Abstraction-based Reduction of Input Size for Neural Networks[1]

Peter Backeman, **Edin Jelačić**, Cristina Seceleanu, Ning Xiong, and Tiberiu Seceleanu

Mälardalen University, School of Innovation, Design and Engineering, Västerås, Sweden

26/10 2023 - AISoLA

HITACHI

ERICSSON

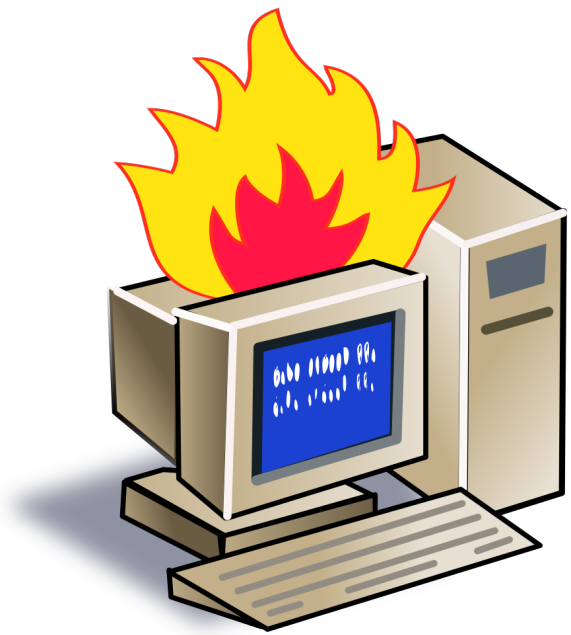
Mälardalen University

[1]Supported by the Knowledge Foundation

Use Case

Use Case: Predicting CPU overload.

- We have a set of functions $F = \{f_1, \ldots, f_n\}$.
- For a particular subset of F (a configuration), we can measure a CPU load.
- Question: for what configurations is there risk for an overload?

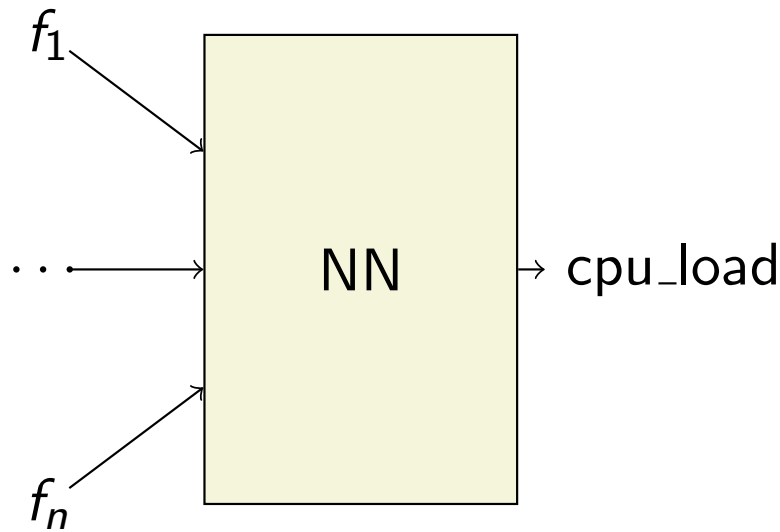


Mälardalen University

Modeling

We model the cpu load for a configuration using a neural network.

- Input: activated functions (e.g., $f_1 = 1, f_2 = 0, \ldots$)
- Output: estimated cpu load.



- Important: we do not assume monotonicity (e.g., activating a function *can reduce* load)

Mälardalen University

Problem & Opportunities

We obatin such a NN, which has many inputs and is a black box model.

- Hard to check all combinations.
- Not clear the cause of overload.

Problem & Opportunities

We obatin such a NN, which has many inputs and is a black box model.

- Hard to check all combinations.
- Not clear the cause of overload.

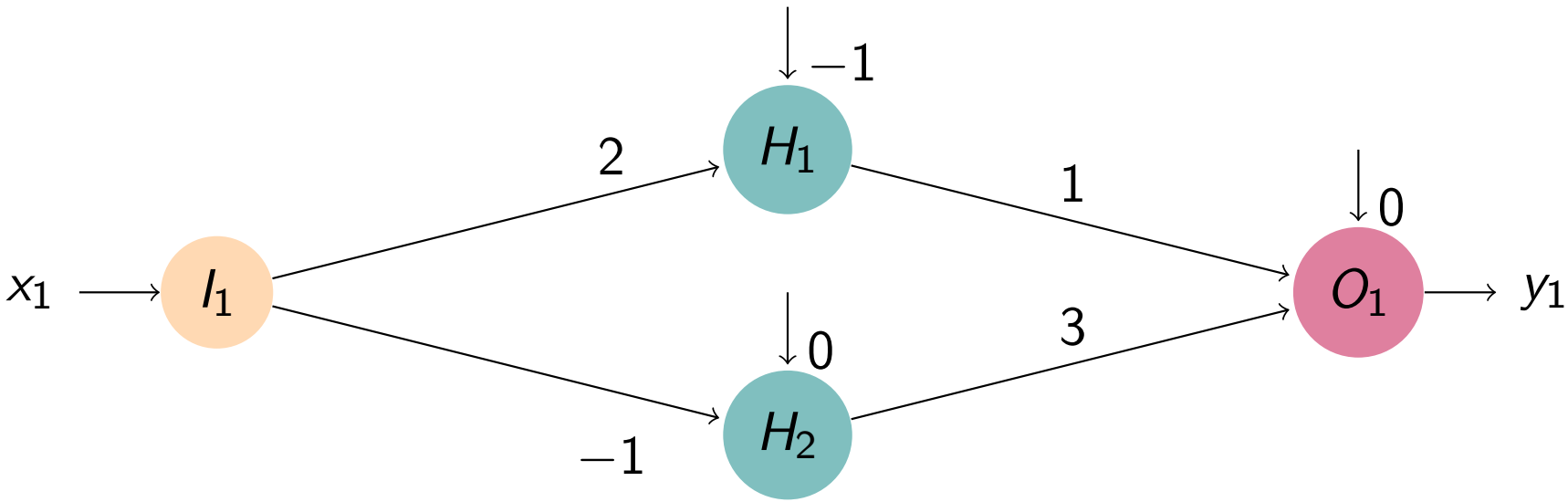
Identifying insignficant inputs.

- Are some inputs *insignificant*?
- Can we remove them?
- What is the effect on the network? Can it be bounded?

