Introduction to the verification of neural networks running on a PLC: an LHC cooling tower example

Ignacio D. Lopez-Miguel ignacio.lopez@tuwien.ac.at,
Borja Fernández Adiego, Faiq Ghawash, Enrique Blanco Viñuela
20/04/2023
Roadmap

Neural networks → Verification of Neural networks

Neural-network based controllers → Verification of neural-network based controllers

Case study → Properties to verify → Different approaches to verify those properties

Conclusion
Roadmap

- Neural networks
- Verification of Neural networks
- Neural-network based controllers
- Verification of neural-network based controllers
- Case study
- Properties to verify
- Different approaches to verify those properties
- Conclusion
Artificial neural networks

What is an artificial neural network?
Artificial neural networks

What is an artificial neural network?

![Diagram of an artificial neural network with inputs, hidden layers, and outputs showing a dog and a cat.]
Artificial neural networks

What is an artificial neural network?

Mathematically
Artificial neural networks

What is an artificial neural network?

Mathematically

Neuron’s input: \( x_1w_{11}^1 + x_2w_{21}^1 + b_1^1 \)

Neuron’s output (RELU): \( \max(0, x_1w_{11}^1 + x_2w_{21}^1 + b_1^1) \)
Artificial neural networks

What is an artificial neural network?

Mathematically

Neuron’s input  
\[ x_1 w_{11}^1 + x_2 w_{21}^1 + b_1^1 \]

Neuron’s output (RELU)  
\[ \max(0, x_1 w_{11}^1 + x_2 w_{21}^1 + b_1^1) \]  
Non-linear  
Non-convex
Roadmap

Neural networks

Verification of Neural networks

Verification of neural-network based controllers

Neural-network based controllers

Properties to verify

Different approaches to verify those properties

Case study

Conclusion
Verification of neural networks

Why verification of NN?
Verification of neural networks

Why verification of NN?
Adversarial attacks / Robustness

Raw image

Adversarial image
Verification of neural networks

Why verification of NN?
Adversarial attacks / Robustness

Raw image

Adversarial image

Dog

+ noise

Ostrich
Verification of neural networks

Why verification of NN?
Adversarial attacks / Robustness

Verification of neural networks

Why verification of NN?
Guaranteeing properties
Verification of neural networks

Why verification of NN?
Guaranteeing properties

Robustness

Image
Noise
Verification of neural networks

Why verification of NN?
Guaranteeing properties

Robustness

Image

Noise

Image

Noise
Verification of neural networks

Why verification of NN?
Guaranteeing properties

Robustness

Monotonicity

Image

Noise

Input

Output
Verification of neural networks

Why verification of NN?
Guaranteeing properties

Robustness
- Image
- Noise

Monotonicity
- Output
  - Input
- Temperature
  - Fan speed
Verification of neural networks

Why verification of NN?
Guaranteeing properties

Robustness

Monotonicity

Safety

- Image
- Noise
- Output
- Input
- Safe region
- Temperature
- Fan speed
- Input

Guaranteeing properties:
- Monotonicity
- Safety
- Robustness

Image

Noise

Temperature

Fan speed
Verification of neural networks

Why verification of NN?
Guaranteeing properties

Robustness
- Image
- Noise

Monotonicity
- Output
- Input

Safety
- Output
- Input
- Safe region

---

Temperature → Fan speed
Why verification of NN?
Verification of neural networks

Why verification of NN?

**Explainability**

- Is it possible that a given situation ever occurs?
- Is there a combination of inputs that leads to a given output?
Verification of neural networks

Why verification of NN?

**Explainability**

- Is it possible that a given situation ever occurs?
- Is there a combination of inputs that leads to a given output?

**Fairness (robustness)**

- Similar inputs but very different outputs?

