# From Robustness to Worst-Case Timing Analysis: Reusing Neural Network Abstract Interpretation with DeepWCET

November 4, 2026
34th International Conference on Real-Time Networks and Systems (RTNS 2026), Toulouse, France

Tobias Häberlein, Dominik Penk, Dominik Riedelbauch, Isabella Stilkerich, Peter Wägemann

Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)
Schaeffler Technologies AG & Co. KG



Lehrstuhl für Informatik 4
Systemsoftware



Friedrich-Alexander-Universität
Faculty of Engineering

Supported by the Schaeffler Hub for Advanced Research at FAU (SHARE at FAU)

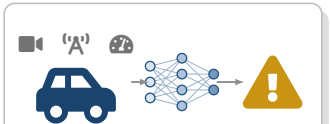
# Neural Networks in Real-Time Systems

## More and more neural networks run on microcontrollers



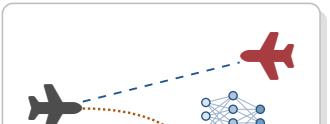
**Temperature calculation**

Kirchgässner et al.: **Estimating Electric Motor Temperatures With Deep Residual Machine Learning**, IEEE TPEL '21



**Fault detection**

Theissler et al.: **Predictive Maintenance Enabled by Machine Learning: Use Cases and Challenges in the Automotive Industry**, RESS '21



**Collision avoidance**

Julian et al.: **Deep Neural Network Compression for Aircraft Collision Avoidance Systems**, JGCD '19

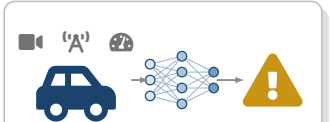
# Neural Networks in Real-Time Systems

## More and more neural networks run on microcontrollers



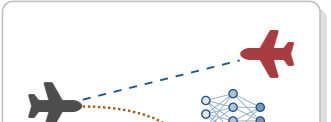
**Temperature calculation**

Kirchgässner et al.: **Estimating Electric Motor Temperatures With Deep Residual Machine Learning**, IEEE TPEL '21



**Fault detection**

Theissler et al.: **Predictive Maintenance Enabled by Machine Learning: Use Cases and Challenges in the Automotive Industry**, RESS '21



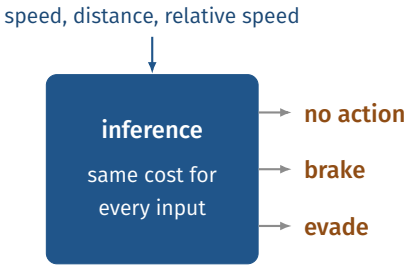
**Collision avoidance**

Julian et al.: **Deep Neural Network Compression for Aircraft Collision Avoidance Systems**, JGCD '19

