Understanding and Verifying Neural Networks

Divya Gopinath

Research Scientist, RSE group NASA AMES, KBR Inc.

SafeDNN: Safety of Deep Neural Networks

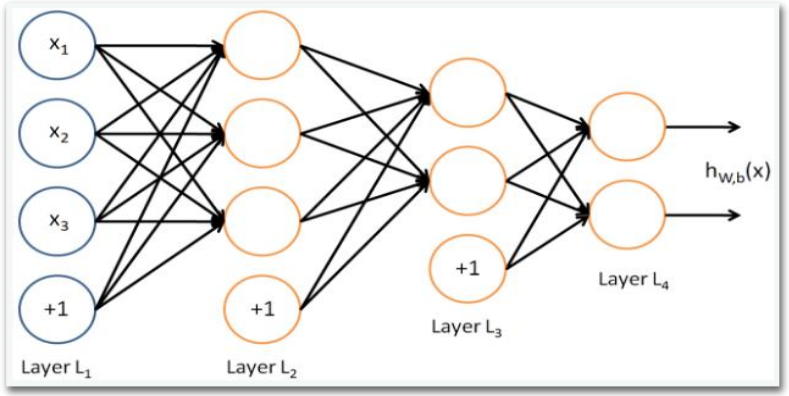
https://ti.arc.nasa.gov/tech/rse/research/safednn/

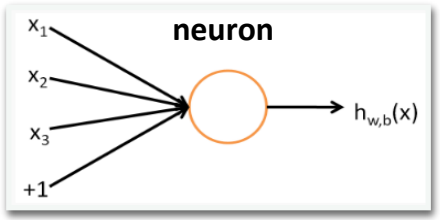
- NASA project that explores techniques and tools to ensure that systems that use Deep Neural Networks (DNN) are safe, robust and interpretable
- Project Members: Corina Pasareanu, Divya Gopinath
  - Many students and collaborators
- This talk focuses on *Prophecy* that infers properties from neural network models, which could be used to
  - Provide *formal guarantees wrt safety and robustness for DNNs*
  - Obtain *compact, formal explanations* of DNN behavior
  - *Debugging and improving testing* of DNNs

- *Divya Gopinath, Hayes Converse, Corina S. Pasareanu, Ankur Taly:* ***Property Inference for Deep Neural Networks. ASE 2019***
- *Edward Kim, Divya Gopinath, Corina S. Pasareanu, Sanjit A. Seshia:* ***A Programmatic and Semantic Approach to Explaining and Debugging Neural Network Based Object Detectors. CVPR 2020***
- *Muhammad Usman, Divya Gopinath, Youcheng Sun, Yannic Noller, Corina S. Pasareanu:* ***NNrepair: Constraint-Based Repair of Neural Network Classifiers. CAV 2021***
- *Burak Kadron, Divya Gopinath, Corina Pasareanu, Huafeng Yu:* ***Case Study: Analysis of Autonomous Center lineTracking Neural Networks. VSTTE21 (To Appear)***

Deep Neural Networks

- Deep Neural Networks have gained immense popularity in recent times in tackling real world problems





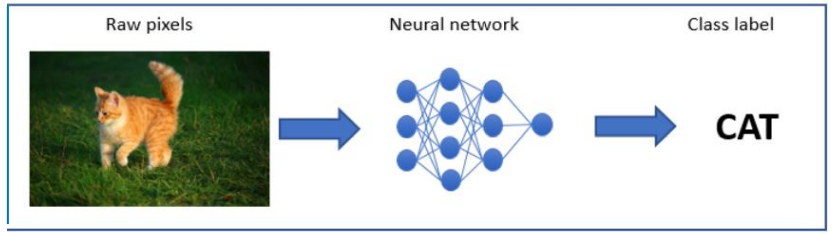
ReLU activation function

$f(x) = max(0,x)$

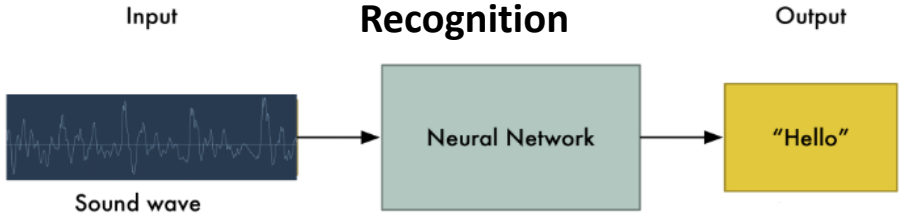
$h_{W,b}(x) = f(W^T x) = f(\sum_{i=1}^{3} Wixi + b)$

