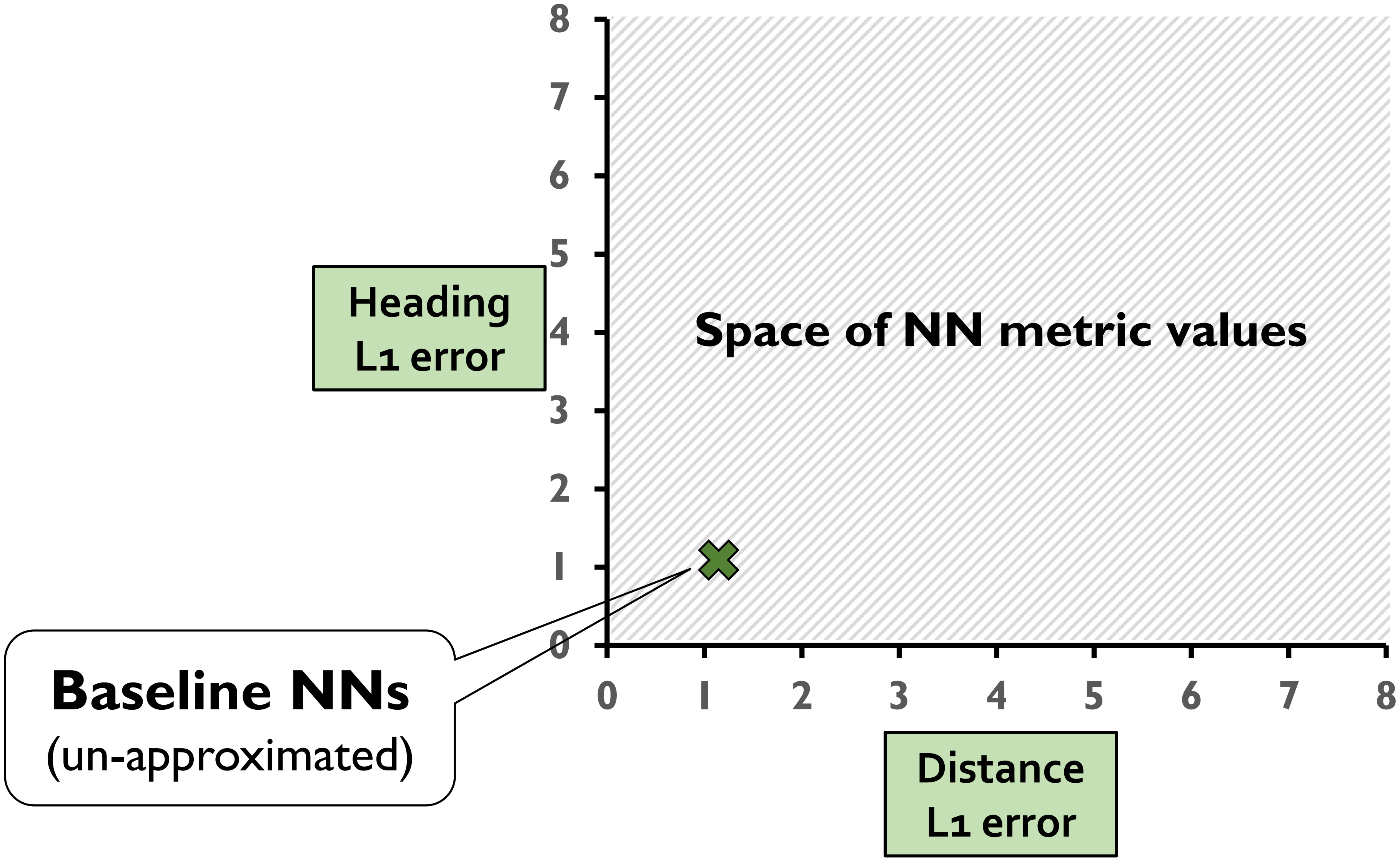
**Key Challenge:
Empirical QoS evaluations
are expensive**

Error Calibration

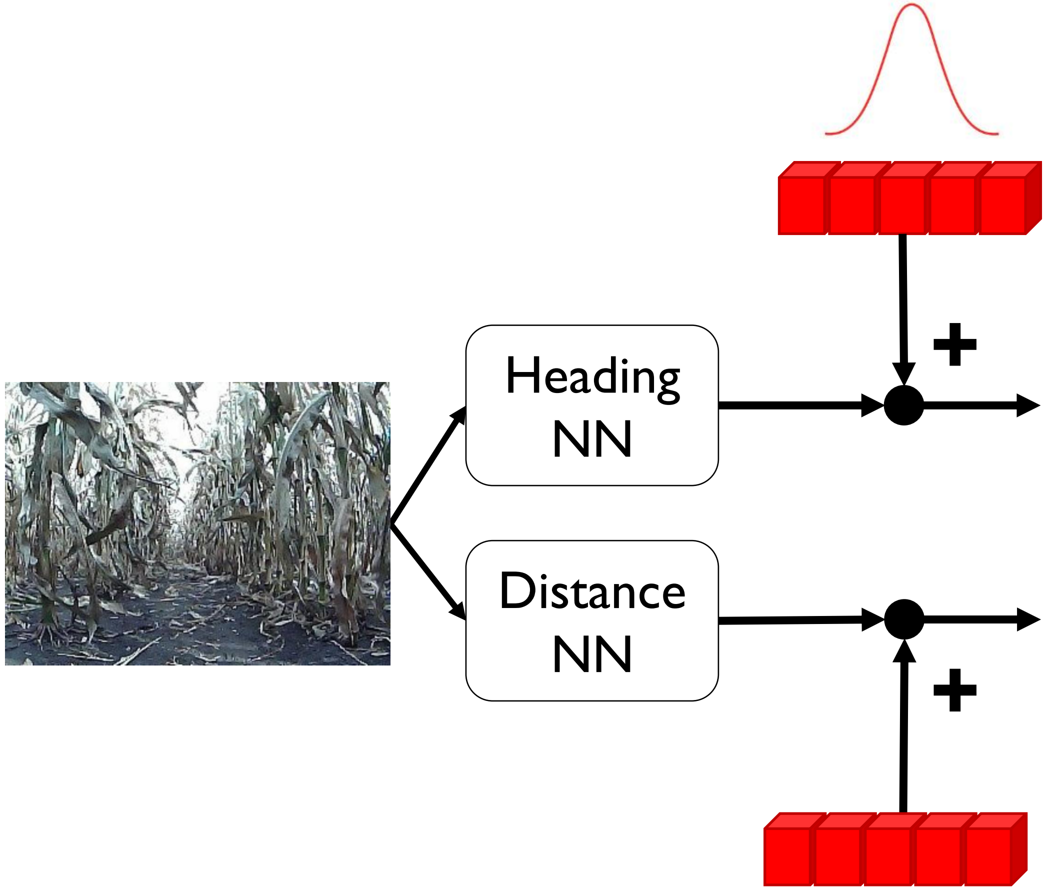
Goal: **predict** if application QoS is met from NN errors

NNs Definition (PyTorch/Keras)	NN-specific metrics	Application QoS Evaluator
Heading NN	Heading L1 error	Number of collisions
Distance NN	Distance L1 error	

