DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars

Yuchi Tian, Kexin Pei, Suman Jana, Baishakhi Ray

ICSE 2017

Table of Contents

Table of Contents

Introduction

Despite the tremendous progress, just like traditional software, DNN-based software, including the ones used for autonomous driving, often demonstrates incorrect/unexpected corner case behaviors.

The existing mechanisms for detecting such erroneous behaviors depend heavily on manual collection of labeled test data or ad hoc simulation.

At a conceptual level, the erroneous corner-case behaviors in DNN-based software are analogous to logic bugs in traditional software.

Introduction

The internals of traditional software and new DNN-based software are fundamentally different.

- DNN-based software automatically learns its logic from data with minimal human guidance. (Manually written in traditional software)
- DNNs use weights for edges between different neurons and nonlinear activation functions to express the logic of the program. (Control flow statements used in traditional software)

Introduction

Key challenges behind automated testing of DNNs below.

- Maximizing branch/code coverage is not very useful for DNN-based software as the logic is not encoded using control flow.
- Finding inputs that will result in high model (e.g., finite state machines) coverage in a DNN is significantly more challenging due to the non-linearity of the functions modeled by DNNs.
- The Satisfiability Modulo Theory (SMT) solvers do not scale well to real-world-sized DNNs.
- Manually creating specifications for complex DNN systems like autonomous cars is infeasible as the logic is too complex

Introduction

- They leverage the notion of neuron coverage (i.e., the number of neurons activated by a set of test inputs) to systematically explore different parts of the DNN logic.
- They demonstrate that different image transformations that mimic real-world differences in driving conditions result in activation of different sets of neurons in the self-driving car DNNs.
- They show that by combining these image transformations, the neuron coverage can be increased by 100% on average compared to the coverage achieved by manual test inputs.
- They use transformation-specific metamorphic relations between multiple executions of the tested DNN (e.g., a car should behave similarly under different lighting conditions).

Table of Contents

Background

The key component of an autonomous vehicle is the perception module controlled by the underlying Deep Neural Network (DNN)
Inputs:

- Camera,
- Light detection and ranging sensor (LiDAR)
- IR (infrared) sensor etc.

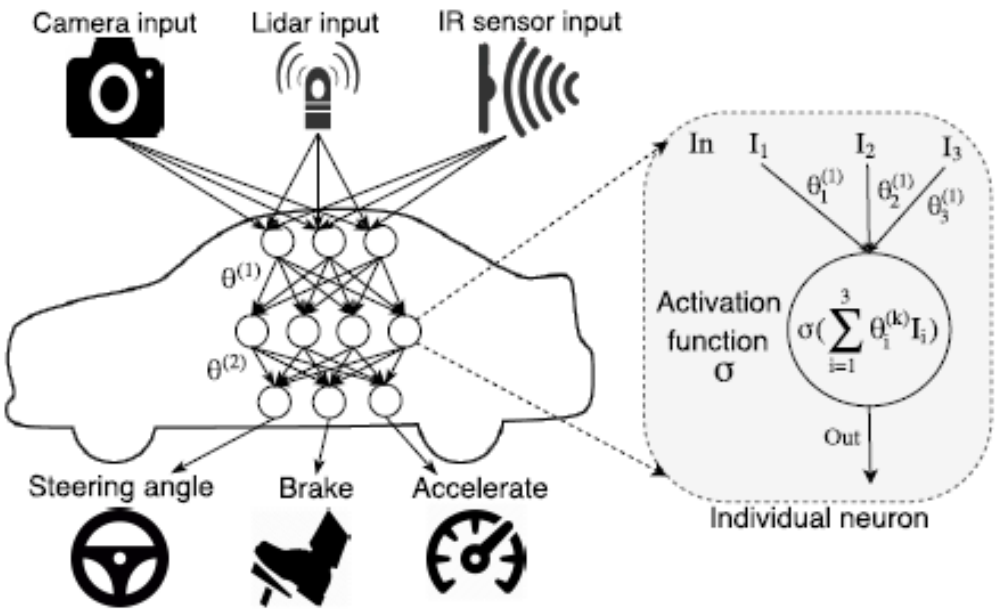
Outputs:

- Steering angle
- Braking
- Acceleration etc

Background

Once trained, a DNN can be used for prediction without any further changes to the weights on edges. For example, an autonomous car DNN can predict the steering angle based on input images.