Sentiment Analysis

Image Classification

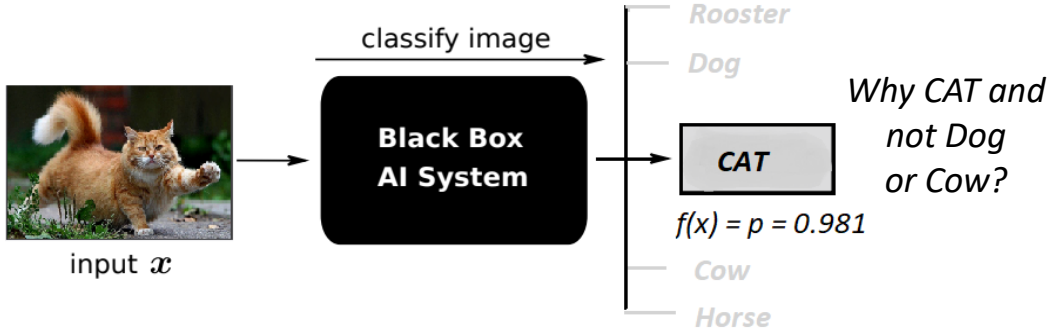


Speech Recognition



Autonomous Driving

Challenges



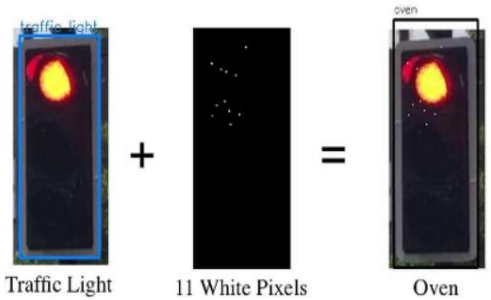
Will this also be classified as a CAT?

Are we sure this will not be classified as a CAT?

Adversarial Perturbations

Street sign

Birdhouse



Traffic Light

11 White Pixels

Oven

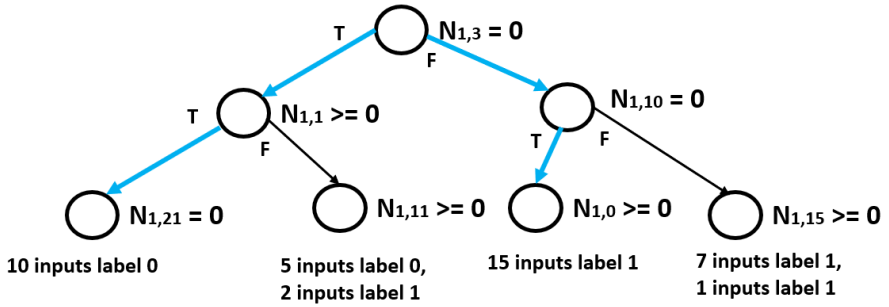
- Lack of *explainability*
  - It is not well understood why the network makes its decisions
- Lack of *guarantees for network behavior*
  - Lack of correctness oracle or specifications
- *Design not amenable for analysis*
  - Logic not interpretable
  - Networks are very large, highly interconnected structures; often have huge input spaces
- Big impediment for safety-critical domains
  - *Trust* and *Certifiability* are crucial

Prophecy
Property Inference for Deep Neural Networks [ASE 2019]

- **Key Idea:** Given a trained DNN model, *Prophecy* aims to decompose the complex model into a set of compact properties amenable to analysis

- Each property is in the form of an assume-guarantee rule, *Pre => Post*
  - Post: A property of the output of the network , such as the output of a classifier being a certain label
  - Pre: Pre-condition characterizing inputs with similar network behavior wrt Post
    - Formally specified as constraints on the neurons of the network
    - In [ASE'19] We consider *(on/off) activation patterns of ReLU neurons* enabling a programmatic view of the network ; ReLU nodes = conditional statements

- Given a set of labelled data, the neuron outputs/activations for every input and the respective network output is fed to a decision-tree learner to extract compact rules wrt Post
  - Each rule is a conjunction of linear constraints on a set of neurons and is associated a support metric; number of instances from the labelled data satisfying the Pre and Post



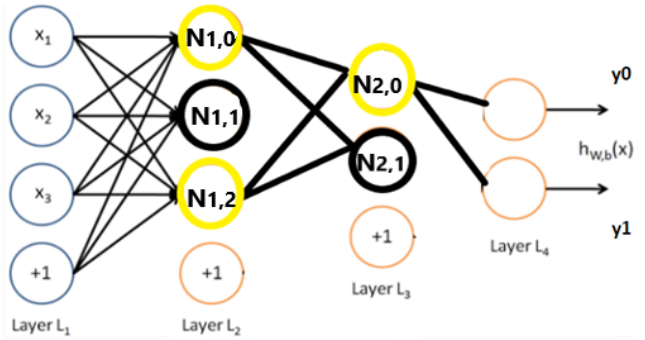
$(N_{1,3} = 0 \wedge N_{1,1} \geq 0 \wedge N_{1,21} = 0) \Rightarrow label\ 0, \quad Support = 10$

$(N_{1,3} \geq 0 \wedge N_{1,10} = 0 \wedge N_{1,0} \geq 0) \Rightarrow label\ 1, \quad Support = 15$

Prophecy
Property Inference for Deep Neural Networks [ASE 2019]

