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

Yuchi Tian, Kexin Pei, Suman Jana, and

Baishakhi Ray

ICSE 2018

Web Security & Privacy Lab

KAIST

About paper

- Problem
  - How to systematically detect erroneous behaviors in DNN-driven vehicle.
- Contribution
  - Present a systematic technique to automatically synthetize test cases that maximize neuron coverage.
  - Show that synthetic input generated can be used to retrain the DNN
- Result
  - DeepTest found thousands of erroneous behaviors from the three top performing DNNs in the Udacity self-driving car challenge

Motivations

Ford

GENERAL MOTORS

Google

Motivation

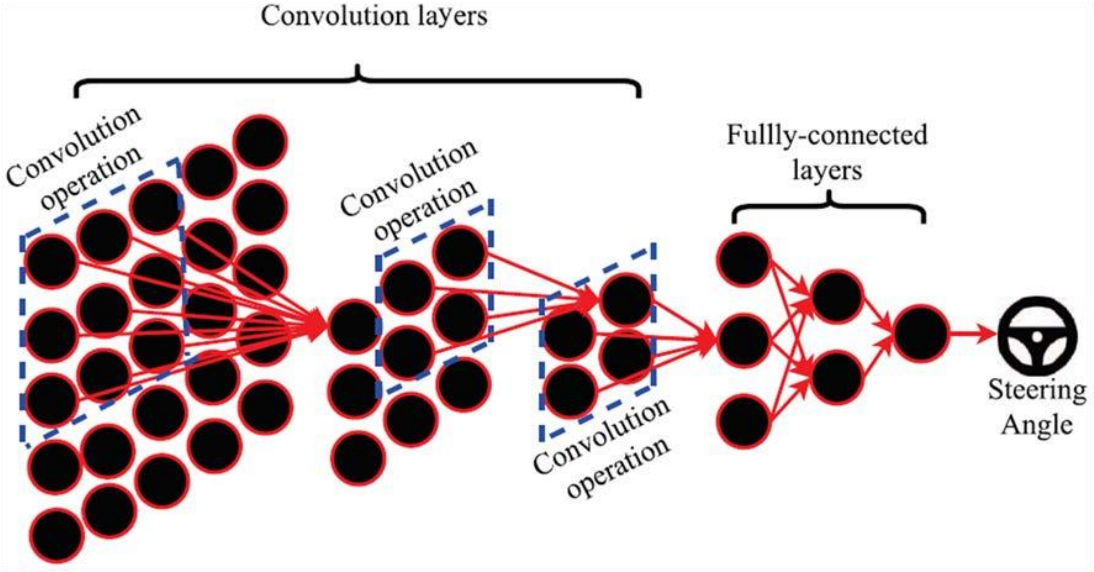
- Unexpected behaviors
- Current detecting mechanisms depend heavily on manual collection of labeled data

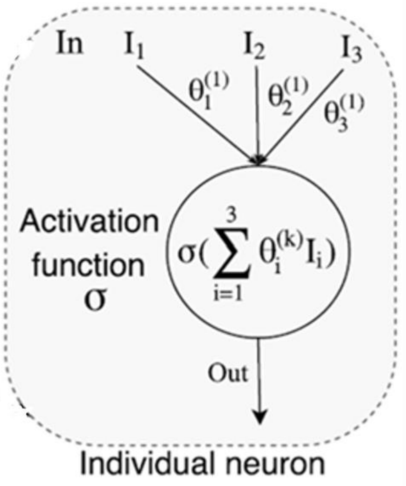
Motivation

- Challenges to Build a DNN testing software
  - DNN learn its logic from a large amount of data with minimal human intervention.
  - Express its logic in weight and activation function than control flow.
  - Code coverage not an efficient way to evaluate the quality o f a testing software

