UFMG

Low-cost Quantization Through Abstract Interpretation

Compilers Laboratory

Rafael Sumitani
rafael.sumitani@gmail.com

Motivation

Motivation



Motivation

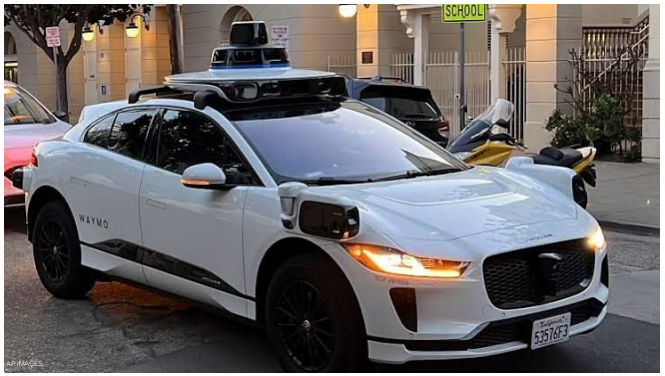




Motivation







Motivation

DCC

What do they have in common?

Motivation

DCC

What do they have in common?

- Resource constrained

Motivation

DCC

What do they have in common?

- Resource constrained
- Latency-sensitive

Motivation

DCC

What do they have in common?

- Resource constrained
- Latency-sensitive
- Neural networks!

Motivation

DCC

What do they have in common?

- Resource constrained
- Latency-sensitive
- Neural networks! => weights

Motivation

DCC

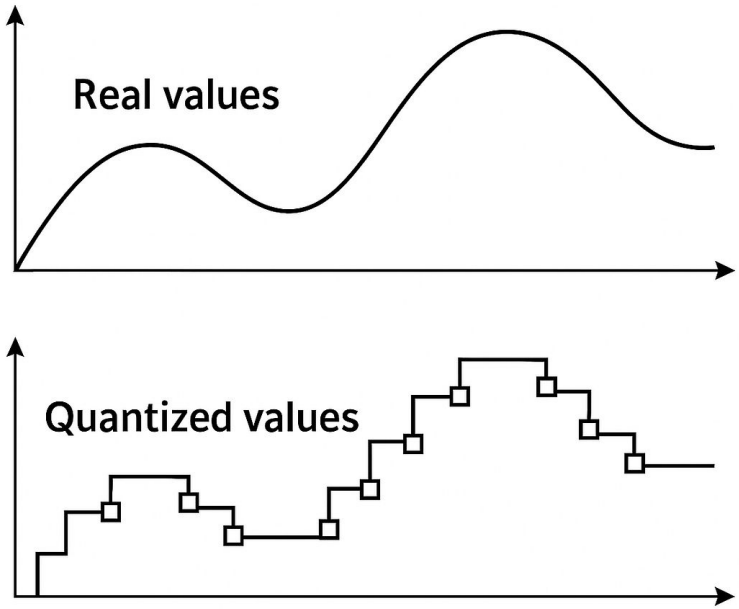
How to optimize this?

Motivation

How to optimize this?  Quantization!

Motivation

How to optimize this? ➡ Quantization!



Quantization

DCC

$Q(r) = ?$

Quantization

$Q(r) = int(r/S)$

Quantization

DCC

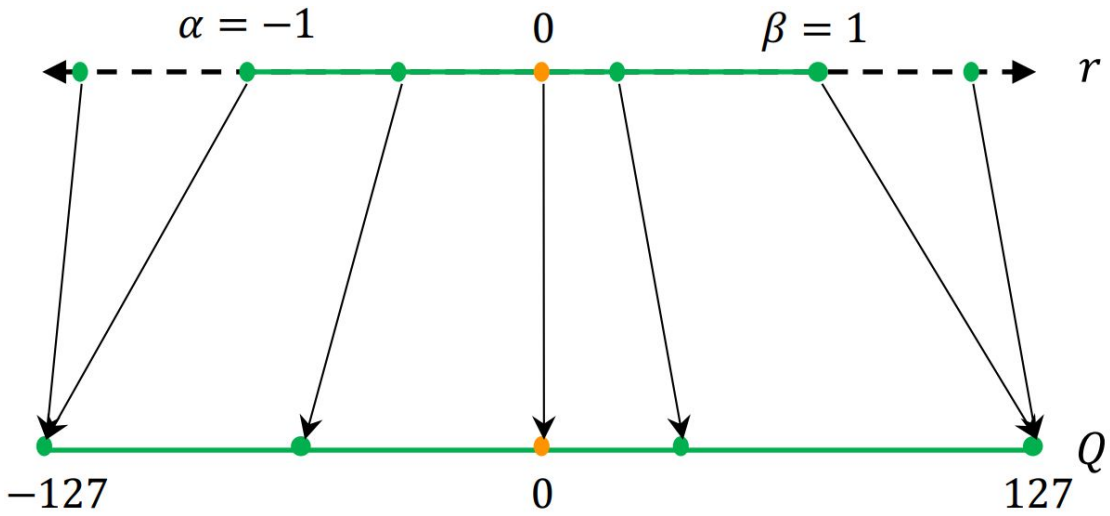
$Q(r) = int(r/S)$

S = **scale**

Quantization

Q(r) = int(r/S)

S = **scale**



Quantization

DCC

$Q(r) = int(r/S)$

S = **scale**

Quantization

$Q(r) = \text{int}(r/S) - Z$

S = **scale**

Quantization

$Q(r) = \text{int}(r/S) - Z$

S = **scale**

Z = **offset**

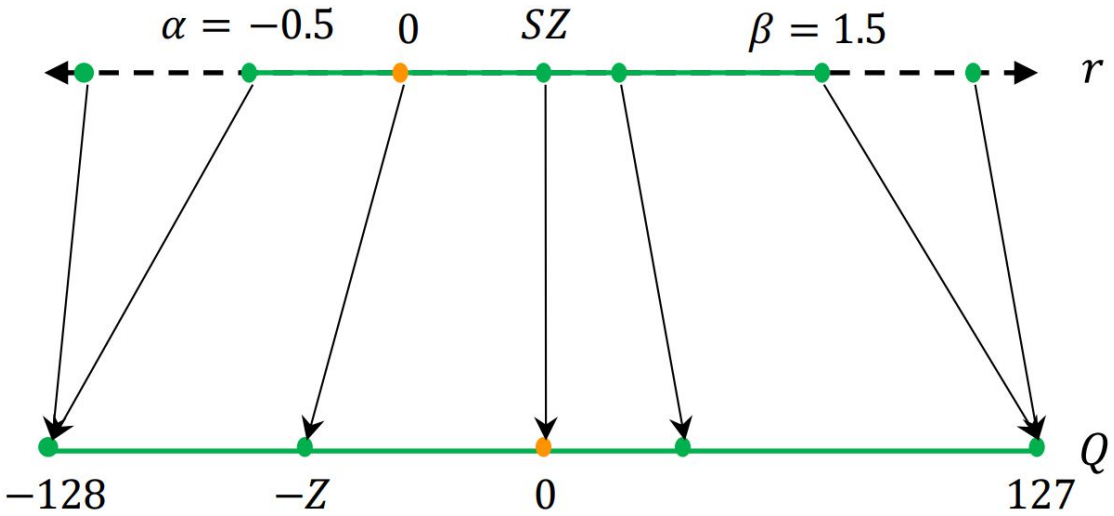
Quantization

DCC

Q(r) = int(r/S) − Z

S = **scale**

Z = **offset**



Quantization

DCC

Quantization

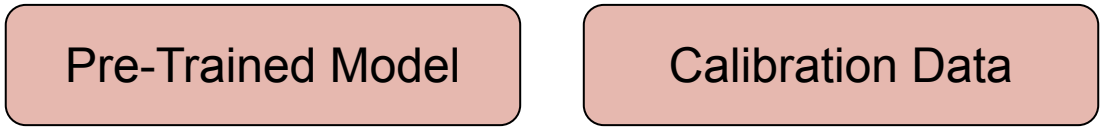
DCC

Post-Training Quantization (PTQ)

Quantization

DCC

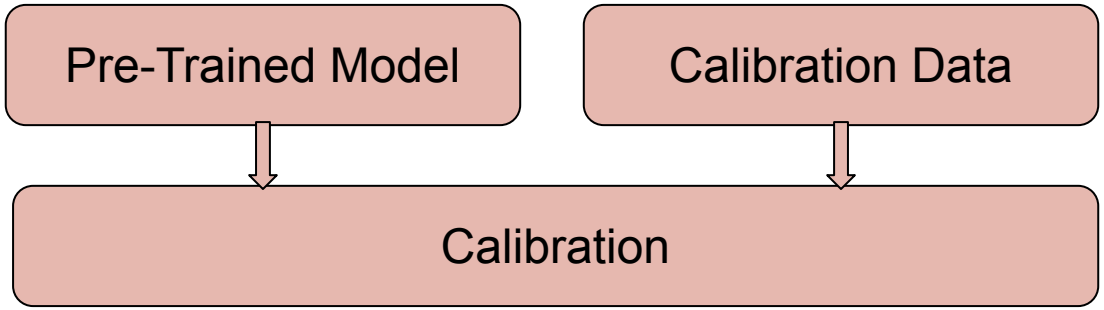
Post-Training Quantization (PTQ)