No interview  
Interview
Verification vs Testing

Verification explores more possibilities than testing
Verification vs Testing

Verification explores more possibilities than testing

Input1 → Neural network → Output1
Input2 → Neural network → Output2
Verification explores more possibilities than testing

If Input1 is 1 and Input2 is 2, then Output2 is >5
Verification vs Testing

Verification explores more possibilities than testing

If Input1 is 1 and Input2 is 2, then Output2 is >5

If Output1 is >1, then Output2 shall be >5

Input1 \rightarrow Neural network \rightarrow Output1
Input2 \rightarrow Neural network \rightarrow Output2
Verification vs Testing

Verification explores more possibilities than testing

Input1 \rightarrow \text{Neural network} \rightarrow \text{Output1}
Input2 \rightarrow \text{Output2}

Functionality requirement

If \text{Input1} is 1 and \text{Input2} is 2, then \text{Output2} is >5

Safety requirement

If \text{Output1} is >1, then \text{Output2} shall be >5

2 Integer (16-bit) input variables \rightarrow 2^{16 \cdot 2} \approx 4.3 \cdot 10^9 \text{ combinations}
Verification vs Testing

Verification explores more possibilities than testing

If Input1 is 1 and Input2 is 2, then Output2 is >5

If Output1 is >1, then Output2 shall be >5

2 Integer (16-bit) input variables

2^{16 \cdot 2} \approx 4.3 \cdot 10^9 \text{ combinations}

explore all input combinations

2 Integer (16-bit) input variables

Formal verification
Verification of neural networks

How to verify a NN?
Verification of neural networks

How to verify a NN?
Verification of neural networks

How to verify a NN?

It considers a set of infinite points as the input of the neural network.
Verification of neural networks

How to verify a NN?

It considers a set of infinite points as the input of the neural network

Mathematically

$$(x_0 - \varepsilon, x_0 + \varepsilon) \xrightarrow{\text{Neural network}} f(x) \xrightarrow{(y_0 - \alpha, y_0 + \alpha)} \exists y_{\text{non-safe}}$$
Verification of neural networks

How to verify a NN?

It considers a set of infinite points as the input of the neural network:

\[(x_0 - \varepsilon, x_0 + \varepsilon) \rightarrow \text{Neural network} \rightarrow (y_0 - \alpha, y_0 + \alpha) \ni y_{\text{non-safe}}\]

Mathematically

Algorithms

Mixed-integer linear programming (MILP), satisfiability modulo theories (SMT), Simplex,...
Roadmap

- Neural networks
- Verification of Neural networks
- Neural-network based controllers
- Verification of neural-network based controllers
- Case study
- Properties to verify
- Different approaches to verify those properties
- Conclusion
Neural networks as controllers

Why NN-based controllers?
Neural networks as controllers

Why NN-based controllers?

Tasks hard to specify
- Autonomous driving

Several rule exceptions

Neural networks as controllers

Why NN-based controllers?

Tasks hard to specify
- Autonomous driving

Computationally fast
- Matrix multiplication

Several rule exceptions

10.5937/fme2101029P.
Neural networks as controllers

Why NN-based controllers?

<table>
<thead>
<tr>
<th>Tasks hard to specify</th>
<th>Computationally fast</th>
<th>Versatile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autonomous driving</td>
<td>Matrix multiplication</td>
<td>Non-linearities, No need to linearize</td>
</tr>
</tbody>
</table>

Several rule exceptions

Neural networks as controllers

Why NN-based controllers?

Tasks hard to specify
- Autonomous driving

Computationally fast
- Matrix multiplication

Versatile
- Non-linearities
- No need to linearize

Only data needed
- No physical modelling required
- Collect data

Several rule exceptions

Neural networks as controllers

Why NN-based controllers?

Tasks hard to specify
- Autonomous driving

Computationally fast
- Matrix multiplication

Versatile
- Non-linearities
- No need to linearize

Only data needed
- No physical modelling required
- Collect data

Several rule exceptions

Roadmap

- Neural networks
- Verification of Neural networks
- Neural-network based controllers
- Verification of neural-network based controllers
- Case study
- Properties to verify
- Different approaches to verify those properties
- Conclusion
Why verification of NN-based controllers?
Verification of NN-based controllers

Why verification of NN-based controllers?

Reachability

System’s output

Time

Stability

System’s output

Time
Verification of NN-based controllers

Why verification of NN-based controllers?

