DeepXplore: Automated Whitebox Testing of Deep Learning Systems

by Kexin Pei, Yinzhi Cao, Junfeng Yang, Suman Jana

Eric Langlois

University of Toronto

March 11, 2019

DeepXplore

Fast automatic generation of test inputs for a set of neural networks, where

- the networks disagree, and
- the examples have high diversity.

Contributions

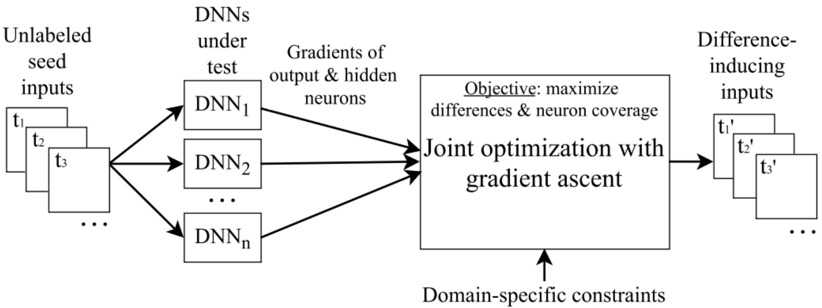
Introduced neuron coverage as a testing metric for DL systems.

Formulated the task of finding behaviour differences in a set of networks as a gradient descent optimization problem.

Created the DeepXplore open-source deep learning testing framework.

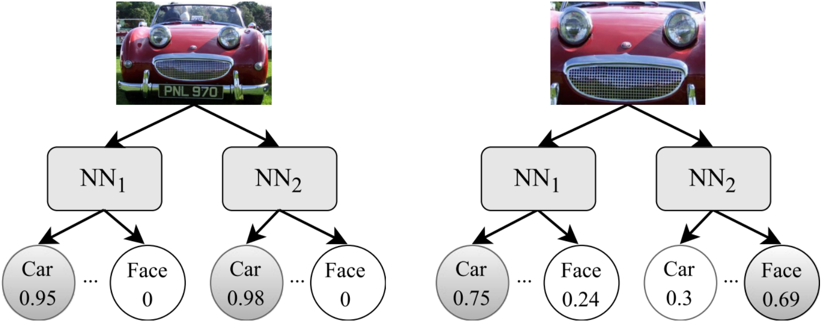
Showed that training on DeepXplore-generated tests can increase classification accuracy.

Algorithm



Unlabeled seed inputs
t_1
t_2
t_3
...
DNNs under test
DNN_1
DNN_2
...
DNN_n
Gradients of output & hidden neurons
Objective: maximize differences & neuron coverage
Joint optimization with gradient ascent
Domain-specific constraints
Difference-inducing inputs
t_1'
t_2'
t_3'
...

Network Prediction Differences

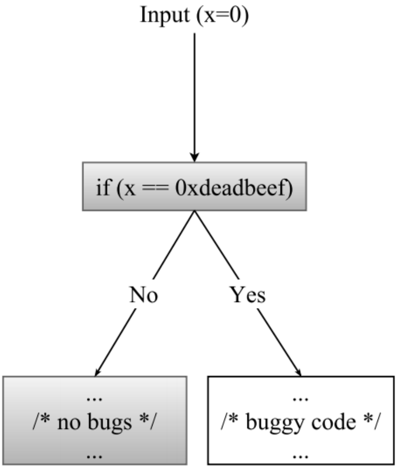


$$\mathrm{obj}_1(\boldsymbol{x}) = \sum_{k \neq j} F_k(\boldsymbol{x})[c] - \lambda_1 F_j(x)[c]$$

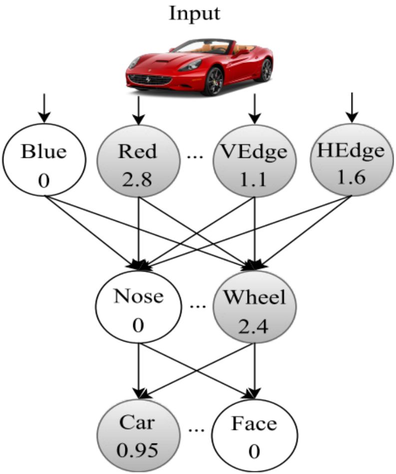
$F_k(\boldsymbol{x})[c]$ = The probability according to network k that input $\boldsymbol{x}$ is class c.

j is chosen randomly.

Coverage



Input (x=0)
if (x == 0xdeadbeef)
No
Yes
...
/* no bugs */
...
...
/* buggy code */
...



