Analysis and Testing for AI software

Wei Le

April 5, 2021

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]

Guiding Deep Learning System Testing using Surprise Adequacy

Testing neural network in the era of AI engineering:

- in machine learning community, we use cross-folding, separate training dataset, validation dataset (tuning), test dataset randomly for 3-10 times.
- moving to AI engineering, we need high quality products
- introducing software testing methodology, e.g., test criterion, when we gain confidence that the model is ready to deploy

Some recent work on neural network test criteria

Are all the neurons activated? when the nerons are activated, are a range of values/boundary values covered?

- DeepXplore's Neuron Coverage (NC)
- Three Neuron-level Coverages (NLCs) introduced by Deep-Gauge [27]: k-Multisection Neuron Coverage (KMNC), Neuron Boundary Coverage (NBC), and Strong Neuron Activation Coverage (SNAC).
  - k-multisection neuron coverage: given a neuron n, the criterion measures how thoroughly the given set of test inputs T covers the range [low_n, high_n]
  - neuron boundary converage: how many tests cover the corner cases of (high_n, ∞), and ($-\infty$, low_n)
  - strong neuron activation coverage: how many corner cases w.r.t. the upper boundary value high_n has been covered

Surprise of an input

- after training, the neural network should be able to handle similar input
- how surprise a given input compared to the training set
- what is the metric to meausre the surprise

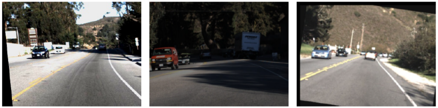
Surprise metrics: LSA

$$\hat{f}(x) = \frac{1}{|A_{N_L}(\mathbf{T})|} \sum_{x_i \in \mathbf{T}} K_H(\alpha_{N_L}(x) - \alpha_{N_L}(x_i)) \qquad (1)$$

Adopting common approach of converting probability density to a measure of rareness [26], [39], we define LSA to be the negative of the log of density:

$$LSA(x) = -log(\hat{f}(x)) \qquad (2)$$

Surprise metric LSA matches human perception



(a) Low LSA

(b) Medium LSA

(c) High LSA

Fig. 3: Synthetic images for Chauffeur model generated by DeepTest. Images with higher LSA values tend to be harder to recognise and interpret visually.

Surprise metrics: LSA

$$\hat{f}(x) = \frac{1}{|A_{N_L}(\mathbf{T})|} \sum_{x_i \in \mathbf{T}} K_H(\alpha_{N_L}(x) - \alpha_{N_L}(x_i)) \quad (1)$$

Adopting common approach of converting probability density to a measure of rareness [26], [39], we define LSA to be the negative of the log of density:

$$LSA(x) = -log(\hat{f}(x)) \quad (2)$$

Surprise metric DSAL: Euclidian Distance of between two activation traces

$$
\begin{aligned}
x_a &= \operatorname*{argmin}_{\mathbf{D}(x_i)=c_x} \|\alpha_{\mathbf{N}}(x) - \alpha_{\mathbf{N}}(x_i)\|, \\
dist_a &= \|\alpha_{\mathbf{N}}(x) - \alpha_{\mathbf{N}}(x_a)\|
\end{aligned} \tag{3}
$$

Subsequently, from x_a, we find the closest neighbour of x_a in a class other than c_x, x_b, and the distance $dist_b$, as follows:

$$
\begin{aligned}
x_b &= \operatorname*{argmin}_{\mathbf{D}(x_i)\in C\setminus\{c_x\}} \|\alpha_{\mathbf{N}}(x_a) - \alpha_{\mathbf{N}}(x_i)\|, \\
dist_b &= \|\alpha_{\mathbf{N}}(x_a) - \alpha_{\mathbf{N}}(x_b)\|
\end{aligned} \tag{4}
$$

Intuitively, DSA aims to compare the distance from the AT of a new input x to known ATs belonging to its own class, c_x, to the known distance between ATs in class c_x and ATs in other classes in $C \setminus \{c_x\}$. If the former is relatively larger than the latter, x would be a surprising input for class c_x to the classifying DL system $\mathbf{D}$. While there are multiple ways to formalise this we select a simple one and calculate DSA as the ratio between $dist_a$ and $dist_b$. Investigation of more complicated formulations is left as future work.

$$
DSA(x) = \frac{dist_a}{dist_b} \tag{5}
$$

Research Questions

- RQ1. Surprise: Is SADL capable of capturing the relative surprise of an input of a DL system?
- RQ2. Layer Sensitivity: Does the selection of layers of neurons used for SA computation have any impact on how accurately SA reflects the behaviour of DL systems?
- RQ3. Correlation: Is SC correlated to existing coverage criteria for DL systems?
- RQ4. Guidance: Can SA guide retraining of DL systems to improve their accuracy against adversarial examples and synthetic test inputs generated by DeepXplore?