Background

- DNNS :
  - CNN

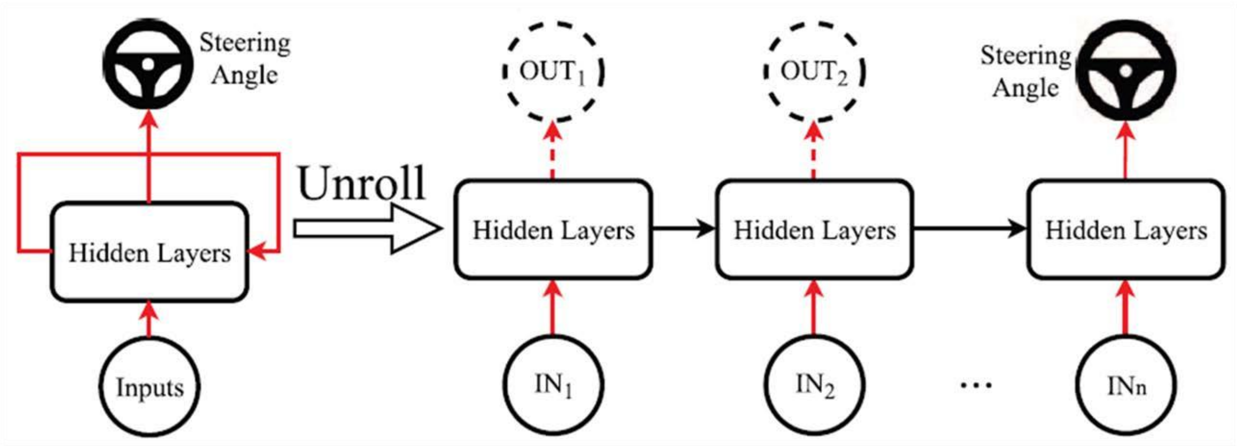




- High accuracy on the image recognition
- Each filter can abstract features from the previous layer

Background

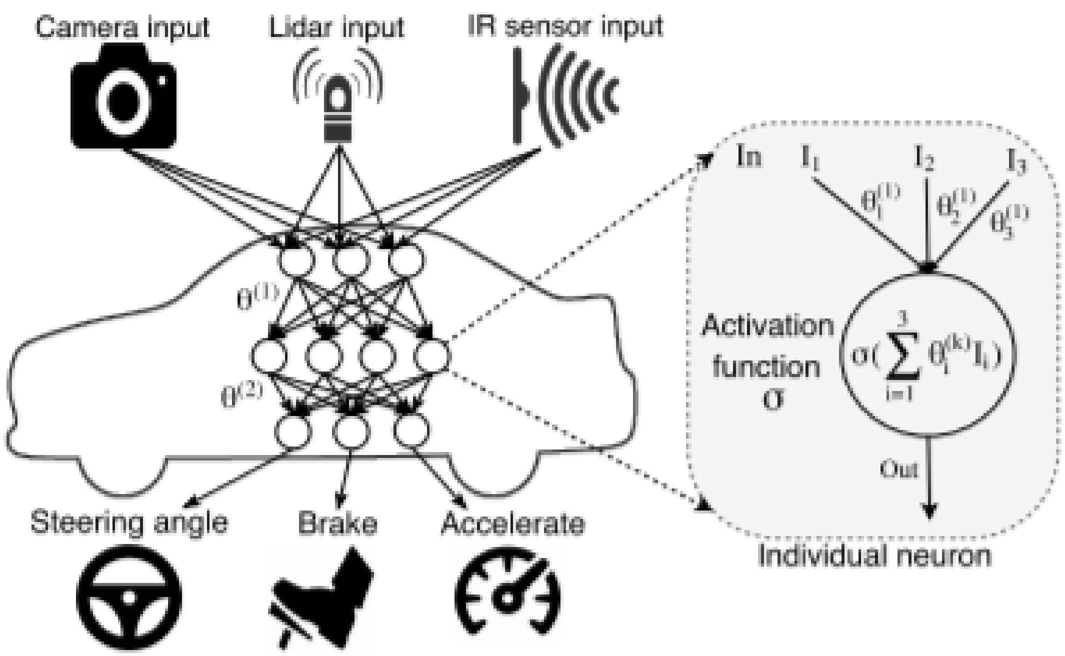
- DNNS :
  - Recurrent Neural Networks



- Current prediction is fed to the next predictions
- Consider previous inputs to predict current input

Background

- DNNS in Autonomous Vehicles :