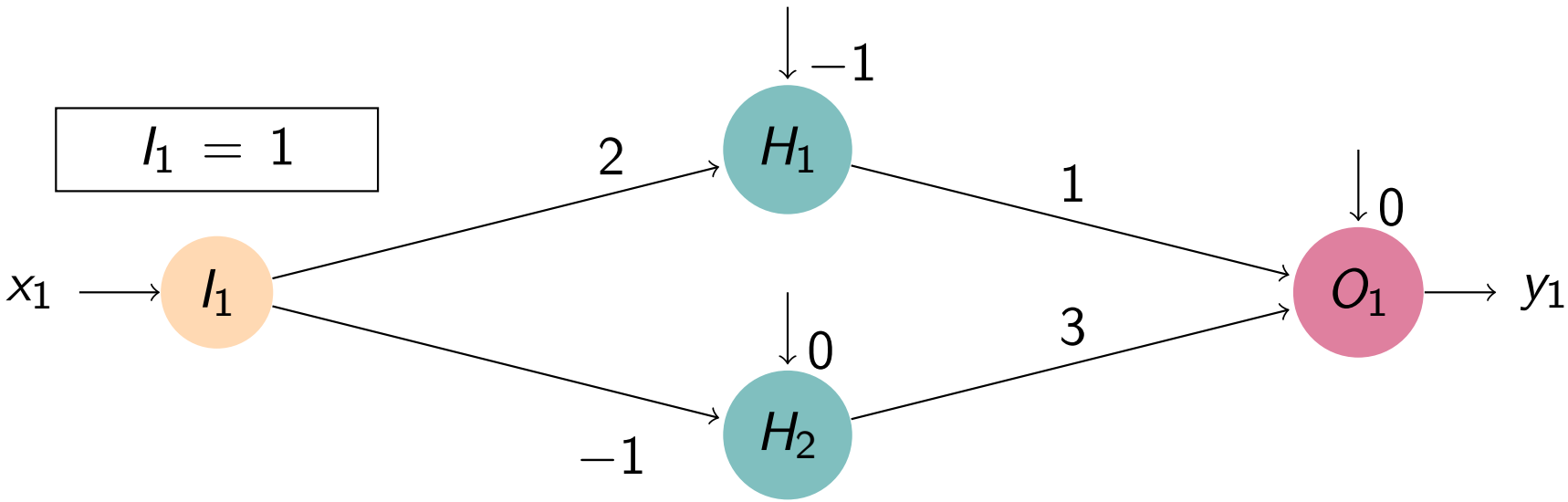
Mälardalen University

Neural Networks

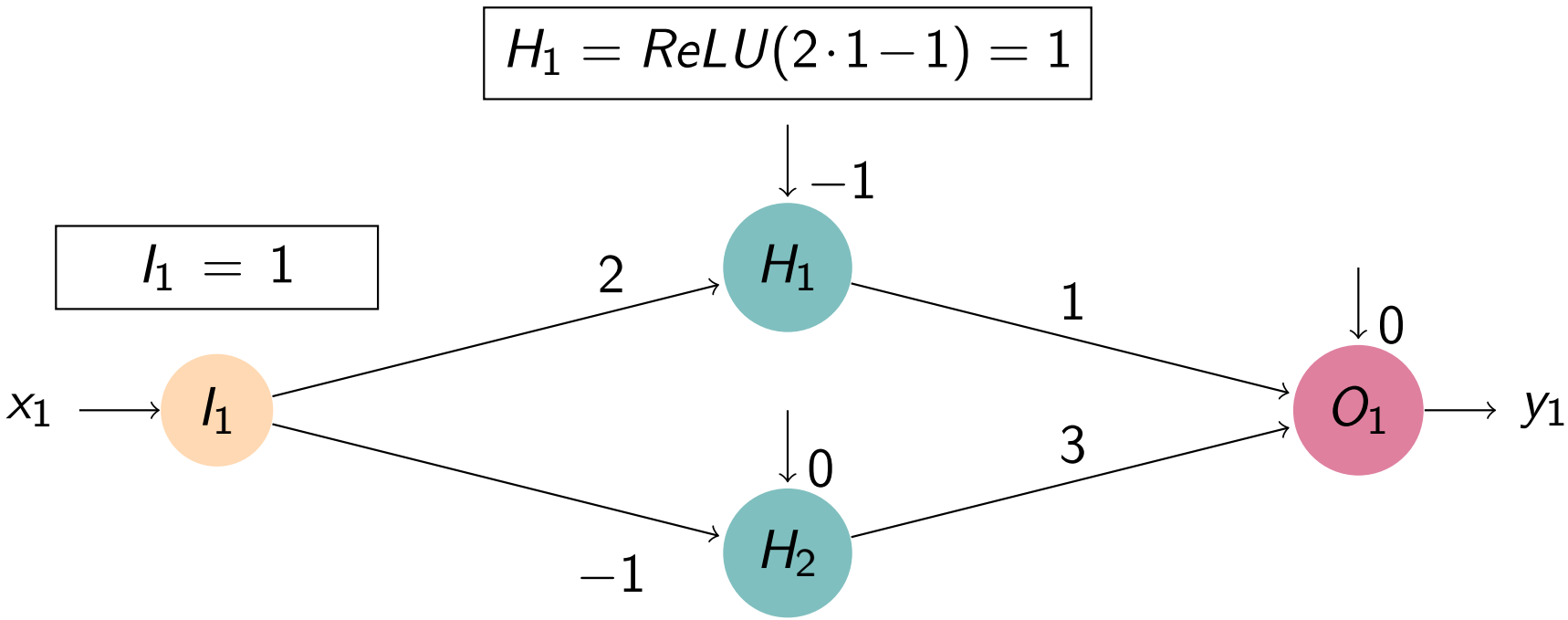
Neural Networks



Neural Networks

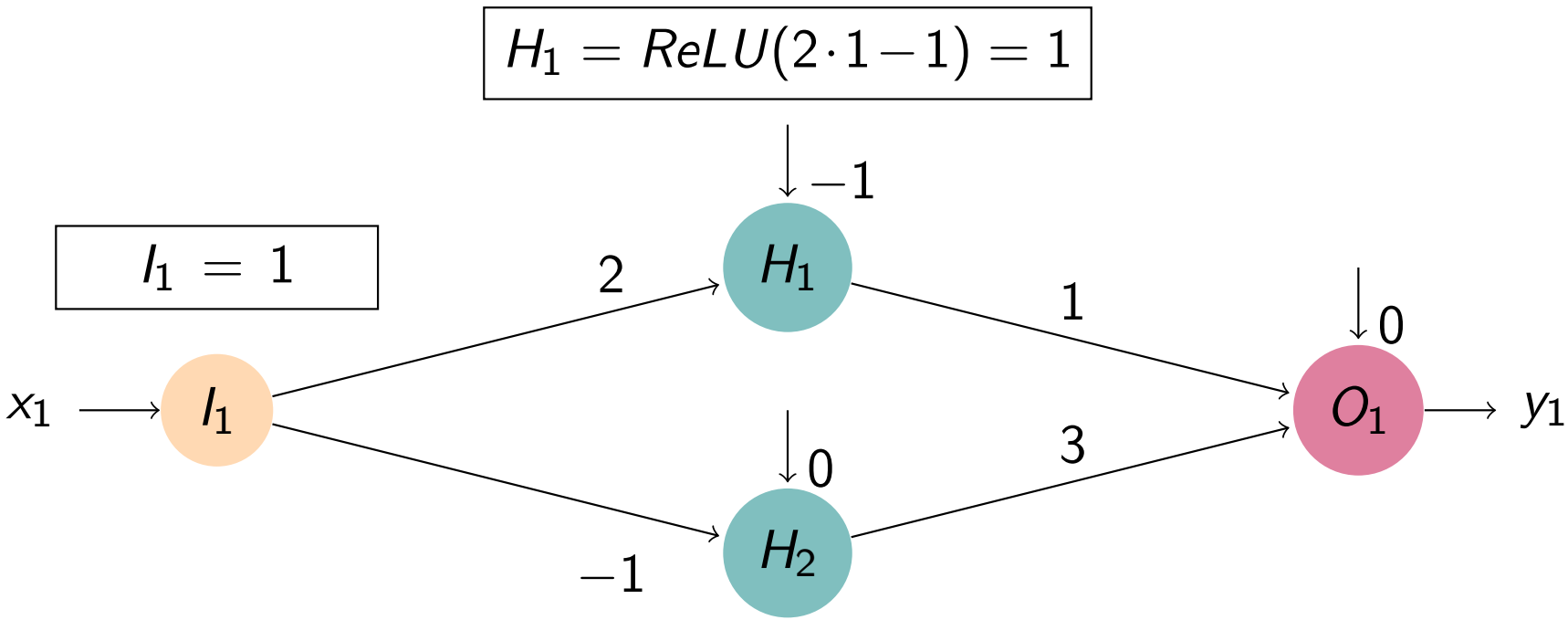


Neural Networks



Mälardalen University

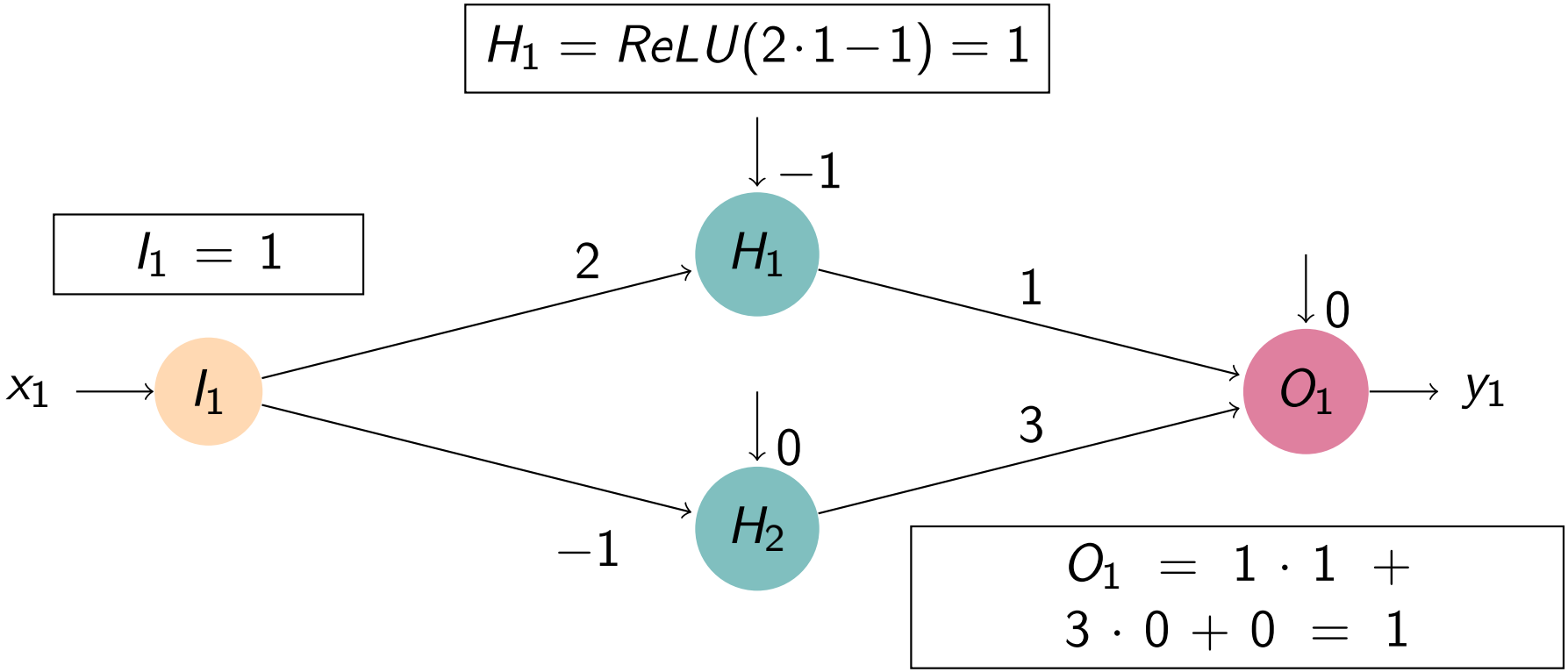
Neural Networks



$$H_2 = ReLU(-1 \cdot 1 + 0) = 0$$

Mälardalen University

Neural Networks



Mälardalen University

NN Verification

Tools for verifying NN.

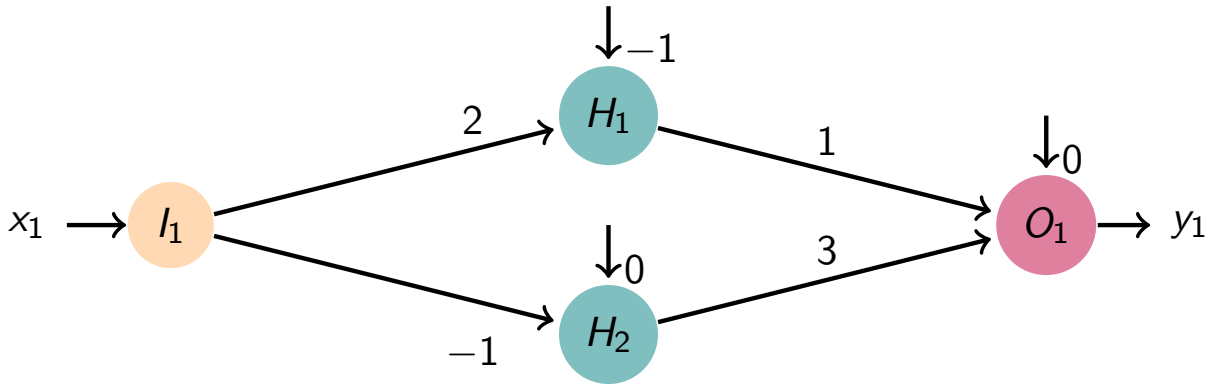
- Marabou can check properties of NN
- For example: establish (linear) upper bound on output node
- Under some constraints on input

Mälardalen University

NN Verification

Tools for verifying NN.

- Marabou can check properties of NN
- For example: establish (linear) upper bound on output node
- Under some constraints on input



For example, Marabou can show that $O_1(x) \leq 1$, for all $x \in [0, 1]$.

Mälardalen University