Reachability

Stability

System’s output

System’s output

Time

Time

Stable under external perturbations

Verification of NN-based controllers

Why verification of NN-based controllers?

Reachability

Stability

Safety

System’s output

Time

System’s output

Time

System’s output

Input

Reachability

Stability

Safety

System’s output

Input

System’s output

Time

Safe region

Stable under external perturbations


[3]
Verification of NN-based controllers

Why verification of NN-based controllers?

Reachability
- System's output vs. Time

Stability
- System's output vs. Time

Safety
- System's output vs. Input
  - Safe region
  - Centered beam

Stable under external perturbations

Roadmap

Neural networks → Verification of Neural networks

Neural-network based controllers → Verification of neural-network based controllers

Case study → Properties to verify → Different approaches to verify those properties

Conclusion
Case study

LHC cooling towers controls
LHC cooling towers controls

- Induced draft cooling towers (IDCTs)
LHC cooling towers controls

- Induced draft cooling towers (IDCTs)
- Three modes:
  - Ventilation
  - Showering
  - Bypass
LHC cooling towers controls

- Induced draft cooling towers (IDCTs)
- Three modes:
  - Ventilation
  - Showering
  - Bypass
- Fan speed
LHC cooling towers controls

- Induced draft cooling towers (IDCTs)
- Three modes:
  - Ventilation
  - Showering
  - Bypass
- Fan speed
- Control objective:
  - Keep outlet water temperature within strict limits
  - Utilize minimum amount of energy
LHC cooling towers controls

- Induced draft cooling towers (IDCTs)
- Three modes:
  - Ventilation
  - Showering
  - Bypass
- Fan speed
- Control objective:
  - Keep outlet water temperature within strict limits
  - Utilize minimum amount of energy

Model predictive controller (MPC)
LHC cooling towers controls

- Induced draft cooling towers (IDCTs)
- Three modes:
  - Ventilation
  - Showering
  - Bypass
- Fan speed
- Control objective:
  - Keep outlet water temperature within strict limits
  - Utilize minimum amount of energy
Case study

The NN is implemented in a PLC
Case study

The NN is implemented in a PLC

Why NN on PLCs?

Example: SIMATIC S7-1200
Case study

The NN is implemented in a PLC

Why NN on PLCs?

• Widely used in the process industry
• Hardware robustness
• Communication capabilities
• Modularity
• Simple to be programmed

Example: SIMATIC S7-1200
Case study

The NN is implemented in a PLC

Why NN on PLCs?

• Widely used in the process industry
• Hardware robustness
• Communication capabilities
• Modularity
• Simple to be programmed

Example: SIMATIC S7-1200

NN on PLC code (structured control language)

```plaintext
scnd_lyr_neurons := 8;
FOR i := 0 TO scnd_lyr_neurons - 1 DO
    FOR j := 0 TO frst_lyr_neurons - 1 DO
        temp := temp + frst_lyr_out[j] * scnd_lyr_weights.Data[j, i];
    END_FOR;
    scnd_lyr_out[i] := MAX(IN1:=0, IN2:=(temp+scnd_lyr_bias.Data[i])); temp:=0;
END_FOR;
```
Roadmap

Neural networks → Verification of Neural networks

Neural-network based controllers → Verification of neural-network based controllers

Case study → Properties to verify → Different approaches to verify those properties

Conclusion
Properties that were verified
Case study

Properties that were verified

Operational modes reachability

Ventilation → Input1
Showering → Input2
Bypass → Input3

Do these inputs exist?
Case study

Properties that were verified

Operational modes reachability

- Ventilation
- Showering
- Bypass

Input1
Input2
Input3

Do these inputs exist?

Fan speed constraint satisfaction

Does this input exist?

Input1
Ventilation

or

60% 100%
Case study

Properties that were verified

Operational modes reachability

- Ventilation
- Showering
- Bypass

Do these inputs exist?

Fan speed constraint satisfaction

Does this input exist?

Monotonicity

Fan speed

\[ s_0 < s_1 \]

\[ t_0 < t_1 \]

Do \( t_0 < t_1 \) exist such that \( s_0 > s_1 \)?
Case study

Properties that were verified

Operational modes reachability

- Ventilation
- Showering
- Bypass

Do these inputs exist?

