Over-estimation guarantees for Surrogate Neural Networks for Braking Distance Estimation

High probabilistic guarantees and formal verification

Ducoffe Melanie

Nov 22 th 2022

The results of multiple collaborations

ONERA
THE FRENCH AEROSPACE LAB

C. Pagetti

IRIT

I. Ober

I. Ober

AIRBUS

AIR
Airbus AI Research

J. Sen Gupta

E.Sudre

A. Gauffriau

G. Vidot

C. Gabreau

DEEL
DEpendable & Explainable Learning

ANITI
ARTIFICIAL & NATURAL INTELLIGENCE
TOULOUSE INSTITUTE

F. Malgouyres

S. Gerchinovitz

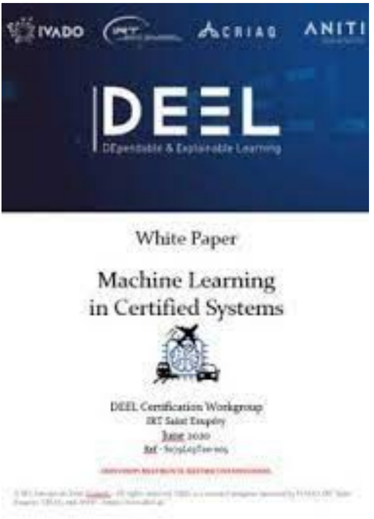
Trustworthiness of AI in the transport industry

#Computer Vision
OneAtlas
#Times Series Analysis
skywise.
Fortion® Massive Intelligence
Focus Human Expertise on Added Value Tasks
#NLP

WE ARE HERE

On Ground
Not Critical

On Board
Critical



IVADO IRT ACRIAQ ANITI

DEEL

DEpendable & Explainable Learning

White Paper

Machine Learning
in Certified Systems

DEEL Certification Workgroup
IRT Saint Exupéry
June 2020
Ref - [illegible]

EUROCAE

EASA
European Union Aviation Safety Agency

Industrial needs

Challenges

- **Qualified models developed by engineering are usually ‘heavy’:**
  - Memory
  - Computing Power
  - Computing time

- **Embedding complex reference models for assistance:** Braking Distance Estimation, Structure Load Estimation, etc
- Surrogate is necessary for embedding reference physics models validated by authorities: How can we assess the safety of the surrogate compared to its reference ?

EASA: 41% accidents involving small non commercial airplanes happen during landing (1991-2017)

EUROCAE ED250: Minimum Operational Performance Standard for a Runway Overrun Awareness and Alerting System”, 2017

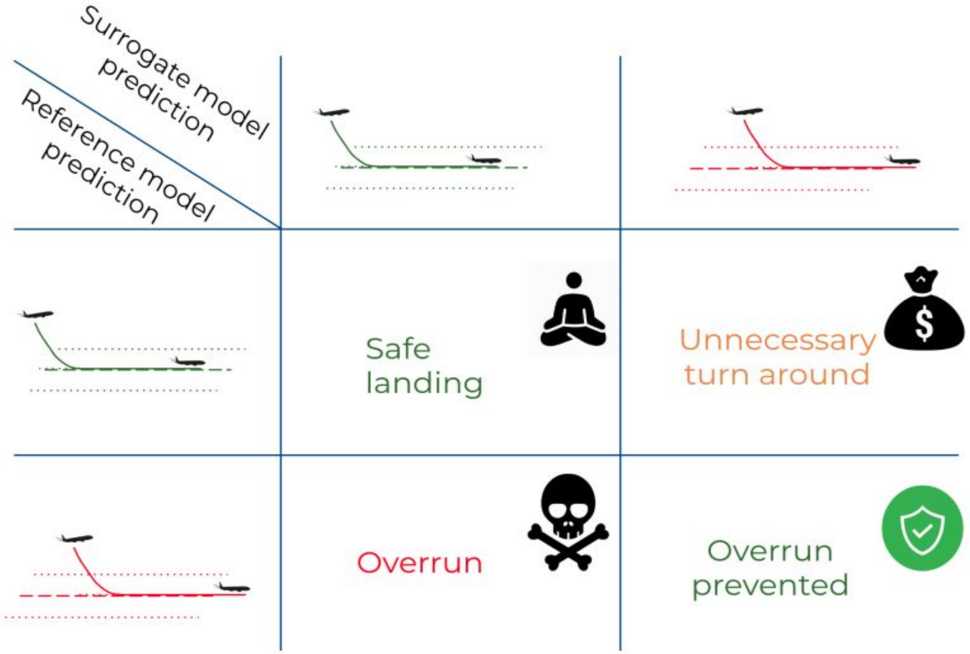
23/09/2022: a Boeing overrun in Montpellier

Braking Distance Estimation

Consequences of under-estimation and over-estimation of a surrogate model are not aligned.

How can we ensure safety in surrogate model learning?

Special case where safety ⇔ over-estimation of the reference model.



Surrogate model prediction

Reference model prediction

Safe landing

Unnecessary turn around

Overrun

Overrun prevented

Embedding Neural Networks for Surrogate Modeling

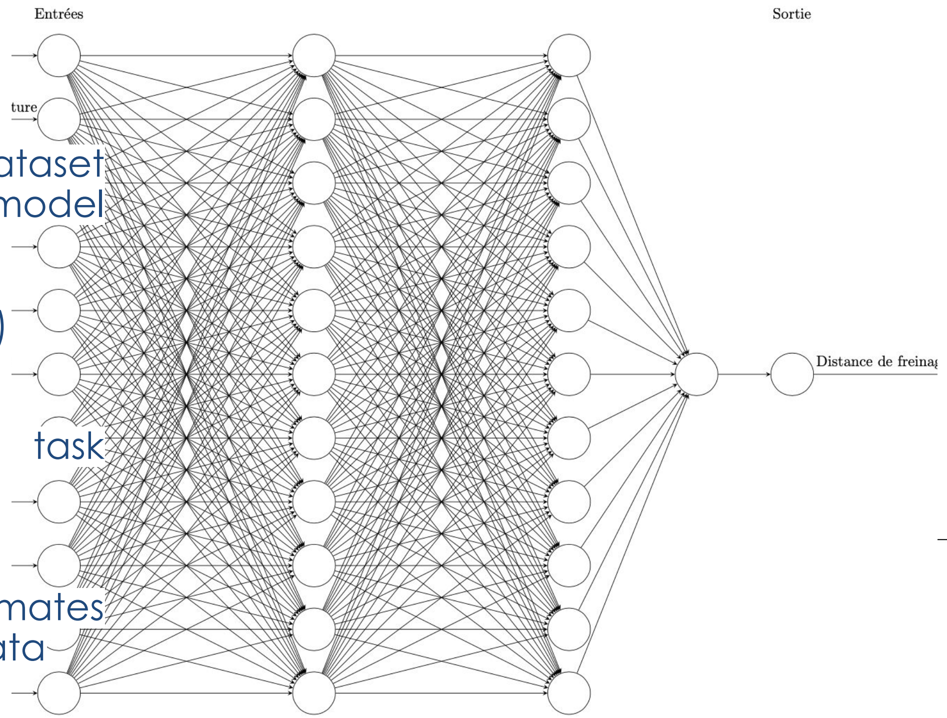
Training shallow neural networks for a regression task

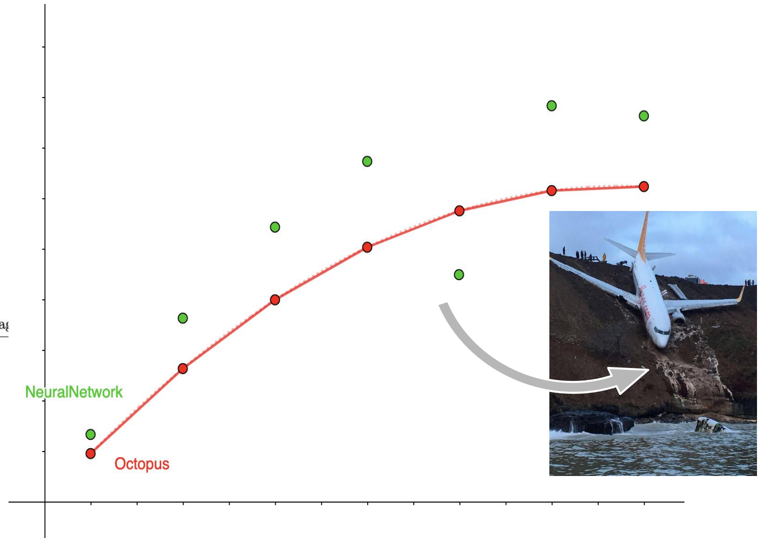
Step 1:
- Sample training/testing dataset
  Using the reference model

Step 2:
- Design a shallow neural network (NN)

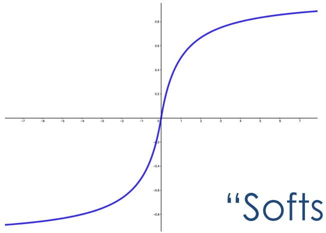
Step 3:
- Train the NN for a regression task
  (symmetric or asymmetric)

Step 4:
- Check the the NN over-estimates
  The reference function on the test data



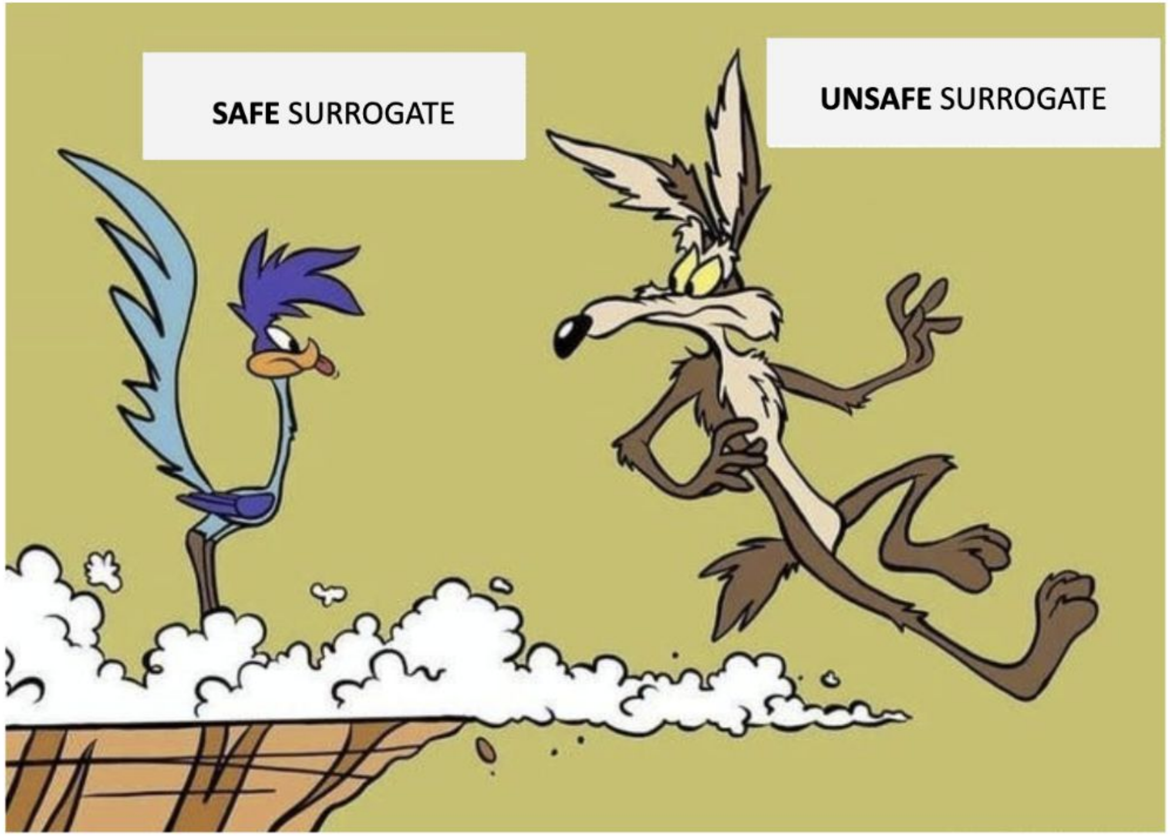


Program testing can be used to show the presence of bugs but never their absence (E. Dijkstra))



