# Computational techniques for boosting verification of deep learning algorithms

Julien Girard-Satabin (CEA LIST), Guillaume Charpiat (INRIA TAU), Zakaria Chihani (CEA LIST), Marc Schoenauer (INRIA TAU)

24 septembre 2019

# Outline

# Reminder on deep learning

# Deep neural networks : what they are

A neural network is a directed, acyclic, weighted, graph (within our verifications problem)

Weights are learned through a learning procedure which we will not detail much. Key point : constrained optimization problem to minimize a cost function (that's where the "deep" comes from)

# Deep neural networks : why do we use them

Theoretical justifications : *For an activation function* $\phi$ *that is non-constant, continuous and bounded, a neural network* $f(x) = \phi(w^T x + b)$ *can approximate any continuous function on compacts of* $\mathbb{R}^n$ (Cybeko, 1989, universal approximation theorem, and follow up work for width-bounded DNN Lu et al. 2017)

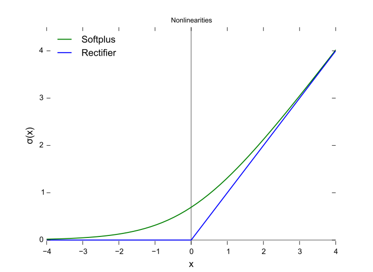
In practice, achieve good results on non-structured data, lot of tools to replicate and deploy them, hype since the convergence between GPUs and vast availability of data

# Deep neural networks : some things to keep in mind

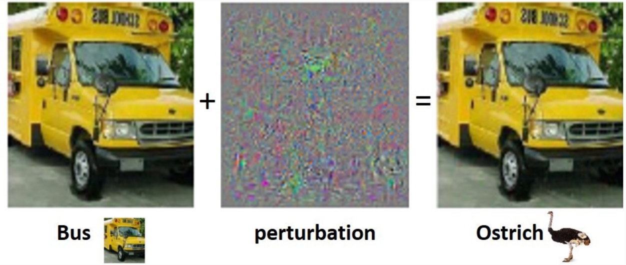
- conceptually simple programs : no loops, no explicit conditionals, just a bunch of additions and multiplications
- modern architectures have about billions of weights
- activations functions are important

# Deep neural networks : activations functions

- gives the DNN its expressivity (non linear functions such as XOR)
- usually occur after some linear operations
- some popular ones : Sigmoid, Rectified Linear Unit : $ReLU(x) = max(x, 0)$



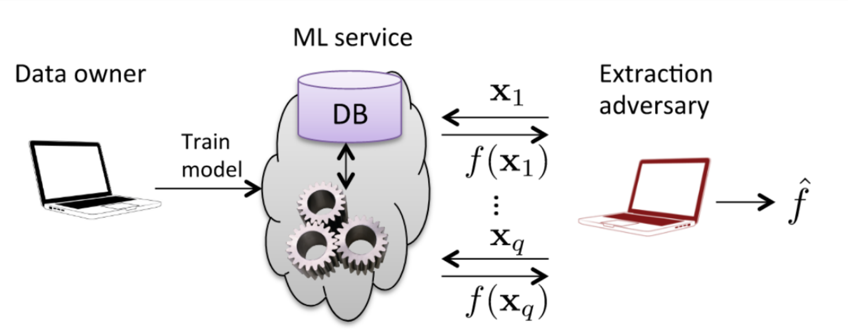
# Necessity to certify deep neural networks and challenges



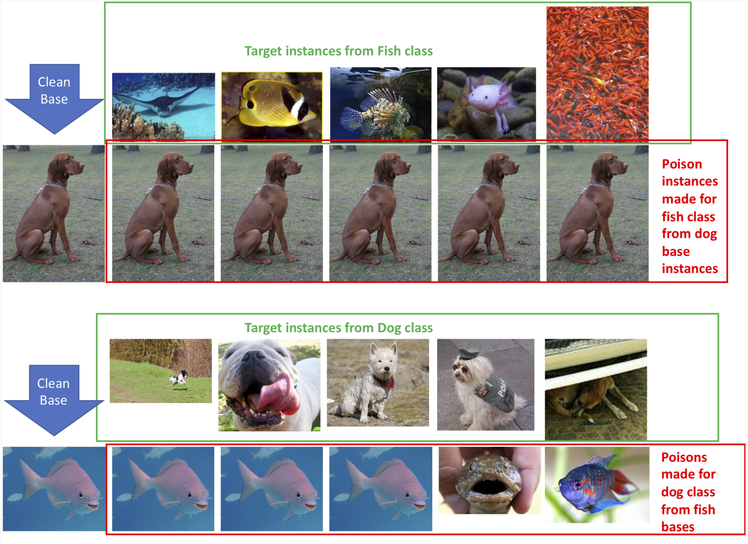


Innocuous to humans, transferable between datasets, not systematic detection method

