DeepTest

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

a paper of Anh Nguyen, Jason Yosinki and Jeff Clune

presented by Nils Wenzler

Problem

DNNs show incorrect and unexpected corner-case behaviours

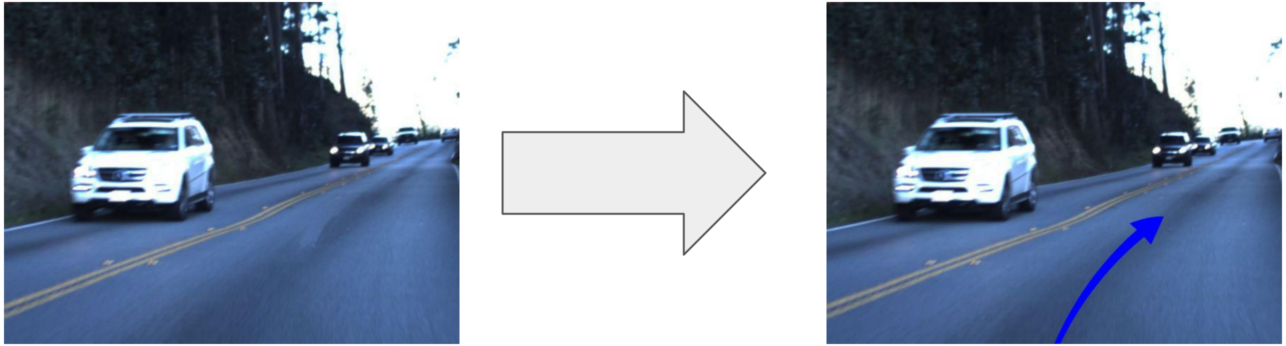
These corner-case behaviours can be potential lethal

How can we test the behaviour of a DNN in such corner-cases to verify their correctness?

Setting

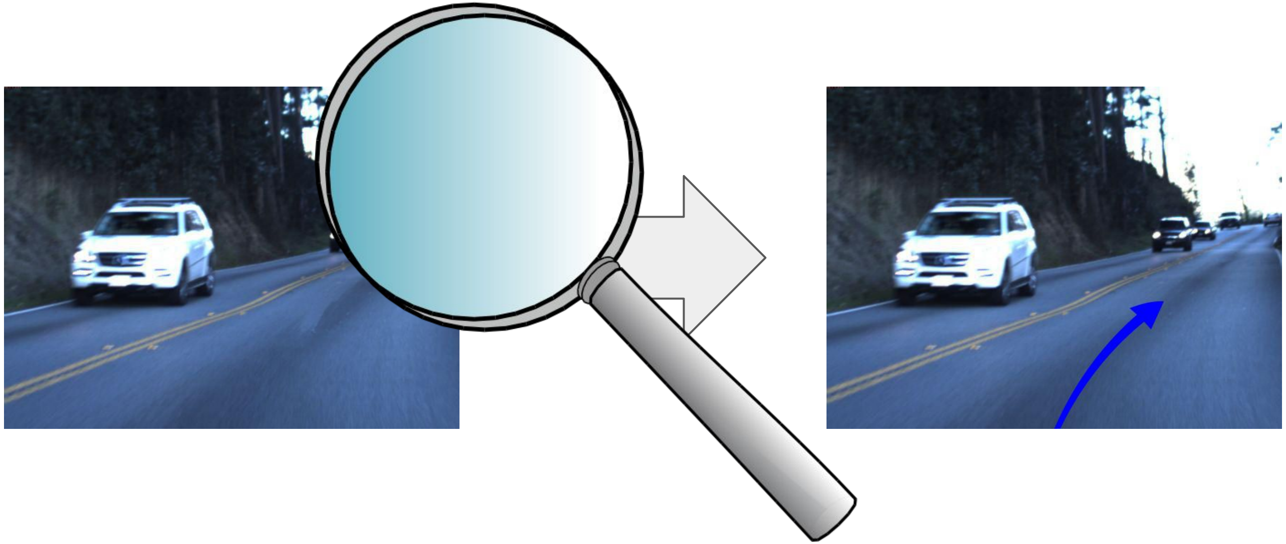
Udacity self-driving car challenge:

Build and train a neural network that given an input image predicts a corresponding steering angle and direction



Empirical example

Test three of the top scoring models of the Udacity self-driving car challenge for corner-case behaviours.



Classical Solution

Test software with

1. **automatically generated test cases**
2. that **optimize**
3. a specific **coverage criterion** (e.g. branch coverage)

to show that all major behaviour patterns of the software perform as expected.

