Proving that safety-critical neural networks do what they're supposed to!

Lindsey Kuper (Parallel Computing Lab, Intel Labs)
Never Graduate Week
May 8, 2017

This talk is about work being done in collaboration with:
Guy Katz (CS @ Stanford)
Clark Barrett (CS @ Stanford)
Mykel Kochenderfer (Aeronautics & Astronautics @ Stanford)
Tatiana Shpeisman (Parallel Computing Lab, Intel Labs)
Justin Gottschlich (Parallel Computing Lab, Intel Labs)

Verifying safety-critical software

Safety-critical system:
"A computer, electronic or electromechanical system whose failure may cause injury or death to human beings"
(foldoc.org/safety-critical system)

Software verification:
Using formal methods to rigorously prove that certain properties hold of a program

Safety-critical systems use DNNs...

End to End Learning for Self-Driving Cars

Mariusz Bojarski NVIDIA Corporation Holmdel, NJ 07735

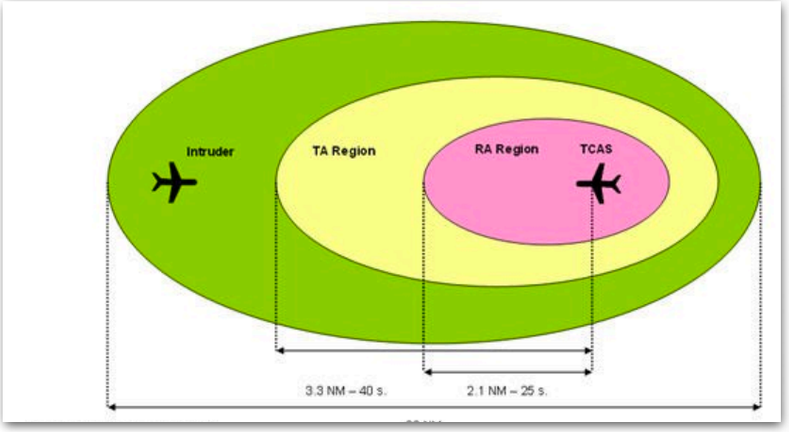
Davide Del Testa NVIDIA Corporation Holmdel, NJ 07735

Daniel Dworakowski NVIDIA Corporation Holmdel, NJ 07735

Bernhard Firner NVIDIA Corporation Holmdel, NJ 07735

Beat Flepp **Prasoon Goyal** **Lawrence D. Jackel** **Mathew Monfort**

"End to End Learning for Self-Driving Cars" (NVIDIA, 2016)



"Policy Compression for Aircraft Collision Avoidance Systems" (Julian *et al.*, 2016)

Policy Compression for Aircraft Collision Avoidance Systems

Kyle D. Julian*, Jessica Lopez†, Jeffrey S. Brush†, Michael P. Owen‡ and Mykel J. Kochenderfer*
*Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, 94305
†Applied Physics Laboratory, Johns Hopkins University, Laurel, MD, 20723
‡Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, MA, 02420

***Abstract*—One approach to designing the decision making logic for an aircraft collision avoidance system is to frame the problem as Markov decision process and optimize the system using dynamic programming. The resulting strategy can be represented as a numeric table. This methodology has been used in the development of the ACAS X family of collision avoidance systems for manned and unmanned aircraft. However, due to the high dimensionality of the state space, discretizing the state variables can lead to very large tables. To improve storage efficiency, we propose two approaches for compressing the lookup table. The first approach exploits redundancy in the table. The table is decomposed into a set of lower-dimensional tables, some of which can be represented by single tables in areas where the lower-dimensional tables are identical or nearly identical with respect to a similarity metric. The second approach uses a deep neural network to learn a complex non-linear function approximation of the table. With the use of an asymmetric loss function and a**

is extremely large, requiring hundreds of gigabytes of floating point storage. A simple technique to reduce the size of the score table is to downsample the table after dynamic programming. To minimize the deterioration in decision quality, states are removed in areas where the variation between values in the table are smooth. This allows the table to be downsampled with only minor impact on overall decision performance. The downsampling reduces the size of the table by a factor of 180 from that produced by dynamic programming. For the rest of this paper, we refer to the downsampled ACAS Xu horizontal table as our baseline, original table.

Even after downsampling, the current table requires over 2GB of floating point storage. Discretized score tables like this have been compressed with Gaussian processes [6] and

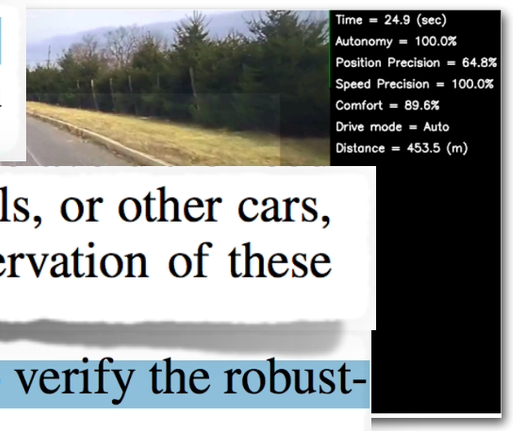
Safety-critical systems use DNNs...

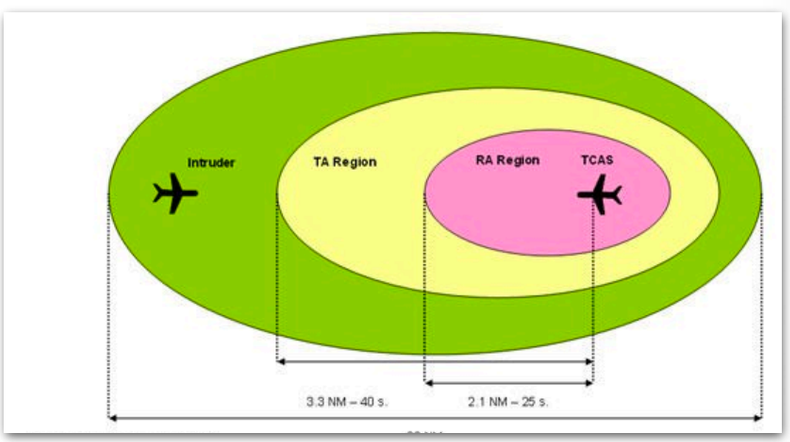
"End to End Learning for Self-Driving Cars" (NVIDIA, 2016)

End to End Learning for Self-Driving Cars

We trained a convolutional neural network (CNN) to map raw pixels from a single front-facing camera directly to steering commands. This end-to-end approach proved surprisingly powerful. With minimum training data from humans the system learns to recognize specific human-designated features, such as lane markings, guard rails, or other cars, and to avoid having to create a collection of "if, then, else" rules, based on observation of these features. This paper describes preliminary results of this new effort.