“Softsign’ function widely used in SCADE implementation

Reproducibility



- Notebooks on open source data (CESSNA C172)
- Landing Distance Estimation

https://github.com/ducoffeM/safety_braking_distance_estimation

Probabilistic Assessment For Over-estimation

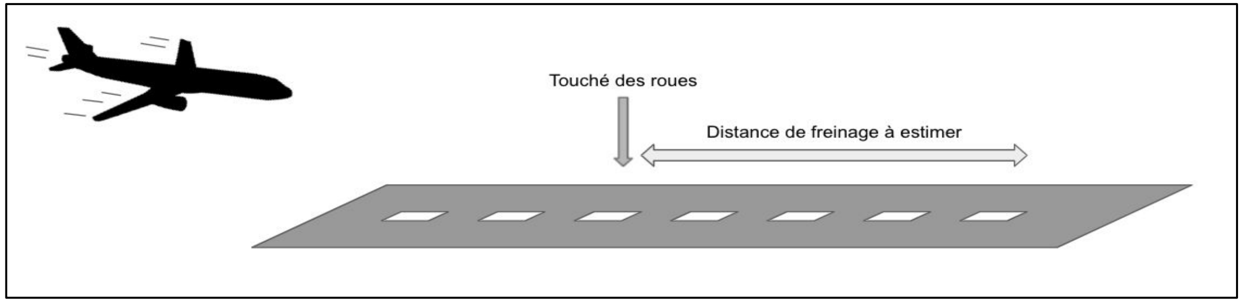
A High-Probability Safety Guarantee for Shifted Neural Network Surrogates

Ducoffe Mélanie, Sébastien Gerchinovitz and Jayant Sen Gupta

Safe AI@AAAI 2020

Ensuring Safety with Probabilistic Assessment

Our Goal: Prove that a surrogate model over-estimates a reference function. The reference function is a black-box system which can be evaluated at any point.



Our Solution: We derive probabilistic inequalities to prove high-probability safety bounds on the surrogate model and on *shifted* versions of it.

A natural probabilistic definition of safety: A surrogate model $\hat{f}$ is (1-**ε**) safe if it over-approximates the reference function f with probability at least 1-**ε**:

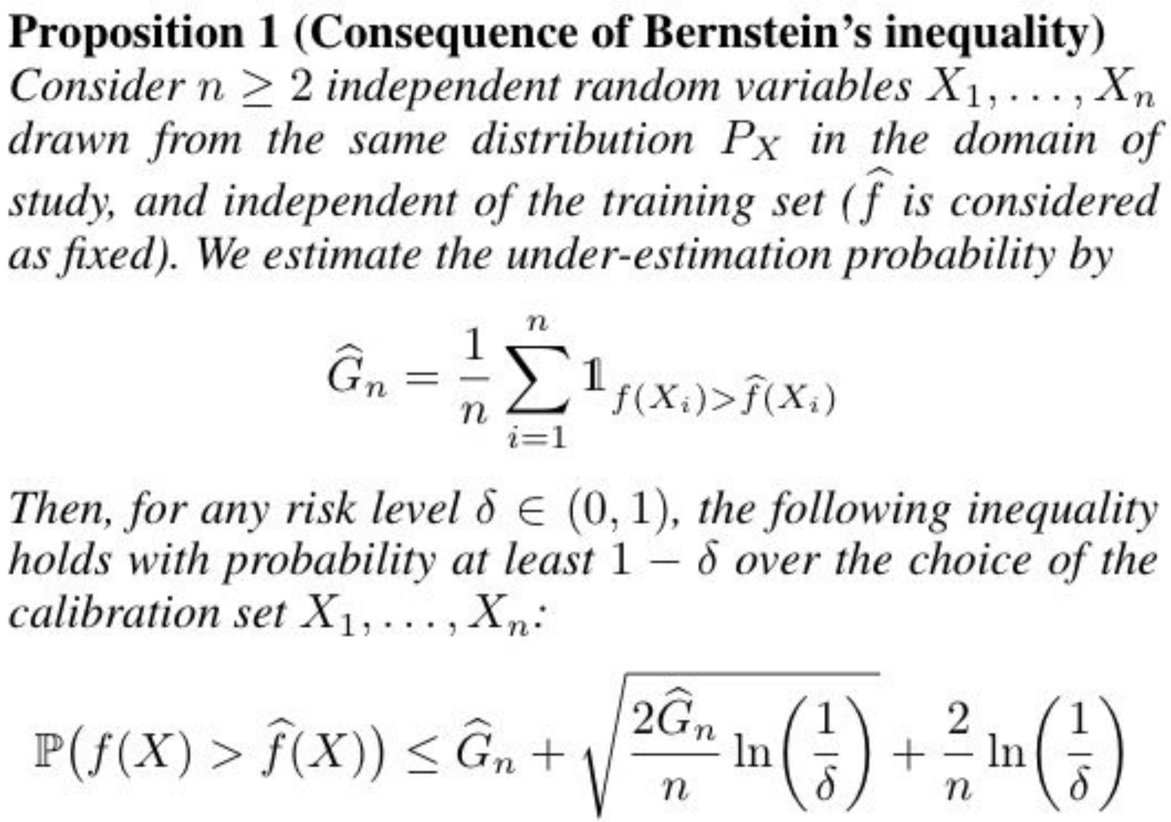
$$\mathbb{P}\left(\widehat{f}(X) \geq f(X)\right) \geq 1 - \varepsilon$$

where X is drawn at random from a given probability distribution P_X on the input domain.

Estimating the probability from samples

We estimate $\mathbb{P}(f(X) > \widehat{f}(X))$ by simply counting how many times we have overestimation among i.i.d. test samples X_1, …, X_n.

We use Bernstein's inequality to relate the unknown probability to its estimate.



Proposition 1 (Consequence of Bernstein's inequality) *Consider $n \geq 2$ independent random variables $X_1, \ldots, X_n$ drawn from the same distribution P_X in the domain of study, and independent of the training set ($\widehat{f}$ is considered as fixed). We estimate the under-estimation probability by*

$$\widehat{G}_n = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{f(X_i) > \widehat{f}(X_i)}$$

Then, for any risk level $\delta \in (0,1)$, the following inequality holds with probability at least $1-\delta$ over the choice of the calibration set $X_1, \ldots, X_n$:

$$\mathbb{P}(f(X) > \widehat{f}(X)) \leq \widehat{G}_n + \sqrt{\frac{2\widehat{G}_n}{n}\ln\left(\frac{1}{\delta}\right)} + \frac{2}{n}\ln\left(\frac{1}{\delta}\right)$$

Interpretation:

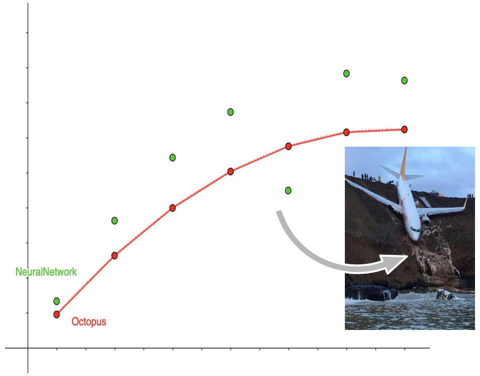
For a large fraction 1-□ of all possible sequences x_1, …, x_n that we could observe, the unknown probability of error is bounded by:

- the observed proportion of errors
- plus a small remainder term

The risk level □ quantifies how unlikely it is to observe a sequence x_1,…, x_n for which the guarantee fails.

Shited Surrogate

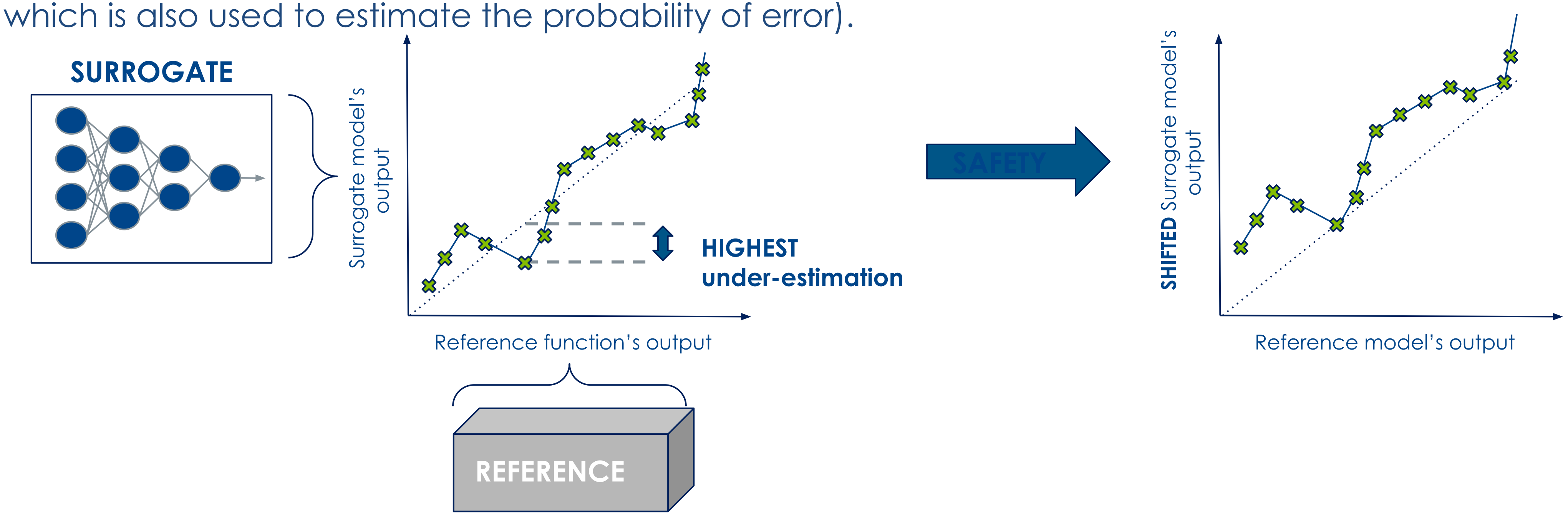
Can we prevent the drop in accuracy with a suitable loss function?