Figure: The DNN shown here essentially models the function $\sigma(\theta^{(2)}.\sigma(\theta^{(1)})$ where θ represent the weights of the edges and σ is the activation function.



Example: `playground.tensorflow.com`

Background

Most DNNs used in autonomous vehicles can be categorized into two types:

- Feed-forward Convolutional Neural Network (CNN)
- Recurrent neural network (RNN).

Background
CNN

The most significant difference between a CNN and a fully connected DNN is the presence of a convolution layer.
Convolution layers have two major benefits:

- They greatly reduce the number of trainable weights by allowing sharing of weights among multiple connections and thus significantly cut down the training time.
- The application of convolution kernels is a natural fit for image recognition as it resembles the human visual system (layer-wise representation of visual input)

Background

RNN

RNNs, unlike CNNs, allow loops in the network.

Specifically, the output of each layer is not only fed to the following layer but also flow back to the previous layer.

Therefore, the presence of loops can potentially help in modelling stateful sequential inputs such as text/video sequences with temporal dependence.

Figure: Simplified CNN and RNN architectures.

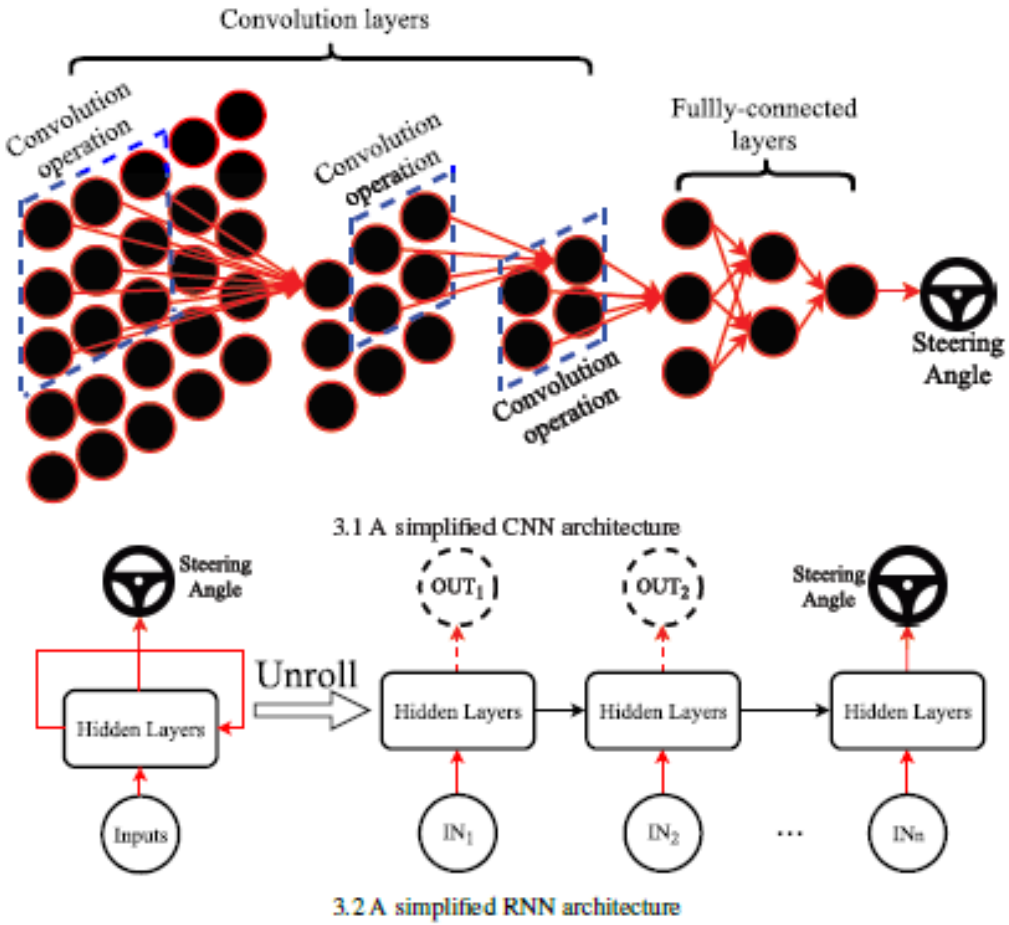


Table of Contents

Method

Questions to answer for developing an automated testing methodology for DNN driven autonomous cars:

- How do we systematically explore the input-output spaces of an autonomous car DNN?
- How can we synthesize realistic inputs to automate such exploration?
- How can we optimize the exploration process?
- How do we automatically create a test oracle that can detect erroneous behaviors without detailed manual specifications?