Abstracting for Over-approximation

Abstracting for Over-approximation

- We are considering the establishment of an *upper bound*.
- This means, we can replace a network $\mathcal{NN}$ with an abstracting $\mathcal{NN}'$ if:

$$\forall x. \mathcal{NN}(x) \leq \mathcal{NN}'(x)$$

- If we find a upper bound U for $\mathcal{NN}'(x)$, then:

$$\forall x. \mathcal{NN}(x) \leq \mathcal{NN}'(x) \leq U$$

Mälardalen University

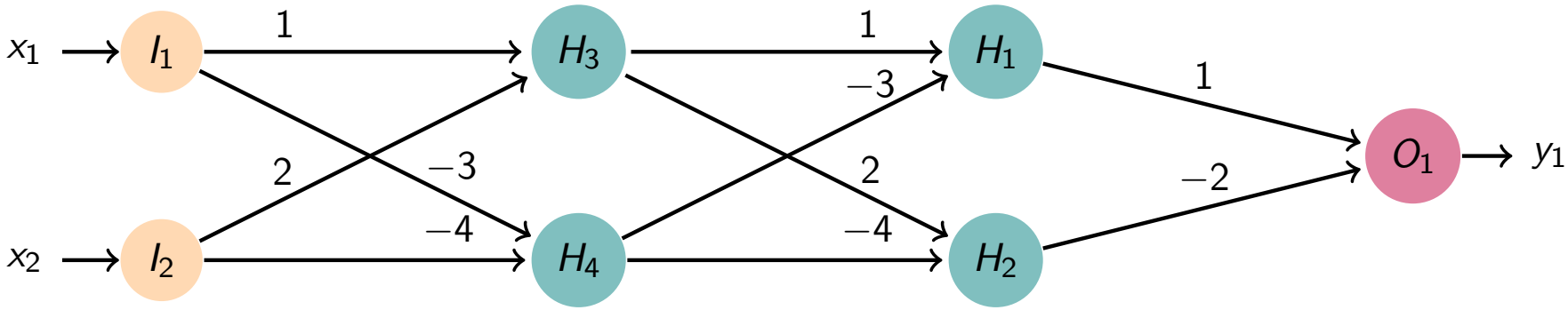
Abstraction idea presented in:
Yizhak Yisrael Elboher, Justin Gottschlich, and Guy Katz. "An abstraction-based framework for neural network verification". In: *Computer Aided Verification: 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21–24, 2020, Proceedings, Part I 32*. Springer. 2020, pp. 43–65:

Classify nodes into increasing and decreasing nodes

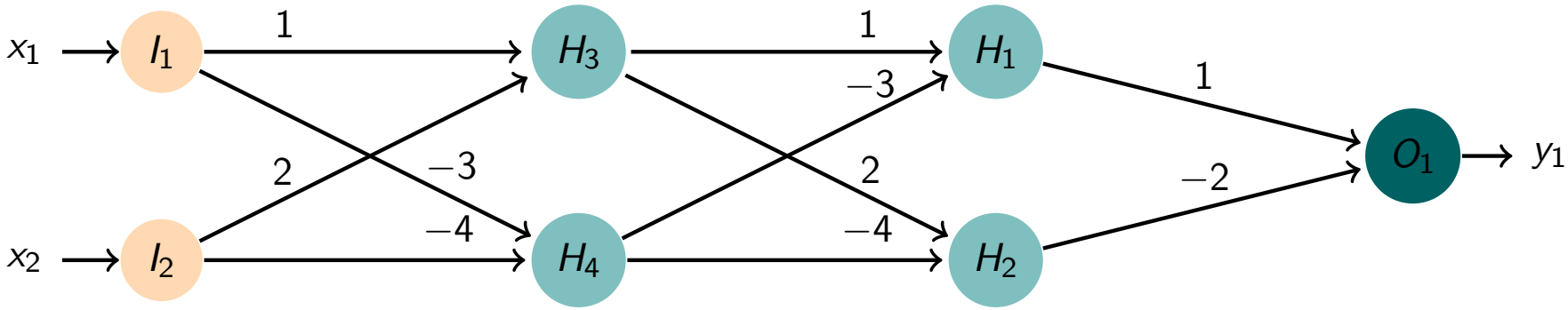
We demonstrate through an example.