# Model theft (Tramèr et al. 2018)

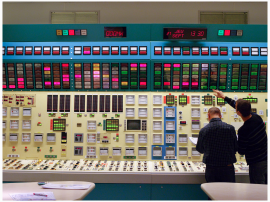


# Dataset poisoning (Shafahi et al. 2018)



# Context : critical systems

*A critical system is a system whose failure may cause physical harm, economical losses or damage the environment*




# How to bring confidence in software systems ?

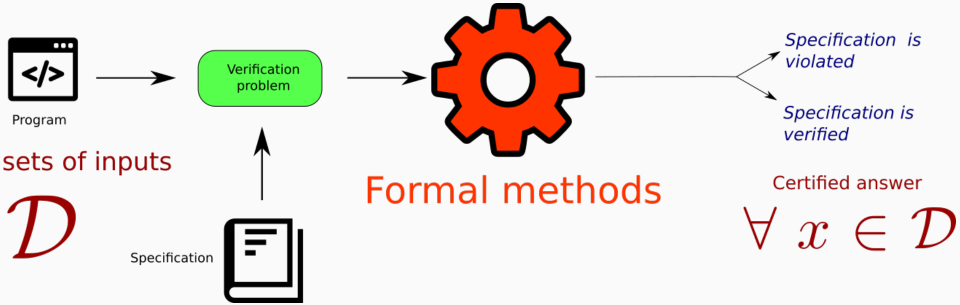
**Goal** : guarantee that the system respects a *safety specification*

$\mathcal{P}$ : *an autonomous car will not run over pedestrians*

# Formal methods history

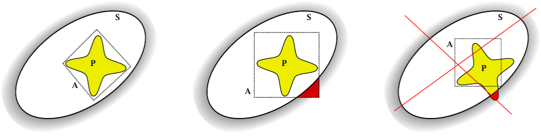
- Studied in the academics since 1930 ($\lambda-$calculus, Church, Turing)
- Different techniques : abstract interpretation (Cousot and Cousot 1977), SAT/SMT (Davis and Putman 1960 ; Tinelli 2009), deductive verification (Coquand 1989), etc.
- Used in industrial settings such as aerospace, automated transports, energy to *formally* certify

# Key points




Work on domains $\mathcal{D}$ of inputs (global properties)

Answer is sound, formally guaranteed by mathematical logic



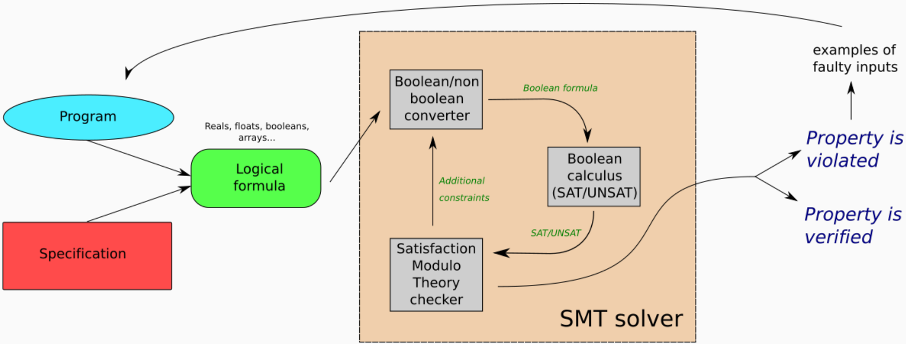

precise analysis $A \subseteq S \implies P \subseteq S$ (a) | false alarm $A \not\subseteq S$ but $P \subseteq S$ (b) | unsound analysis $A \subseteq S$ but $P \not\subseteq S$ (c)

**Figure 1.6:** Proving that a program $P$ satisfies a safety specification $S$, i.e., that $P \subseteq S$, using an abstraction $A$ of $P$: (a) succeeds, (b) fails with a false alarm, and (c) is not a possible configuration for a sound analysis.

