

Neural Network Verification with Vehicle

Ekaterina Komendantskaya and Matthew Daggitt (today's presentors), on behalf of the Vehicle team

The Vehicle Team



Matthew Daggitt



Wen Kokke



Bob Atkey



Rob Stewart



Luca Arnaboldi

Kathrin Stark



Marco Casadio





Table of Contents



Neural Network Verification: overview of the new domain

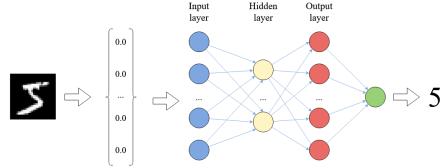
The lifecycle of neural network verification

Challenges and Languages

Vehicle's Role and Purpose of this Tutorial

Neural nets for classification





Formally,

a neural network is a function $N: \mathbb{R}^n \to \mathbb{R}^m$.



... are ideal for "perception" tasks:

- ▶ approximate functions when exact solution is hard to get
- ▶ tolerant to noisy and incomplete data



... are ideal for "perception" tasks:

- ▶ approximate functions when exact solution is hard to get
- ▶ tolerant to noisy and incomplete data



... are ideal for "perception" tasks:

- ▶ approximate functions when exact solution is hard to get
- ▶ tolerant to noisy and incomplete data



... are ideal for "perception" tasks:

- ▶ approximate functions when exact solution is hard to get
- ▶ tolerant to noisy and incomplete data

BUT

- ➤ solutions not easily conceptualised (lack of explainability)
- ▶ prone to a new range of safety and security problems:



... are ideal for "perception" tasks:

- ▶ approximate functions when exact solution is hard to get
- ▶ tolerant to noisy and incomplete data

BUT

- ▶ solutions not easily conceptualised (lack of explainability)
- ▶ prone to a new range of safety and security problems: adversarial attacks data poisoning catastrophic forgetting



























the perturbations are imperceptible to human eye









the perturbations are imperceptible to human eye attacks transfer from one neural network to another





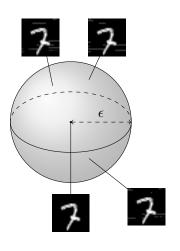




the perturbations are imperceptible to human eye attacks transfer from one neural network to another affect any domain where neural networks are applied

Verification Property: " ϵ -ball robustness"





An ϵ -ball $\mathbb{B}(\hat{\mathbf{x}}, \epsilon) = {\mathbf{x} \in \mathbb{R}^n : |\hat{\mathbf{x}} - \mathbf{x}| \le \epsilon}$

Classify all points in $\mathbb{B}(\hat{\mathbf{x}}, \epsilon)$ "robustly".

Another example property: ACAS Xu



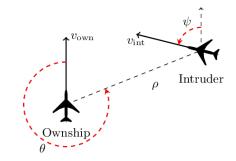
A collision avoidance system for unmanned autonomous aircraft.

Inputs:

- \triangleright Distance to intruder, ρ
- \triangleright Angle to intruder, θ
- ▶ Intruder heading, φ
- ightharpoonup Speed, v_{own}
- ightharpoonup Intruder speed, v_{int}

Outputs:

- ► Clear of conflict
- Strong left
- ▶ Weak left
- ► Weak right
- Strong right



ACAS Xu

The system was originally implemented as a 2Gb lookup table but was replaced with a neural network in order to improve size and latency requirements.

ACAS Xu

The system was originally implemented as a 2Gb lookup table but was replaced with a neural network in order to improve size and latency requirements.

10 different specified properties in total.

ACAS Xu

The system was originally implemented as a 2Gb lookup table but was replaced with a neural network in order to improve size and latency requirements.

10 different specified properties in total.

Definition (ACAS Xu: Property 1)

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.

ACAS X11

The system was originally implemented as a 2Gb lookup table but was replaced with a neural network in order to improve size and latency requirements.

10 different specified properties in total.

