Engineering Trustworthy AI Systems

Foutse Khomh, PhD, Ing.
foutse.khomh@polymtl.ca
@SWATLab
Canada CIFAR AI Chair on Trustworthy Machine Learning Systems
FRQ-IVADO Research Chair on Software Quality Assurance for Machine Learning Applications

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

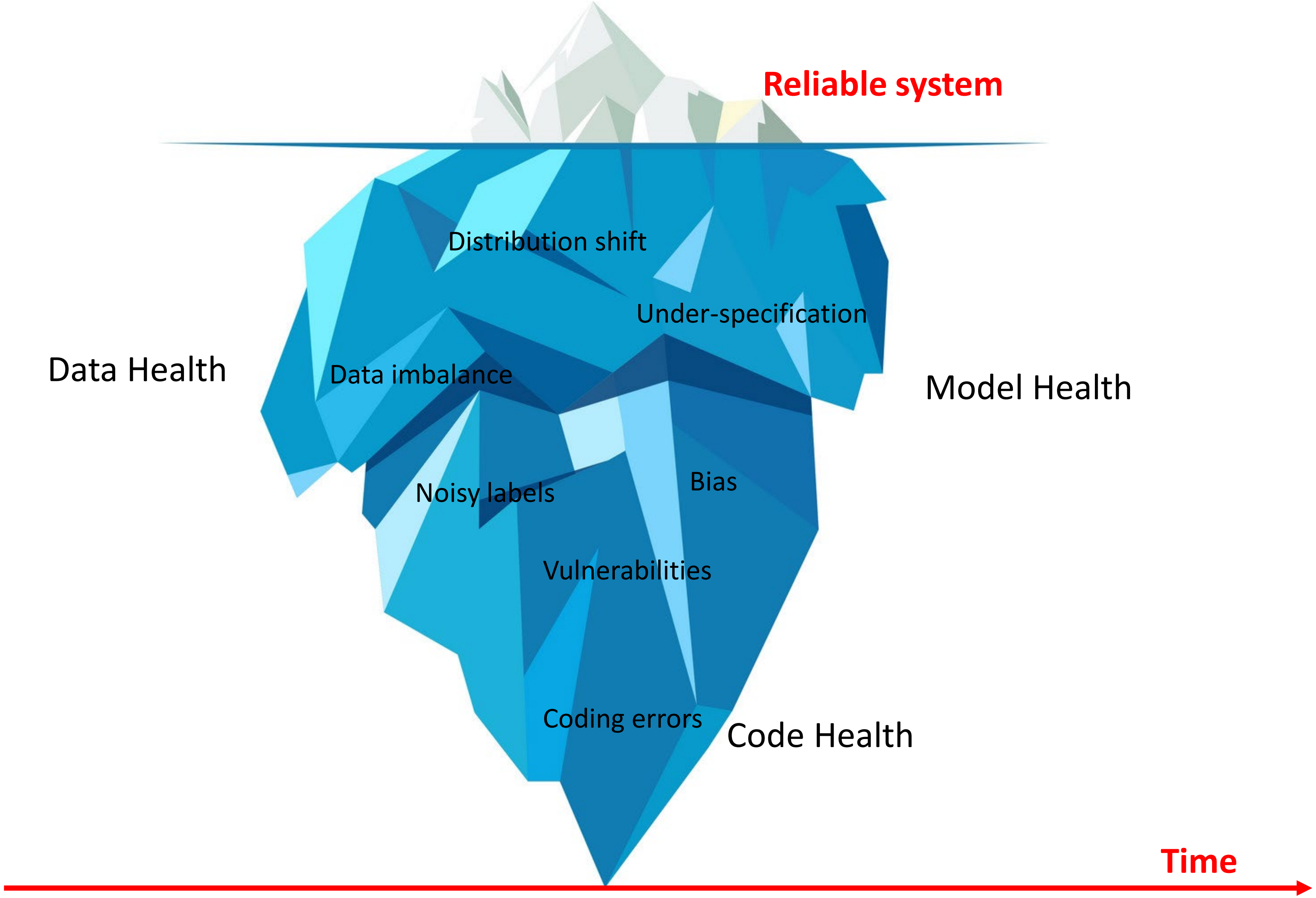
Mila

SEMLA
Trustworthy Engineering of AI Software

S.W.A.T
SoftWare Analytics and Technologies Lab

Engineering Trustworthy AI systems

System evolution & continuous delivery



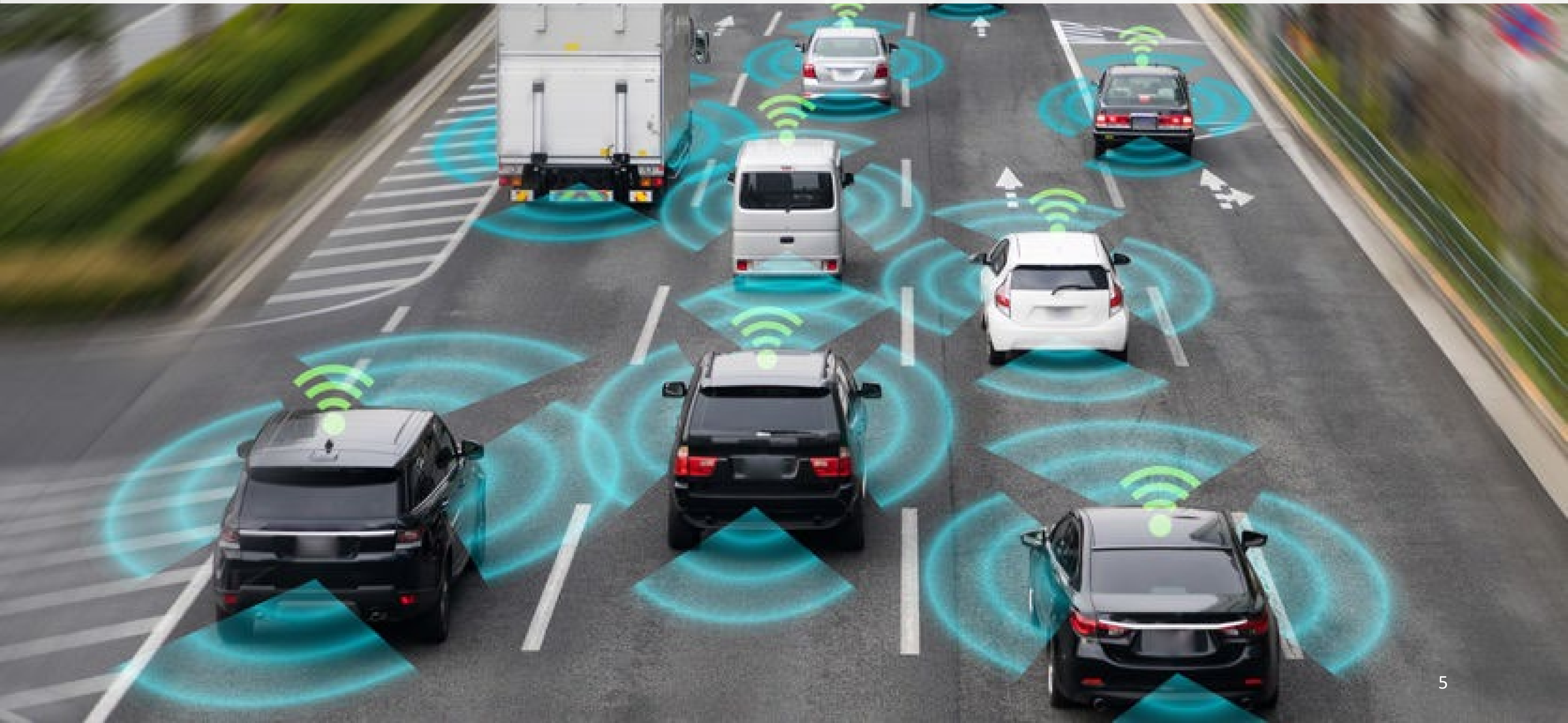