Abstract intepretation [1], symbolic execution

1. Cousot et Cousot, 1977, courtesy to Antoine Minet for the figure

# SAT/SMT calculus

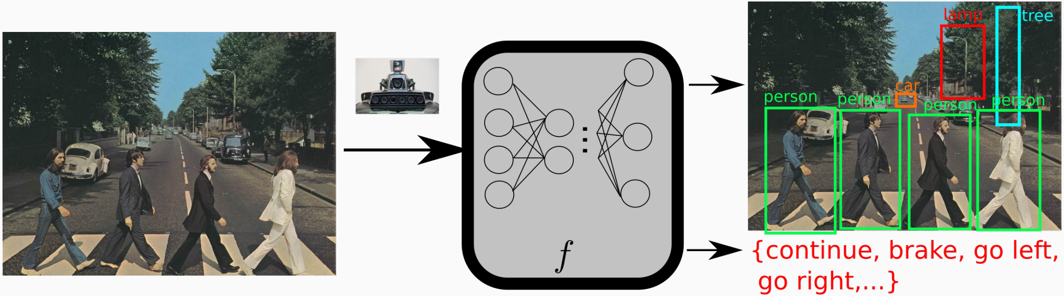



Explicit enumeration of variables instanciations with various search strategies and algorithms (backtracking, clause-driven learning,. . .) → exhaustive and sound but costly

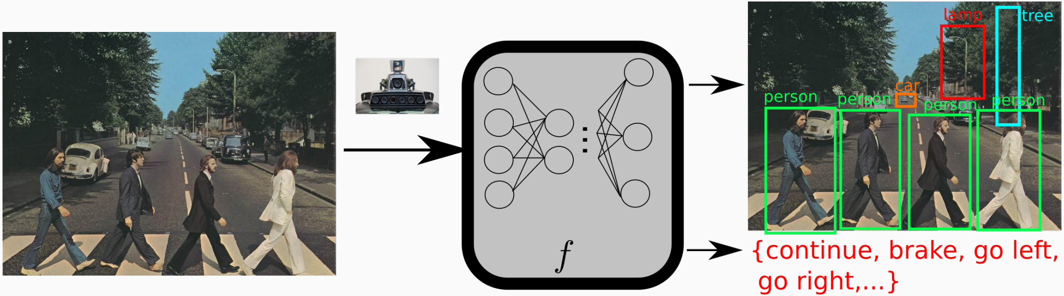
# Question

What prevents us to use formal methods directly on learned programs ?