Fan speed constraint satisfaction

Does this input exist?

or

60% 100%

Monotonicity

Fan speed

$S_0$ $S_1$

t_0 t_1 Temperature

Do $t_0 < t_1$ exist such that $S_0 > S_1$?

Softmax overflow

$\frac{e^{out_i}}{e^{out_1} + e^{out_2} + e^{out_3}}$

Ventilation Bypass

Showering

Input1

$\text{Input 1}$

Ventilation

$\text{Input 2}$

$\text{Input 3}$
Case study

Properties that were verified

Operational modes reachability

- Ventilation
- Showering
- Bypass

Do these inputs exist?

Fan speed constraint satisfaction

- Input1
- Ventilation
- or
- 60%
- 100%

Monotonicity

- Fan speed
- Temperature

Do $t_0 < t_1$ exist such that $s_0 > s_1$?

Softmax overflow

\[
\frac{e^{out_1}}{e^{out_1} + e^{out_2} + e^{out_3}}
\]

Robustness

- Input1
- $+\varepsilon$
- Bypass
- Ventilation
- Does this point exist?
Case study

Properties that were verified

Operational modes reachability
- Ventilation
- Showering
- Bypass
Input1
Input2
Input3

Do these inputs exist?

Fan speed constraint satisfaction
Does this input exist?
Input1 → Ventilation

Monotonicity

Softmax overflow

Robustness

They help to better understanding the NN

Fan speed constraint satisfaction

\[ \frac{e^{out_1}}{e^{out_1} + e^{out_2} + e^{out_3}} \]

Do \( t_0 < t_1 \) exist such that \( s_0 > s_1 \)?

\[ t_0 \quad t_1 \]

Temperature

Fan speed

\[ s_0 \quad s_1 \]
Case study

Properties that were verified

Operational modes reachability

- Ventilation
- Showering
- Bypass

Do these inputs exist?

Fan speed constraint satisfaction

- Input1
- Ventilation

Does this input exist?

Monotonicity

- Fan speed: $s_0, s_1$
- Temperature: $t_0, t_1$

Do $t_0 < t_1$ exist such that $s_0 > s_1$?

Softmax overflow

- Input1
- NN

$$e_{out_i} = \frac{e_{out_1} + e_{out_2} + e_{out_3}}{e_{out_1} + e_{out_2} + e_{out_3}}$$

Robustness

- Input1
- Ventilation
- Bypass
- Showering

Does this point exist?

They help to better understanding the NN

They represent problems
Roadmap

Neural networks → Verification of Neural networks

Neural-network based controllers → Verification of neural-network based controllers

Case study → Properties to verify → Different approaches to verify those properties

Conclusion
Approaches to verify the properties

Different methods used

- NN in Python
  - keras
  - h5

- Python script

- NN in PLC code

- Properties in natural language
Approaches to verify the properties

Different methods used

- NN in Python keras h5
- NN in PLC code
- NN in Python
- Python API for Z3
- Python script

Manually

- keras2onnx
- onnx
- nnet
- nnet2onnx

- Assertions /patterns in PLCverif
- Assertions in Z3
- Properties in natural language
- Assertions in vnnlib
- Conditions

Semi-automatic

Manually

Manually
Approaches to verify the properties

Different methods used

NN in Python keras h5
- Python script
  - Manually

NN in PLC code
- Manually
- Automatic
- Manually

NN in Python
- Manually

Python script
- Manually

keras2onnx
- nnet
- nnet2onnx
- onnx

nnenum
- Assertions in vnnlib
  - Semi-automatic
  - Manually

PLCverif
- Assertions/patterns in PLCverif
  - Semi-automatic
  - Manually

Z3
- Assertions in Z3
  - Manually
- Conditions

Properties in natural language
- Manually

Conditions
- Manually
PLCverif

Pipeline

NN in Python keras h5

Python script

NN in PLC code

Automatic

PLCverif

Assertions/patterns in PLCverif

Semi-automatic

Properties in natural language
PLCverif workflow
It is possible that the selected mode is ventilation

```
FOR i := 0 TO scnd_lyr_neurons - 1
DO FOR j := 0 TO frst_lyr_neurons - 1 DO
    temp := temp + frst_lyr_out[j] * scnd_lyr_weights.Data[j, i];
END_FOR;
scnd_lyr_out[i] := MAX(IN1:=0, IN2:=(temp+scnd_lyr_bias.Data[i]));
temp:=0;
END_FOR;
```
It is possible that the selected mode is ventilation