Some Team Members

Engineering Trustworthy AI systems requires

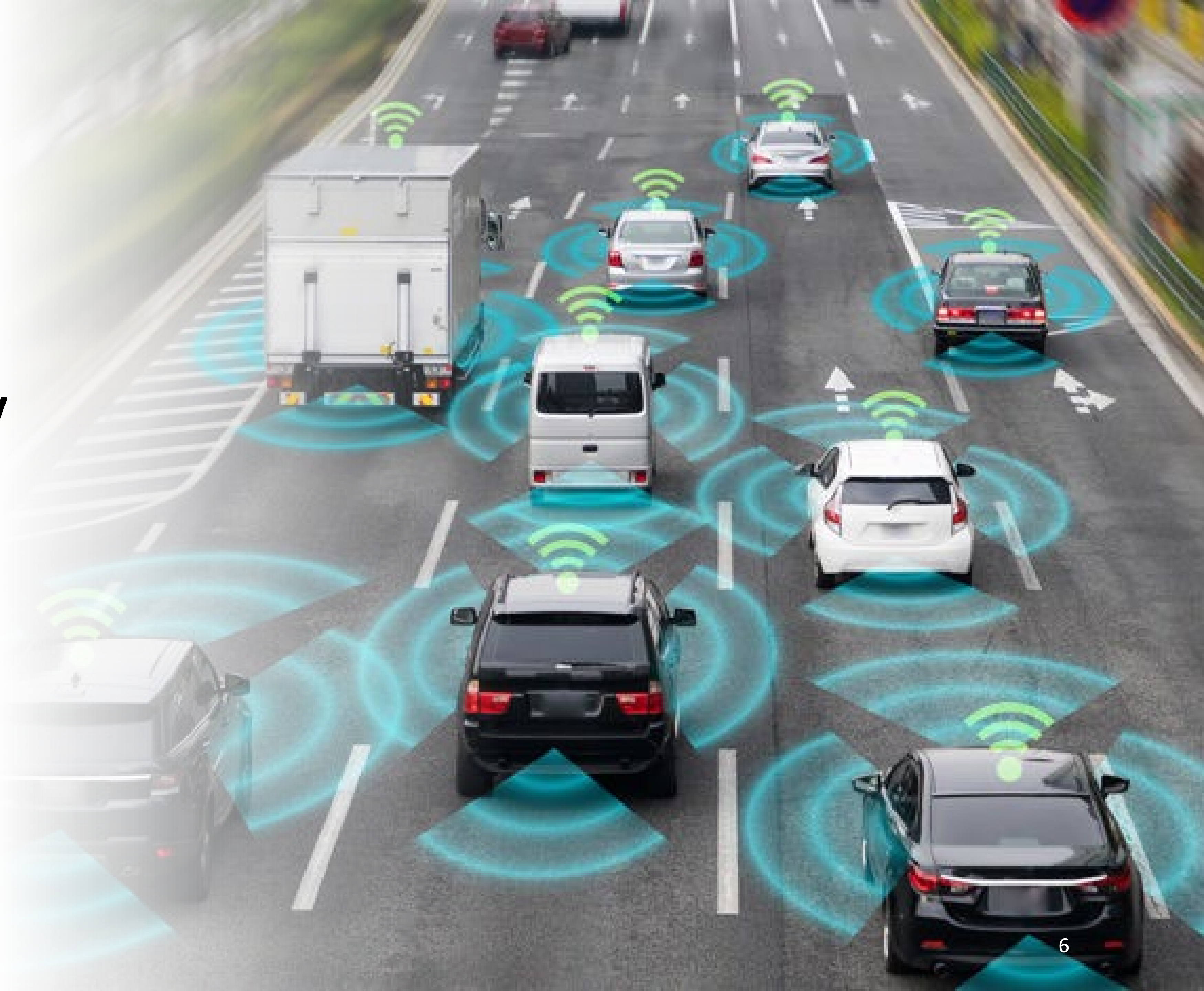
Developing AI models and algorithms that are **not only accurate**, but also :

- ✓ **Explainable,**
- ✓ **Fair,**
- ✓ **Privacy-preserving,**
- ✓ **Causal, and**
- ✓ **Robust.**

