Trustworthy Machine Learning-Enabled Systems

Lionel Briand

http://www.lbriand.info

IFIP WG 10.4, 2022

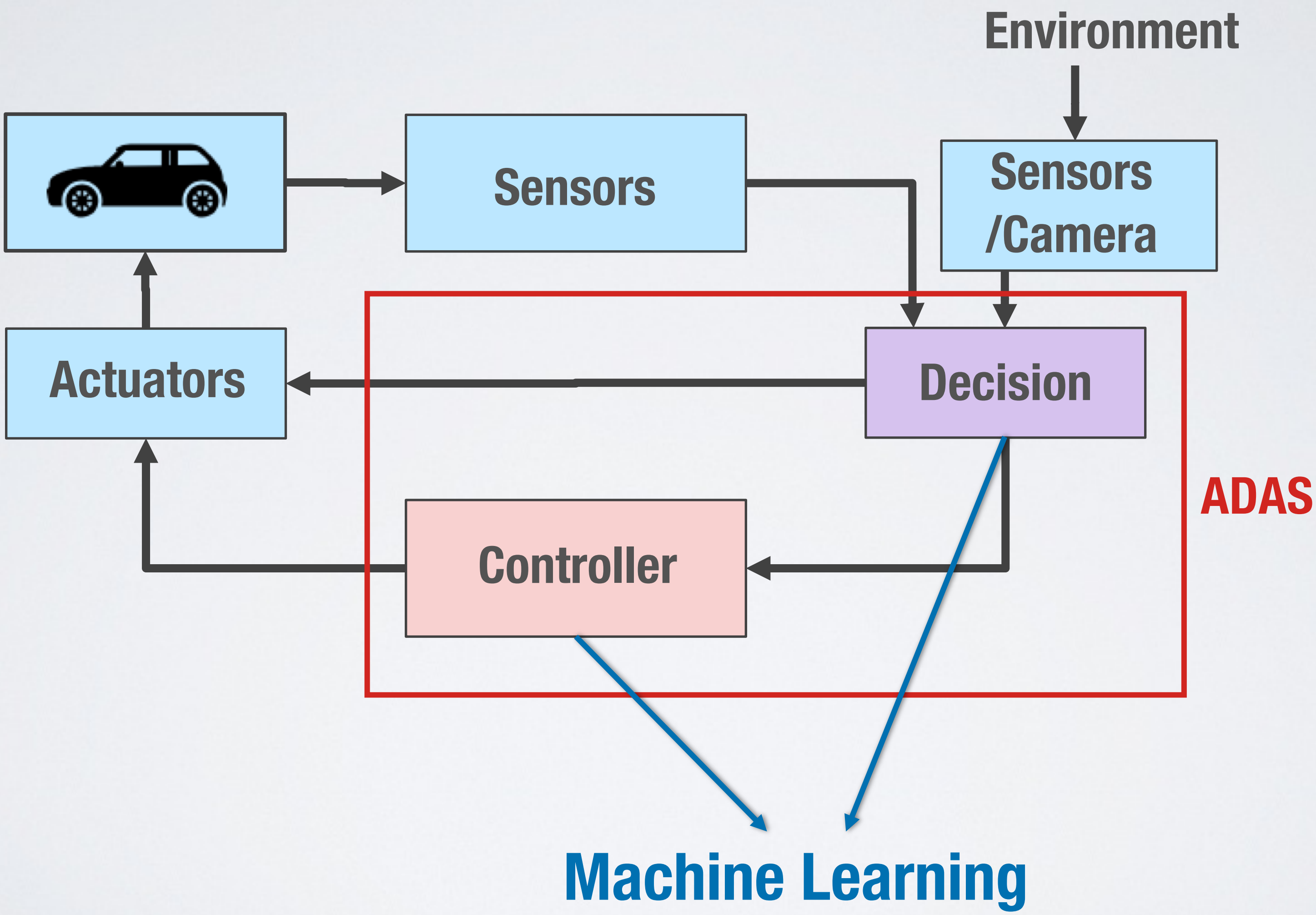
Canada Research Chairs
Chaires de recherche du Canada
Canada

Context and Motivations

Importance

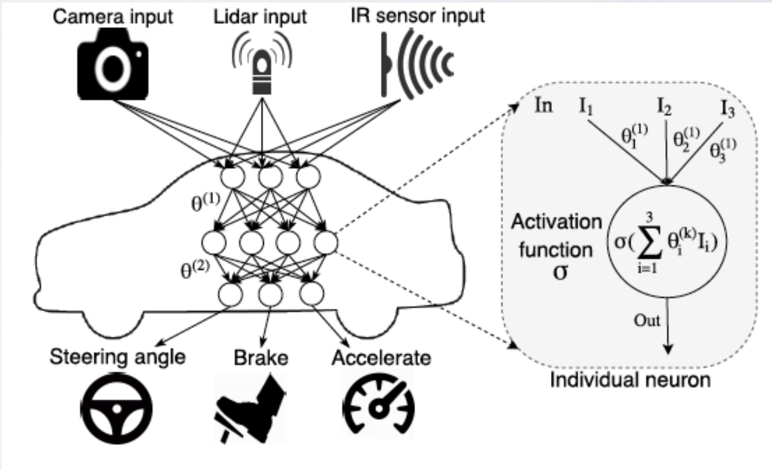
- ML components are increasingly part of safety- or mission-critical systems (ML-enabled systems - MLS)
- Many domains, including aerospace, automotive, health care, …
- Many ML algorithms, supervised vs. unsupervised, classification vs regression, etc.
- But increasing use of deep learning and reinforcement learning

ML-Enabled Systems (MLS)



Example Automotive Applications

- Object detection, identification, classification, localization and prediction of movement
- Sensor fusion and scene comprehension, e.g., lane detection
- Driver monitoring
- Driver replacement
- Functional safety, security
- Powertrains, e.g., improve motor control and battery management



Tian et al. 2018

Testing Levels

- Testing is still the main mechanism through which to gain trust
- Levels: model (e.g., Deep Neural Networks or DNN) , integration, system
- Research largely focused on model testing
- Integration: Issues that arise when multiple models and components are integrated
- System: Test the MLS in its target environment, in-field or simulated
- Cross-cutting concerns: scalability, realism

Information Access

- Black-box: Model inputs and outputs
- Data-box: Training and test set originally used
- White-box: runtime state (neuron activation), hyperparameters, weight and biases
- In practice, data-box and white-box access are often not guaranteed, e.g., third party provider

Model Testing Objectives

- Correctness of classifications and predictions (regression)
- Robustness (to noise or attacks)
- Fairness (e.g., gender, race …)
- Efficiency: Learning and prediction speed
- Causes of failures: imperfect training (training set, overfitting …), hyper-parameters, model structure …
- But what do these failures really entail for the system?

Challenges: Overview

- Behavior driven by training data and learning process
- Neither specifications nor code
- Huge input space, especially for autonomous systems
- Test suite adequacy, i.e., when is it good enough?
- Automated test oracles / verdicts
- Models are never perfect, but how do we decide whether they are good enough?

Challenges

Large Input Space

- Inputs take a **variety of forms**: images, code, text, simulation configuration parameters, …
- Incredibly large input spaces
- **Cost of test execution** (including simulation) can be high
- **Labelling effort**, when no automation is possible, is high

Automated Emergency Braking System (AEB)

Decision making

Vision (Camera)

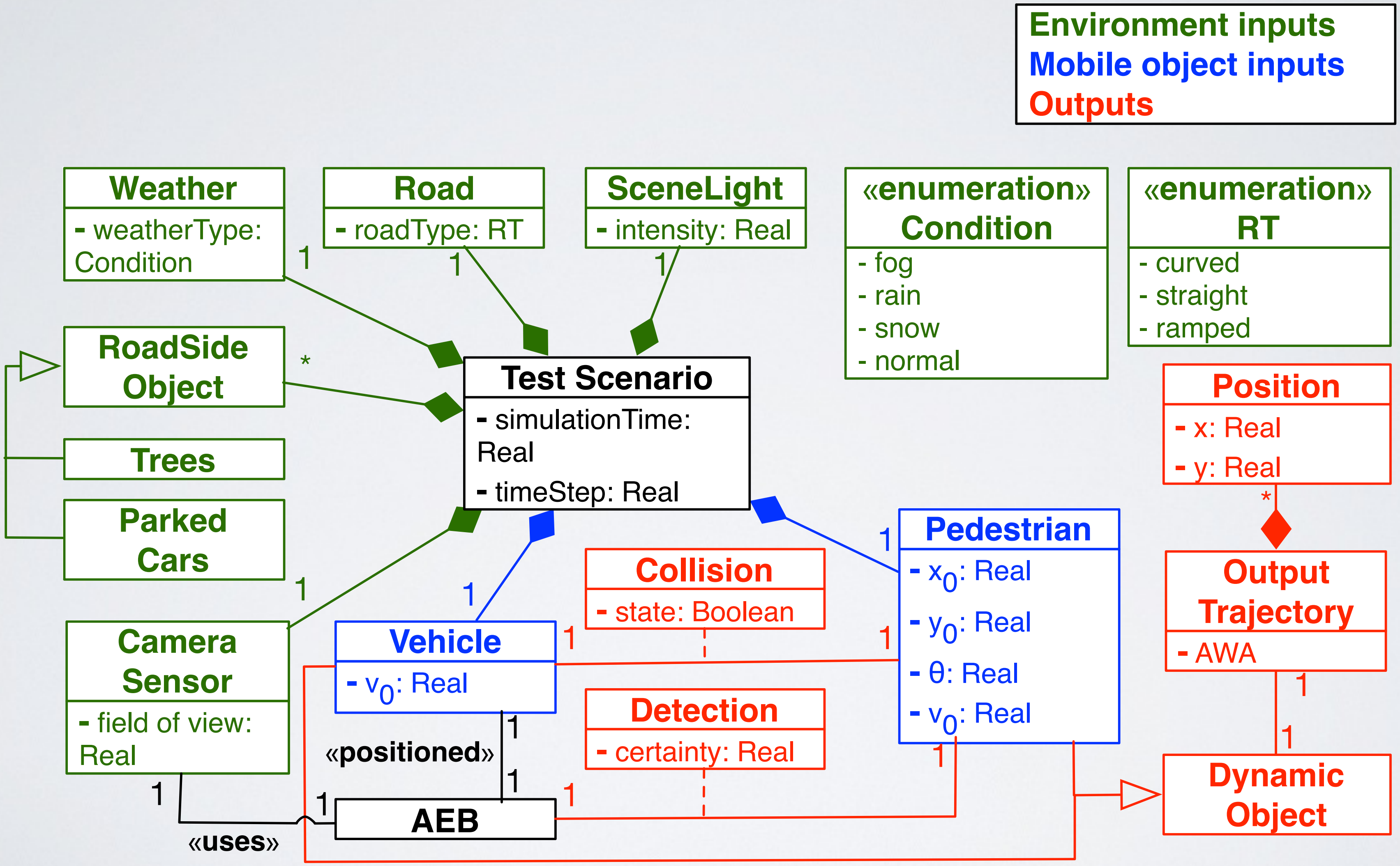
Sensor

Objects' position/speed

Brake Controller

"Brake-request" when braking is needed to avoid collisions

AEB Input-Output Domain



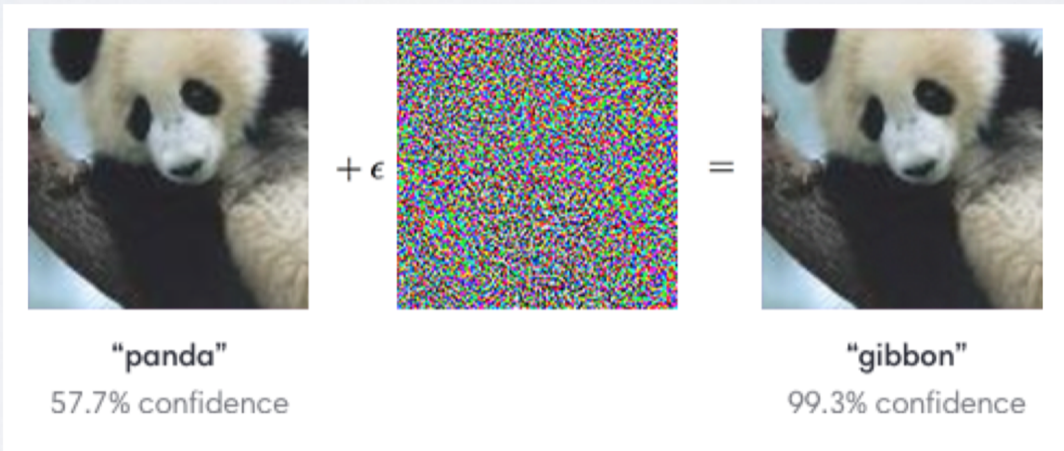
Inputs: Adversarial or “Natural”?