- **Input Property:** Pre is a conjunction of constraints on all neurons from the first hidden layer until a certain layer
  - Equivalent to a path in an imperative program, trace convex regions in the input space

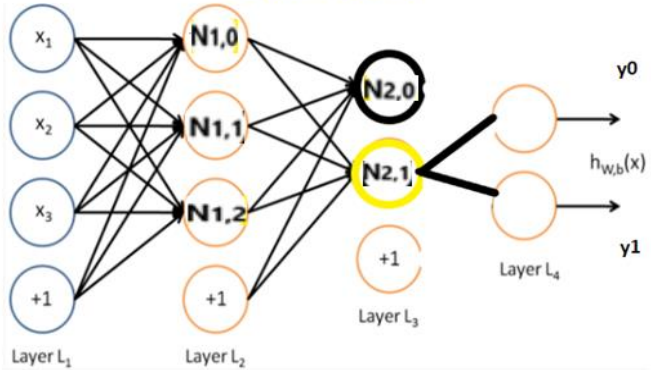
Input Property



$$(N_{1,0} \geq 0 \wedge N_{1,1} = 0 \wedge N_{1,2} \geq 0 \wedge N_{2,0} \geq 0 \wedge N_{2,1} = 0) \Rightarrow (y_0 > y_1)$$

- **Layer Property:** Pre is a conjunction of constraints on neurons at a single intermediate layer of the network
  - Intends to capture the logic wrt input features that the network uses to make its decisions

Layer Property



$$(N_{2,1} \geq 0 \wedge N_{2,0} = 0) \Rightarrow (y_1 > y_0)$$

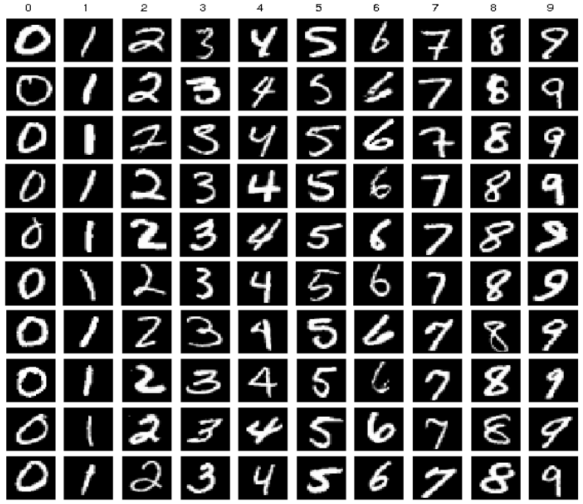
Applications

- The properties extracted using Prophecy have many applications
  - Obtaining robustness guarantees
  - Network Distillation
  - Proof Decomposition
  - Interpretability and Explainability of network behavior
  - Debugging and repair

- Case studies on perception networks, controller networks, classifier and regression models
  - Feed forward networks
  - With fully connected, convolution layers, maxpool layers
  - ReLU , eLU activation functions

Case Study on Image Classification networks

- **MNIST (Modified National Institute of Standards and Technology database)**
  - Perception network for image classification of hand-written digits
  - 10 FC layers with 10 nodes at each layer with ReLU
  - 8 layers CONV-CONV-MAX-CONV-CONV-MAX-FC-FC with ReLU



- **LFW (Labelled Faces in the Wild)**
  - Perception network for face recognition
  - 3 CONV , 4 MAX and 2 FC layers with 16 and 5 ReLU nodes respy

Formal Robustness guarantees (MNIST)

- Properties could be formally verified using a decision procedure and serve as assume-guarantee type contracts;

$$\forall x \ s.t. \ \ pattern(x) \ => (NN(x) = label)$$

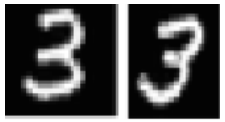
  - We proved 22 input properties and 8 layer properties across all labels for the 10 layer MNIST network using Marabou as the decision procedure

- An input property traces a convex region grouping images that are close to each other in the input space and have the same network label
  - The consistent labeling of all inputs within the convex region represents adversarial robustness of the network
- A layer property represents a union of convex regions, images that do not look the same but have same features
  - Proving a layer property indicates robustness of the network over the feature space