Method

Systematic Testing with Neuron coverage

The input-output space of a complex system like an autonomous vehicle is too large.

Therefore, they devise the space into different equivalence classes and try to cover all equivalence classes by picking one sample from each of them.

They leverage neuron coverage as a mechanism for partitioning the input space (all inputs that have similar neuron coverage are part of the same equivalence class).

$$NeuronCoverage = \frac{|ActivatedNeurons|}{|TotalNeurons|}$$

Method

Increasing Coverage with Synthetic Images

DeepTest focuses on generating realistic synthetic images by applying image transformations on seed images.

we investigate nine different realistic image transformations (**changing brightness**, **changing contrast**, **translation**, **scaling**, **horizontal shearing**, **rotation**, **blurring**, **fog effect**, and **rain effect**).

These transformations can be classified into three groups:

- linear
- affine
- convolutional

Method

Increasing Coverage with Synthetic Images

Linear transformations: Adjusting *brightness* and *contrast*.

Affine transformations: *Translation*, *scaling*, *horizontal shearing*, and *rotation*.

Convolutional transformations: *Blurring* and adding *fog/rain effects*.

Method

Combining Transformations to Increase Coverage

As the image transformations individually increase neuron coverage, one obvious question is whether they can be combined to further increase the neuron coverage.

They provide a neuron-coverage-guided greedy search technique for efficiently finding combinations of image transformations that result in higher coverage.

Method

Combining Transformations to Increase Coverage

Figure: Greedy search for combining image transformations to increase neuron coverage

```
Input    : Transformations T, Seed images I
Output   : Synthetically generated test images
Variable : S: stack for storing newly generated images
           Tqueue: transformation queue
1  ---------------------------------------------------------------------
2  Push all seed imgs ∈ I to Stack S
3  genTests = ϕ
4  while S is not empty do
5      img = S.pop()
6      Tqueue = ϕ
7      numFailedTries = 0
8      while numFailedTries ≤ maxFailedTries do
9          if Tqueue is not empty then
10             T1 = Tqueue.dequeue()
11         else
12             Randomly pick transformation T1 from T
13         end
14         Randomly pick parameter P1 for T1
15         Randomly pick transformation T2 from T
16         Randomly pick parameter P2 for T2
17         newImage = ApplyTransforms(image, T1, P1, T2, P2)
18         if covInc(newimage) then
19             Tqueue.enqueue(T1)
20             Tqueue.enqueue(T2)
21             UpdateCoverage()
22             genTest = genTests ∪ newimage S.push(newImage)
23         else
24             numFailedTries = numFailedTries + 1
25         end
26     end
27 end
```

Method

Creating a Test Oracle with Metamorphic Relations

For example, the autonomous car's steering angle should not change significantly for the same image under any lighting/weather conditions, blurring etc.

If a DNN model infers a steering angle θ_o for an input seed image I_0o and a steering angle θ_t for a new synthetic image I_t , which is generated by applying the transformation t on I_o , one may define a simple metamorphic relation where θ_o and θ_t are identical.

However, a car can safely tolerate small variations.

We observe that the set of outputs predicted by a DNN model for the **original images**, say $\theta_{o1}, \theta_{o2}, \ldots, \theta_{on}$, in practice, result in a small but non-trivial number of errors w.r.t. their respective **manual labels** $\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_n$.

Such errors are usually measured using Mean Squared Error (MSE), where $MSE_{orig} = \frac{1}{n}\sum_{i=1}^{n}(\theta_{oi} - \hat{\theta}_i)^2$

Thus: $(\hat{\theta}_i - \theta_{ti})^2 < \lambda MSE_{orig}$

Table of Contents

Implementation

Figure: Details of DNNs used to evaluate DeepTest

Model	Sub-Model	No. of Neurons	Reported MSE	Our MSE
Chauffeur	CNN LSTM	1427 513	0.06	0.06
Rambo	S1(CNN) S2(CNN) S3(CNN)	1625 3801 13473	0.06	0.05
Epoch	CNN	2500	0.08	0.10

† dataset HMB_3.bag [17]

Figure: Transformations and parameters used by DeepTest for generating synthetic images.