- **Adversarial inputs:** Focus on robustness, e.g., noise or attacks
- **Natural inputs:** Focus on functional aspects, e.g., functional safety

Adversarial Examples

- Szegedy et al. first indicated an intriguing weakness of DNNs in the context of image classification

  - “Applying an imperceptible perturbation to a test image is possible to arbitrarily change the DNN’s prediction”



Adversarial example due to noise (Goodfellow et al., 2014)

Adversarial Inputs

- Input changes that are **not expected to lead to any (significant) change** in model prediction or decision
- **Techniques:** Image processing, image transformations (GAN), fuzzing
- Are often **not realistic** …

“Natural” Inputs

- Focused on functional aspects
- Inputs should be **realistic**
- **Suffer from the oracle problem**: what should be the expected classification or prediction for new inputs?

Single-Image Test Inputs

- In the context of ADAS test inputs have been generated by applying label-preserving changes to existing already-labeled data (Tian et al., Zhang et al., 2018)

Original image

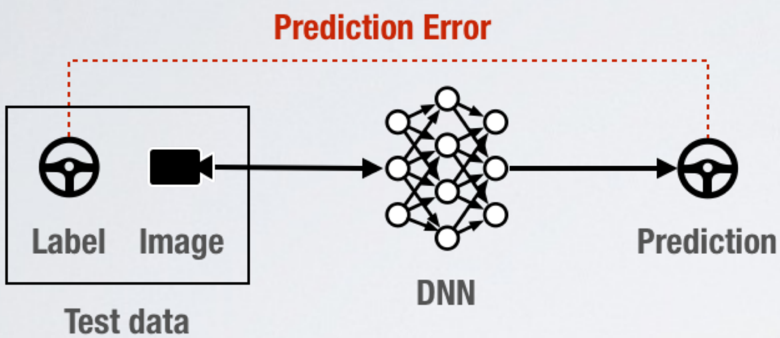
Test image
(generated by adding fog)

Test Scenarios

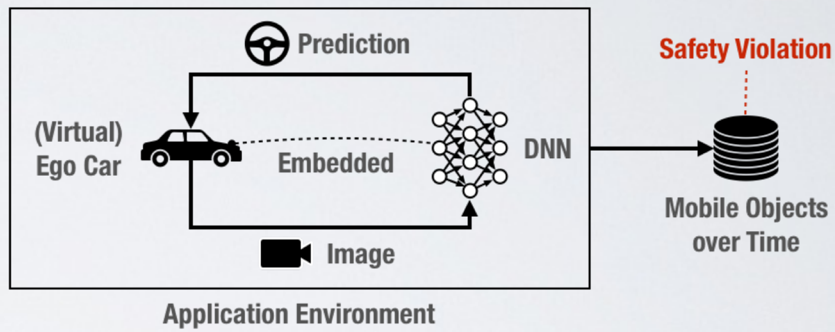
- Most of existing research focuses on
  - **Testing DNN components**, not systems containing them
  - **Single inputs** with label-preserving changes, e.g., to images
- Limited solutions accounting for the impact of **object dynamics** (e.g., car speed) in different **scenarios** (e.g., specific configurations of roads).
- Limited solutions regarding **functional safety** over scenarios.
- **ISO/PAS Road vehicles (SOTIF) requirements:** In-the-loop testing of "relevant" scenarios in different environmental conditions

Offline and Online Testing

Offline



Online



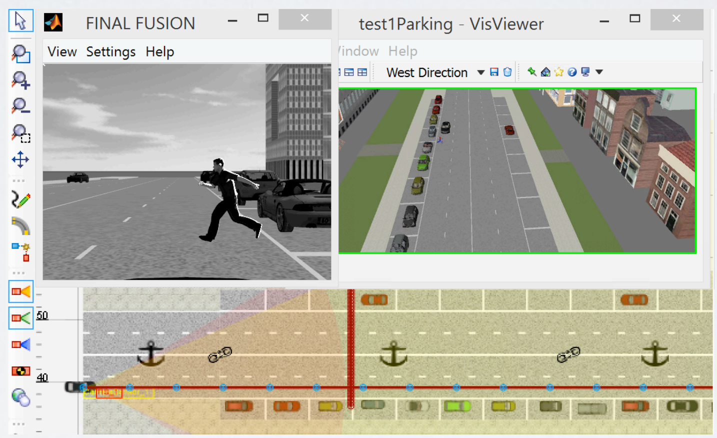
- For many MLS, considering **single inputs is not adequate**. Sequences must be considered as context, e.g., images for steering angle DNN.
- Offline testing is less expensive but does not account for **physical dynamics and cumulative effects** of prediction uncertainty over time.
- How do offline and online testing results **differ and complement** each other?

Testing via Physics-based Simulation

Model (Matlab/Simulink)

ADAS (SUT)

Simulator (Matlab/Simulink)



- Physical plant (vehicle / sensors / actuators)
- Other cars
- Pedestrians
- Environment (weather / roads / traffic signs)

Test input

Test output

time-stamped output

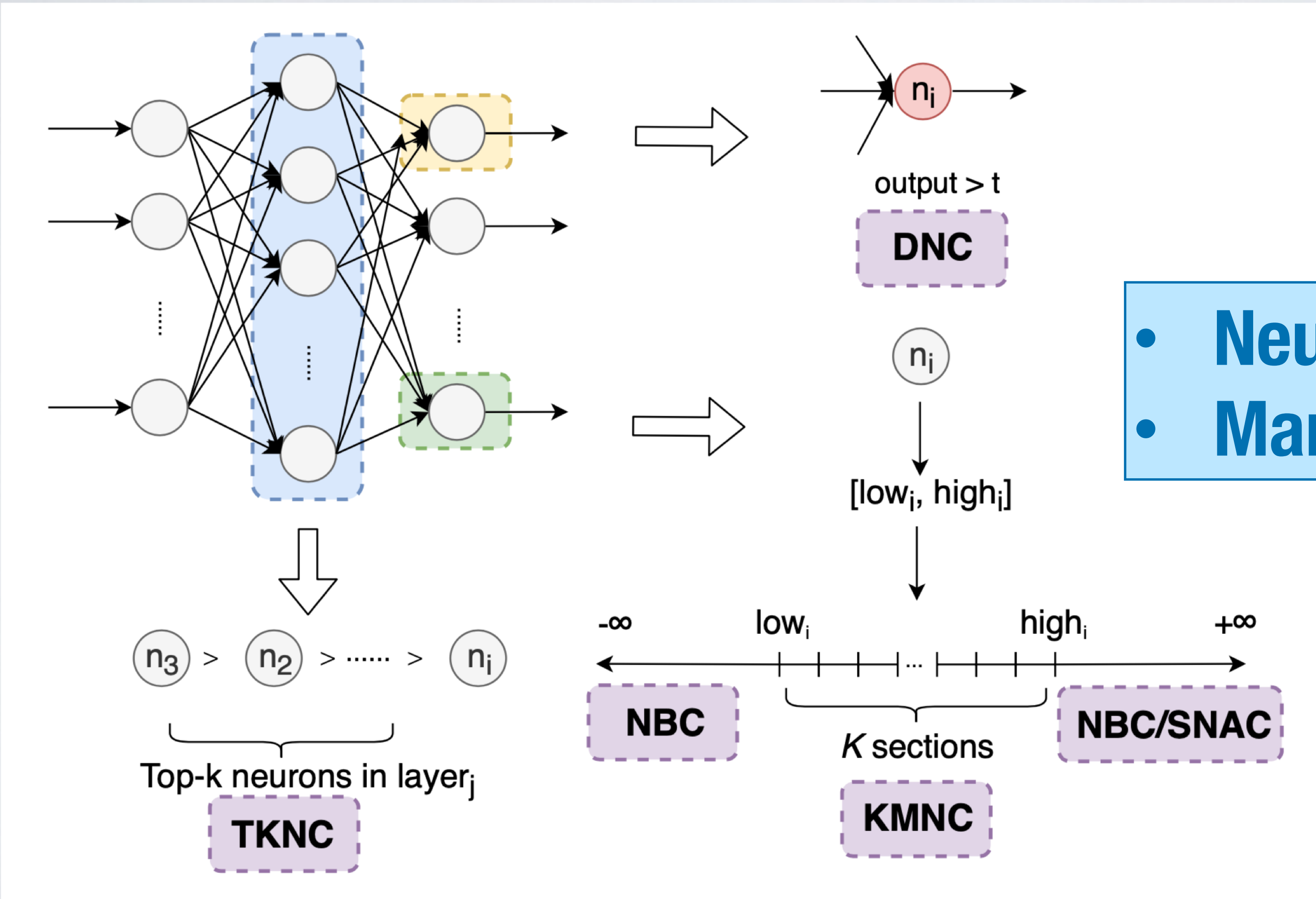
Simulation: Challenges

- How to effectively guide the simulator?
- Simulation is highly expensive
- Fidelity of simulation

Test Adequacy Criteria

- **Model testing of DNNs**
- **Goal:** Assess test suite adequacy, guide test selection
- Can help devise **minimal and fault-revealing** test suites
- Require access to the **DNN internals** and sometimes the training set. Not realistic in many practical settings.

Structural Coverage Criteria



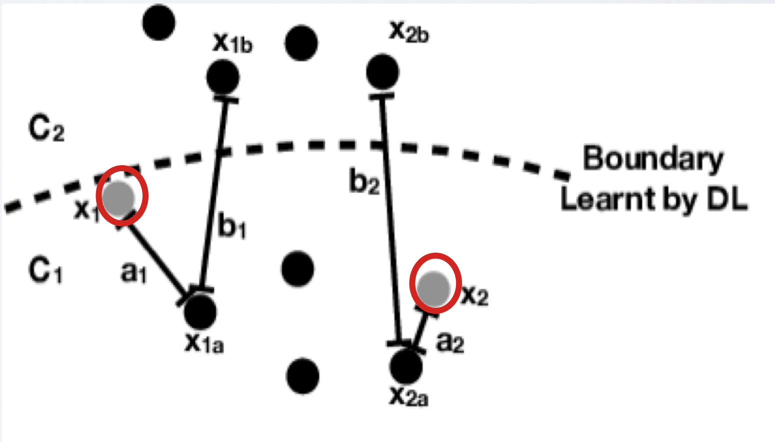
- **Neuron coverage**
- **Many variants**

Chen et al. 2020

Surprise Adequacy Criteria

- **Not structural**
- **Surprise Adequacy (SA)** aims to measure the relative novelty (i.e., surprise) of a given new input with respect to the inputs used for training.
- **Assumption:** DNNs are likely to be more error prone for inputs that are unfamiliar.
- **DSA:** Test inputs closer to the class boundaries are more valuable

DSA(x1) = a1/b1
DSA(x2) = a2/b2
DSA(x1) > DSA(x2)



Kim et al. 2019

Limitations

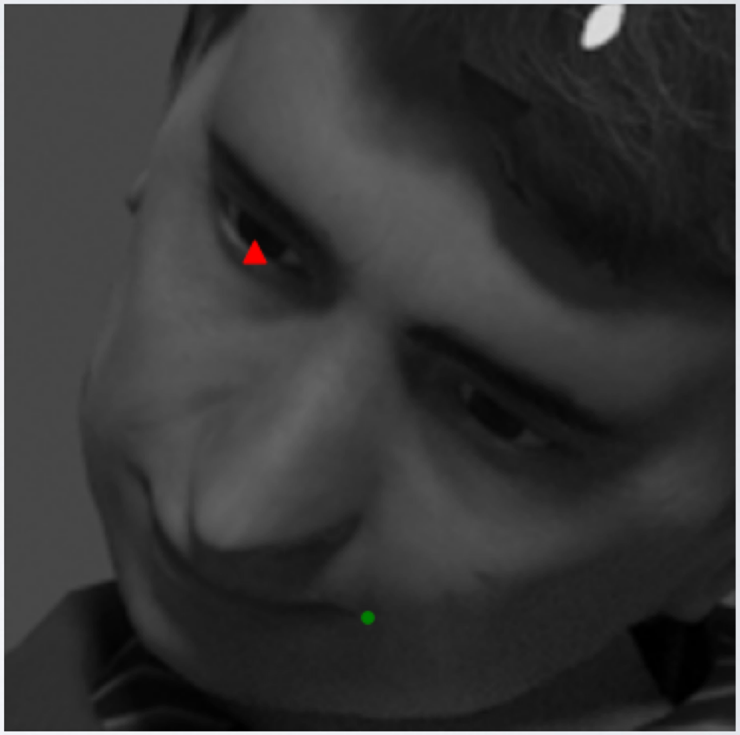
- Code coverage assumes:
  - (1) the homogeneity of inputs covering the same part of a program
  - (2) the diversity of inputs to be indicated by coverage metrics
- According to Li et al. (2019):
  - These assumptions break down for DNNs and adversarial inputs
  - There is a weak correlation between coverage and misclassification for natural inputs
- Scalability and applicability for the most complex coverage metrics?