//#ASSERT mode<>ventilation

AG(mode<>ventilation)
It is possible that the selected mode is ventilation

//#ASSERT mode<>ventilation

AG(mode<>ventilation)
PLCverif

It is possible that the selected mode is ventilation

// ASSERT mode<>ventilation

AG(mode<>ventilation)
It is possible that the selected mode is ventilation

```
FOR i := 0 TO scnd_lyr_neurons - 1
DO FOR j := 0 TO frst_lyr_neurons - 1 DO
    temp := temp + frst_lyr_out[j] * scnd_lyr_weights.Data[j, i];
END_FOR;
scnd_lyr_out[i] := MAX(IN1 := 0, IN2 := (temp + scnd_lyr_bias.Data[i]));
temp := 0;
END_FOR;
```

//#ASSERT mode<>ventilation

Pros:
- Possible to express complex properties
- Easy to use
- Final system

Cons:
- Not efficient for neural networks
NN verifier - nnenum

Pipeline

- NN in Python h5
  - keras2onnx
  - nnet
  - nnet2onnx
  - onnx
  - nnenum

- NN in PLC code
  - Python script
  - Manually
  - Assertions in vnnlib
  - Manually
  - Properties in natural language

- Properties in natural language
Pros:
• Very efficient
• Scalable

Cons:
• Limited to certain architectures
• No loops
• No complex properties
NN verifier - nnenum

**Pros:**
- Very efficient
- Scalable

**Cons:**
- Limited to certain architectures
- No loops
- No complex properties

**Property encoding (vnnlib)**

(declare-const X_0 Real)
(declare-const X_1 Real)
(declare-const X_2 Real)
(declare-const Y_0 Real)
(declare-const Y_1 Real)
(declare-const Y_2 Real)
(assert (>= X_0 20.0))
(assert (<= X_0 25.0))
(assert (>= X_1 23))
(assert (<= X_1 27))
(assert (>= X_2 8.0))
(assert (<= X_2 21.0))
(assert (>= Y_0 Y_1))
(assert (>= Y_0 Y_2))
Pros:
- Very efficient
- Scalable

Cons:
- Limited to certain architectures
- No loops
- No complex properties

Property encoding (vnnlib)

(declare-const X_0 Real)
(declare-const X_1 Real)
(declare-const X_2 Real)
(declare-const Y_0 Real)
(declare-const Y_1 Real)
(declare-const Y_2 Real)
(assert (>= X_0 20.0))
(assert (<= X_0 25.0))
(assert (>= X_1 23))
(assert (<= X_1 27))
(assert (>= X_2 8.0))
(assert (<= X_2 21.0))
(assert (>= Y_0 Y_1))
(assert (>= Y_0 Y_2))
NN verifier - nnenum

Pros:
- Very efficient
- Scalable

Cons:
- Limited to certain architectures
- No loops
- No complex properties

Property encoding (vnnlib)

(declare-const X_0 Real)
(declare-const X_1 Real)
(declare-const X_2 Real)
(declare-const Y_0 Real)
(declare-const Y_1 Real)
(declare-const Y_2 Real)
(assert (>= X_0 20.0))
(assert (<= X_0 25.0))
(assert (>= X_1 23))
(assert (<= X_1 27))
(assert (>= X_2 8.0))
(assert (<= X_2 21.0))
(assert (>= Y_0 Y_1))
(assert (>= Y_0 Y_2))
NN verifier - nnenum

Pros:
- Very efficient
- Scalable

Cons:
- Limited to certain architectures
- No loops
- No complex properties

Property encoding (vnnlib)