Transformations		Parameters	Parameter ranges
Translation		(t_x, t_y)	(10, 10) to (100, 100) step (10, 10)
Scale		(s_x, s_y)	(1.5, 1.5) to (6, 6) step (0.5, 0.5)
Shear		(s_x, s_y)	(−1.0, 0) to (−0.1, 0) step (0.1, 0)
Rotation		q (degree)	3 to 30 with step 3
Contrast		α (gain)	1.2 to 3.0 with step 0.2
Brightness		β (bias)	10 to 100 with step 10
Blur	Averaging	kernel size	3 × 3, 4 × 4, 5 × 5, 6 × 6
	Gaussian	kernel size	3 × 3, 5 × 5, 7 × 7 , 3 × 3
	Median	aperture linear size	3, 5
	Bilateral Filter	diameter, sigmaColor, sigmaSpace	9, 75, 75

Table of Contents

RQ1

Do different input-output pairs result in different neuron coverage?

Figure: Relation between neuron coverage and test output

Model	Sub-Model	Steering Angle	Steering Direction	
		Spearman Correlation	Wilcoxon Test	Effect size (Cohen's d)
Chauffeur	Overall	-0.10 (***)	left (+ve) > right (-ve) (***)	negligible
	CNN	0.28 (***)	left (+ve) < right (-ve) (***)	negligible
	LSTM	-0.10 (***)	left (+ve) > right (-ve) (***)	negligible
Rambo	Overall	-0.11 (***)	left (+ve) < right (-ve) (***)	negligible
	S1	-0.19 (***)	left (+ve) < right (-ve) (***)	large
	S2	0.10 (***)	not significant	negligible
	S3	-0.11 (***)	not significant	negligible
Epoch	N/A	0.78 (***)	left (+ve) < right (-ve) (***)	small

*** indicates statistical significance with p-value $< 2.2 * 10^{-16}$

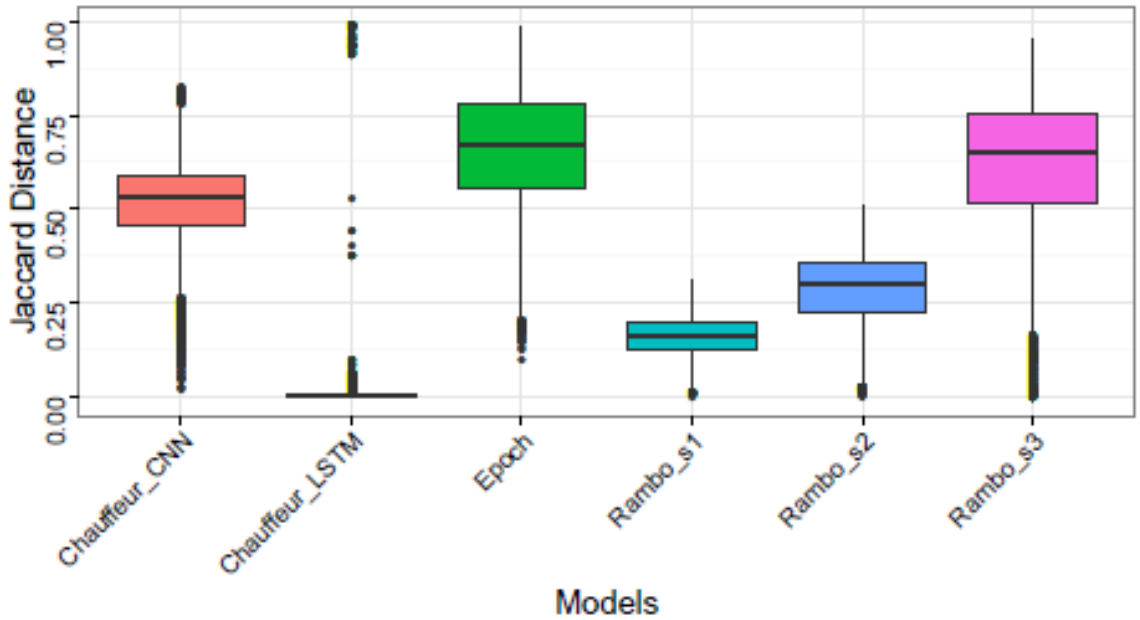
Result 1

Neuron coverage is strongly associated with input output diversity and can be used to systematic test generation.