Input
Blue
0
Red
2.8
...
VEdge
1.1
HEdge
1.6
Nose
0
...
Wheel
2.4
Car
0.95
...
Face
0

Coverage Metric

Input Set Coverage

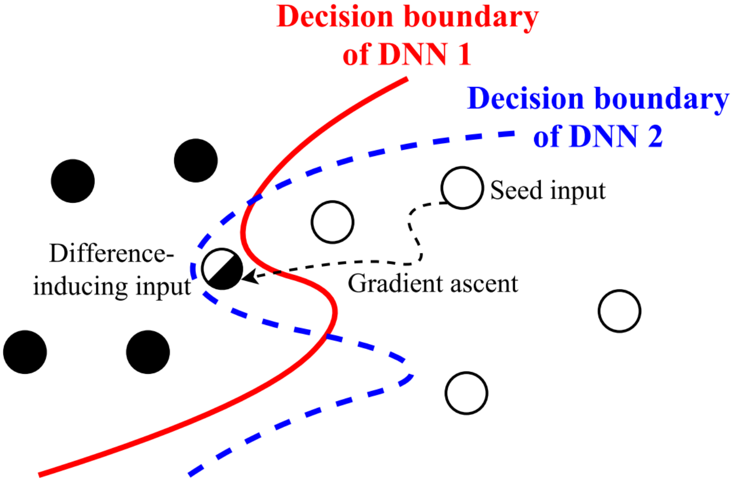
For a test set T, neuron set N, and threshold t

$$\mathrm{NCov}(T, t) = \frac{|\{n \in N \mid \overset{\exists}{\cancel{\forall}} \boldsymbol{x} \in T, \mathrm{out}(n, \boldsymbol{x}) > t\}|}{|N|}$$

Coverage Loss

$$\mathrm{obj}_2(\boldsymbol{x}) = \sum_k \text{value of neuron } n_k \text{ in network } k \text{ on input } \boldsymbol{x}$$

n_k is chosen randomly among neurons that are not yet covered.



$$\max_{\boldsymbol{x}} \mathrm{obj}_1(\boldsymbol{x}) + \mathrm{obj}_2(\boldsymbol{x})$$

Constraints

Generated inputs should be realistic.
Enforce this with domain-specific constraints.

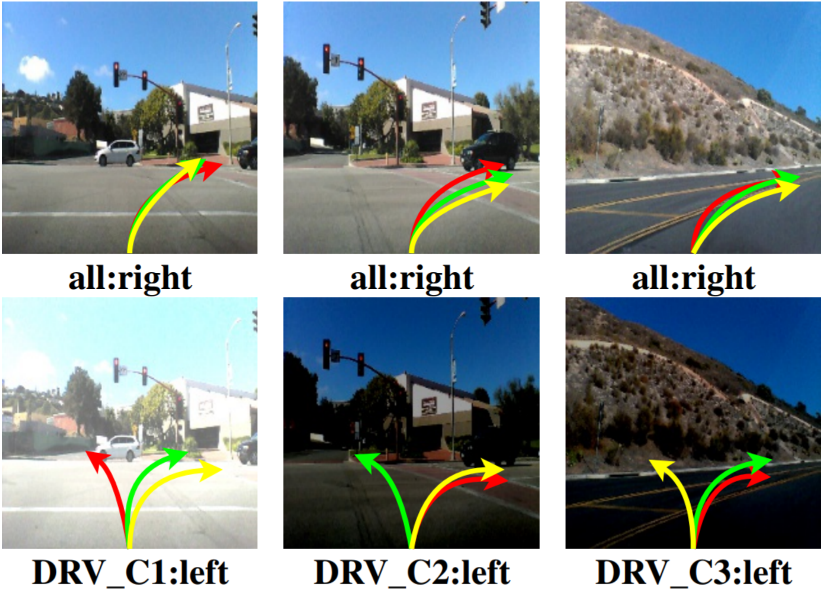
Images

- Pixel value bounds: [0, 255]
- Only modify brightness
- Only modify a small region
- Only add small black boxes