Inputs from the convex region of input properties

Inputs from the region for layer property for label 4

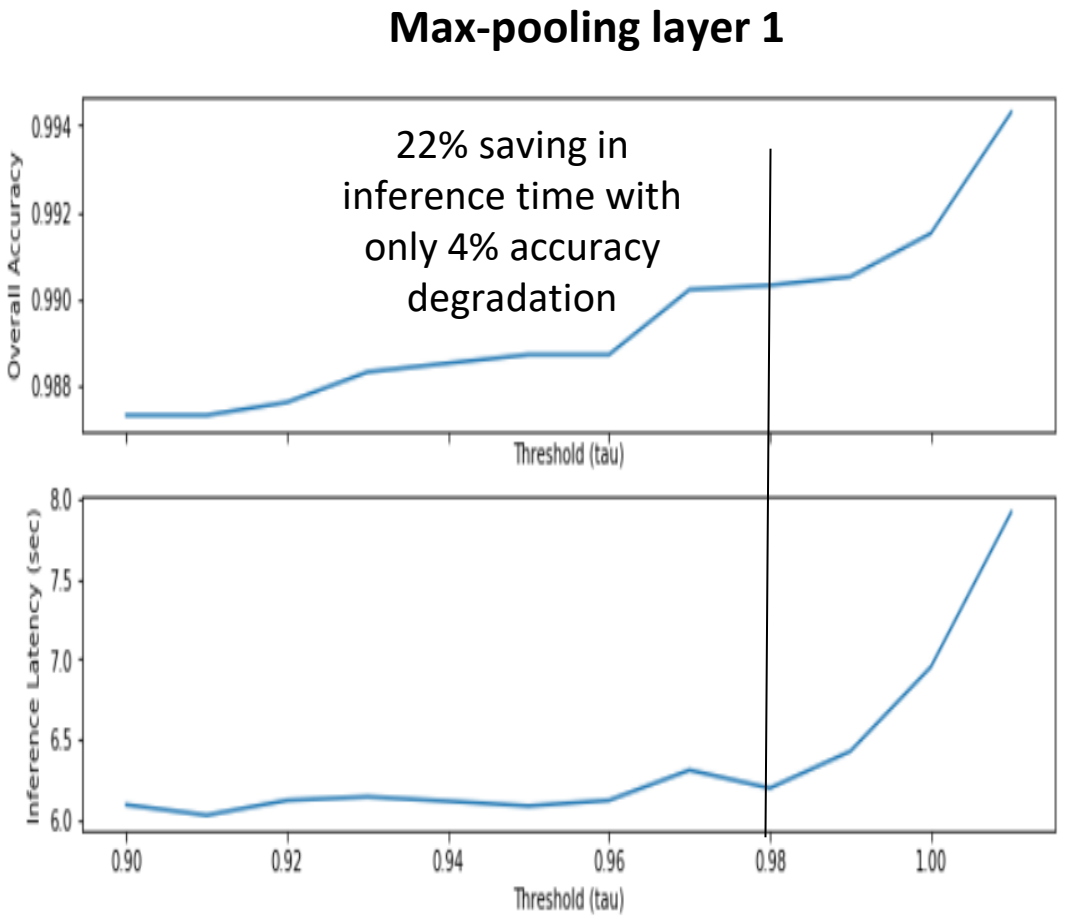
Inputs from the region for layer property for label 3



Semantically similar inputs on which the network has the same behavior

Network Distillation
(MNIST)

- Layer properties could be used as *distillation rules to save computation time*
- The 8 layer MNIST convolutional network has high accuracy (0.9943) but is computationally expensive (~ 7.9 secs per image)
- With a validation accuracy threshold of 0.98, there is 22% savings in time with only 4% loss of accuracy



NNrepair

Constraint-Based Repair of Neural Network Classifiers [CAV'21]

- Problem: The network is faulty; Low accuracy, lack of robustness, poisoned training data
- Retraining could be used to alter the neural network parameters and repair for faults
  - Expensive, does not guarantee correcting the faults, catastrophic forgetting
- Given a set of passing and failing tests, NNrepair attempts to perform a repair that precisely fixes the behavior of failing inputs while not altering the behavior on passing inputs
  - Prophecy used to extract rules corresponding to correct behavior for every label at an intermediate layer
    - Act as oracles for altering the behavior of failing tests
    - Help with localizing neurons whose behavior needs to be altered and also localizing weights which need to be modified
    - Help with preserving behavior of the network on passing tests
  - The repair constraints are solved using Z3 to precisely fix the behavior of the failing tests

NNrepair

Constraint-Based Repair of Neural Network Classifiers [CAV'21]

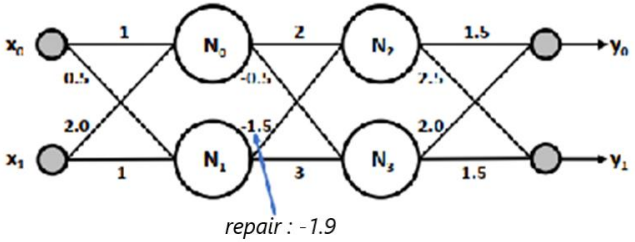


Fig. 1. Example

Table 1. Data for Example

	x_0	x_1	N_0	N_1	N_2	N_3	y_0	y_1	class	ideal
x_0	1	1	1	1	0	1	8	6	0	0
x_1	0	1	1	1	1	1	0.25	9.25	1	1
x_2	1	0	0	1	0	1	3	2.25	0	0
x_3	-1	1	1	1	1	0	-7.87	13.12	1	1
x_4	1.5	2	1	1	1	1	12.68	12.68	1	0
after repair	1.5	2	1	1	0	1	13.3	10.5	0	0