Every task needs a **sound** and **tight** WCET bound, including the network

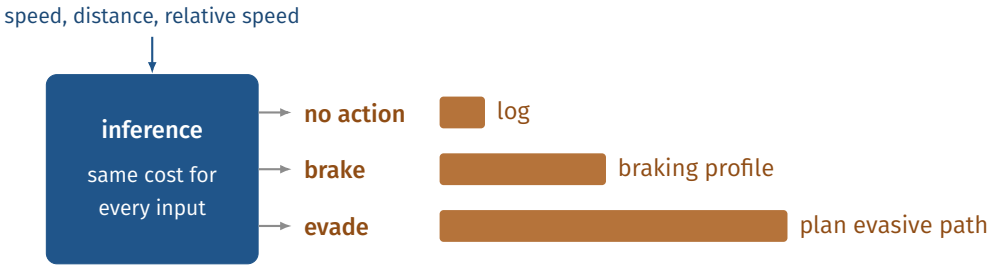
# WCET for Neural Networks

**Simplified example:** driving assistant



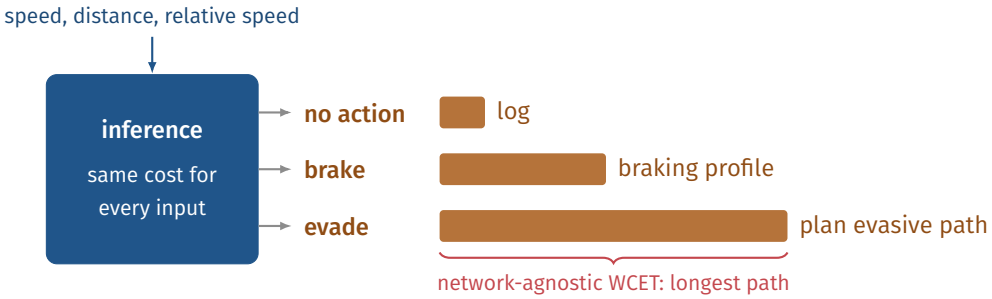
# WCET for Neural Networks

**Simplified example:** driving assistant



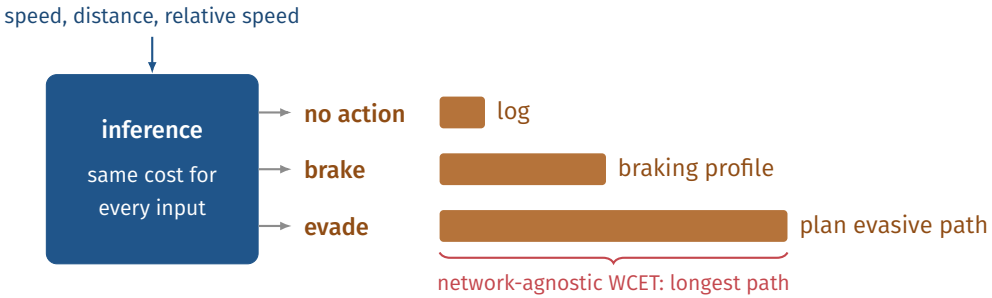
# WCET for Neural Networks

**Simplified example:** driving assistant



# WCET for Neural Networks

**Simplified example:** driving assistant




Which actions are possible depends on the **driving situation**

# WCET for Neural Networks: Operating Modes



**Highway**
speed 130+ km/h

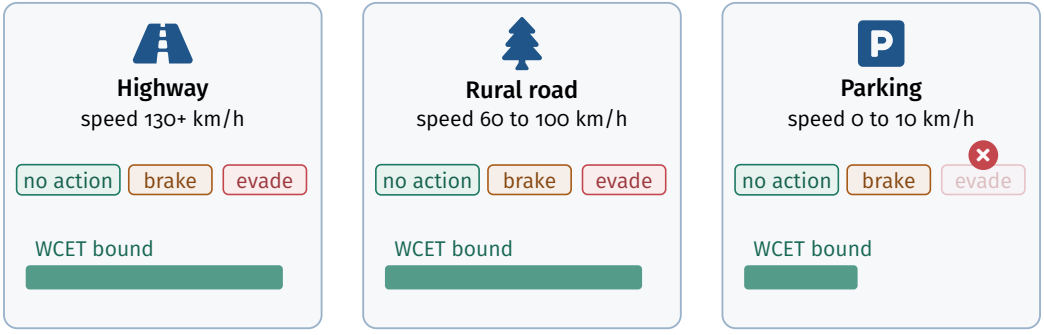


**Rural road**
speed 60 to 100 km/h

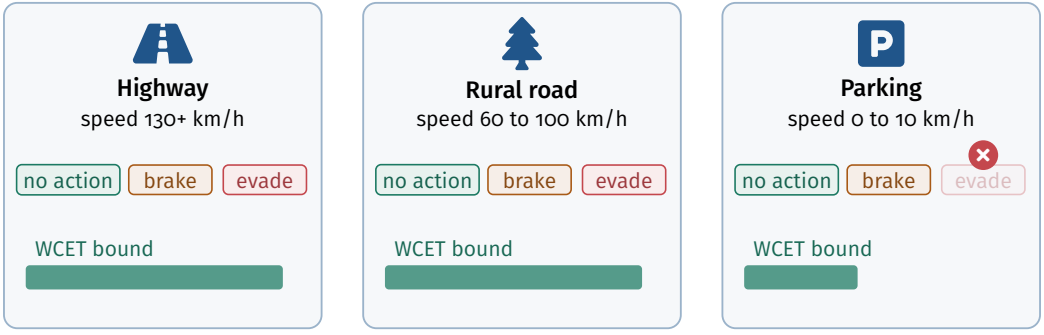


**Parking**
speed 0 to 10 km/h

# WCET for Neural Networks: Operating Modes



# WCET for Neural Networks: Operating Modes




Goal: an **individual WCET** per operating mode

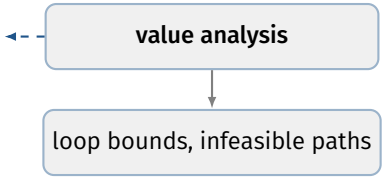
# Value Analysis in WCET Tools

```
speed  ← read_speed()
dist   ← read_distance()
c      ← classify(speed, dist)
switch c:
  no action  → log()
  brake      → for i = 1 to 200:  brake_step()
  evade      → for i = 1 to 900:  plan_step()
```

# Value Analysis in WCET Tools

```
speed  ← read_speed()            // ∈ [0, 10] km/h
dist   ← read_distance()         // ∈ [0.2, 5] m
c      ← classify(speed, dist)   // ∈ ?

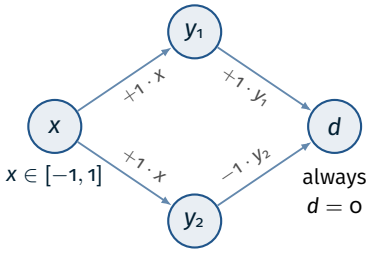
switch c:
  no action  → log()
  brake      → for i = 1 to 200:  brake_step()
  evade      → for i = 1 to 900:  plan_step()
```




State of the art: **intervals**, one range per variable (non-relational)

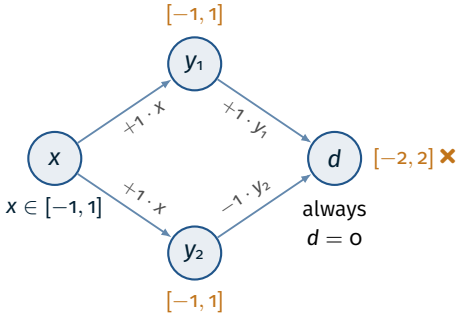
# Value Analysis with Intervals

Neurons are weighted sums, so they combine the *same* inputs



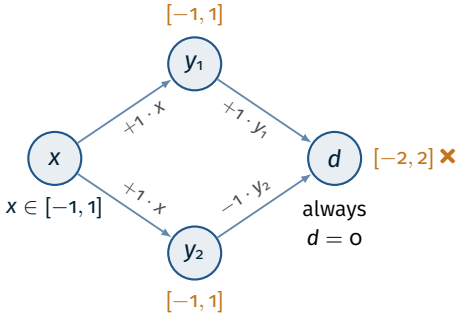
# Value Analysis with Intervals

Neurons are weighted sums, so they combine the *same* inputs

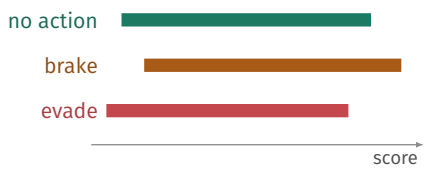


# Value Analysis with Intervals

Neurons are weighted sums, so they combine the *same* inputs



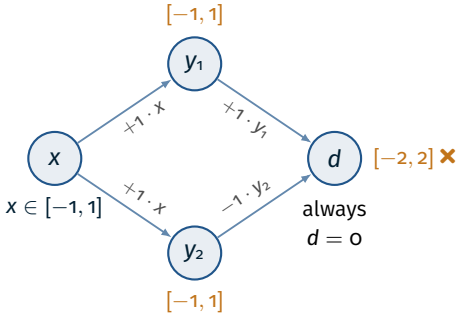

**intervals for the classifier example**



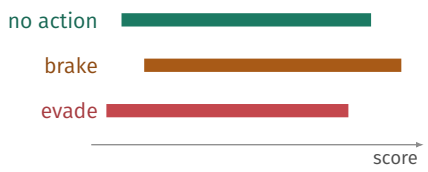

all output ranges overlap
⇒ no output can be excluded

# Value Analysis with Intervals

Neurons are weighted sums, so they combine the *same* inputs




**intervals for the classifier example**

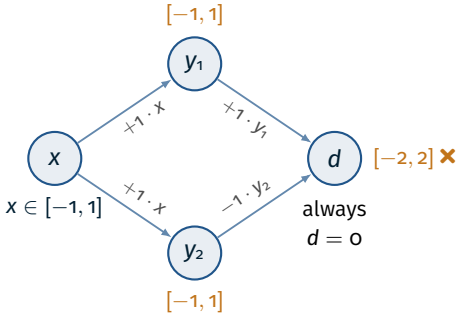



all output ranges overlap
⇒ no output can be excluded

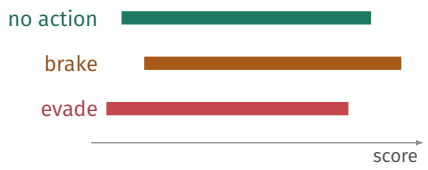
Dependencies are lost in every layer ⇒ **highly pessimistic WCET**