Failures in MLS

- Model level: misclassifications, square error (regression)
- Uncertainty inherent to ML training
- What is a failure then in an MLS?
- Expected robustness of MLS to ML errors
- Failure at system level: Requirement violation
- MLS failures result from both ML mispredictions and effectiveness of countermeasures, e.g., safety monitors

Example: Key-points Detection

- DNNs used for key-points detection in images
- Many applications, e.g., face recognition
- Testing: **Find test suite that causes DNN to poorly predict as many key-points as possible within time budget**
- Impact of poor predictions on MLS? **Alternative key-points** can be used for the same purpose.



Ground truth
Predicted

Oracles (1)

- How to identify misclassifications or mispredictions?
- Required for testing purposes
- It may be difficult to manually determine the **correct outputs of a model** for a (large) set of inputs
- Effort-intensive, **third-party data labelling** companies

Oracles (2)

- **Simulators** can help automate the oracle, if they have **sufficient fidelity**. Common in many industrial domains.
- **Important:** Mispredictions may be unavoidable, and accepted, e.g., shadows in images
- **Minimizing the test suite** is often the only option

Practical Accuracy Estimation (PACE)

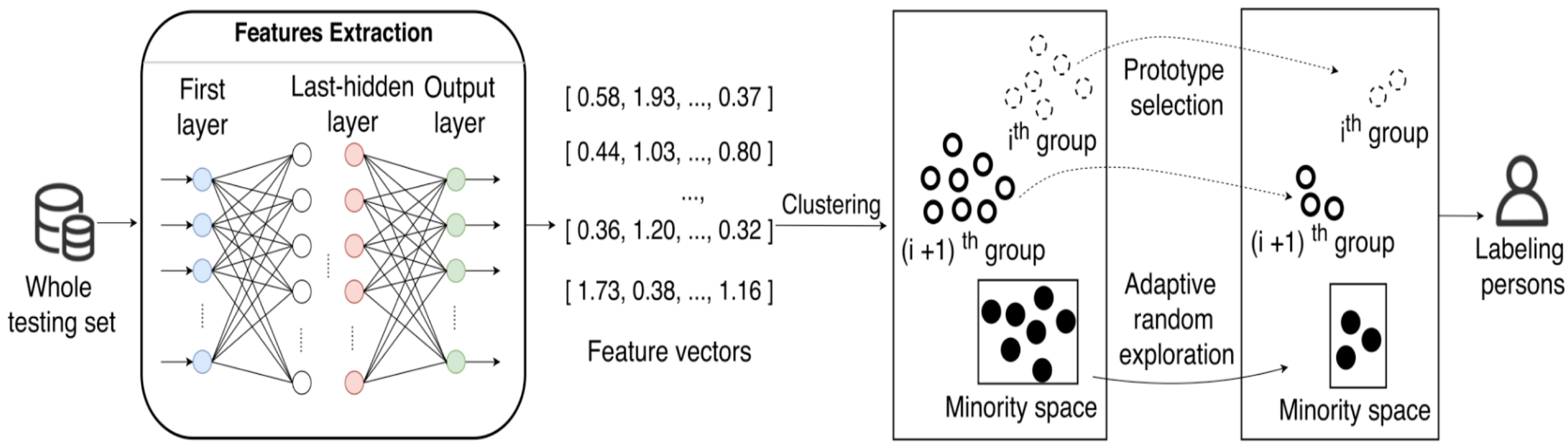
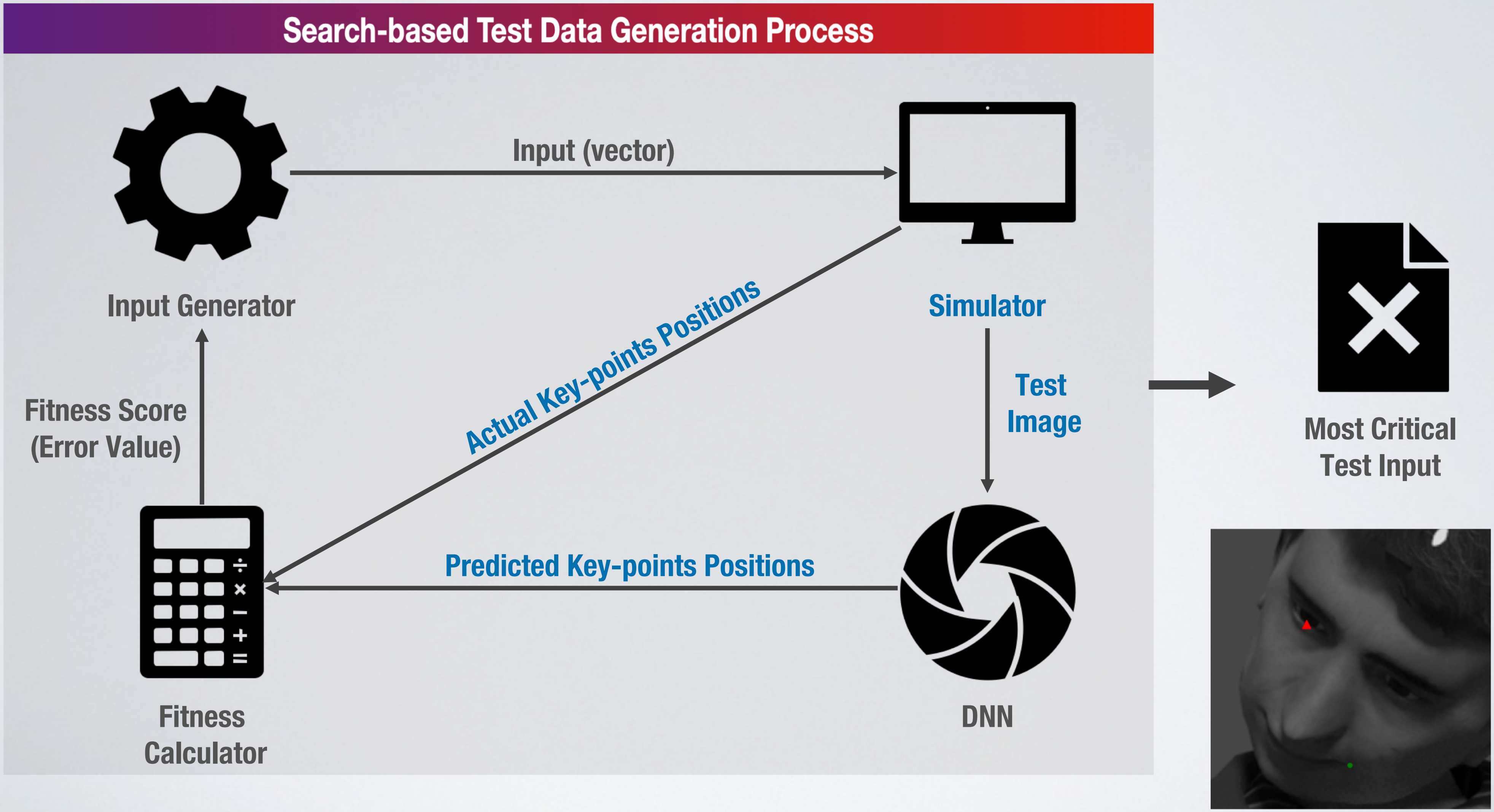


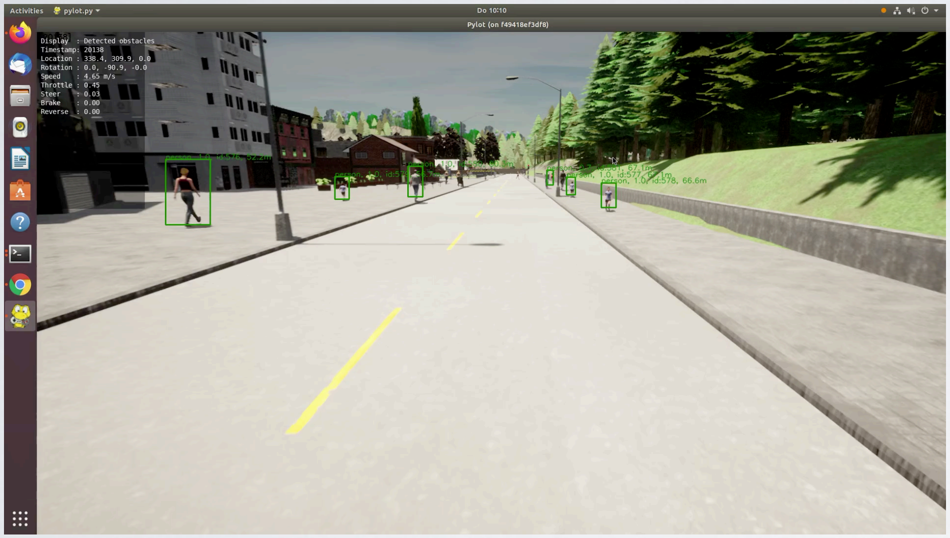
Fig. 1. Overview of PACE.

- Chen et al., TOSEM 2020: Minimizing test suites

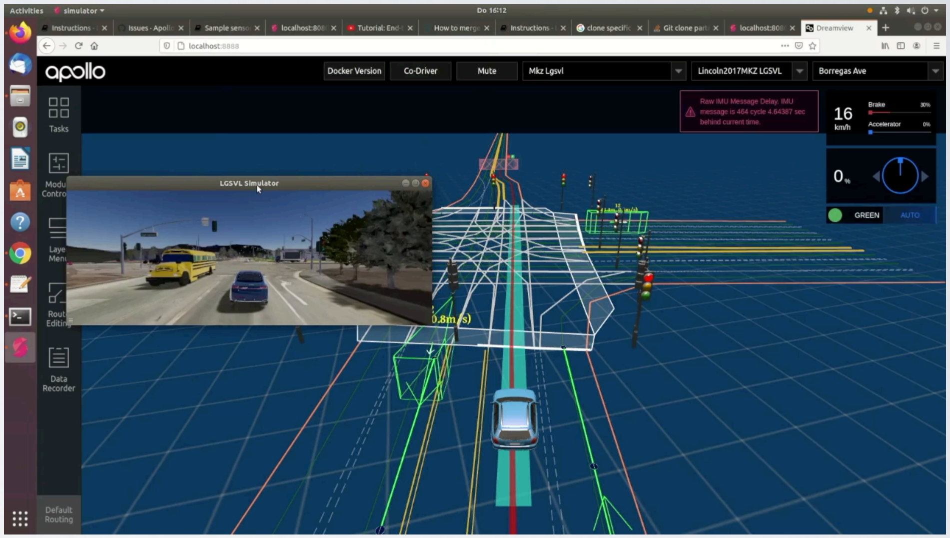
Simulation: Key-points Detection



Simulation+DNN Examples



Pylot + Carla



Apollo + LGSVL

High-fidelity simulators

- Carla
- LGSVL

DNN-based ADAS

- Pylot: many DNN models
- Apollo: 20 DNN models

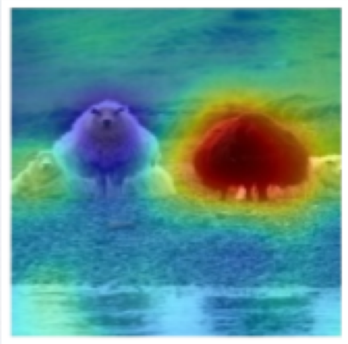
ML and Functional Safety

- Requires to **assess risks** in a realistic fashion
- **Account for conditions** and **consequences** of failures
- Is the **uncertainty** associated with an ML model acceptable?
- With ML, **automated support** is required, given the difficulties in interpreting model test results

Explaining Misclassifications

- **Based on visual heatmaps:** use colors to capture the extent to which different features contribute to the misclassification of the input.

