Property Inference for Deep Neural Networks

Thanh Le-Cong

Research Engineer, Singapore Management University

Hanoi, August 4th 2022

Self-Introduction

Personal Information:

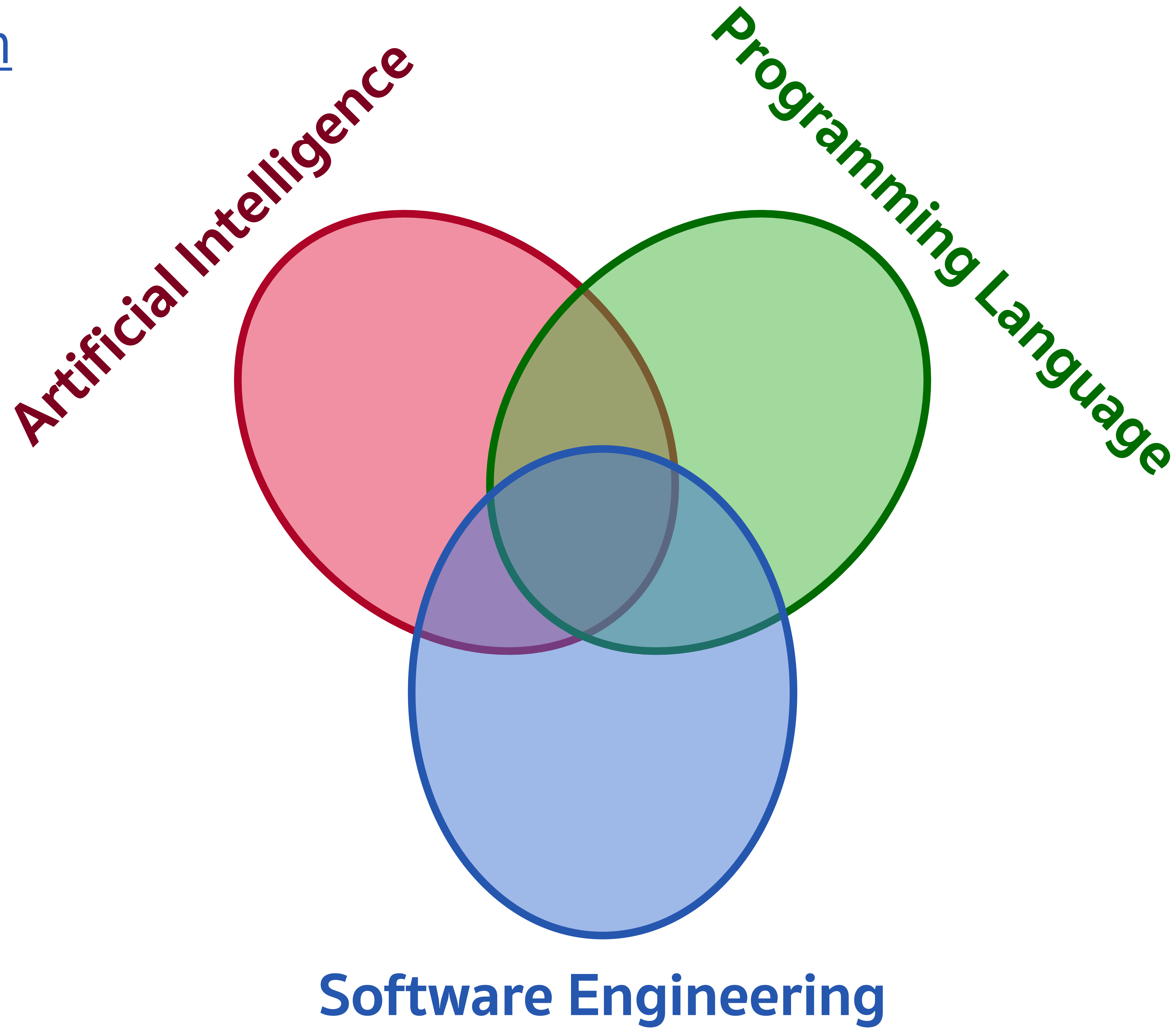
- Le - Cong Thanh
- Email: tlecong@smu.edu.sg, thanhcls1316@gmail.com
- Website: thanhlecongg.github.io

Education:

- B.Eng in Information Technology from HUST, K61

Research Interests:

- Automated Debugging:
  - Bug Detection, Localization & Repair
- Deep Neural Networks Analysis



This Talk - Overview

Agenda

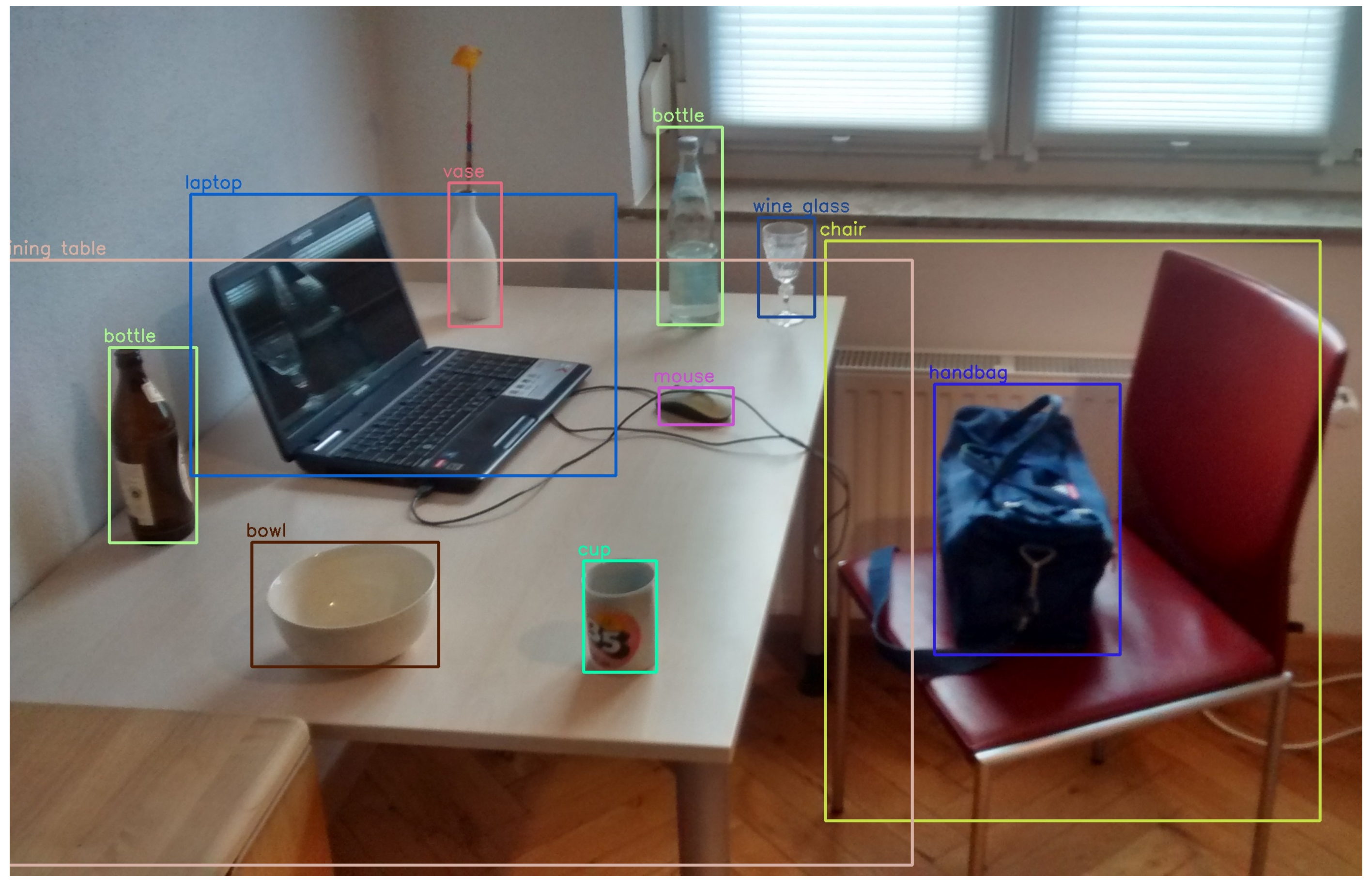
1. DNN & Its Challenges

2. Prophecy: Property Inference for Deep Neural Network

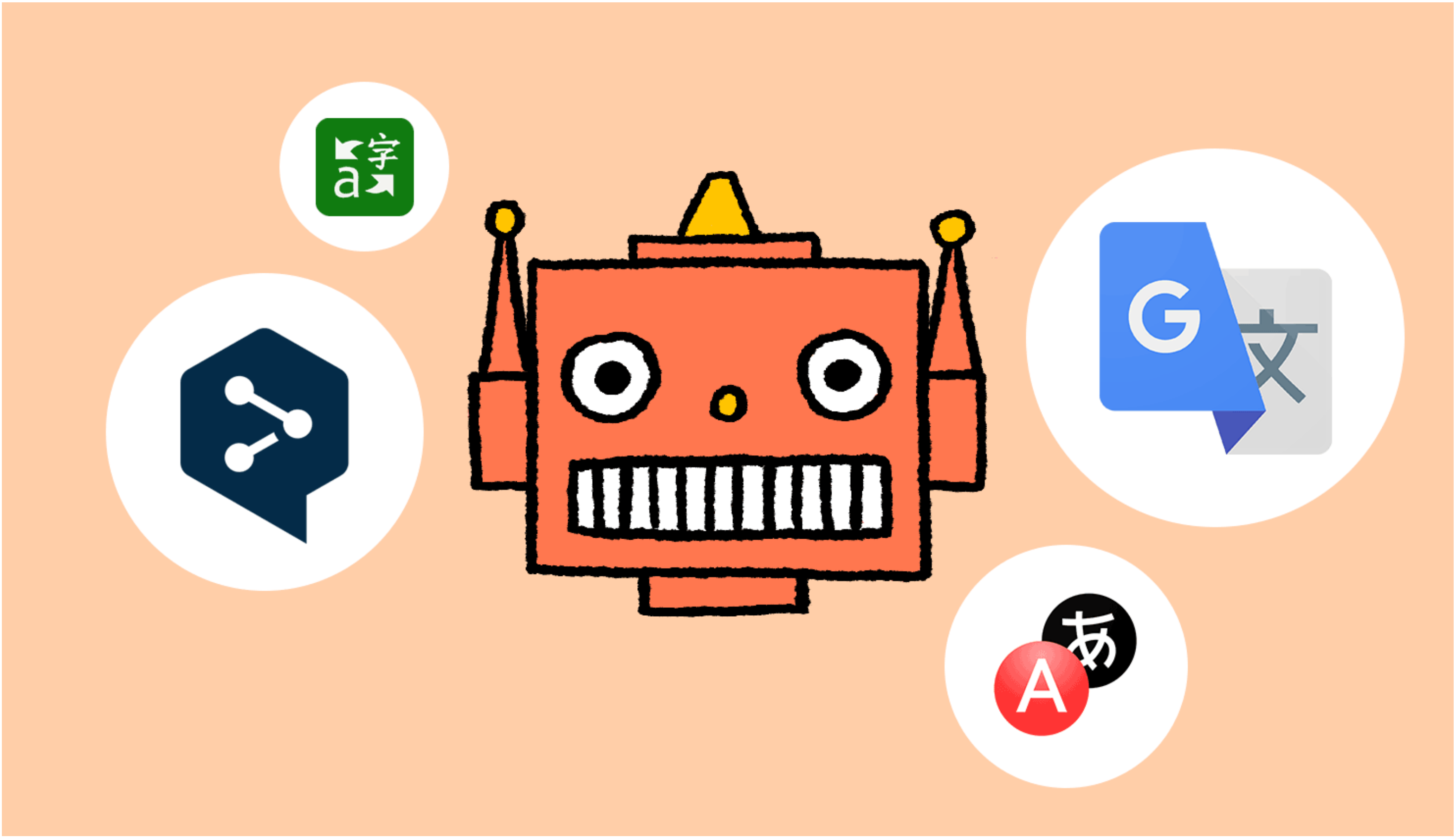
3. GNN-Infer: Towards the Analysis of Graph Neural Network

DNNs - A powerful framework for solving complex tasks, ...