New Solution

Test deep neural networks with

1. **automatically generated test cases**
2. that **optimize**
3. a specific **coverage criterion**

to show that all major behaviour patterns of the software perform as expected.

New Solution

Test deep neural networks with

1. **automatically generated test cases**
   a. how to automatically generate new and realisic inputs
   b. how to automatically find fitting labels for these inputs
2. that **optimize**
   a. how to choose a good set of test cases although dealing with non-linearity and non-convexity
3. a specific **coverage criterion**
   a. how to measure “behaviour coverage” for a DNN

to show that all major behaviour patterns of the software perform as expected.

New Solution

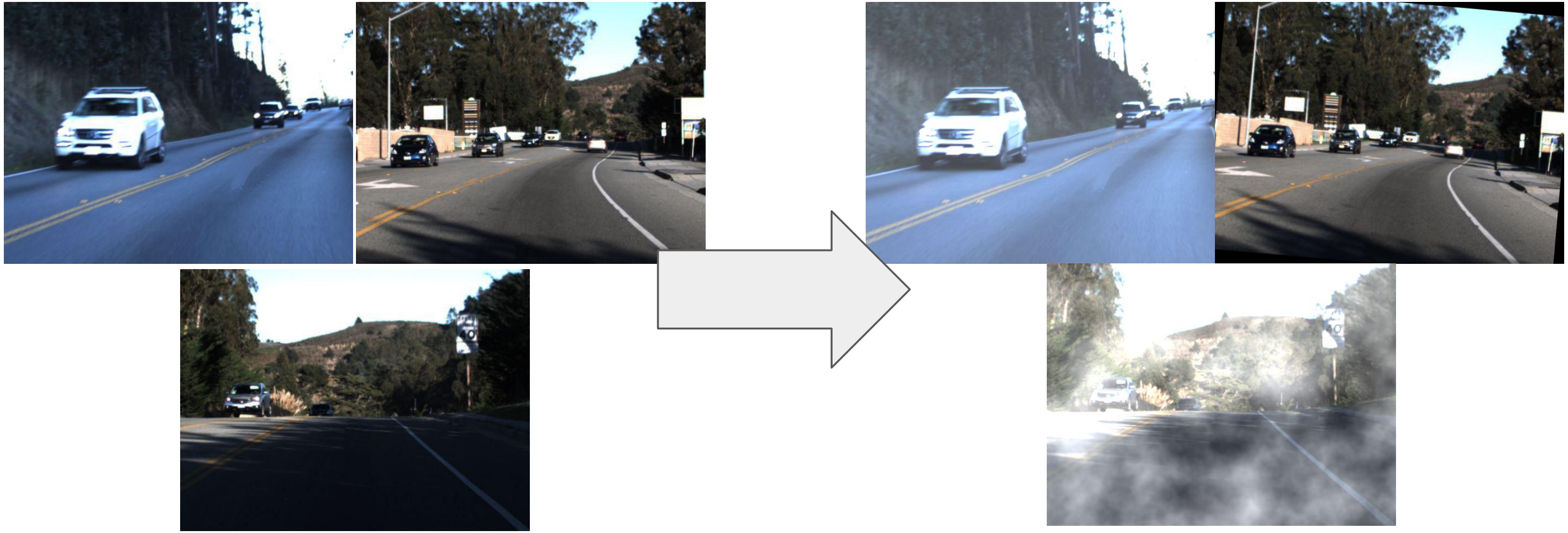
Test deep neural networks with

1. **automatically generated test cases**
   a. how to automatically generate new inputs
   b. how to automatically find fitting labels for these inputs
2. that **optimize**
   a. how to choose a good set of test cases
3. a specific **coverage criterion**
   a. how to measure “behaviour coverage” for a DNN

to show that all major behaviour patterns of the software perform as expected.