Mälardalen University

Abstraction Example

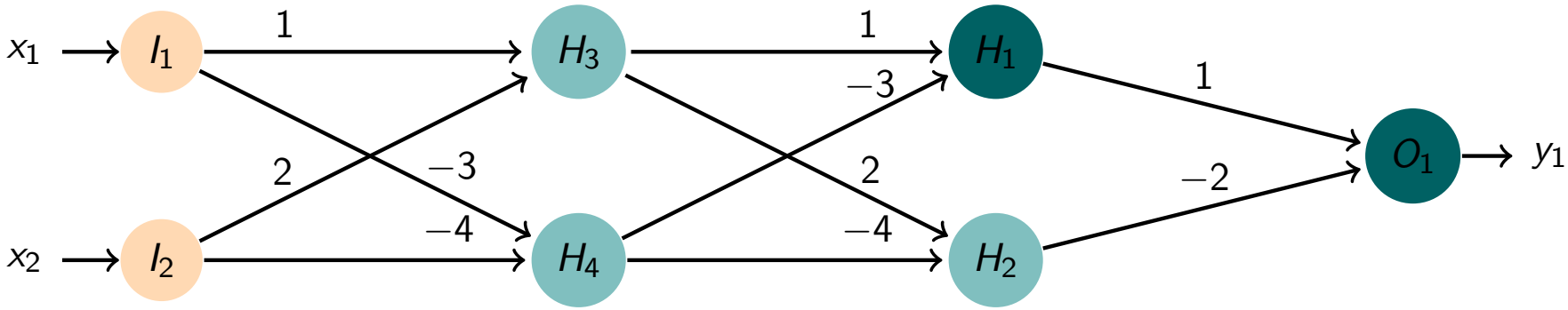


Abstraction Example



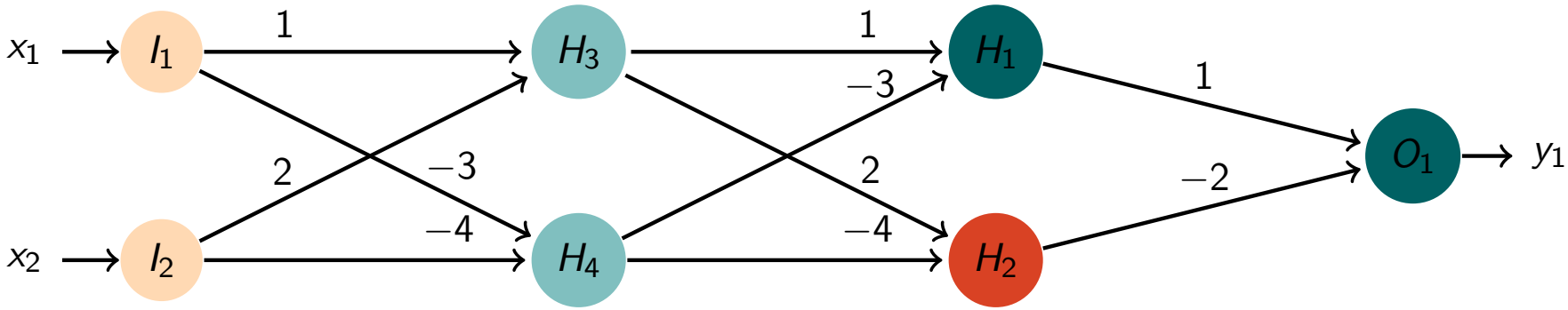
Mälardalen University

Abstraction Example



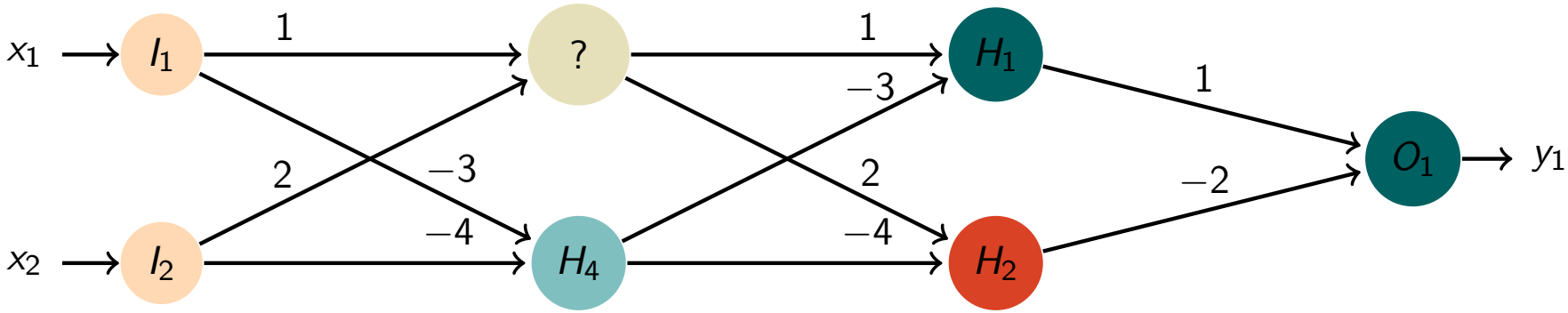
Mälardalen University

Abstraction Example



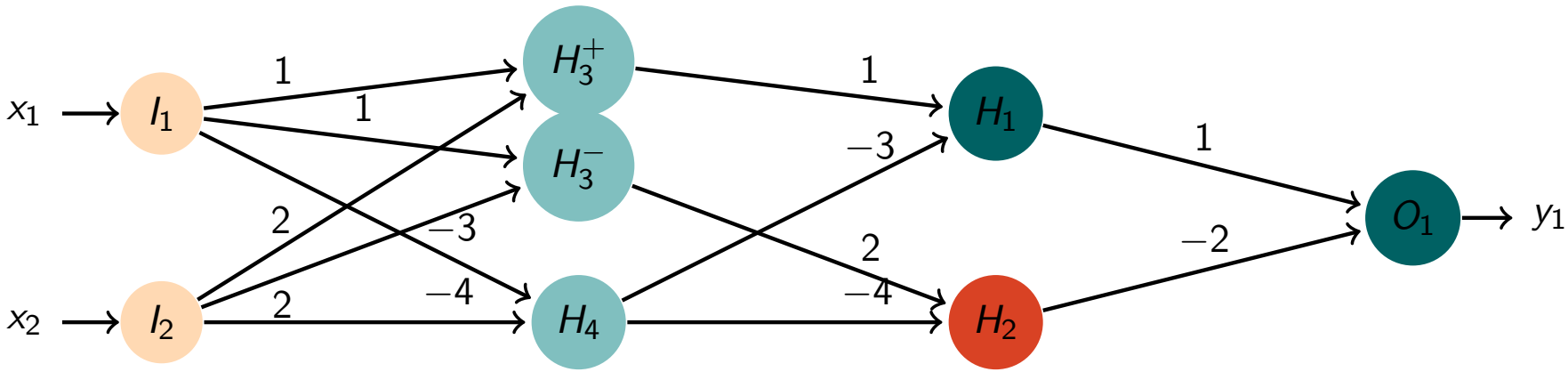
Mälardalen University

Abstraction Example



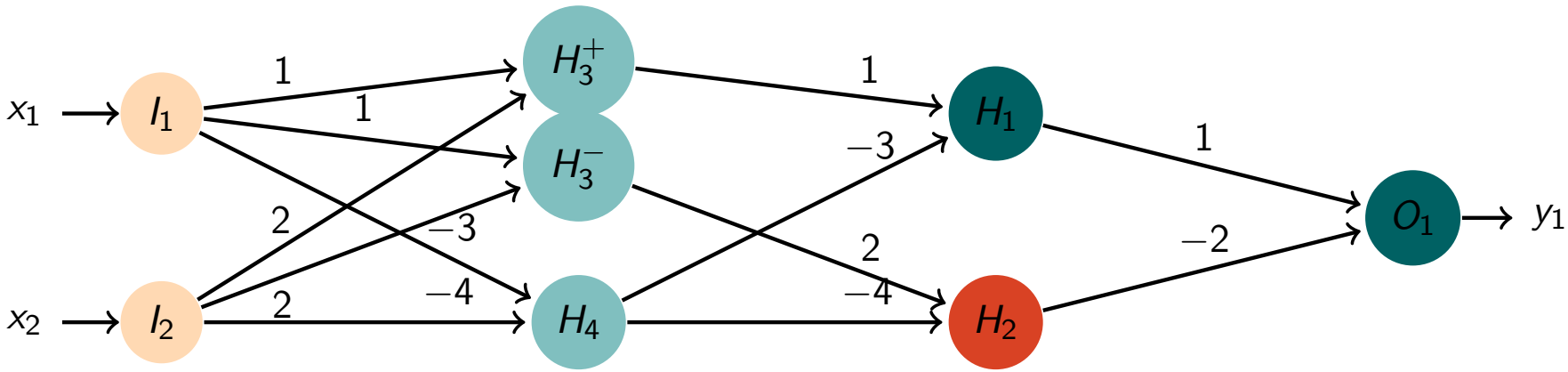
Let's copy the node!