Error Calibration



Error Injection

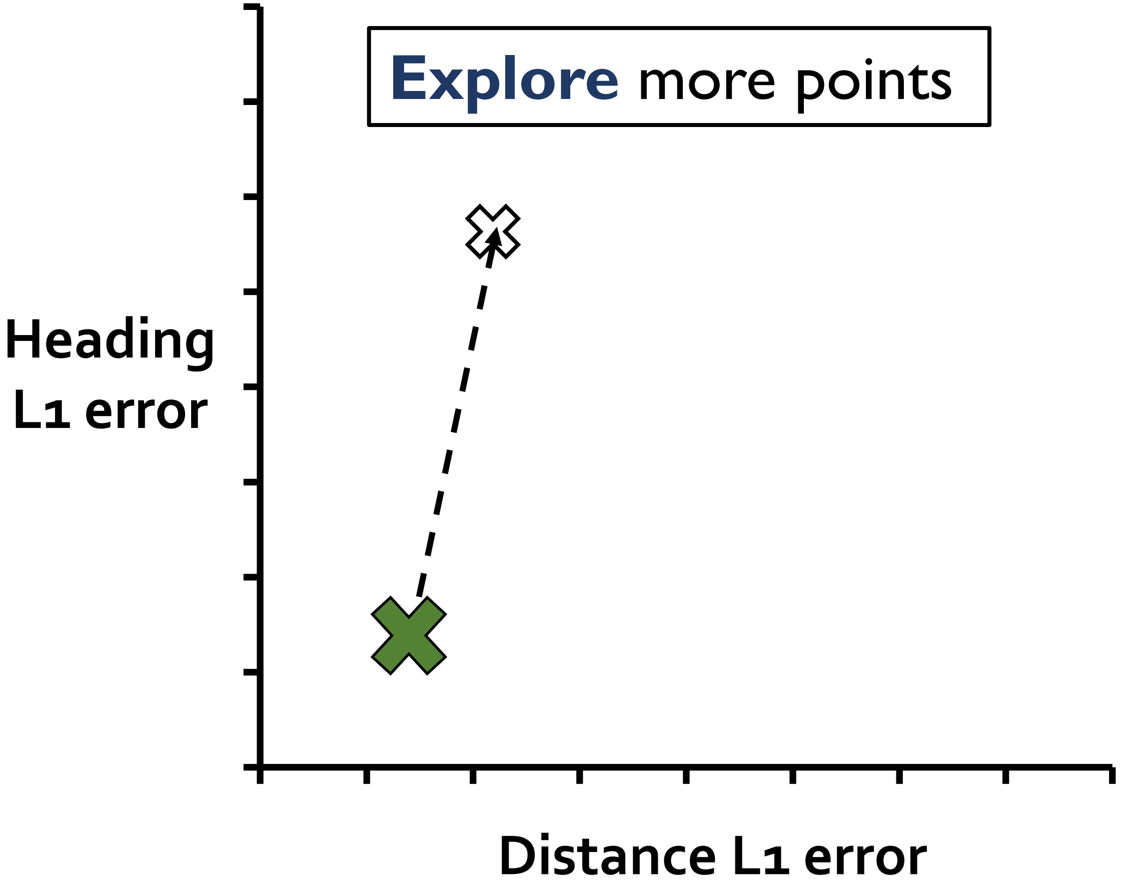


Heading NN

Distance NN

+

+

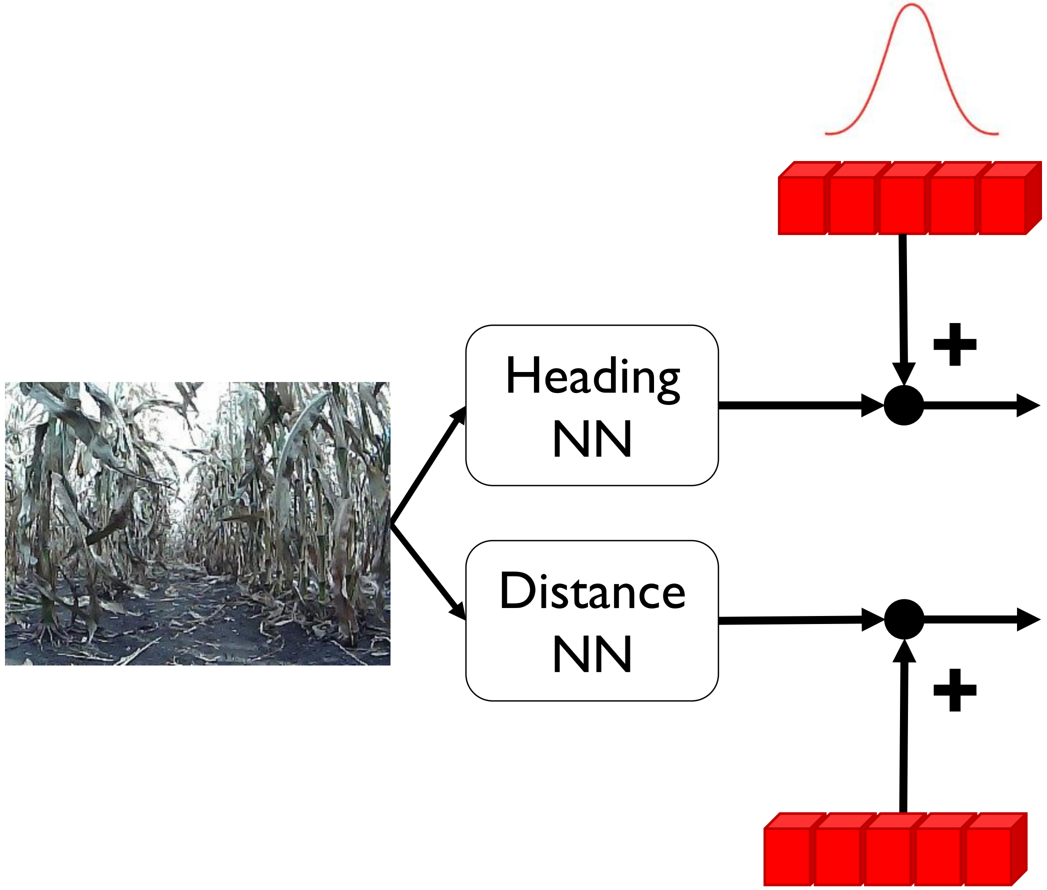


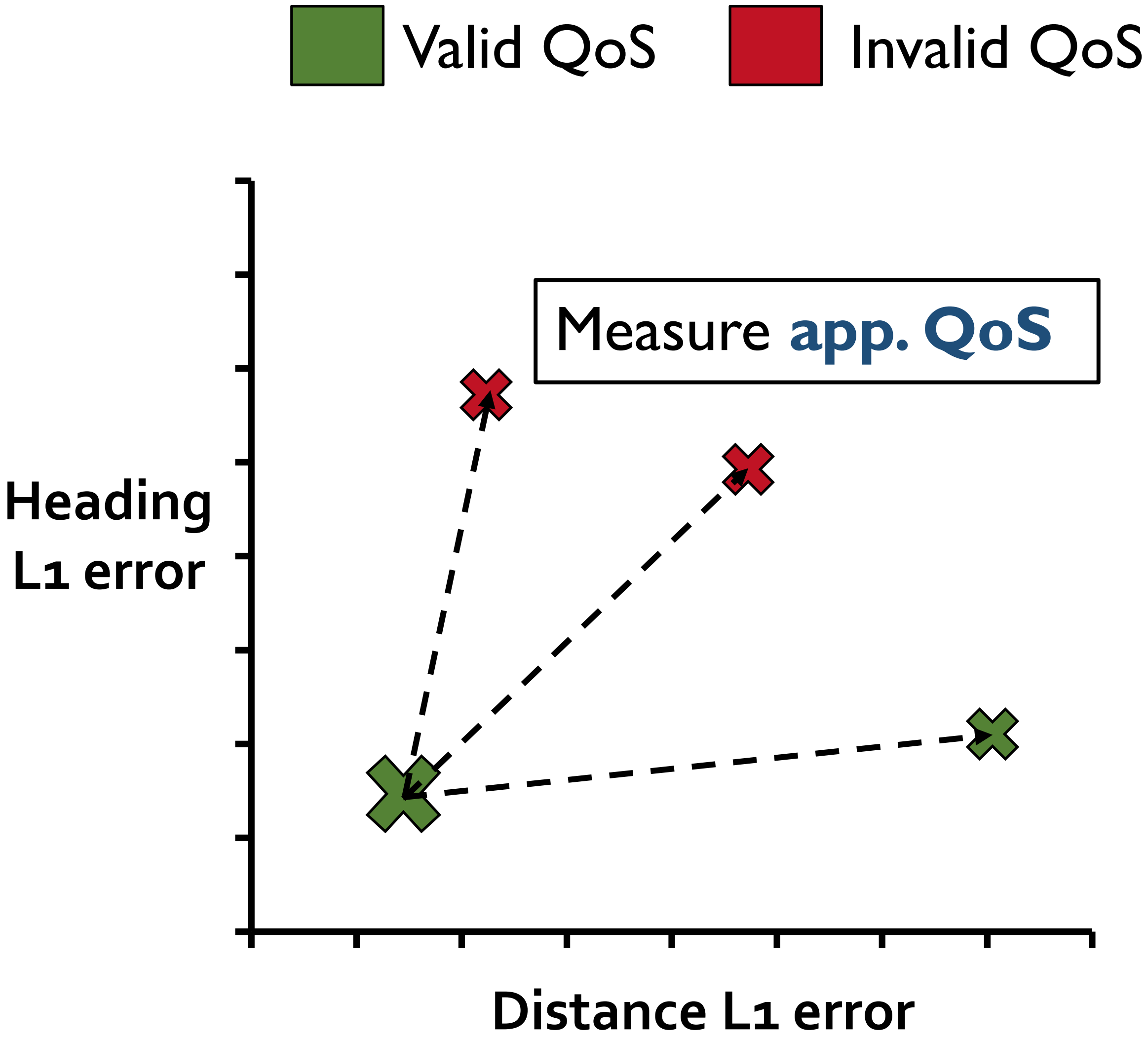
Explore more points

Heading L1 error

Distance L1 error