Autonomous Driving Systems are expected to change mobility

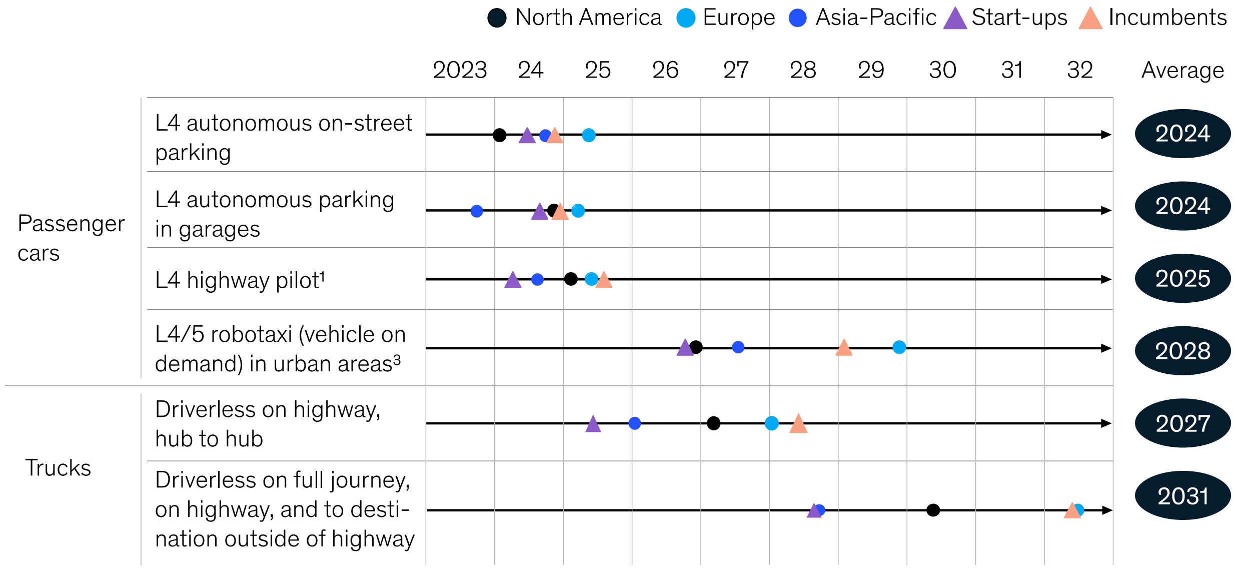


- **Improved road safety**
- **Increase Productivity**
- **Increased accessibility**
- **Reduce Costs?**
- **Reduce Congestion?**



"By 2035, autonomous driving could create $300 billion to $400 billion in revenue."

Most survey respondents expect L4 use cases to emerge by 2024 or 2025.



[1]Driver can use the time on highways for work or leisure activities using in-car or own solutions but needs to take over at highway exits.
[2]Driver can use the time on highways in urban environments for work or leisure activities using in-car or own solutions but may require some driver intervention.
[3]Robotaxis drive everywhere fully automated with no driver and accept and conduct transportation requests (goods, passengers). Passenger can use the travel time for work or leisure activities.
Question: In your estimation, what is the rollout (ie, commercial availability of vehicles/service) timeline for autonomous driving across use cases in your region?
Source: 75 respondents (North America, n = 31; Europe, n = 33; Asia–Pacific, n = 11)

McKinsey & Company

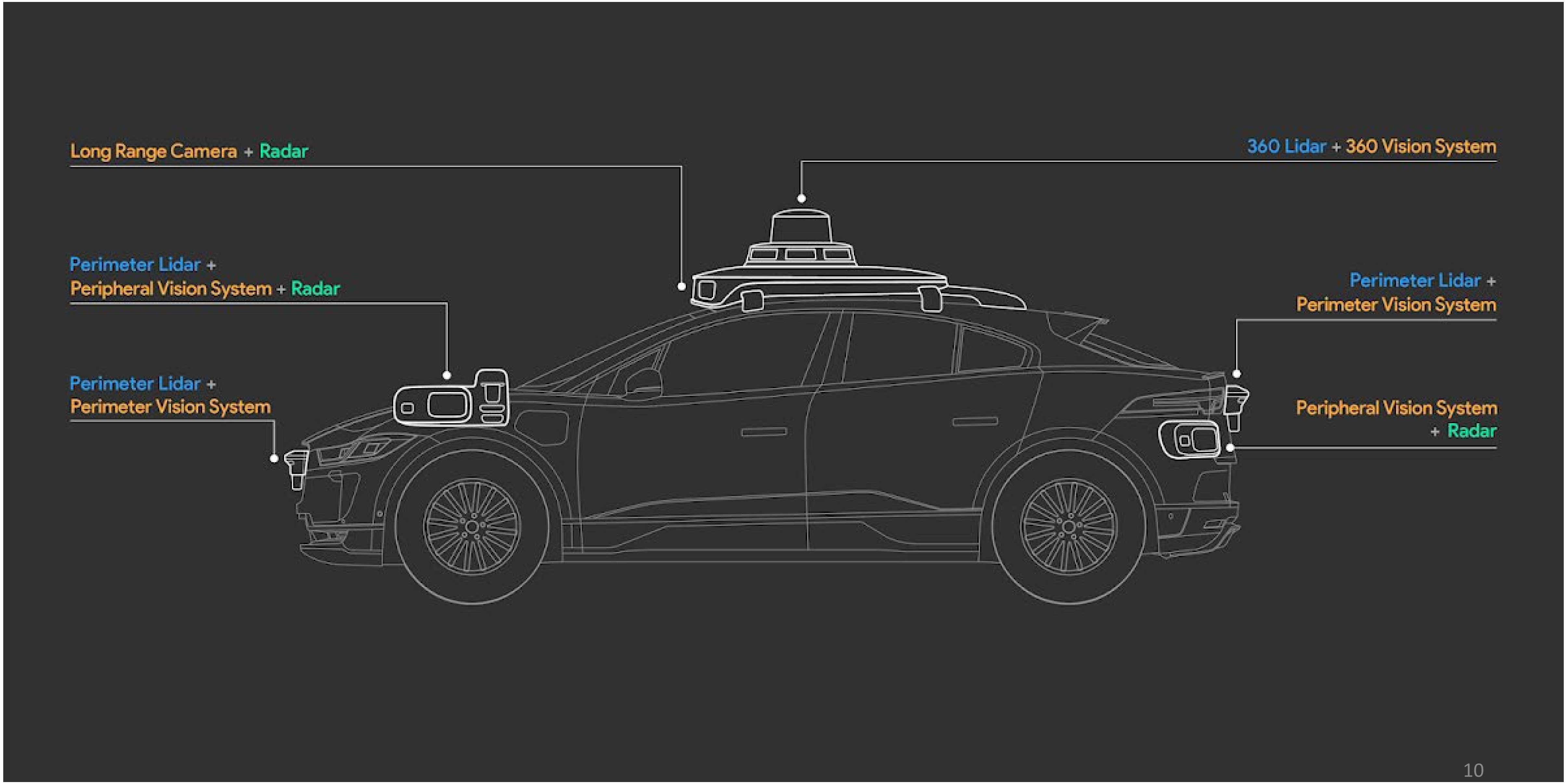
SAE J3016™ LEVELS OF DRIVING AUTOMATION™