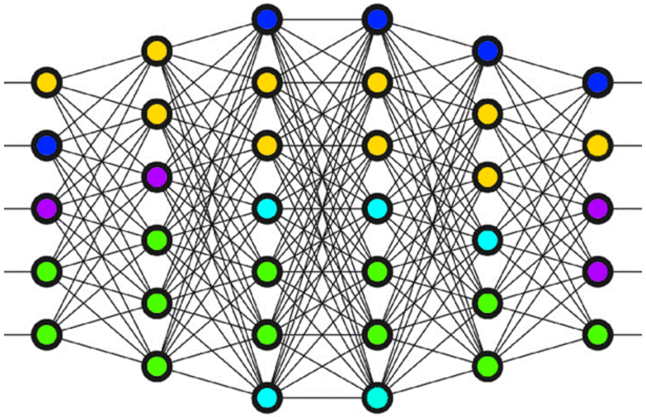


Methodology

- 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?

How do we systematically explore the input-output spaces of an autonomous car DNN?

- Input-output space too large to be able to fully explore it
- Assume that inputs that have similar neuron coverage behave similarly
- Partition the space into equivalence classes and cover each class by picking one sample from it.



How do we systematically explore the input-output spaces of an autonomous car DNN?

- When can we consider that a neuron is activated ?
- For fully connected layers, if neuron's output is above the activation threshold the neuron is considered as activated.
- For CNNs, compute the average of the output feature map to convert multidimensional output of a neuron into a scalar and compare it with the neuron activation threshold of 0,2
- For RNNs, Treat each neuron in the unrolled layers as a separate individual neuron for the purpose of neuron coverage computation

How can we synthesize realistic inputs to automate such exploration?

- Unlike DeepXplore, DeepTest synthesizes inputs by applying image transformations that mimic real-world phenomena to seed images.

DeepXplore :

all:right

DRV_C1:left

DeepTest :

1.1 original

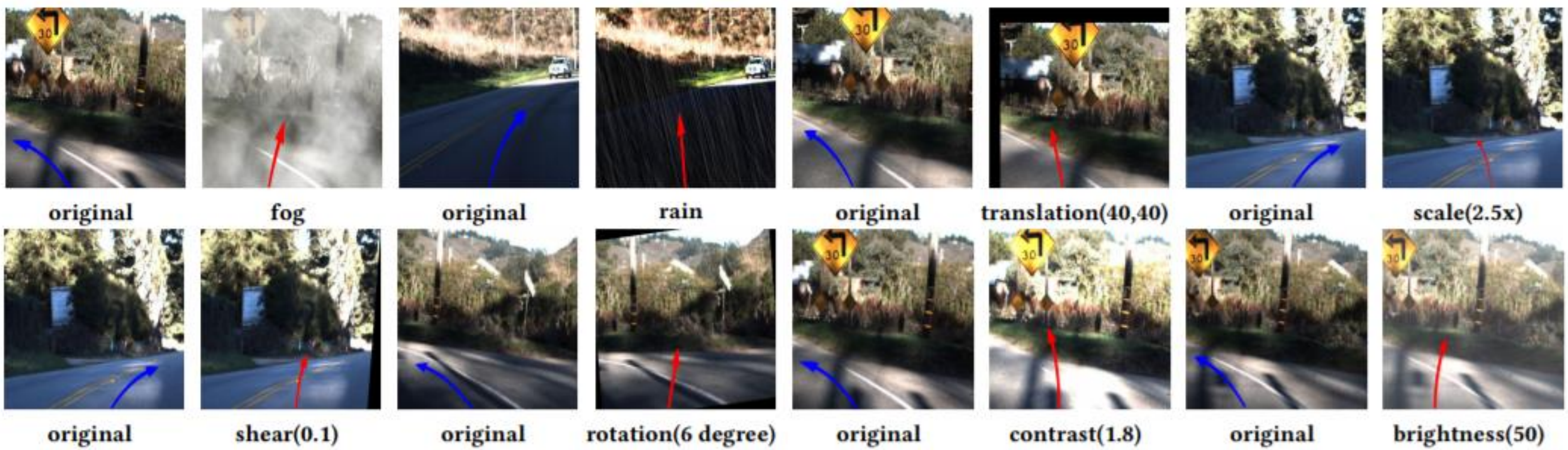
1.2 with added rain

Web Security & Privacy Lab

KAIST

How can we synthesize realistic inputs to automate such exploration?