Quantization

DCC

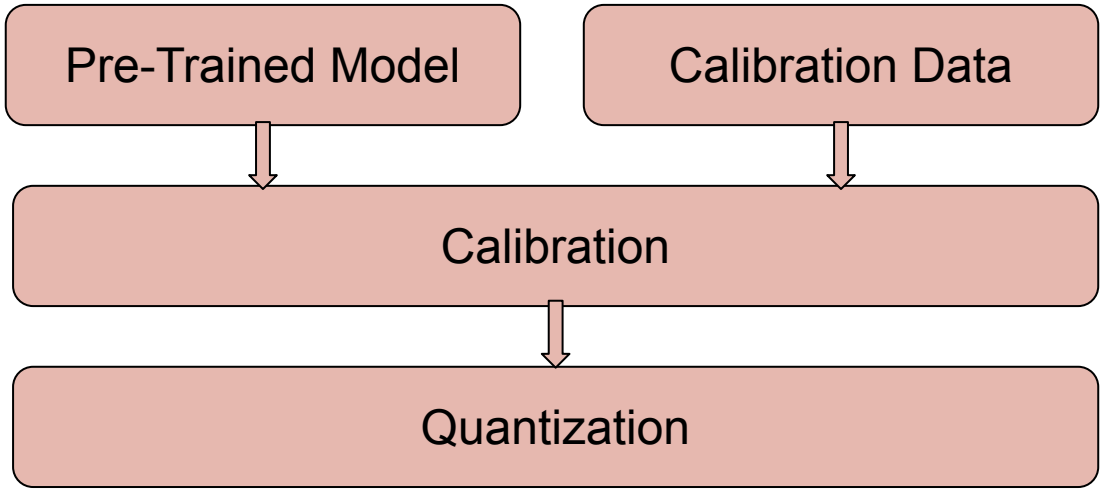
Post-Training Quantization (PTQ)



Quantization

DCC

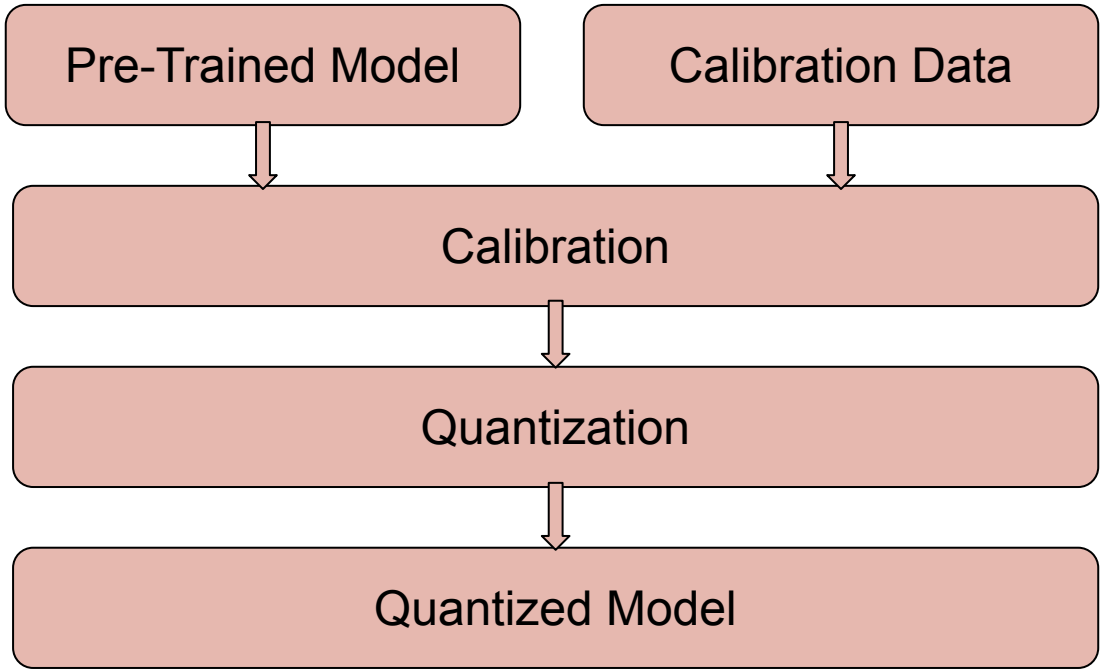
Post-Training Quantization (PTQ)



Quantization

DCC

Post-Training Quantization (PTQ)



Problem

DCC

Problem

Calibration may be expensive!

Problem

DCC

Calibration may be expensive!

- Can we quantize without running inference?

Problem

DCC

Calibration may be expensive!

- Can we quantize without running inference?
- Can we reuse information from previous runs?

Proposed Solution

Proposed Solution

DCC

Static analysis based on abstract interpretation

Proposed Solution

DCC

Static analysis based on abstract interpretation: Vishap

Proposed Solution

DCC

Static analysis based on abstract interpretation: Vishap

Propagate statistical information about tensors

Proposed Solution

DCC

Static analysis based on abstract interpretation: Vishap

Propagate statistical information about tensors **without running inference**

Proposed Solution

DCC

Static analysis based on abstract interpretation: Vishap

Propagate statistical information about tensors **without running inference**

Reuse profiles for modified networks

Abstract Domain

DCC

Abstract Domain

DCC

4-tuple $(\alpha, \beta, \mu, \sigma^2)$

Abstract Domain

DCC

4-tuple $(\alpha, \beta, \mu, \sigma^2)$

- α = minimum value

Abstract Domain

DCC

4-tuple $(\alpha, \beta, \mu, \sigma^2)$

- α = minimum value
- β = maximum value

Abstract Domain

DCC

4-tuple $(\alpha, \beta, \mu, \sigma^2)$

- α = minimum value
- β = maximum value
- μ = average value

Abstract Domain

DCC

4-tuple $(\alpha, \beta, \mu, \sigma^2)$

- α = minimum value
- β = maximum value
- μ = average value
- σ^2 = variance

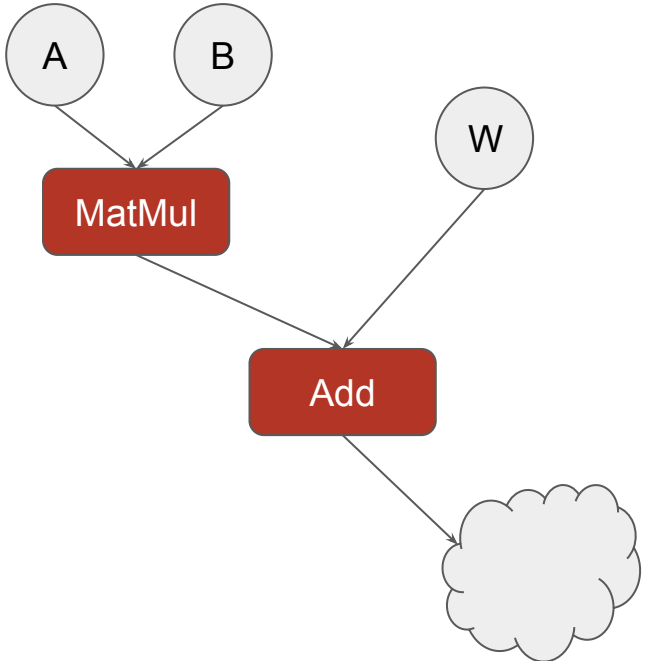
Transfer Function

Transfer Function

Applied over nodes in the computational graph

Transfer Function

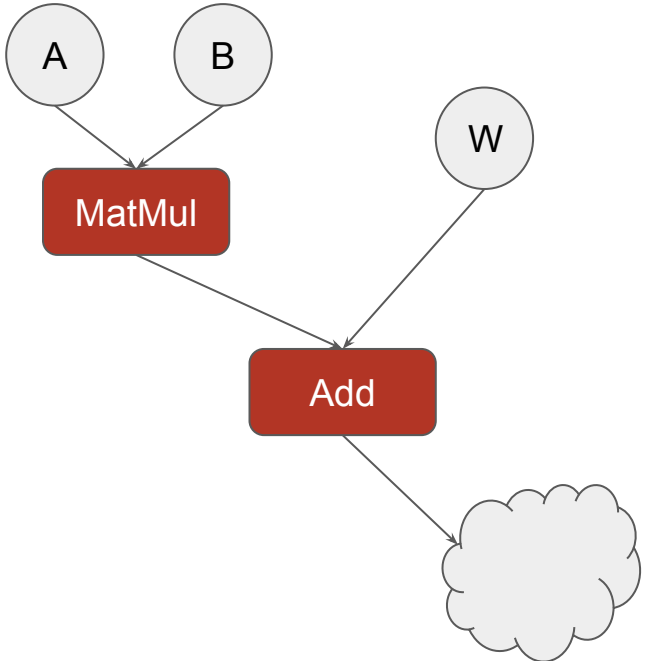
Applied over nodes in the **computational graph**