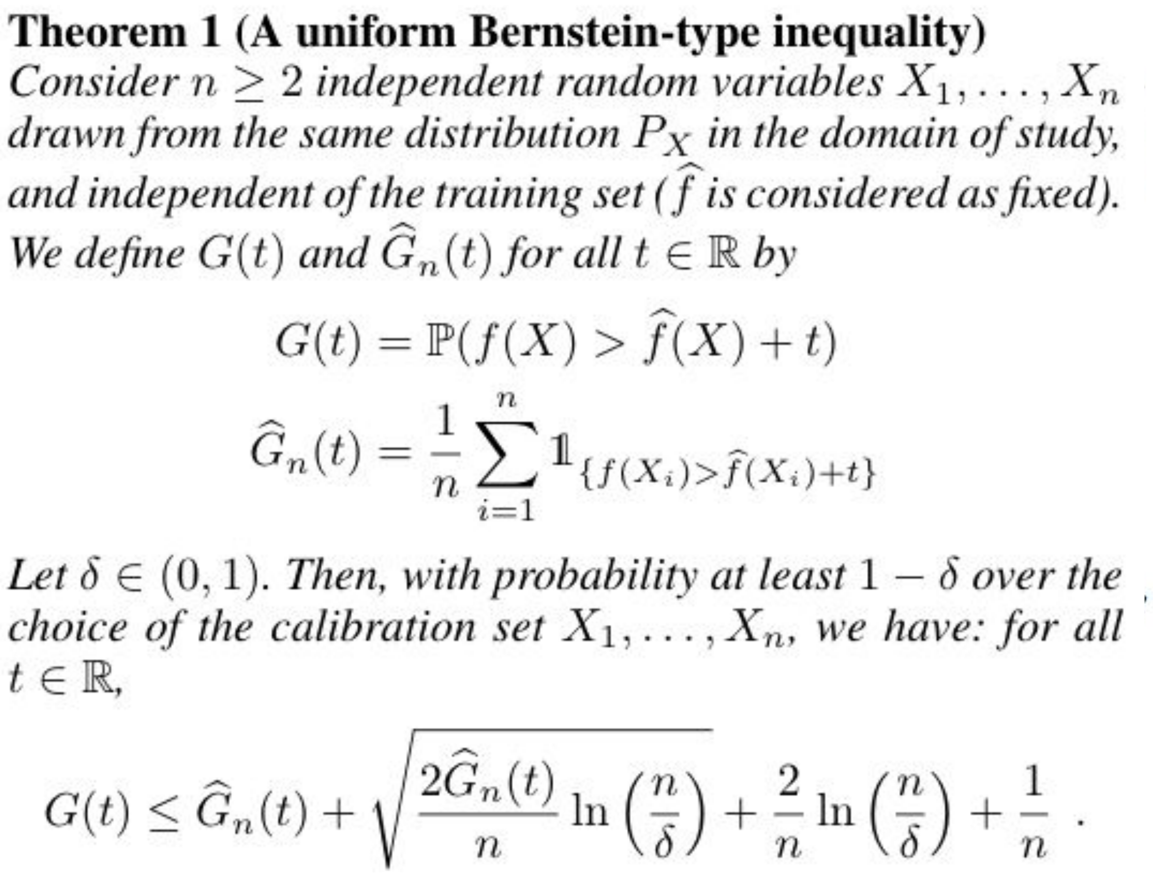
What can we do when the probabilistic guarantee is too loose? (when $\widehat{G}_n$ is large)

We could re-train the model from scratch and pray for a miracle…
Or we can use a basic trick: **shift** all surrogate predictions upwards (by a positive quantity).
Shifting is a simple option and will be efficient **empirically**.

Issue: we lose the guarantees from Prop 1 since it requires to know the value of the shift before-hand. We cannot 'cheat' and optimize our shift as is (the optimized shift depends on the calibration set which is also used to estimate the probability of error).



Probabilistic guarantee for all shifted surrogates simultaneously



Theorem 1 (A uniform Bernstein-type inequality) *Consider $n \geq 2$ independent random variables $X_1, \ldots, X_n$ drawn from the same distribution P_X in the domain of study, and independent of the training set ($\widehat{f}$ is considered as fixed). We define $G(t)$ and $\widehat{G}_n(t)$ for all $t \in \mathbb{R}$ by*

$$G(t) = \mathbb{P}(f(X) > \widehat{f}(X) + t)$$

$$\widehat{G}_n(t) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{f(X_i) > \widehat{f}(X_i) + t\}}$$

Let $\delta \in (0,1)$. Then, with probability at least $1 - \delta$ over the choice of the calibration set $X_1, \ldots, X_n$, we have: for all $t \in \mathbb{R}$,

$$G(t) \leq \widehat{G}_n(t) + \sqrt{\frac{2\widehat{G}_n(t)}{n} \ln\left(\frac{n}{\delta}\right)} + \frac{2}{n} \ln\left(\frac{n}{\delta}\right) + \frac{1}{n} \; .$$

Interpretation:

For most calibration sets (a proportion 1-□ of them), the probability of error of **all shifted surrogates** is bounded by

- their observed proportion of errors
- plus a small remainder term

The guarantee is valid for all shifts t simultaneously.

In particular, it is valid for a shift chosen **after** observing the calibration set.

Safety proof for shifted surrogates

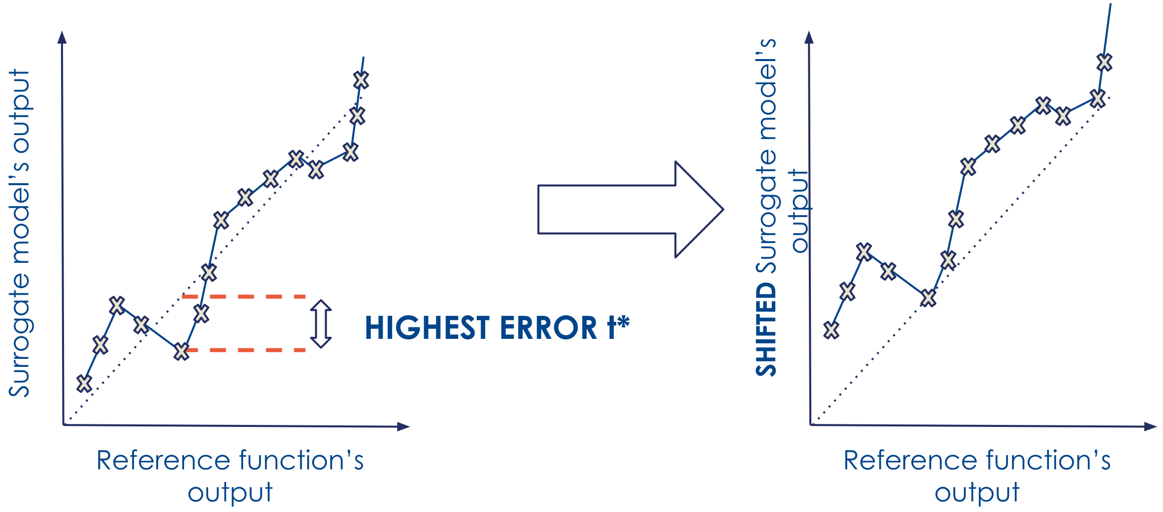
How to post-process (shift) a surrogate to guarantee its safety?

$$G(t) \leq \widehat{G}_n(t) + \sqrt{\frac{2\widehat{G}_n(t)}{n} \ln\left(\frac{n}{\delta}\right)} + \frac{5.67}{n} \ln\left(\frac{n}{\delta}\right)$$

E.g., choose the minimal shift t* for which the observed proportion of errors $\widehat{G}_n(t)$ vanishes.

For this shift t*, the previous theorem shows that the shifted surrogate is (1–ε)–safe with

$$\varepsilon = 5.67 \ln(n/\delta)/n$$


Surrogate model's output
HIGHEST ERROR t*
Reference function's output
SHIFTED Surrogate model's output
Reference function's output

Of course, shifting decreases the model's accuracy.
Can we train a surrogate in order to reduce the impact of shifting on accuracy?

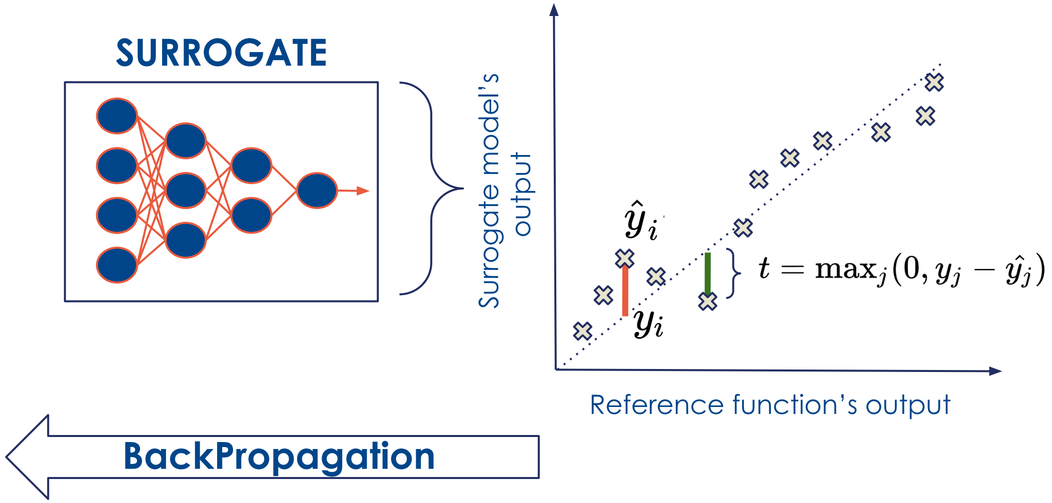
Shifted Training

Can we prevent the drop in accuracy with a suitable loss function?

Neural Networks are known to be ‘good’ surrogate models. But shifting them post training may drastically impact the accuracy.

Instead of training with a standard loss function for regression (Mean Squared Error), we can consider the impact of shifting directly at the training stage.

We propose to add an estimate of the post-processing shift in the loss function.