- 3 different type of transformation
  - Affine (translation, scale, shear, rotation)
  - Linear (contrast, brightness)
  - Convolutional (Fog, rain, blurring)



How can we optimize the exploration process?

Algorithm 1: Greedy search for combining image tranformations 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

Push all seed imgs ∈ I to Stack S
genTests = ϕ
while S is not empty do
    img = S.pop()
    Tqueue = ϕ
    numFailedTries = 0
    while numFailedTries ≤ maxFailedTries do
        if Tqueue is not empty then
            T1 = Tqueue.dequeue()
        else
            Randomly pick transformation T1 from T
        end
        Randomly pick parameter P1 for T1
        Randomly pick transformation T2 from T
        Randomly pick parameter P2 for T2
        newImage = ApplyTransforms(image, T1, P1, T2, P2)
        if covInc(newimage) then
            Tqueue.enqueue(T1)
            Tqueue.enqueue(T2)
            UpdateCoverage()
            genTest = genTests ∪ newimage S.push(newImage)
        else
            numFailedTries = numFailedTries + 1
        end
    end
end
return genTests
```

- Use Combination of transformation

How can we optimize the exploration process?

Algorithm 1: Greedy search for combining image tranformations 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
──────────────────────────────────────────────
Push all seed imgs ∈ I to Stack S
genTests = ϕ
while S is not empty do
    img = S.pop()
    Tqueue = ϕ
    numFailedTries = 0
    while numFailedTries ≤ maxFailedTries do
        if Tqueue is not empty then
            T1 = Tqueue.dequeue()
        else
            Randomly pick transformation T1 from T
        end
        Randomly pick parameter P1 for T1
        Randomly pick transformation T2 from T
        Randomly pick parameter P2 for T2
        newImage = ApplyTransforms(image, T1, P1, T2, P2)
        if covInc(newimage) then
            Tqueue.enqueue(T1)
            Tqueue.enqueue(T2)
            UpdateCoverage()
            genTest = genTests ∪ newimage S.push(newImage)
        else
            numFailedTries = numFailedTries + 1
        end
    end
end
return genTests
```

- Keep track of the transformations that successfully increase neuron coverage for a given image and prioritize them.

How can we optimize the exploration process?