RQ2

Do different realistic image transformations activate different neurons?

Figure: Difference in neuron coverage caused by different image transformations



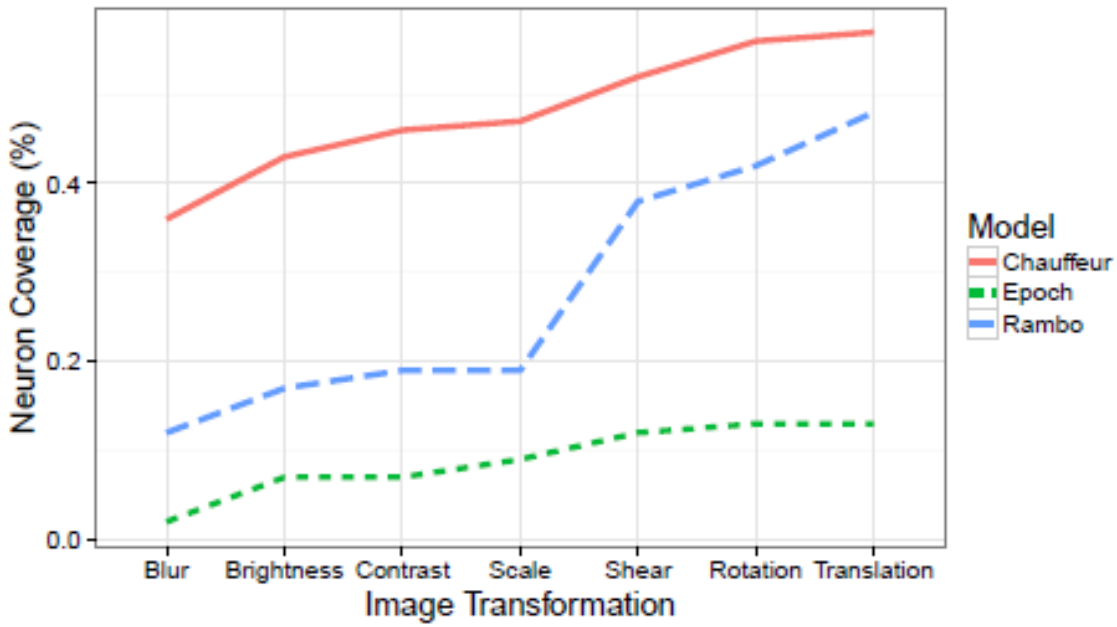
Total of 70,000 new synthetic images.

Jaccard distance: 1- $\frac{|N_1 \cap N_2|}{|N_1 \cup N_2|}$

Results

RQ2

Figure: Average cumulative neuron coverage per input image



Result 2

Different image transformations tend to activate different sets of neurons.

Results

RQ3

RQ3

Can neuron coverage be further increased by combining different image transformations?

They perform this experiment by measuring neuron coverage in two different settings:

- applying a set of transformations
- combining transformations using coverage-guided search.

Results

RQ3

Figure: Neuron coverage achieved by cumulative and guided transformations applied to 100 seed images

Model	Baseline	Cumulative Transformation	Guided Generation	% increase of guided *w.r.t.* Baseline	% increase of guided *w.r.t.* Cumulative
Chauffeur-CNN	658 (46%)	1,065 (75%)	1,250 (88%)	90%	17%
Epoch	621 (25%)	1034 (41%)	1,266 (51%)	104%	22%
Rambo-S1	710 (44%)	929 (57%)	1,043 (64%)	47%	12%
Rambo-S2	1,146 (30%)	2,210 (58%)	2,676 (70%)	134%	21%
Rambo-S3	13,008 (97%)	13,080 (97%)	13,150 (98%)	1.1%	0.5%