Definition (ACAS Xu: Property 1)

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.

$$(\rho \ge 55947.691) \land (v_{own} \ge 1145) \land (v_{int} \le 60)$$

 \Rightarrow the score for COC is at most 1500

More Generally



Given $N: \mathbb{R}^n \to \mathbb{R}^m$

Verification of such functions most commonly boils down to specifying admissible intervals for the function's output given an interval for its inputs.

More Generally



Given $N: \mathbb{R}^n \to \mathbb{R}^m$

Verification of such functions most commonly boils down to specifying admissible intervals for the function's output given an interval for its inputs.



Casadio, M., Komendantskaya, E., Daggitt, M.L., Kokke, W., Katz, G., Amir, G., Refaeli, I.: Neural network robustness as a verification property: A principled case study. In: Computer Aided Verification (CAV 2022).

Overview of The Verification Landscape





I have this specification
I want to verify!

Szegedy et al. (2013)





What tools are available? **2015**



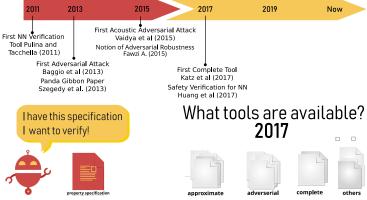


complete

others

Overview of The Verification Landscape





Overview of The Verification Landscape





\$

A whole range of domain-specific verifiers exist:

A whole range of domain-specific verifiers exist:

► Marabou (SMT technology)



A whole range of domain-specific verifiers exist:

- ► Marabou (SMT technology)
- ► ERAN (abstract interpretation + MILP)





A whole range of domain-specific verifiers exist:

- ► Marabou (SMT technology)
- ► ERAN (abstract interpretation + MILP)
- ► Verisig (interval arithmetic)
- ► AlphaBetaCROWN (linear bound propagation)
- **.** . . .

International Standards and Competitions

https://www.vnnlib.org/

\$

A whole range of domain-specific verifiers exist:

- ► Marabou (SMT technology)
- ► ERAN (abstract interpretation + MILP)
- ► Verisig (interval arithmetic)
- ► AlphaBetaCROWN (linear bound propagation)
- **.** . . .

International Standards and Competitions

https://www.vnnlib.org/

Marabou is our current choice as it is complete, and the set of expressible queries is large!



Guy Katz, Clarke Barrett, D. Dill, K. Julian, and M. Kochenderfer. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In CAV, 2017.

Table of Contents



Neural Network Verification: overview of the new domain

The lifecycle of neural network verification

Challenges and Languages

Vehicle's Role and Purpose of this Tutorial



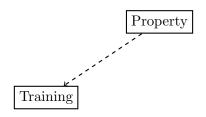
Property



Property

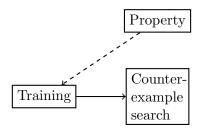
Training





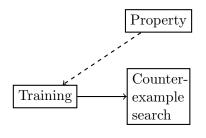
DL2 ACT etc.





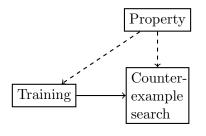
DL2 ACT etc.





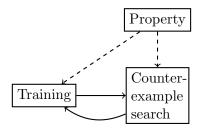
DL2 ACT etc.





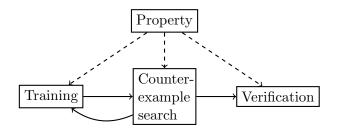
DL2 PGD ACT FGSM etc. etc.





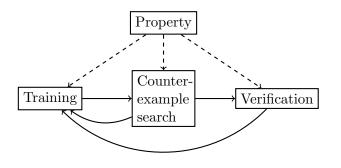
DL2 PGD ACT FGSM etc. etc.





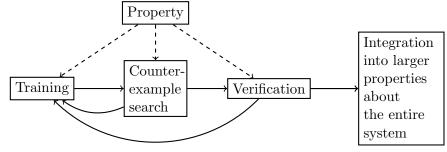
DL2 ACT etc. PGD FGSM etc. Marabou Eran etc.





DL2 ACT etc. PGD FGSM etc. Marabou Eran etc.