# Value Analysis with Intervals

Neurons are weighted sums, so they combine the *same* inputs




**intervals for the classifier example**




all output ranges overlap
⇒ no output can be excluded

Dependencies are lost in every layer ⇒ **highly pessimistic WCET**
Better, **relational** domains are established in robustness verification

# Problem Statement

How can we derive **tight**, yet **sound** WCET bounds
for code whose control flow depends on a **neural network's output**?

# Approach in a Nutshell

How can we derive **tight**, yet **sound** WCET bounds
for code whose control flow depends on a **neural network's output**?

**In a Nutshell:**

- Bound network outputs per operating mode with relational domains
  - Reuse established domains from robustness verification
  - Excluded classes result in tighter WCET bounds

# Approach in a Nutshell

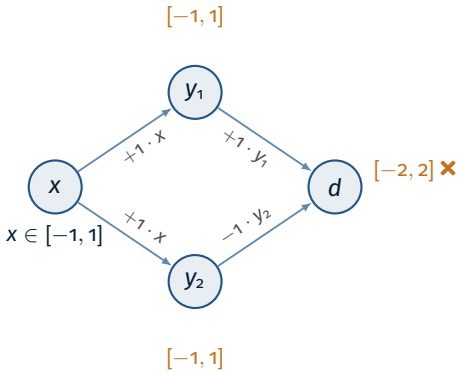
How can we derive **tight**, yet **sound** WCET bounds
for code whose control flow depends on a **neural network's output**?

**In a Nutshell:**

- Bound network outputs per operating mode with relational domains
  - Reuse established domains from robustness verification
  - Excluded classes result in tighter WCET bounds
- DeepWCET: compiler-integrated WCET analysis framework
  - Automatic: value analysis inside ONNX-MLIR, no manual analysis
  - Case study: up to **42 % lower WCET** for certain operating modes

# Relational Abstract Domains

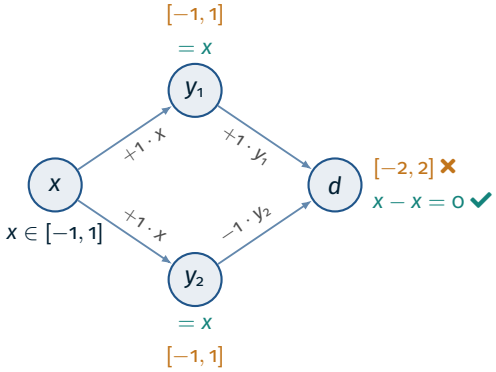
# Relational Domains Keep Relations Between Neurons




**Interval domain**

One range per neuron

# Relational Domains Keep Relations Between Neurons




**Interval domain**

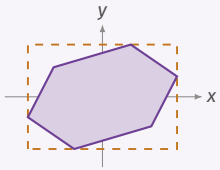
One range per neuron

**Relational domain**

Neurons as expressions over the shared inputs ⇒ shared terms cancel

# DeepZ and DeepPoly

## DeepZ: zonotopes

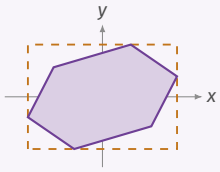



zonotope

intervals

# DeepZ and DeepPoly

## DeepZ: zonotopes




zonotope

intervals

Expression over shared $\varepsilon_i \in [-1, 1]$

**store** $y = c_0 + \sum_i c_i \varepsilon_i$

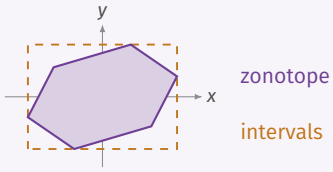
**range** $c_0 \pm \sum_i |c_i|$

**relation** same input $\Rightarrow$ same $\varepsilon_i$

Singh et al.: **Fast and Effective Robustness Certification**, NeurIPS '18

# DeepZ and DeepPoly

## DeepZ: zonotopes




Expression over shared $\varepsilon_i \in [-1, 1]$

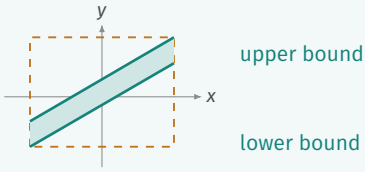
**store** $y = c_0 + \sum_i c_i \varepsilon_i$

**range** $c_0 \pm \sum_i |c_i|$

**relation** same input $\Rightarrow$ same $\varepsilon_i$

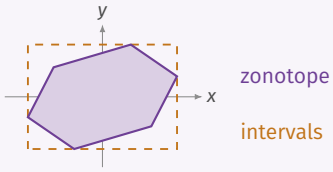
Singh et al.: **Fast and Effective Robustness Certification**, NeurIPS '18

## DeepPoly: linear bounds



# DeepZ and DeepPoly

## DeepZ: zonotopes




Expression over shared $\varepsilon_i \in [-1, 1]$

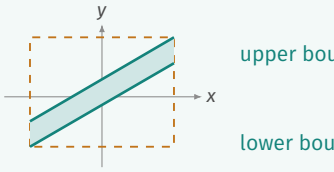
**store** $y = c_0 + \sum_i c_i \varepsilon_i$