# Case study : a self-driving car perception unit

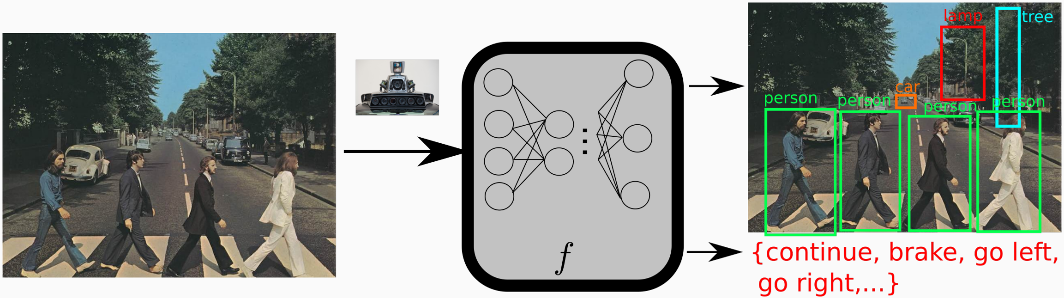


# Case study : a self-driving car perception unit



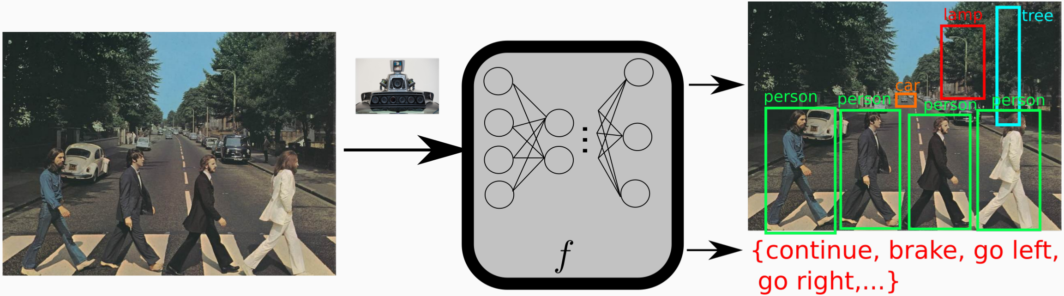

Dream property $\mathcal{P}$ : *the autonomous car never run over pedestrians*

# Case study : a self-driving car perception unit




Dream property $\mathcal{P}$ : *the autonomous car never run over pedestrians*

no *formal characterization* of what a pedestrian is !




Dream property $\mathcal{P}$ : *the autonomous car never run over pedestrians*

no *formal characterization* of what a pedestrian is !

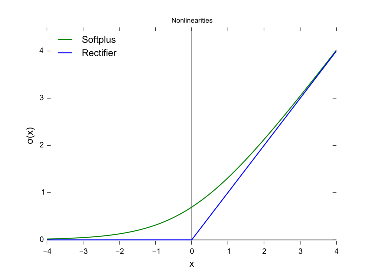
*Lack of formal definition on inputs prevents from formulating interesting safety properties*

# It’s hard to use formal methods on deep learning

| Classical software | Machine learning |
|---|---|
| Explicit control flow | Generated control flow |
| Explicit specifications | Data-driven specifications (lack of generality) |
| Abstractions and well known concepts | Very few abstractions and reusability |
| Documented and understood vulnerabilites | Flaws without systematic characterization |

Some differences between classical software and machine learning

# Another difficulty : performance of verification tools




2 cases per ReLU node for the solvers

Several million ReLU nodes $\rightarrow$ $2^{O(10^6)}$ case splits

Combinatory explosion (if done naively)

# Take home message

1. A combinatorial problem
2. A specification problem

# Tricks of the trade

## Local properties : adversarial robustness

For a *given* input $\mathbf{x}$, a classification function $f$, an adversarial perturbation $\delta$ :

find delta

satisfying

classifier misclassification

such that

perturbation stays below a certain threshold

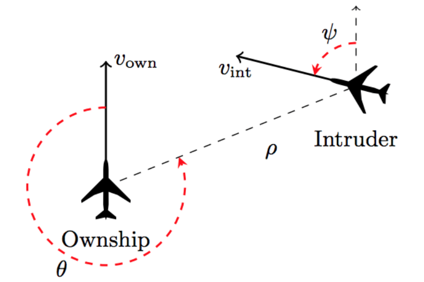
# Local properties : adversarial robustness

For a *given* input $\mathbf{x}$, a classification function $f$, an adversarial perturbation $\delta$ :

find delta

satisfying $$f(x) \neq f(x + \delta)$$

such that $$\|\delta\|_p \leq \varepsilon$$




*If the intruder is distant and is significantly slower than the ownship, the score of a COC advisory will always be below a certain fixed threshold.*

*Bounds* : $\rho \geq 55947.691, v_{own} \geq 1145, v_{int} \leq 60$

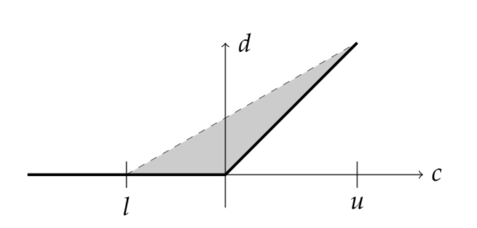
Critical system

# NN specifics functions (1)

SMT

```
define-fun relu (Int) Int (ite (>= x 0) x 0)
define-fun max (Int Int) Int (+ y relu((- x y)))
```

- $\hat{z} = ReLu(z) = max(z, 0)$
- $u$ : upper bound, $l$ : lower bound
- overapproximation : $\hat{z} \geq 0, \hat{z} \geq z, -uz + (u - l)\hat{z} \leq -ul$ (Ehlers et al., 2017)




MILP

- $\hat{z} = ReLu(z)$
- $\hat{z} \leq zl(1-a) \wedge (\hat{z} \geq z) \wedge (\hat{z} \leq ua) \wedge (\hat{z} \geq 0) \wedge (a \in (0,1))$ (Tjeng et al.,2019)