Black sheep misclassified as cow



- State-of-the-art
  - black-box techniques: perturbations of input image
  - white-box techniques: backward propagation of prediction score
  - They require, in our context, **unreasonable amounts of manual analysis work** to help explain safety violations based on image heatmaps

Research Directions

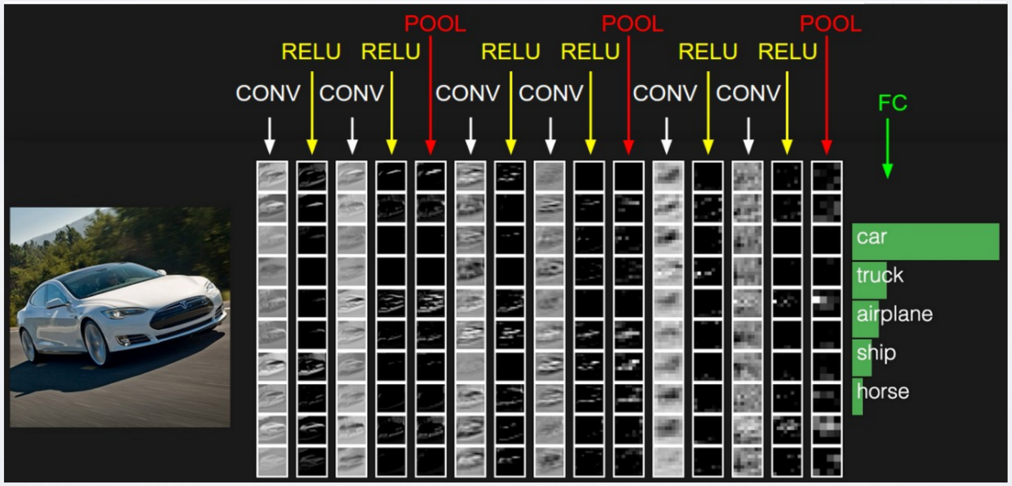
Evaluating and Selecting DNN Test suites

Objectives

- Effectively and efficiently explore the space of possible DNN inputs to identify and characterize unsafe parts of the input space.
- The main motivation is to decrease the manual effort required for labeling test data.
- Black-box approach based on measuring the diversity of test inputs.
- The more diverse, the more likely they are to reveal faults

Extracting Image Features

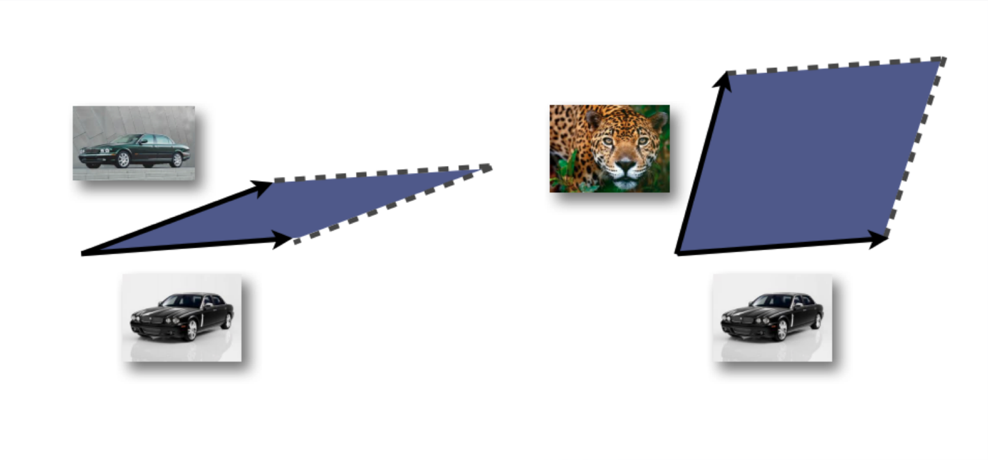
- **VGG16** is a convolutional neural network trained on a subset of the **ImageNet** dataset, a collection of over 14 million images belonging to 22,000 categories.



Geometric Diversity (GD)

- Given a **dataset X** and its corresponding **feature vectors V**, the geometric diversity of a subset $S \subseteq X$ is defined as the **hyper-volume of the parallelepiped spanned by the rows of V**, i.e., feature vectors of items in S, the larger the volume, the more diverse is the feature space of S

$$G(S) = det(V_S * V_S^T)$$



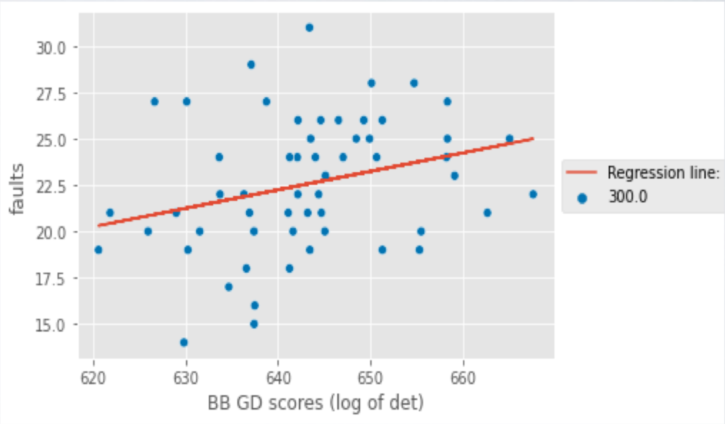
Kulesza et al., 2012

Representative Results

- 60 subsets with size of 300 from MNIST
- Correlation between geometric diversity and faults:

  Spearman= 32.89% p-value = 0.013

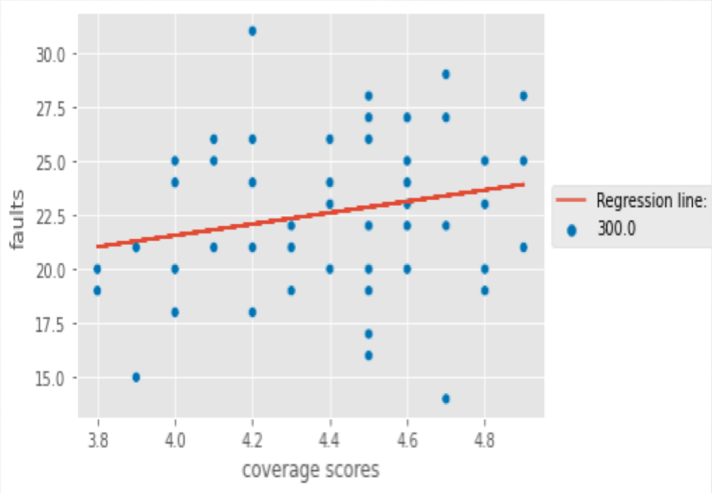
  Pearson=29.91% p-value = 0.027



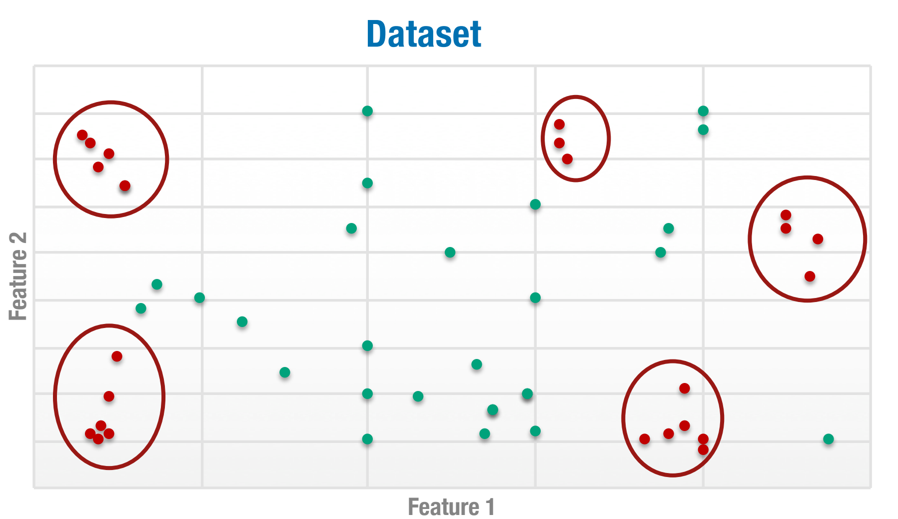
- 60 subsets with size of 300 from MNIST
- Correlation between surprise adequacy coverage and faults:

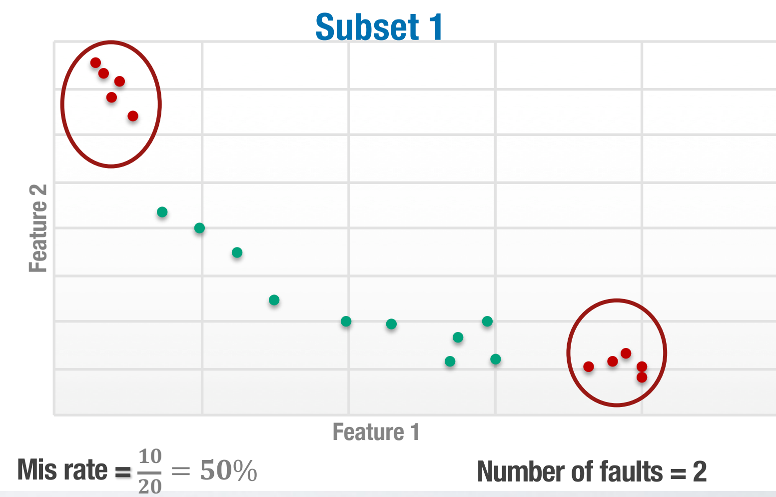
  Spearman= 24.55% p-value = 0.07

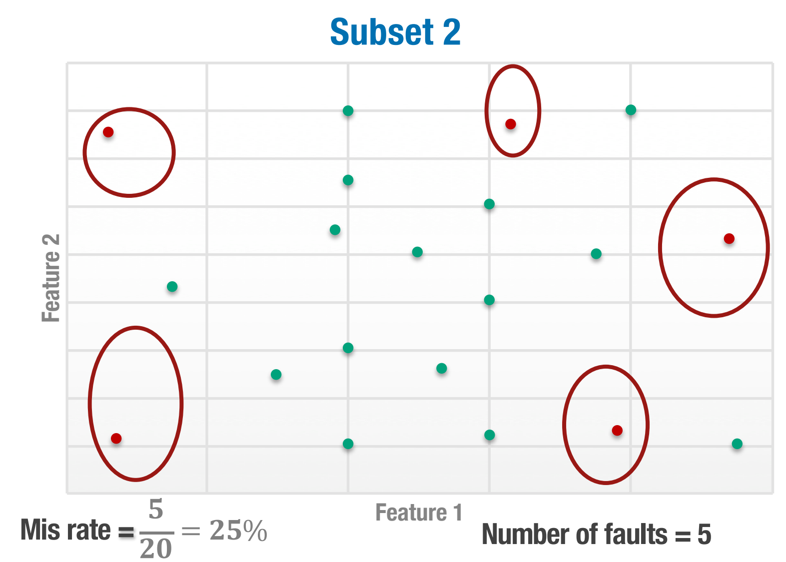
  Pearson= 22% p-value = 0.11



Mispredictions vs. Faults

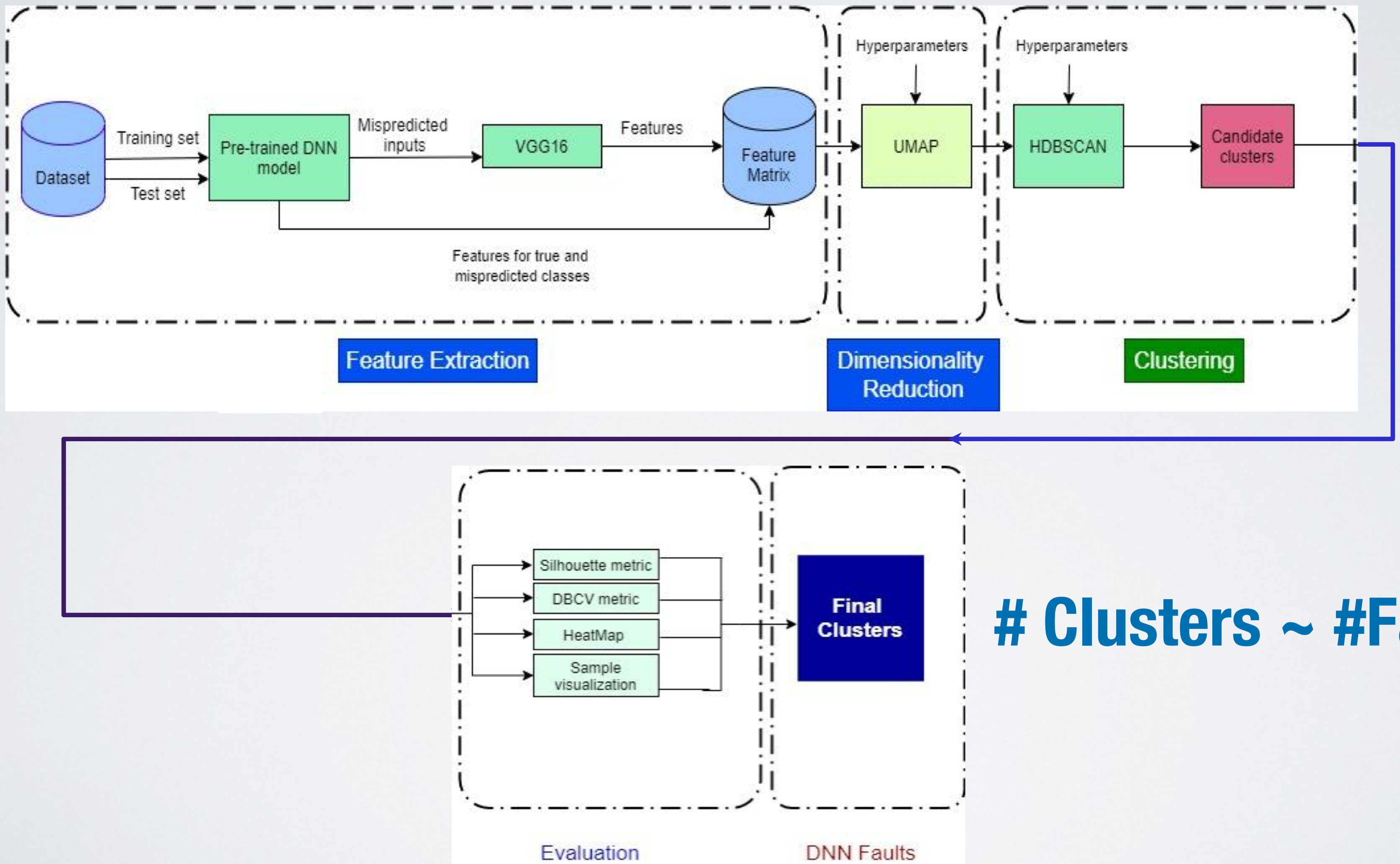






Misprediction rates are misleading to evaluate DNN diversity or coverage metrics

Estimating Faults with Clustering



 # Clusters ~ #Faults

Summary

- How to define and compute diversity?
- Geometric diversity based on extracted features (black-box)
- Moderate, positive and statistically significant correlations between geometric diversity and faults.
- Coverage is not strongly and positively correlated with fault detection.
- We are approximating fault counts in DNN by applying clustering techniques. This could affect correlations.

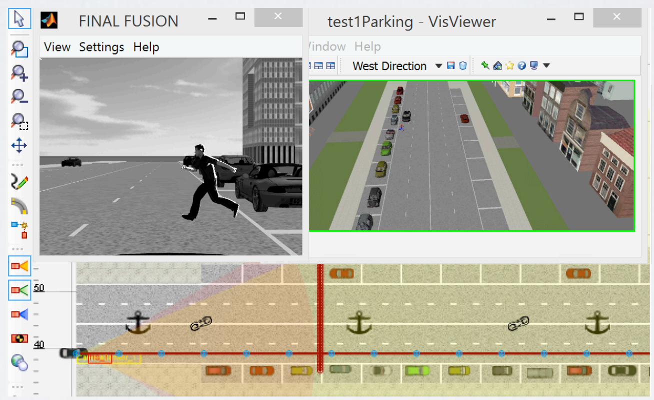
Simulation-Based Testing of MLS

Objectives

- Effectively and efficiently explore the space of possible system scenarios to identify and characterize unsafe parts of the scenario space.
- Automate online testing
- Requires simulation with sufficient fidelity
- Scalability

Example: ADAS Testing

Simulator (Matlab/Simulink)



- Physical plant (vehicle / sensors / actuators)
- Other cars
- Pedestrians
- Environment (weather / roads / traffic signs)

Model (Matlab/Simulink)

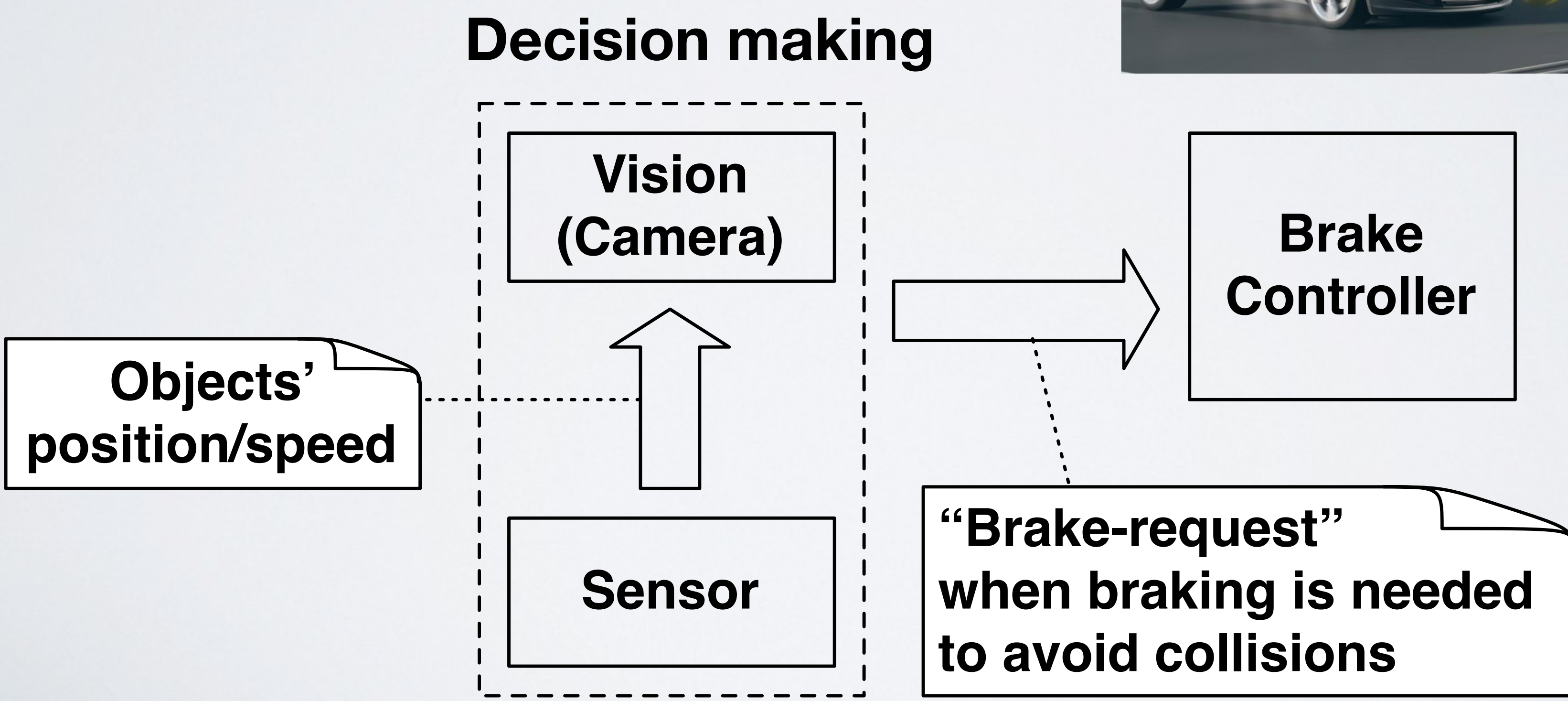
ADAS (SUT)

Test input

Test output

time-stamped output

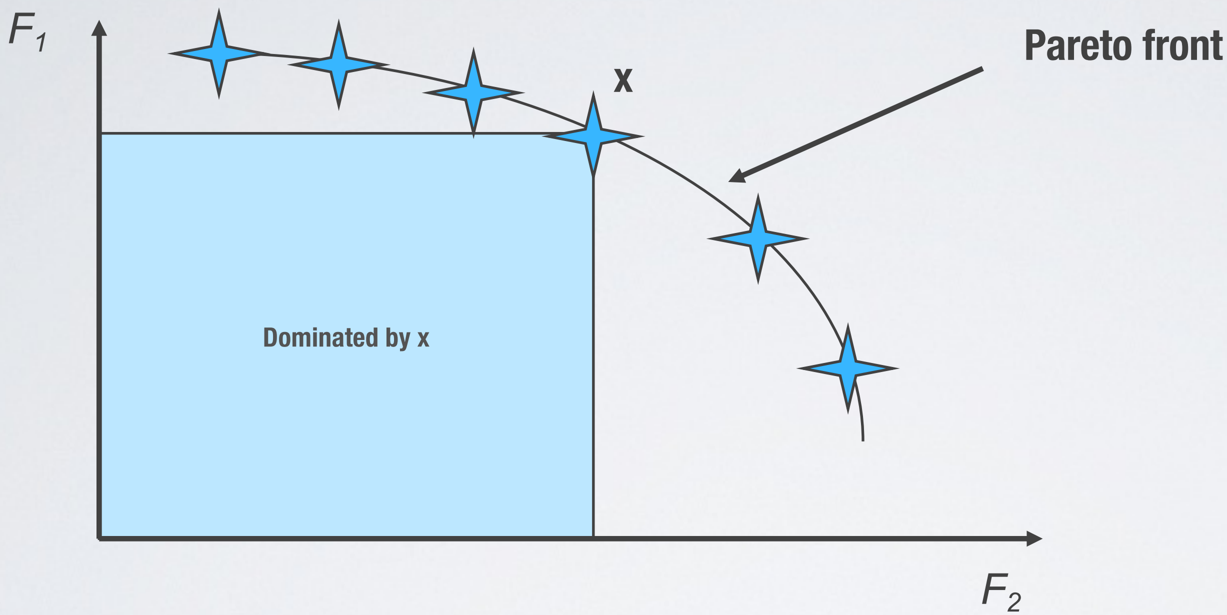
Automated Emergency Braking System (AEB)



Test Approach

- We use **multi-objective** search algorithm (NSGA II).
- **Objective Functions:**
  1. Minimum distance between the pedestrian and the field of view
  2. The car speed at the time of collision
  3. The probability that the object detected is a pedestrian
- We use **decision tree classification models** to speed up the search and explain violations.
- Each search iteration **calls simulation** to compute objective functions.