Experiment Setup

TABLE I: List of datasets and models used in the study.

Dataset	Description	DNN Model	# of Neuron	Synthetic Inputs	Performance
MNIST	Handwritten digit images composed of 50,000 images for training and 10,000 images for test.	A five layer ConvNet with max-pooling and dropout layers.	320	FGSM, BIM-A, BIM-B, JSMA, C&W.	99.31% (Accuracy)
CIFAR-10	Object recognition dataset in ten different classes composed of 50,000 images for training and 10,000 images for test.	A 12 layer ConvNet with max-pooling and dropout layers.	2,208	FGSM, BIM-A, BIM-B, JSMA, C&W.	82.27% (Accuracy)
Udacity Self-driving Car Challenge	Self-driving car dataset that contains camera images from the vehicle, composed of 101,396 images for training and 5,614 images for test. The goal of the challenge is to predict steering wheel angle.	Dave-2 [6] architecture from Nvidia.	1,560	DeepXplore's test input generation via joint optimization.	0.09 (MSE)
		Chauffeur [1] architecture with CNN and LSTM.	1,940	DeepTest's combined transformation.	0.10 (MSE)

Results: suprise of the adversial examples

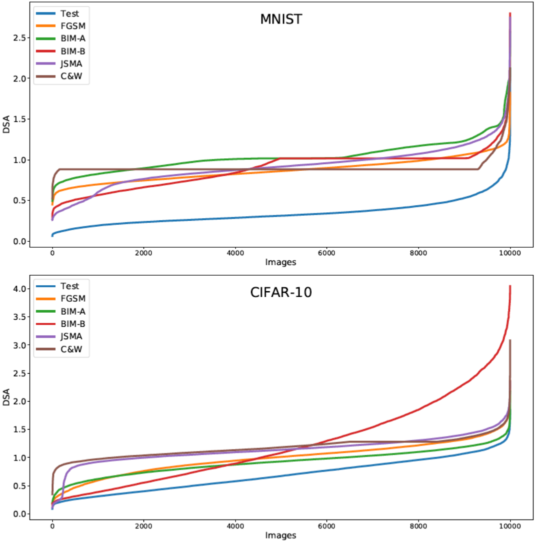


Fig. 4: Sorted DSA values of adversarial examples for MNIST and CIFAR-10.

Results: sensitve to layer selection

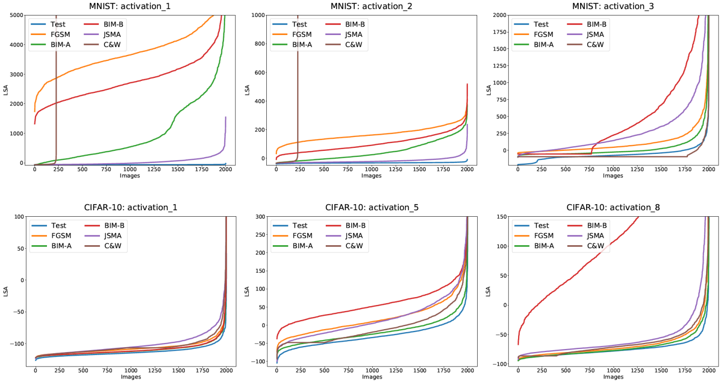


Fig. 5: Sorted LSA of randomly selected 2,000 adversarial examples for MNIST and CIFAR-10 from different layers

Results: correlating with other test criteria

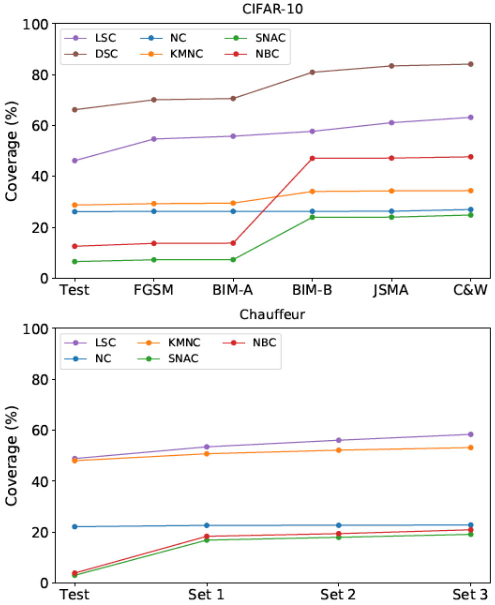


Fig. 6: Visualisation of CIFAR-10 and Chauffeur in Table VI.