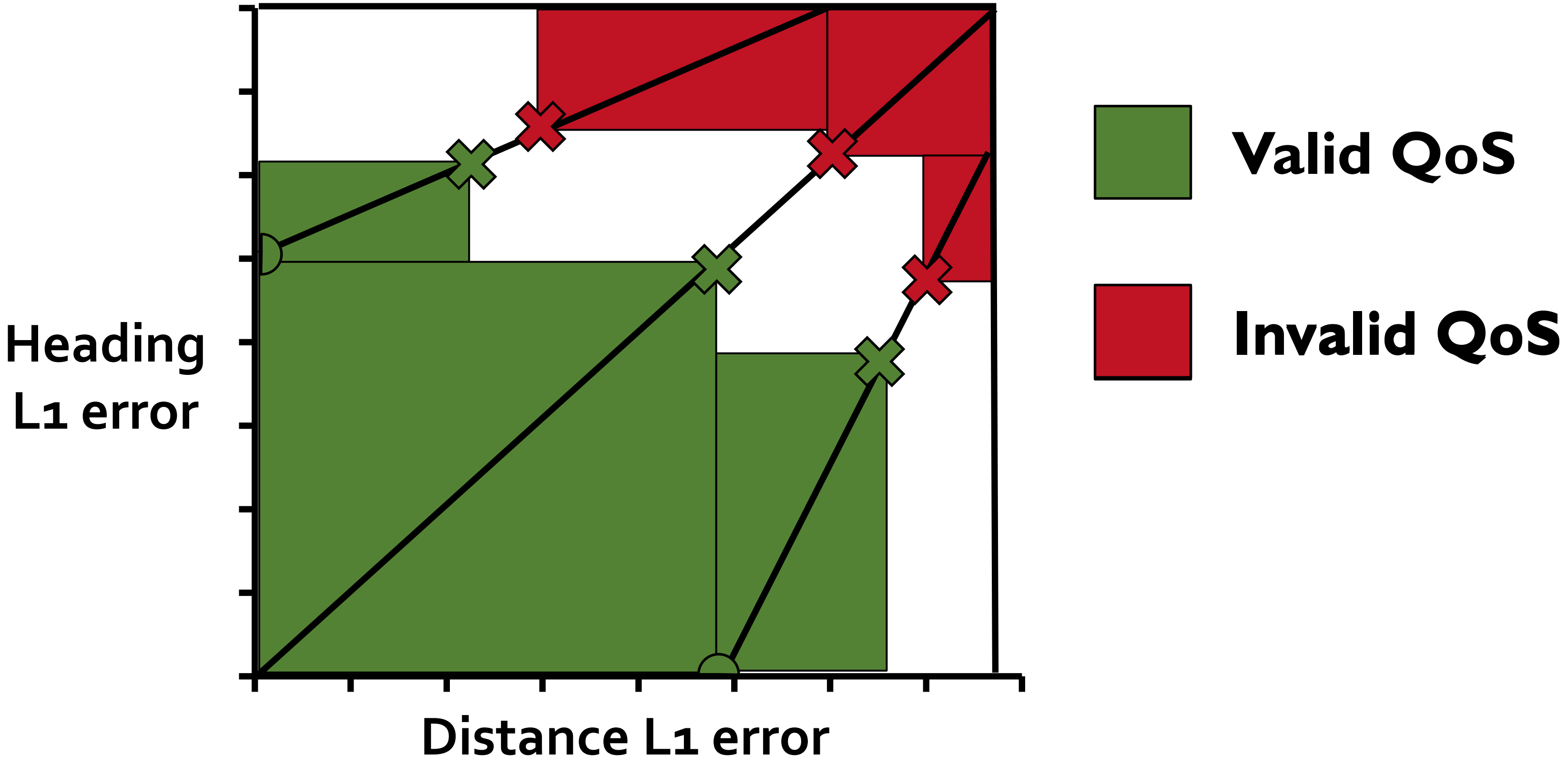
Error Injection





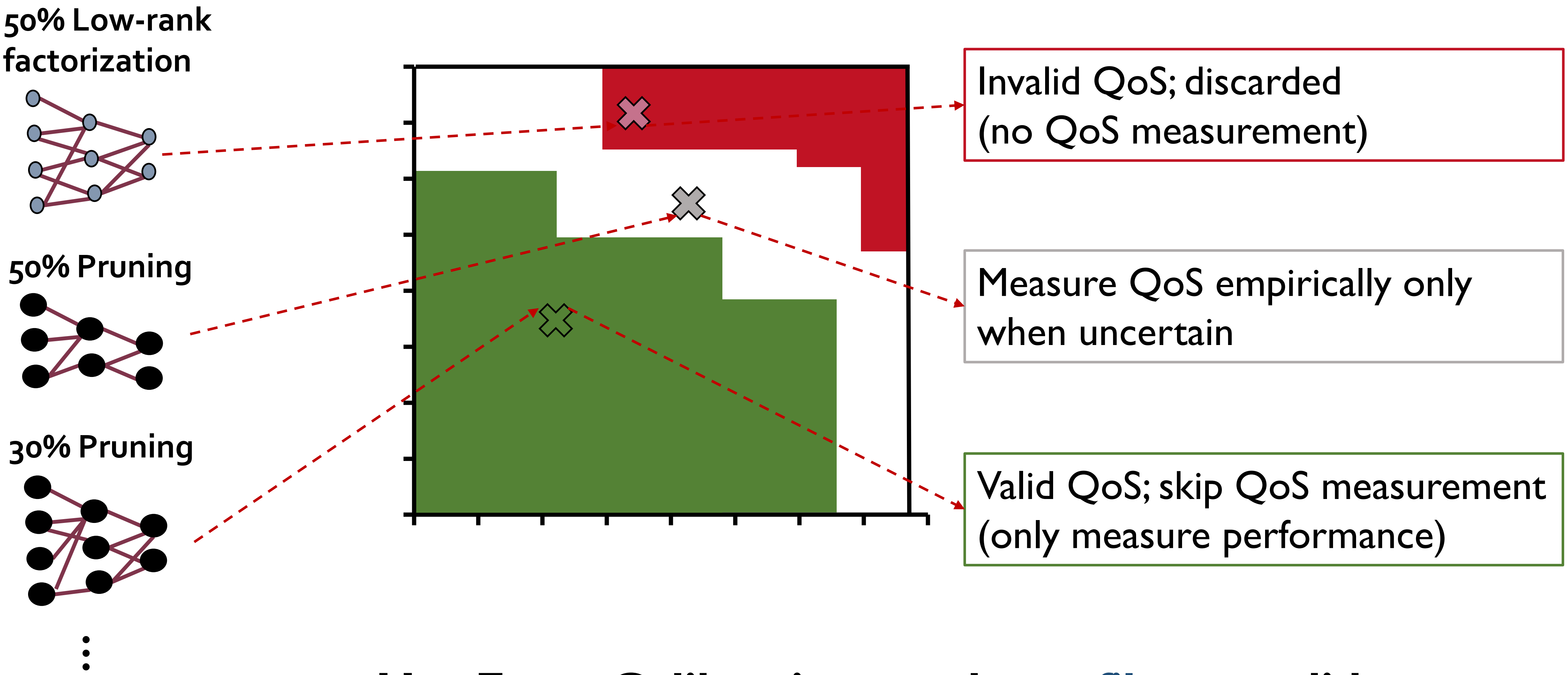
Measure **app. QoS**

Acceptable Accuracy Budgets



Determine which **region** yields acceptable app QoS

Guided Approximation Tuning