More work is needed to improve the robustness of the network, to find methods to verify the robustness, and to improve visualization of the network-internal processing steps.





"Policy Compression for Aircraft Collision Avoidance Systems" (Julian *et al.*, 2016)

Policy Compression for Aircraft Collision Avoidance Systems

Kyle D. Julian*, Jessica Lopez†, Jeffrey S. Brush†, Michael P. Owen‡ and Mykel J. Kochenderfer*
*Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, 94305
†Applied Physics Laboratory, Johns Hopkins University, Laurel, MD, 20723
‡Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, MA, 02420

***Abstract*—One approach to designing the decision making logic for an aircraft collision avoidance system is to frame the problem as Markov decision process and optimize the system using dynamic programming. The resulting strategy can be represented as a numeric table. This methodology has been used in the development of the ACAS X family of collision avoidance systems for manned and unmanned aircraft. However, due to the high dimensionality of the state space, discretizing the state variables can lead to very large tables. To improve storage efficiency, we propose two approaches for compressing the lookup table. The first approach exploits redundancy in the table. The table is decomposed into a set of lower-dimensional tables, some of which can be represented by single tables in areas where the lower-dimensional tables are identical or nearly identical with respect to a similarity metric. The second approach uses a deep neural network to learn a complex non-linear function approximation of the table. With the use of an asymmetric loss function and a**

is extremely large, requiring hundreds of gigabytes of floating point storage. A simple technique to reduce the size of the score table is to downsample the table after dynamic programming. To minimize the deterioration in decision quality, states are removed in areas where the variation between values in the table are smooth. This allows the table to be downsampled with only minor impact on overall decision performance. The downsampling reduces the size of the table by a factor of 180 from that produced by dynamic programming. For the rest of this paper, we refer to the downsampled ACAS Xu horizontal table as our baseline, original table.

Even after downsampling, the current table requires over 2GB of floating point storage. Discretized score tables like this have been compressed with Gaussian processes [6] and

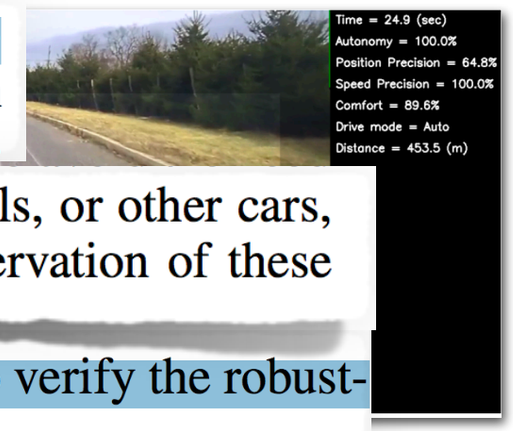
Safety-critical systems use DNNs...

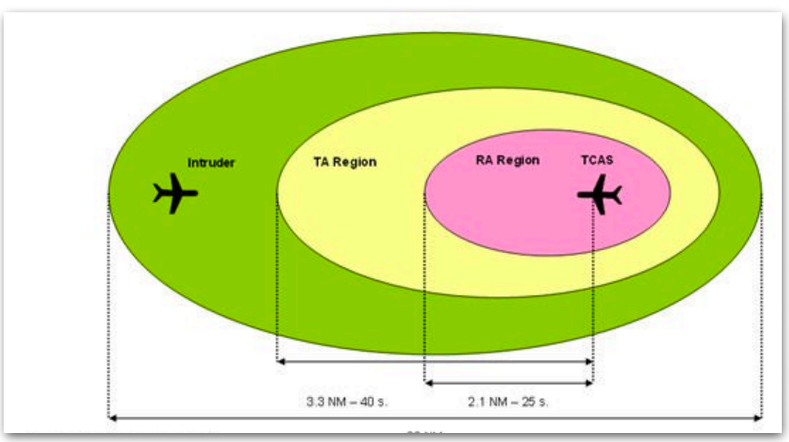
"End to End Learning for Self-Driving Cars" (NVIDIA, 2016)

End to End Learning for Self-Driving Cars

We trained a convolutional neural network (CNN) to map raw pixels from a single front-facing camera directly to steering commands. This end-to-end approach proved surprisingly powerful. With minimum training data from humans the system learns to recognize specific human-designated features, such as lane markings, guard rails, or other cars, and to avoid having to create a collection of "if, then, else" rules, based on observation of these features. This paper describes preliminary results of this new effort.

More work is needed to improve the robustness of the network, to find methods to verify the robustness, and to improve visualization of the network-internal processing steps.





"Policy Compression for Aircraft Collision Avoidance Systems" (Julian *et al.*, 2016)

Policy Compression for Aircraft Collision Avoidance Systems

to a similarity metric. The second approach uses a deep neural network to learn a complex non-linear function approximation of the table. With the use of an asymmetric loss function and a preserving the relative preferences of the possible advisories for each state. As a result, the table can be approximately represented by only the parameters of the network, which reduces the required storage space by a factor of 1000. Simulation studies show that system performance is very similar using either

crete representation. Although there are significant certification concerns with neural network representations, which may be addressed in the future, these results indicate a promising way

...even if they don't like to advertise it

After Mastering Singapore's Streets, NuTonomy's Robo-taxis Are Poised to Take on New Cities

An AI alternative to deep learning makes it easier to debug the startup's self-driving cars

(spectrum.ieee.org/transportation/self-driving/after-mastering-singapores-streets-nutonomys-robotaxis-are-poised-to-take-on-new-cities)

...even if they don't like to advertise it

After Mastering Singapore's Streets, NuTonomy's Robo-taxis Are Poised to Take on New Cities

An AI alternative to deep learning makes it easier to

Formal logic, on the other hand, gives you provable guarantees that the car will obey the rules required to stay safe even in situations that it's otherwise completely unprepared for, using code that a human can read and understand. "It's a rigorous algorithmic process that's translating specifications on how the car should behave into verifiable software," explains nuTonomy CEO and cofounder Karl Iganemma. "That's something that's

to figure out the underlying rules of good driving, then apply those rules to scenarios that it hasn't seen before. This approach has been generally successful for many self-driving cars, and in fact nuTonomy is using machine learning to help with the much different problem of interpreting sensor data—just not with decision making. That's because it's very hard to figure out why

(spectrum.ieee.org/transportation/self-driving/after-mastering-singapores-streets-nutonomys-robotaxis-are-poised-to-take-on-new-cities)

