Deep Validation: Toward Detecting Real-world Corner Cases for Deep Neural Networks

Weibin Wu, Hui Xu, Sanqiang Zhong, Michael R. Lyu, and Irwin King

The Chinese University of Hong Kong
Outline

• Introduction
• Taxonomy and necessity
• Our work
  • Motivation
  • Method
  • Experiments
• Conclusions
Outline

• Introduction
• Taxonomy and necessity
• Our work
  • Motivation
  • Method
  • Experiments
• Conclusions
Introduction – background

• Popular deployment of deep learning models in dependability-critical systems
• Downside: high-performance, but black-box
• Concern: good score in test set -> safety in practice?
• Focus: deep learning vision classification systems (i.e., DNN based)
Introduction – definition

• Training samples: clean & representative samples
• **Real-world corner cases**: unusual **natural examples** that a model has never seen during training
• DNNs often misbehave under real-world corner cases

[Zhang et al., 2018]
Introduction – real-world evidence

- Failed Deep Neural Network (DNN) models
  - A Tesla car ignored a trailer
  - An autonomous Uber vehicle misclassified a pedestrian
- Challenge: how to ensure safety when encountering such unexpected errors?
Outline

• Introduction
• Taxonomy and necessity
• Our work
  • Motivation
  • Method
  • Experiments
• Conclusions
Taxonomy

Dependable DNN-based systems

Safety (accidental failures): real-world corner cases
- Testing & debugging
- Simulation & retraining
- Detection
- Our work

Security (intentional failures): adversarial samples
- Testing & debugging
- Synthesis & retraining
- Detection

Our work
Taxonomy

Dependable DNN-based systems

Safety (accidental failures): real-world corner cases
- Testing & debugging
- Simulation & retraining
- Detection
- Our work

Security (intentional failures): adversarial samples
- Testing & debugging
- Synthesis & retraining
- Detection

Our work
Corner case detection - necessity

• Model debugging
  • Retrain model with known corner cases or adversarial samples
  • Poor generalization to unseen anomalies

• Necessity for corner case detection
  • Innumerable variations of working conditions during real-world deployment
  • Indispensable safety tool: enable fail-safe action
Outline

• Introduction
• Taxonomy and necessity
• Our work
  • Motivation
  • Method
  • Experiments
• Conclusions
Our work – motivation

• Analogy between DNNs and traditional software
• DNNs
  • Each layer in DNNs performs relatively simple functions

![A typical DNN Model](image)
Our work – motivation

• Analogy between DNNs and traditional software
• A toy routine of an image processing program
  • Compute the pixel value of the complement of an image $x$
  • $x \leftarrow 1 - x$
  • To function normally, suppose pixel value $\in [0, 1]$
Our work – motivation

• DNNs
  • Each layer expect a valid input range that it has learned to handle during training
  • Cannot guarantee correctness when encountering too different samples (e.g. real-world corner cases)
Our work – motivation

• Analogy between DNNs and traditional software
• A toy routine of an image processing program
  • To improve dependability: data validation

/* invert pixel value */
\[ x \leftarrow 1 - x \]

pixel value \( \in [0, 1] \)?

Yes

No

data preprocessing
Our work – motivation

- DNNs
  - To improve dependability and safety: Deep Validation (our work)
  - Validate the input/states of each layer within DNNs
  - Spot real-world corner cases
Our work – method

- $f_i$: the output of the $i$-th hidden layer
- $d_i$: estimate the discrepancy of the state of layer $i$ to its normal range (i.e., input to next layer)
Our work – method

• Challenge:
  • Unlike traditional program, DNN’s function are learned automatically from training data, instead of programmed by developers

• Resort to training data
  • Naive memorization strategy: reject all samples other than ones occur during training
  • Our choice: one-class SVM
Our work – method

- One-class SVM
  - Only require normal samples
- $d_i$: signed distance to learned separating hyperplane
Our work – method

- $y'$: label prediction
- $d_i$: signed distance to learned separating hyperplane in layer $i \rightarrow$ single validator in layer $i$
- $\text{Joint}(d_1, \ldots, d_{L-1}) = \sum d_i \rightarrow$ joint validator
Experiments – setup

• Datasets and models
  • MNIST: seven-layer CNN
  • CIFAR-10: DenseNet
  • SVHN: seven-layer CNN
• Metric: ROC-AUC score
  • Reflect both true positive rate (TPR) and false positive rate (FPR)
  • Enable fair comparison
Experiments – setup

• Real-world corner case synthesis: metamorphic testing technique
Experiments – setup