Image Classification



Object Detection



Machine Translation

..., and even safety-critical tasks

Autonomous
Driving

Medical
Diagnosis

Security
System

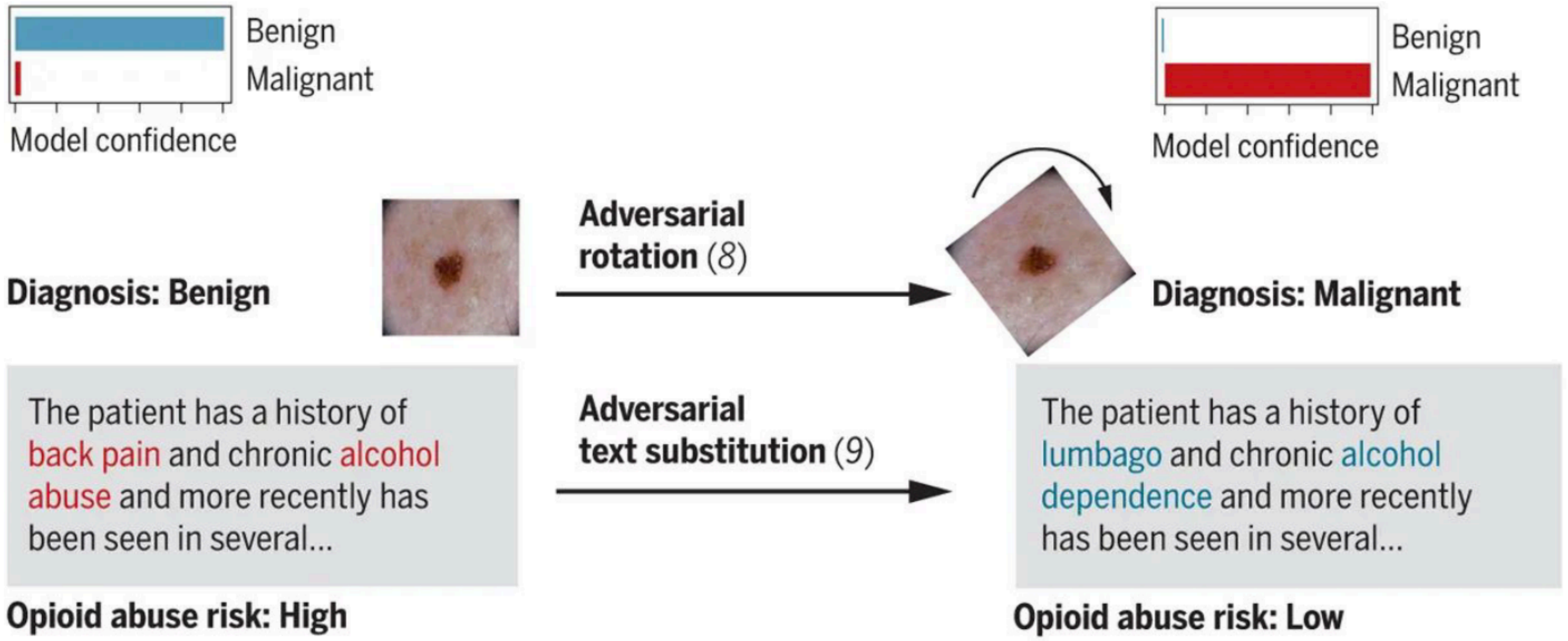
However, can we trust DNNs ?

Autonomous
Driving

Medical
Diagnosis

Security
System

Researcher say "NO"



"Adversarial attacks on medical machine learning"
by Finlayson et al., Science (2019)

Challenges

Lack of Robustness

- **Small** changes to an input may lead to **unexpected behaviours**

Lack of Explainability

- It is **not well understood** why a network gives a particular output

Scalability

- DNNs are very **large**, **highly interconnected structures**; often have **huge input spaces**

  $\implies$ prevent thorough verification/testing

Challenges

Lack of Robustness

- **Small** changes to an input may lead to **unexpected behaviours**

Lack of Explainability

- It is **not well understood** why a network gives a particular output

Scalability

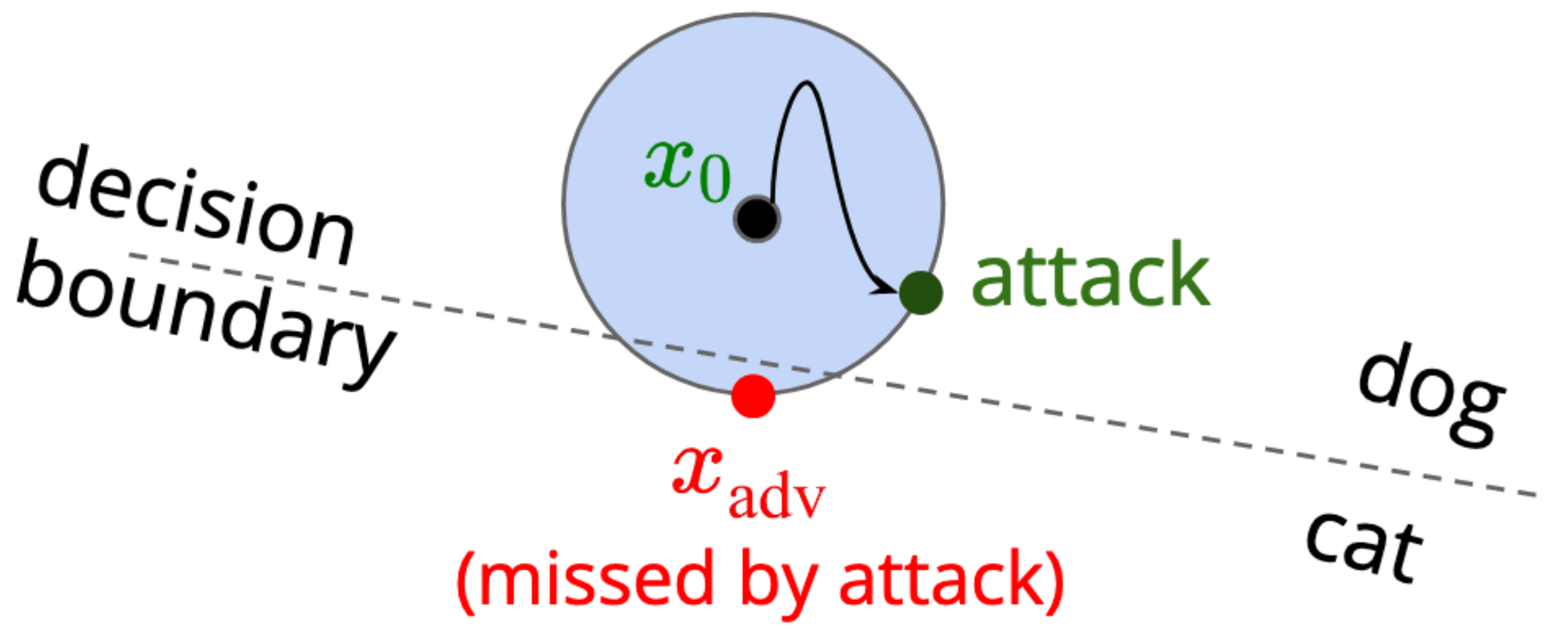
- DNNs are very **large**, **highly interconnected structures**; often have **huge input spaces**

  $\Longrightarrow$ prevent thorough verification/testing

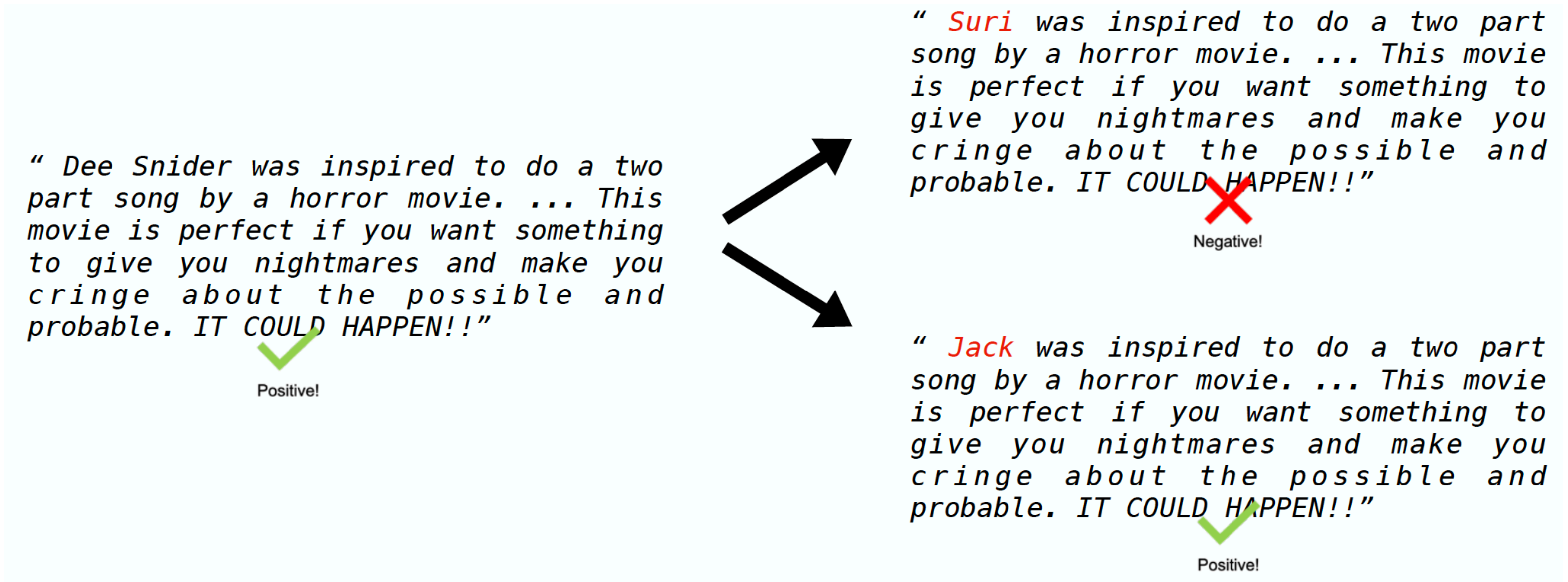
Can we provide insights about the behaviors of DNN?