Verifying DNNs is an open problem

Verifying DNNs is an open problem

- 2003: Neural networks "**represent a class of systems that do not fit into the current paradigms of software development and certification**" (Taylor *et al*. 2003)

Verifying DNNs is an open problem

- 2003: Neural networks "**represent a class of systems that do not fit into the current paradigms of software development and certification**" (Taylor *et al*. 2003)
- 2010: an SMT solver that could verify safety properties of small networks of 10-20 neurons (Pulina and Tacchella 2010)

Verifying DNNs is an open problem

- 2003: Neural networks "**represent a class of systems that do not fit into the current paradigms of software development and certification**" (Taylor *et al*. 2003)
- 2010: an SMT solver that could verify safety properties of small networks of 10-20 neurons (Pulina and Tacchella 2010)
- 2016: "When possible, **it is desirable for systems in safety-critical situations, for example, self-driving cars, to be verifiable** [...] but **much remains to be done before it will be possible**" (Russell *et al*. 2016, "Research Priorities for Robust and Beneficial Artificial Intelligence"; see futureoflife.org/ai-open-letter/; arxiv.org/abs/1602.03506)

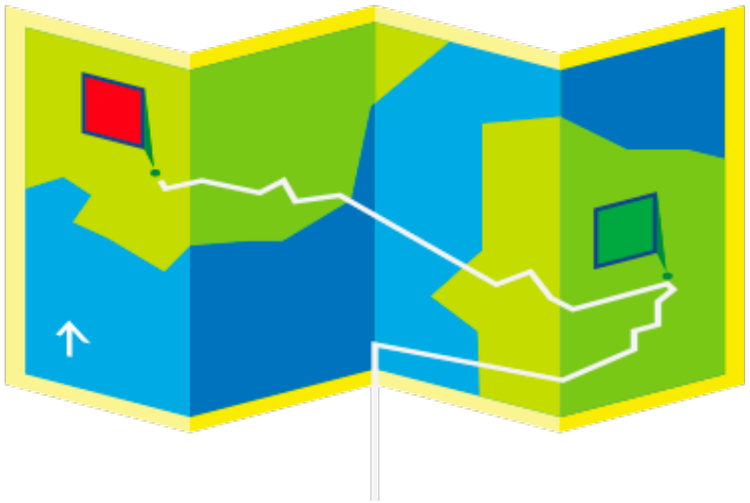
Verifying DNNs is an ~~open problem~~ opportunity!

- 2003: Neural networks "**represent a class of systems that do not fit into the current paradigms of software development and certification**" (Taylor *et al*. 2003)
- 2010: an SMT solver that could verify safety properties of small networks of 10-20 neurons (Pulina and Tacchella 2010)
- 2016: "When possible, **it is desirable for systems in safety-critical situations, for example, self-driving cars, to be verifiable** [...] but **much remains to be done before it will be possible**" (Russell *et al*. 2016, "Research Priorities for Robust and Beneficial Artificial Intelligence"; see futureoflife.org/ai-open-letter/; arxiv.org/abs/1602.03506)

Goal of this project

Create **automated verification tools**, such as **SMT solvers**, that can **verify properties of trained DNN models** used in real-world **safety-critical systems**

The rest of this talk

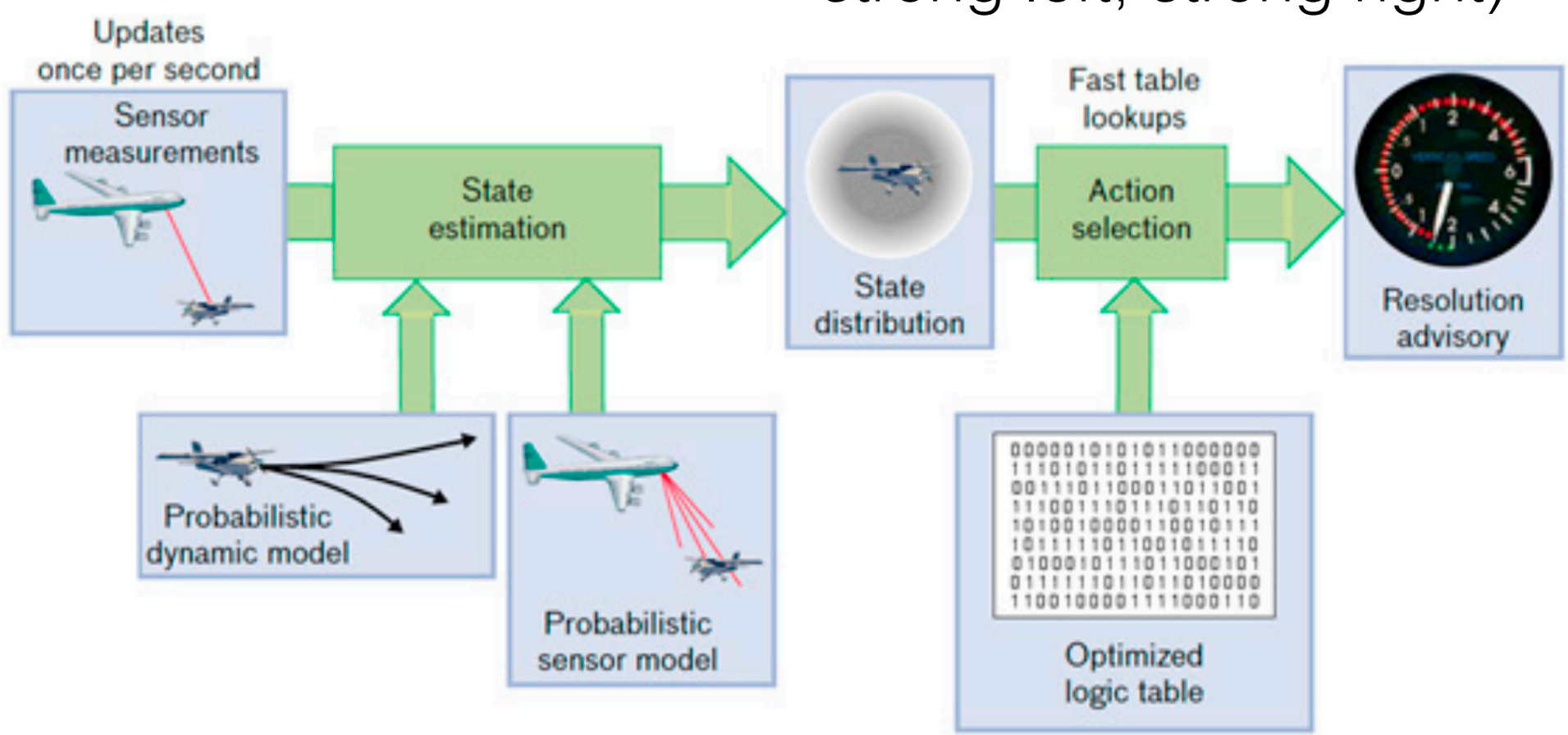


- Motivating problem
  - The ACAS Xu flight collision avoidance system, its DNN implementation, and what we want to verify about it
- Verifying DNNs with SMT solvers
  - Quick intro to SMT solving
  - What makes (D)NNs a challenge for SMT solvers?
  - How our verification tool works
  - What we can verify so far about the ACAS Xu system
- Future plans and key takeaways