DL2 ACT etc. PGD FGSM etc. Marabou Eran etc.

Table of Contents



Neural Network Verification: overview of the new domain

The lifecycle of neural network verification

Challenges and Languages

Vehicle's Role and Purpose of this Tutorial



▶ Theory: finding appropriate verification properties



- ▶ Theory: finding appropriate verification properties
- ➤ Solvers: undecidability of non-linear real arithmetic and scalability of neural network verifiers



- ▶ Theory: finding appropriate verification properties
- ➤ Solvers: undecidability of non-linear real arithmetic and scalability of neural network verifiers
- ► ML: understanding and integrating property-driven training



- ▶ Theory: finding appropriate verification properties
- ▶ Solvers: undecidability of non-linear real arithmetic and scalability of neural network verifiers
- ► ML: understanding and integrating property-driven training
- ▶ Programming: finding the right languages to support these developments



- ► Theory: finding appropriate verification properties
- ➤ Solvers: undecidability of non-linear real arithmetic and scalability of neural network verifiers
- ► ML: understanding and integrating property-driven training
- Programming: finding the right languages to support these developments
- Complex systems: integration of neural net verification into complex systems



Some of these problems are aggravated by insufficient programming language or API support

Lets look under the hood...

Training framework: DL2

```
$
```

```
class RobustnessConstraint(Constraint):
126
127
141
         def get domains(self, x batches, v batches):
              assert len(x batches) == 1
142
143
              n_batch = x_batches[0].size()[0]
144
145
              return [[Box(np.clip(x batches[0][i].cpu().numpv() - self.eps, 0, 1),
146
                           np.clip(x batches[0][i].cpu().numpy() + self.eps, 0, 1))
147
                      for i in range(n batch)]]
148
149
         def get_condition(self, z_inp, z_out, x_batches, y_batches):
150
              n_batch = x_batches[0].size()[0]
151
              z_out = transform_network_output(z_out, self.network_output)[0]
152
              #z logits = F.log softmax(z out[0], dim=1)
153
154
              pred = z out[np.arange(n batch), v batches[0]]
155
156
              limit = torch.FloatTensor([0.3])
157
             if self.use cuda:
158
                  limit = limit.cuda()
159
              return dl2.GEO(pred, torch.log(limit))
```



Fischer, M., Balunovic, M., Drachsler-Cohen, D., Gehr, T., Zhang, C., and Vechev, M. T. DL2: training and querying neural networks with logic. In Proc. of the 36th Int. Conf. Machine Learning, ICML 2019

Training framework: ART



```
@classmethod
333
334
          def property6a(cls, dom: AbsDom):
              p = AcasProp(name='property6a', dom=dom, safe fn='cols is min', viol fn='cols not min',
                           fn args=[AcasOut.CLEAR OF CONFLICT])
336
337
              p.set input bound(AcasIn.RHO, new low=12000, new high=62000)
             p.set input bound(AcasIn.THETA, new low=0.7, new high=3.141592)
338
339
              p.set input bound(AcasIn.PSI, new low=-3.141592, new high=-3.141592 + 0.005)
              p.set input bound(AcasIn.V OWN, new low=100, new high=1200)
340
              p.set input bound(AcasIn.V INT, new low=0, new high=1200)
341
342
             p.set_all_applicable_as(False)
343
              p.set_applicable(1, 1, True)
344
              return p
```



Lin, X., Zhu, H., Samanta, R., and Jagannathan, S. (2020). Art: Abstraction refinement-guided training for provably correct neural networks. In FMCAD 2020