MSE $\frac{1}{N}\sum_i (y_i - \hat{y}_i)^2$

Asymmetric MSE $\frac{1}{N}\sum_i \left(e^{\alpha(y_i - \hat{y}_i)} - \alpha(\hat{y}_i - y_i) - 1\right)$

Shifted MSE $\frac{1}{N}\sum_i (y_i - \hat{y}_i - t)^2$

Industrial Use Case

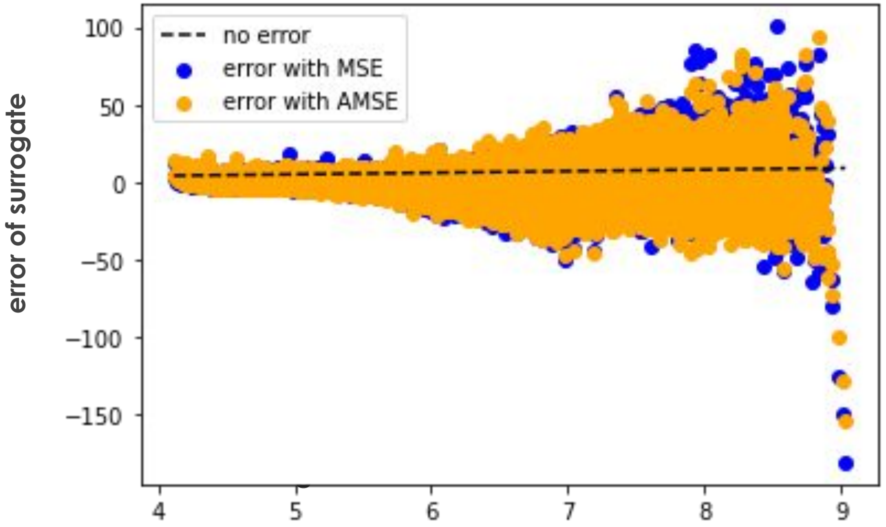
Surrogate for Braking Distance Estimation

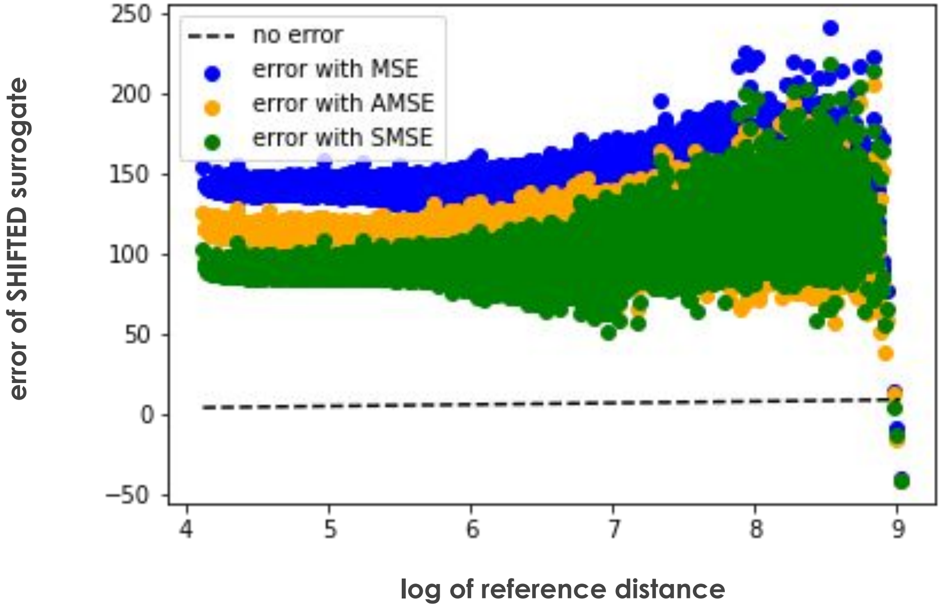
Over-estimate the braking distance with a surrogate neural network

Training samples= 544000, Calibration samples= 181000, Test samples= 181000

Learning with MSE, AMSE and our loss **SMSE**

Error = surrogate - reference





Good accuracy:

- AMSE (MSE): 5.7
- MSE (MSE): 5.3

But with large probability of error. Estimated on the test set:

- AMSE : ε=53%
- MSE : ε=55%

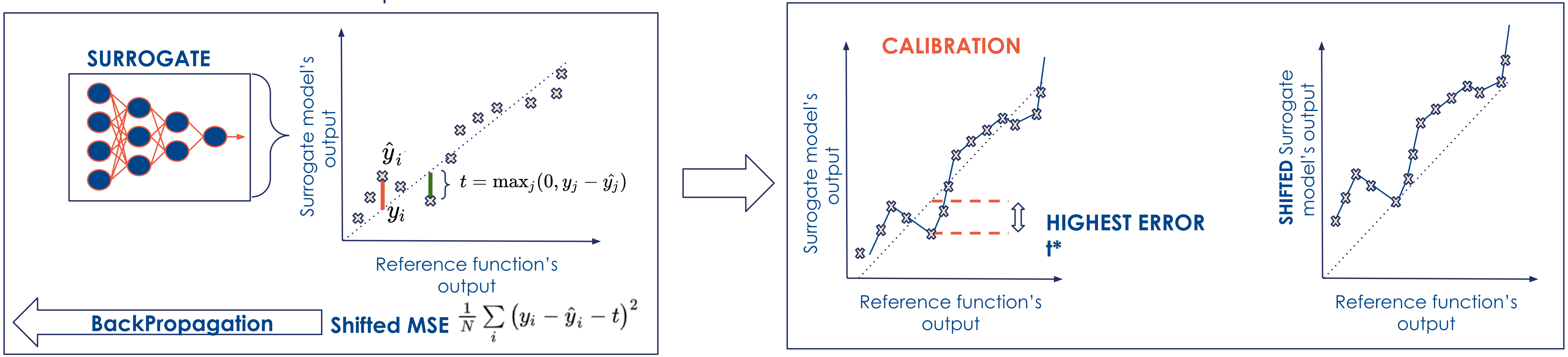
Drop in accuracy:

- SMSE (MSE): 94
- AMSE (MSE): 112
- AMSE (MSE): 142

But with small probability of error:
- estimate on the test set: ε <= 1e-5
- safety guarantee (with □=1e-9) : ε <= 4e-4

In a Nutshell

Our method in two steps:


SURROGATE
Surrogate model's output
$\hat{y}_i$
y_i
$t = \max_j(0, y_j - \hat{y}_j)$
Reference function's output
BackPropagation
Shifted MSE $\frac{1}{N}\sum_i (y_i - \hat{y}_i - t)^2$
CALIBRATION
Surrogate model's output
HIGHEST ERROR
t*
Reference function's output
SHIFTED Surrogate model's output
Reference function's output

Theoretical guarantee: high probability (not worst-case), but comes with certified bound without any assumptions on the surrogate and reference models.

Perspectives:

- beyond surrogate models (ongoing)
- other loss functions, other learning tasks

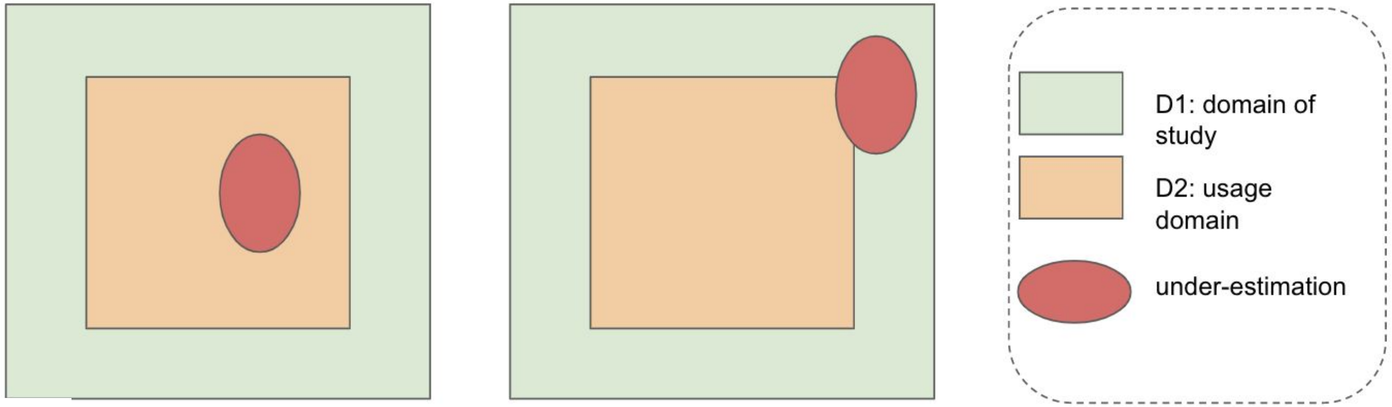
Limitation of (1-ε) safety

Surrogate model $\hat{f}$ is (1-ε)-safe iff $\mathbb{P}\left(\widehat{f}(X) \geq f(X)\right) \geq 1 - \varepsilon$

$\widehat{f}(x)$

The input distribution P_X can be chosen a priori (e.g., uniform) or correspond to real data. All we assume in the sequel is we have access to samples $X_1, \ldots, X_n$ from P_X.

If P_X is uniform on a domain of study D, the choice of D is important for meaningful interpretation.



⚠ Certification authorities require to over-approximate the operational domain

⚠ A single surrogate for multiple operational domains

https://github.com/airbus/decomon

Formal Verification For Over-estimation

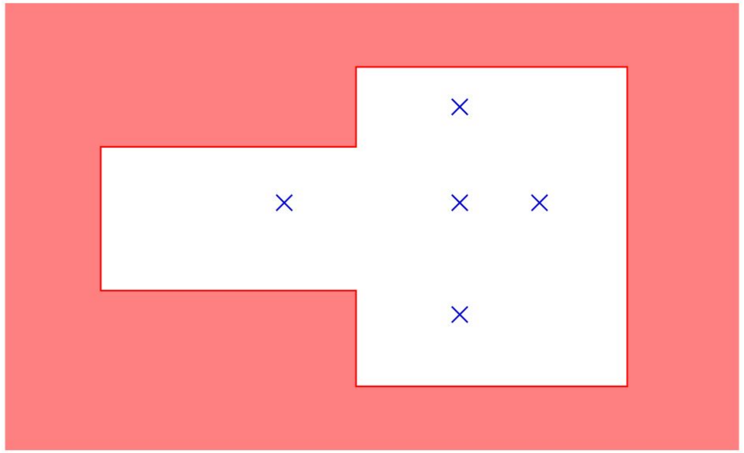
Over-estimation learning with Guarantees
Gauffriau Adrien, Malgouyres François and Ducoffe Mélanie
Safe AI@AAAI 2021

Formal Monotony Analysis of Neural Networks with Mixed Inputs
Vidot Guillaume et al.
International Conference on Formal Methods for Industrial Critical Systems, 2022

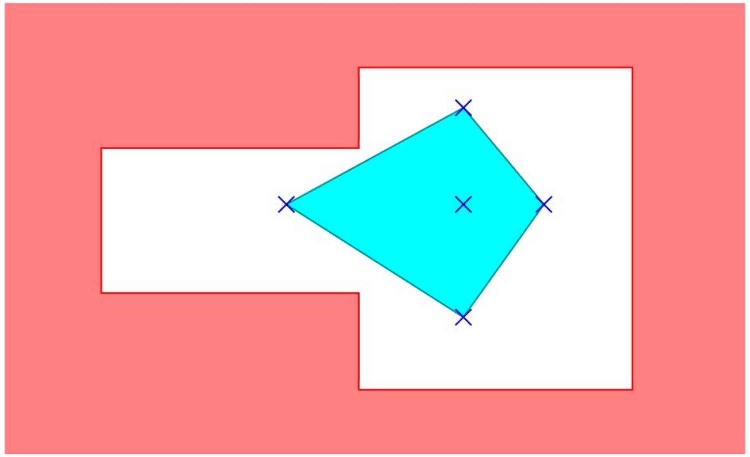
AIRBUS

Airbus Legacy on Formal Verification

Broaden formal verification to Neural Networks



Program is correct (× ∩ □=∅).