Transfer Function

Applied over nodes in the **computational graph**

Specialized per node kind



Transfer Function

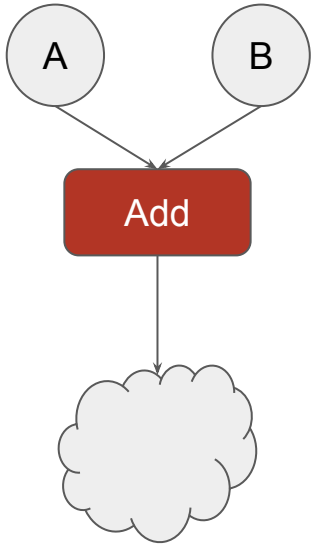
DCC

- Example: element-wise addition

Transfer Function

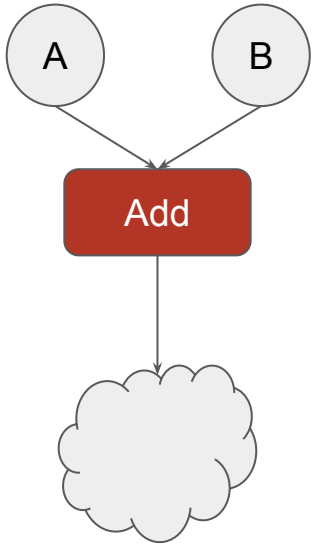
DCC

- Example: element-wise addition



Transfer Function

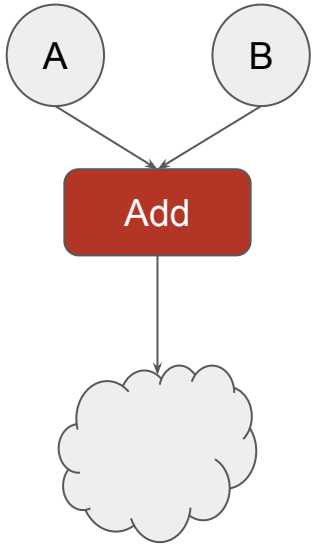
- Example: element-wise addition



$$f_{Add}[\alpha] = f_A[\alpha] + f_B[\alpha]$$

Transfer Function

- Example: element-wise addition

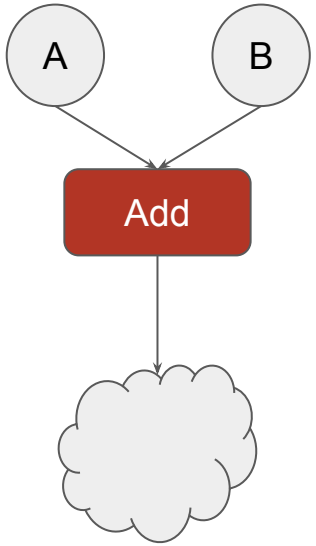


$$f_{Add}[\alpha] = f_A[\alpha] + f_B[\alpha]$$

$$f_{Add}[\beta] = f_A[\beta] + f_B[\beta]$$

Transfer Function

- Example: element-wise addition



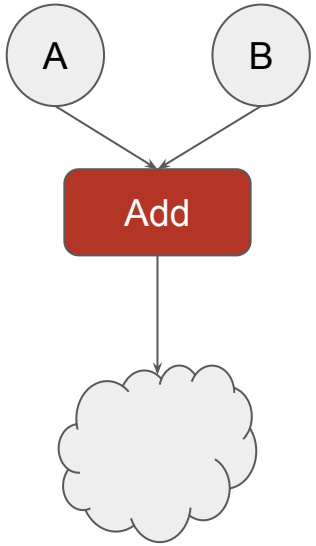
$$f_{Add}[\alpha] = f_A[\alpha] + f_B[\alpha]$$

$$f_{Add}[\beta] = f_A[\beta] + f_B[\beta]$$

$$f_{Add}[\mu] = f_A[\mu] + f_B[\mu]$$

Transfer Function

- Example: element-wise addition



$$f_{Add}[\alpha] = f_A[\alpha] + f_B[\alpha]$$

$$f_{Add}[\beta] = f_A[\beta] + f_B[\beta]$$

$$f_{Add}[\mu] = f_A[\mu] + f_B[\mu]$$

$$f_{Add}[\sigma^2] = f_A[\sigma^2] + f_B[\sigma^2]$$

Vishap

DCC

Vishap

DCC

Implemented in MLIR

Vishap

DCC

Implemented in MLIR

Quantization with ONNX Runtime library

Vishap

DCC

Implemented in MLIR

Quantization with ONNX Runtime library

Inject estimated clipping ranges

Vishap

DCC

Implemented in MLIR

Quantization with ONNX Runtime library

Inject estimated clipping ranges

Heuristic: [$\mu-\lambda*\sigma$, $\mu+\lambda*\sigma$]

Evaluation

DCC

Evaluation

RQ1

RQ2

Evaluation

RQ1: How precise is our solution vs. calibration?

RQ2

Evaluation

DCC

RQ1: How precise is our solution vs. calibration?

RQ2: How fast does our solution run vs. calibration?

Evaluation

DCC

RQ1: How precise is our solution vs. calibration?

RQ2: How fast does our solution run vs. calibration?

Benchmarks

Evaluation

DCC

RQ1: How precise is our solution vs. calibration?

RQ2: How fast does our solution run vs. calibration?

Benchmarks: Bennu

Evaluation

DCC

RQ1: How precise is our solution vs. calibration?

RQ2: How fast does our solution run vs. calibration?

Benchmarks: Bennu

- 20 ONNX models

Evaluation

DCC

RQ1: How precise is our solution vs. calibration?

RQ2: How fast does our solution run vs. calibration?

Benchmarks: Bennu

- 20 ONNX models
- All classification

Evaluation

DCC

Baselines

Evaluation

Baselines: ONNX Runtime calibration

Evaluation

DCC

Baselines: ONNX Runtime calibration + random quantization

Evaluation

DCC

Baselines: ONNX Runtime calibration + random quantization

Vishap variants

Evaluation

DCC

Baselines: ONNX Runtime calibration + random quantization

Vishap variants:

- Single input

Evaluation

DCC

Baselines: ONNX Runtime calibration + random quantization

Vishap variants:

- Single input
- 100 aggregated inputs

Evaluation

DCC

Baselines: ONNX Runtime calibration + random quantization

Vishap variants:

- Single input
- 100 aggregated inputs
- Mixed

Evaluation

DCC

Baselines: ONNX Runtime calibration + random quantization

Vishap variants:

- Single input
- 100 aggregated inputs
- Mixed (25%, 50%, 75%)

Evaluation: RQ1

How precise is our solution vs. calibration?