Learn more here: sae.org/standards/content/j3016_202104

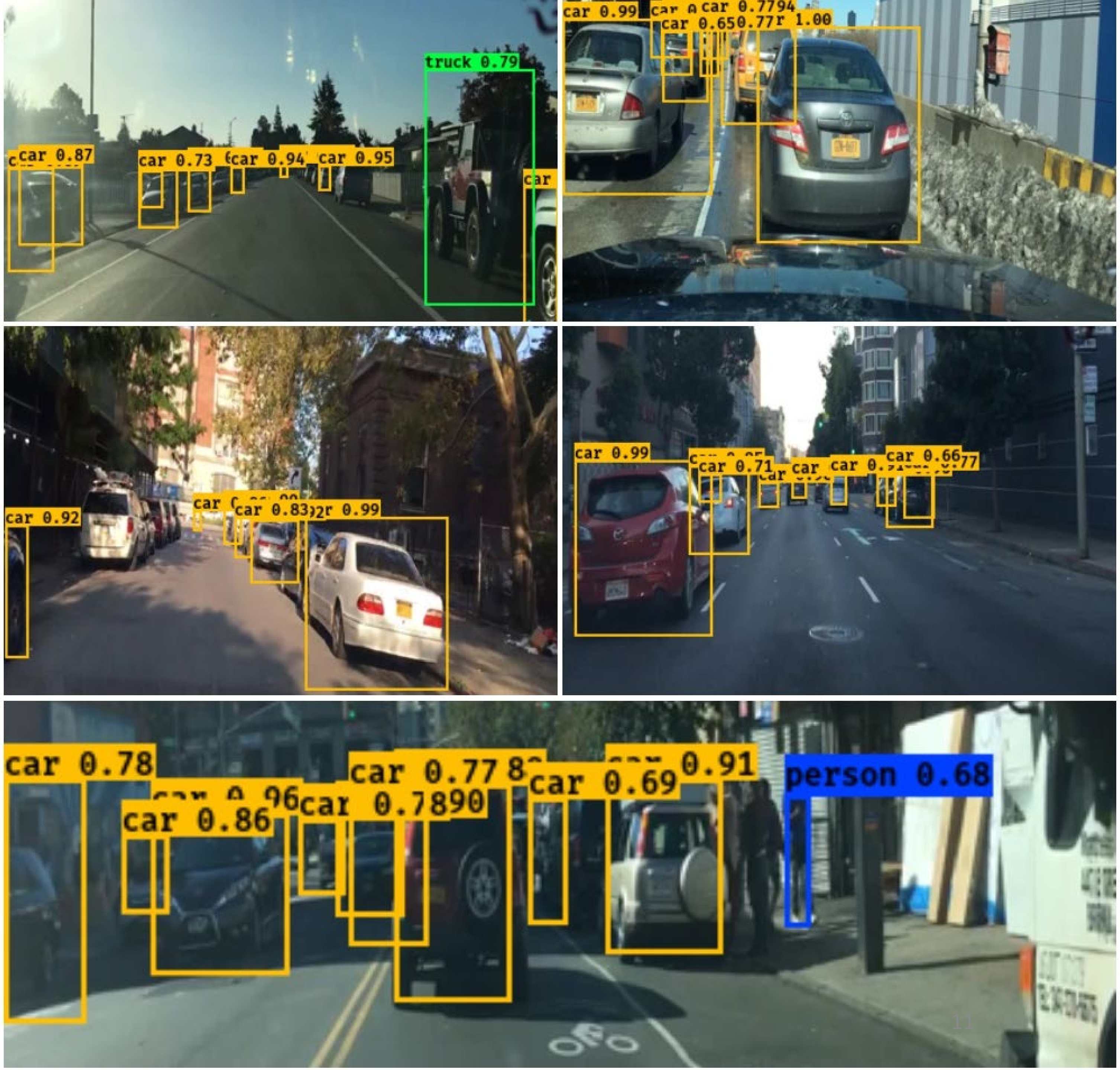
	SAE LEVEL 0™	SAE LEVEL 1™	SAE LEVEL 2™	SAE LEVEL 3™	SAE LEVEL 4™	SAE LEVEL 5™
What does the human in the driver's seat have to do?	**You are driving** whenever these driver support features are engaged – even if your feet are off the pedals and you are not steering			**You are not driving** when these automated driving features are engaged – even if you are seated in "the driver's seat"		
	You must constantly supervise these support features; you must steer, brake or accelerate as needed to maintain safety			When the feature requests, you must drive	These automated driving features will not require you to take over driving	
	These are driver support features			These are automated driving features		
What do these features do?	These features are limited to providing warnings and momentary assistance	These features provide steering OR brake/ acceleration support to the driver	These features provide steering AND brake/ acceleration support to the driver	These features can drive the vehicle under limited conditions and will not operate unless all required conditions are met		This feature can drive the vehicle under all conditions
Example Features	• automatic emergency braking • blind spot warning • lane departure warning	• lane centering OR • adaptive cruise control	• lane centering AND • adaptive cruise control at the same time	• traffic jam chauffeur	• local driverless taxi • pedals/ steering wheel may or may not be installed	• same as level 4, but feature can drive everywhere in all conditions