Polyhedral abstraction proves correctness (*Cyan* ∩ □=∅).

Airbus A340-300 (2003)

Airbus A380 (2004)

(case study for) ESA ATV (2008)

- size: from 70 000 to 860 000 lines of C
- analysis time: from 45mn to ≃40h
- alarm(s): 0 (proof of absence of run-time error)

Astrée
www.astree.ens.fr

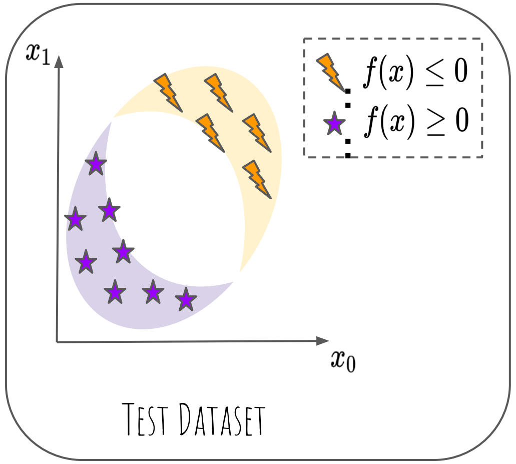
AbsInt
www.absint.com

Applications in software engineering:

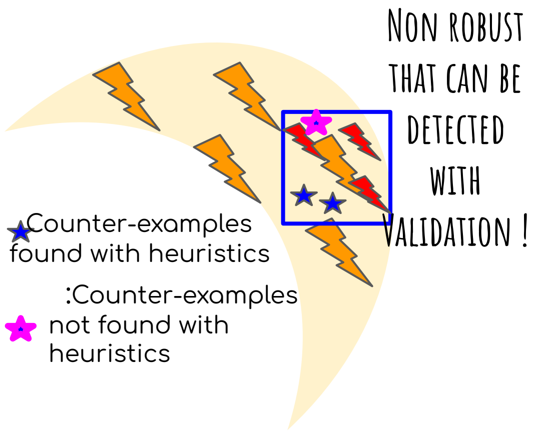
- Analysis of **run-time errors** (arithmetic overflows, array overflows, divisions by 0, …)
- On embedded critical **C** software (no dynamic memory allocation, no recursivity)
- **control/command** software (reactive programs, intensive floating point computations)

Awareness on Formal methods for AI

Verification of Local Robustness



human decisions are locally stable… so should be the decision of a machine learning model


NON ROBUST THAT CAN BE DETECTED WITH VALIDATION !
Counter-examples found with heuristics
:Counter-examples not found with heuristics

Adversarial example: Program testing can be used to show the presence of bugs but never their absence (E. Dijkstra))

Formal Robustness

Given an input domain $\Omega \nexists x \in \Omega$ s.t $f(x) > 0$

COMPLETE

$$\max_{x \in \Omega} f(x) \leq 0$$

INCOMPLETE

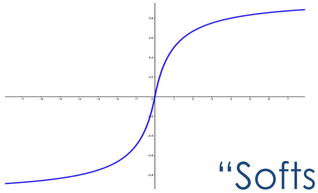
$$\max_{x \in \Omega} \hat{f}(x) \leq 0$$

$$\forall x \in \Omega \; f(x) \leq \hat{f}(x)$$

Few complete verification methods are compatible with s-shape activation
Incomplete verification methods is compatible with any native Deep Learning activation

~~COMPLETE~~

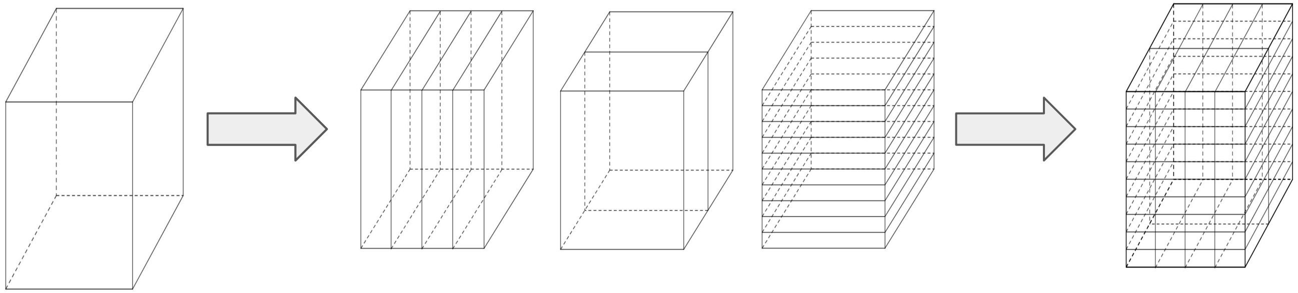
$$\cancel{\max_{x \in \Omega} f(x) \leq 0}$$

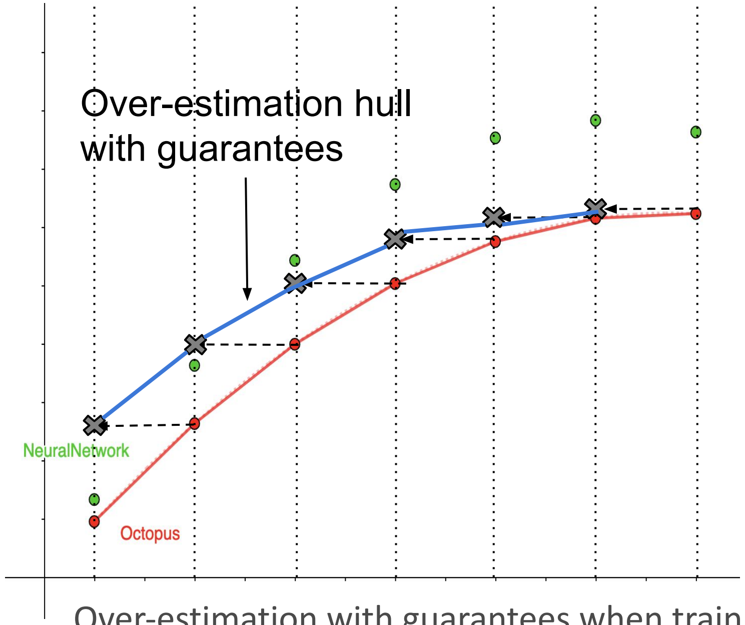


"Softsign' function widely used in SCADE implementation

Over-estimating and Majoring Points

When the properties involve both the Neural Network and the reference function…



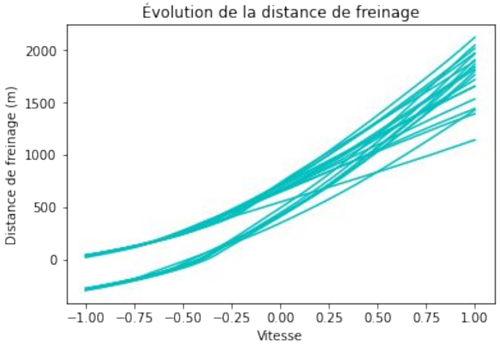


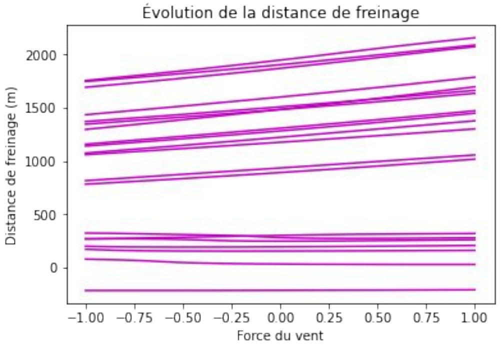
Over-estimation with guarantees when training a Monotonic Neural Network on Majoring Points

The reference function over the input domain needs to be (over) approximated:

- Either with a fine grained approximation (Look Up Table based on expert knowledge, too heavy to be embedded)
- Over-approximated under Monotony assumption

  (speed ↗ => braking distance)



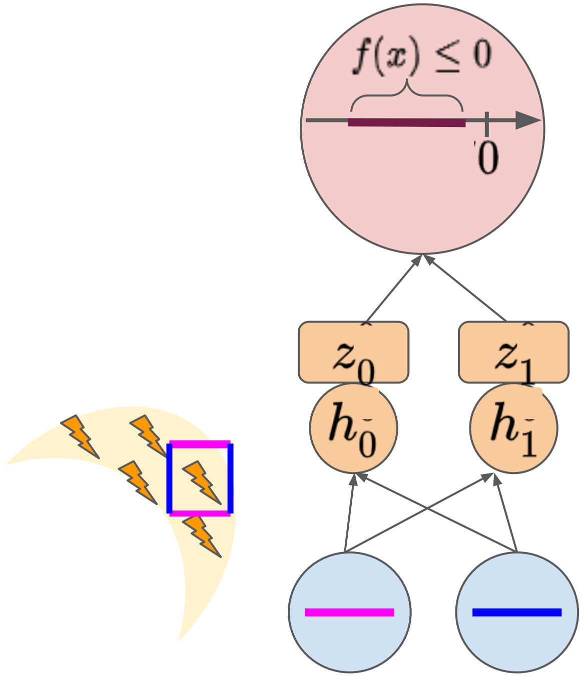


Method	*# train*	*RMSE*	*Guarantee*
0-baseline	150k	3.3	✗
300-baseline	150k	302.6	✗
ONN with MP	110k	445.7	✓

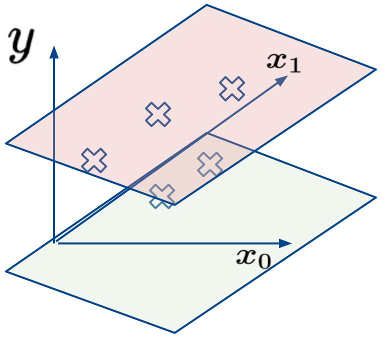
Verification with Majoring Points on an industrial dataset

X-baseline: we train a neural network with MSE on the training distance with the X additive constraints.