Evaluation: RQ1

Metric

Evaluation: RQ1

DCC

Metric: top-1 agreement

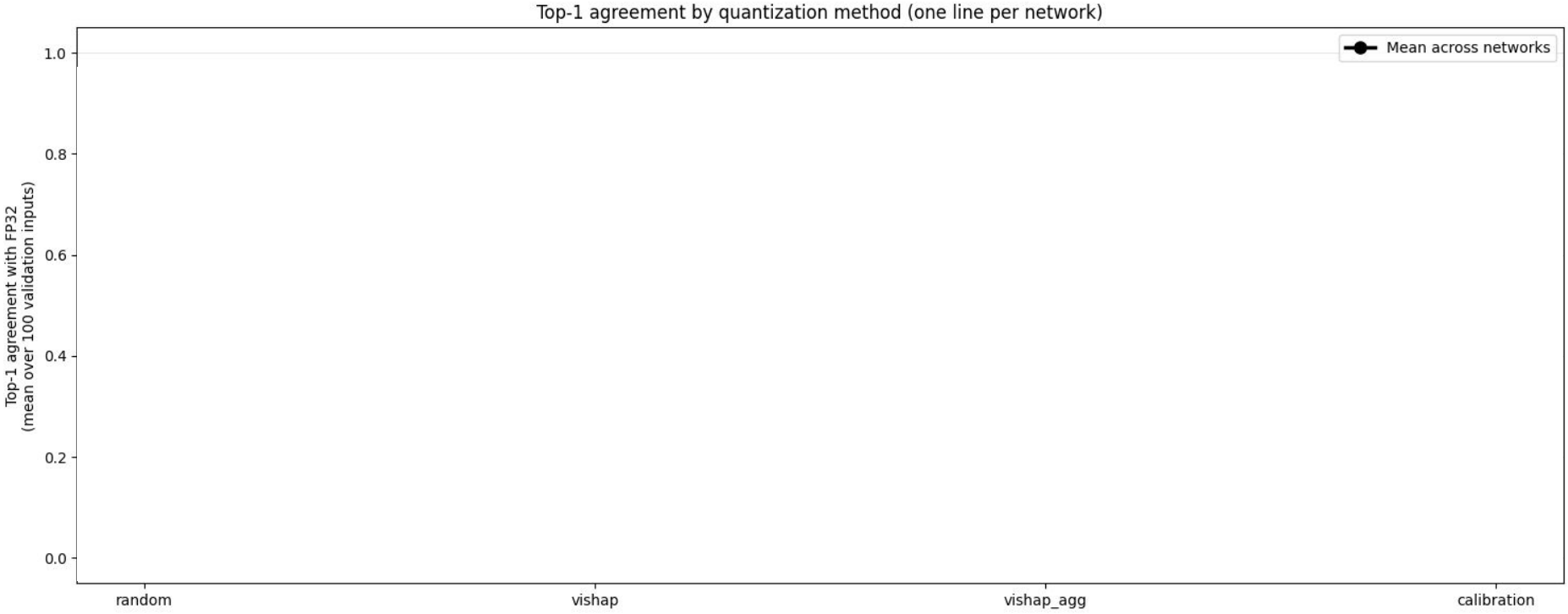
Evaluation: RQ1

DCC

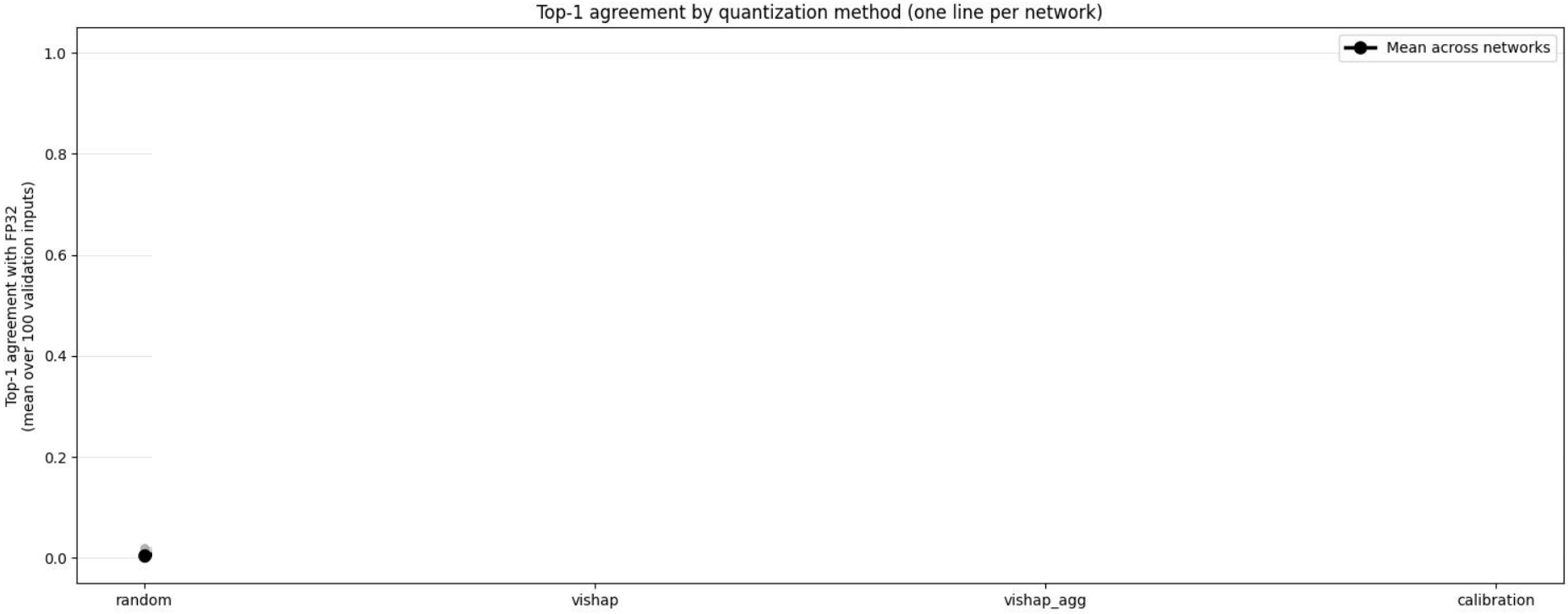
Metric: top-1 agreement

100 validation inputs

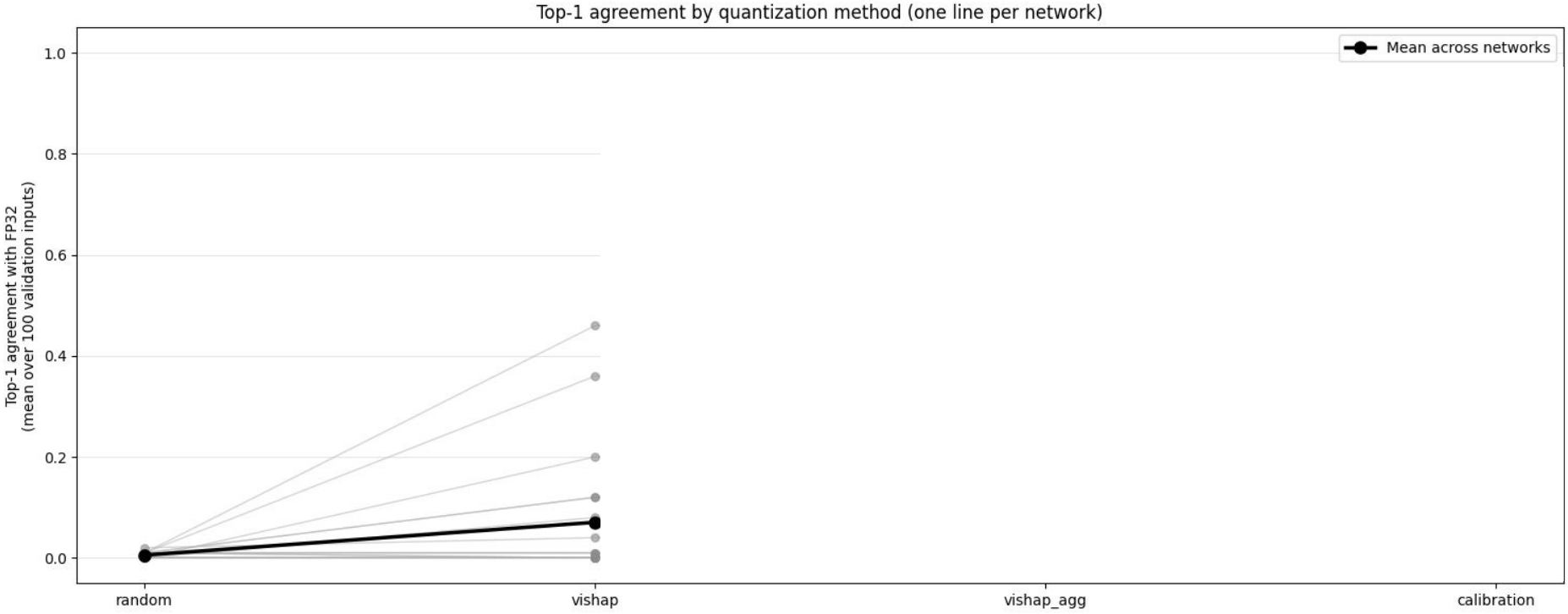
Evaluation: RQ1



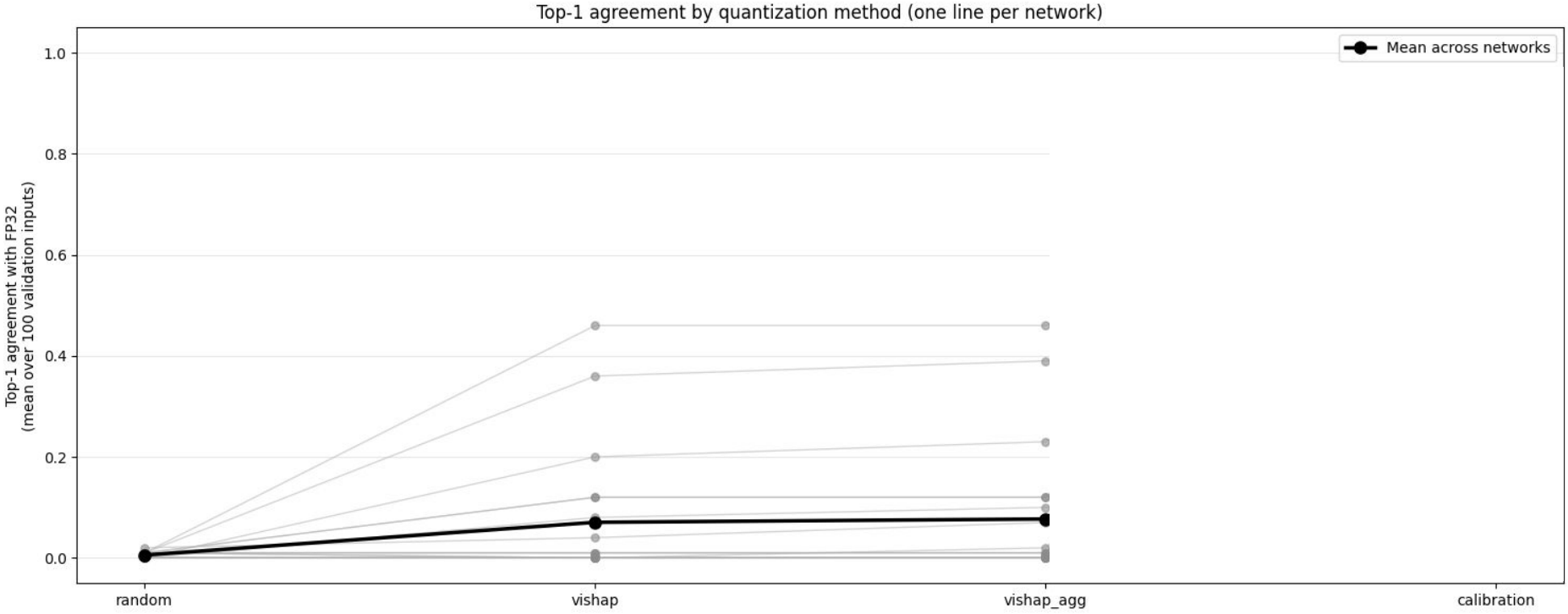
Evaluation: RQ1



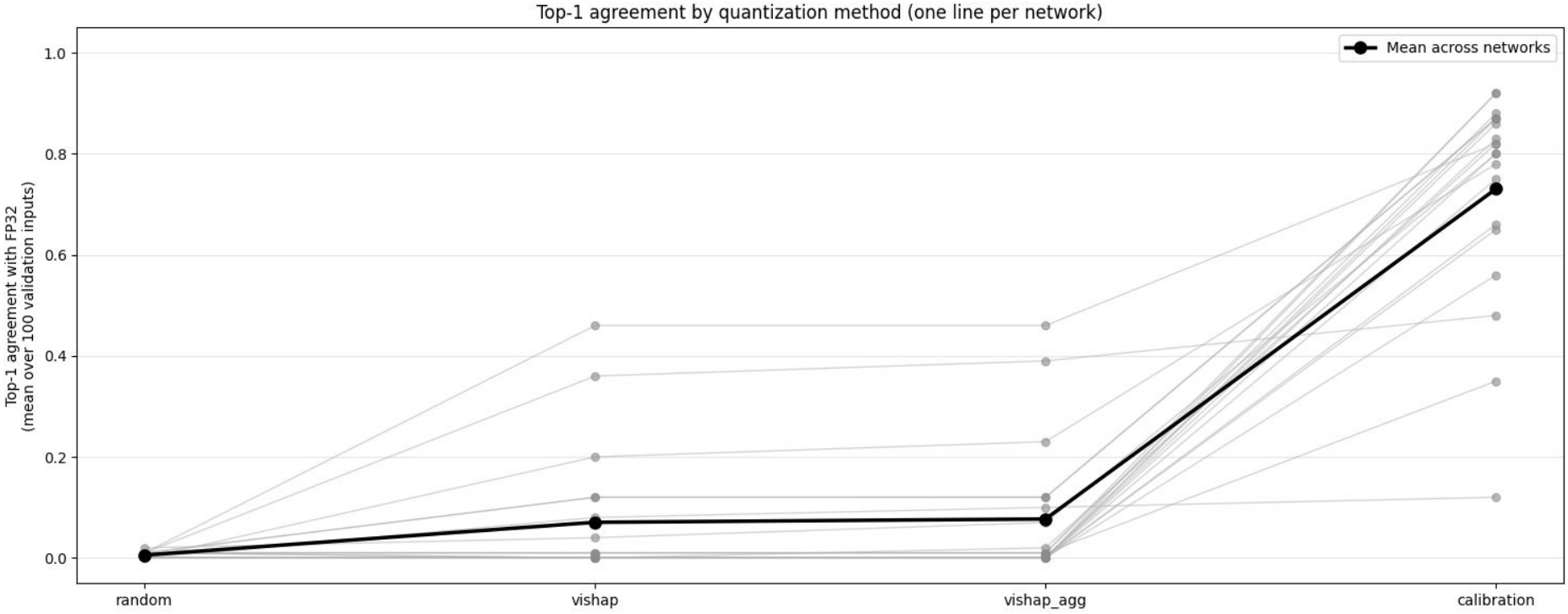
Evaluation: RQ1



Evaluation: RQ1



Evaluation: RQ1



Evaluation: RQ2

DCC

How fast does our solution run vs. calibration?

Evaluation: RQ2