Analysis of Deep Neural Networks

DNN Verification



Robustness Testing (Adversarial Attack)



Fairness Testing

Property Inference
for Deep Neural Network

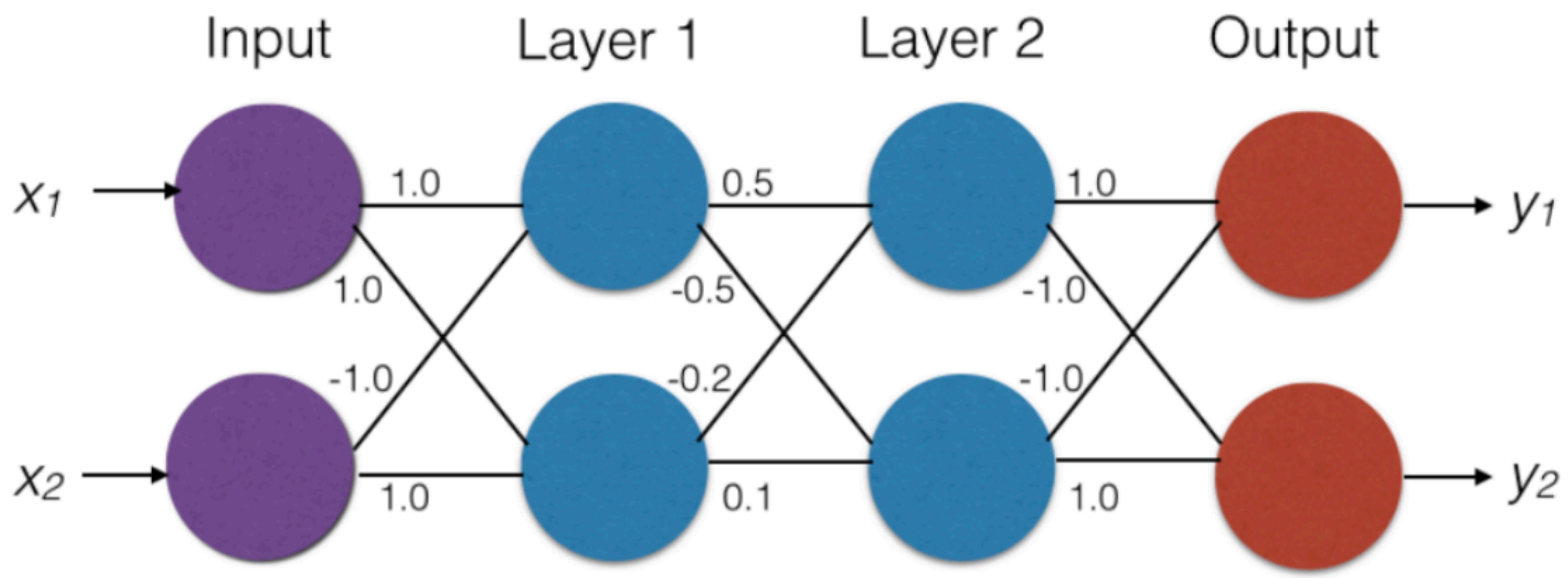
Property Inference for Deep Neural Networks

Gopinath, D., Converse, H., Pasareanu, C. and Taly, A.

IEEE/ACM International Conference on Automated Software Engineering 2019

Key ideas

- Infer properties, a.k.a explanations, of a DNN
- Properties can be **proved to be valid (formal guarantee)** on the network using a decision procedure



$$(x_1 \geq 3) \wedge (x_2 \leq 1) \Rightarrow y_1 > y_2$$

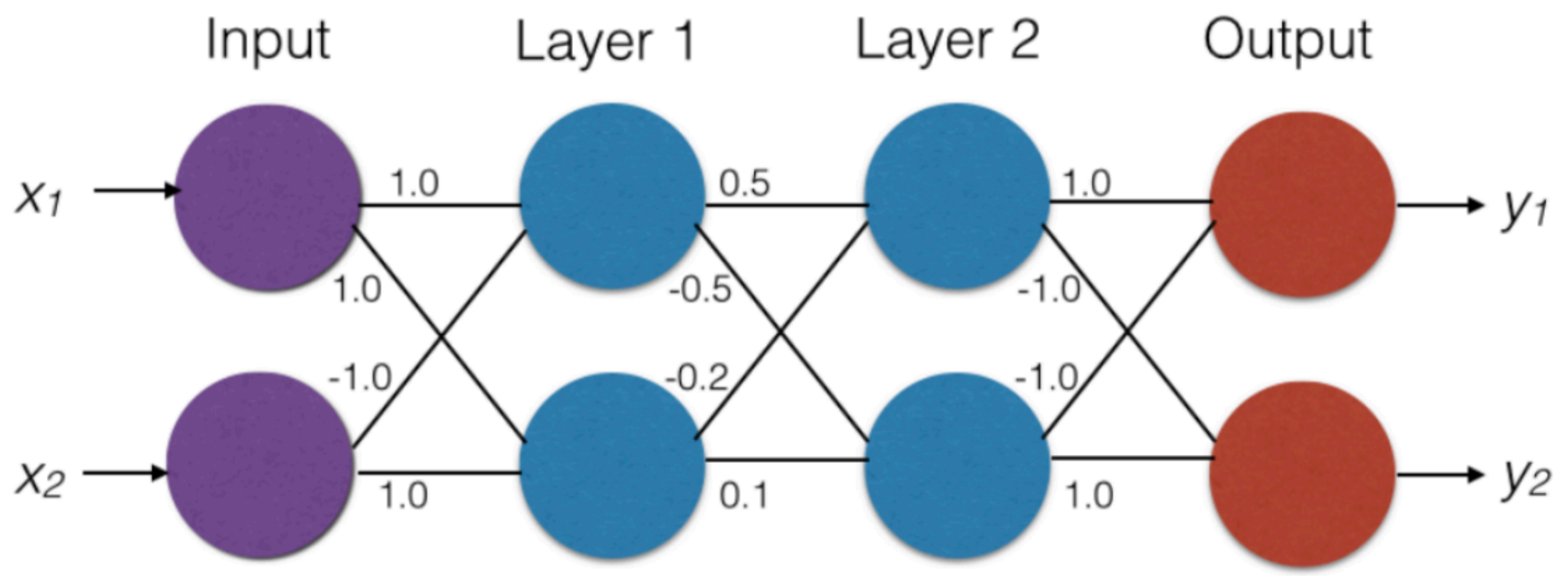
Prophecy: Property Inference for Deep Neural Networks

Gopinath, D., Converse, H., Pasareanu, C. and Taly, A.

IEEE/ACM International Conference on Automated Software Engineering 2019

Key ideas

- Infer properties, a.k.a explanations, of a DNN
- Properties can be **proved to be valid (formal guarantee)** on the network using a decision procedure, i.e., verification



Decompose the **black-box** DNN into **a set of rules** aids in interpreting and understanding the behavior of DNNs

$$(x_1 \geq 3) \land (x_2 \leq 1) \Rightarrow y_1 > y_2$$
$$(x_1 \geq 1) \land (x_2 \geq 6) \Rightarrow y_2 > y_1$$
$$(x_1 \geq 5) \land (x_2 \geq 4) \Rightarrow y_2 > y_1$$

Prophecy: Property Inference for Deep Neural Networks

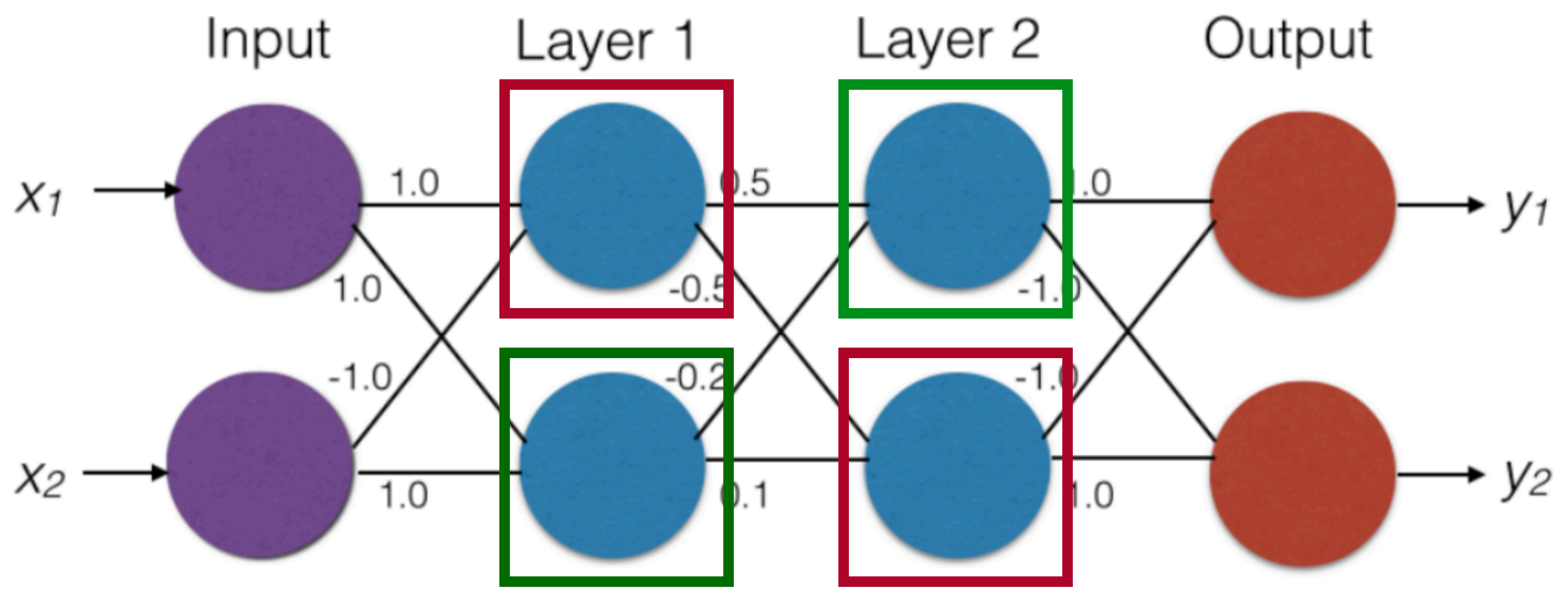
Gopinath, D., Converse, H., Pasareanu, C. and Taly, A.

IEEE/ACM International Conference on Automated Software Engineering 2019

Formalizing properties

- A constraints in terms of the on/off activation pattern of neurons of the neural network
  - ReLU: f(x) = max(0,x)
  - ReLU(x) is on if x>0 and off if x <= 0
- Intuition: Piecewise linear nodes equivalent to conditional statements of traditional programs

  $\Rightarrow$ logic of network can be capture in the activation patterns of neurons



$$off(N_{1,1}) \wedge on(N_{1,2}) \wedge on(N_{2,1}) \wedge off(N_{2,2}) \Rightarrow y_1 > y_2$$

Prophecy: Property Inference for Deep Neural Networks

Gopinath, D., Converse, H., Pasareanu, C. and Taly, A.

IEEE/ACM International Conference on Automated Software Engineering 2019

Properties: Pre $\Rightarrow$ Post

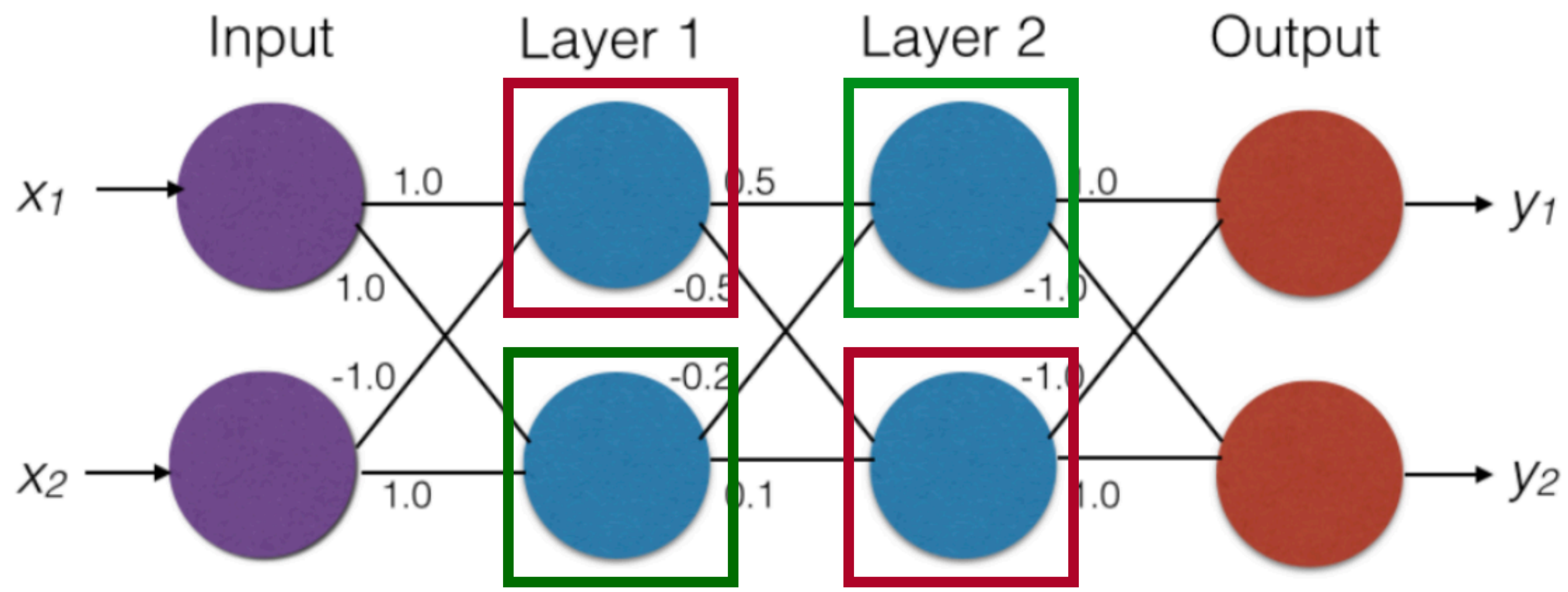
- Pre is a conjunction of constraints on **neurons**
- Post is a certain output property
- Pre is actually convex region in input space