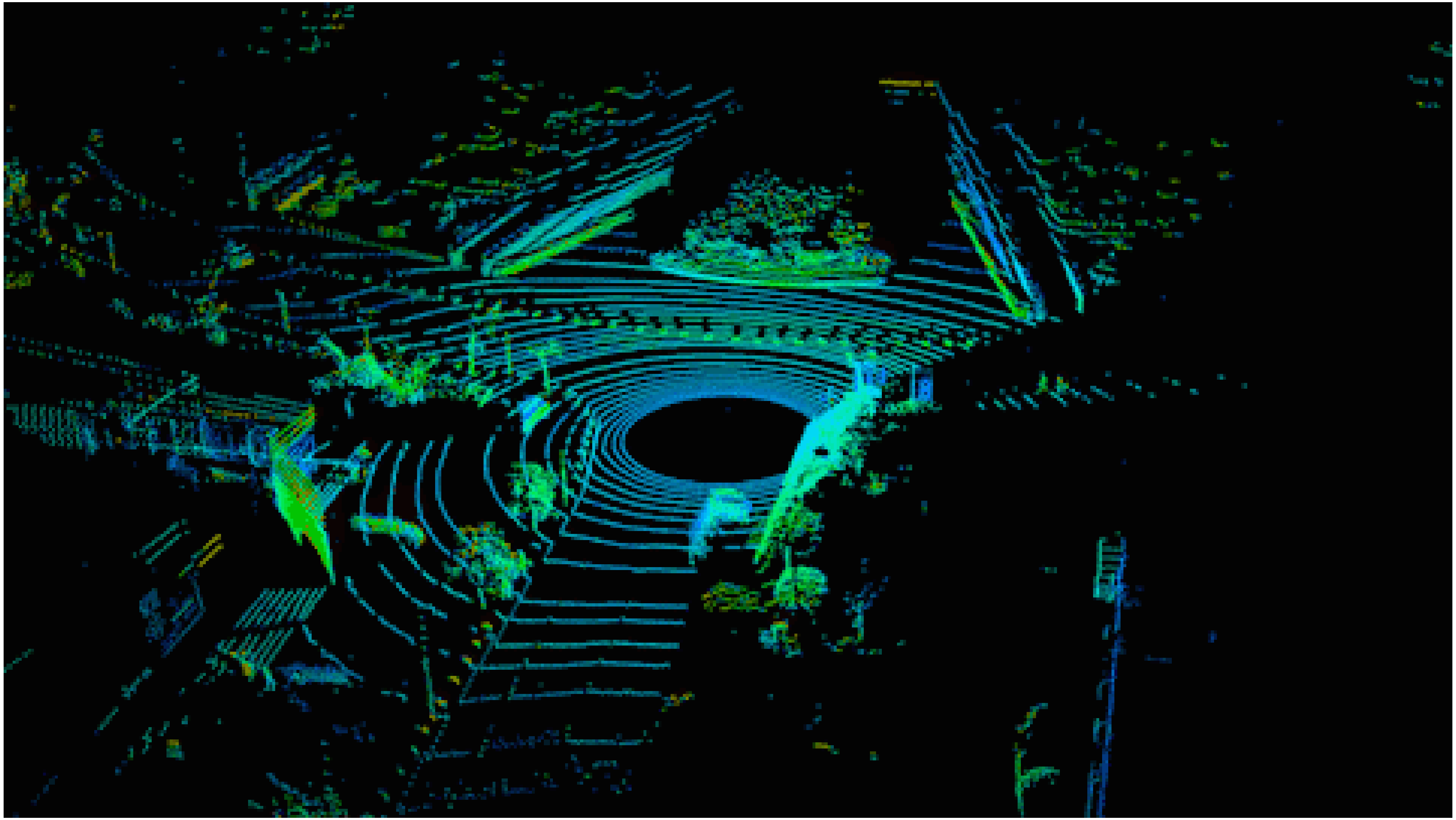
AI is leading the way for the launch of Level 4/5 autonomous vehicles