Verification framework: Marabou



```
def test_acas_1_normalize():
    """

Test the 1,1 experimental ACAS Xu network.

By passing "normalize=true" to read_nnet, Marabou adjusts the parameters of the first and last layers of the network to incorporate the normalization.
As a result, properties can be defined in the original input/output spaces without any manual normalization.
"""
filename = "acasxu/ACASXU_experimental_v2a_1_1.nnet"
testInputs = [
    [10000.9, 0.0, -1.5, 100.0, 100.0],
    [10000.0, -3.0, -1.5, 300.0, 300.0],
    [5000.0, -3.0, 0.0, 300.0, 600.0]]
]
testOutputs = [
    [177.87553729, 173.75796115, 193.05920806, 153.07876146, 195.00495022],
    [-0.55188079, 0.48863711, 0.44250383, 0.44151988, 0.43959133],
    [20.9190734, 27.2386958, 45.02497222, 14.5610455, 46.86448056]]
network = evaluateFile(filename, testInputs, testOutputs, normalize = True)
```



Katz, G., Huang, D. A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljic, A., Dill, D. L., Kochenderfer, M. J., and Barrett, C. W. (2019). The Marabou framework for verification and analysis of deep neural networks. In CAV 2019

Verification framework: ERAN



```
1 [12000, 62000]
2 [0.7, 3.141592][-3.141592, -0.7]
3 [-3.141592, -3.136592]
4 [100, 1200]
5 [0, 600]
1 5
2 y0 min
```



Singh, G., Gehr, T., Püschel, M., and Vechev, M. T. (2019). An abstract domain for certifying neural networks. PACMPL, 3(POPL):41:1–41:30.

Verification property language: VNNLIB



```
28
     (assert (or
29
         (and (<= X 0 0.700434925) (>= X 0 -0.129289109)
30
              (<= X 1 0.499999896) (>= X 1 0.11140846)
31
              (<= X 2 -0.499204121) |(>= X 2 -0.499999896)
32
              (\le X \ 3 \ 0.5) \ (\ge X \ 3 \ -0.5)
33
              (<= X 4 0.5) (>= X 4 -0.5))
34
         (and (<= X 0 0.700434925) (>= X 0 -0.129289109)
35
              (<= X 1 -0.11140846) (>= X 1 -0.499999896)
36
              (<= X 2 -0.499204121) (>= X 2 -0.499999896)
37
              (<= X_3 0.5) (>= X_3 -0.5)
38
              (\le X 4 0.5) (\ge X 4 - 0.5))
39
     ))
40
41
     : unsafe if coc is not minimal
42
     (assert (or
43
         (and (<= Y 1 Y 0))
44
        (and (<= Y 2 Y 0))
45
         (and (<= Y 3 Y 0))
46
         (and (<= Y 4 Y 0))
47
    ))
48
```





 I^O Interoperability – properties are not portable between training/counter-example search/ verification.



- I^O Interoperability properties are not portable between training/counter-example search/ verification.
- I^P Interpretability code is not easy to understand.



- Interoperability properties are not portable between training/counter-example search/verification.
- I^P Interpretability code is not easy to understand.
- I^{\int} Integration properties of networks cannot be linked to larger control system properties.

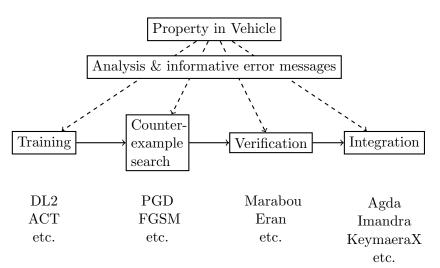


- Interoperability properties are not portable between training/counter-example search/verification.
- I^P Interpretability code is not easy to understand.
- I^{\int} Integration properties of networks cannot be linked to larger control system properties.
- E^G Embedding gap little support for translation between problem space (as in original spec) and input space (at neural network level).

Vehicle is designed to address all of these problems

Vehicle ...

is a domain-specific functional language for writing high-level property specifications for neural networks



Other Similar APIs

➤ Socrates [in Python]: Given a spec and a network (in JSON), calls different NN verifiers.





Long H. Pham, Jiaying Li, and Jun Sun. 2020. SOCRATES: Towards a Unified Platform for Neural Network Verification. CoRR abs/2007.11206 (2020).

Left unresolved: I^O , I^P , I^{\int} , E^G

Other Similar APIs

▶ Socrates [in Python]: Given a spec and a network (in JSON), calls different NN verifiers.