Theorem: For all $\prec$-closed patterns σ, $\sigma(X)$ is **convex**, and has the form:

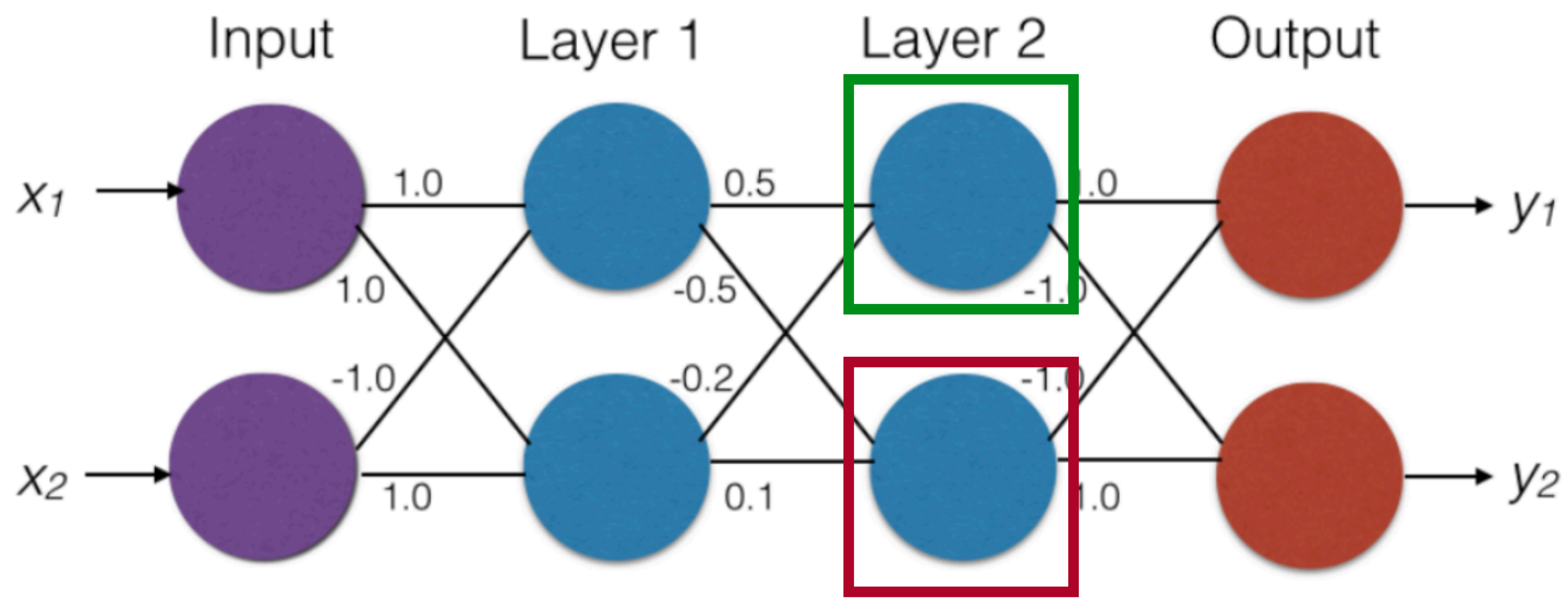
$$\bigwedge_{i \text{ in } 1..|on(\sigma)|} W_i \cdot X + b_i > 0 \wedge \bigwedge_{j \text{ in } 1..|off(\sigma)|} W_j \cdot X + b_j \leq 0$$

W_i, b_i, W_j, b_j are constants derived from the weight and bias parameters of the network.

Input Properties



Layer Properties



Property Inference for Deep Neural Networks

Gopinath, D., Converse, H., Pasareanu, C. and Taly, A.

IEEE/ACM International Conference on Automated Software Engineering 2019

Property Inference:

- Concolic Execution and Iterative Relaxation of path constraints;
- Decision Tree over on/off activation patterns on training dataset; verify patterns with decision procedure, i.e. Reluplex

Algorithm 1 Iterative relaxation algorithm to extract input properties from input X.

// Let k be the layer before output layer
// We write $\mathcal{N}^l$ for the neurons at layer l
$\sigma = \sigma_X$ // Activation signature of input X
$sat = \textbf{DP}(\sigma(X), P(F(X)))$
if sat **then return** $\sigma(X) \wedge P(F(X))$
$l = k$
while $l > 1$ **do**
    $\sigma = \sigma \setminus \mathcal{N}^l$
    $sat = \textbf{DP}(\sigma(X), P(F(X))$
    if sat **then**
        // Critical layer found
        $cl = l$
        // Add back activations from critical layer
        $\sigma = \sigma \cup \mathcal{N}^{cl}$
        for each $N \in \mathcal{N}^{cl}$ **do**
            $\sigma' = \sigma \setminus \{N\}$
            $sat = \textbf{DP}(\sigma'(X), P(F(X))$
            if $\neg sat$ **then**
                // Neuron N can remain unconstrained
                $\sigma = \sigma'$
        return $\sigma(X)$
    else
        $l = l - 1$

Prophecy: Property Inference for Deep Neural Networks

Gopinath, D., Converse, H., Pasareanu, C. and Taly, A.

IEEE/ACM International Conference on Automated Software Engineering 2019

Property Inference:

- Concolic Execution and Iterative Relaxation of path constraints;
- Decision Tree over on/off activation patterns on training dataset; verify patterns with decision procedure, i.e. Reluplex

Check if an activation pattern σ implies a certain output property $P(F(x))$ (*)

Algorithm 1 Iterative relaxation algorithm to extract input properties from input X.

// Let k be the layer before output layer
// We write $\mathcal{N}^l$ for the neurons at layer l
$\sigma = \sigma_X$ // Activation signature of input X
$sat = \mathbf{DP}(\sigma(X), P(F(X)))$
if sat **then return** $\sigma(X) \wedge P(F(X))$
$l = k$
while $l > 1$ **do**
    $\sigma = \sigma \setminus \mathcal{N}^l$
    $sat = \mathbf{DP}(\sigma(X), P(F(X))$
    if sat **then**
        // Critical layer found
        $cl = l$
        // Add back activations from critical layer
        $\sigma = \sigma \cup \mathcal{N}^{cl}$
        for each $N \in \mathcal{N}^{cl}$ **do**
            $\sigma' = \sigma \setminus \{N\}$
            $sat = \mathbf{DP}(\sigma'(X), P(F(X))$
            if $\neg sat$ **then**
                // Neuron N can remain unconstrained
                $\sigma = \sigma'$
        return $\sigma(X)$
    else
        $l = l - 1$

Prophecy: Property Inference for Deep Neural Networks

Gopinath, D., Converse, H., Pasareanu, C. and Taly, A.

IEEE/ACM International Conference on Automated Software Engineering 2019

Property Inference:

- Concolic Execution and Iterative Relaxation of path constraints;
- Decision Tree over on/off activation patterns on training dataset; verify patterns with decision procedure, i.e. Reluplex

Find minimal activation patterns such that (*) holds

Algorithm 1 Iterative relaxation algorithm to extract input properties from input X.

// Let k be the layer before output layer
// We write $\mathcal{N}^l$ for the neurons at layer l
$\sigma = \sigma_X$ // Activation signature of input X
$sat = \mathbf{DP}(\sigma(X), P(F(X)))$
if sat **then return** $\sigma(X) \wedge P(F(X))$
$l = k$
while $l > 1$ **do**
  $\sigma = \sigma \setminus \mathcal{N}^l$
  $sat = \mathbf{DP}(\sigma(X), P(F(X))$
  if sat **then**
    // Critical layer found
    $cl = l$
    // Add back activations from critical layer
    $\sigma = \sigma \cup \mathcal{N}^{cl}$
    for each $N \in \mathcal{N}^{cl}$ **do**
      $\sigma' = \sigma \setminus \{N\}$
      $sat = \mathbf{DP}(\sigma'(X), P(F(X))$
      if $\neg sat$ **then**
        // Neuron N can remain unconstrained
        $\sigma = \sigma'$
    return $\sigma(X)$
  else
    $l = l - 1$

Prophecy: Property Inference for Deep Neural Networks

Gopinath, D., Converse, H., Pasareanu, C. and Taly, A.

IEEE/ACM International Conference on Automated Software Engineering 2019

Property Inference:

- Concolic Execution and Iterative Relaxation of path constraints;
- Decision Tree over on/off activation patterns on training dataset; verify patterns with decision procedure, i.e. Reluplex

Algorithm 1 Iterative relaxation algorithm to extract input properties from input X.

```
// Let k be the layer before output layer
// We write N^l for the neurons at layer l
σ = σ_X // Activation signature of input X
sat = DP(σ(X), P(F(X)))
if sat then return σ(X) ∧ P(F(X))
l = k
while l > 1 do
    σ = σ \ N^l
    sat = DP(σ(X), P(F(X))
    if sat then
        // Critical layer found
        cl = l
        // Add back activations from critical layer
        σ = σ ∪ N^cl
        for each N ∈ N^cl do
            σ' = σ \ {N}
            sat = DP(σ'(X), P(F(X))
            if ¬sat then
                // Neuron N can remain unconstrained
                σ = σ'
        return σ(X)
    else
        l = l − 1
```

Remove each layer until (*) does not hold

Property Inference for Deep Neural Networks

Gopinath, D., Converse, H., Pasareanu, C. and Taly, A.

IEEE/ACM International Conference on Automated Software Engineering 2019

Property Inference:

- Concolic Execution and Iterative Relaxation of path constraints;
- Decision Tree over on/off activation patterns on training dataset; verify patterns with decision procedure, i.e. Reluplex

Algorithm 1 Iterative relaxation algorithm to extract input properties from input X.

// Let k be the layer before output layer
// We write $\mathcal{N}^l$ for the neurons at layer l
$\sigma = \sigma_X$ // Activation signature of input X
$sat = \mathbf{DP}(\sigma(X), P(F(X)))$
if sat **then return** $\sigma(X) \wedge P(F(X))$
$l = k$
while $l > 1$ **do**
    $\sigma = \sigma \setminus \mathcal{N}^l$
    $sat = \mathbf{DP}(\sigma(X), P(F(X))$
    if sat **then**
        // Critical layer found
        $cl = l$
        // Add back activations from critical layer
        $\sigma = \sigma \cup \mathcal{N}^{cl}$
        for each $N \in \mathcal{N}^{cl}$ **do**
            $\sigma' = \sigma \setminus \{N\}$
            $sat = \mathbf{DP}(\sigma'(X), P(F(X))$
            if $\neg sat$ **then**
                // Neuron N can remain unconstrained
                $\sigma = \sigma'$
        return $\sigma(X)$
    else
        $l = l - 1$

Remove each neurons of critical layer until (*) does not hold

Prophecy: Property Inference for Deep Neural Networks

Gopinath, D., Converse, H., Pasareanu, C. and Taly, A.

IEEE/ACM International Conference on Automated Software Engineering 2019

Property Inference:

- Concolic Execution and Iterative Relaxation of path constraints;
- Decision Tree over on/off activation patterns on training dataset; verify patterns with decision procedure, i.e. Reluplex

Algorithm 1 Iterative relaxation algorithm to extract input properties from input X.