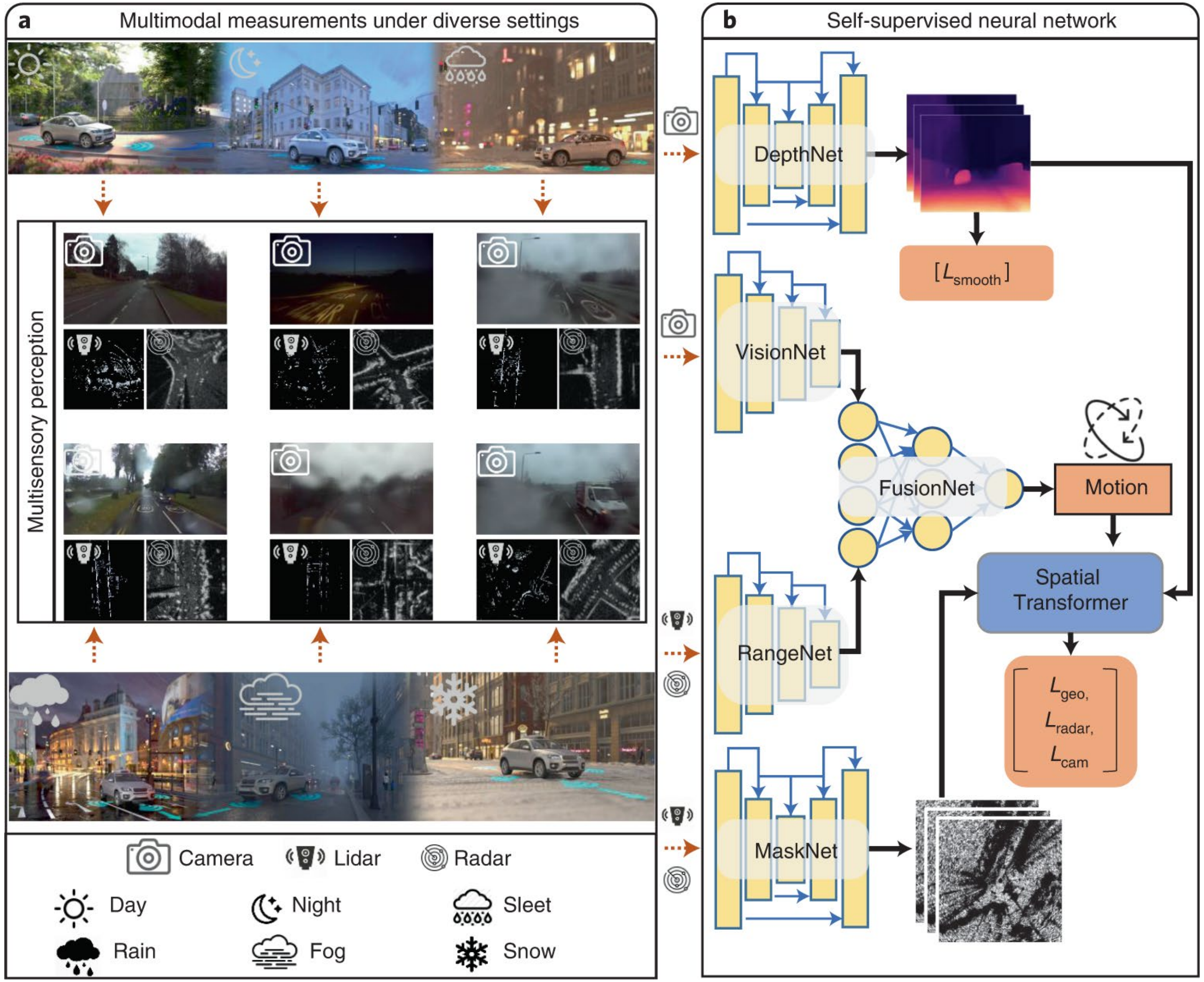
A Typical Autonomous Driving Car today!