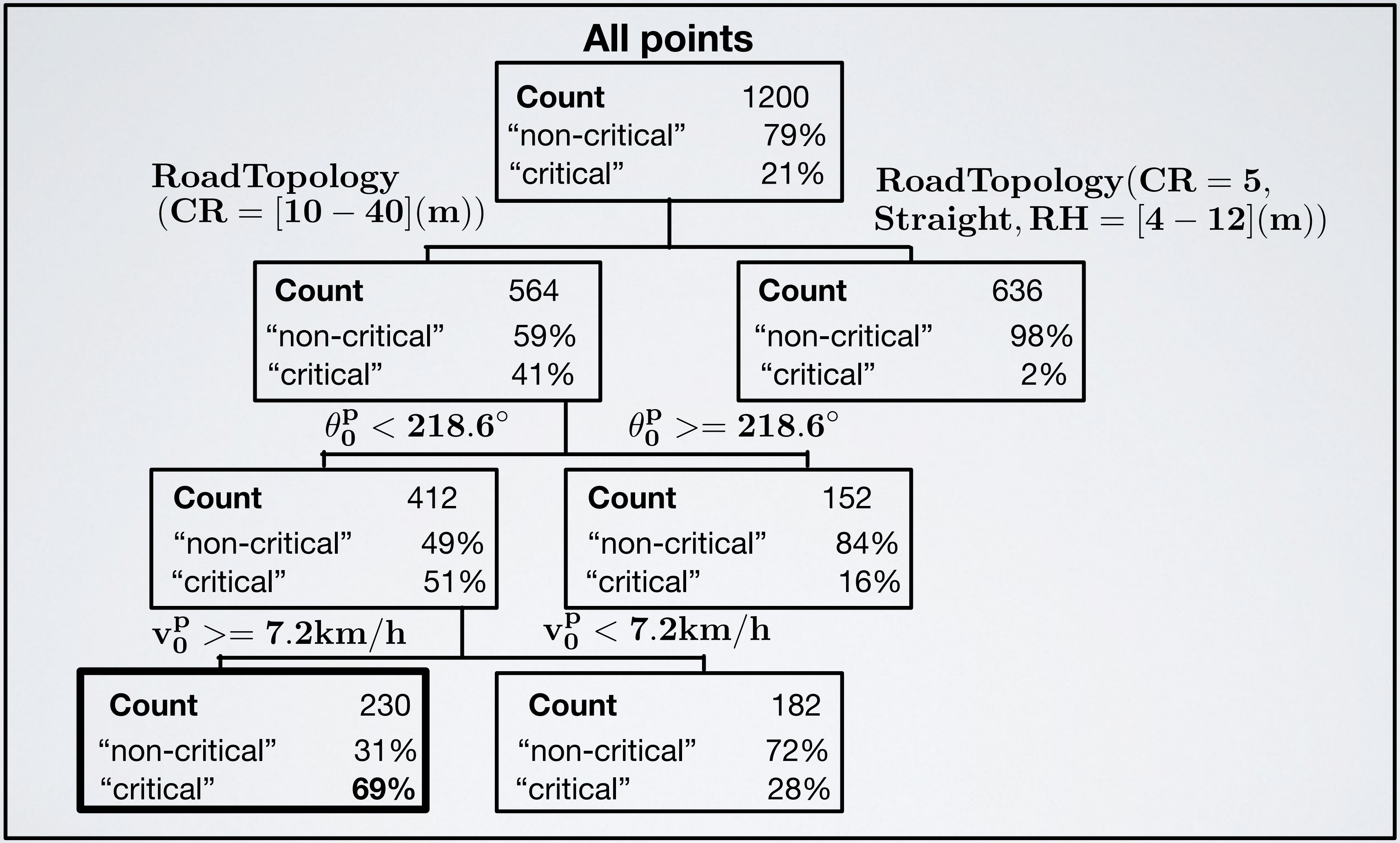
Multiple Objectives Search



Individual A Pareto dominates individual B if A is at least as good as B in every objective and better than B in at least one objective.

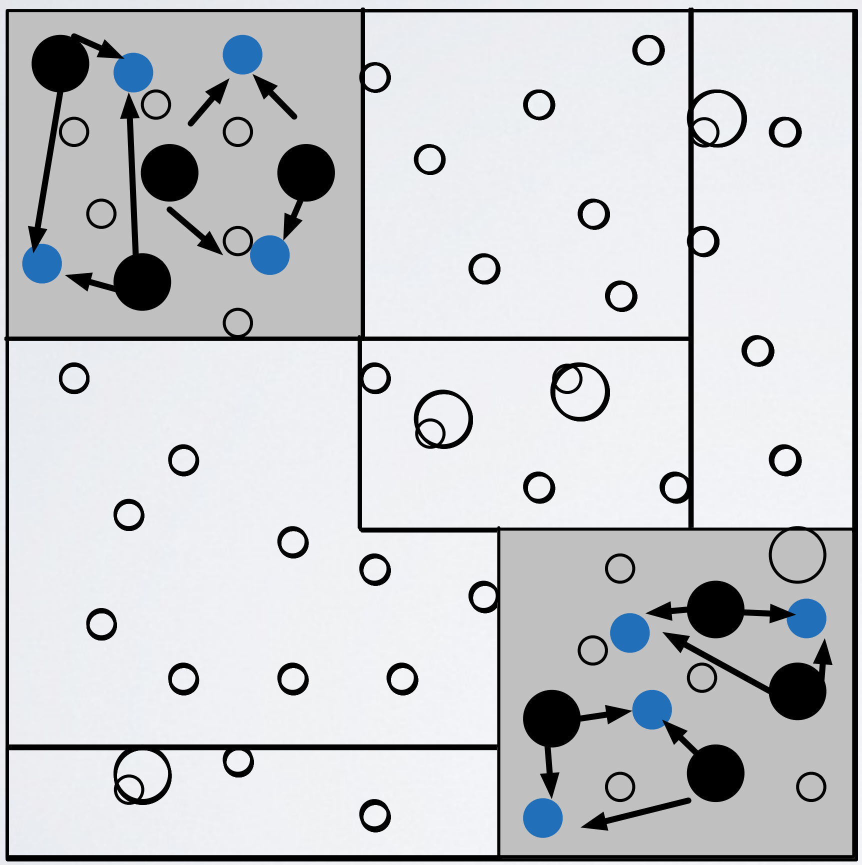
- A multi-objective optimization algorithm (e.g., NSGA II) must:
  - **Guide the search towards the global Pareto-Optimal front.**
  - **Maintain solution diversity in the Pareto-Optimal front.**

Decision Trees



Partition the input space into homogeneous regions

Genetic Evolution Guided by Classification



~~Initial input~~

~~Fitness computation~~

~~Classification~~

~~Selection~~

Breeding

Engineers' Feedback

- The characterizations (decision trees) of the different critical regions can help with:

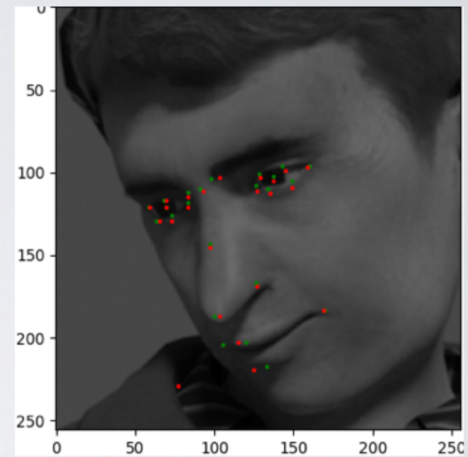
  (1) **Debugging** the system model

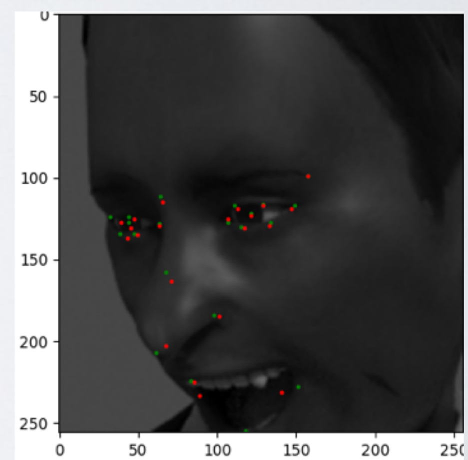
  (2) **Identifying possible hardware changes** to increase ADAS safety

  (3) **Providing proper warnings** to drivers

Key-points Detection

- Automatically **detecting key-points** in an image or a video, e.g., face recognition, drowsiness detection
  - Key-point Detection DNNs (KP-DNNs) are widely used to detect key-points in an image
- It is essential to check **how accurate** KP-DNNs are when applied to various test data



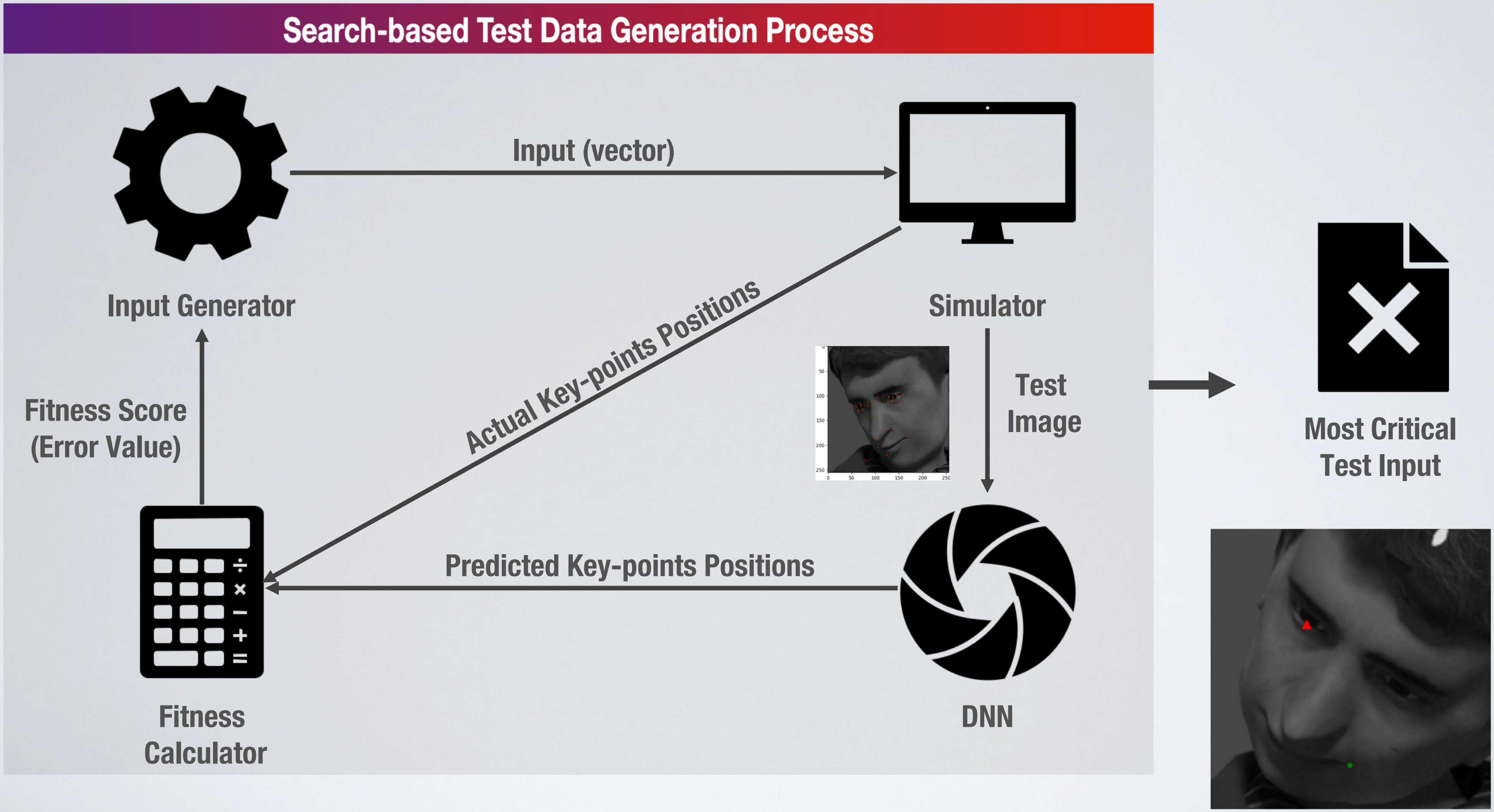


Ground truth
Predicted

Example Application

- **Drowsiness or gaze detection** based on interior camera monitoring the driver
- In the drowsiness or gaze detection problem, **each Key-Point (KP) may be highly important for safety**
  - **Each KP leads to a requirement** and test objective
  - For our subject DNN, we have **27 requirements**
- **Goal:** Cause the DNN to mis-predict as many key-points as possible
- **Solution:** Many-objective search algorithms combined with simulator

Overview


Search-based Test Data Generation Process
Input (vector)
Input Generator
Simulator
Actual Key-points Positions
Fitness Score
(Error Value)
Test
Image
Predicted Key-points Positions
Fitness
Calculator
DNN
Most Critical
Test Input

Results

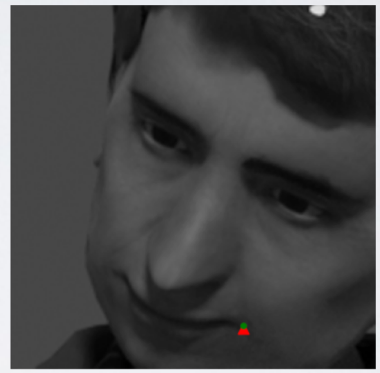
- Our approach is effective in generating test suites that cause the DNN to severely mispredict more than 93% of all key-points on average
- Not all mispredictions can be considered failures …
- Some key-points are more severely predicted than others, detailed analysis revealed two reasons:
  - Under-representation of some key-points (hidden) in the training data
  - Large variation in the shape and size of the mouth across different 3D models (more training needed)

Interpretation

Representative rules derived from the decision tree for KP26
(M: Model-ID, P: Pitch, R: Roll, Y: Yaw, NE: Normalized Error)

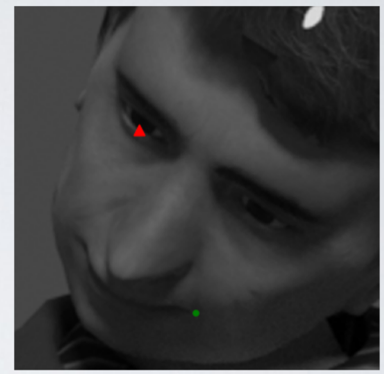
Image Characteristics Condition	NE
$M = 9 \land P < 18.41$	0.04
$M = 9 \land P \geq 18.41 \land R < -22.31 \land Y < 17.06$	0.26
$M = 9 \land P \geq 18.41 \land R < -22.31 \land 17.06 \leq Y < 19$	0.71
$M = 9 \land P \geq 18.41 \land R < -22.31 \land Y \geq 19$	0.36

(A) A test image satisfying the first condition



NE = 0.013

(B) A test image satisfying the third condition



NE = 0.89

- **Regression trees**
- **Detailed analysis to find the root causes of high NE values, e.g., shadow on the location of KP26 is the cause of high normalized (NE) values**
- **The average MAE from all the trees is 0.01 (far less than the pre-defined threshold: 0.05) with average tree size of 25.7. Excellent accuracy, reasonable size.**

Summary

- Effective search of the test input set based on evolutionary computing and machine learning.
- Mechanism to learn conditions leading to (safety) violations to enable risk analysis and improvements.