// Let k be the layer before output layer
// We write $\mathcal{N}^l$ for the neurons at layer l
$\sigma = \sigma_X$ // Activation signature of input X
$sat = \mathbf{DP}(\sigma(X), P(F(X)))$
if sat **then return** $\sigma(X) \wedge P(F(X))$
$l = k$
while $l > 1$ **do**
    $\sigma = \sigma \setminus \mathcal{N}^l$
    $sat = \mathbf{DP}(\sigma(X), P(F(X))$
    if sat **then**
        // Critical layer found
        $cl = l$
        // Add back activations from critical layer
        $\sigma = \sigma \cup \mathcal{N}^{cl}$
        for each $N \in \mathcal{N}^{cl}$ **do**
            $\sigma' = \sigma \setminus \{N\}$
            $sat = \mathbf{DP}(\sigma'(X), P(F(X))$
            if $\neg sat$ **then**
                // Neuron N can remain unconstrained
                $\sigma = \sigma'$
        return $\sigma(X)$
    else
        $l = l - 1$

Remaining neurons are a minimal activation pattern

Scalability

Algorithm 1 Iterative relaxation algorithm to extract input properties from input X.

// Let k be the layer before output layer
// We write $\mathcal{N}^l$ for the neurons at layer l
$\sigma = \sigma_X$ // Activation signature of input X
$sat = \mathbf{DP}(\sigma(X), P(F(X)))$
if sat **then return** $\sigma(X) \wedge P(F(X))$
$l = k$
while $l > 1$ **do**
    $\boxed{\sigma = \sigma \setminus \mathcal{N}^l}$
    $sat = \mathbf{DP}(\sigma(X), P(F(X)))$
    if sat **then**
        // Critical layer found
        $cl = l$
        // Add back activations from critical layer
        $\sigma = \sigma \cup \mathcal{N}^{cl}$
        for each $N \in \mathcal{N}^{cl}$ **do**
            $\boxed{\sigma' = \sigma \setminus \{N\}}$
            $\boxed{sat = \mathbf{DP}(\sigma'(X), P(F(X)))}$
            if $\neg sat$ **then**
                // Neuron N can remain unconstrained
                $\sigma = \sigma'$
        return $\sigma(X)$
    else
        $l = l - 1$

Depends on the number of layers & neurons
But, the number of neurons can be very large

Decision Procedure/ Verification is time-consuming for big DNNs

Property Inference for Deep Neural Networks

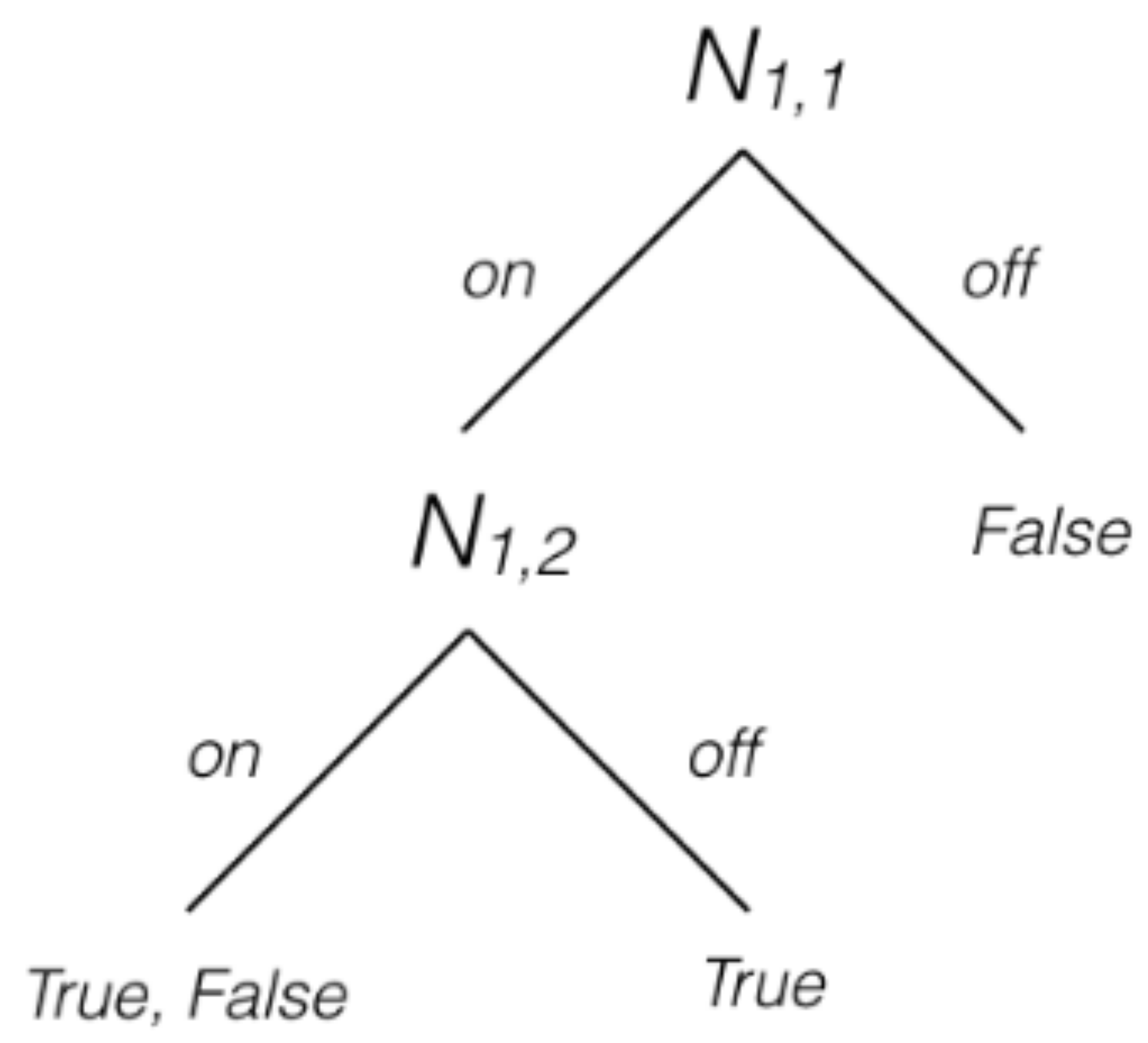
Gopinath, D., Converse, H., Pasareanu, C. and Taly, A.

IEEE/ACM International Conference on Automated Software Engineering 2019

Property Inference:

- Concolic Execution and Iterative Relaxation of path constraints; under-approx. boxes with LP solving
- Decision Tree over on/off activation patterns on training dataset; verify patterns with decision procedure, i.e. Reluplex

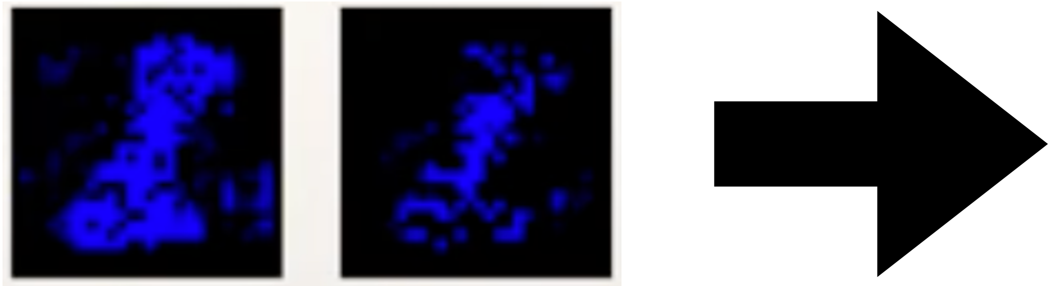
$\langle x_1, x_2 \rangle$	$\langle N_{1,1}, N_{1,2} \rangle$	$P(F(X))$
$\langle 0, -1 \rangle$	$\langle on, off \rangle$	True
$\langle 1, 0 \rangle$	$\langle on, on \rangle$	True
$\langle 0, 1 \rangle$	$\langle off, on \rangle$	False
$\langle 4, 3 \rangle$	$\langle on, on \rangle$	False
$\langle 1, -1 \rangle$	$\langle on, off \rangle$	True



Applications

- Generating explanations with formal guarantee
- Finding adversarial examples
- Build simple models (distillation)
- Decompose hard proofs, i.e. verification

Results for MNIST



Prediction = 1

P = 1 P = 2 P = 2

Results for ACASXU

- Discover **novel properties** validated by domain experts

Property 1: All inputs within the following region, $36000 \leq range \leq 60760$, $0.7 \leq \theta \leq 3.14$, $-3.14 \leq \psi \leq -3.14 + 0.01$, $900 \leq v_{own} \leq 1200$, $600 \leq v_{int} \leq 1200$, should have the turning advisory as COC. This property takes approx. 31 minutes to check with Reluplex.

Property 2: All the inputs within the following region: $12000 \leq range \leq 62000$, $(0.7 \leq \theta \leq 3.14)$ or $(-3.14 \leq \theta \leq -0.7)$, $-3.14 \leq \psi \leq -3.14 + 0.005$, $100 \leq v_{own} \leq 1200$, $0 \leq v_{int} \leq 1200$, should have the turning advisory as COC. This property has a large input region and direct verification with Reluplex times out after 12 hours.

Property 3: All the inputs within the following region: $range > 55947.691$, $-3.14 \leq \theta \leq 3.14$, $-3.14 \leq \psi \leq 3.14$, $1145 \leq v_{own} \leq 1200$, $0 \leq v_{int} \leq 60$, should have the turning advisory as Clear-of-Conflict (COC). This property takes approx. 5 hours to check with Reluplex.

Follow Up Work

Follow-up Works

- Abduction-Based Explanations for Machine Learning Models, AAAI 2019
- Property Inference in ReLU nets using Linear Interpolants, VNN 2020
- Programmatic and Semantic Approach to Explaining and Debugging Neural Network Based Object Detectors, CVPR 2020
- Scaling Symbolic Methods using Gradients for Neural Model Explanation, ICLR 2021
- Towards the Analysis of Graph Neural Network, ICSE 2022

Applications

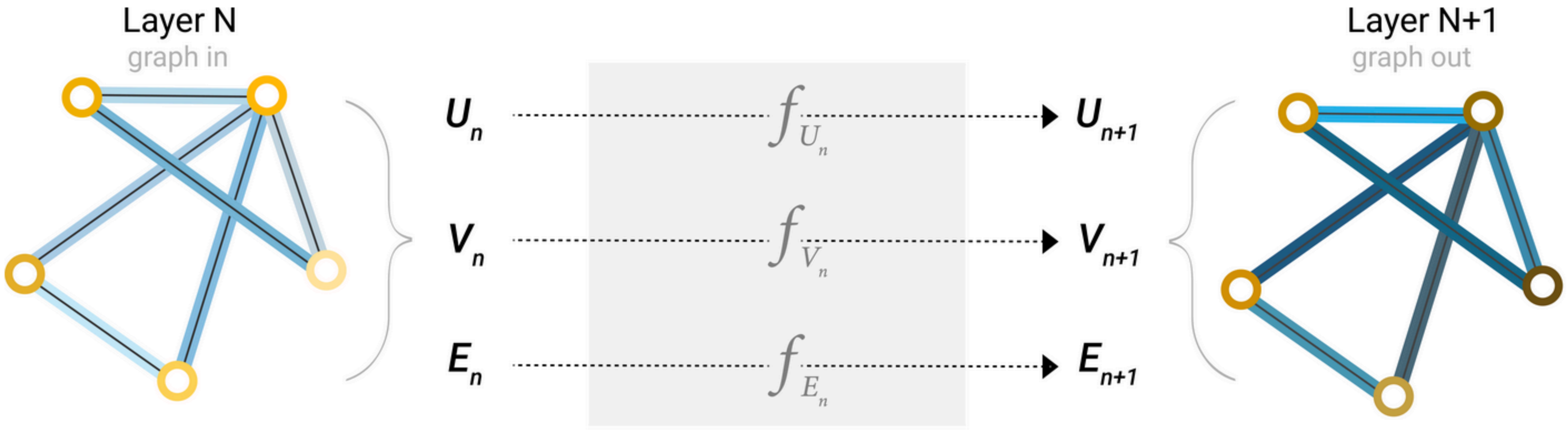
- NNrepair: Constraint-Based Repair ofNeural Network Classifiers, CAV 2021
- Provably Robust Adversarial Examples, ICLR 2022