Abstraction Example



Mälardalen University

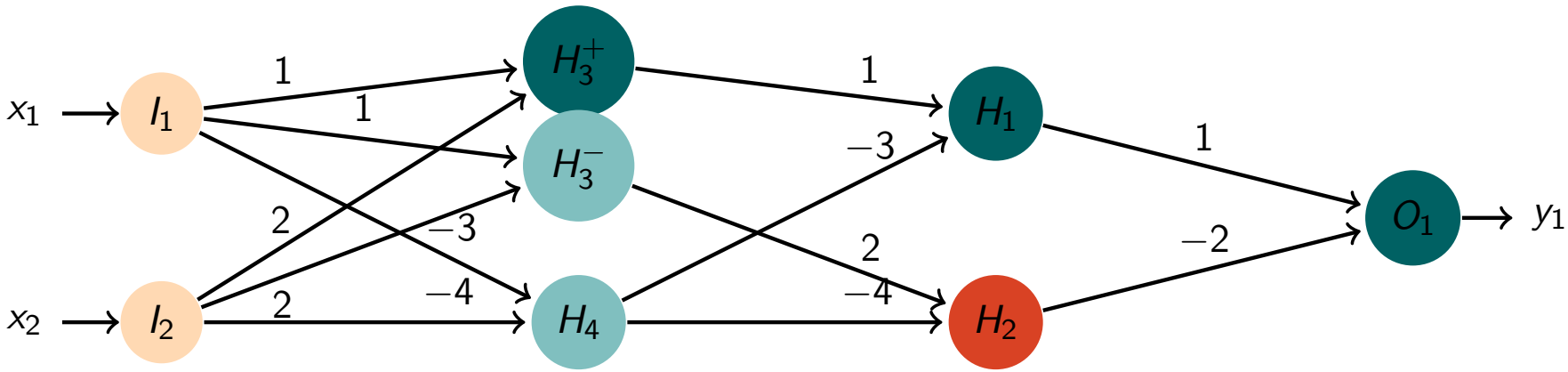
Abstraction Example



This computes the exact same function.

Mälardalen University

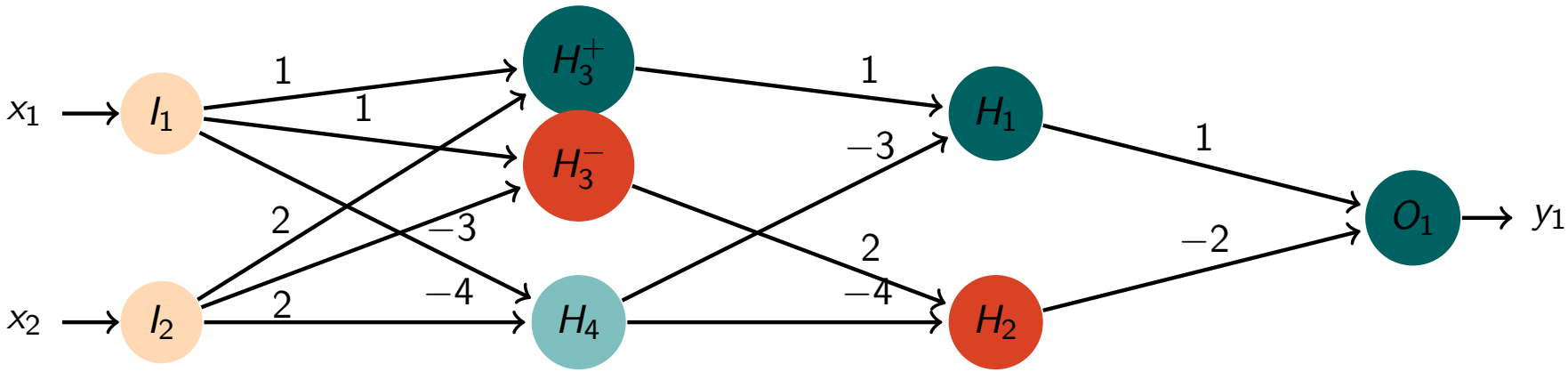
Abstraction Example



This computes the exact same function.

Mälardalen University

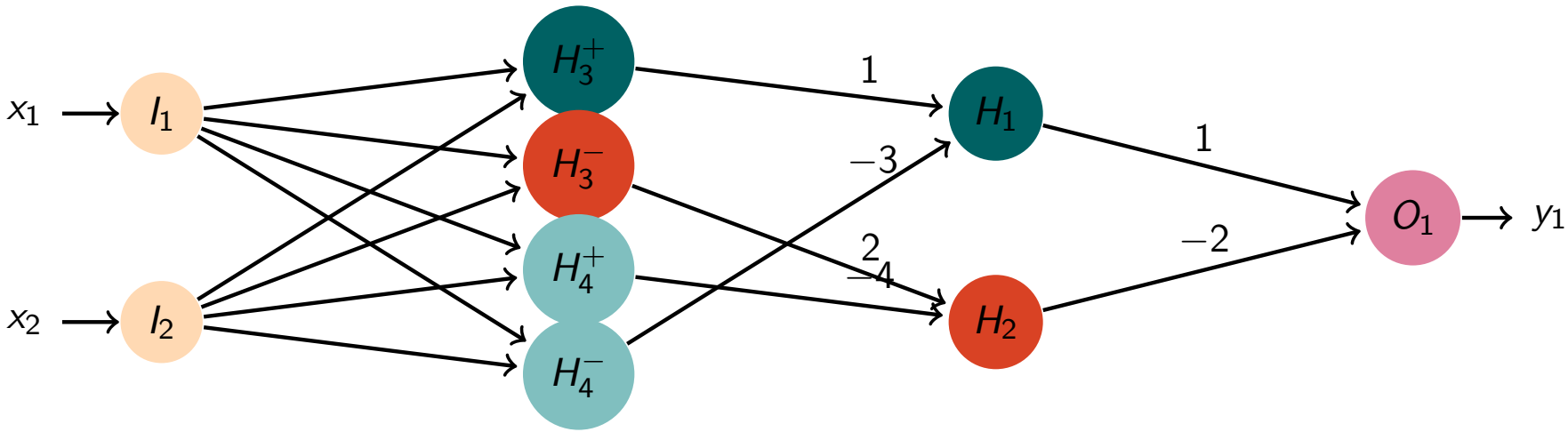
Abstraction Example



This computes the exact same function.