The ACAS X system

Input: sensor data once per second

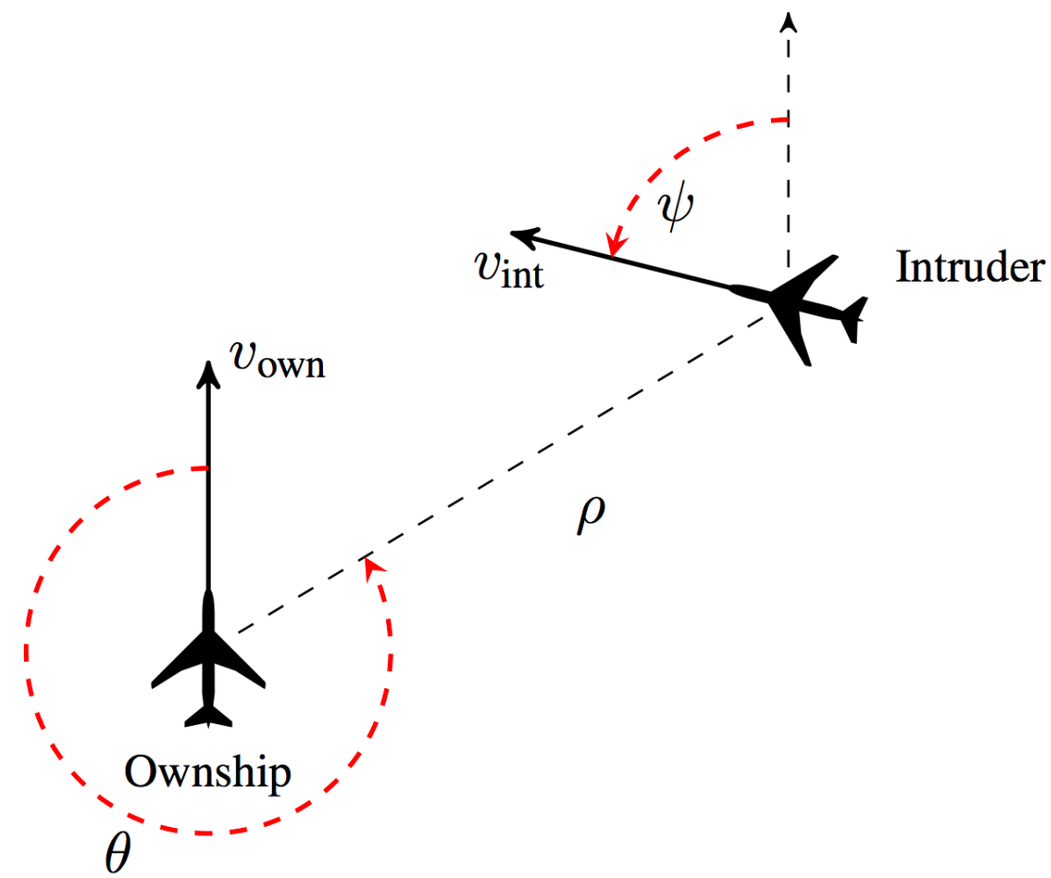
Output: one of 5 resolution advisories (COC, weak left, weak right, strong left, strong right)



(www.ll.mit.edu/publications/technotes/TechNote_ACASX.pdf)

The ACAS Xu score table

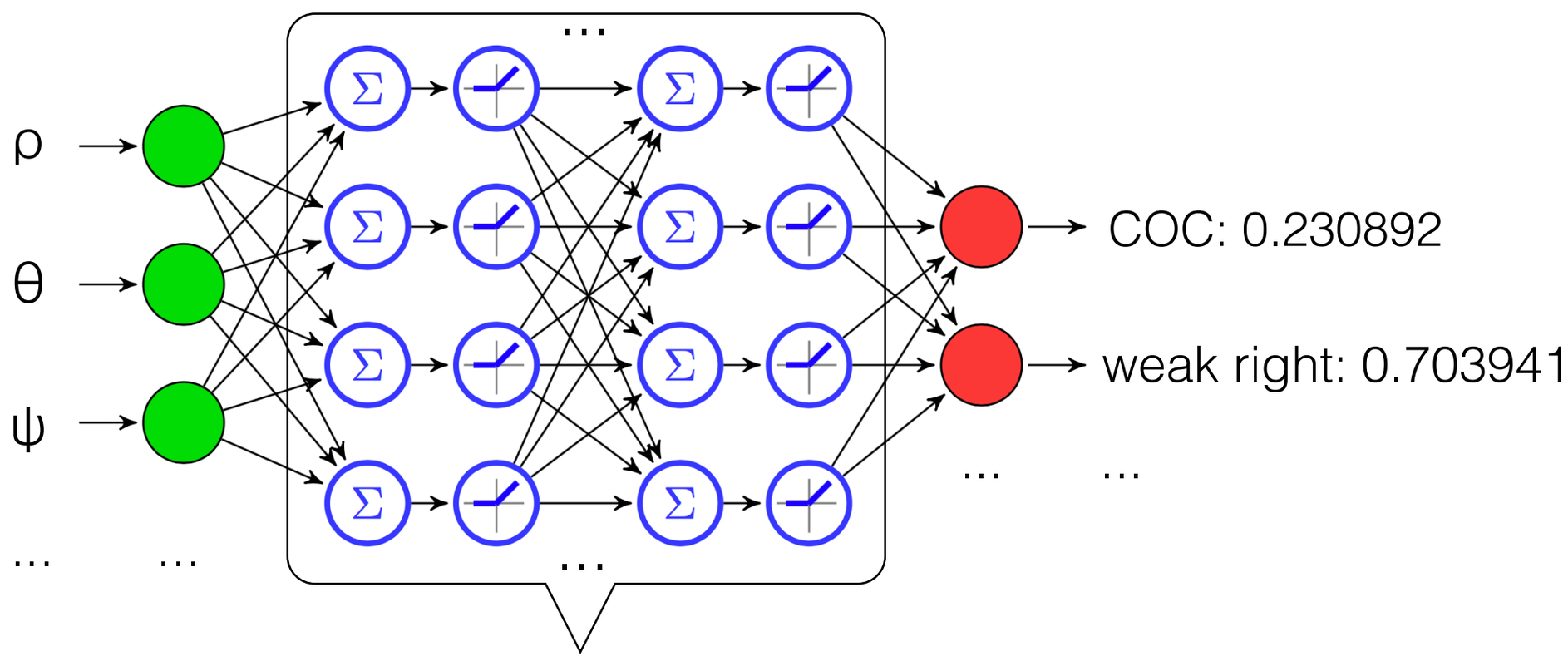
- Associates a score with each (state, advisory) pair
- 7-dimensional state:
  - ρ: distance from ownship to intruder
  - θ: angle to intruder
  - ψ: heading angle of intruder
  - v_{own}: speed of ownship
  - v_{int}: speed of intruder
  - τ: time until loss of vertical separation
  - a_{prev}: previous advisory