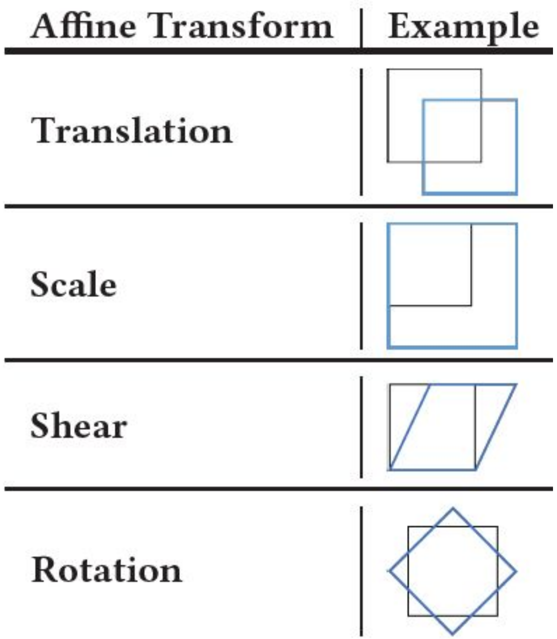
Input generation

Use existing training data and augment:



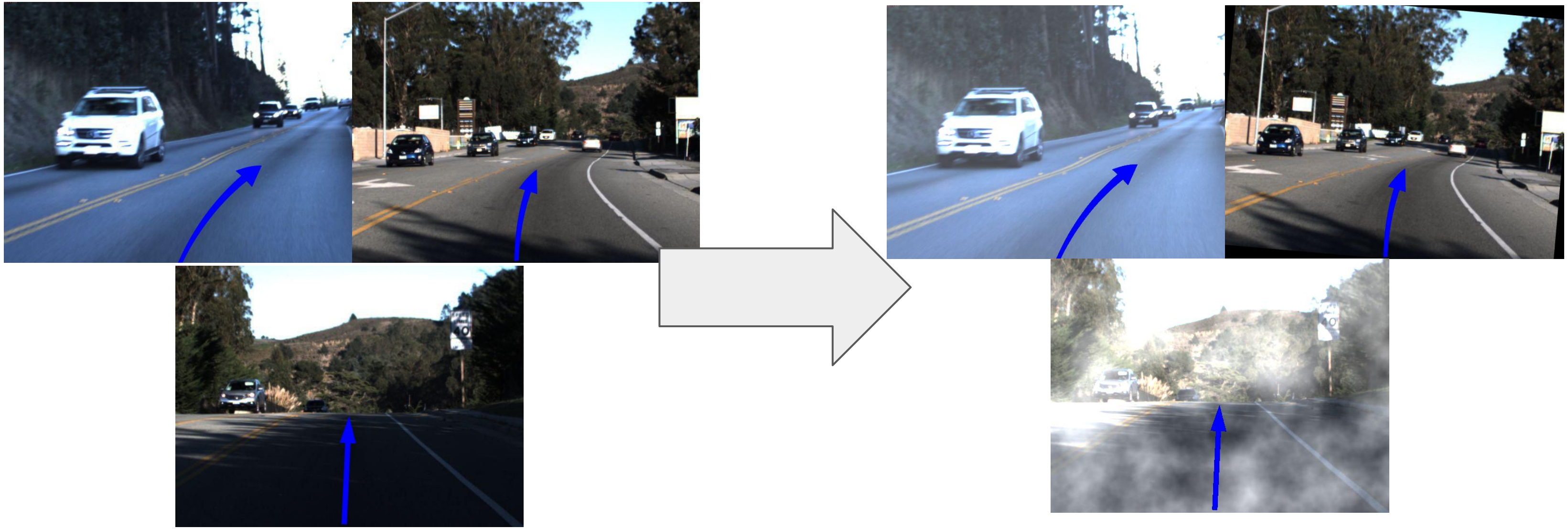
Used Transformations

- Brightness
- Contrast
- Translation
- Rotation
- Scale
- Blur
- Shear
- Rain
- Fog



Label generation

Use existing training data labels and use metamorphic relations:



New Solution

Test software with

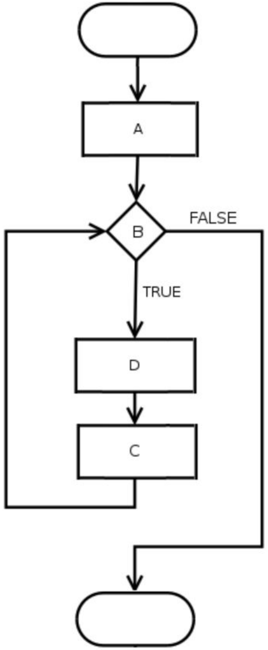
1. **automatically generated test cases**
   a. how to automatically generate new inputs
   b. how to automatically find fitting labels for these inputs
2. that **optimize**
   a. how to choose a good set of test cases
3. a specific **coverage criterion**
   a. how to measure behaviour coverage for a DNN

to show that all major behaviour patterns of the software perform as expected.