- Pattern based repair helped in all three scenarios for the MNIST network
  - Improved the overall accuracy of the model from 96.34% to 96.54% (without new data or re-training)
  - Improved the accuracy of a poisoned model from 10.38% to 12.91% on poisoned data without any decrease in accuracy on normal data
  - Improved the accuracy of the model on adversarial inputs from 28.37 % to 32.37%

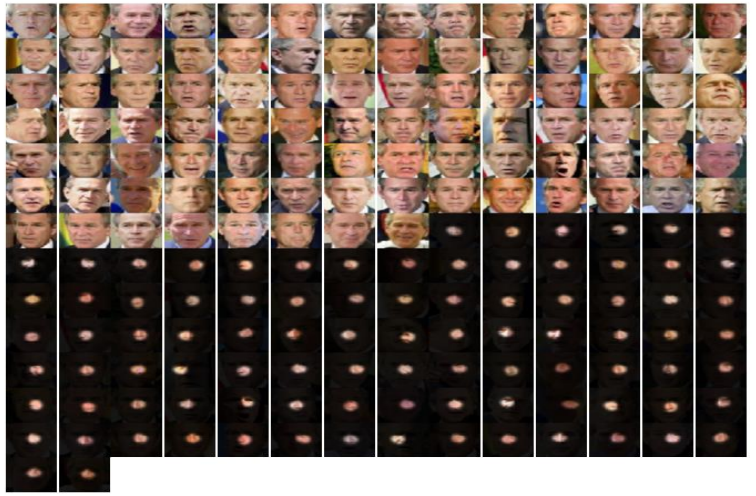
Understanding network behavior (LFW)

- Using patterns for interpretability and explainability
  - Help understand the mechanics of the network or how it processes data
  - Help the user understand decisions

- A layer pattern represents logic over features of the input images that impact network behavior
  - We visualize these features by highlighting the input pixels that impact the pattern
  - We extracted a layer patterns for every label from the first fully connected layer of the LFW network
  - We used a gradient based approach to identify pixels that impact the neurons in the pattern akin to attribution approaches
  - This highlighted the nose to be the feature of the faces that the LFW network uses to make its decisions

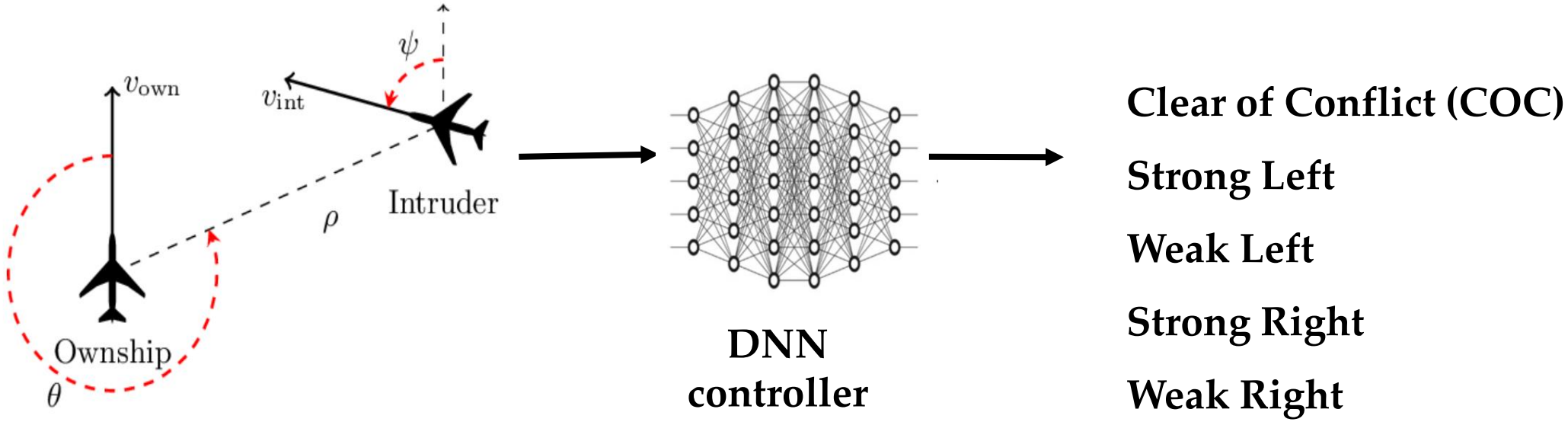
Images of Powel satisfying the same layer property

Images of Bush satisfying the same layer property



Case study on a Controller network

ACAS-Xu (Airborne Collision Avoidance System-Xu)