Surrogate Models

- **Online testing, coupled with a simulator**, is highly important in many domains, such as autonomous driving systems.
- E.g., more likely to find safety violations
- But online testing is **computationally expensive**
- **Surrogate model:** Model that mimics the simulator, to a certain extent, while being much less computationally expensive
- **Research:** Combine search with surrogate modeling to decrease the computational cost of testing (Ul Haq et al., ICSE 2022)

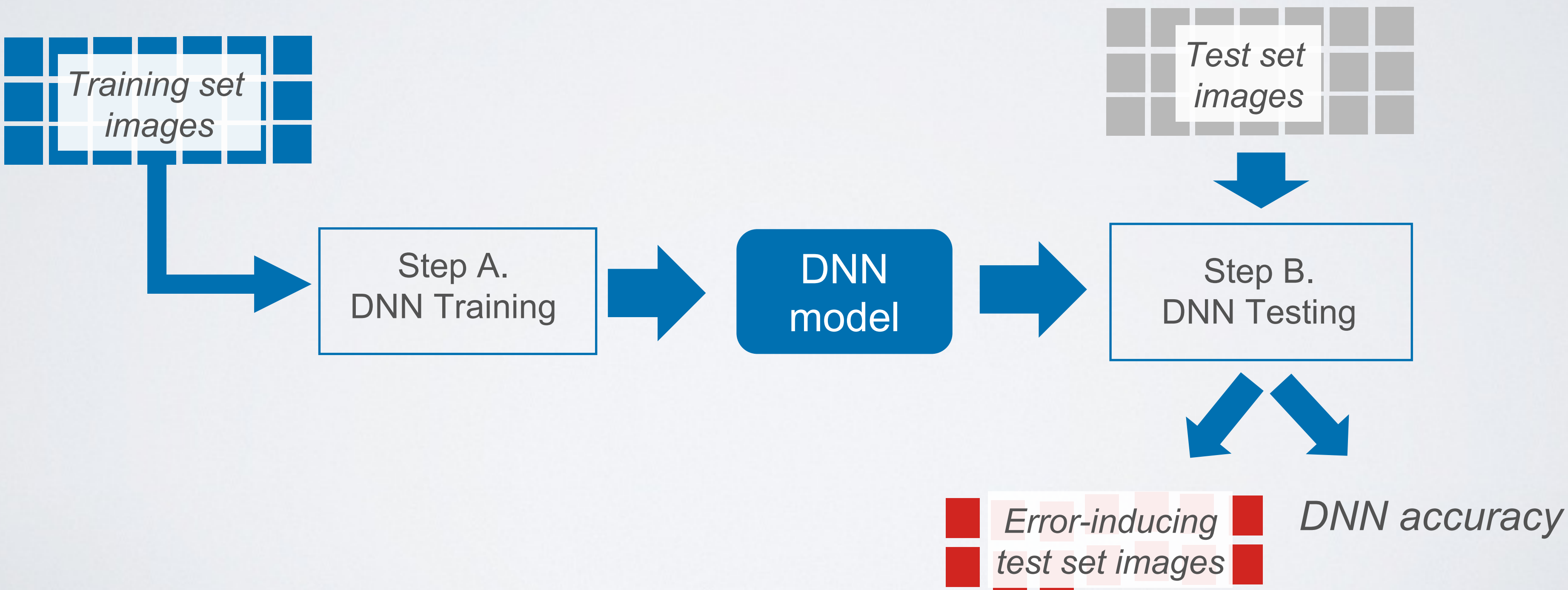
Safety Engineering in MLS

Objectives

- Understand **conditions of critical failures** in various settings
- Simulator: In terms of **configuration parameters**
- Real images: In terms of the presence of **concepts**
- Required for **risk assessment**
- **Research:** Techniques to achieve such understanding

Typical DNN Evaluation

- Example with images



Identification of Unsafe Situations

- Current practice is based on manual **root cause analysis**: identification of the **characteristics of the system inputs** that induce the DNN to generate erroneous results
    - manual inspection is **error prone (many images)**
    - automated identification of such characteristics is the objective of research on **DNN safety analysis approaches**

DNN Heatmaps

- Generate heatmaps that capture the extent to which the pixels of an image impacted a specific result

An heatmap can show that long hair is what caused a female doctor to be classified as nurse [Selvaraju'16]

Predicted: Nurse

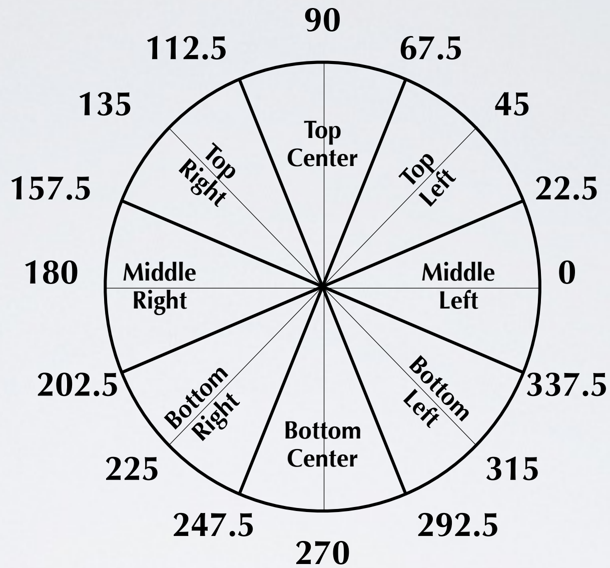
- Limitations:
  - Heatmaps should be manually inspected to determine the reason for misclassification
  - Underrepresented (but dangerous) failure causes might be unnoticed
  - DNN debugging (i.e., improvement) not automated

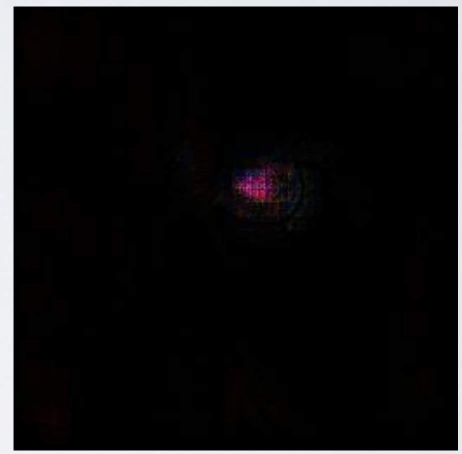
Heatmap-based Unsupervised Debugging of DNNs (HUDD)

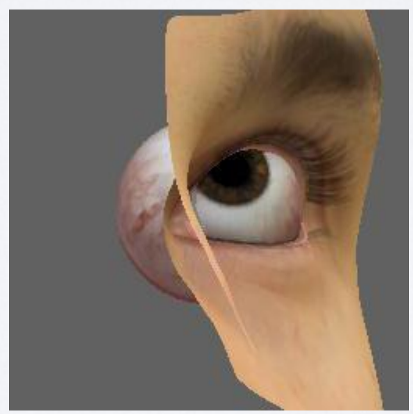
Rely on hierarchical agglomerative **clustering**
to identify the **distinct root causes** of DNN errors in
the **heatmaps of internal DNN layers**
and use this information
to automatically **retrain** the DNN