Classical mitigation: Control flow based testing

	Statement Coverage	**Branch Coverage**	**Modified Condition/ Decision Coverage**
ASIL A	highly recommended		
ASIL B	highly recommended	highly recommended	
ASIL C	recommended	highly recommended	
ASIL D	recommended	highly recommended	highly recommended

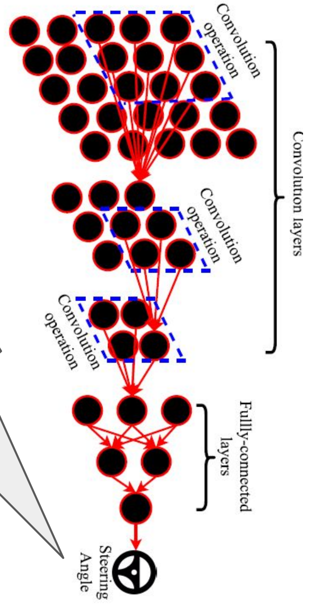
Classical Programming vs Machine Learning



(Wikipedia)

Logic lies in control flow

Coverage measured by looking at different contr
flows



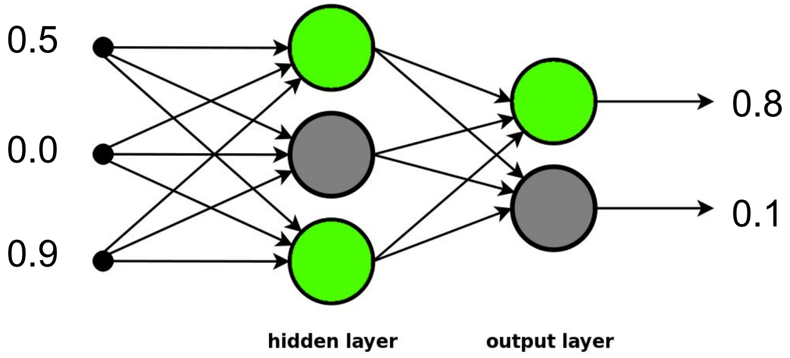
(Paper)

Logic lies within training data/learned weights

Need a coverage criterion for this kind of logic encoding

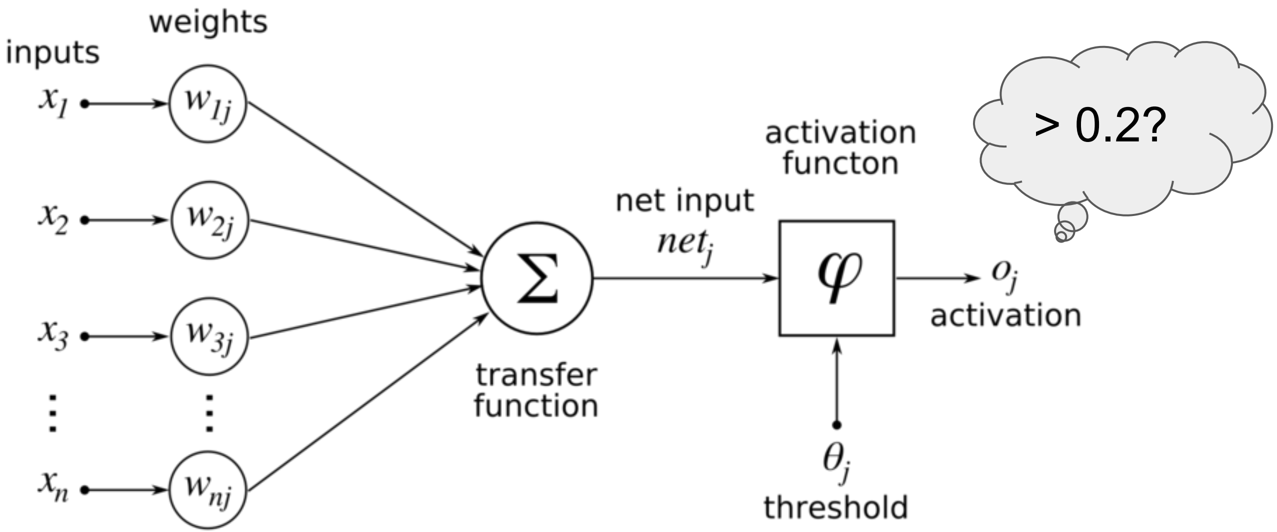
Proposed solution: Neuron Coverage (Pei et al.)

Measure how many neurons have been activated in a neuronal network



1. What is an activation?
2. Does Neuron Coverage relate to different behaviours of the network?

What is an activation?