50% Low-rank factorization

50% Pruning

30% Pruning

⋮

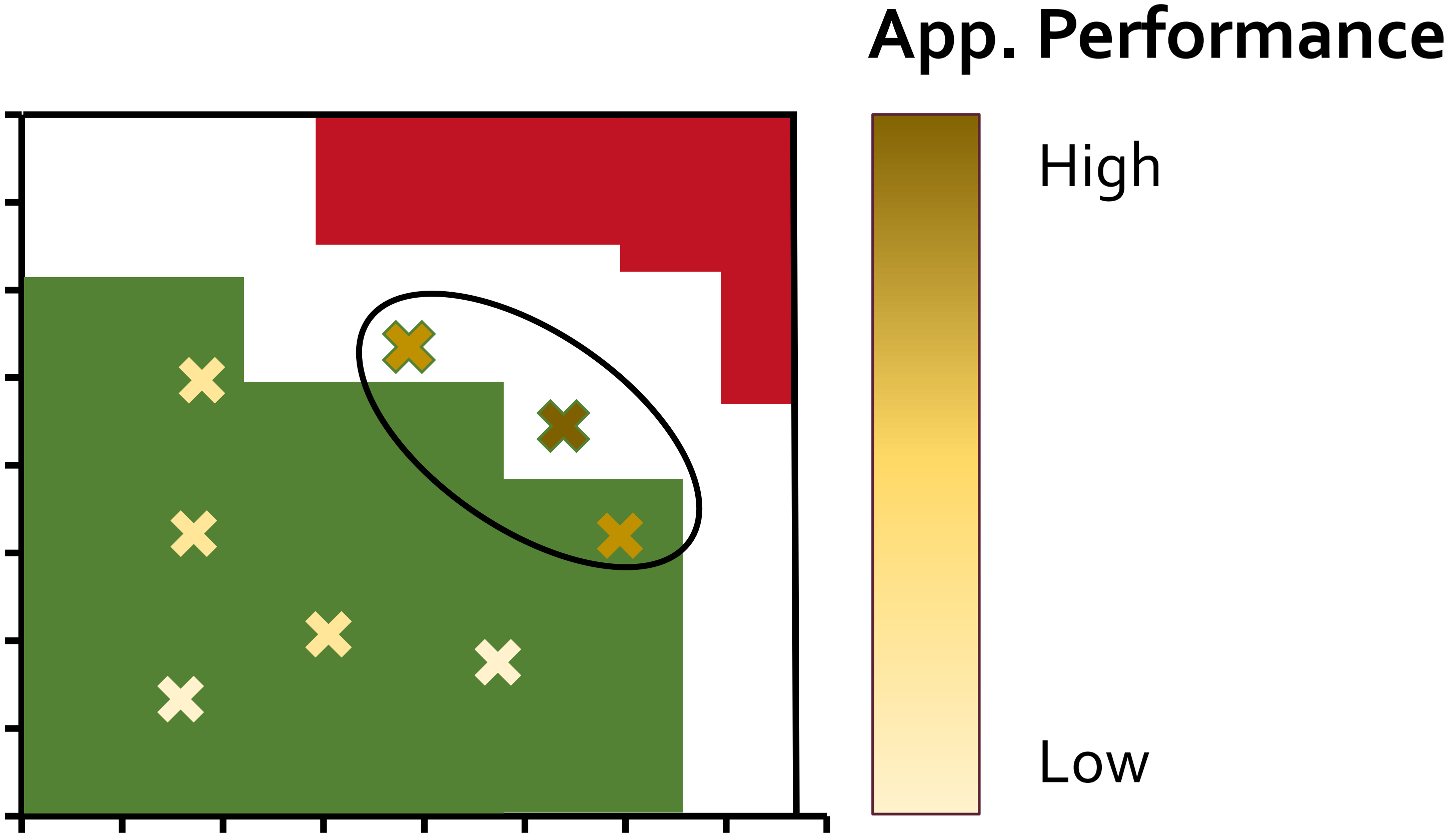
Invalid QoS; discarded (no QoS measurement)

Measure QoS empirically only when uncertain

Valid QoS; skip QoS measurement (only measure performance)