In a Nutshell : Linear Relaxation for Neural Networks

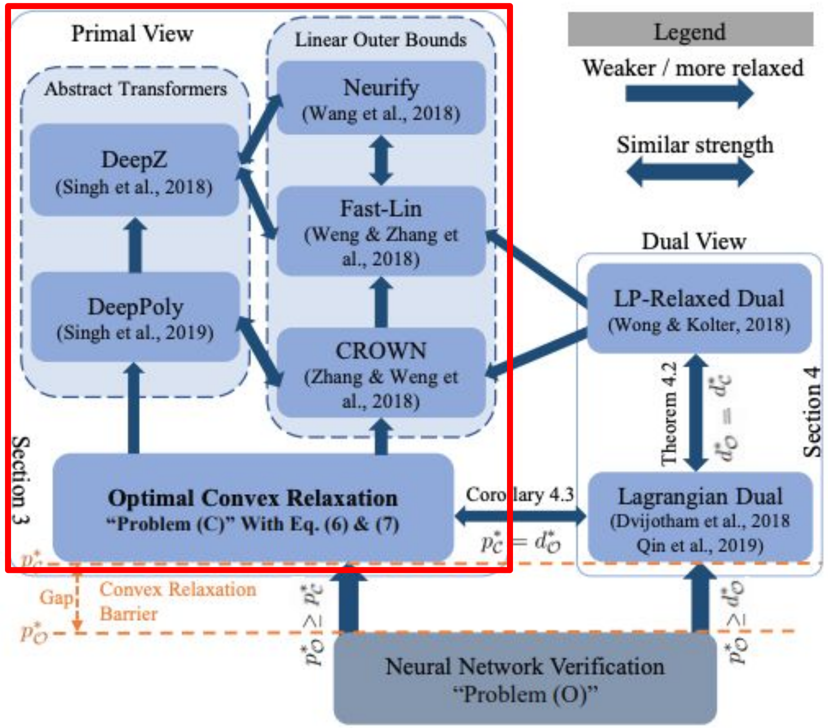


CASE 1: ROBUST

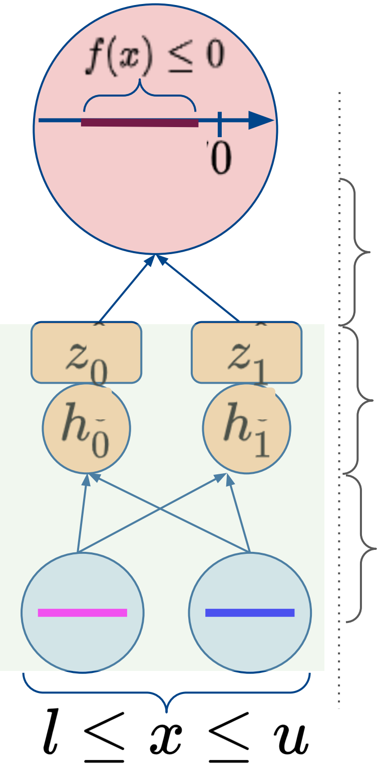


draw two hyperplanes that are possible solutions
ill-posed

Different ‘recipes’ in the litterature that balance efficiency and scalability: primal approaches that propagate linear relaxations through the network



Interval Bound Propagation



$$y = W_1 \cdot z + b_1$$

$$z_j = \max(0, h_j^i)$$

$$h = W_0 \cdot x + b_0$$

Step 1: optimize a linear activation function given constant input bounds

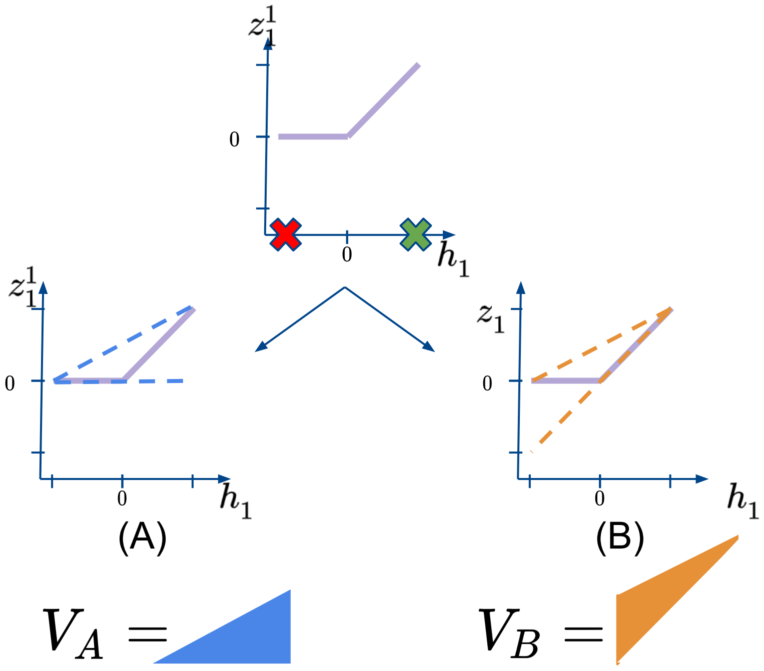
$$u_h : \max_{l \leq x \leq u} W_0 \cdot x + b_0 = W_0^{\geq 0} \cdot u + W_0^{\leq 0} \cdot l + b_0$$

$$l_h : \min_{l \leq x \leq u} W_0 \cdot x + b_0 = W_0^{\geq 0} \cdot l + W_0^{\leq 0} \cdot u + b_0$$

Step 2: optimize increasing activation function

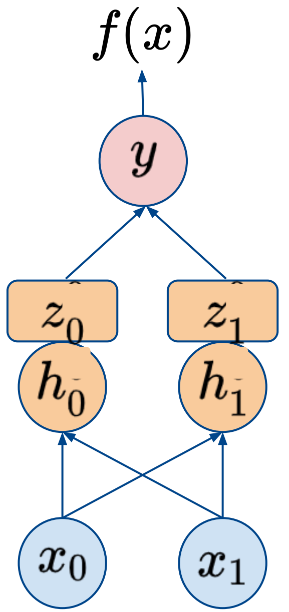
$$u_z : \max_{l_h \leq h \leq u_h} max(0, h) = max(0, u_h^0)$$

$$l_z : \min_{l_h \leq h \leq u_h} max(0, h) = max(0, l_h^0)$$



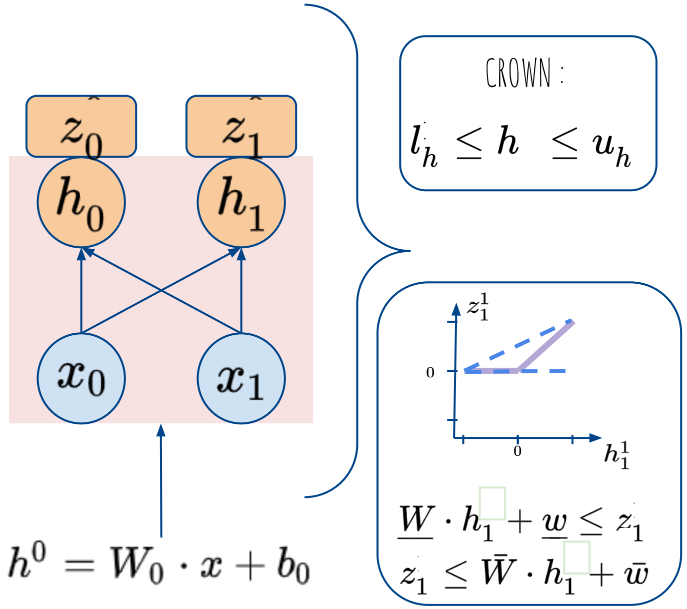
$V_A \leq V_B$: **Pick A**

$V_B \leq V_A$: **Pick B**



$f(x) \leq W_1 \cdot z + b_1$

$W_1 \cdot z + b_1 \leq f(x)$



$\underline{W} \cdot h_1 + \underline{w} \leq z_1$

$z_1 \leq \bar{W} \cdot h_1 + \bar{w}$

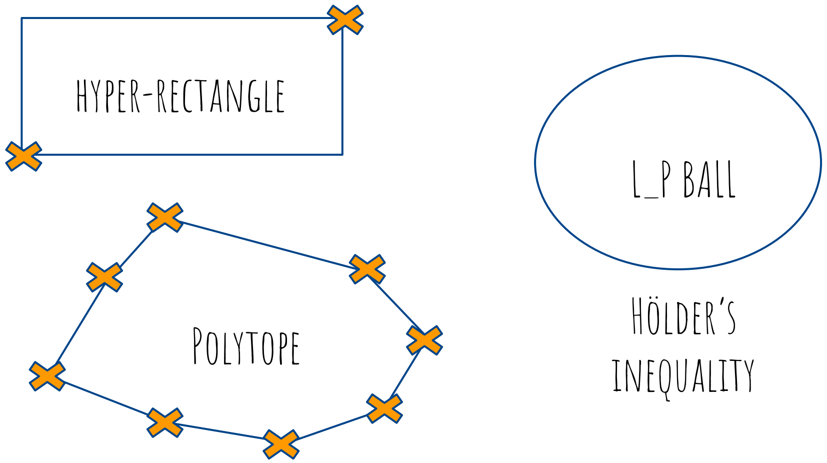
$h^0 = W_0 \cdot x + b_0$

Computing linear relaxations over non-linear operations (ReLU) require to bound the input domain with:

* a lower bound ✖

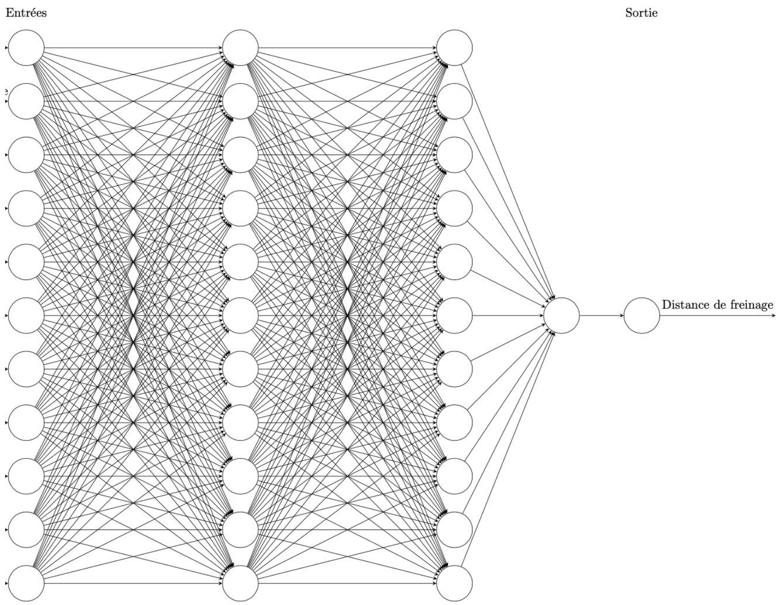
* an upper bound ✖

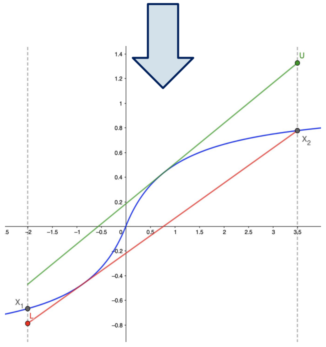
With affine functions, bounds can be computed symbolically for specific domains:



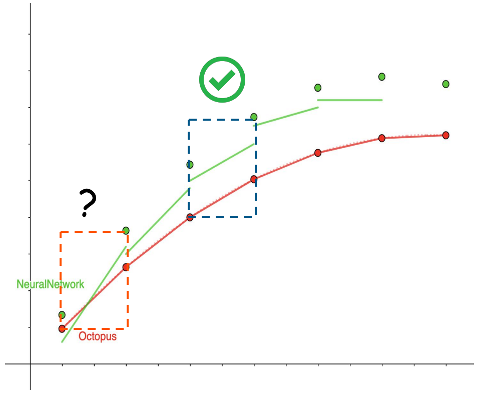
Formal Verification of Over-estimation

Pipeline of Linear Relaxation

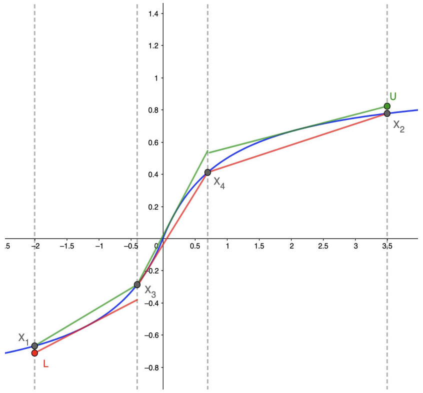


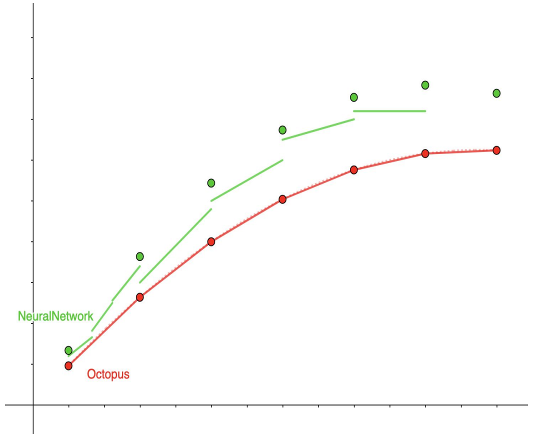


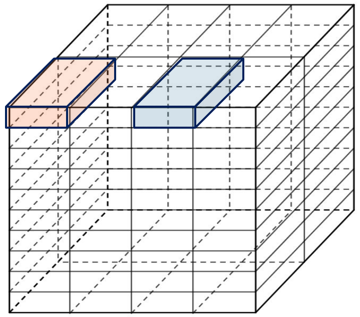
Local Linear relaxation



Branch and Bound
Input partitioning
(Gradient)

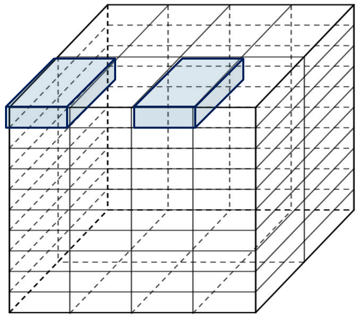






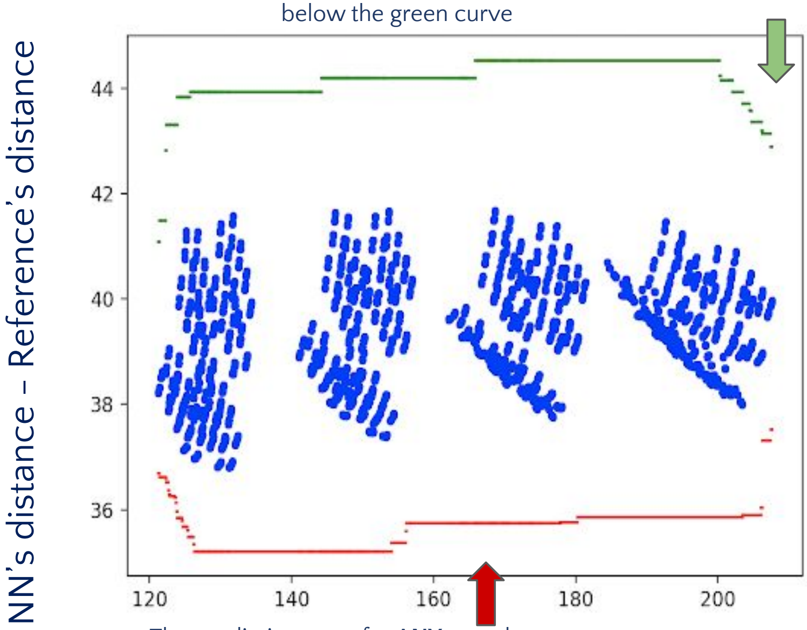
Too loose ?

- ❑ The partition is only for the neural network’s linear relaxation
- ❑ **No extra sampling on the reference model is required**



Incomplete Verification For the Safety of Braking Distance Estimation

The prediction error for **ANY** sample from is below the green curve



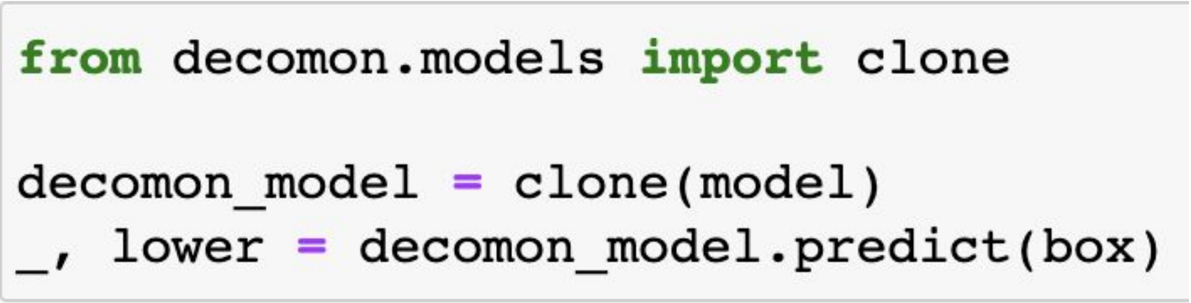
The prediction error for **ANY** sample from is over the red curve

● Prediction error between the neural network and the reference on test samples (the corners)

🟩 Formal upper bound on the prediction error between the neural network and the reference

🟥 Formal lower bound on the prediction error between the neural network and the reference

Highly conservative Neural Network
Formal method can help tightening the over-estimation
No constraints on NN's monotonicity



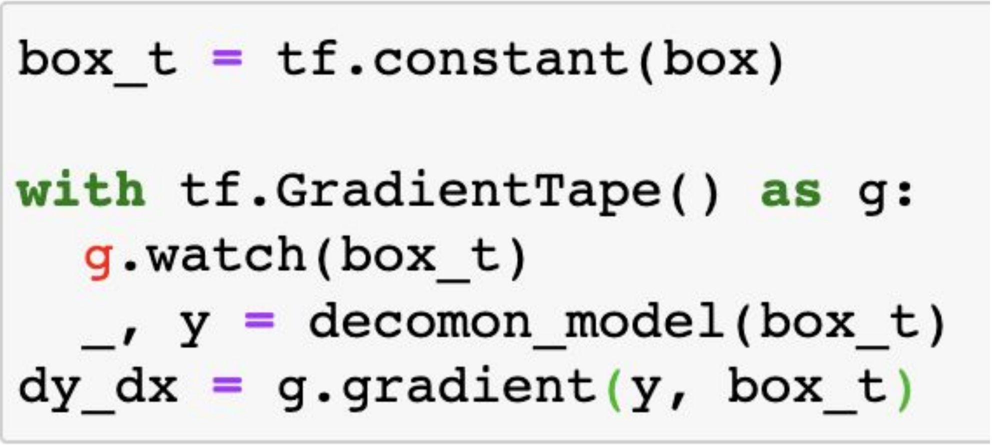
```
from decomon.models import clone

decomon_model = clone(model)
_, lower = decomon_model.predict(box)
```

Plug and Play library to compute Linear Relaxation.
Airbus open source library
With the support of ANITI

https://github.com/airbus/decomon

Input partitioning



```
box_t = tf.constant(box)

with tf.GradientTape() as g:
  g.watch(box_t)
  _, y = decomon_model(box_t)
dy_dx = g.gradient(y, box_t)
```

Partial Input Monotonicity for Safety

Over-estimation will not be the solely pre-requisite for safety
Property 2: If monotony is not enforced in the design, it may be safety critical given some inputs
Only on the Neural Network

$$\forall (x_1, x_2) \in X^2 : x_1{\downarrow_{\bar\alpha}} = x_2{\downarrow_{\bar\alpha}} \wedge x_1{\downarrow_{\alpha}} \preceq x_2{\downarrow_{\alpha}} \implies f(x_1) \preceq f(x_2)$$

$$\begin{array}{cccccc} x_1 & & x_2 & & f(x_1) & f(x_2) \\ \begin{pmatrix}\text{speed}\\ \text{weight}\\ \text{dry runway}\end{pmatrix} & \begin{matrix} = \\ = \\ < \end{matrix} & \begin{pmatrix}\text{speed}\\ \text{weight}\\ \text{wet runway}\end{pmatrix} & \implies & \text{BDE}_1 < & \text{BDE}_2 \end{array}$$

Previous works consider PIM on continuous inputs (gradient)
No existing work on discrete inputs

2 layers neural network

60 neurons in total

1409 parameters

ReLU activation function

15 Inputs

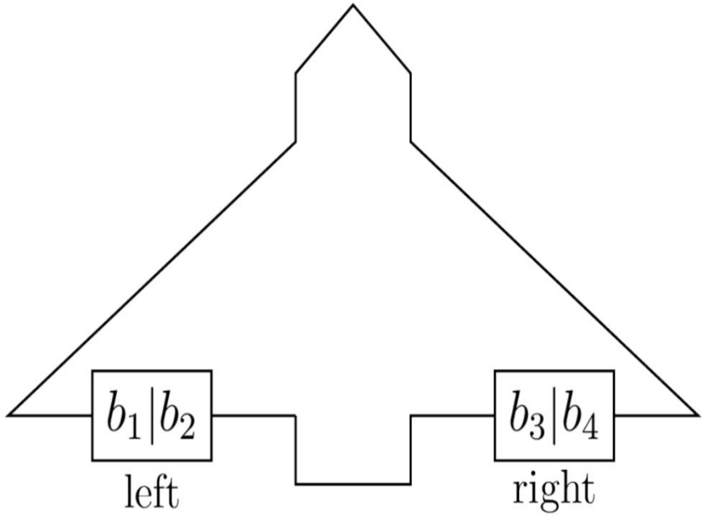
13 discrete features

2 continuous features

When the brakes' state deteriorates, the braking distance should increase.