• Complement transformation

Complement grey images

Complement color images
Experiments – detection results

- Discrepancy score distributions

- MNIST

- CIFAR-10

- SVHN
Experiments – detection results

• Single validator vs. joint validator (MNIST as an example)
• ROC-AUC results

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Transformation Method Used to Synthesize Corner Cases</th>
<th>Overall ROC-AUC Score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Layer No.</td>
<td>Rotation</td>
</tr>
<tr>
<td>Single Validator</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.8760</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.9200</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.9741</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.9740</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.9732</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.9659</td>
</tr>
<tr>
<td>Best Transformation-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>specific Single Validator</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Joint Validator</td>
<td></td>
<td>0.9891</td>
</tr>
</tbody>
</table>

• A joint validator often provides addition gains
Experiments – detection results

- Do adversarial sample detection methods (Feature Squeezing, Kernel Density Estimation) really capture the valid input range of DNNs?  - No!

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>Overall ROC-AUC Score (SCCs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>Deep Validation</td>
<td>0.9937</td>
</tr>
<tr>
<td></td>
<td>Feature Squeezing</td>
<td>0.9784</td>
</tr>
<tr>
<td></td>
<td>Kernel Density Estimation</td>
<td>0.1436</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>Deep Validation</td>
<td>0.9805</td>
</tr>
<tr>
<td></td>
<td>Feature Squeezing</td>
<td>0.8796</td>
</tr>
<tr>
<td></td>
<td>Kernel Density Estimation</td>
<td>0.1254</td>
</tr>
<tr>
<td>SVHN</td>
<td>Deep Validation</td>
<td>0.9506</td>
</tr>
<tr>
<td></td>
<td>Feature Squeezing</td>
<td>0.6870</td>
</tr>
<tr>
<td></td>
<td>Kernel Density Estimation</td>
<td>0.2543</td>
</tr>
</tbody>
</table>
Experiments – detection results

• Can Deep Validation also spot adversarial samples as invalid inputs? – Yes, with good promise in both reliability and security
  • SAE: only view successful adversarial samples as true positives
  • AE: view all adversarial samples as true positives

<table>
<thead>
<tr>
<th>Attack Method</th>
<th>FGSM</th>
<th>BIM</th>
<th>CW_∞</th>
<th>CW_2</th>
<th>CW_0</th>
<th>JSMA</th>
<th>Overall ROC-AUC Score</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Target Label</strong></td>
<td>-</td>
<td>-</td>
<td>Next</td>
<td>LL</td>
<td>Next</td>
<td>LL</td>
<td>Next</td>
</tr>
<tr>
<td><strong>Success Rate</strong></td>
<td>0.4300</td>
<td>0.9100</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td><strong>SAEs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deep Validation</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9992</td>
<td>0.9965</td>
<td>0.9347</td>
<td>0.9758</td>
<td>0.9329</td>
</tr>
<tr>
<td>Feature Squeezing</td>
<td>0.9970</td>
<td>0.9972</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9993</td>
<td>0.9996</td>
<td>0.9920</td>
</tr>
<tr>
<td><strong>AEs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deep Validation</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9992</td>
<td>0.9965</td>
<td>0.9347</td>
<td>0.9758</td>
<td>0.9329</td>
</tr>
<tr>
<td>Feature Squeezing</td>
<td>0.9441</td>
<td>0.9691</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9993</td>
<td>0.9996</td>
<td>0.9920</td>
</tr>
</tbody>
</table>
Experiments – detection results

• Detection rate on fail corner cases (FCC) and successful ones (SCC)
  • (MNIST, scale) as an example

Deep Validation
• Introduction
• Taxonomy and necessity
• Our work
  • Motivation
  • Method
  • Experiments
• Conclusions
Conclusions

• Deep Validation: The first framework to automatically validate internal inputs/states and identify error-inducing real-world corner cases for DNN.

• Conduct extensive experiments to confirm the promising performance of Deep Validation on eight different categories of corner cases.

• Detection methods tailored for intentional attacks can capture the valid input range of DNNs for real-world corner cases.
Thank You!

Q&A
References


Image credits

• Slide 4
  • Tim Pollard - https://www.carmagazine.co.uk/car-news/tech/autonomous-car-levels-different-driverless-technology-levelsexplained/
  • Shufti Pro - https://shuftipro.com/blogs/biometric-authentication/
  • Tonie Hansen - https://blogs.nvidia.com/blog/2016/02/09/deep-learning-3/

• Slide 6
  • ABC15 ARIZONA - https://www.telegraph.co.uk/technology/2018/03/20/ubers-self-driving-car-speeding-killed-pedestrian-police-reveal/

• Slide 63
  • Ag2gaeh - https://en.wikipedia.org/wiki/Ellipsoid#/media/File:Ellipsoide.svg