**range** $c_0 \pm \sum_i |c_i|$

**relation** same input $\Rightarrow$ same $\varepsilon_i$

Singh et al.: **Fast and Effective Robustness Certification**, NeurIPS '18

## DeepPoly: linear bounds




Linear bounds over the previous layer $x$

**store** $a^{\leq} x + c^{\leq} \leq y \leq a^{\geq} x + c^{\geq}$

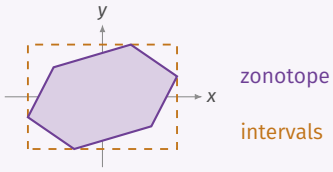
**range** substitute back to the inputs

**relation** shared terms cancel on the way

Singh et al.: **An Abstract Domain for Certifying Neural Networks**, POPL '19

# DeepZ and DeepPoly

## DeepZ: zonotopes




Expression over shared $\varepsilon_i \in [-1, 1]$

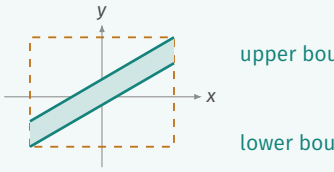
**store** $y = c_o + \sum_i c_i \varepsilon_i$

**range** $c_o \pm \sum_i |c_i|$

**relation** same input $\Rightarrow$ same $\varepsilon_i$

Singh et al.: **Fast and Effective Robustness Certification**, NeurIPS '18

## DeepPoly: linear bounds




Linear bounds over the previous layer $x$

**store** $a^{\leq}x + c^{\leq} \leq y \leq a^{\geq}x + c^{\geq}$

**range** substitute back to the inputs

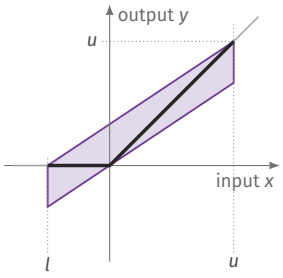
**relation** shared terms cancel on the way

Singh et al.: **An Abstract Domain for Certifying Neural Networks**, POPL '19

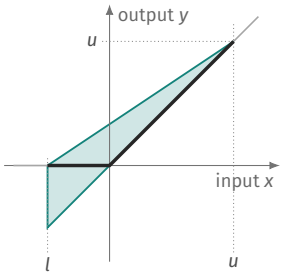
**Exact** for affine layers, **approximation** for nonlinear activation functions

# ReLU - Approximation

$$\mathrm{ReLU}(x) = \max(0, x)$$



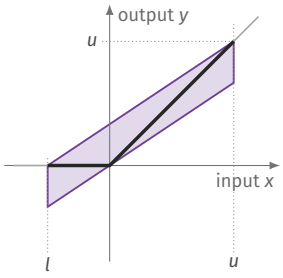

**DeepZ: parallelogram**
one fresh noise symbol $\varepsilon_{\mathrm{new}}$



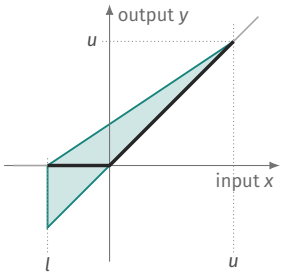

**DeepPoly: triangle**
no new symbols,
lower bound $y \geq 0$ or $y \geq x$

# ReLU - Approximation

$$\mathrm{ReLU}(x) = \max(0, x)$$




**DeepZ: parallelogram**
one fresh noise
symbol $\varepsilon_{\mathrm{new}}$

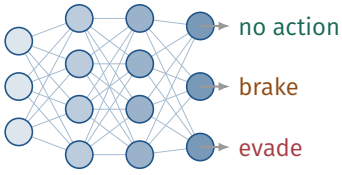



**DeepPoly: triangle**
no new symbols,
lower bound $y \geq 0$ or $y \geq x$

Precision is only lost at **unstable ReLUs**, relations between neurons stay

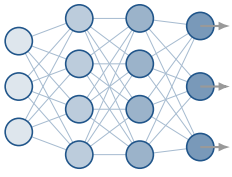
# Symbolic and Concrete Bounds at the Output

The class with the **highest score** is selected



# Symbolic and Concrete Bounds at the Output

The class with the **highest score** is selected




**symbolic bounds**

$n_o + \sum_i n_i \varepsilon_i$

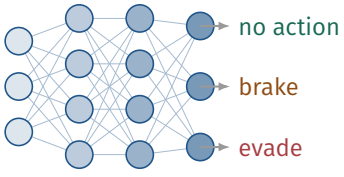
$b_o + \sum_i b_i \varepsilon_i$

$e_o + \sum_i e_i \varepsilon_i$

(DeepZ form)

# Symbolic and Concrete Bounds at the Output

The class with the **highest score** is selected




**symbolic bounds**

$n_o + \sum_i n_i \varepsilon_i$

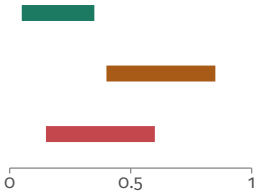
$b_o + \sum_i b_i \varepsilon_i$

$e_o + \sum_i e_i \varepsilon_i$

(DeepZ form)

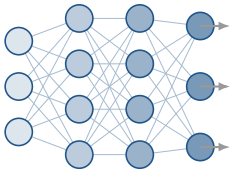
evaluate

**concrete bounds**



# Symbolic and Concrete Bounds at the Output

The class with the **highest score** is selected




inputs $x \in X_m$

**symbolic bounds**

$n_o + \sum_i n_i \varepsilon_i$

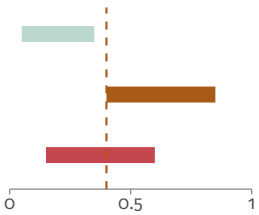
$b_o + \sum_i b_i \varepsilon_i$

$e_o + \sum_i e_i \varepsilon_i$

(DeepZ form)

evaluate

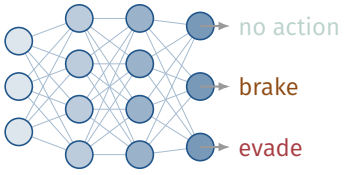
**concrete bounds**




✔ excluded

# Symbolic and Concrete Bounds at the Output

The class with the **highest score** is selected




**symbolic bounds**

$n_o + \sum_i n_i \varepsilon_i$

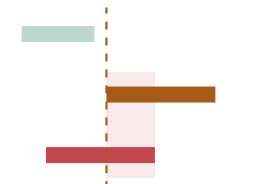
$b_o + \sum_i b_i \varepsilon_i$

$e_o + \sum_i e_i \varepsilon_i$

(DeepZ form)