(declare-const X_0 Real)
(declare-const X_1 Real)
(declare-const X_2 Real)
(declare-const Y_0 Real)
(declare-const Y_1 Real)
(declare-const Y_2 Real)
(assert (>= X_0 20.0))
(assert (<= X_0 25.0))
(assert (>= X_1 23))
(assert (<= X_1 27))
(assert (>= X_2 8.0))
(assert (<= X_2 21.0))
(assert (>= Y_0 Y_1))
(assert (>= Y_0 Y_2))
Pros:
- Very efficient
- Scalable

Cons:
- Limited to certain architectures
- No loops
- No complex properties

Property encoding (vnnlib)

(declare-const X_0 Real)
(declare-const X_1 Real)
(declare-const X_2 Real)
(declare-const Y_0 Real)
(declare-const Y_1 Real)
(declare-const Y_2 Real)
(assert (>= X_0 20.0))
(assert (<= X_0 25.0))
(assert (>= X_1 23))
(assert (<= X_1 27))
(assert (>= X_2 8.0))
(assert (<= X_2 21.0))
(assert (>= Y_0 Y_1))
(assert (>= Y_0 Y_2))

Inputs

Outputs

Range of the inputs:
- \( 25 \geq x_0 \geq 20 \)
- \( 27 \geq x_1 \geq 23 \)
- \( 21 \geq x_2 \geq 8 \)

Property to verify:
- \( y_0 \geq y_1 \land y_0 \geq y_2 \)

It is possible that the selected mode is ventilation
NN verifier - nnenum

**Pros:**
- Very efficient
- Scalable

**Cons:**
- Limited to certain architectures
- No loops
- No complex properties

**Goal**
Find an example that satisfies all conditions, i.e., a set of \{x_0, x_1, x_2\} such that \((y_0 \geq y_1 \land y_0 \geq y_2)\)

**Property encoding (vnnlib)**

**Inputs**
- (declare-const X_0 Real)
- (declare-const X_1 Real)
- (declare-const X_2 Real)
- (declare-const Y_0 Real)
- (declare-const Y_1 Real)
- (declare-const Y_2 Real)
- (assert (>= X_0 20.0))
- (assert (<= X_0 25.0))
- (assert (>= X_1 23))
- (assert (<= X_1 27))
- (assert (>= X_2 8.0))
- (assert (<= X_2 21.0))

**Range of the inputs:**
\[
25 \geq x_0 \geq 20 \\
27 \geq x_1 \geq 23 \\
21 \geq x_2 \geq 8
\]

**Outputs**
- (declare-const X_0 Real)
- (declare-const X_1 Real)
- (declare-const X_2 Real)
- (assert (>= Y_0 Y_1))
- (assert (>= Y_0 Y_2))

**Property to verify:**
\[y_0 \geq y_1 \land y_0 \geq y_2\]

*It is possible that the selected mode is ventilation*
Pros:

- Very efficient
- Scalable

Property encoding (vnnlib)

(declare-const X_0 Real)
(declare-const X_1 Real)
(declare-const X_2 Real)
(declare-const Y_0 Real)
(declare-const Y_1 Real)
(declare-const Y_2 Real)
(assert (>= X_0 20.0))
(assert (<= X_0 25.0))
(assert (>= X_1 23))
(assert (<= X_1 27))
(assert (>= X_2 8.0))
(assert (<= X_2 21.0))
(assert (>= Y_0 Y_1))
(assert (>= Y_0 Y_2))

Cons:

- Limited to certain architectures
- No loops
- No complex properties

Goal

Find an example that satisfies all conditions, i.e., a set of \( \{x_0, x_1, x_2\} \) such that \( (y_0 \geq y_1 \land y_0 \geq y_2) \)

Execution

It is possible that the selected mode is ventilation
Pros:
- Very efficient
- Scalable