Needs 100s of GB of storage—too big for verified hardware!

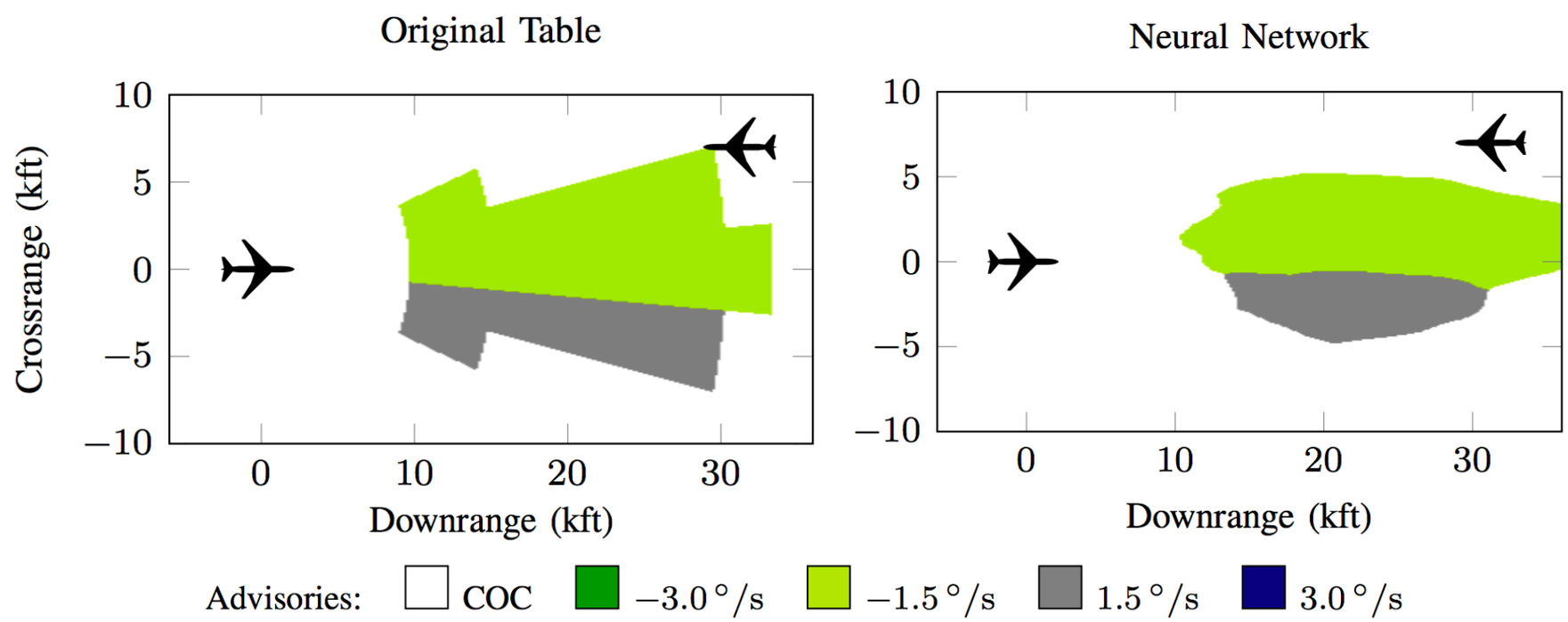
The ACAS Xu deep neural network

- Trained on the score table
- Input: [ρ, θ, ψ, vown, vint, τ, aprev]
- Output: Score associated with each of 5 possible advisories
- ~600K parameters; trained model just a few MB



9 fully-connected layers w/ ReLU activations (*f(x) = max(0, x)*)

Can we trust the ACAS Xu DNN?