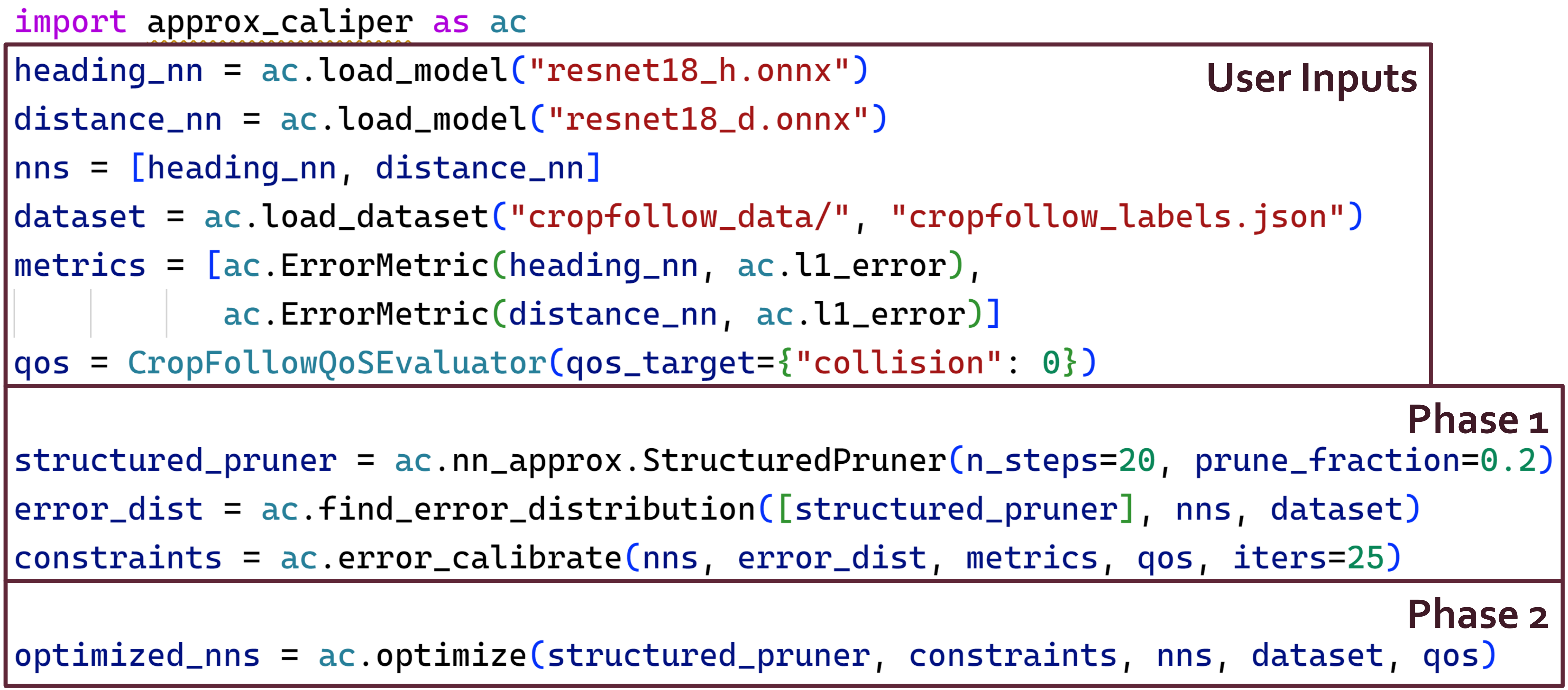
Use Error Calibration results to filter candidates

Guided Approximation Tuning



Select candidates with valid QoS and best performance

ApproxCaliper Programming Interface



User Inputs

Phase 1

Phase 2

```python
import approx_caliper as ac
heading_nn = ac.load_model("resnet18_h.onnx")
distance_nn = ac.load_model("resnet18_d.onnx")
nns = [heading_nn, distance_nn]
dataset = ac.load_dataset("cropfollow_data/", "cropfollow_labels.json")
metrics = [ac.ErrorMetric(heading_nn, ac.l1_error),
           ac.ErrorMetric(distance_nn, ac.l1_error)]
qos = CropFollowQoSEvaluator(qos_target={"collision": 0})
```

```python
structured_pruner = ac.nn_approx.StructuredPruner(n_steps=20, prune_fraction=0.2)
error_dist = ac.find_error_distribution([structured_pruner], nns, dataset)
constraints = ac.error_calibrate(nns, error_dist, metrics, qos, iters=25)
```

```python
optimized_nns = ac.optimize(structured_pruner, constraints, nns, dataset, qos)
```

Evaluation Setup: Applications Optimized

Agricultural Robot
TerraSentia

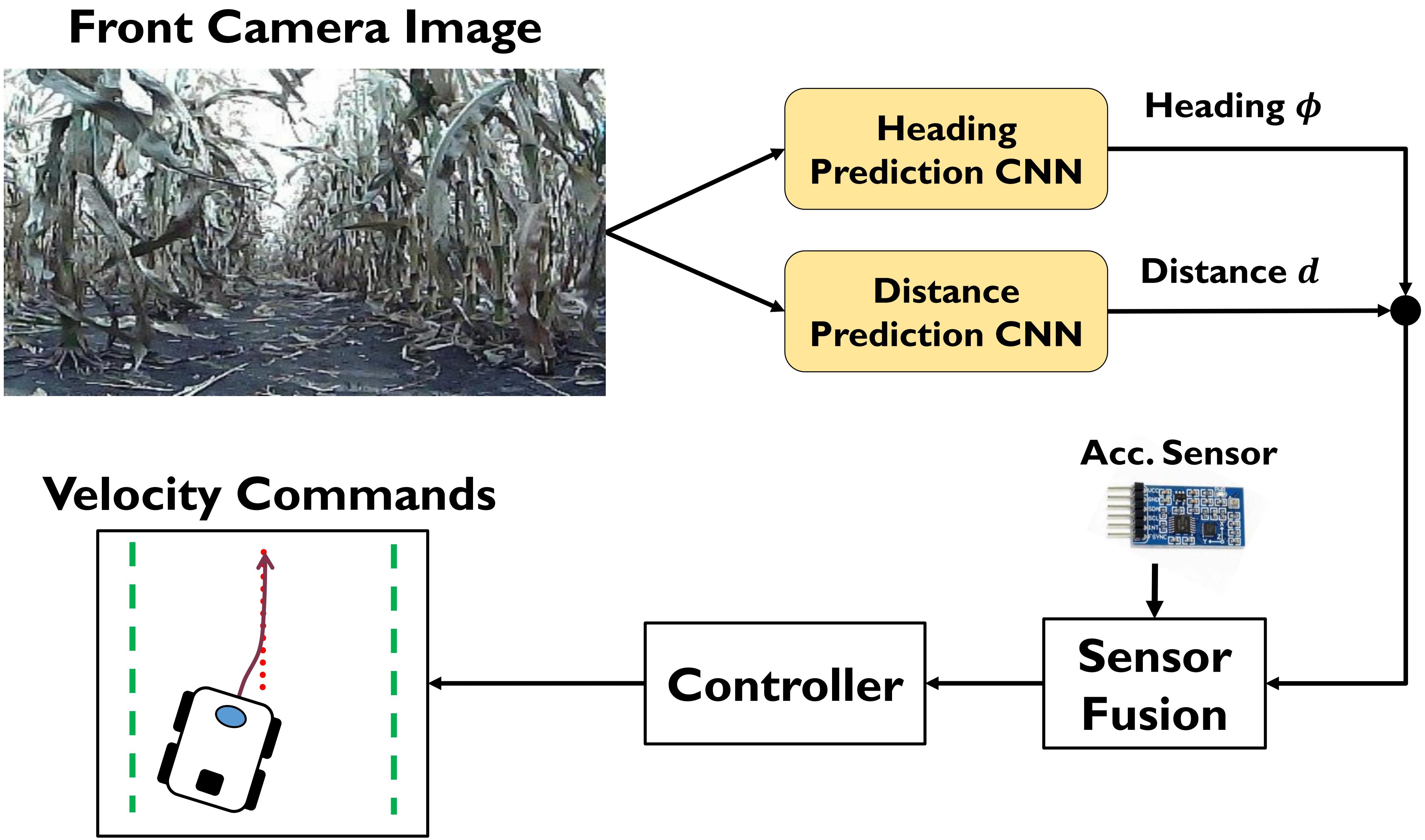
Task: autonomous row-following for various agricultural applications

Cart simulator
Polaris-GEM

Steps: 1 Real Time Factor: 0.83 Sim Time: 00 00:41:07.200 Real Time: 00 00:00:37.761 Iterations: 29962 FPS: 62.95