(Wikipedia)

Neuron Coverage:
different behaviours of the network?

Empirical evidence:

- Strong correlation between steering angle and neuron coverage
  - Spearman rank correlation
- Neuronal coverage varies between left steering and right steering significantly
  - nonparametric Wilcoxon test

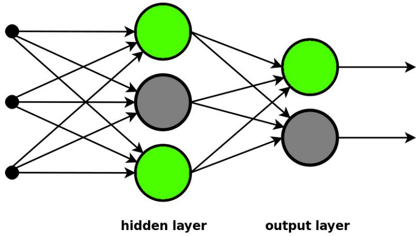
New Solution

Test software with

1. **automatically generated test cases**
   a. how to automatically generate new inputs
   b. how to automatically find fitting labels for these inputs
2. that **optimize**
   a. how to choose a good set of test cases
3. a specific **coverage criterion**
   a. how to measure behaviour coverage for a DNN

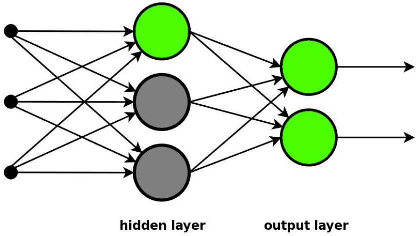
to show that all major behaviour patterns of the software perform as expected.

Neuron coverage of a whole data set



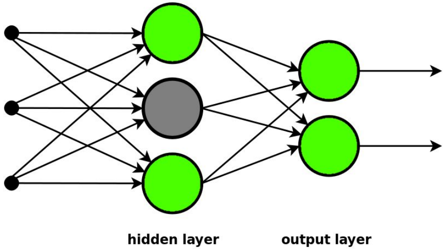
60% coverage

+



60% coverage

=



80% coverage

Optimization of Neuronal Coverage

Perform a greedy search for combinations of transformations

Input image

+

Transformations

Translation

Scale

+

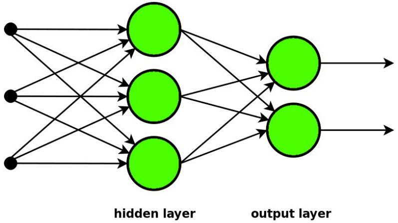
Parameters

(20, 10)

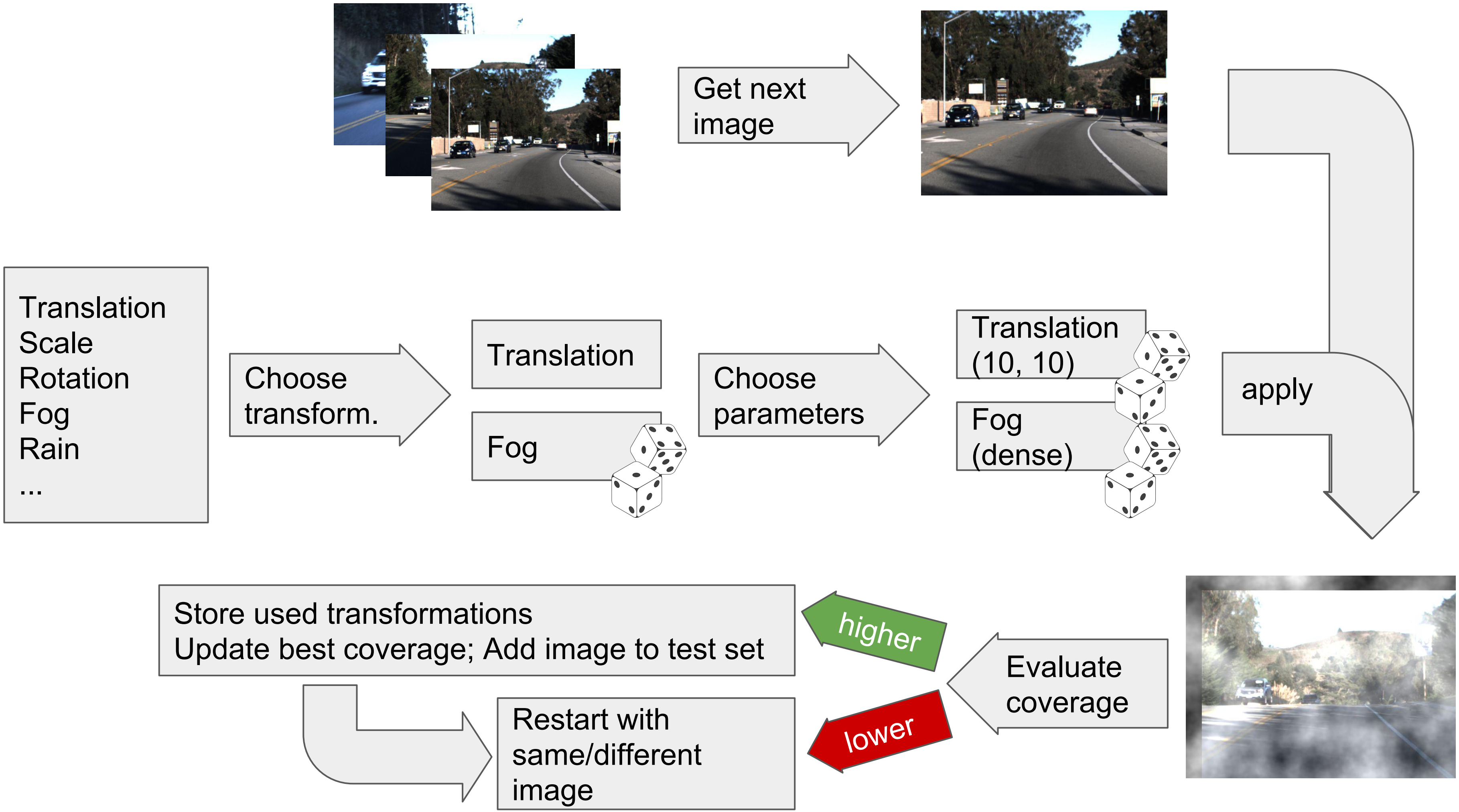
(10, 20)