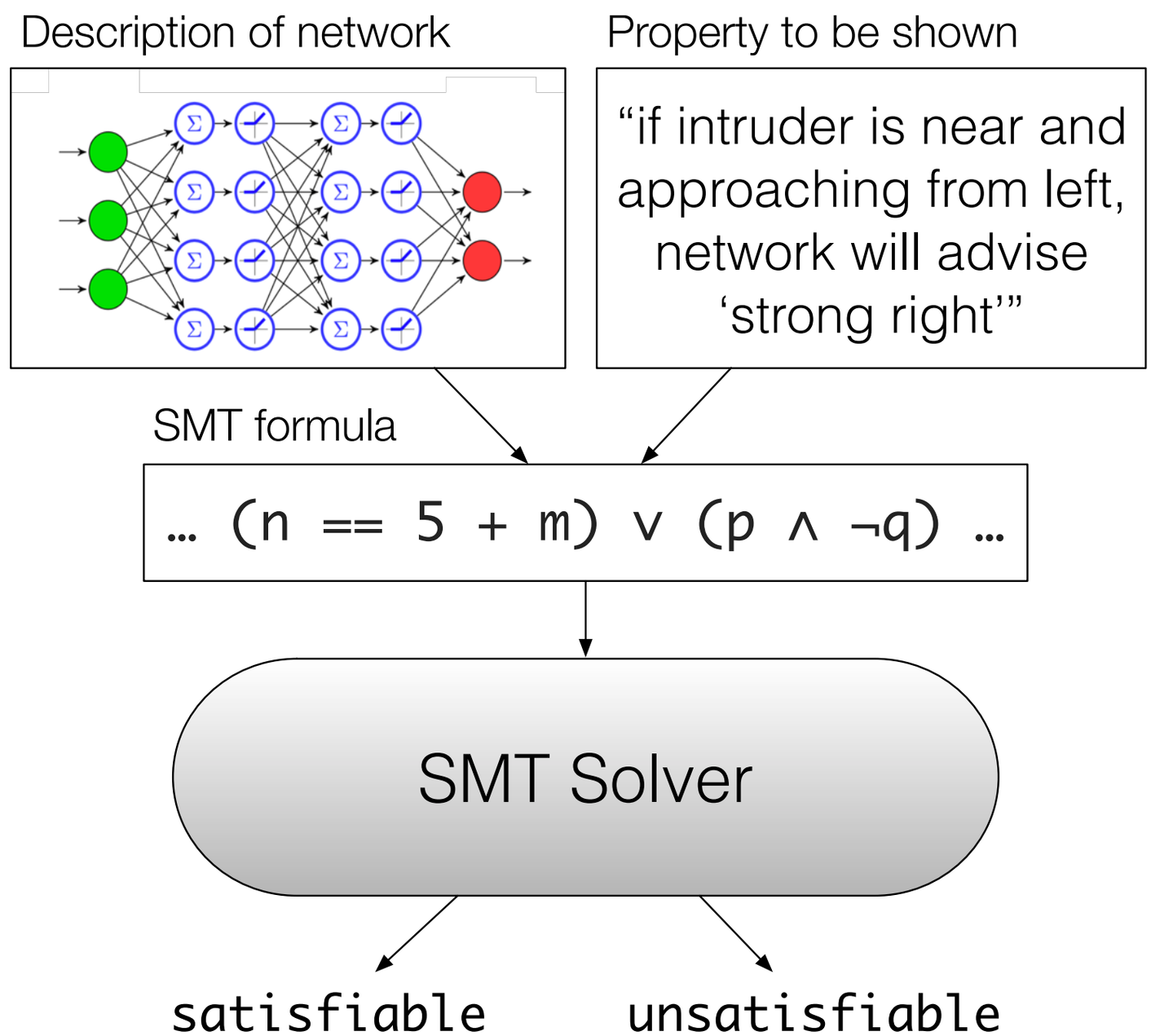


- It's only an approximation of the original table
- Is it safe? We want to verify properties such as:
  - If intruder is directly ahead and moving towards ownship, score for COC will not be minimal
  - If intruder is near and approaching from left, network will advise "strong right"
  - ...and more (see arxiv.org/abs/1702.01135)

Verification strategy

SAT: determine if a Boolean formula (containing only Boolean variables, parens, ∧, ∨, ¬) is satisfiable

SMT: determine satisfiability of a formula with respect to some *theory* (*e.g.*, theory of linear real arithmetic)

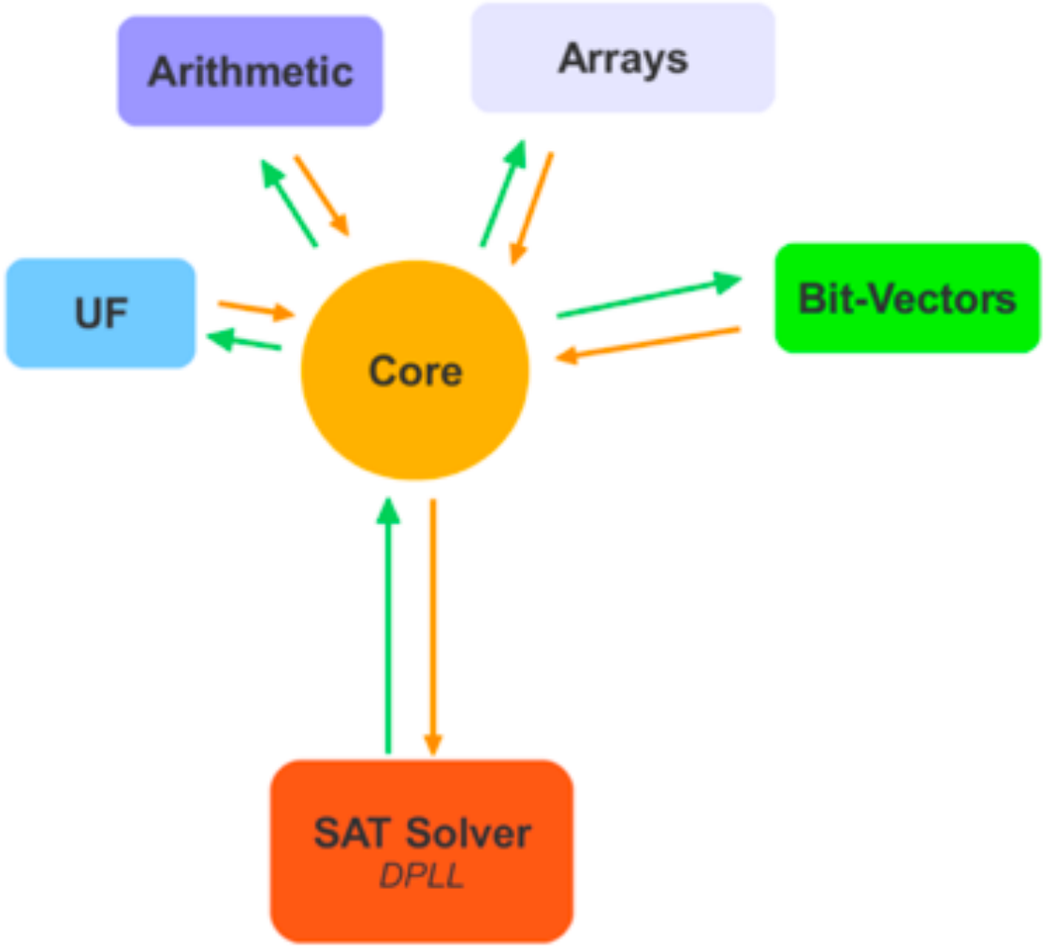


The virtues of laziness

Eager approach: convert whole SMT formula to SAT formula immediately, then solve with SAT solver

Lazy approach: use *theory solvers*, each specific to a particular theory—allows exploiting domain knowledge for efficiency

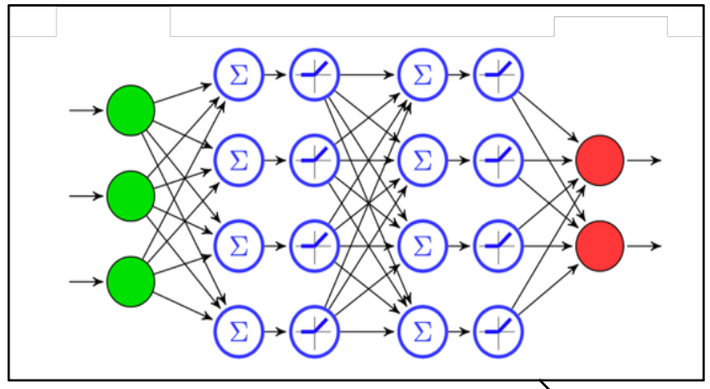
To exploit laziness, we need theory solvers specially for handling DNNs!



