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

DNNs under test

Gradients of output & hidden neurons

DNN_1

DNN_2

... DNN_n

Joint optimization with gradient ascent

Objective: maximize differences & neuron coverage

Difference-inducing inputs

Domain-specific constraints
Network Prediction Differences

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

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

\( j \) is chosen randomly.
Coverage

Input (x=0)

\[
\text{if (x == 0xdeadbeef)}
\]

No

... */ no bugs */ ...

Yes

... */ 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 $t$

$$\text{NCov}(T, t) = \frac{|\{n \in N | \forall x \in T, \text{out}(n, x) > t\}|}{|N|}$$

Coverage Loss

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

$n_k$ is chosen randomly among neurons that are not yet covered.
Test Inputs via Optimization

Decision boundary of DNN 1

Decision boundary of DNN 2

Difference-inducing input

Seed input

Gradient ascent

\[ \max_x \text{obj}_1(x) + \text{obj}_2(x) \]
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

\[ \mathbf{x} \leftarrow \mathbf{x} + \text{step}_\text{size} \cdot \text{constrain}_\text{grad}(\frac{\partial \text{obj}(\mathbf{x})}{\mathbf{x}}) \]
Examples — Constraint: Brightness

- all:right
- DRV_C1:left
- DRV_C2:left
- DRV_C3:left
Examples — Constraint: Occlusion

DRV_C1: left

DRV_C2: left

DRV_C3: right
Examples — Constraint: Black Boxes

all:left

DRV_C1:right

DRV_C2:right

DRV_C3:right
## 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 $t = 0, 0.25, \text{ or } 0.75$
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

Add generated inputs to the training data and retrain.
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?