Results: useful to guide test input generation

DNN Model	SA	R	FGSM		BIM-A		BIM-B		JSMA		C&W	
			μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
MNIST		$\emptyset$	11.65	-	9.38	-	9.38	-	18.88	-	8.92	-
	LSA	1/4	25.81	1.95	95.14	0.69	**41.00**	0.01	72.67	3.09	92.51	0.51
		2/4	28.45	2.91	95.71	0.41	40.98	0.12	75.03	2.68	92.55	0.67
		3/4	**29.66**	3.63	95.87	0.98	40.97	0.10	75.48	2.60	92.41	1.03
		4/4	23.70	4.98	**95.90**	0.79	40.93	0.18	**77.37**	1.75	**92.56**	0.77
	DSA	1/4	15.60	2.12	93.67	3.42	9.90	1.05	74.56	2.62	12.80	0.96
		2/4	19.67	4.32	**95.78**	0.70	9.40	0.05	76.16	2.69	12.46	1.00
		3/4	26.37	6.15	95.37	0.93	40.81	0.22	**78.01**	1.87	12.37	1.14
		4/4	**27.69**	5.59	95.31	0.98	**40.94**	0.04	76.60	2.38	**13.61**	1.19
CIFAR-10		$\emptyset$	6.13	-	0.00	-	0.00	-	2.68	-	0.31	-
	LSA	1/4	11.07	1.20	32.34	1.70	0.59	1.76	32.80	2.05	34.38	2.83
		2/4	**12.96**	2.18	32.68	2.07	**0.89**	2.10	33.84	2.52	42.99	2.78
		3/4	12.79	2.17	32.14	2.40	**0.89**	2.10	35.81	2.81	45.58	2.23
		4/4	12.53	1.19	**32.79**	2.29	0.60	1.76	**35.83**	2.54	**45.74**	2.04
	DSA	1/4	**14.86**	2.16	25.94	2.99	0.01	0.00	34.92	2.01	44.21	2.02
		2/4	14.64	1.95	29.59	3.52	0.01	0.00	34.49	1.89	44.79	2.32
		3/4	13.81	1.85	31.93	2.77	0.01	0.00	35.61	2.40	46.16	2.45
		4/4	13.12	1.41	**32.17**	2.36	**0.60**	1.76	**37.32**	1.58	**46.21**	2.72

(a) MNIST and CIFAR-10

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)

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

Defining CAT functions

represent neural network as *conditional affine transformations (CAT)*

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

Abstract Interpretation

Fig. 6 shows a CAT function $f\colon \mathbb{R}^2 \to \mathbb{R}^2$ that is defined as $f(\overline{x}) = \left(\begin{smallmatrix} 2 & -1 \\ 0 & 1 \end{smallmatrix}\right) \cdot \overline{x}$ and four input points for the function f, given as $X = \{(0,1),(1,1),(1,3),(2,2)\}$. Let the property be $C = \{(y_1, y_2) \in \mathbb{R}^2 \mid y_1 \geq -2\}$, which holds in this example. To reason about all inputs simultaneously, we lift the definition of f to be over a set of inputs X rather than a single input:

$$T_f\colon \mathcal{P}(\mathbb{R}^m) \to \mathcal{P}(\mathbb{R}^n), \qquad T_f(X) = \{f(\overline{x}) \mid \overline{x} \in X\}.$$

The function T_f is called the *concrete transformer* of f. With T_f, our goal is to determine whether $T_f(X) \subseteq C$ for a given input set X. Because the set X can be very large (or infinite), we cannot enumerate all points in X to compute $T_f(X)$. Instead, AI overapproximates sets with abstract elements (drawn from some abstract domain $\mathcal{A}$) and then defines a function, called an *abstract transformer* of f, which works with these abstract elements and overapproximates the effect of T_f. Then, the property C can be checked on the resulting abstract element returned by the abstract transformer.

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