Proof Decomposition, Specifications Inference

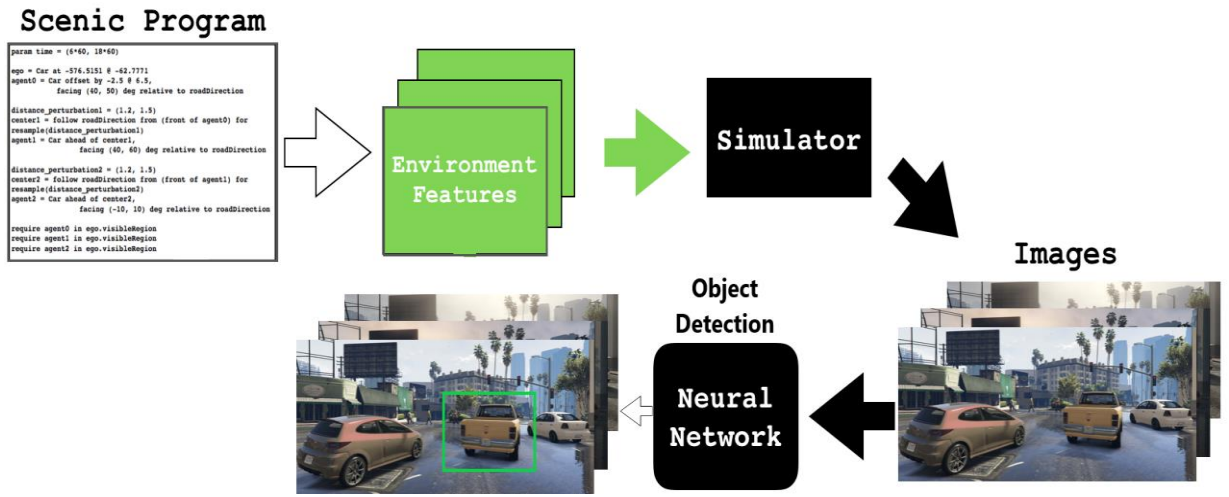
- **ACAS Xu** has domain-level specifications that the network needs to satisfy
  - A => B, where A represents a predicate on the input space and B is a turning advisory
  - Proof on the full network consumes a lot of time using Reluplex
- Decomposed proofs of properties of the form A => B , using "layer patterns" σ,
  - By checking A => σ and σ =>B separately w/ Reluplex;
  - Speedup of upto 75% obtained **speedup** obtained
  - Checked property that timed out with monolithic verification

- **ACAS Xu** has meaningful input variables
  - Representing network properties in terms of input variables leads to the discovery of the specifications of the domain

  - 31900 ≤ range ≤ 37976, 1.684 ≤ θ ≤ 2.5133, ψ = -2.83, 414.3 ≤ vown ≤ 506.86, vint = 300, has turning advisory **COC**
  - range = 499, -0.314 ≤ θ ≤ -3.14, -3.14 ≤ ψ ≤ 0, 100 ≤ vown ≤ 571, 0 ≤ vint ≤ 150, has turning advisory **Strong Left**
  - range = 48608, θ = -3.14, ψ = -2.83, vown(full range), vint (full range) has turning advisory **COC**

Case study on a Object Detection Network

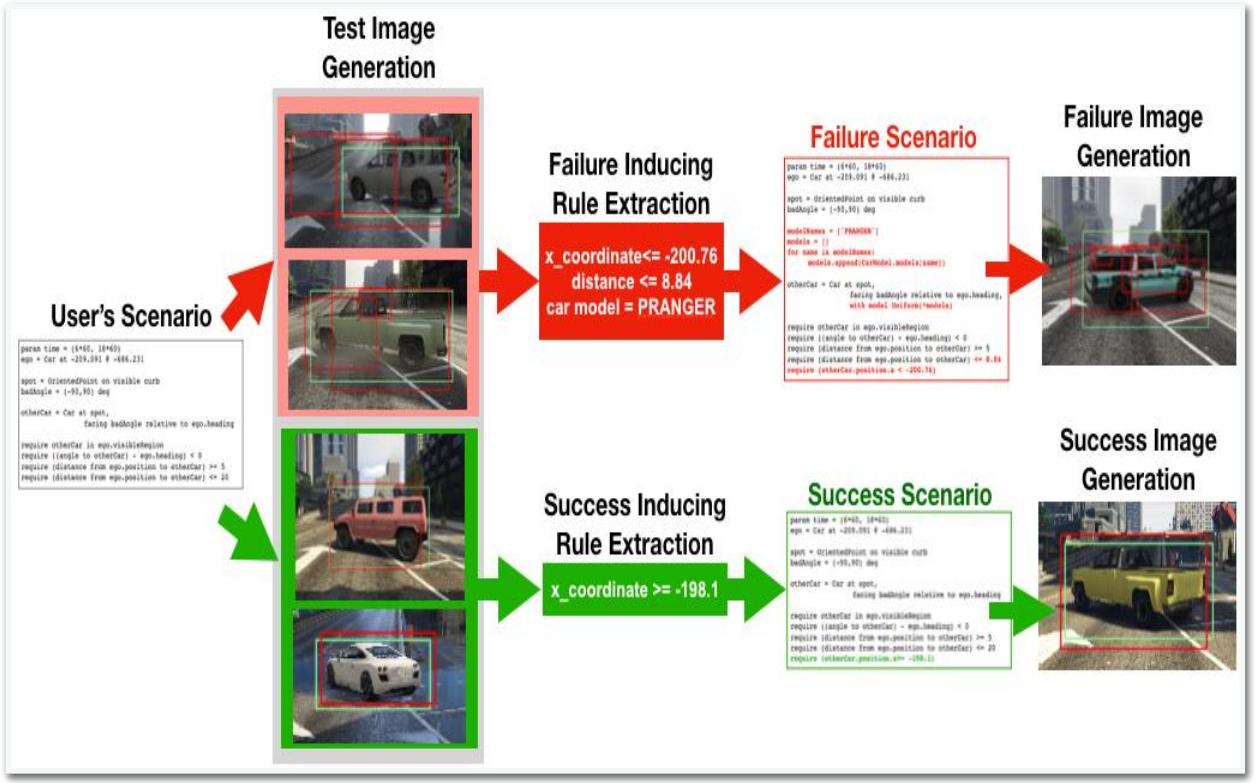
- SqueezeDet is a convolutional neural network for object detection in autonomous cars
- SCENIC is a probabilistic programming language used to describe environments or scenes
- The program generates values for environment variables describing a scene; weather condition, time of day, vehicle type, position so on; high level semantic features
- The environment variables fed to a simulator ((GTA-V) game engine) to create realistic images for the object detector network