Algorithm 1: Greedy search for combining image tranformations 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
-----------------------------------------------------------
Push all seed imgs ∈ I to Stack S
genTests = ϕ
while S is not empty do
    img = S.pop()
    Tqueue = ϕ
    numFailedTries = 0
    while numFailedTries ≤ maxFailedTries do
        if Tqueue is not empty then
            T1 = Tqueue.dequeue()
        else
            Randomly pick transformation T1 from T
        end
        Randomly pick parameter P1 for T1
        Randomly pick transformation T2 from T
        Randomly pick parameter P2 for T2
        newImage = ApplyTransforms(image, T1, P1, T2, P2)
        if covInc(newimage) then
            Tqueue.enqueue(T1)
            Tqueue.enqueue(T2)
            UpdateCoverage()
            genTest = genTests ∪ newimage S.push(newImage)
        else
            numFailedTries = numFailedTries + 1
        end
    end
end
return genTests
```

	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

How do we automatically create a test oracle that can detect erroneous behaviors without detailed manual specifications?

- Creating a test oracle :
  - How to decide whether the output of the model is erroneous ?
  - The authors leverage metamorphic relations between the car behaviors across different synthetic images
  - There is no single correct output for a given image
  - The steering angles deduced for synthetic and original images should be identical.

1.1 original   1.2 with added rain

Web Security & Privacy Lab

KAIST

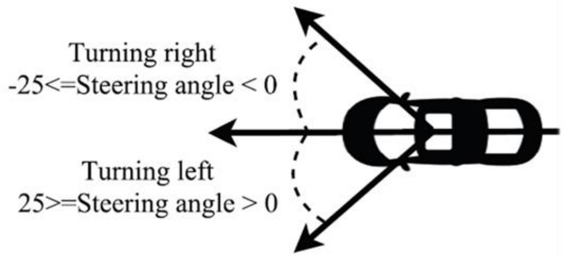
How do we automatically create a test oracle that can detect erroneous behaviors without detailed manual specifications?

- There is no single correct output for a given image
- Propose to use metamorphic relation :
  - even if we do not know the correct output of a single input, we might still know the relations between the outputs of multiple inputs
- We need to balance the metamorphic relation :
  - Tight metamorphic relations : false positives
  - Permissive metamorphic relations : false negatives
- Need to strike a balance between two extremes
  - Metamorphic relation: $(\hat{\theta}_i - \theta_{ti})^2 \leq \lambda\, MSE_{orig}$
  - The set of outputs predicted by a model: : $\{\theta_{o1}, \theta_{o2}, \dots, \theta_{on}\}$
  - The set of manual labels: $\{\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_n\}$
  - $MSE_{orig} = \frac{1}{n}\sum_{i=1}^{n}(\hat{\theta}_i - \theta_{oi})^2$

Implementation

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 [16]

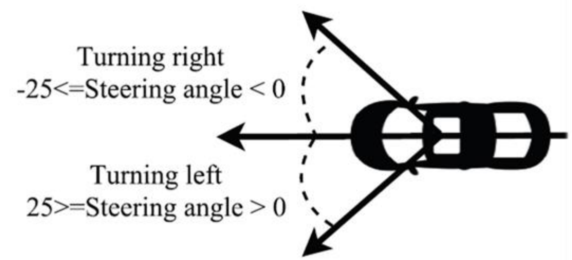


- 3 DNNs from the Udacity self-driving Challenge :
  - Rambo : 2nd CNN
  - Chauffeur : 3rd CNN + LSTM (Long short-term memory)
  - Epoch : 6th CNN
- Steering angle scaled from -1 to 1

Implementation

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 [16]



- 3 DNNs from the Udacity self-driving Challenge :
  - Rambo : 2nd CNN
  - Chauffeur : 3rd CNN + LSTM (Long short-term memory)
  - Epoch : 6th CNN
- Steering angle scaled from -1 to 1

Why ?

Web Security & Privacy Lab

KAIST

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

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 $< 2.2 * 10^{-16}$

- Correlation between neuron coverage and steering angle ?
  - Spearman rank correlation ranges from -1 to 1
  - p-value : probability that the result is caused be chance
  - Authors claim that there is a correlation

Web Security & Privacy Lab

KAIST

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

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 $< 2.2 * 10^{-16}$

- Correlation between neuron coverage and steering direction ?

  The neuron coverage varies with steering direction with statistical significance for all the three overall models