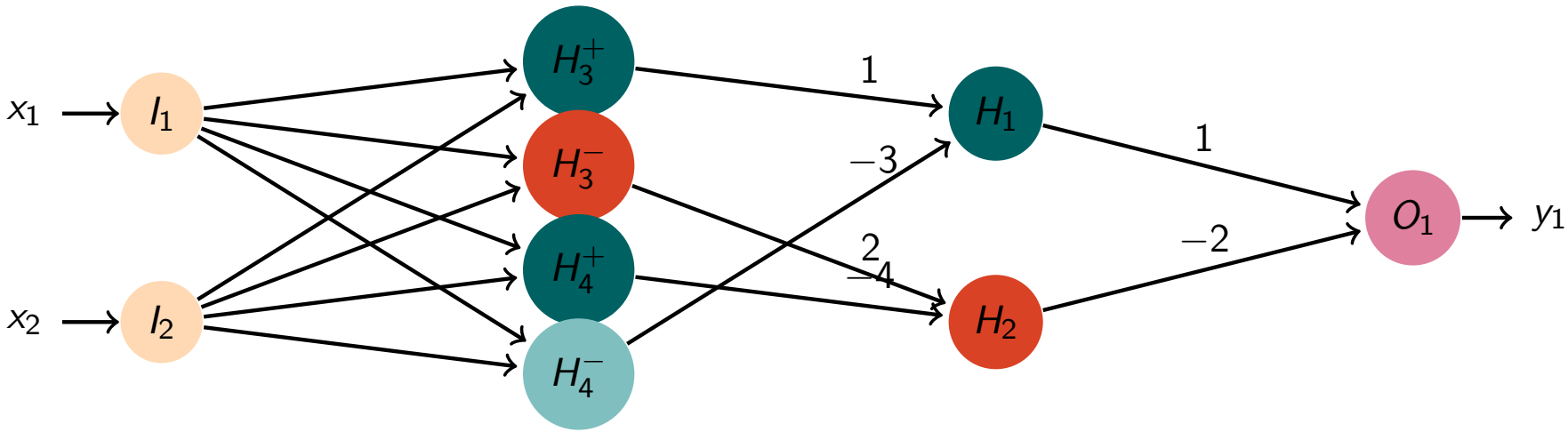
Mälardalen University

Abstraction Example



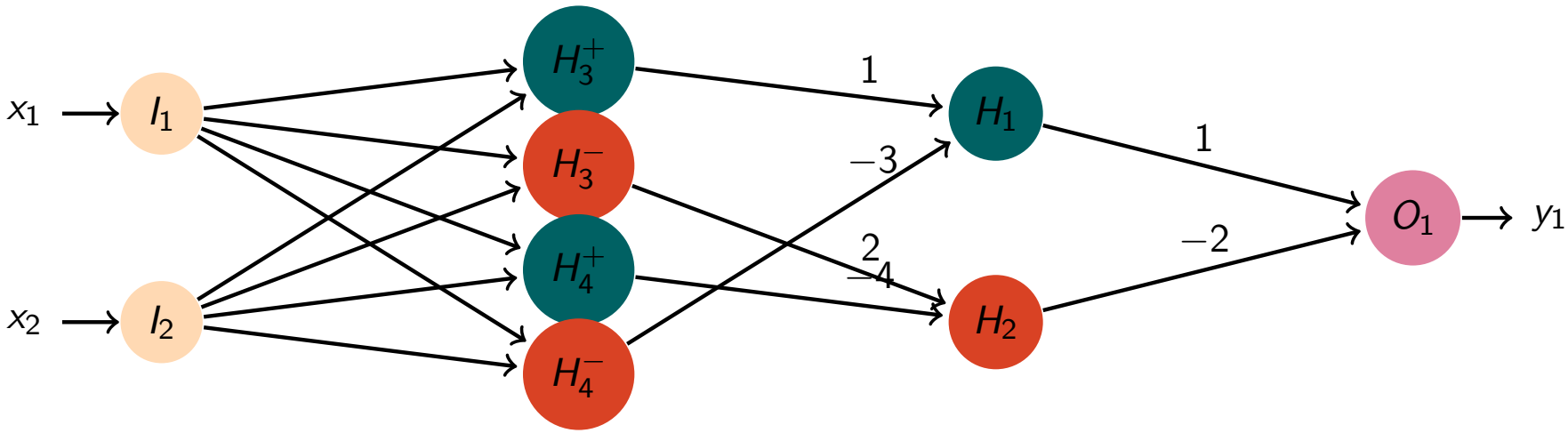
This computes the exact same function.

Abstraction Example



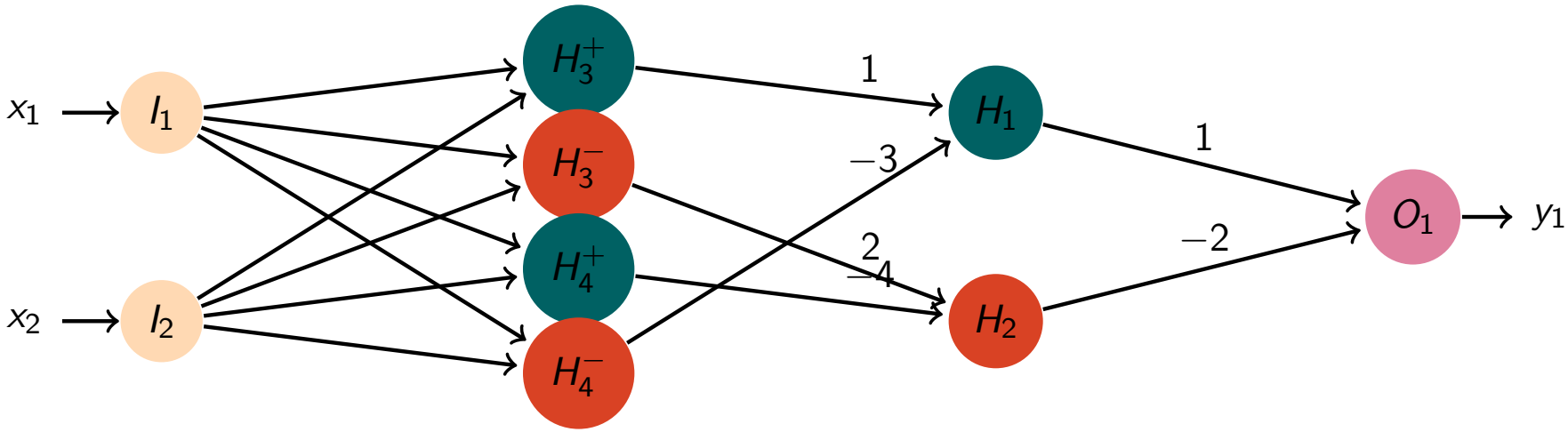
This computes the exact same function.

Abstraction Example



This computes the exact same function.

Abstraction Example

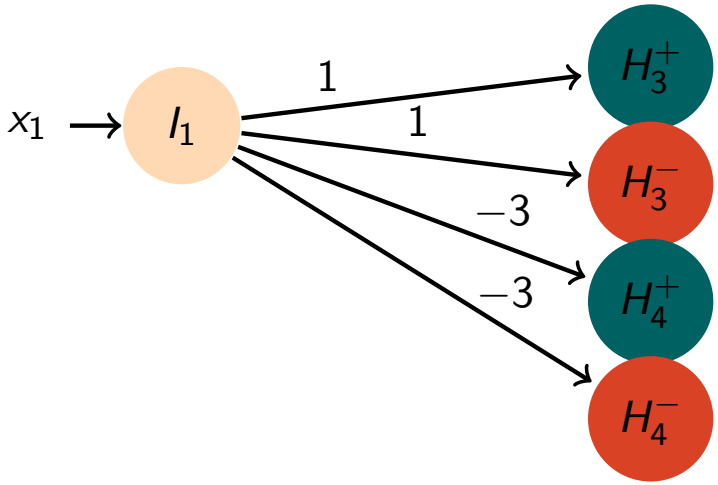


This computes the exact same function.

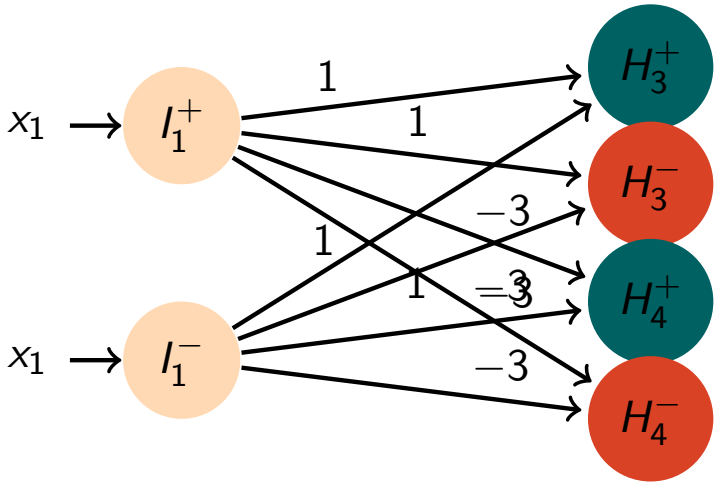
- In this manner we can color each hidden node in network
- Might double number of nodes
- Our proposal: extend this to input layer