Extracting Semantic Explanations
[CVPR'20]

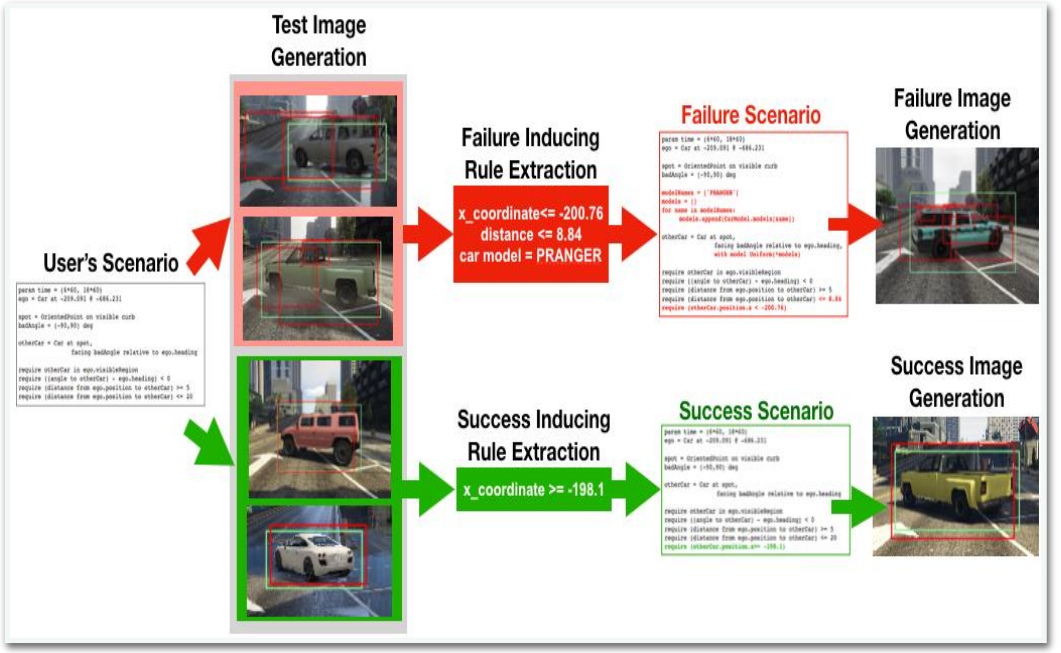
- Given a set of passing and failing tests, we employed Prophecy to extract rules for correct and incorrect behavior

- The rules (in terms of neuron activation patterns) were mapped to high-level semantic features
  - The environment features corresponding to each image were recorded
  - Images were bucketed based on the patterns they satisfy
  - Decision-tree learning was applied to extract rules in terms of the features for images in each bucket



Extracting Semantic Explanations
[CVPR'20]

- The rules in terms of high-level features act as explanations for the correct and incorrect behavior
- The failure inducing rule is used to refine the scenic program to generate more failure inducing images
- The success inducing rule is used to refine or correct the program to generate more passing tests
- Increased correct detection rate from 65.3% to 89.4% and incorrect detection rate from 34.7% to 87.2%
- These additional tests could be used to re-train the object detector network



Case study on a Regression model for Perception (VSTTE'21)

- **TaxiNet** is a neural network designed to take a single picture of the runway as input and return the plane's position w.r.t. the middle of the runway
  - Returns 2 numerical outputs; Cross track error (y0): The distance of the plane from the middle line, Heading error (y1): The angle of the plane w.r.t. the runway
  - A simulator captures an image every 0.1 seconds to use for training and testing

https://blog.send-anywhere.com/blog/2016/01/13/uc-3-entertainment-for-travel/boeing-aircraft-plane-on-runway-free-wallpaper-hd/