Lazy SMT solver architecture
(from fm.csl.sri.com/SSFT16/slides.pdf)

Lazily handling ReLU activations

Description of network



Property to be shown

"if intruder is near and approaching from left, network will advise 'strong right'"

SMT formula

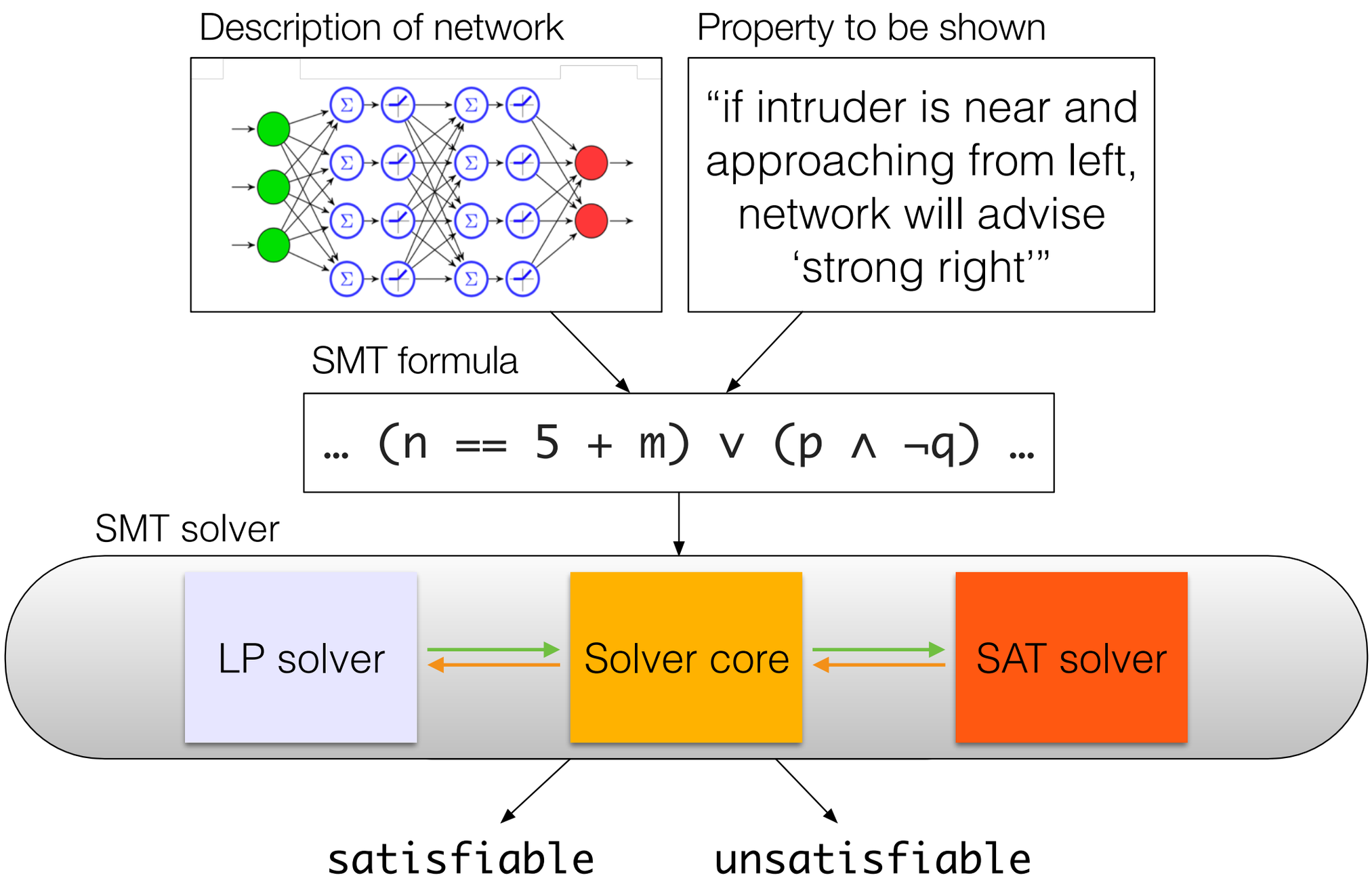
… (n == 5 + m) ∨ (p ∧ ¬q) …

SMT Solver

satisfiable

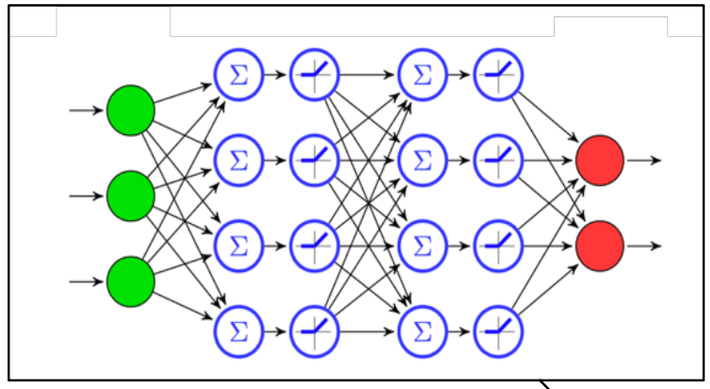
unsatisfiable

Lazily handling ReLU activations



Lazily handling ReLU activations

Description of network



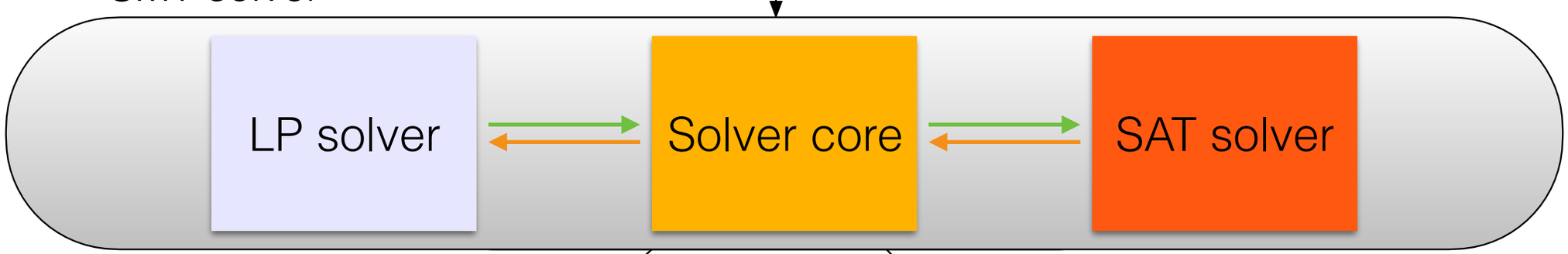
Property to be shown

"if intruder is near and approaching from left, network will advise 'strong right'"