Metric

Evaluation: RQ2

Metric: total quantization time

Evaluation: RQ2

Metric: total quantization time

Calibration

Evaluation: RQ2

DCC

Metric: total quantization time

Calibration: 100 rounds of inference + quantization

Evaluation: RQ2

DCC

Metric: total quantization time

Calibration: 100 rounds of inference + quantization

Vishap

Evaluation: RQ2

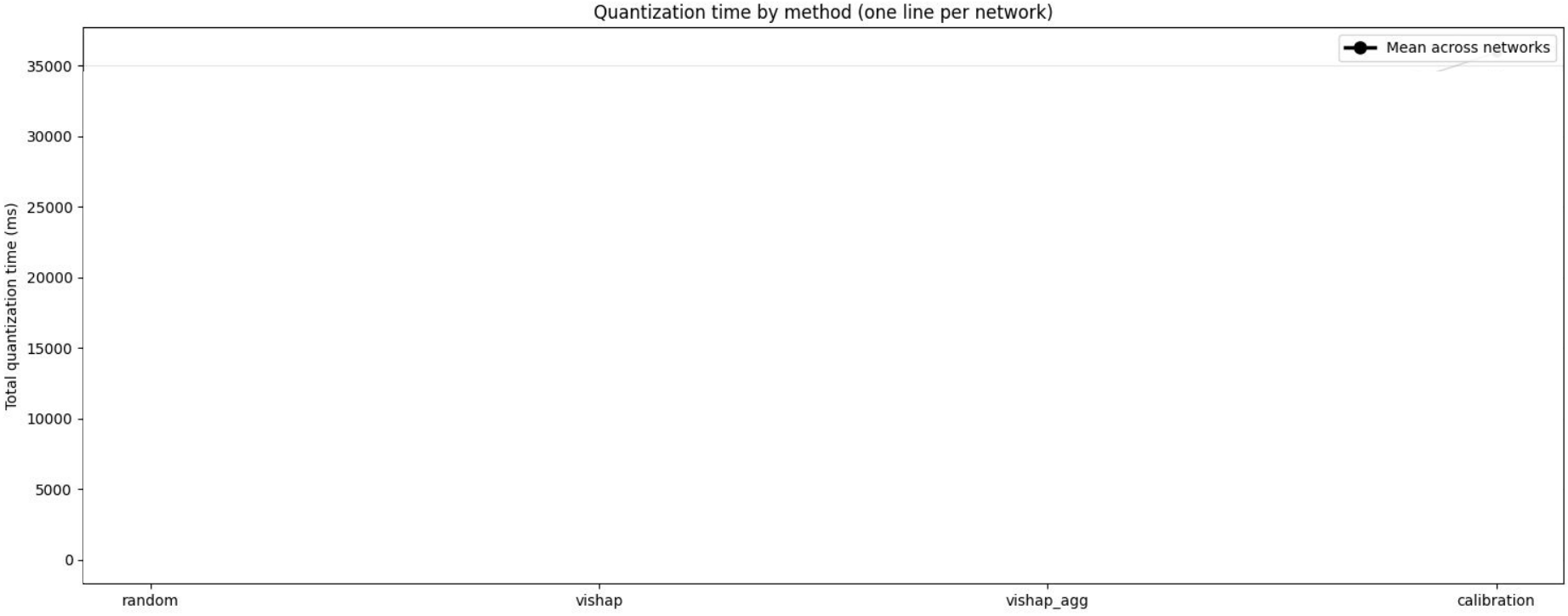
DCC

Metric: total quantization time

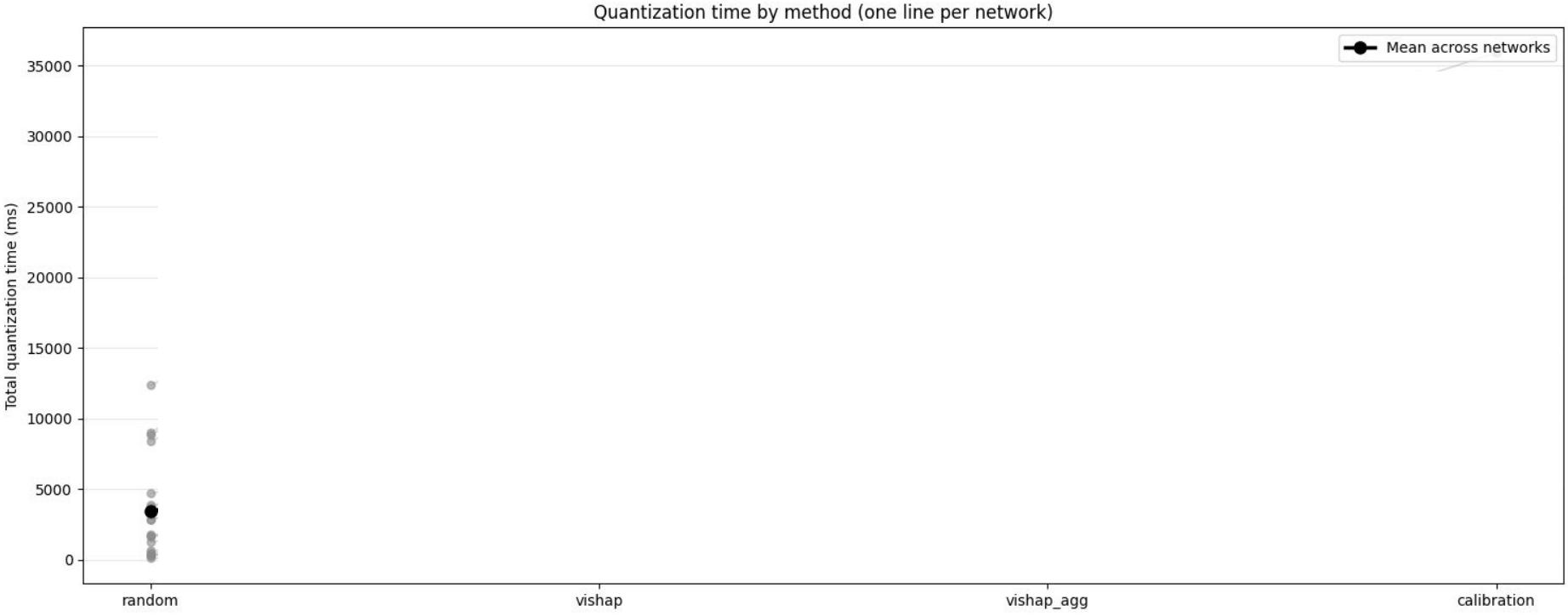
Calibration: 100 rounds of inference + quantization