evaluate

**concrete bounds**



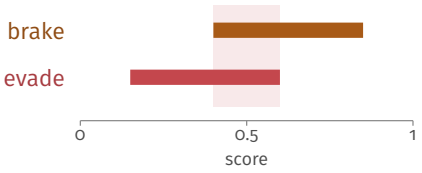

✔ excluded

? overlap

# Overlapping Bounds: Subtract the Terms

Our approach: subtract the **symbolic terms** instead

**Concrete bounds**

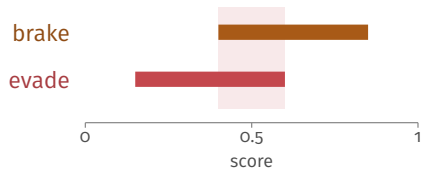



✖ overlap ⇒ evade **not** excluded

# Overlapping Bounds: Subtract the Terms

Our approach: subtract the **symbolic terms** instead

**Concrete bounds**




✖ overlap ⇒ evade **not** excluded

**Subtract the symbolic terms (DeepZ)**

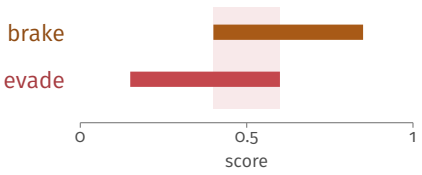
brake $0.625 + 0.2\,\varepsilon_1 + 0.025\,\varepsilon_2$

evade $0.375 + 0.2\,\varepsilon_1 + 0.025\,\varepsilon_3$

# Overlapping Bounds: Subtract the Terms

Our approach: subtract the **symbolic terms** instead

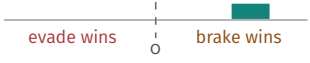
**Concrete bounds**




✖ overlap ⇒ evade **not** excluded

**Subtract the symbolic terms (DeepZ)**

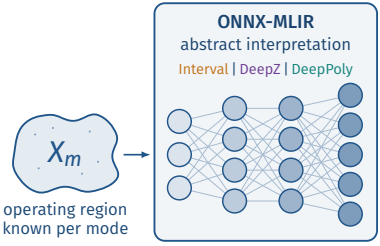
$$\begin{aligned} \text{brake} \quad & 0.625 + \cancel{0.2\,\varepsilon_1} + 0.025\,\varepsilon_2 \\ \text{evade} \quad & 0.375 + \cancel{0.2\,\varepsilon_1} + 0.025\,\varepsilon_3 \\ \hline - \quad & 0.25 + 0.025\,\varepsilon_2 - 0.025\,\varepsilon_3 \;\in\; [0.20,\, 0.30] \end{aligned}$$