SMT formula

$$\ldots\ (n == 5 + m) \vee (p \wedge \neg q)\ \ldots$$

SMT solver



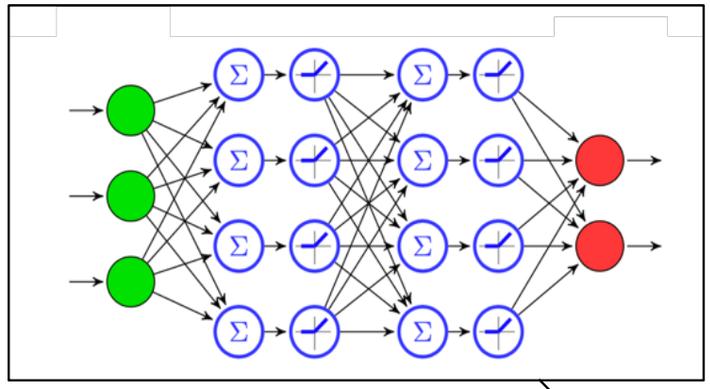
`satisfiable` `unsatisfiable`

ReLU constraints like `x = max(0, y)` can only be encoded as disjunctions!

$$(x \geq 0 \wedge x = y) \vee (x < 0 \wedge x = 0)$$

Lazily handling ReLU activations

Description of network



Property to be shown

"if intruder is near and approaching from left, network will advise 'strong right'"

SMT formula

… (n == 5 + m) ∨ (p ∧ ¬q) …

SMT solver

LP + ReLU solver ⇄ Solver core ⇄ SAT solver

satisfiable unsatisfiable

ReLU constraints like `x = max(0, y)` can only be encoded as disjunctions!

$(x \geq 0 \wedge x = y) \vee (x < 0 \wedge x = 0)$

What can we verify about ACAS Xu?

Property description	Does it hold?	Solver time	Max. ReLU split depth (out of 300)
"if intruder is directly ahead and is moving towards ownship, score for COC will not be minimal"	✓	7.8h	22
"if intruder is near and approaching from left, network advises 'strong right'"	✓	5.4h	46
"if intruder is sufficiently far away, network advises COC"	✓	50h	50
"for large vertical separation and previous 'weak left' advisory, network will either advise COC or continue advising 'weak left'"	✗	11h	69

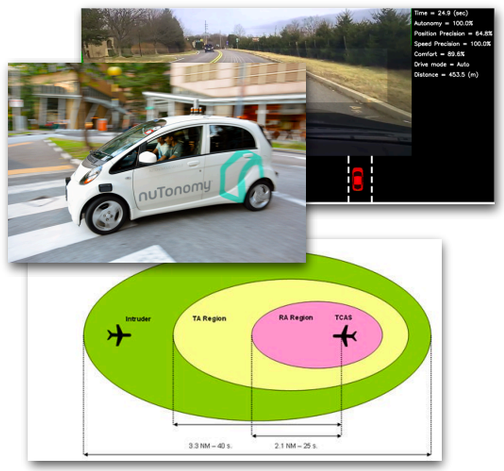
...and more (see arxiv.org/abs/1702.01135)

What's next?

- Support activation functions other than ReLU
- Support network architectures other than fully-connected networks (like convolutional networks!)
- Apply to more safety-critical systems beyond ACAS Xu
  - What do you want verified?

Takeaways from this talk

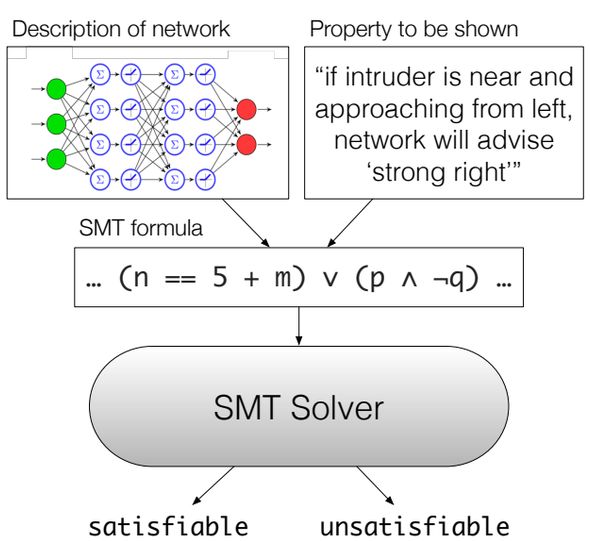
Takeaways from this talk



Deep learning in safety-critical systems is probably here to stay

Takeaways from this talk

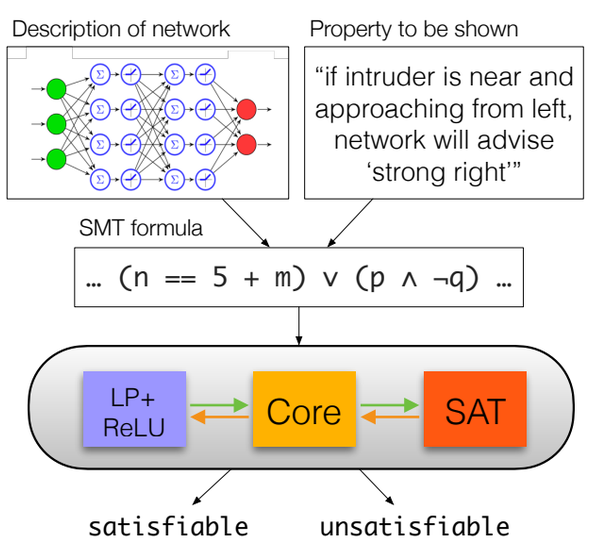
Deep learning in safety-critical systems is probably here to stay



It *is* possible to verify important properties of DNNs using SMT solvers

Takeaways from this talk

Deep learning in safety-critical systems is probably here to stay



It *is* possible to verify important properties of DNNs using SMT solvers

Off-the-shelf solvers aren't enough—we need DNN-specific theory solvers to exploit the "lazy" SMT approach