Vishap: static analysis + graph manipulation

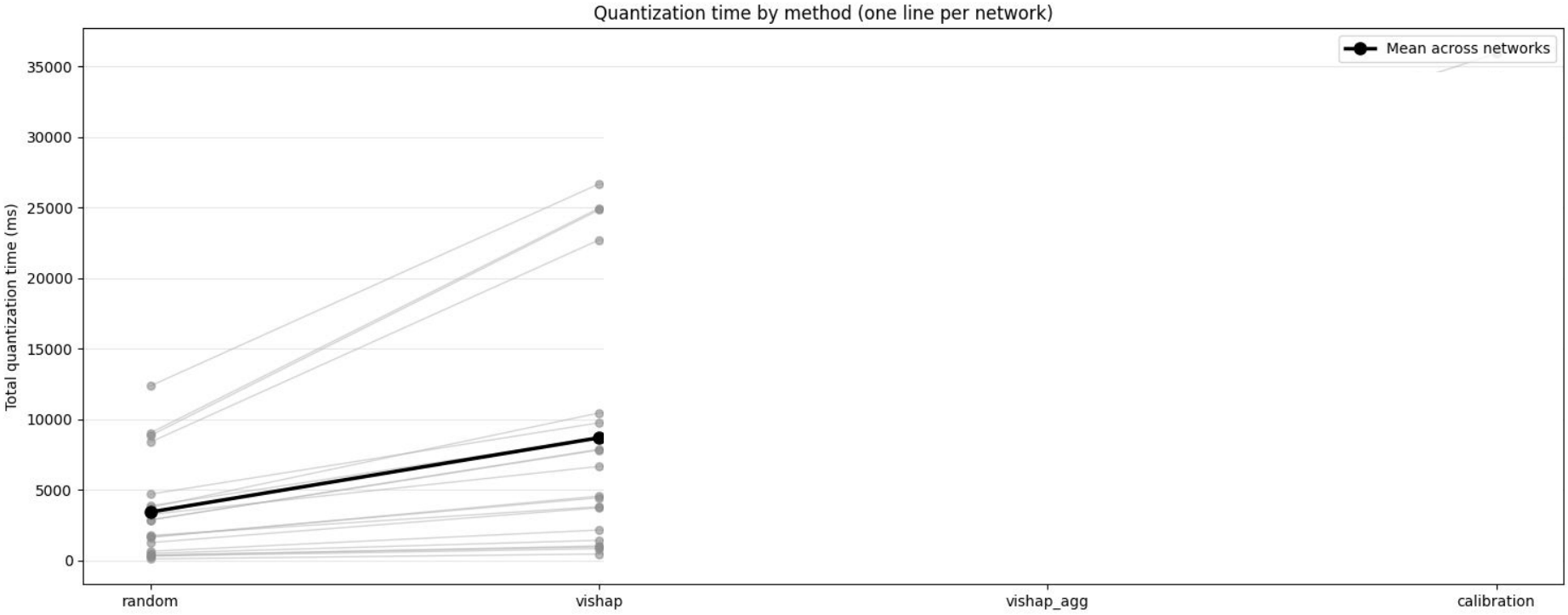
Evaluation: RQ2



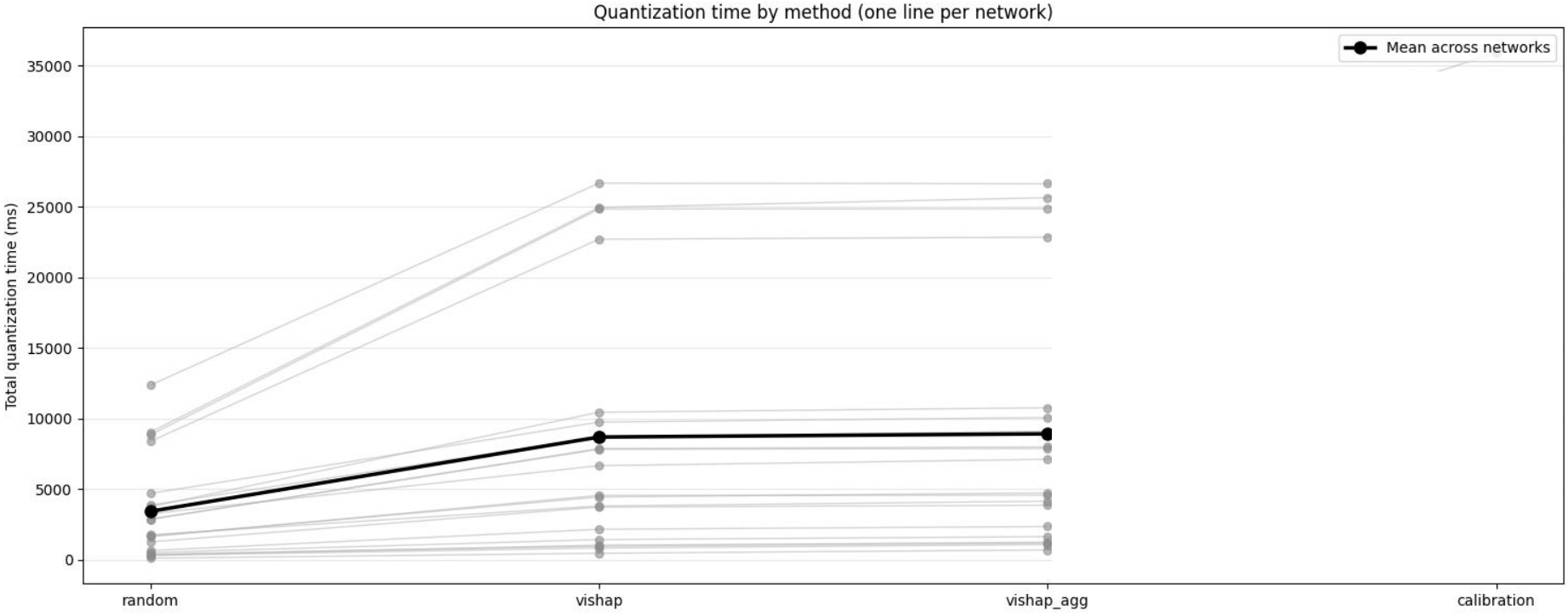
Evaluation: RQ2



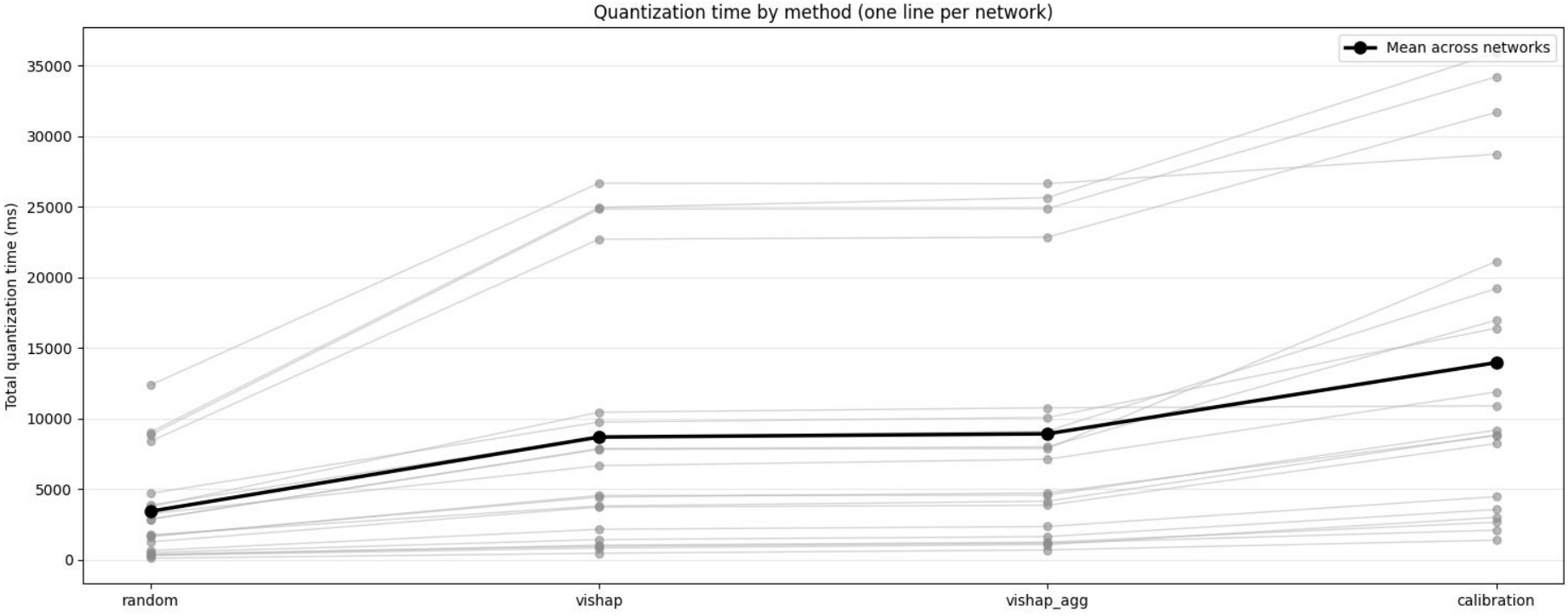
Evaluation: RQ2



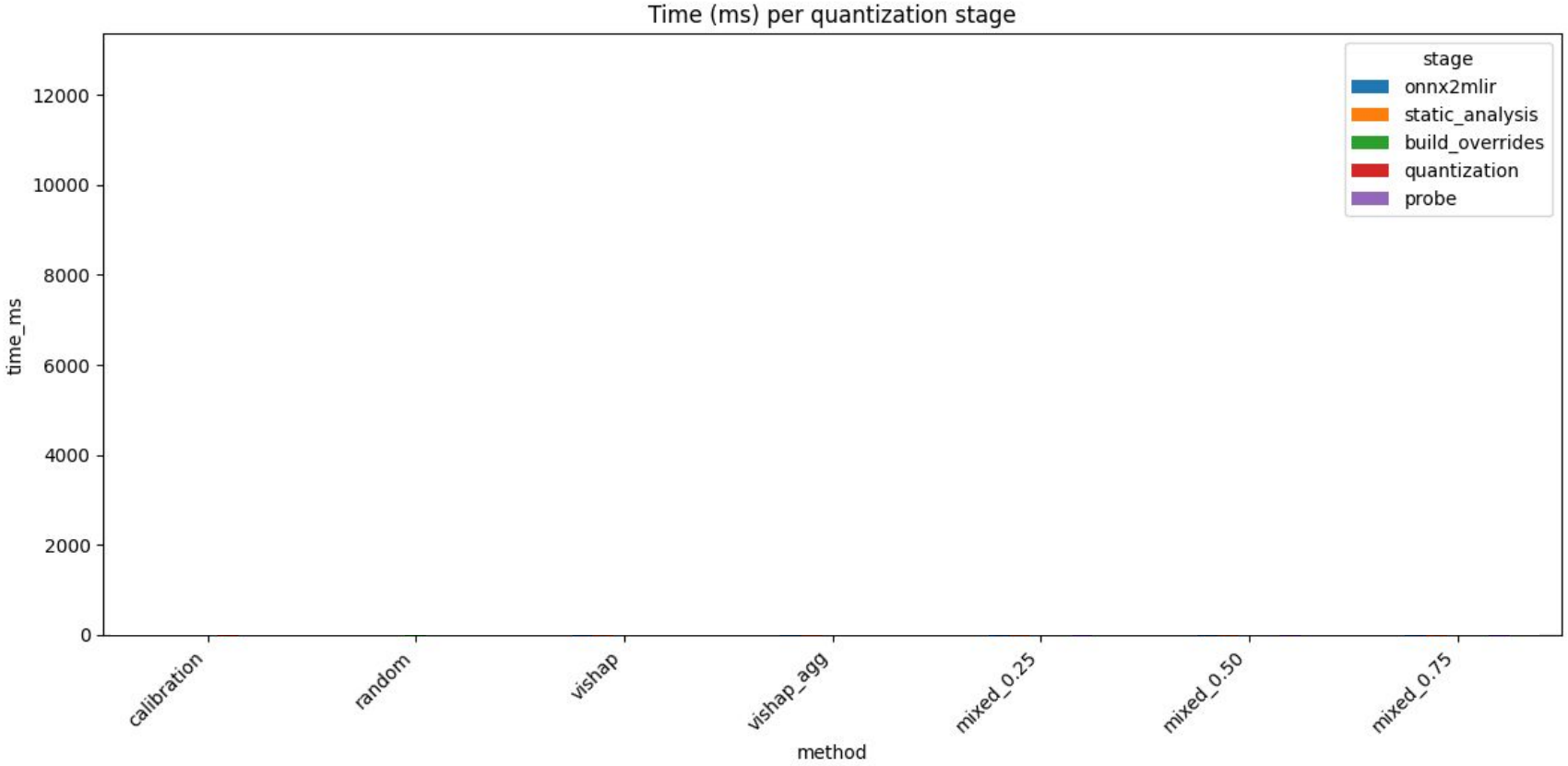
Evaluation: RQ2



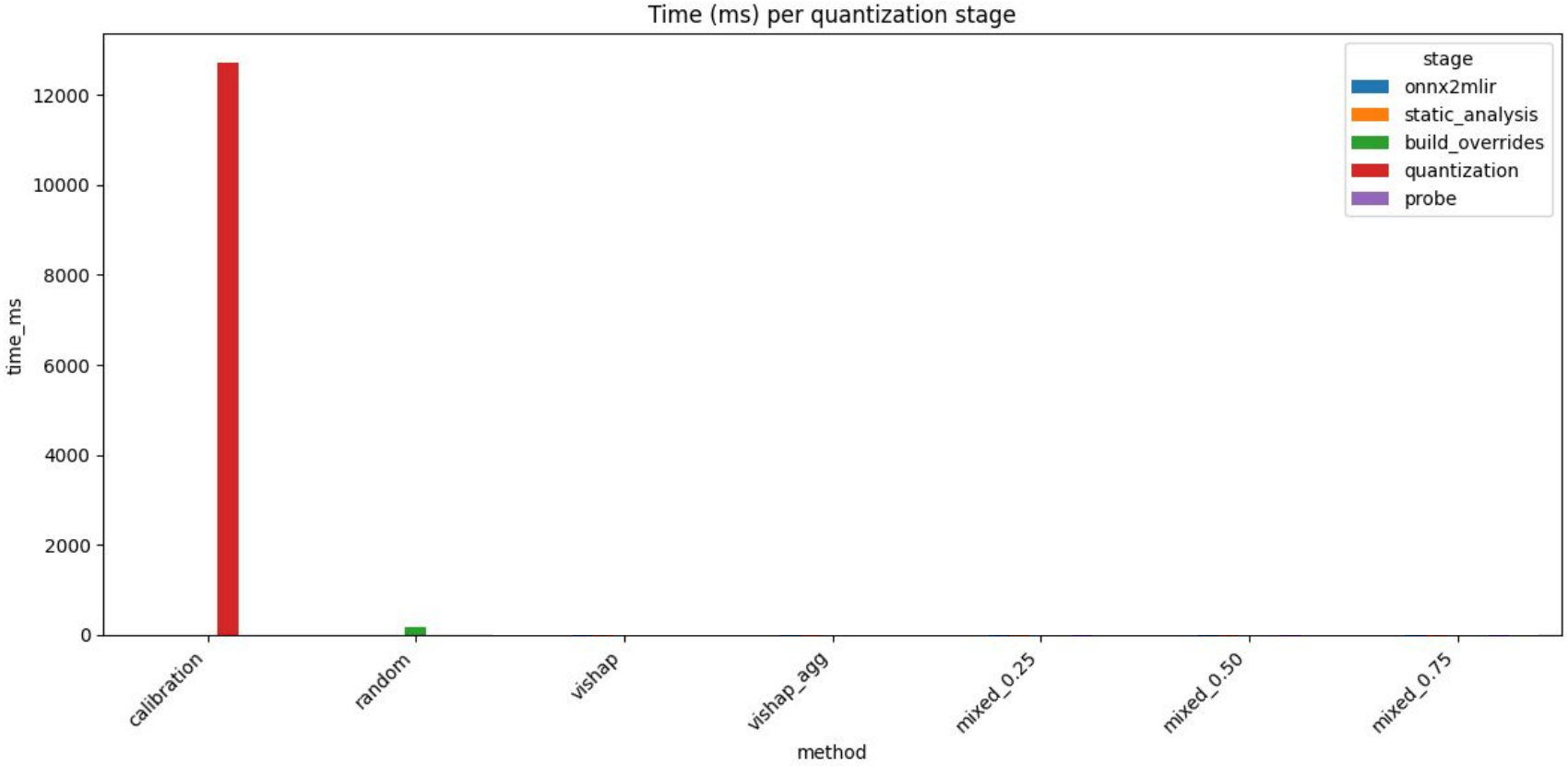
Evaluation: RQ2



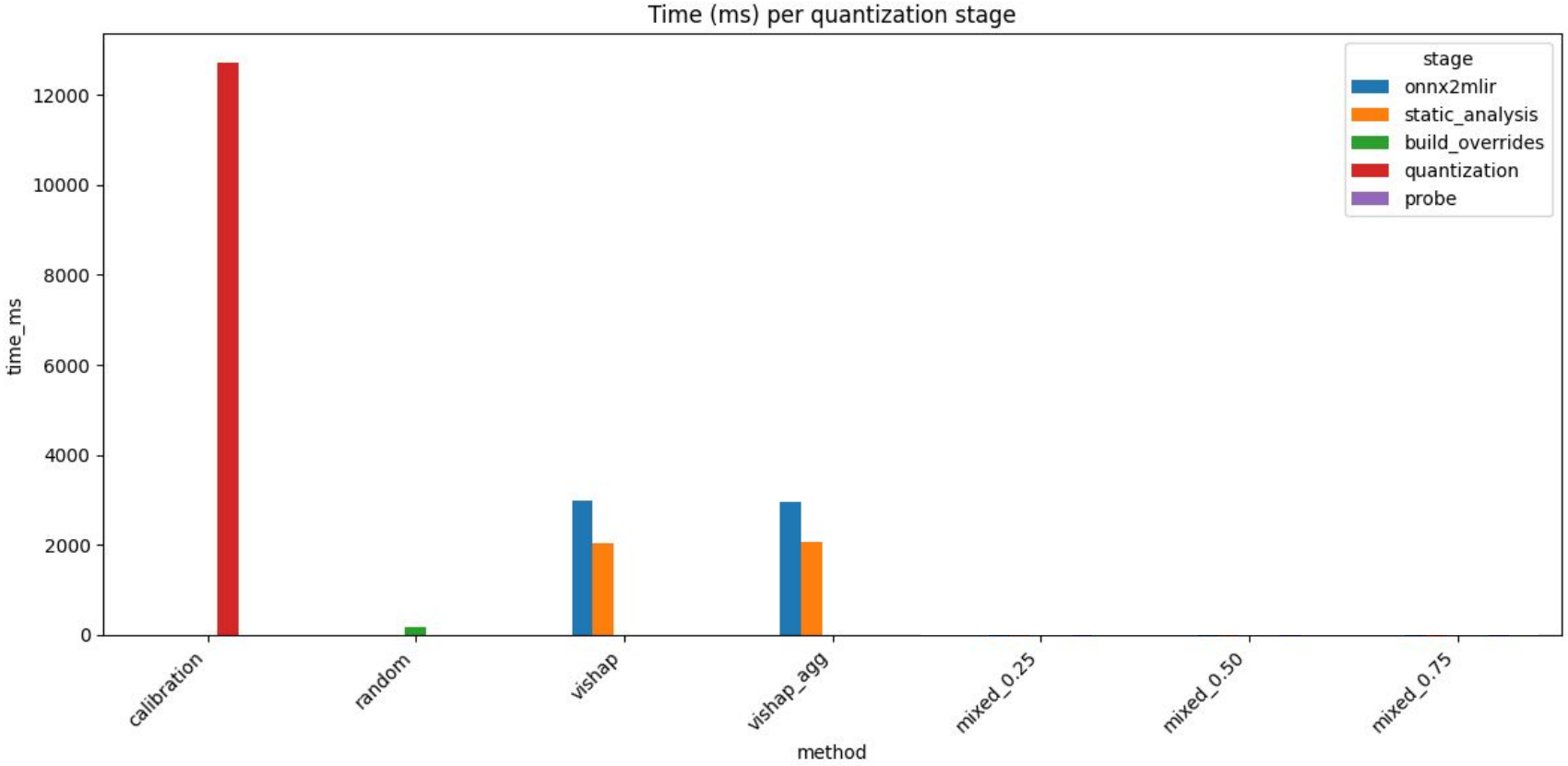
Evaluation: RQ2



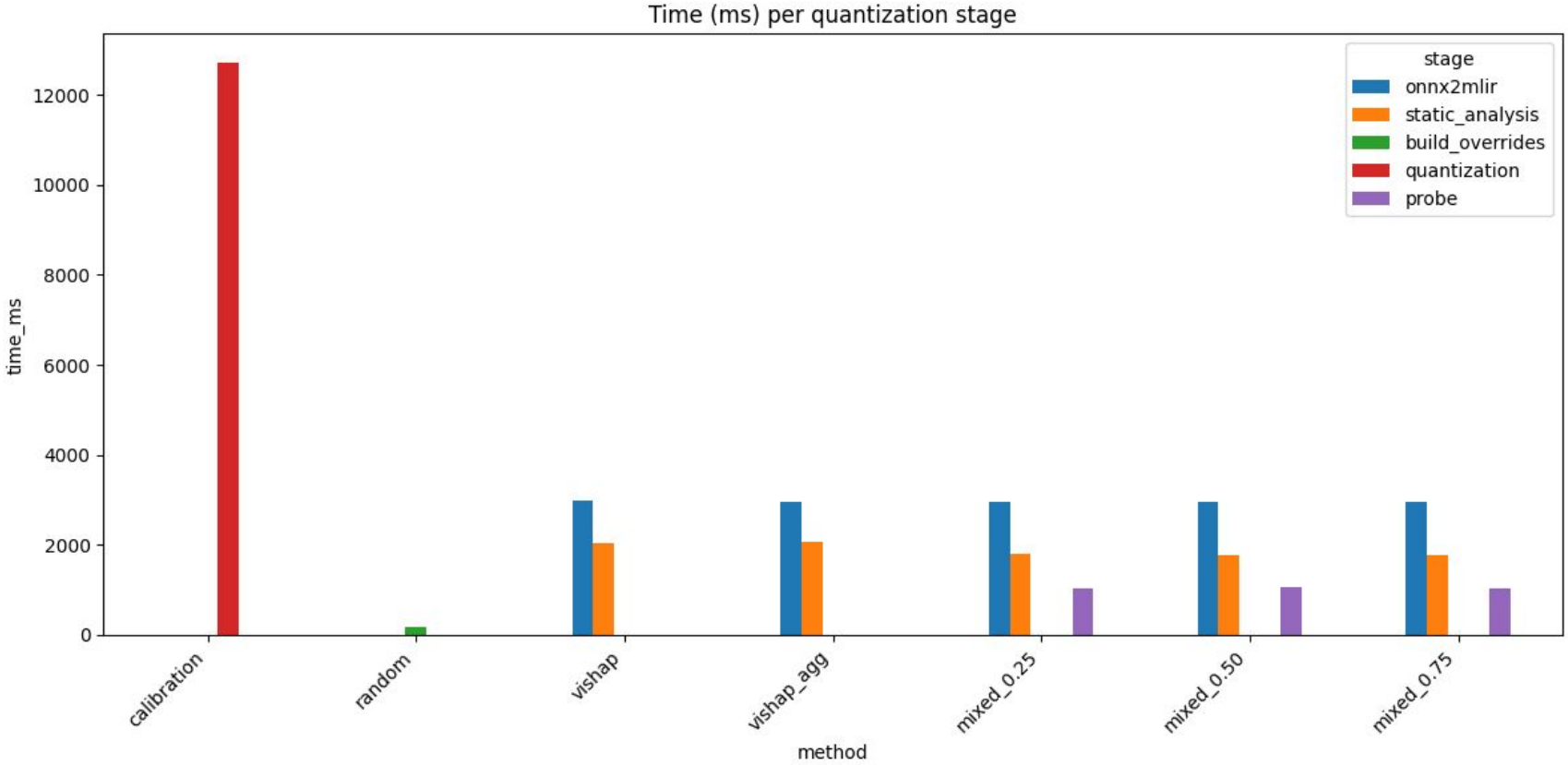
Evaluation: RQ2



Evaluation: RQ2



Evaluation: RQ2



Open Questions + Future Work

Open Questions + Future Work

How can we improve accuracy?

Open Questions + Future Work

DCC

How can we improve accuracy?

- Change the abstract domain?

Open Questions + Future Work

DCC

How can we improve accuracy?

- Change the abstract domain?

Can we improve execution time?

Open Questions + Future Work

DCC

How can we improve accuracy?

- Change the abstract domain?

Can we improve execution time?

- What if everything was in the same infrastructure?

Open Questions + Future Work

DCC

How can we improve accuracy?

- Change the abstract domain?

Can we improve execution time?

- What if everything was in the same infrastructure?

Can we reuse calibration data for a portion of the network?

References & Contact

DCC

Rafael Sumitani

rafael.sumitani@gmail.com

srafael@cadence.com

References

Gholami et al.
A Survey of Quantization Methods for Efficient Neural Network Inference
Low-Power Computer Vision: Improving the Efficiency of Artificial Intelligence, 2022