GNNInfer: Towards the Analysis of Graph Neural Networks

Towards the Analysis of Graph Neural Networks

T. -D. Nguyen, T. Le-Cong, T. H. Nguyen, X. -B. D. Le and Q. -T. Huynh

IEEE/ACM International Conference on Software Engineering 2022



$$m_{j\to i} = f_{\mathrm{msg}}(x_i, x_j, e_{ji})$$
$$m_i = f_{\mathrm{agg}}(\{m_{j\to i} | \forall j \in \mathcal{N}(i)\})$$
$$x_i = f_{\mathrm{upd}}(x_i, m_i)$$

Graph Neural Networks

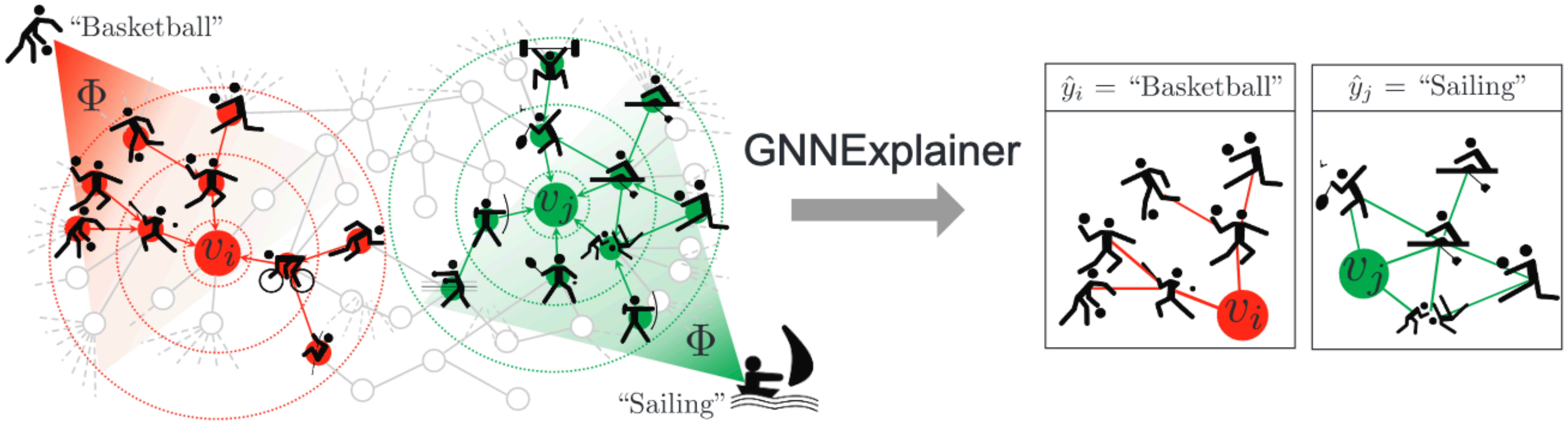
Dynamic Structure

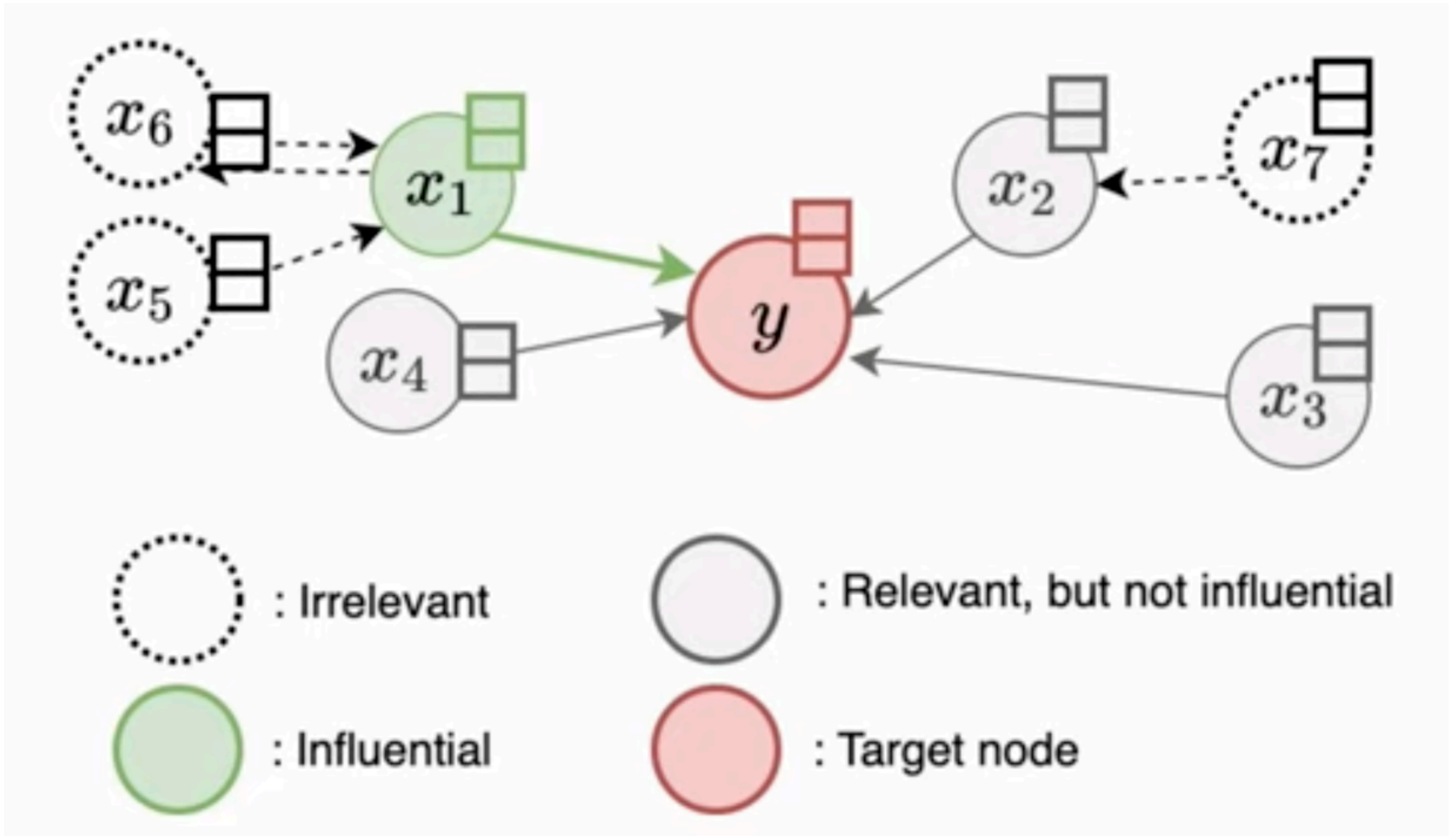
The structure of GNN depends on graph inputs $\Rightarrow$ **dynamic** network structure

Towards the Analysis of Graph Neural Networks

T. -D. Nguyen, T. Le-Cong, T. H. Nguyen, X. -B. D. Le and Q. -T. Huynh

IEEE/ACM International Conference on Software Engineering 2022





Influential Substructures

Previous works[1,2] hinted there exist **influential substructures** that **significantly contribute** to the trained GNN's prediction

[1] GNNExplainer: Generating Explanations for Graph Neural Networks, Ying et al., NIPS 2021

[2] PGMExplainer: Probabilistic graphical model explanations for graph neural networks, Vu et al., NIPS 2020

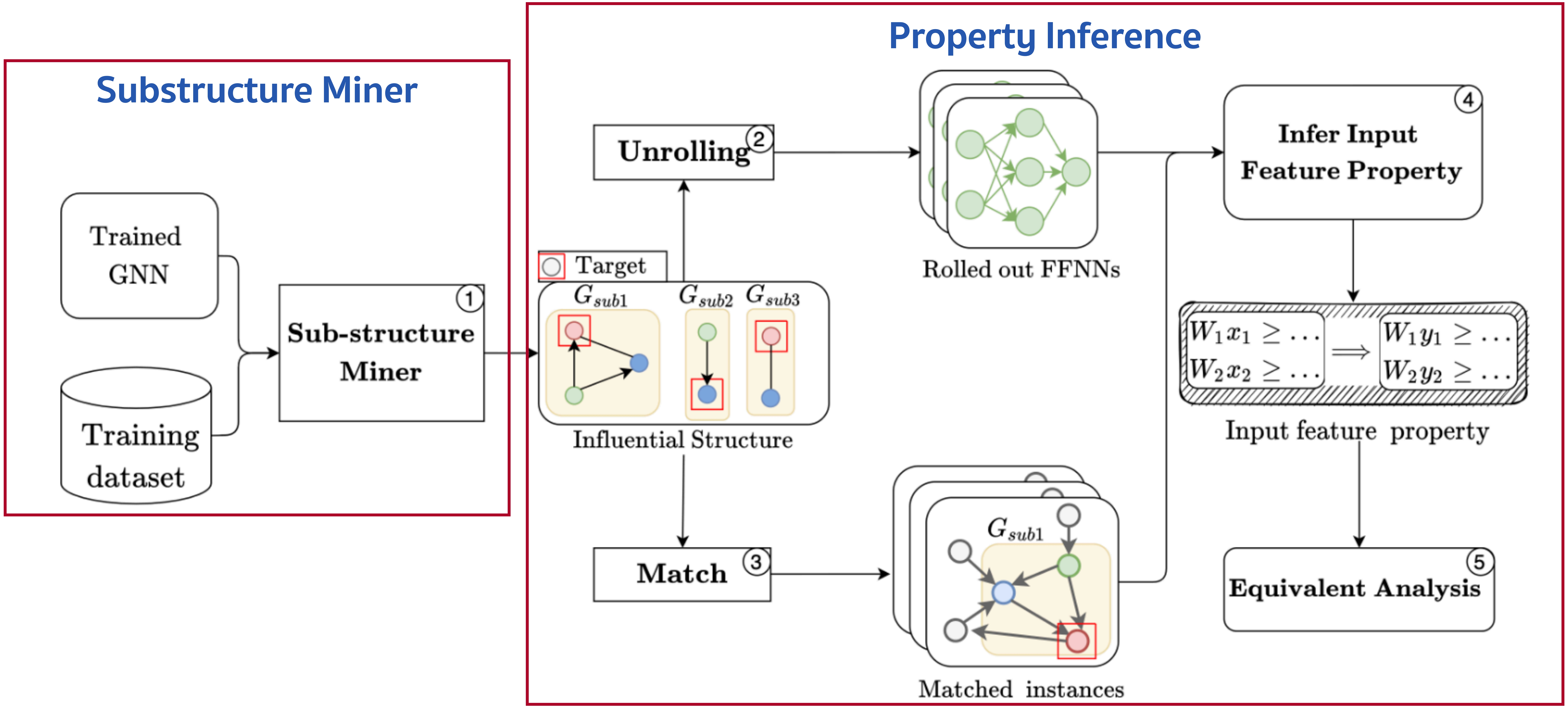
Dynamic Structure

- **Fixed sizes**: convertible to FFNNs for analysis
- **Significant contribution**: less works for analysis to be equivalent to full GNN computation