Cons:
- Limited to certain architectures
- No loops
- No complex properties

Property encoding (vnnlib)

```
(declare-const X_0 Real)
(declare-const X_1 Real)
(declare-const X_2 Real)
(declare-const Y_0 Real)
(declare-const Y_1 Real)
(declare-const Y_2 Real)
(assert (>= X_0 20.0))
(assert (<= X_0 25.0))
(assert (>= X_1 23))
(assert (<= X_1 27))
(assert (>= X_2 8.0))
(assert (<= X_2 21.0))
(assert (>= Y_0 Y_1))
(assert (>= Y_0 Y_2))
```

Inputs

Outputs

Range of the inputs:
- 25 ≥ x₀ ≥ 20
- 27 ≥ x₁ ≥ 23
- 21 ≥ x₂ ≥ 8

Property to verify:
- y₀ ≥ y₁ ∧ y₀ ≥ y₂

Goal

Find an example that satisfies all conditions, i.e., a set of \{x₀, x₁, x₂\} such that \(y₀ ≥ y₁ \land y₀ ≥ y₂\)

Execution

Evidence

It is possible that the selected mode is ventilation
Z3 and testing

Pipeline

NN in Python h5

Python script

NN in PLC code

Manually

NN in Python

Python API for Z3

Manually

testing

Z3

Manually

Assertions in Z3

Properties in natural language

Manually

Conditions

Manually

Assertions in natural language
Z3 and testing

**Z3**

**Pros:**
- Relatively flexible
- Slightly efficient

**Cons:**
- Unwind all loops manually
- Difficult to express complex properties
Z3 and testing

**Z3**

**Pros:**
- Relatively flexible
- Slightly efficient

```python
s = Solver()
s.add(input_layer[0]>=200)
s.add(input_layer[0]<=250)
s.add(input_layer[1]>=230)
s.add(input_layer[1]<=270)
s.add(input_layer[2]>=80)
s.add(input_layer[2]<=210)
s.add(max_mode==0)
if s.check()==sat:
    print(s.model())
```

**Cons:**
- Unwind all loops manually
- Difficult to express complex properties

```python
i=0
temp=0
j=0
temp = temp + eigth_layer_output[j] * mode_layer_weights.Data[j][i]
mode_layer_output[i] = If(0> temp + mode_layer_bias.Data[i],0, temp + mode_layer_bias.Data[i])
j=1
mode_layer_output[i] = If(0> temp + mode_layer_bias.Data[i],0, temp + mode_layer_bias.Data[i])
j=2
mode_layer_output[i] = If(0> temp + mode_layer_bias.Data[i],0, temp + mode_layer_bias.Data[i])
```
Z3 and testing

Z3

Pros:
• Relatively flexible
• Slightly efficient

Cons:
• Unwind all loops manually
• Difficult to express complex properties

Pros: Exhautive testing

• Relatively flexible
• Efficient when low number of inputs

Cons:
• Not scalable
• Finite set of tests

s = Solver()
s.add(input_layer[0]>=200)
s.add(input_layer[0]<=250)
s.add(input_layer[1]>=230)
s.add(input_layer[1]<=270)
s.add(input_layer[2]>=80)
s.add(input_layer[2]<=210)
s.add(max_mode==0)
if s.check()==sat:
    print(s.model())
Roadmap

- Neural networks
  - Verification of Neural networks
    - Verification of neural-network based controllers
    - Different approaches to verify those properties
- Neural-network based controllers
- Case study
  - Properties to verify
- Conclusion
Conclusion
Conclusion

- Important to verify neural networks

Dog  + noise  Ostrich
Conclusion

- Important to verify neural networks

- Possible to verify neural networks in multiple platforms
Conclusion

- Important to verify neural networks
- Possible to verify neural networks in multiple platforms
- Verify as much as possible in the final system
- Use specialized tools if more efficiency is needed

<table>
<thead>
<tr>
<th>Tool</th>
<th>Performance</th>
<th>Scalability</th>
<th>Expressiveness</th>
<th>Same types?</th>
<th>Plug-and-play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLCverif</td>
<td>low</td>
<td>low</td>
<td>high</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>nmenum</td>
<td>very high</td>
<td>high</td>
<td>low</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Z3</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

Testing:
- high
- very low
- medium
Conclusion

- Important to verify neural networks
- Possible to verify neural networks in multiple platforms
- Verify as much as possible in the final system
- Use specialized tools if more efficiency is needed