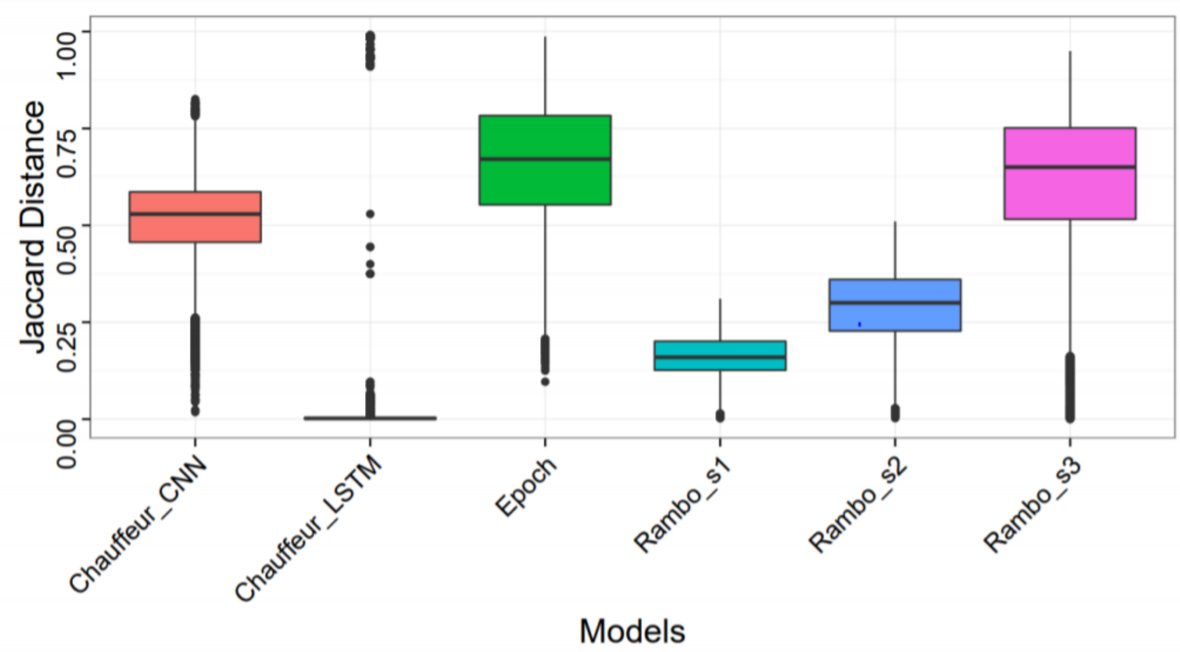
Result - Do different realistic image transformations activate different neurons?

- Pick 1000 pictures to synthetize 70,000 new ones by applying different transformation.
- Evaluation :

$$Jaccard\ distance = 1 - \frac{|N1 \cap N2|}{|N1 \cup N2|}$$

T1 and T2 : two different transformation on a same picture
N1 : neuron activated with the input T1
N2 : neuron activated with the input T2

Result - Do different realistic image transformations activate different neurons?



4.1 Difference in neuron coverage caused by different image transformations

- Conclusion : different image transformations tend to activate different sets of neurons

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

- Cumulative transformations without any guidance
  - Generated a total of 7,000 images from 100 seed images by applying 7 transformations with varying parameters.
  - Compare neuron coverage of synthetized pictures and original ones
- Guided transformation
  - Generated 254, 221, and 864 images from 100 seed images for Chauffeur, Epoch and Rambo. (Why different number ?)

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%

Web Security & Privacy Lab

KAIST

Result - Can we automatically detect erroneous behaviors using metamorphic relations?

$\boldsymbol{I_{org}}$: original images

$\boldsymbol{I_{err}}$: transformed images whose outputs violate the metamorphic relation

MSE of $\boldsymbol{I_{org}}$ = **0.035,** MSE of $\boldsymbol{I_{err}}$ = **0.41**

- Can we say that larger MSEs indicate erroneous behaviors ?
  - Not all violations can be considered buggy. The correct steering angle can vary widely based on the contents of the transformed image

Original image

Translation(50, 50), Epoch

Original image

Shear(0.4), Rambo

Web Security & Privacy Lab

KAIST

Result - Can we automatically detect erroneous behaviors using metamorphic relations?

- We have to report bugs only for the transformations where the correct output should not deviate much from the labels of the corresponding seed images
  - Ex : fog, rain, …

1.1 original

1.2 with added rain

Result - Can we automatically detect erroneous behaviors using metamorphic relations?

- $\left|\boldsymbol{MSE}_{(\boldsymbol{trans},\,\boldsymbol{param})} - \boldsymbol{MSE}_{\boldsymbol{org}}\right| \leq \boldsymbol{\epsilon}$
- Only consider the transformations that obey the equation for counting erroneous behaviors
- This Filter cannot be applied to fog and rain.
- Guided search is not compatible with the filter ether as we cannot filter out a single transformation

Result - Can we automatically detect erroneous behaviors using metamorphic relations?

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

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	$\boxed{4112}$
Guided	89	65	666

Result - Can we automatically detect erroneous behaviors using metamorphic relations?

Manual verification False Positive :

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

Result - Can retraining DNNs with synthetic images improve accuracy?

- Retrain Epoch model with rain and fog samples
  - Why only Epoch ?
  - Why only this 2 transformations ?
- Evaluate the original and retrained model with both synthetized and original pictures

Test set	Original MSE	Retrained MSE
original images	0.10	0.09
with fog	0.18	0.10
with rain	0.13	0.07

Conclusion

- Pros :
  - Present DeepTest a testing tools for DNN-driven vehicles.
    - Use neuron coverage to evaluate the testing quality.
    - Synthetize realistic output.
  - Show that synthetic input generated can be used to retrain the DNN

- Cons :
  - Don't maximize neuron coverage for LSTM.
  - Shuffle dataset to train LSTM, Is that a good way ?
  - They did not test the obstruction of the camera with drops or dust

Questions