Towards the Analysis of Graph Neural Networks

T. -D. Nguyen, T. Le-Cong, T. H. Nguyen, X. -B. D. Le and Q. -T. Huynh

IEEE/ACM International Conference on Software Engineering 2022

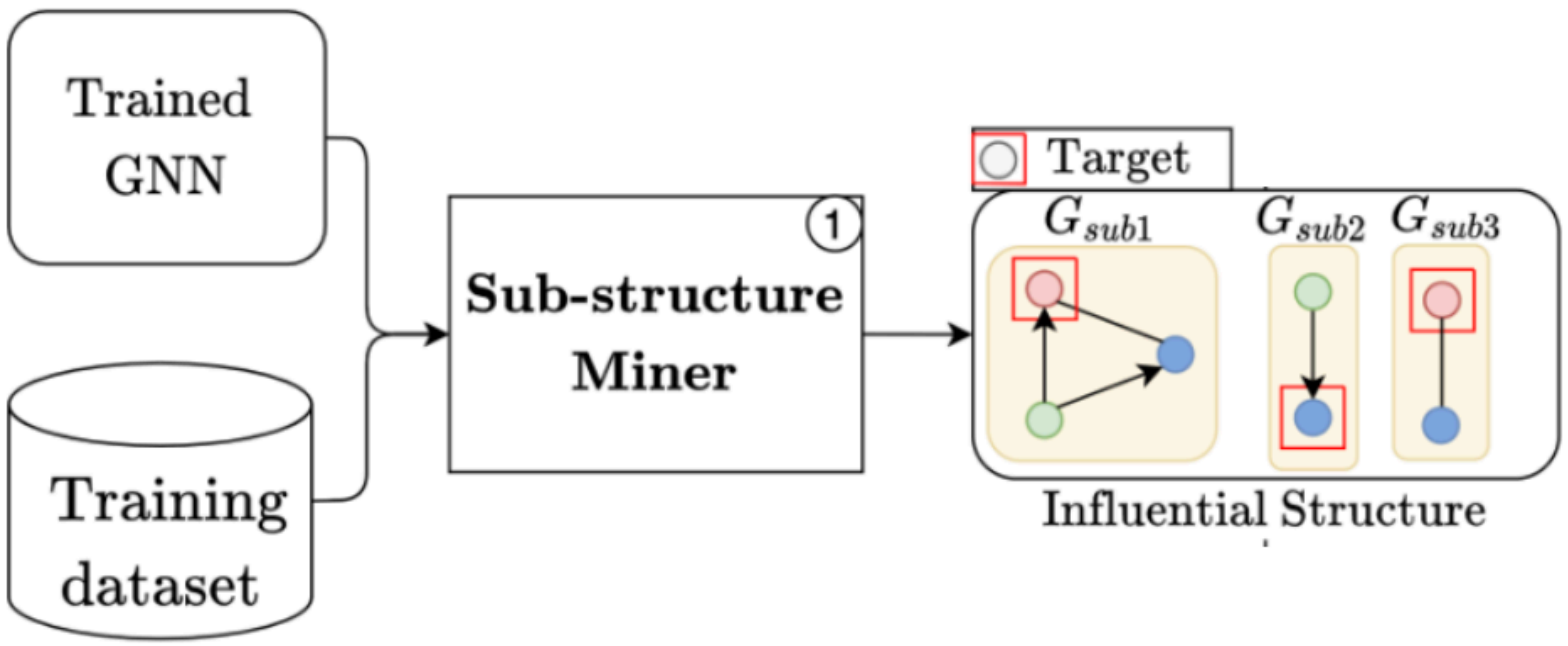


GNN-Infer

Towards the Analysis of Graph Neural Networks

T. -D. Nguyen, T. Le-Cong, T. H. Nguyen, X. -B. D. Le and Q. -T. Huynh

IEEE/ACM International Conference on Software Engineering 2022



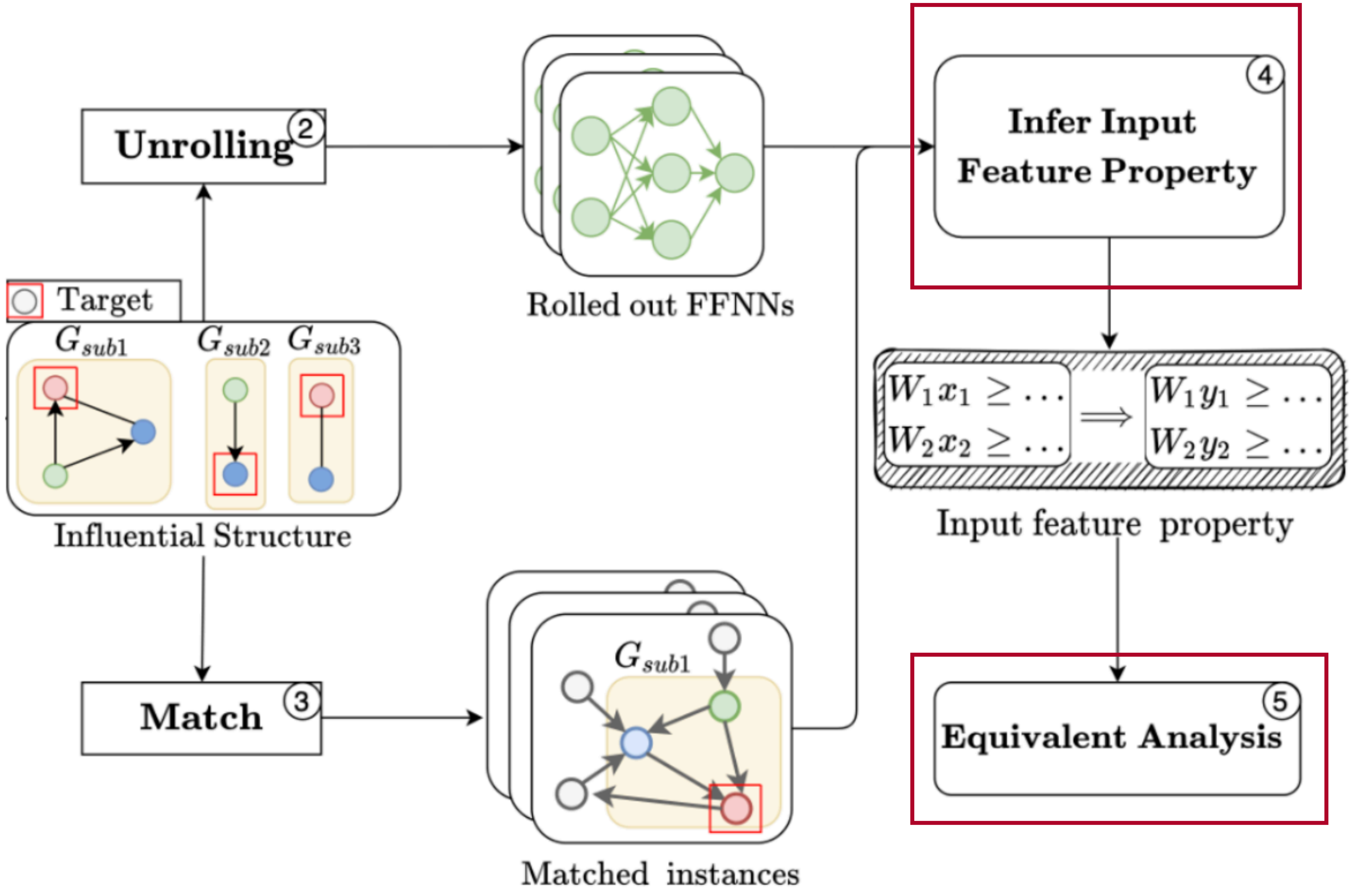
Substructure Miner

- **Influential Substructure Detection:** GNN-Infer employs GNNExplainer to detect **local influential substructures** for each instances in training data
- **Influential Substructure Miner**: GNN-Infer employs Subdue to mine **(frequent)** subgraph from local influential substructures as **(global) influential substructures**

Towards the Analysis of Graph Neural Networks

T. -D. Nguyen, T. Le-Cong, T. H. Nguyen, X. -B. D. Le and Q. -T. Huynh

IEEE/ACM International Conference on Software Engineering 2022



Feature Property Inference

- **Unroll:** GNN-Infer unroll GNN over influential substructures to equivalent FFNNs
- **Match**: GNN-Infer match influential substructures with training instances to obtain inputs for unrolled FFNNs
- **Inference:** GNN-Infer infer properties of unrolled FFNNs using existing DNN analyses, i.e., Prophecy & Marabou
- **Equivalent Analysis:** GNN-Infer employs decision tree to find condition ensuring that feature properties holds on GNN over full graphs

Towards the Analysis of Graph Neural Networks

T. -D. Nguyen, T. Le-Cong, T. H. Nguyen, X. -B. D. Le and Q. -T. Huynh

IEEE/ACM International Conference on Software Engineering 2022

Evaluation

- **Benchmark:** Neural Execution of Graph Algorithm: BFS, DFS, Bellman-Ford, …
- **GNN-Handcraft:** A GNN is manually constructed ==> Correct
- **Criteria:** Check the correctness of inferred properties

Sample: BFS

- **Property 1:**
  $\exists x \in N, (x, t) \in E \wedge v(x) = 1 \Rightarrow v(t) = 1$
- **Property 2:**
  $\forall x \in N, (x, t) \in E \wedge v(x) = 0 \Rightarrow v(t) = 0$
- **Property 3:** $\forall t, v(t) = 1 \Rightarrow v(t) = 1$

Thank You!

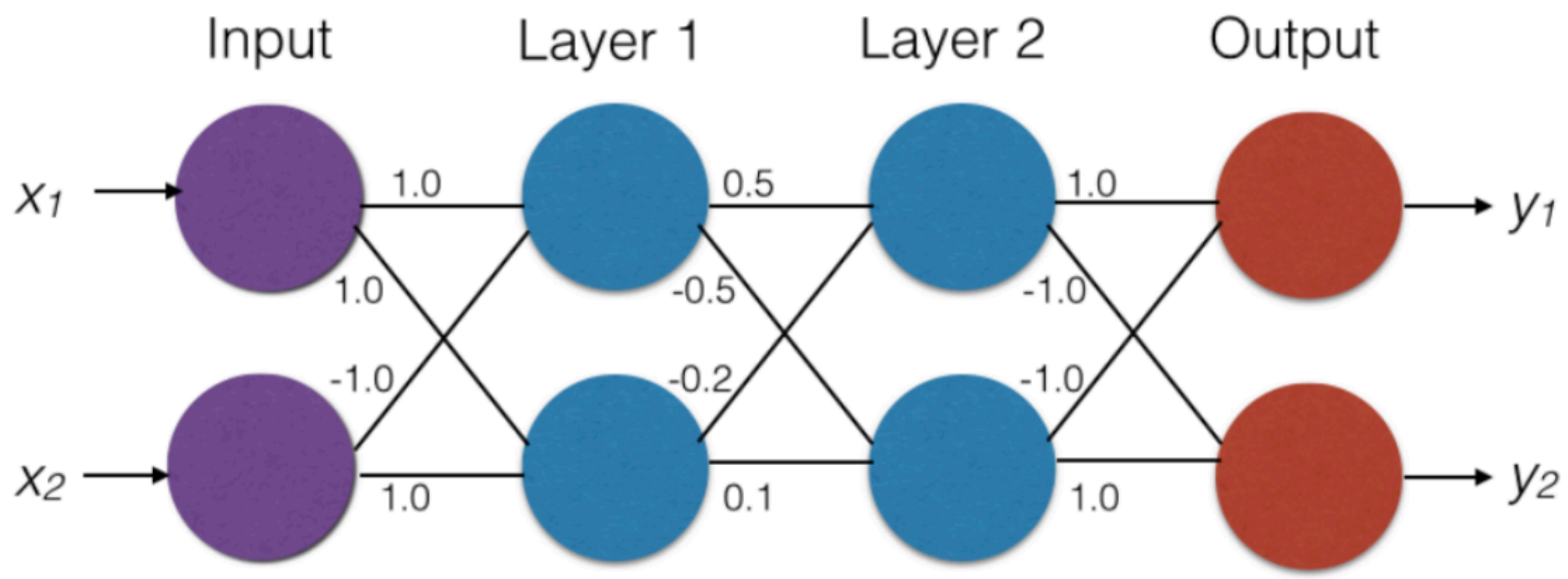
Challenges and Opportunities

Can activation patterns are enough?