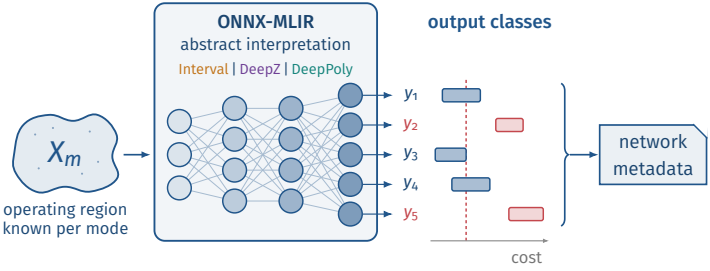

✔ difference always positive
⇒ evade is never selected: **excluded**

# Design

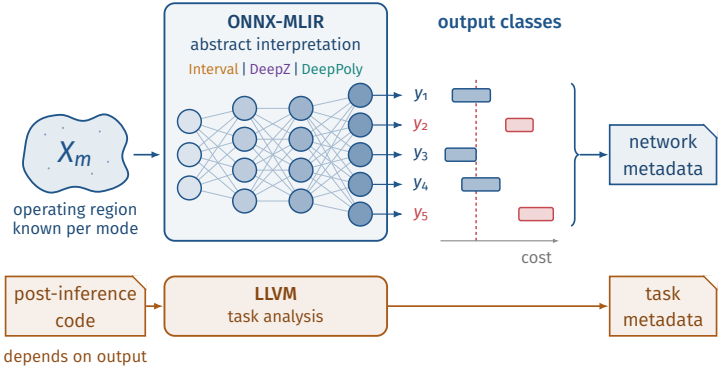
# DeepWCET: Overview



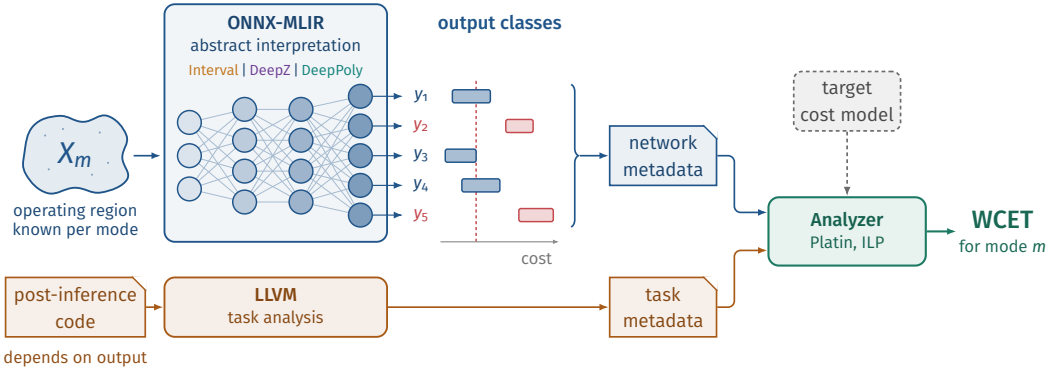
# DeepWCET: Overview



# DeepWCET: Overview



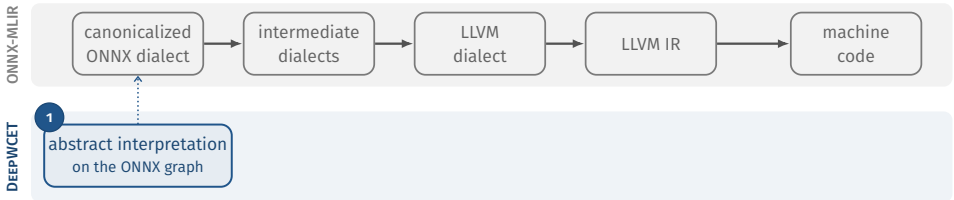
# DeepWCET: Overview




Maroun et al.
**The Platin Multi-Target Worst-Case Analysis Tool**
WCET '24

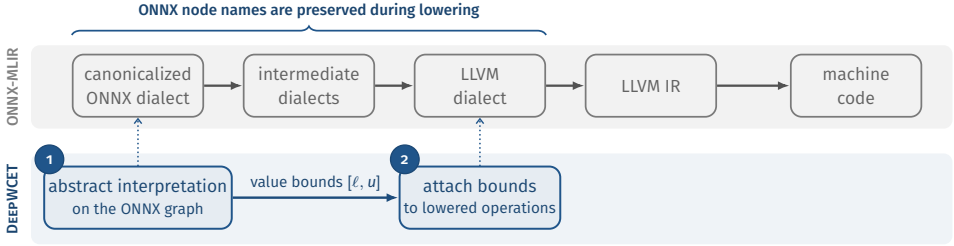
# Implementation in ONNX-MLIR

ONNX-MLIR: open-source compiler, lowers ONNX models step by step



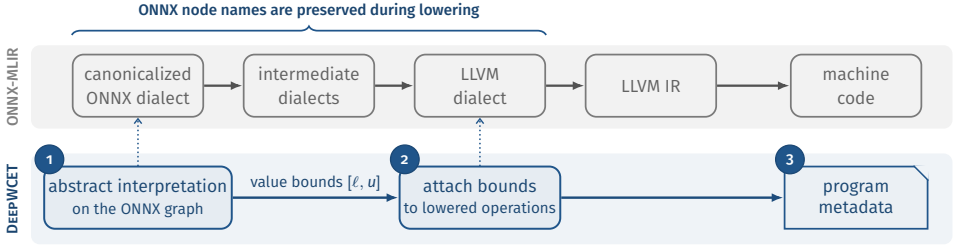
# Implementation in ONNX-MLIR

ONNX-MLIR: open-source compiler, lowers ONNX models step by step



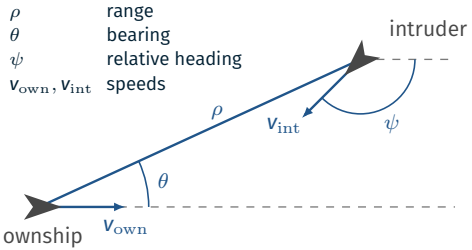
# Implementation in ONNX-MLIR

ONNX-MLIR: open-source compiler, lowers ONNX models step by step



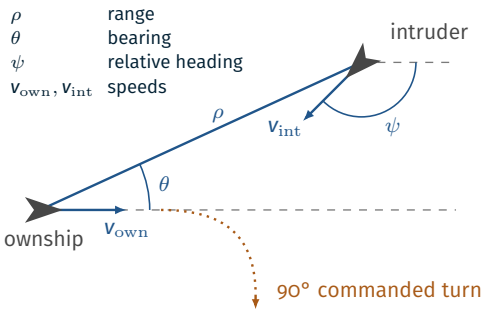
# Evaluation

# Case Study: ACAS Xu




- Collision avoidance for UAVs
- 5 inputs, 5 outputs
- Standard benchmark for robustness verification

# Case Study: ACAS Xu




- Collision avoidance for UAVs
- 5 inputs, 5 outputs
- Standard benchmark for robustness verification

**Advisory → projection loop bound**

| Advisory | Turn rate | Iterations |
|---|---|---|
| clear of conflict | | 0 |
| strong left / right | 3 °/s | 1500 |
| weak left / right | 1.5 °/s | **3000** |

worst case

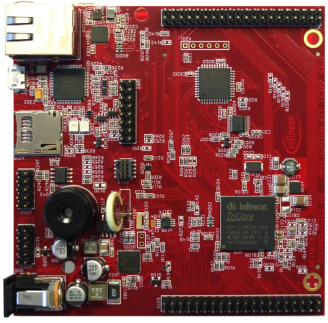
# Evaluation Setup

## Platform

- Infineon AURIX TC397XE, 100 MHz
- Single core, scratchpad, no caches
- One task, no preemption

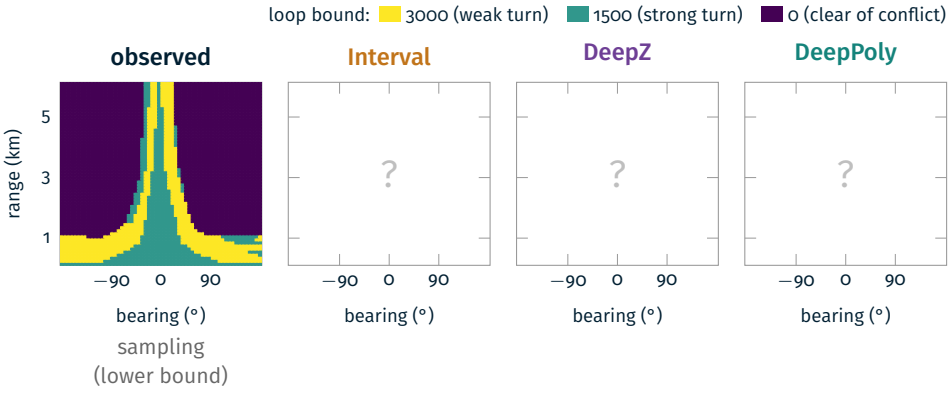
## Toolchain

- ONNX-MLIR, LLVM 23
- Own TriCore backend (ported)

