Analysis and Testing for AI software

Wei Le

April 18, 2019

AI Software

- machine learning applications (neural networks and others)
- code written in probabilistic programming languages

Neural Networks

- training the model using massive amount of data and then predict the labels given new inputs
- autonomous vehicles
- aircraft collision avoidance detection

Testing for Autonomous Vehicles at Uber

- collect real-world data
- generate synthetic data based on real-world data, e.g., lack of collision scenarios
- constantly running testing for the Autonomous Vehicles system
  - can you recognize the background data
  - can you recognize moving object
  - can you predict the next movement of the moving object

Neural Networks Input and Output

- Input: 3 Dimensional inputs (e.g., colored images)
- Reshape the input to vectors
- Output: labels (e.g., is it 1, 2, ...9)
- Using massive amount of labeled data to "parameterize" neural network for a specific problem, which we call *models*, models then used like a classifier

Neural Networks Internal

- consists of sequences of layers, e.g., input/output layer, and hidden layers
- each layer consists of *neurons* (also called *perceptron*)
- *feed-forward*: the output of a neuron is not feedback to the previous layer
- *convolutional neural networks*: *fully connected layers*, *pooling layers* and *convolutional layers*

Challenges

- Robustness: are neural network vulnerable to *adversarial examples* – slightly perturbing an input classified correctly leads to mis-classification?
- Testing: How to test models so we know it is a good model and ready to go for application?
- Debugging: if the prediction is wrong, is it a problem of insufficient training data, implementation errors in networks, or it is an error expected by the algorithm?

Can program analysis help?

- differential analysis: compare the output of neurons for correct and incorrect predictions [4]
- compare with other versions of software [5]
- abstract interpretation [1]

AI^2 Safety and Robustness Certification of Neural Networks with Abstract Interpretation

- AI^2: *abstract interpretation* for artificial intelligence
- Goal: prove safety properties (e.g, robustness)
- Example:
  - FGSM attack: Adding a particular noise vector multiplied by a small number ϵ, can neural net still correctly classify the digit
  - Brighten attack: change all pixes above the threshold 1-δ to the brightest possible value

Attack	Original	Perturbed
FGSM [12], $\epsilon = 0.3$		
Brightening, $\delta = 0.085$		

AI^2

- *abstract interpretation*: Sound, computable and precise finite approximation of potentially infinite sets of behaviors
- Construct an *abstract element* that captures all perturbed images (more than 10^{1154} images)
- Construct *abstract transformer* that compute the effect of neural net on the abstract element
- if the abstract output satisfies the property, all concrete inputs satisfy the property (soundness of abstract intepretation)

Abstract Interpretation

See Alex Aiken's slides for an example of abstract interpretation

Example Lattice

Concrete lattice

$$(L, \sqsubseteq)$$

Abstract lattice

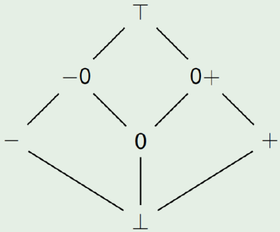
$$(\overline{L}, \overline{\sqsubseteq})$$

Example (Sets of Values)

For a variable ranging over a domain $\mathbb{D}$:

$$(\mathcal{P}(\mathbb{D}), \subseteq)$$

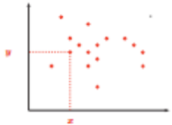
Example (Sign Lattice)



AI^2: abstract domain

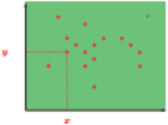
- An *abstract domain* consists of logical formulas that capture certain shape
- The *Zonotope domain*

Partition and Abstract Domain



$\{\ldots, \langle 5, 7\rangle, \ldots, \langle 13, 21\rangle, \ldots\}$

(a) [In]finite Set of Points

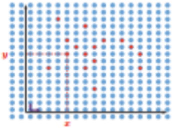


$\begin{cases} x \geq 0 \\ y \geq 0 \end{cases}$

(b) Sign Abstraction

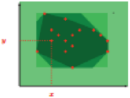
$\begin{cases} x \in [3,\ 27] \\ y \in [4,\ 32] \end{cases}$

(c) Interval Abstraction



$\begin{cases} x = 5 \bmod 8 \\ y = 7 \bmod 9 \end{cases}$

(d) Simple Congruence Abstraction



$\begin{cases} 7x + 3y \leq 5 \\ 2x + 7y \geq 0 \end{cases}$

(b) Polyhedral Abstraction

Design abstract transformers

- represent neural network as *conditional affine transformations (CAT)*
- abstract affine functions
- abstract case functions (conditional)

Defining CAT functions

$$
\begin{array}{rcl}
f(\overline{x}) & ::= & W \cdot \overline{x} + \overline{b} \\
& | & \textbf{case } E_1 : f_1(\overline{x}), \ldots, \textbf{case } E_k : f_k(\overline{x}) \\
& | & f(f'(\overline{x})) \\
E & ::= & E \wedge E \mid x_i \geq x_j \mid x_i \geq 0 \mid x_i < 0
\end{array}
$$

Mapping different layers to CAT functions

Done manually based on algorithms

- activation function (ReLU) to CAT
- Pooling layer (extract features), convolution layer (extract features) and fully connected layer (classification) to CAT

Deep Gauge: Multi-Granuality Testing Criteria for Deep Learning Systems

- each neuron computes an output based on an input
- each layer computes an output based on an input
- cover all possible output values
- design a family of test criteria for neuron level and layer level

Neuron Level Criteria

- k-multisection neuron coverage: given a neuron n, the criterion measures how thoroughly the given set of test inputs T covers the range $[\text{low}_n, \text{high}_n]$
- neuron boundary converage: how many tests cover the corner cases of (high_n, ∞), and $(-\infty, \text{low}_n)$
- strong neuron activation coverage: how many corner cases w.r.t. the upper boundary value high_n has been covered

Layer Level Criteria

- define "active neurons": for a given test input x and neuron n_1 and n_2, we say n_1 is more active than n_2 given x if the output of n_1 regarding x is larger than the output of n_2
- test data should uncover more active neurons
- top k neuron coverage: how many neurons of a layer has been the most active k neurons
- top k neuron patterns: how many top k neuron patterns are covered

Experimental Setup

- test dataset: MNIST and ImageNet
- include also adversarial test dataset

Findings

- the data set cover both main function region and corner cases, but cover the main function region more than corner cases
- the adversarial test dataset boost the coverage criteria
- lower region is more difficult to cover than the higher region

Further Reading

1. AI^2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation
2. Deep Gauge: Multi-Granularity Testing Criteria for Deep Learning Systems
3. Brian McClendon's talk that covers testing for Autonomous Vehicles at Uber
4. MODE: Automated Neural Network Model Debugging via State Differential Analysis and Input Selection
5. CRADLE: Cross-Backend Validation to Detect andLocalize Bugs in Deep Learning Libraries