- **Networks:** ACT TaxiNet (industry partner), KJ-Taxinet (smaller network with ReLU activations)

- **Our Goal**
  - Understand (explain) the behavior of the model and
  - Provide robustness guarantees w.r.t properties of the output
    - Correctness properties: Using error bounds: $|y0-y0ideal| < 1.5m$, $|y1-y1ideal| < 5°$
    - Safety properties: Using runway dimensions: $|y0| < 10.0m$, $|y1| < 90°$

Explaining Correct and Incorrect Behavior (VSTTE'21)

- Prophecy used to extract activation patterns w.r.t satisfaction and violation of the correctness property for ACT Taxinet
  - Example activation pattern for satisfaction of correctness property for y0

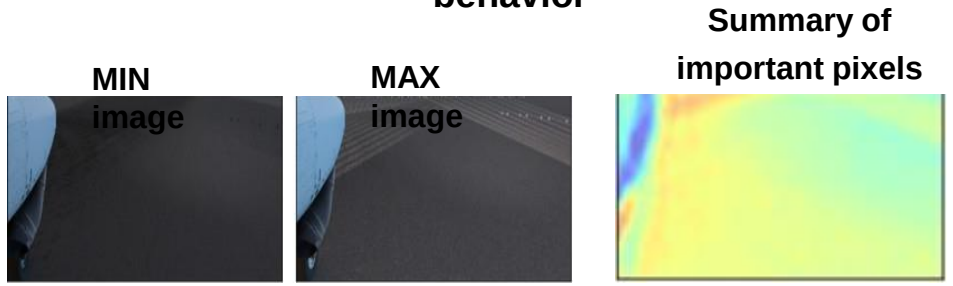
    off(N1,53) /\ off(N1,33) /\ off(N1,71) /\ off(N1,64) /\ off(N1,67) => |y0-y0ideal| <= 1.5
  - Example activation pattern for violation of correctness property for y0

    on(N1,53) /\ off(N1,29) /\ on(N1,20) /\ off(N1,49) /\ off(N1,15) /\ off(N1,95) /\ off(N1,25) => |y0-y0ideal| > 1.5

Sequence of 10 images for correct behavior

- Visualizing the features represented by the patterns
  - For every image satisfying a pattern, used GradCAM++ to identify pixels impacting the neurons in the pattern
  - Leveraged sequence of images satisfying the same pattern to determine a common generic feature across images during an interval of time
    - Heatmap created for a sequence of images (scenario) by using the average gradcam values for every pixel

Sequence of 18 images for incorrect behavior



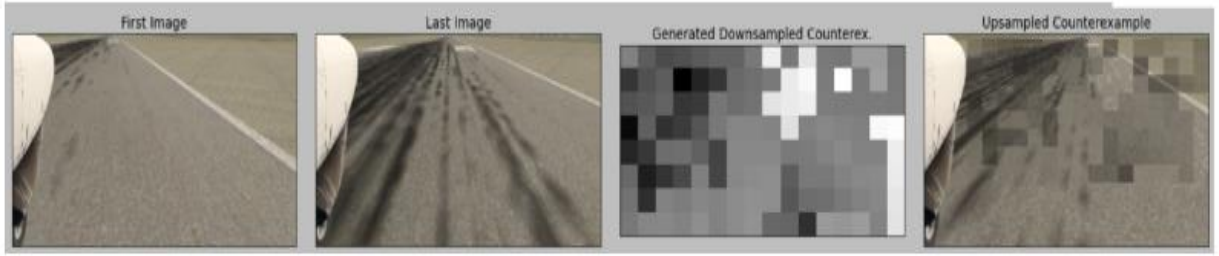
Robustness guarantees
(VSTTE'21)

- Prophecy used to extract activation patterns w.r.t satisfaction of the safety property for KJ-TaxiNet

- We employed Marabou to prove these properties
  - Leveraged sequence of images satisfying the same pattern to determine intervals of time within which the network is under safe operation
    $\forall x \in [x_{min}, x_{max}] \wedge pattern => |y_0| \leq 10m$
  - Obtained proofs for 33 sequences with at least 5 images, the longest sequence with proof had 17 images

- Counter-example to the proofs is an image which is similar to other valid images in the sequence but violates the safety property

Counterexample for an image sequence of length 39 for $|y_0| \leq 10m$



Counterexample for an image sequence of length 5 for $|y_0| \leq 10m$

Future Work

- Runtime monitors
  - Monitor for abnormal behaviors (deviations from expected behavior) based on patterns for correct and incorrect behavior
- Exploring structural coverage metrics for Neural Networks
  - Patterns extracted by Prophecy have the potential to capture the behavioral / functional / feature coverage

Thank You!

https://ti.arc.nasa.gov/tech/rse/research/safednn/