Formalizing properties

- A constraints in terms of the on/off activation pattern of neurons of the neural network
  - ReLU: f(x) = max(0,x)
  - ReLU(x) is on if x>0 and off if x <= 0
- Intuition: Piecewise linear nodes equivalent to conditional statements of traditional programs

  ⇒ logic of network can be capture in the activation patterns of neurons



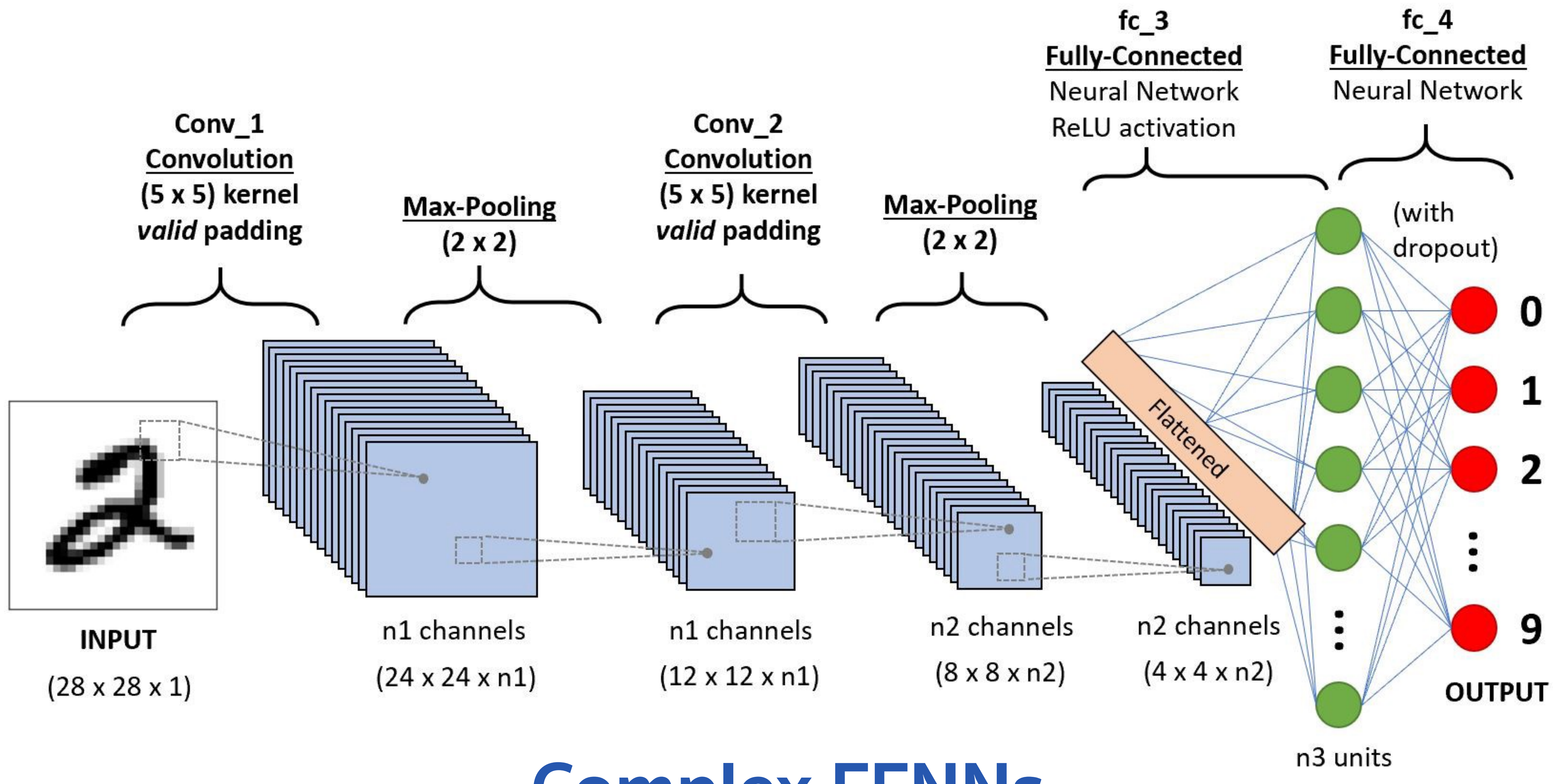
$$((x_0 \leq 3 \wedge x_1 \geq 4) \Rightarrow y = 0)$$
$$\wedge ((x_0 - x_1 \geq 0) \Rightarrow y = 1)$$
$$\wedge ((7x_0 - 6x_1 \geq 0) \Rightarrow y = 1)$$

The intuition is reasonable but ...

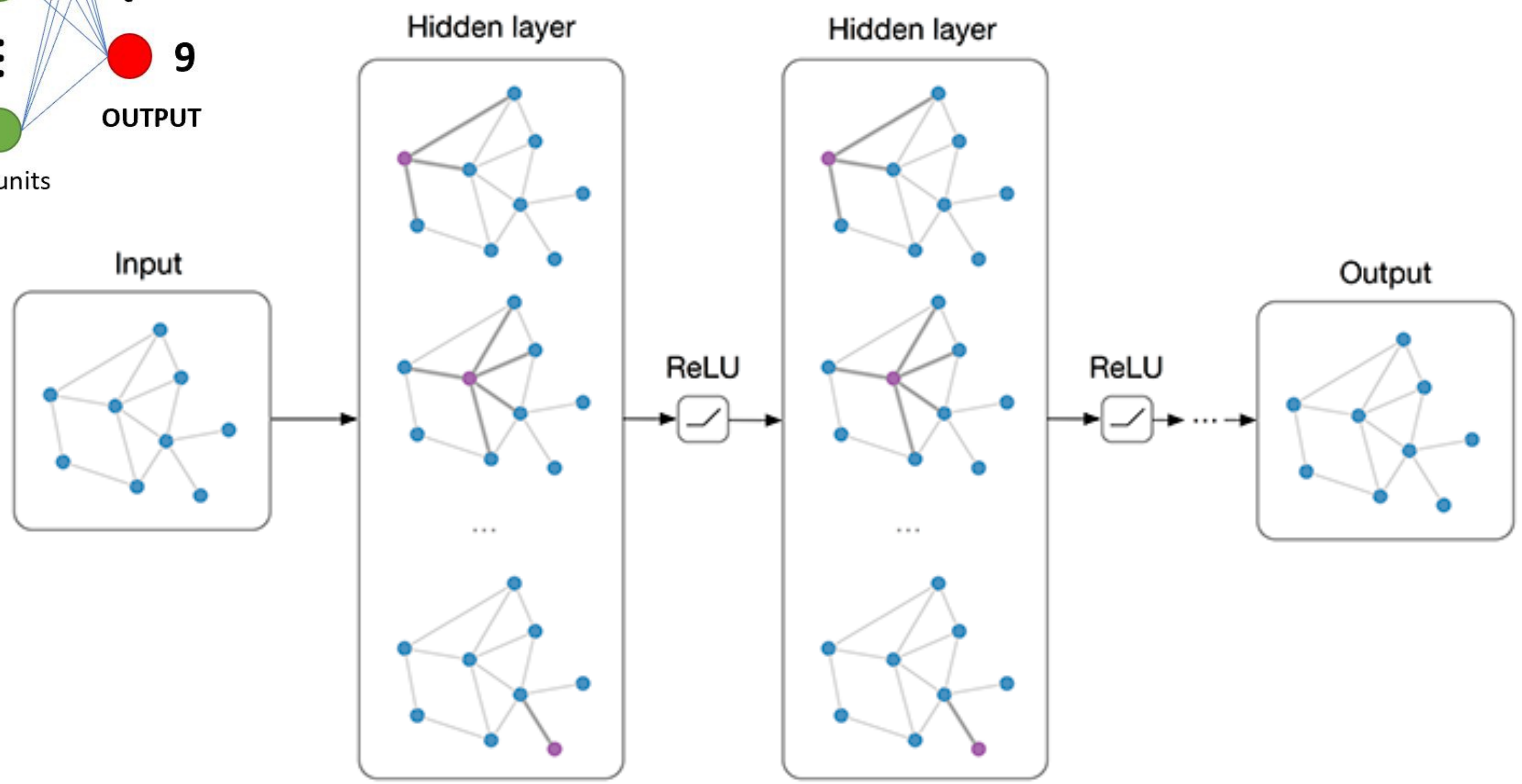
"Is it enough ?"

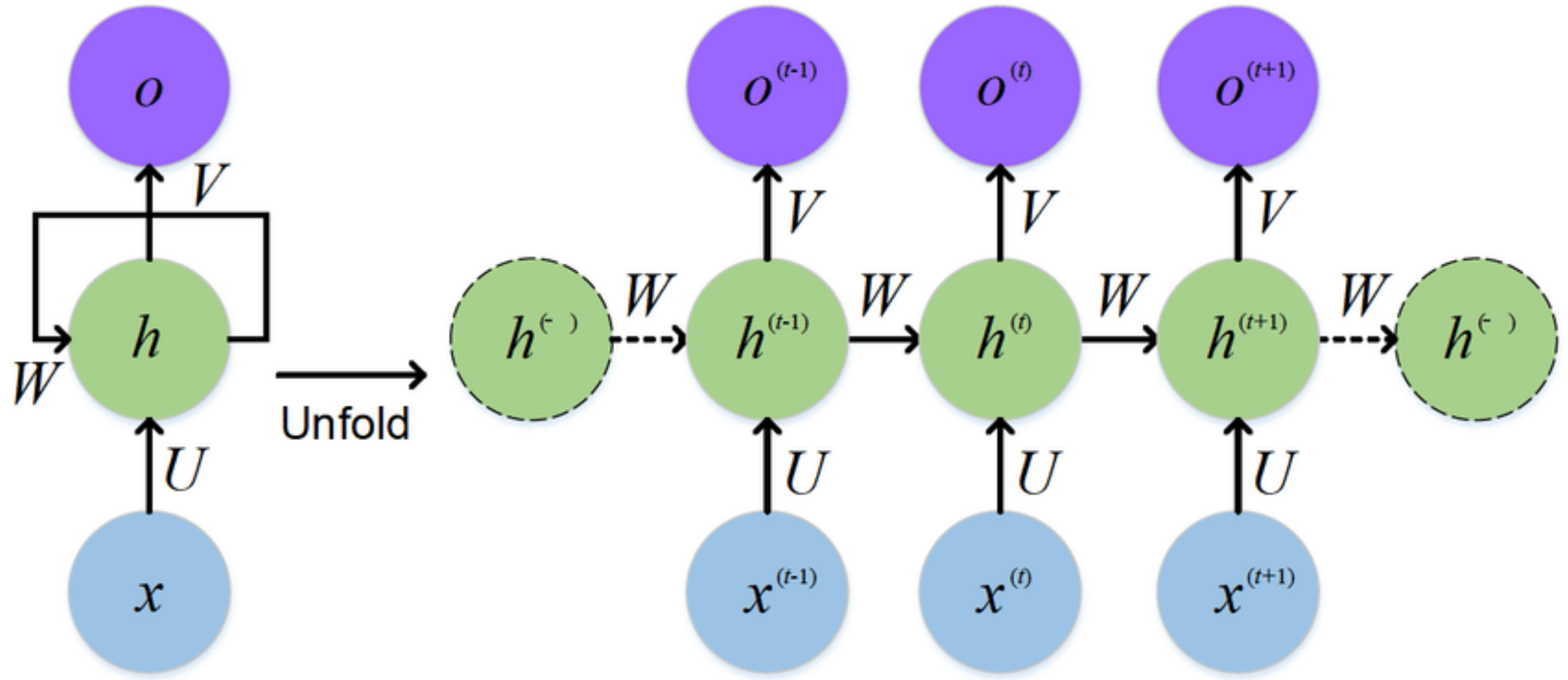
"Are there some logic rules that satisfies different activation patterns ? "

Complex Neural Networks ...



Complex FFNNs





Recurrent Neural Network

Graph Neural Network