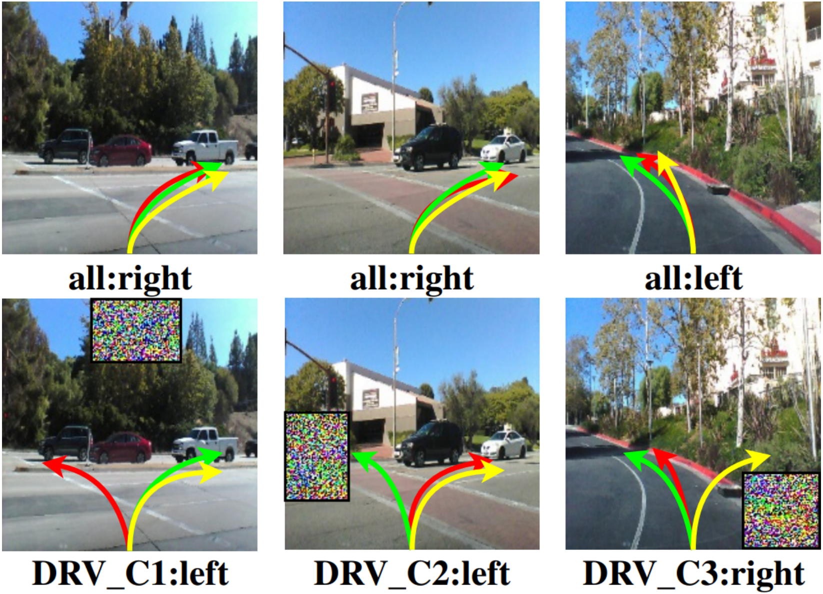
Constraints applied by modifying the gradient

$$\boldsymbol{x} \leftarrow \boldsymbol{x} + \text{step\_size} \cdot \text{constrain\_grad}\left(\frac{\partial \text{obj}(\boldsymbol{x})}{\boldsymbol{x}}\right)$$

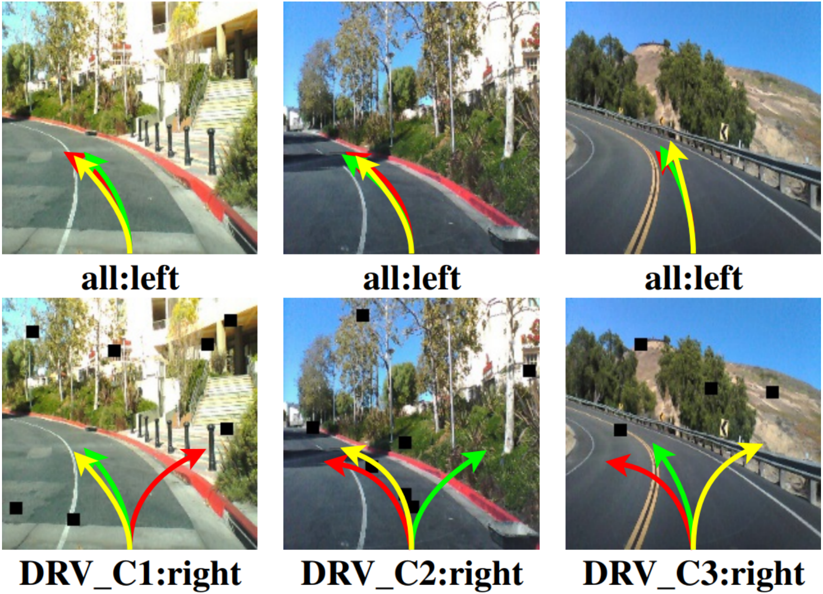
Examples — Constraint: Brightness



Examples — Constraint: Occlusion



Examples — Constraint: Black Boxes



Experiments

Domains

MNIST	:	Classify handwritten digits
ImageNet	:	Classify images
Driving	:	Predict steering angle from images
Contagio	:	Classify PDF malware
Drebin	:	Classify android app malware

Networks

- 3 networks each
- Pre-trained networks or based on popular architecture

Coverage

- Threshold $t = 0$, 0.25, or 0.75

Results

Efficiency

Model differences are found for 40% – 100% of seed inputs.

Speed

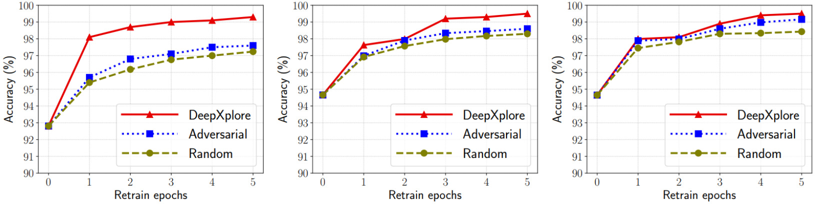
First difference-inducing input is found in seconds,
100% coverage is achieved in 6s – 200s

Design Validation

Using coverage moderately increases example diversity (by L1 distance)

Increasing model dissimilarity increases the number of differences found

Application: Training Data Augmentation



(a) LeNet-1 (b) LeNet-4 (c) LeNet-5

Add generated inputs to the training data and retrain.

Discussion

- How meaningful are the generated examples?
- Is it reasonable to use model disagreement as an objective?
- Are the constraints plausible?
- What are some alternative coverage measures?