Task: lane-following on paved roads

TerraSentia Navigation Pipeline



Front Camera Image

Heading Prediction CNN

Distance Prediction CNN

Heading ϕ

Distance d

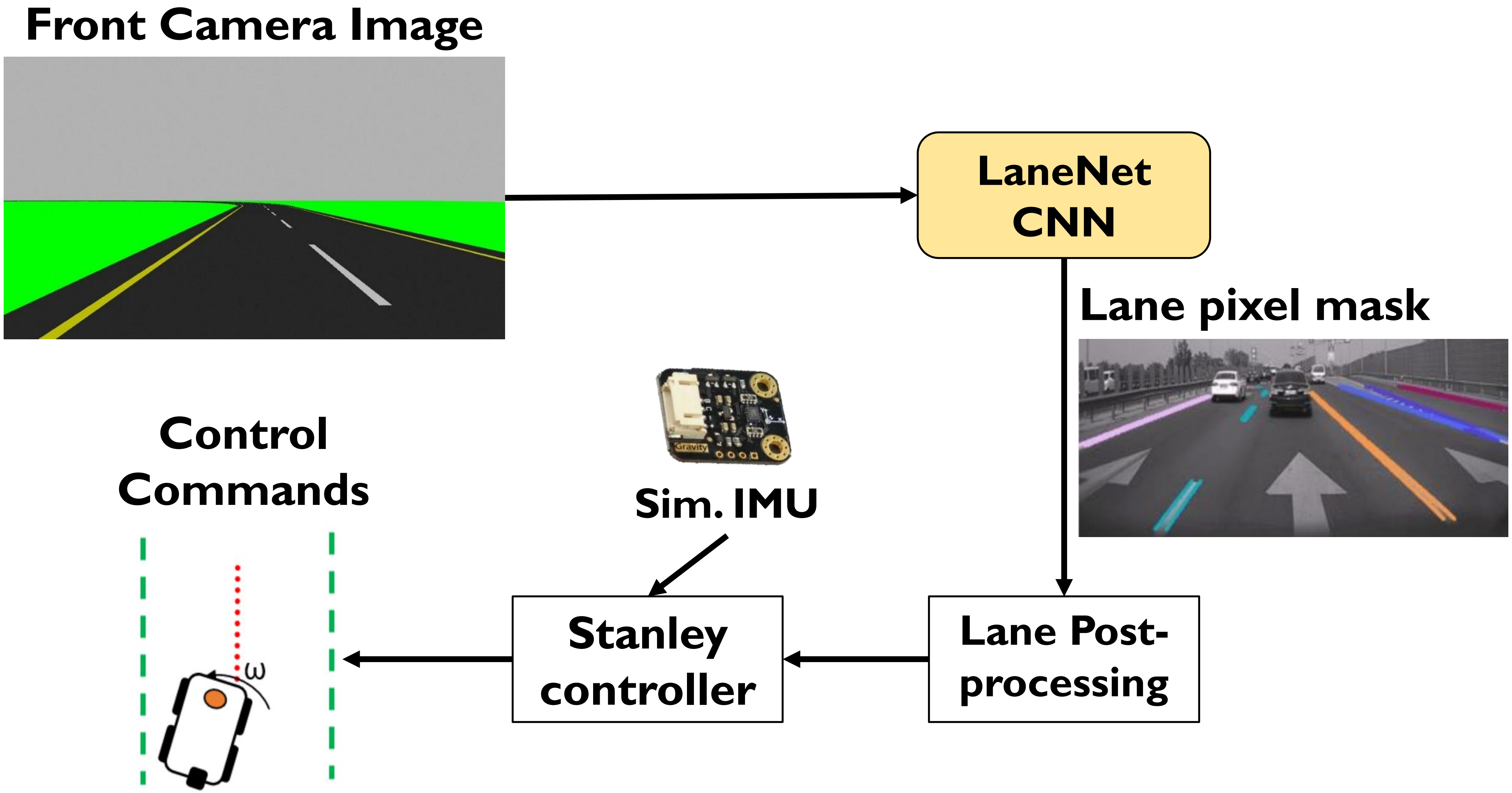
Acc. Sensor

Sensor Fusion

Controller

Velocity Commands

Polaris-GEM Pipeline



Front Camera Image

LaneNet CNN

Lane pixel mask

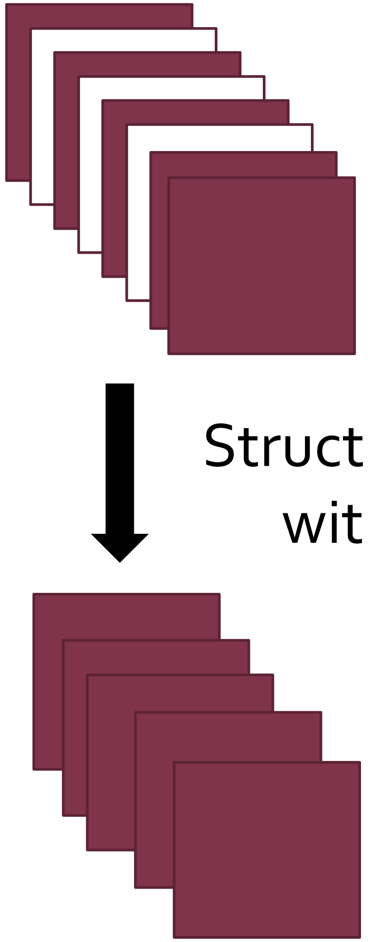
Lane Post-processing

Stanley controller

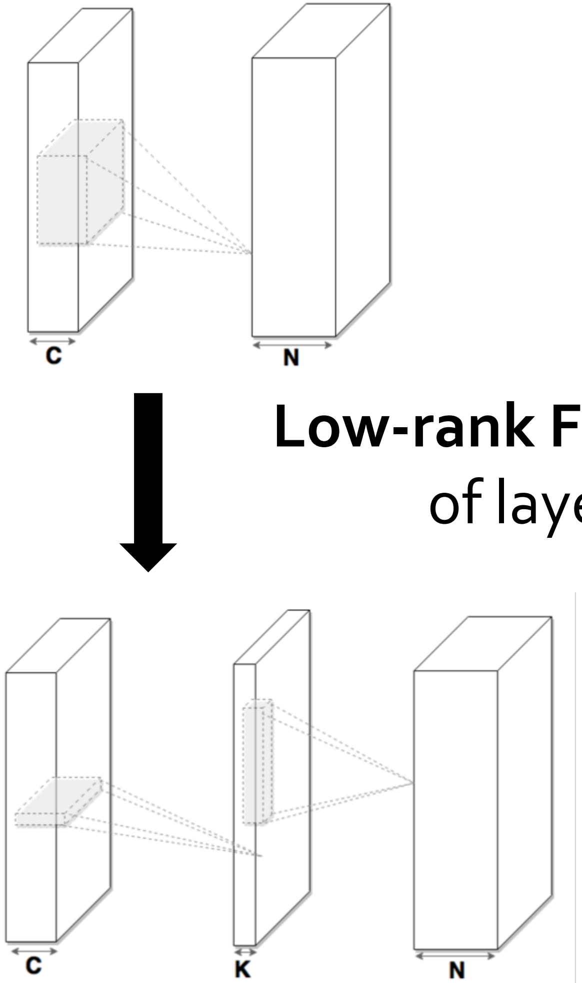
Sim. IMU

Control Commands

Evaluation Setup: Approximations



Structured Pruning
with **LRW** [1]