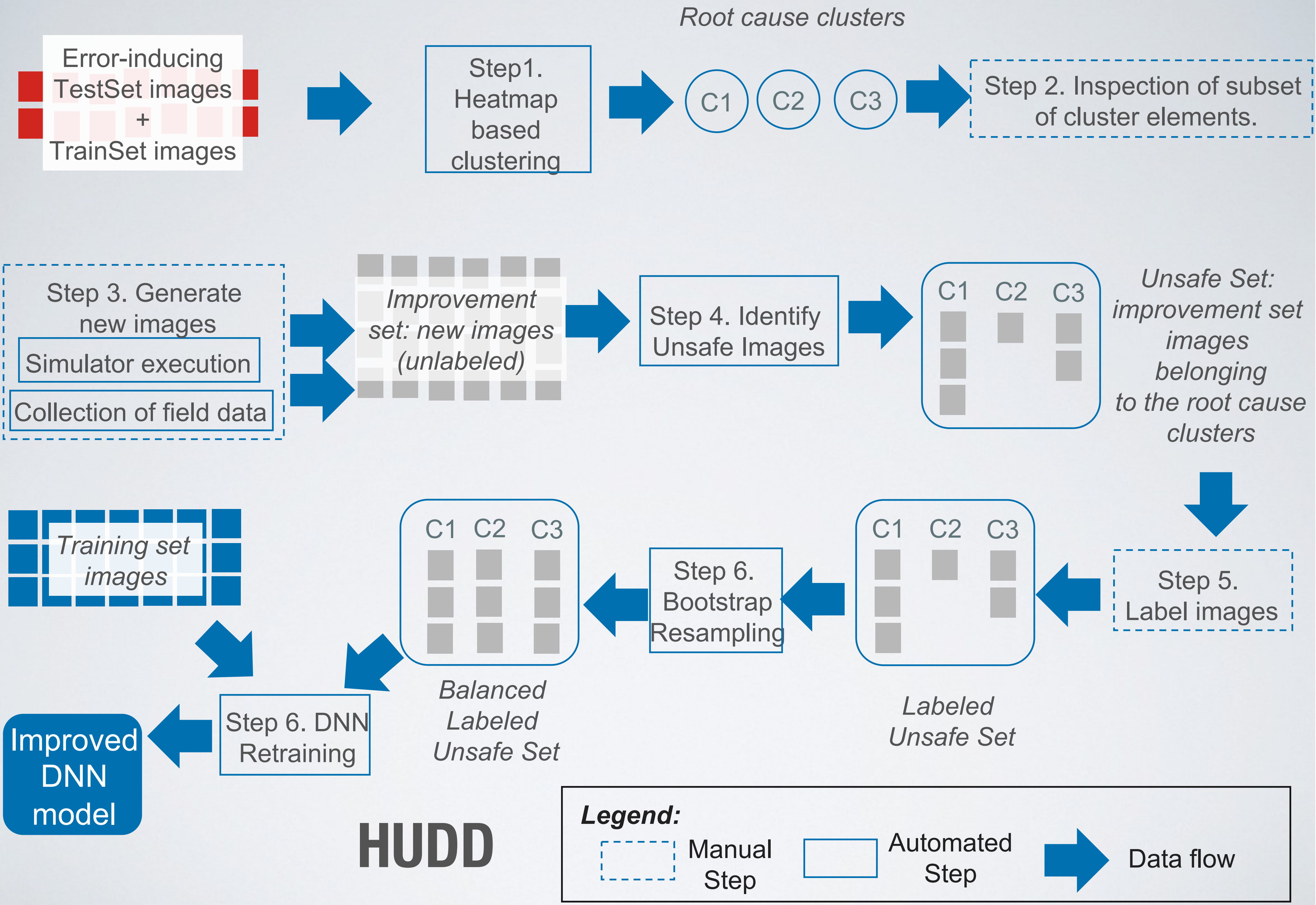
Example Application

- Classification
  - **Gaze Detection**

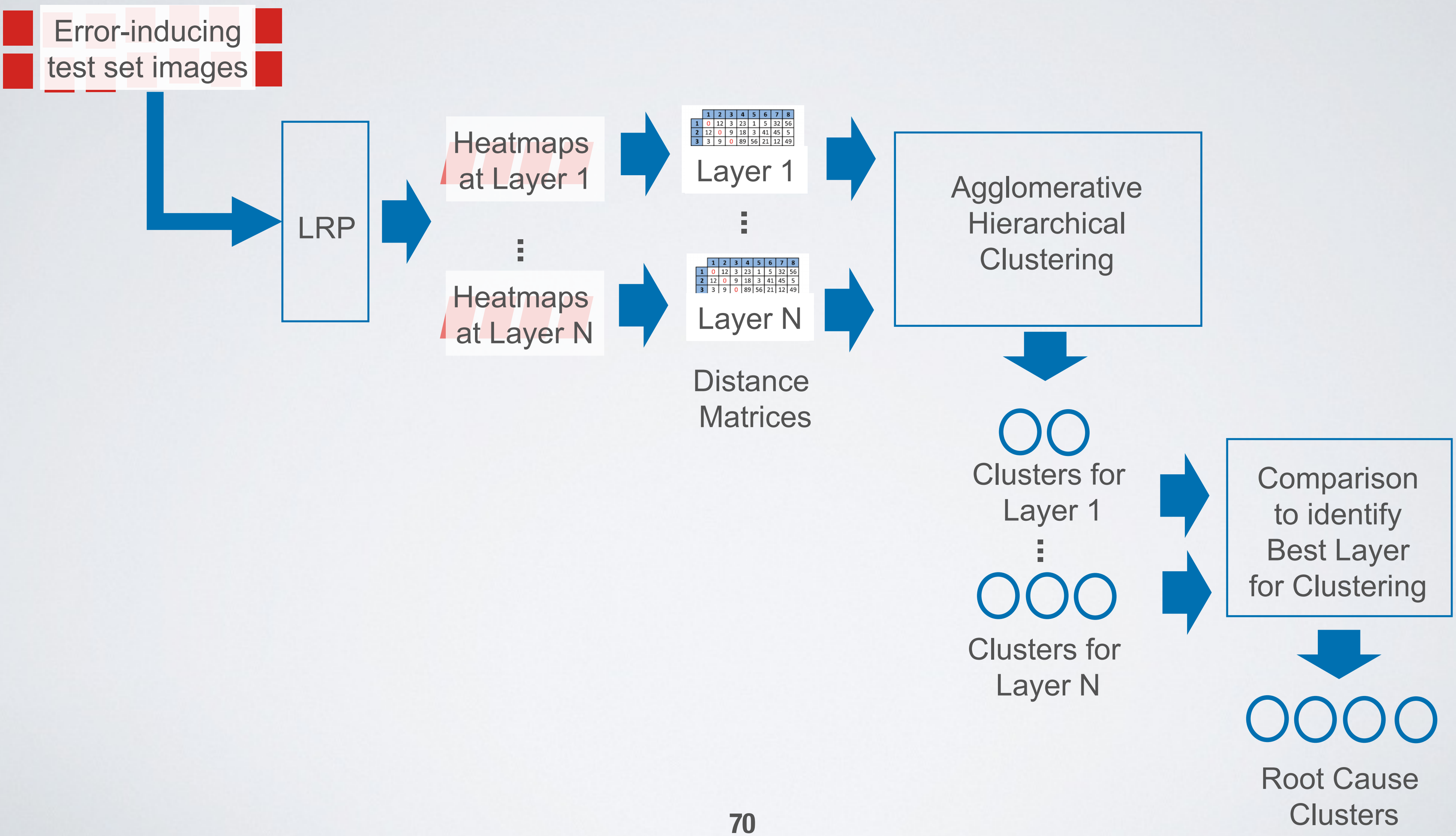







Root cause clusters
Error-inducing TestSet images + TrainSet images
Step1. Heatmap based clustering
C1
C2
C3
Step 2. Inspection of subset of cluster elements.
Step 3. Generate new images
Simulator execution
Collection of field data
Improvement set: new images (unlabeled)
Step 4. Identify Unsafe Images
C1 C2 C3
Unsafe Set: improvement set images belonging to the root cause clusters
Step 5. Label images
C1 C2 C3
Labeled Unsafe Set
Step 6. Bootstrap Resampling
C1 C2 C3
Balanced Labeled Unsafe Set
Training set images
Step 6. DNN Retraining
Improved DNN model
HUDD
Legend:
Manual Step
Automated Step
Data flow

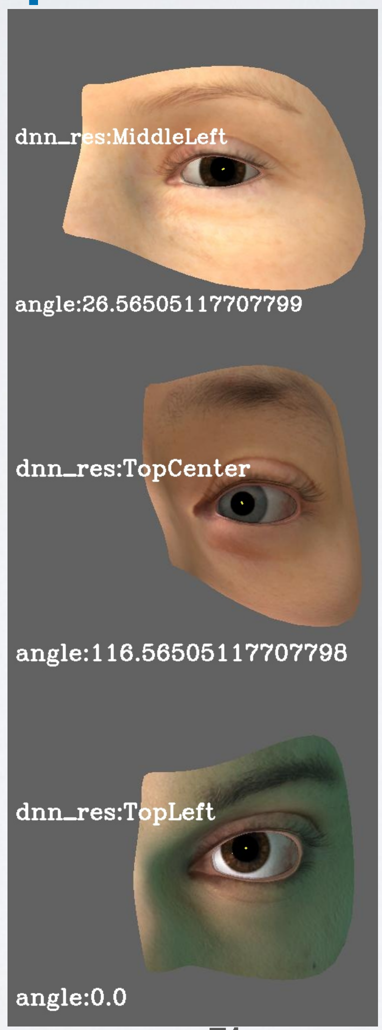
Heatmap Clustering



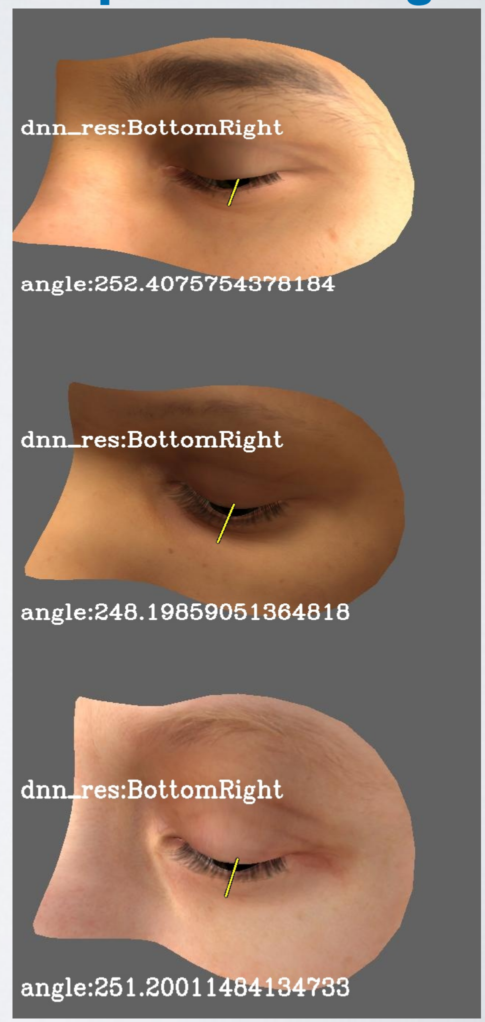
Clusters identify different problems

Cluster 1
(angle ~157.5)
borderline cases

Cluster 2
(eye middle center)
incomplete set of classes



Cluster3
(closed eyes)
incomplete training set



Summary

- Mechanism to group mispredicted/misclassified images whose error root causes are similar.
- This is a basis to better understand the root causes of DNN errors, assess risks, and retrain them to improve their accuracy.
- Current work: Black-box approach based on feature extraction

Conclusions

Testing Community

- It contributes by adapting techniques from classical software testing
- SBST
- Adequacy criteria
- Metamorphic testing
- Mutation analysis
- Empirical methodology for software testing

Re-Focus Research (1)

- But, as usual, research is taking the path of least resistance but we need to shift the focus to increase impact
- More focus on integration and system testing for MLS
- Not only model accuracy, but model-induced risks within a system
- Safety engineering in MLS
- More focus on practical and scalable black-box approaches

Re-Focus Research (2)

- **Scalable online testing** for autonomous systems
- **Scalability** issues due to simulations and large DNNs
- Beyond the perception layer, the **control aspects** need to be considered as well
- **Beyond stateless models (DNN)**: Reinforcement learning …

Trustworthy Machine Learning-Enabled Systems

Lionel Briand

http://www.lbriand.info

IFIP WG 10.4, 2022

References

Selected References

- Briand et al. "Testing the untestable: model testing of complex software-intensive systems.", international conference on software engineering (companion), 2016.
- Ul Haq et al. "Comparing offline and online testing of deep neural networks: An autonomous car case study." IEEE 13th International Conference on Software Testing, Validation and Verification (ICST), 2020.
- Ul Haq et al. "Can Offline Testing of Deep Neural Networks Replace Their Online Testing?." Empirical Software Engineering (Springer), 2021
- Ul Haq et al. "Automatic Test Suite Generation for Key-points Detection DNNs Using Many-Objective Search." ACM International Symposium on Software Testing (ISSTA), 2021
- Ul Haq et al., "Efficient Online Testing for DNN-Enabled Systems using Surrogate-Assisted and Many-Objective Optimization." IEEE/ACM ICSE 2022
- Fahmy et al. "Supporting DNN Safety Analysis and Retraining through Heatmap-based Unsupervised Learning." IEEE Transactions on Reliability, Special section on Quality Assurance of Machine Learning Systems, 2021
- Ben Abdessalem et al., "Testing Vision-Based Control Systems Using Learnable Evolutionary Algorithms", ICSE 2018
- Ben Abdessalem et al., "Testing Autonomous Cars for Feature Interaction Failures using Many-Objective Search", ASE 2018

Selected References

- Goodfellow et al. "Explaining and harnessing adversarial examples." arXiv preprint arXiv:1412.6572 (2014).
- Zhang et al. "DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems." In 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE), 2018.
- Tian et al. "DeepTest: Automated testing of deep-neural-network-driven autonomous cars." In Proceedings of the 40th international conference on software engineering, 2018.
- Li et al. “Structural Coverage Criteria for Neural Networks Could Be Misleading”, IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (NIER)
- Kim et al. "Guiding deep learning system testing using surprise adequacy." In IEEE/ACM 41st International Conference on Software Engineering (ICSE), 2019.
- Ma et al. "DeepMutation: Mutation testing of deep learning systems." In 2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE), 2018.
- Zhang et al. "Machine learning testing: Survey, landscapes and horizons." IEEE Transactions on Software Engineering (2020).
- Riccio et al. "Testing machine learning based systems: a systematic mapping." Empirical Software Engineering 25, no. 6 (2020)
- Gerasimou et al., “Importance-Driven Deep Learning System Testing”, IEEE/ACM 42nd International Conference on Software Engineering, 2020

Backup

Testing in ISO 26262

- Several recommendations for testing at the unit and system levels
  - e.g., Different structural coverage metrics, black-box testing
- However, such testing practices are not adequate for MLS
  - The **input space** of ADAS is much **larger** than traditional automotive systems.
  - No **specifications or code** for DNN components.
  - MLS may fail without the presence of a **systematic fault**, e.g., inherent limitations, incomplete training.
  - Imperfect **environment simulators**.
  - **Traditional testing notions** (e.g., coverage) are not clear for DNN components.

SOTIF

- ISO/PAS 21448:2019 standard: Safety of the intended functionality (SOTIF).
- Autonomy: Huge increase in functionalities relying on advanced sensing, algorithms (ML), and actuation.
- SOTIF accounts for limitations and risks related to nominal performance of sensors and software:
  - The inability of the function to correctly comprehend the situation and operate safely; this also includes functions that use machine learning algorithms;
  - Insufficient robustness of the function with respect to sensor input variations or diverse environmental conditions.