(10, 10)

≈



100% coverage



Get next image
Translation
Scale
Rotation
Fog
Rain
...
Choose transform.
Translation
Fog
Choose parameters
Translation (10, 10)
Fog (dense)
apply
Store used transformations
Update best coverage; Add image to test set
higher
lower
Evaluate coverage
Restart with same/different image

New Solution

Test software with

1. **automatically generated test cases**
   a. how to automatically generate new inputs
   b. how to automatically find fitting labels for these inputs
2. that **optimize**
   a. how to choose a good set of test cases
3. a specific **coverage criterion**
   a. how to measure behaviour coverage for a DNN

to show that all major behaviour patterns of the software perform as expected.

Results

Empirical evaluation with neuronal networks out of Udacity self-driving car challenge

Results

Empirical evaluation with neuronal networks out of Udacity self-driving car challenge

Results

Empirical evaluation with neuronal networks out of Udacity self-driving car challenge

Results

Empirical evaluation with neuronal networks out of Udacity self-driving car challenge

- Detected 6339 erroneous behaviours in the 3 different models
- Neuron coverage can be increased by ~100% w.r.t. original test images
- Guided search of transformations provide ~20% increase compared to random combinations
- The sensitivity concerning single transformations varies between models

Problems/Criticism

- Realistic images resulting out of transformations?
- Neuronal coverage general justification?
- Does not perform well for Recurrent Neuronal Networks (RNNs)
- Transformations do not lead to exhaustive validation

Conclusions

Nguyen et al. propose a new automatic testing approach for DNNs.

Although being simplistic and not fully scientifically justified it is able to find major erroneous behaviours in otherwise well-performing DNNs.

The algorithm

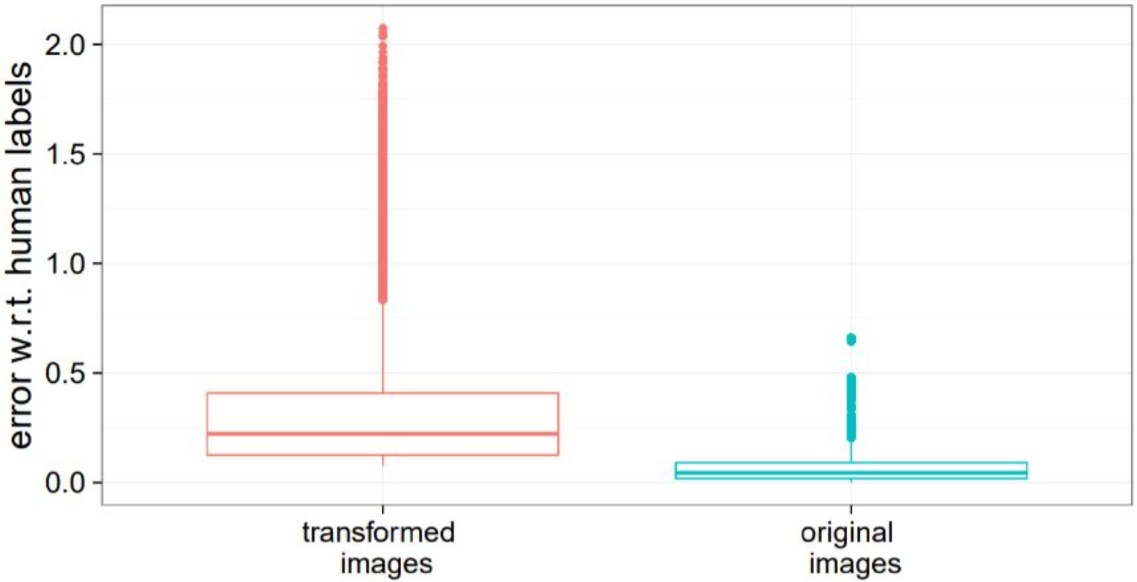
```
Input    : Transformations T, Seed images I
Output   : Synthetically generated test images
Variable : S: stack for storing newly generated images
           Tqueue: transformation queue

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
```