Mälardalen University

Abstract input layer

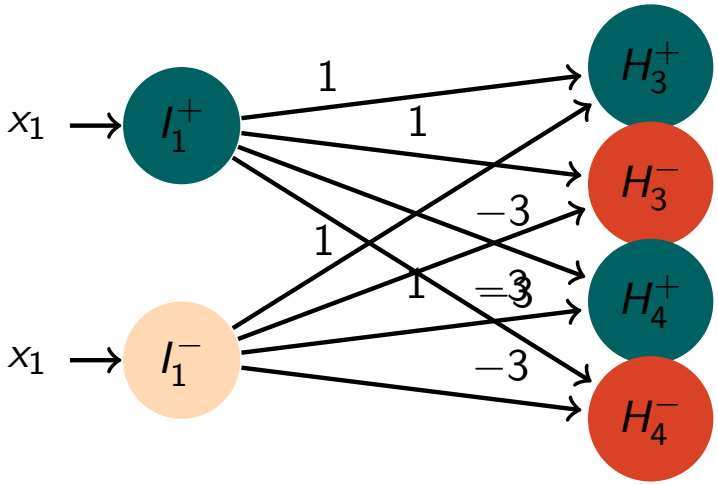


Abstract input layer

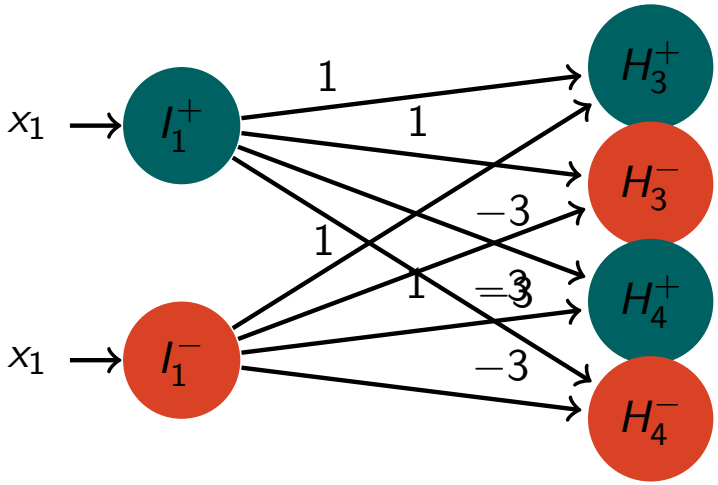


Mälardalen University

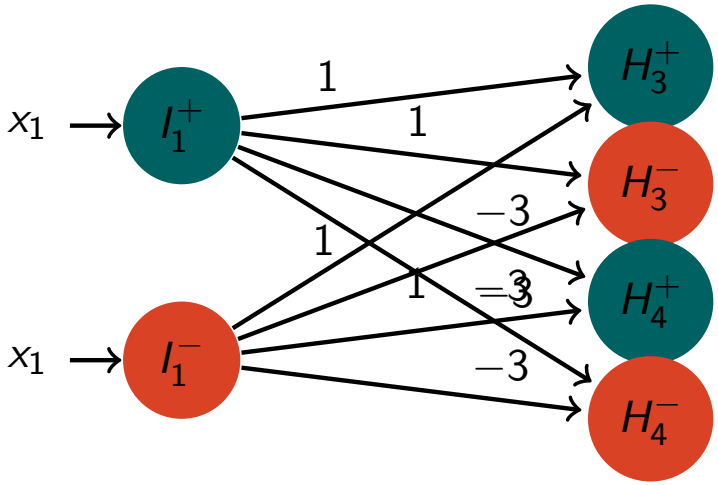
Abstract input layer



Abstract input layer



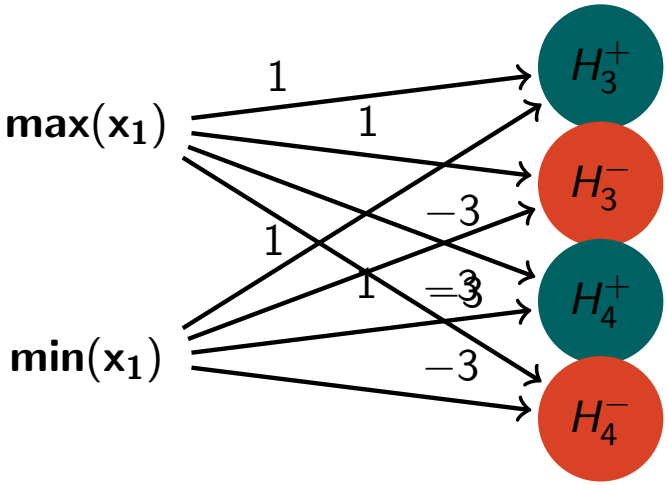
Abstract input layer



- I_1^+ is the increasing component of I_1
- We set it to the maxmimum value of x_1
- Similarily, I_1^- is set to the minimum value of x_1

Mälardalen University

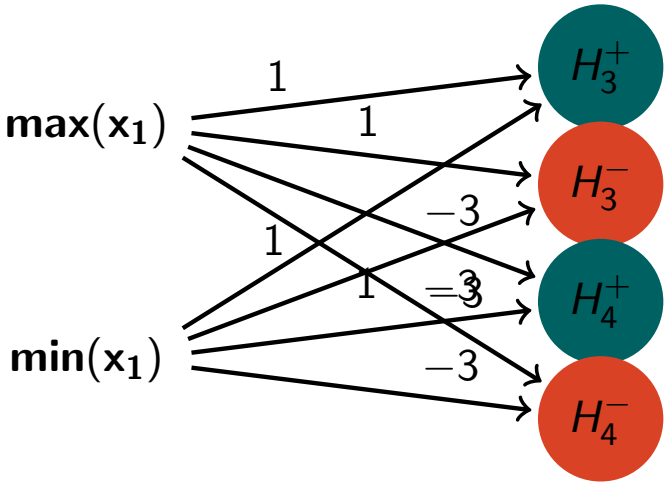
Abstract input layer



- I_1^+ is the increasing component of I_1
- We set it to the maxmimum value of x_1
- Similarily, I_1^- is set to the minimum value of x_1

Mälardalen University

Abstract input layer



- I_1^+ is the increasing component of I_1
- We set it to the maxmimum value of x_1
- Similarily, I_1^- is set to the minimum value of x_1
- Now we have removed input I_1!

Mälardalen University

Removing one input

- In this manner we can remove an input and I_i and create an over-approximating network $\mathcal{NN}_i^+$, s.t.

$$\mathcal{NN}(x) \leq \mathcal{NN}_i^+(x')$$

  for all x, x' (where x' is x with x_i removed)
- $\mathcal{NN}_i^+$ is throwing x_i out of the window, but *at what cost*?

Mälardalen University

Bounding approximation cost

- In the same manner as we create $\mathcal{NN}_i^+$, we can create an *under-approximating* network $\mathcal{NN}_i^-$.
- We can use the combination to bound the approximation cost.