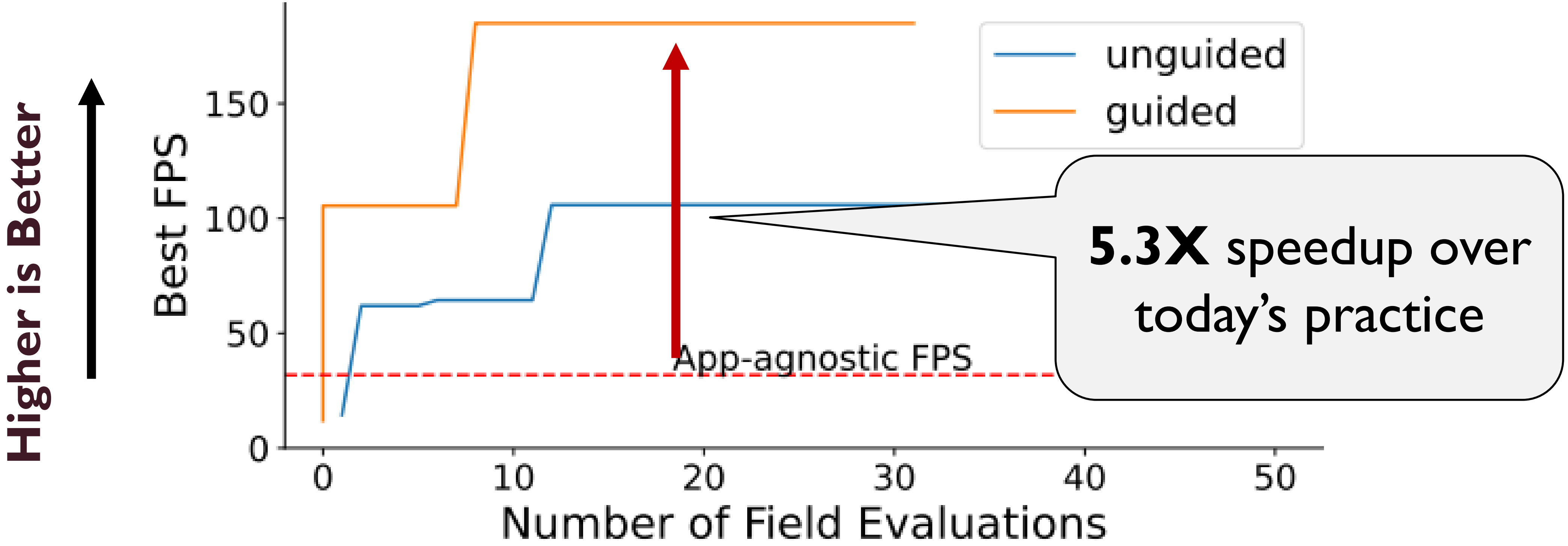


Low-rank Factorization [2]
of layer weights

[1] Renda, Alex, Jonathan Frankle, and Michael Carbin. "Comparing Rewinding and Fine-tuning in Neural Network Pruning." International Conference on Learning Representations. 2019
[2] Tai, C., Xiao, T., Zhang, Y., Wang, X., et al. Convolutional neural networks with low-rank regularization. International Conference on Learning Representations (ICLR), 2016