# NN specifics functions (2)

For abstract intepretation techniques (Vechev's team), abstract transformers for ReLus, Linear, Conv, Sigmoid, Tanh, MaxPool... (Mirman et al., 2018, Singh et al., 2019) over the zonotope and hybrid zonotope domain (Goubault et Putot, 2008)

- For a matrix $M$ : $T_f^{\#}(h) = \langle M \cdot h_C, M \cdot h_B, M \cdot h_E \rangle$
  Includes sum, scalar multiplication, convolutions...
- For ReLUs :
  - if $u \leq 0$, propagate 0
  - if $l \geq 0$, propagate the value
  - if phase is not clear, add a noise symbol and propagate linear approximation (linear transformer not accurate for very deep networks)

# Lower bound on adversarial robustness (Weng et al., 2018, Singh et al., 2018, Boopathy et al., 2019)

- Basic idea : propagation of constraints in the network
- Constraints : $A * W + B$ for IBM
- $\delta < \varepsilon$ for ZTH

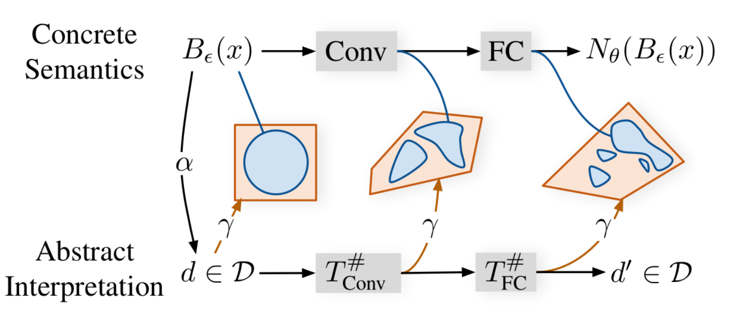
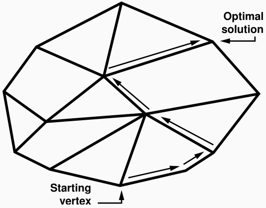



Illustration of workflow from Mirman et al.,2018

Other approaches such as symbolic propagation (Wang et al. 2018, Yang et al. 2019)

Improve adversarial robustness on 100 samples from CIFAR-10 from 0 to 80%, $\varepsilon = 8/255$, 3 hidden layers, convolutional network

# Lazy evaluation of ReLUs (Katz et al. 2017, Katz et al. 2019)




Simplex with ReLUs

New class of variables : ReLUs pairs : $b = ReLU(a)$

If $a \geq 0$ then $a = b$, else $b = 0$

1. start for an initial set of constraints $\mathcal{S}$ on variables $v_i \in \mathcal{V}$
2. if $v_i$ violates a constraint $s_i \in \mathcal{S}$, add a constraint $s_j$ on $v_j, j \neq i$ that solve $s_i$ (pivot)
3. if it is not possible to add $s_j$, do a case split
4. repeat until convergence (SAT, UNSAT, TIMEOUT)

Exact verification of several *global properties* on a ACAS-Xu implementation

# Some search strategies (Tedrake et al. 2018, Wong et Kolter 2018, Singh et al. 2018)

MILP : progressive computation of tighter bounds and presolving using basic domain knowledge

Combine MILP with abstract intepretation to compute tighter bounds :
$fp(\frac{u}{u-l})z + fp(-\frac{ul}{2(u-l)}) + fp(-\frac{ul}{2(u-l)}) * \varepsilon_{new}$ (Singh et al.)

Search strategies : solve first neurons with high weight and high $u - l$

Alternatively, linear relaxations with LP (dual formulation)

# Some possible enhancements

# This slide will be horrible

- Jointly contraint groups of ReLUs instead of linearising them independantly
- Start from backward reasoning then propagates again : bound refinement
- inputs dependancy, such as pixels correlation
- add another metric using the learning dataset
- use verification to output a class of counterexample
- new classification paradigm : activated ReLUs
- pruning networks to enhance verification
- ML can help too (active learning, learning to solve SMT Formula)

# New properties to verify ?

One big challenge unaddressed here : property formulation

# New properties to verify ?

One big challenge unaddressed here : property formulation

# Questions ?

Questions ? :)