Brakes' states: Normal, Altered, Emergency, Burst, Release

Order on Brakes's states: $N \prec_b A \prec_b E \prec_b B \prec_b R$



sym / asym N,A,E,B,R / N,A,E,B,R $\quad b_1|b_2$ / $b_3|b_4$

(4,0,0,0,0/0,0,0,0,0) ≡ NN / NN

↓

(3,1,0,0,0/-1,1,0,0,0) ≡ NA / NN

sym = left + right

asym = left - right

Exact Verification

Only for piecewise linear activation (ReLU…)

MILP Generic Problem Definition

$\boldsymbol{min\ c_1 x_1 + c_2 x_2 + \cdots + c_n x_n}$	objective
$\boldsymbol{a_{11} x_1 + \cdots + a_{1n} x_n \le b_1}$ .... $\boldsymbol{a_{m1} x_1 + \cdots + a_{mn} x_n \le b_m}$	constraints
$\boldsymbol{l_i \le x_i \le u_i \quad 1 \le i \le n}$	bounds on continuous $\boldsymbol{x_i}$
$\boldsymbol{x_j \in Z}$	some $\boldsymbol{x_j}$ are integer

- $\boldsymbol{c_i, a_{ij}, b_i \in R}$ and $\boldsymbol{l, u \in R^n}$
- some $\boldsymbol{x_j}$ ‘s can be integers (or even binary), hence Mixed-Integer problem
- state-of-the-art solvers (e.g., Gurobi) require bounds on $\boldsymbol{x_i}$ ’s

Several MILP solvers: Gurobi, Venus, MIPVerify

ReLU definition is:

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

MILP ReLU encoding is:

$$\boldsymbol{y \le x - l * (1 - a)}$$

$$\boldsymbol{y \ge x}$$

$$\boldsymbol{y \le u * a}$$

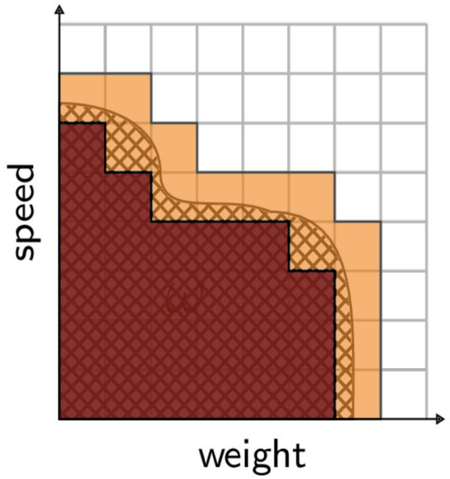
$$\boldsymbol{y \ge 0}$$

$\boldsymbol{a \in \{0, 1\}}$ → $\boldsymbol{a}$ is a binary integer variable

This assumes we have computed lower $\boldsymbol{l}$ and upper $\boldsymbol{u}$ bounds for the input neuron $\boldsymbol{x}$ (e.g., by using Box beforehand).

Application to Braking Distance Estimation

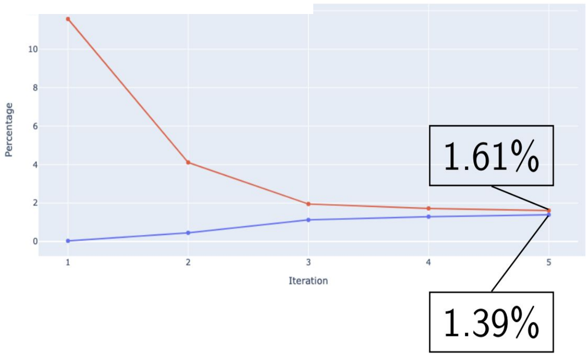
Identify the sub-spaces where the **monotony does not hold** using a Mixed Integer Linear Programming (MILP) solver

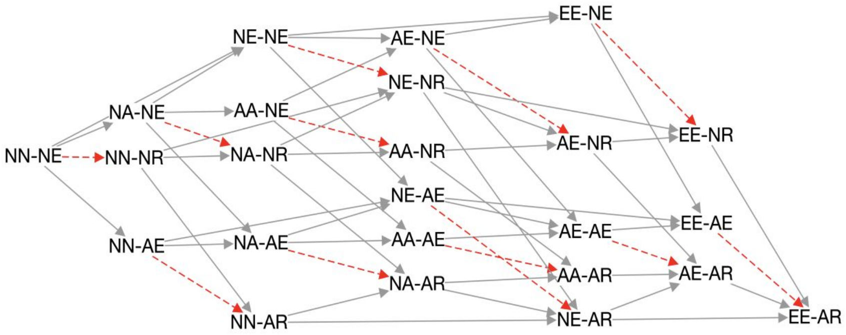


▩ exact sub-space (ω) where the monotony property does not hold

■ lower bound $\underline{\Omega}$ of ω

■ + ■ upper bound $\overline{\Omega}$ of ω

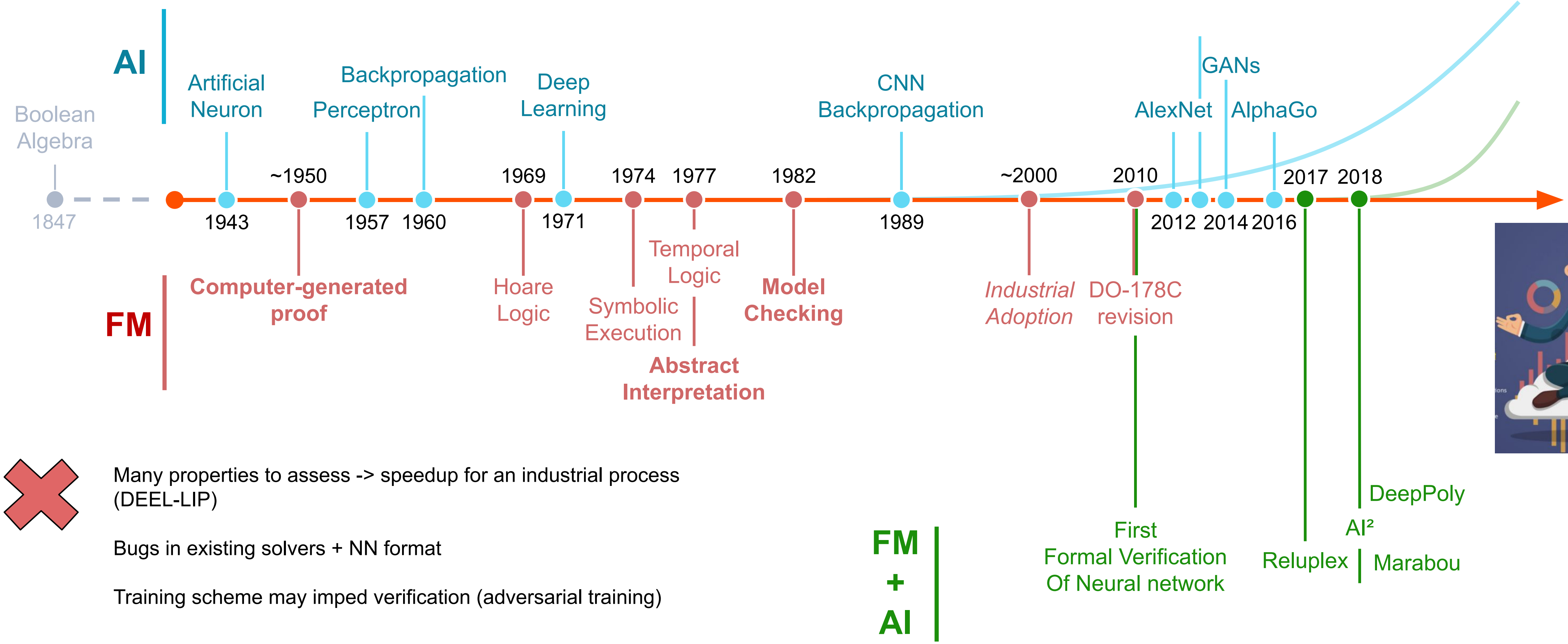




Gurobi 9
<10 hours (MacBook Pro 8 core 2.3 GHz Intel Core i9 with 32 Gb)

Food for thought: Verification in an Industrial pipeline

Challenge 1: Recent field of research



Many properties to assess -> speedup for an industrial process (DEEL-LIP)

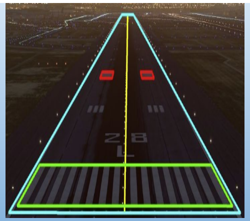
Bugs in existing solvers + NN format

Training scheme may imped verification (adversarial training)

verification mainly intended for data scientists: high level of code, no assurance that the property is kept when embedded

Food for thought: Verification in an Industrial pipeline

Challenge 3: Scalability

Verification For	Shallow NN Hybrid AI	Computer Vision	Time Series	Decision Making	NLP
Business	Surrogate Modeling	 VBL, HMF	advisedlib Predictive Maintenance	 Manufacturing Air Mobility	 ATC
Scalability # NN parameters	~50 000	[~2M ,~6M]		[~700K, ~70M]	> 86M

Too loose ?

- ❑ Input dimension
- ❑ Network depth
- ❑ Certification authorities may require a unique process independently from the depth (jurisprudence)

Reconciliation with Statistics

PROVEN: Certifying Robustness of Neural Networks with a Probabilistic Approach
-> deriving a worst case bound for probability of local robustness risk independently

Statistical Certification of Acceptable Robustness for Neural Networks
-> Hoeffding inequality for neural networks

Thank you

PROVEN: Certifying Robustness of Neural Networks with a Probabilistic Approach
-> deriving a worst case bound for probability of local robustness risk independently

Food for thought: Verification in an Industrial pipeline

Challenge 2: Tools to be matured

✓

COMPLETE and INCOMPLETE verification for tiny/medium neural networks
Full demonstration on industrial usecase

Growing community (synergies with DEEL-LIP)

DEEL LIP
LIPSCHITZ KERAS LAYERS

https://github.com/deel-ai/deel-lip

✗

Many properties to assess -> speedup for an industrial process (DEEL-LIP)

Bugs in existing solvers + NN format

Training scheme may imped verification (adversarial training)

verification mainly intended for data scientists: high level of code, no assurance that the property is kept when embedded

NETWORK	NB Safe (INCOMPLETE)	Time (INCOMPLETE)	Time (COMPLETE)
reluplex	74.01%	15 min	198h
corners	84.13%	8 min	9h
adversarial	69.83%	7 min	4h20

ACAS-XU local robustness verification in 3D (304 000 boxes)

DEEL
DEpendable & Explainable Learning

AIR
Airbus AI Research

ANITI
ARTIFICIAL & NATURAL INTELLIGENCE
TOULOUSE INSTITUTE

speaker

Mélanie Ducoffe --- speaker

Industrial Research Data Scientist - Airbus CRT / ONERA / ANITI

AIR Airbus AI Research

ONERA THE FRENCH AEROSPACE LAB

ANITI ARTIFICIAL & NATURAL INTELLIGENCE TOULOUSE INSTITUTE

Mélanie Ducoffe est chercheuse industrielle au centre de recherche et de technologie d'Airbus depuis 2019 et détachée à mi-temps dans le projet DEEL pour l'étude de la robustesse en machine learning et ses applications aux systèmes critiques. Avant de rejoindre Toulouse, elle a validé ses études de master par un stage sur l'apprentissage génératif avec Yoshua Bengio, puis effectué un doctorat en machine learning au CNRS de Nice Sophia Antipolis sur l'apprentissage actif des réseaux de neurones profonds. Ses principales activités de recherche actuelles sont sur la robustesse des réseaux de neurones, notamment par les méthodes formelles.

The results of multiple collaborations

C. Pagetti

I. Ober

I. Ober

J. Sen Gupta

E.Sudre

A. Gauffriau

G. Vidot

C. Gabreau

F. Malgouyres

S. Gerchinovitz