Optimization Results – Terrasentia

Baseline: **application-agnostic** approximation – retain accuracy

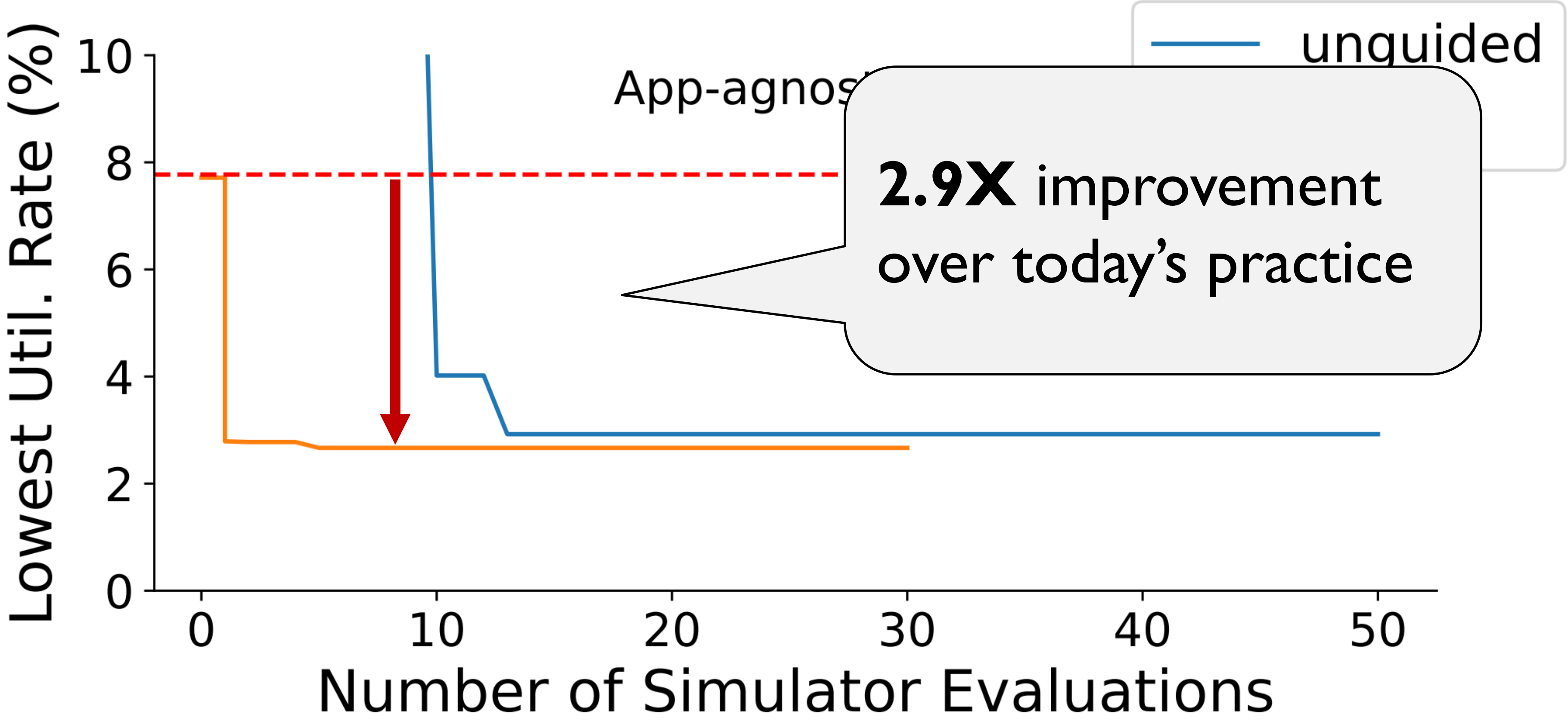


5.3X speedup over today's practice

Optimization Results – Polaris-GEM

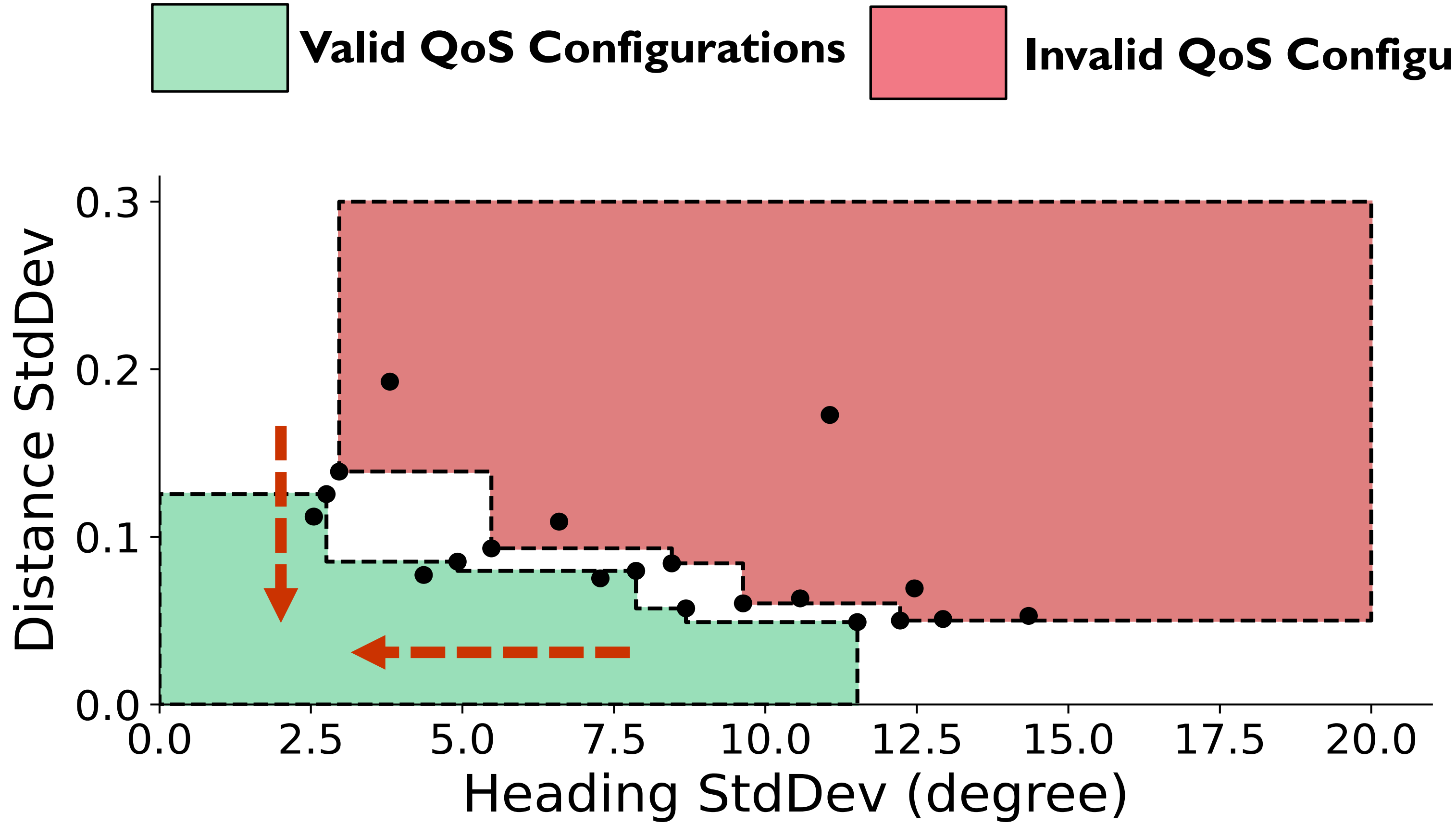
Baseline: application-agnostic approximation – retain accuracy

Lower is Better



2.9X improvement over today's practice

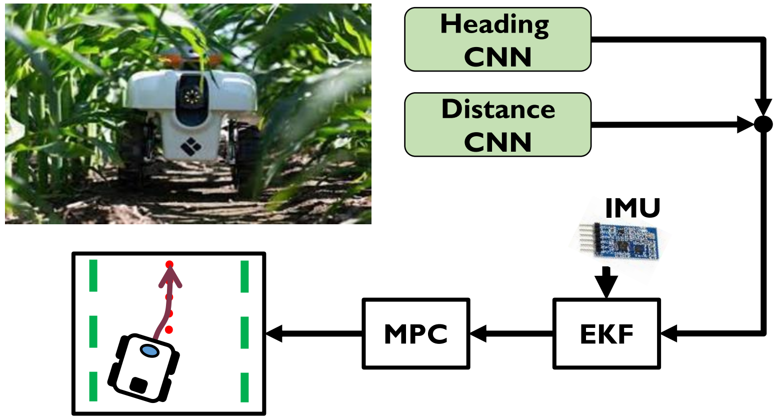
Error Calibration Results – TerraSentia



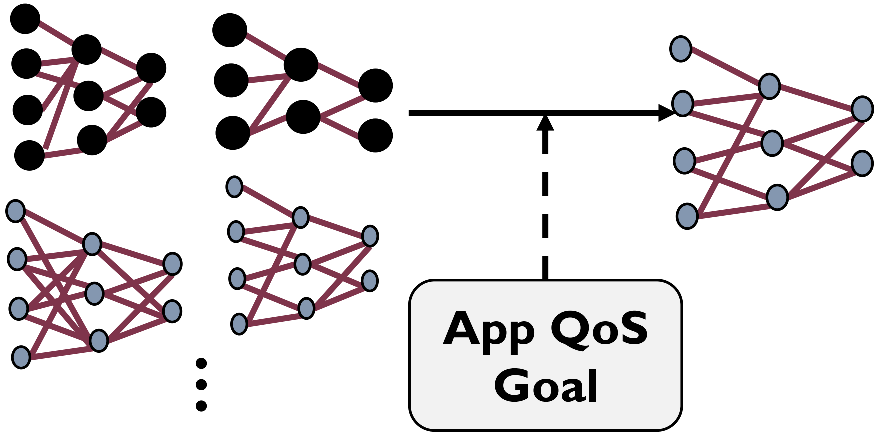
✓ **ApproxCaliper models error interactions across NNs**

ApproxCaliper Takeaways

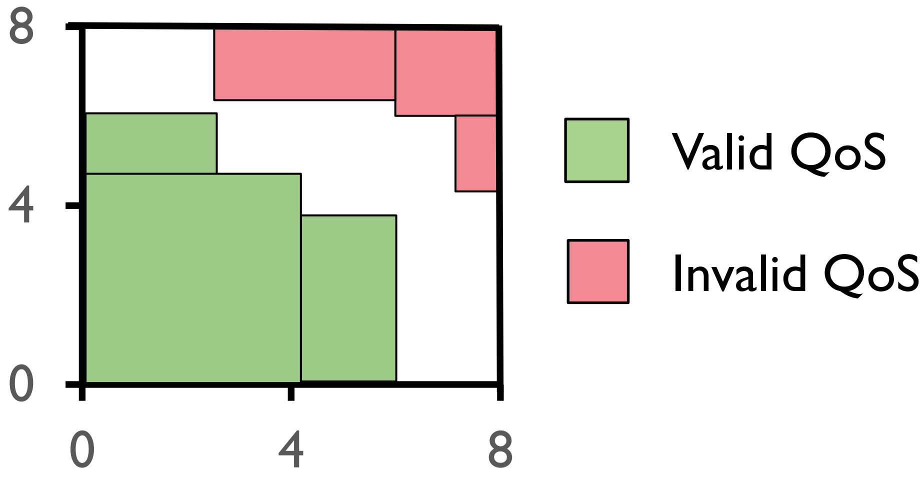
Complex ML Pipelines



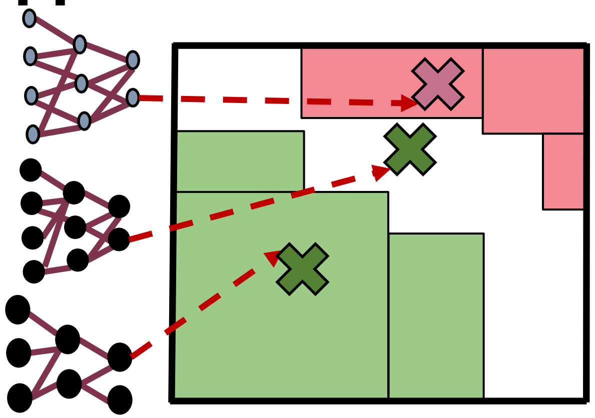
Aggressive Accuracy Optimization



Identify Acceptable Accuracy Budgets



Guided Approximation Tuning



https://github.com/uiuc-arc/approxcaliper