Paper to appear in the proceeding of the **Engineering Applications and Advances of Artificial Intelligence 2023** conference: [https://eannconf.org/2023/](https://eannconf.org/2023/). Software can be found here: [https://doi.org/10.48436/fww3h-2y402](https://doi.org/10.48436/fww3h-2y402).
Conclusion

• Important to verify neural networks

• Possible to verify neural networks in multiple platforms

• BE-ICS is planning to keep working in this direction

• Verify as much as possible in the final system
• Use specialized tools if more efficiency is needed

<table>
<thead>
<tr>
<th>PLCverif</th>
<th>nmenum</th>
<th>Z3</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>very high</td>
<td>low</td>
<td>no</td>
</tr>
<tr>
<td>medium</td>
<td>high</td>
<td>very low</td>
<td>no</td>
</tr>
<tr>
<td>high</td>
<td>low</td>
<td>medium</td>
<td>no</td>
</tr>
<tr>
<td>same types?</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>plug-and-play?</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>

Conclusion

• Important to verify neural networks
• Possible to verify neural networks in multiple platforms
• BE-ICS is planning to keep working in this direction
• Contact:
  – ignacio.lopez@tuwien.ac.at
  – https://ignaciolopezmiguel.github.io/

• Verify as much as possible in the final system
• Use specialized tools if more efficiency is needed

<table>
<thead>
<tr>
<th>PLCverify</th>
<th>mnenum</th>
<th>Z3</th>
<th>Testing</th>
<th>performance</th>
<th>scalability</th>
<th>expressiveness</th>
<th>same types?</th>
<th>plug-and-play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>very high</td>
<td>medium</td>
<td>high</td>
<td>low</td>
<td>no</td>
<td>low</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>medium</td>
<td>high</td>
<td>low</td>
<td>medium</td>
<td>low</td>
<td>no</td>
<td>low</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>medium</td>
<td>very low</td>
<td>medium</td>
<td>high</td>
<td>medium</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

Conclusion

Thank you! Questions?

• Important to verify neural networks

• Possible to verify neural networks in multiple platforms

• Possible to verify neural networks

• BE-ICS is planning to keep working in this direction

• Contact:
  – ignacio.lopez@tuwien.ac.at
  – https://ignaciolopezmiguel.github.io/

• Verify as much as possible in the final system

• Use specialized tools if more efficiency is needed

<table>
<thead>
<tr>
<th></th>
<th>performance</th>
<th>scalability</th>
<th>expressiveness</th>
<th>same types?</th>
<th>plug-and-play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLCverif</td>
<td>low</td>
<td>low</td>
<td>high</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>nnenum</td>
<td>very high</td>
<td>low</td>
<td>low</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Z3</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Testing</td>
<td>high</td>
<td>very low</td>
<td>medium</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>


Verification of neural networks meets PLC code: An LHC cooling tower control system at CERN

Ignacio D. Lopez-Miguel1, Borja Fernández Arlego2, Faiq Ghuswah3, and Enrique Blanco Viñuela2

1 TU Wien, Vienna, Austria ignacio.lopez@tuwien.ac.at
2 European Organization for Nuclear Research (CERN), Geneva, Switzerland borja.fernandez.arlego@cern.ch
3 Norwegian University of Science and Technology, Trondheim, Norway faiq.ghuswahtnm.no

Abstract. In the last few years, control engineers have started to use artificial neural networks (NNs) embedded in advanced feedback control algorithms. Its natural integration into existing control systems, such as programmable logic controllers (PLCs) or close to them, represents a challenge. Besides, the application of these algorithms in critical applications still raises concerns among control engineers due to the lack of safety guarantees. Building trustworthy NNs is still a challenge and their verification is attracting more attention nowadays. This paper discusses the peculiarities of formal verification of NNs controllers running on PLCs. It outlines a set of properties that should be satisfied by a NN that is intended to be deployed in a critical high-safety installation at CERN. It compares different methods to verify this NN and sketches our future research directions to find a safe NN.