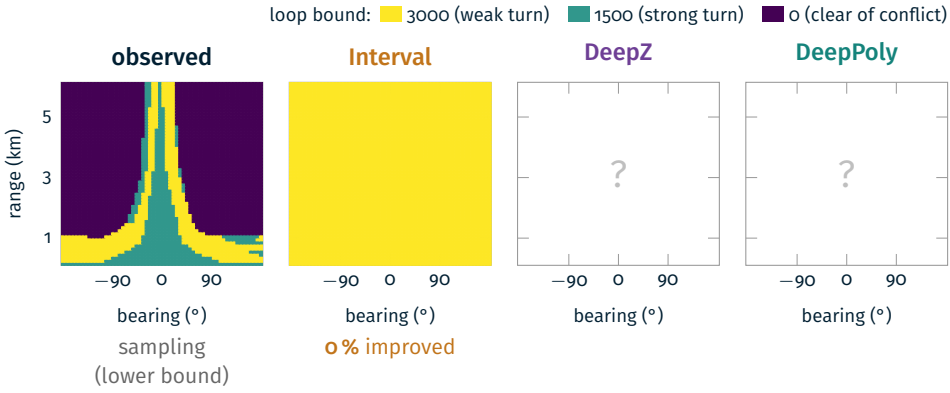


Infineon AURIX TC397 application kit

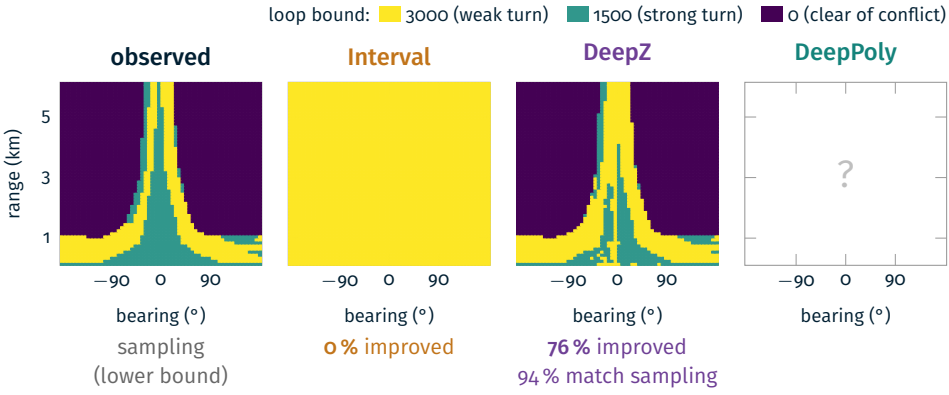
# Advisory Exclusion over the Operating Space



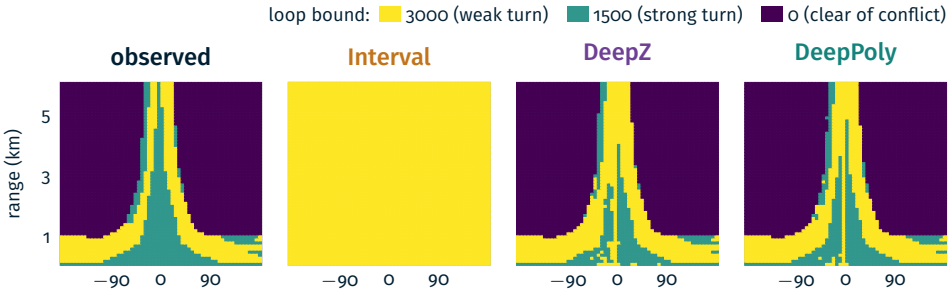
# Advisory Exclusion over the Operating Space



# Advisory Exclusion over the Operating Space

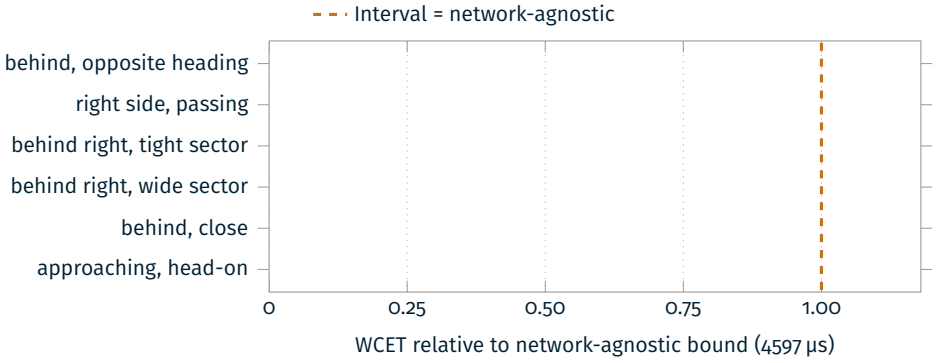


# Advisory Exclusion over the Operating Space

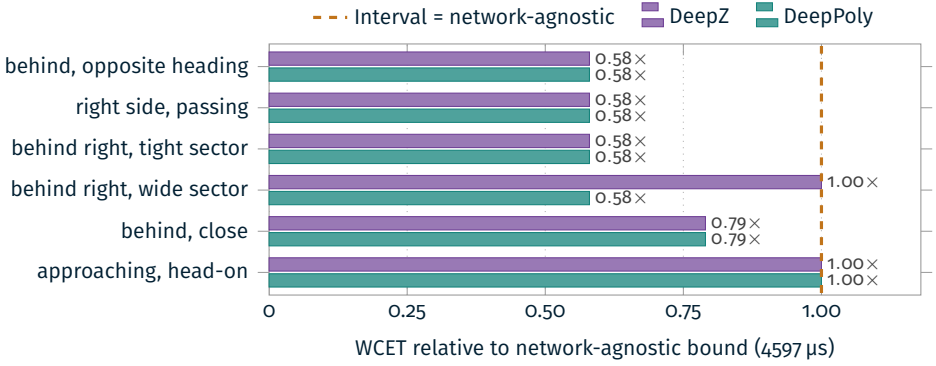



Intervals exclude **nothing**, DeepZ and DeepPoly **significantly** reduce the possible actions