Bounding approximation cost

$$\mathcal{NN}_i^+$$

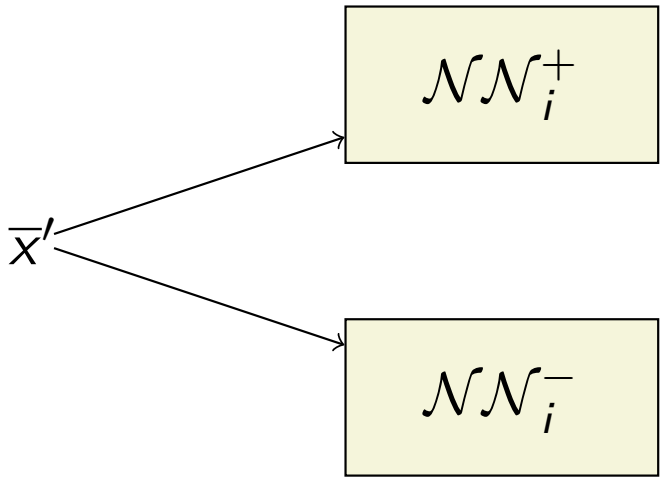
Bounding approximation cost

$\mathcal{NN}_i^+$

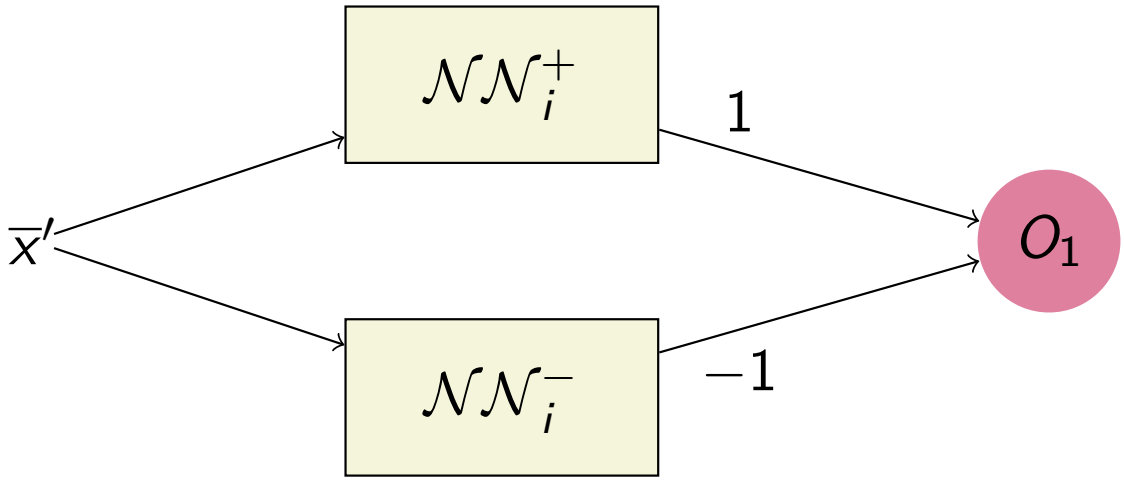
$\mathcal{NN}_i^-$

Mälardalen University

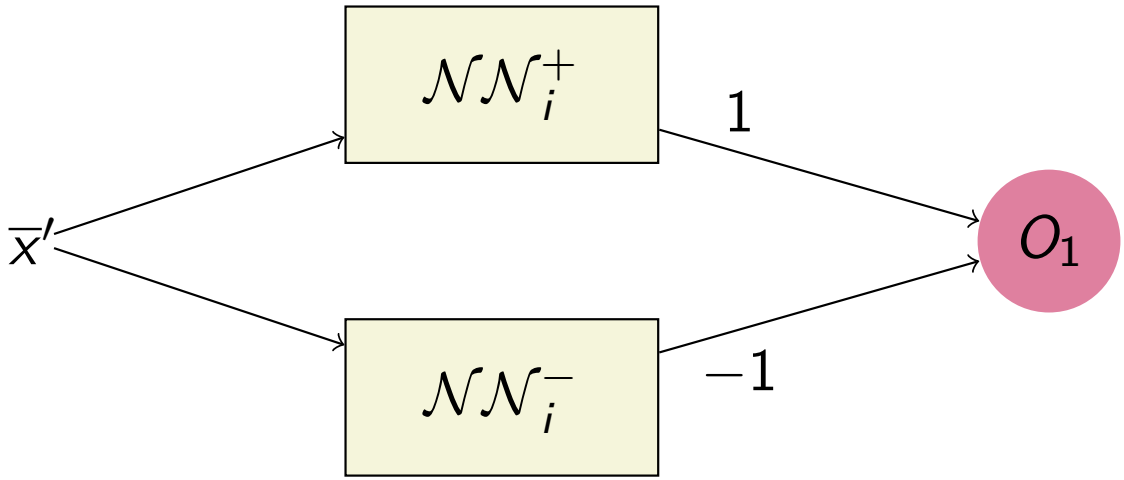
Bounding approximation cost



Bounding approximation cost



Bounding approximation cost



- O_1 gives the an over-approximation of the *maximum* effect of x_i.
- Let's call this network $\mathcal{NN}_i^?$.

Mälardalen University

Bounding approximation cost

- We have a network $\mathcal{NN}_i^?$ giving the over-approximation.
- We use Marabou to establish an upper bound:

$$\mathcal{NN}_i^? \leq L$$

  for some limit L.
- If true, x_i affects output of network by at most L
- If L small, we call x_i *insignficant*.

Mälardalen University

Use Case

Use Case: Predicting CPU overload.

- We have a set of functions $F = \{f_1, \ldots, f_n\}$.
- For a particular subset of F (a configuration), we can measure a CPU load.
- Question: for what configurations is there risk for an overload?

Mälardalen University

Use Case

Use Case: Predicting CPU overload.

- We have a set of functions $F = \{f_1, \ldots, f_n\}$.
- For a particular subset of F (a configuration), we can measure a CPU load.
- Question: for what configurations is there risk for an overload?

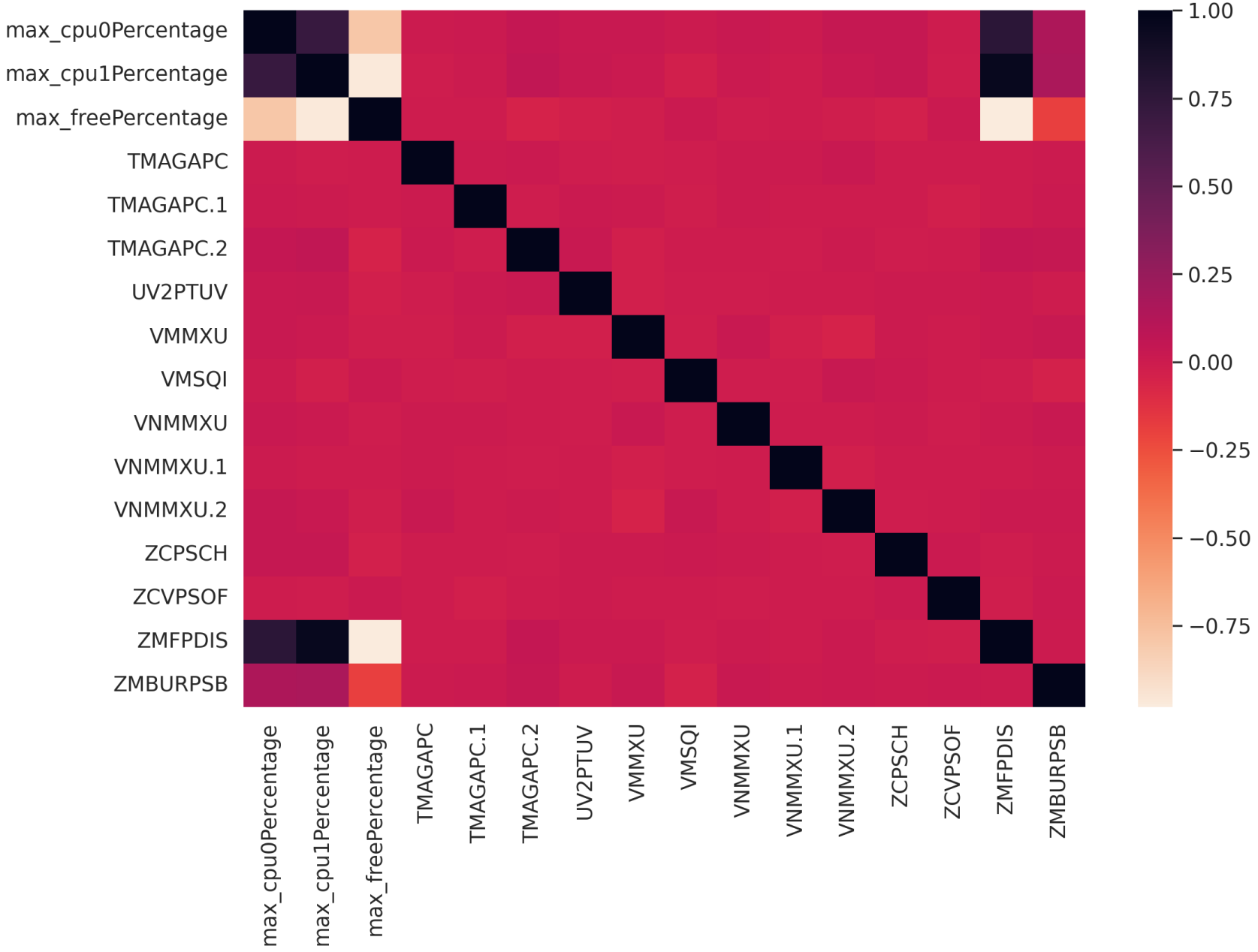
- We can remove an input f_i and (maybe) find a limit L.
- This means that function f_i can be ignored if we add L to the final load.

Mälardalen University

(Simple) Validation

- We harvest data from embedded system
- Roughly 3000 samples for 13 functions
- Train neural network (10 hidden nodes)

Correlation



Our results

- We try to idenity each function and see if it is *insignificant*
- A limit of 2 identifies $\mathrm{ZMFPDIS}$ and $\mathrm{ZMBURPSB}$ as significant inputs.
- A limit of 1 only identifies four insignficant inputs.
- A limit of 0.5 identifies no insignficant inputs.

Mälardalen University

Conclusions

- Process non-deterministic
- Finding bound requires a guess

Mälardalen University

Conclusions

- Process non-deterministic
- Finding bound requires a guess

- We can extract information of model (but is model correct?)
- We can reduce complexity (inputs) of the model, by adding a constant to the output

Mälardalen University

Conclusions

- Process non-deterministic
- Finding bound requires a guess

- We can extract information of model (but is model correct?)
- We can reduce complexity (inputs) of the model, by adding a constant to the output

- Continue validation
- Consider tuning training of network

Mälardalen University

References

[1] Yizhak Yisrael Elboher, Justin Gottschlich, and Guy Katz. "An abstraction-based framework for neural network verification". In: *Computer Aided Verification: 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21–24, 2020, Proceedings, Part I 32*. Springer. 2020, pp. 43–65.

Mälardalen University