Retraining for the rescue?

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

Metamorphic relations

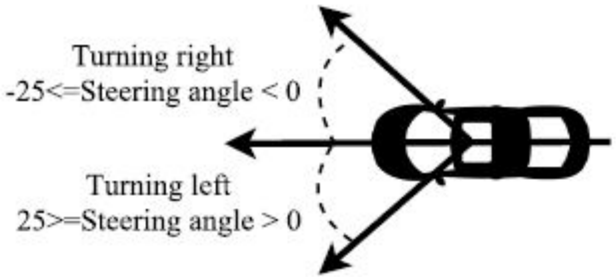
$$(\hat{\theta}_i - \theta_{ti})^2 \leq \lambda \, MSE_{orig}$$



Tested models

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]

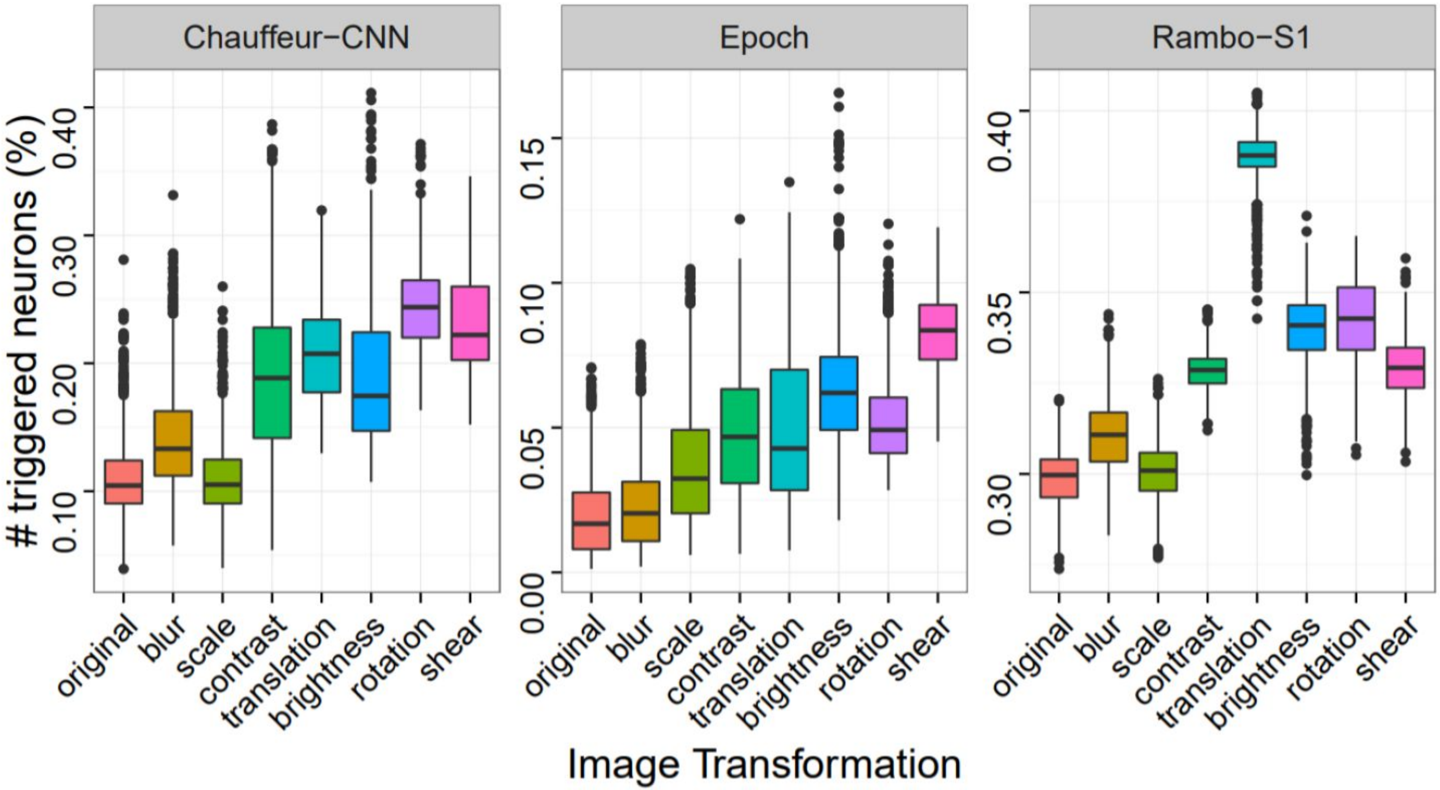


Neuron coverage valid?

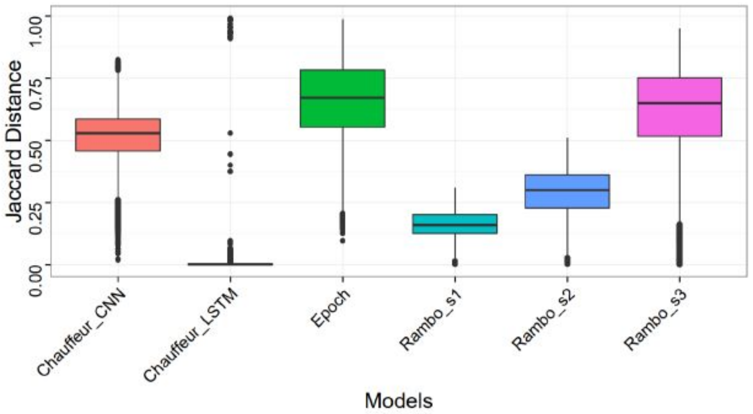
Model	Sub-Model	Steering Angle	Steering Direction	
		Spearman Correlation	Wilcoxon Test	Effect size (Cohen's d)
Chauffeur	Overall	-0.10 (***)	left (+ve) > right (-ve) (***)	negligible
	CNN	0.28 (***)	left (+ve) < right (-ve) (***)	negligible
	LSTM	-0.10 (***)	left (+ve) > right (-ve) (***)	negligible
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 $< 2.2 * 10^{-16}$

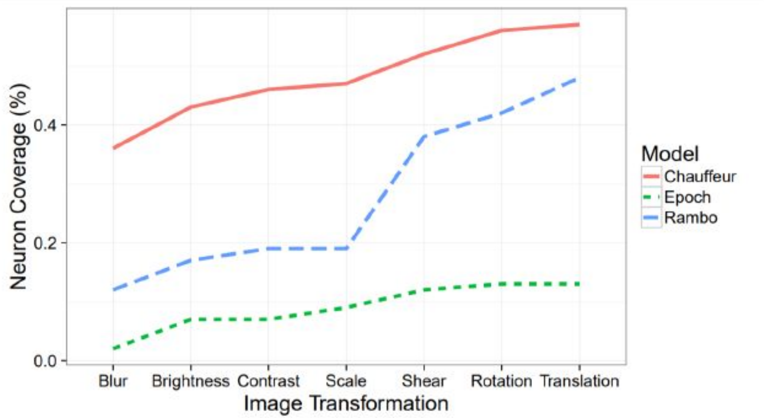
Transformations affecting neural coverage (1)



Transformations affecting neural coverage (2)



4.1 Difference in neuron coverage caused by different image transformations



4.2 Average cumulative neuron coverage per input image

Figure 4: Different image transformations activate significantly different neurons. In the top figure the median Jaccard distances for Chauffeur-CNN, Chauffeur-LSTM, Epoch, Rambo-S1, Rambo-S2, and Rambo-S3 models are 0.53, 0.002, 0.67, 0.12, 0.17, 0.30, and 0.65.

Used parameters for transformations

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

λ (see Eqn. 2)	**Simple Tranformation** ϵ (see Eqn. 3)					**Composite Transformation**		
	0.01	0.02	0.03	0.04	0.05	**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	4112
Guided	89	65	666