Left unresolved: I^O , I^P , I^{\int} , E^G

abs/2007.11206 (2020).

▶ NeVer 2.0 [in Python]: added training, prunning and quantization to this functionality.

Dario Guidotti, Luca Pulina, and Armando Tacchella. 2020. NeVer 2.0: Learning, Verification and Repair of Deep Neural Networks. CoRR abs/2011.09933 (2020).

Resolved: I^O (partially). Left untesolved: I^P , I^{\int} , E^G

Other Similar APIs

➤ Socrates [in Python]: Given a spec and a network (in JSON), calls different NN verifiers.



Long H. Pham, Jiaying Li, and Jun Sun. 2020. SOCRATES: Towards a Unified Platform for Neural Network Verification. CoRR abs/2007.11206 (2020).

Left unresolved: I^O , I^P , I^{\int} , E^G

▶ NeVer 2.0 [in Python]: added training, prunning and quantization to this functionality.



Dario Guidotti, Luca Pulina, and Armando Tacchella. 2020. NeVer 2.0: Learning, Verification and Repair of Deep Neural Networks. CoRR abs/2011.09933 (2020).

Resolved: I^O (partially). Left untesolved: I^P , I^{\int} , E^G

 Caisar [in OCAML] – general specification language and connection to several NN Verifiers



Julien Girard-Satabin, Michele Alberti, François Bobot, Zakaria Chihani, and Augustin Lemesle. 2022. CAISAR: A platform for Characterizing Artificial Intelligence Safety and Robustness. In AISafety (CEUR-Workshop Proceedings). Vienne, Austria.

Resolved: I^P , I^{\int} . Left unresolved: I^O , E^G

Table of Contents



Neural Network Verification: overview of the new domain

The lifecycle of neural network verification

Challenges and Languages

Vehicle's Role and Purpose of this Tutorial

Vehicle's Aim...



... is to resolve the problems I^O , I^P , I^{\int} , E^G

Vehicle's Aim...

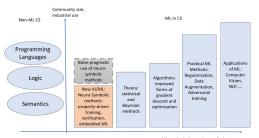


... is to resolve the problems I^O , I^P , I^{\int} , E^G

... and support community's effort towards resolution of the "Grand Challenges"

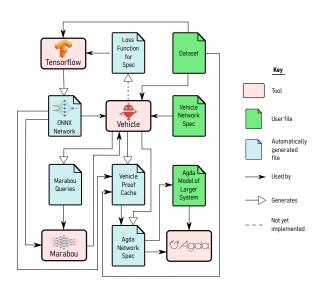
Which of the "Grand Challenges" **Vehicle** address

- ▶ Theory: finding appropriate verification properties
- Solvers: undecidability of non-linear real arithmetic and scalability of neural network verifiers
- ▶ ML: understanding and integrating property-driven training
- ▶ Programming: finding the right languages to support these developments
- ► Complex systems: integration of neural net verification into complex systems



Vehicle Architecture





Sources





M. Daggitt, R. Atkey, W. Kokke, E. Komendantskaya, L. Arnaboldi: Compiling Higher-Order Specifications to SMT Solvers: How to Deal with Rejection Constructively. CPP 2023



N. Slusarz, E. Komendantskaya, M. Daggitt, R. Stewart, K. Stark: Logic of Differentiable Logics: Towards a Uniform Semantics of DL. LPAR 2023.



Matthew L. Daggitt, Wen Kokke, Robert Atkey, Luca Arnaboldi, Ekaterina Komendantskaya: Vehicle: Interfacing Neural Network Verifiers with Interactive Theorem Provers. FOMLAS



Vehicle Team: The Vehicle language: https://github.com/vehicle-lang 2023.



M.Daggitt and W.Kokke: Vehicle User Manual. 2023.

Purpose of this Tutorial...



- ► Introduce Vehicle specification language at the user level
- ➤ Convince FOMLAS audience that it maybe a convenient tool to use (and develop)
- ► Gather feedback and obtain community support

Thanks

... to Marabou team and FOMLAS organisers for the continuing support!