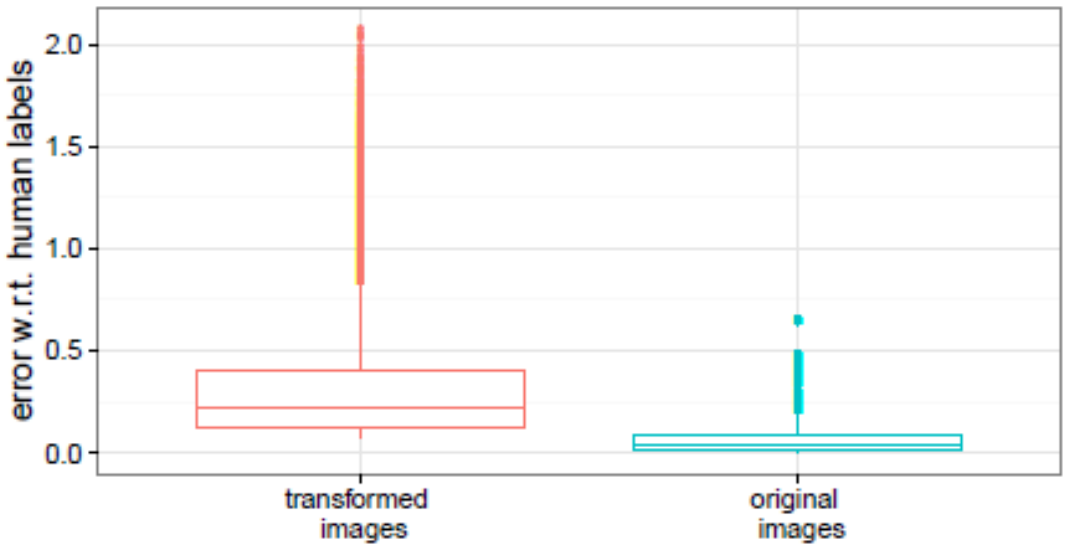
Result 3

By systematically combining different image transformations, especially with a guided approach, neuron coverage can be improved by around 100% w.r.t. the coverage achieved by the original seed images.

RQ4

Can we automatically detect erroneous behaviors using metamorphic relations?

Figure: Deviations from the human labels for images that violate the metamorphic relation is higher compared to the deviations for original images.



This figure shows that these synthetic images have a high chance to show

Results

RQ4

$$|MSE_{(trans,param)} - MSE_{org}| \leq \epsilon$$

Figure: Number of erroneous behaviors reported by DeepTest across all tested models at different thresholds

λ (see Eqn. 2)	Simple Tranformation ϵ (see Eqn. 3) 0.01	0.02	0.03	0.04	0.05	Composite Transformation Fog	Rain	Guided Search
1	15666	18520	23391	24952	29649	9018	6133	1148
2	4066	5033	6778	7362	9259	6503	2650	1026
3	1396	1741	2414	2627	3376	5452	1483	930
4	501	642	965	1064	4884	4884	997	872
5	95	171	330	382	641	4448	741	820
6	49	85	185	210	359	4063	516	764
7	13	24	89	105	189	3732	287	721
8	3	5	34	45	103	3391	174	668
9	0	1	12	19	56	3070	111	637
10	0	0	3	5	23	2801	63	597

Results

RQ4

Figure: Number of unique erroneous behaviors reported by DeepTest for different models with $\lambda = 5$

Transformation	Chauffeur	Epoch	Rambo
Simple Transformation			
Blur	3	27	11
Brightness	97	32	15
Contrast	31	12	-
Rotation	-	13	-
Scale	-	10	-
Shear	-	-	23
Translation	21	35	-
Composite Transformation			
Rain	650	64	27
Fog	201	135	4112
Guided	89	65	666

Results

RQ4

Figure: Some sample false positives produced by DeepTest for $\lambda = 5$ and $\epsilon = 0.03$

Model	Simple Transformation	Guided	Rain	Fog	Total
Epoch	14	0	0	0	14
Chauffeur	5	3	12	6	26
Rambo	8	43	11	28	90
Total	27	46	23	34	130

original translation(50,50), epoch original shear(0.4), rambo

Result 4

With neuron guided synthesized images, DeepTest successfully detects more than 1,000 erroneous behavior as predicted by the three models with low false positives

Results

RQ5

RQ5

Can retraining DNNs with synthetic images improve accuracy?

Result 5

Accuracy of a DNN can be improved by up to 46% by retraining the DNN with synthetic data generated by DeepTest.