# Mode-Specific WCET

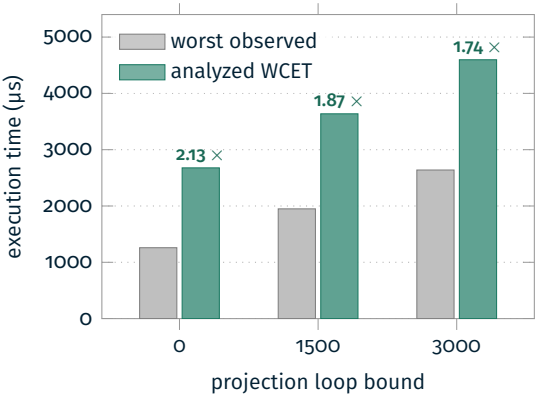


# Mode-Specific WCET



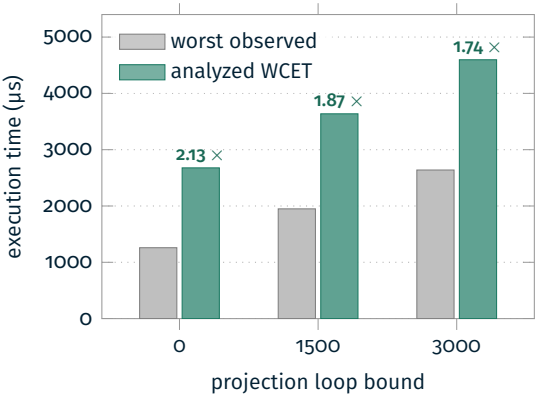

Up to **42 % lower WCET** in modes where only clear of conflict remains

# Analyzed vs. Measured Execution Time




- Timing model: single core, no caches, no timing anomalies
- Overestimation: 1.74 to 2.13×
  - 44 to 60 % of instructions overlap with another one

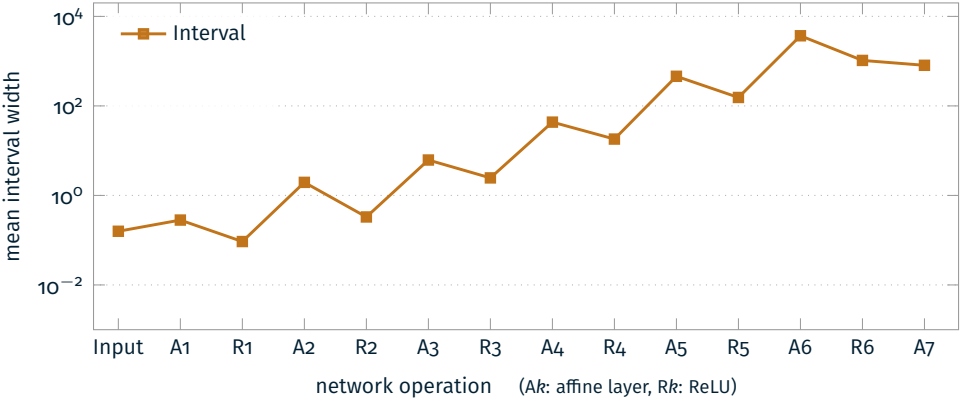
# Analyzed vs. Measured Execution Time



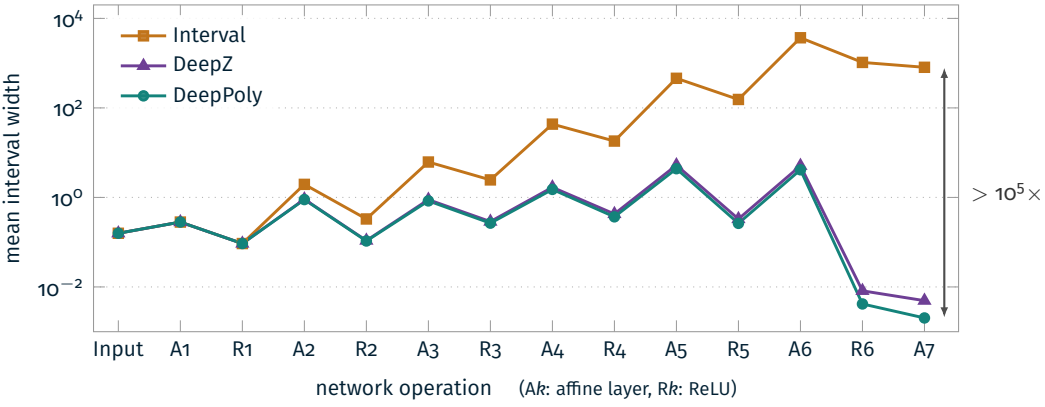

- Timing model: single core, no caches, no timing anomalies
- Overestimation: 1.74 to 2.13×
  - 44 to 60 % of instructions overlap with another one

All bounds hold, a **microarchitecture-aware** timing model would tighten them

# Interval Width Through the Network



# Interval Width Through the Network




Intervals grow with **every layer**, relational domains keep the bounds **tight**

# Conclusion

# Conclusion

## Future Work

- Infer input regions automatically (backward abstract interpretation)
- Other network types, support for more operators

# Conclusion

## Future Work

- Infer input regions automatically (backward abstract interpretation)
- Other network types, support for more operators

## DeepWCET

✔ Reuses relational domains from robustness verification

→ Excluded network outputs result in tighter per-mode WCET

# Conclusion

## Future Work

- Infer input regions automatically (backward abstract interpretation)
- Other network types, support for more operators

## DeepWCET

- ✓ Reuses relational domains from robustness verification
  - → Excluded network outputs result in tighter per-mode WCET
- ✓ Implementation inside the ONNX-MLIR compiler
  - → Supports any networks in the ONNX format

# Conclusion

## Future Work

- Infer input regions automatically (backward abstract interpretation)
- Other network types, support for more operators

## DeepWCET

- ✓ Reuses relational domains from robustness verification
  - → Excluded network outputs result in tighter per-mode WCET
- ✓ Implementation inside the ONNX-MLIR compiler
  - → Supports any networks in the ONNX format
- ✓ ACAS Xu: per-mode WCET down to $0.58\times$ ($-42\,\%$)
  - → DeepZ/DeepPoly: $> 10^5\times$ tighter than intervals

# Conclusion

## Future Work

- Infer input regions automatically (backward abstract interpretation)
- Other network types, support for more operators

## DeepWCET

- ✓ Reuses relational domains from robustness verification
  - → Excluded network outputs result in tighter per-mode WCET
- ✓ Implementation inside the ONNX-MLIR compiler
  - → Supports any networks in the ONNX format
- ✓ ACAS Xu: per-mode WCET down to $0.58\times$ ($-42\,\%$)
  - → DeepZ/DeepPoly: $> 10^5\times$ tighter than intervals

**Thank you for your attention!**