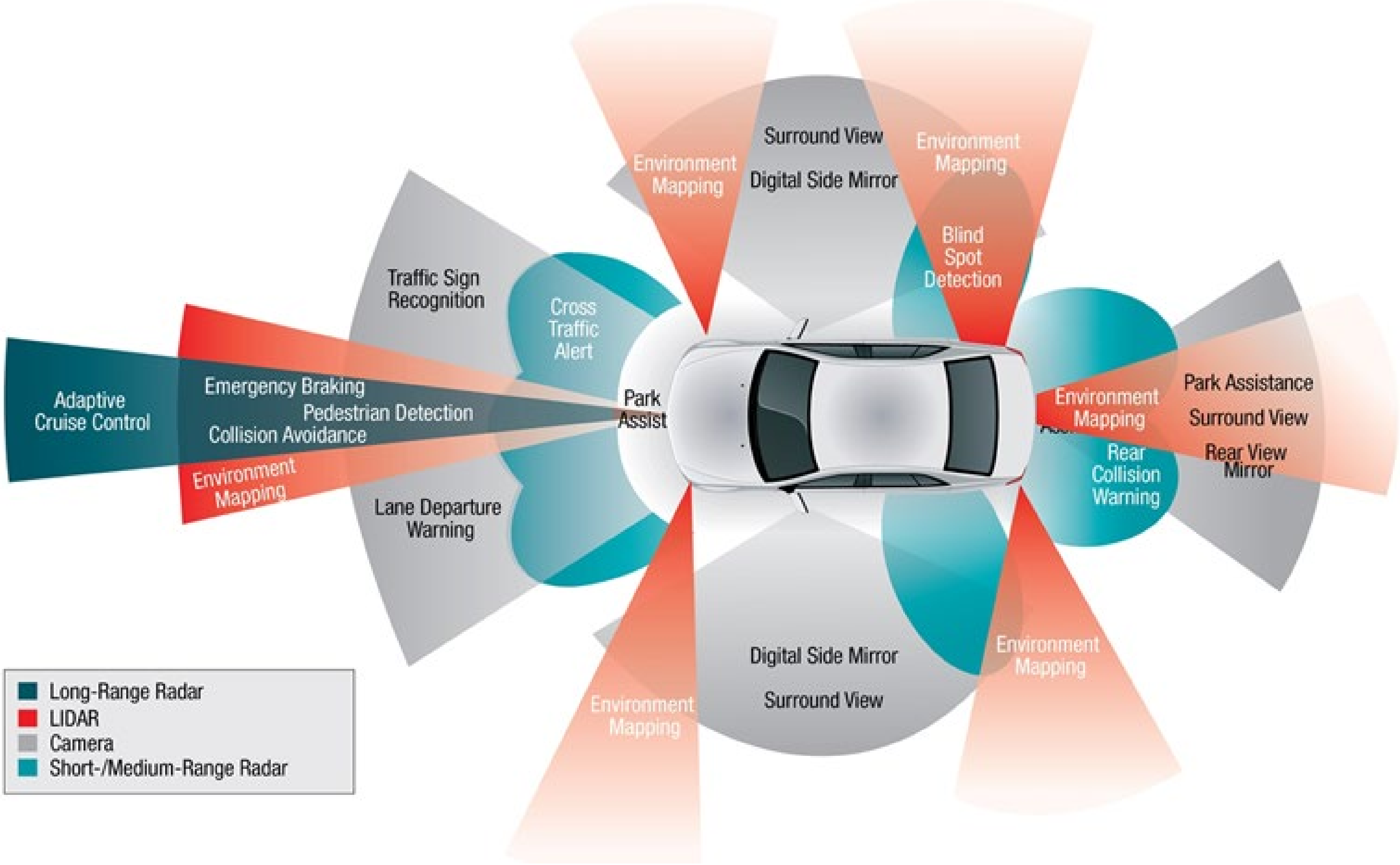
Neural Networks are at the core of their perception system!



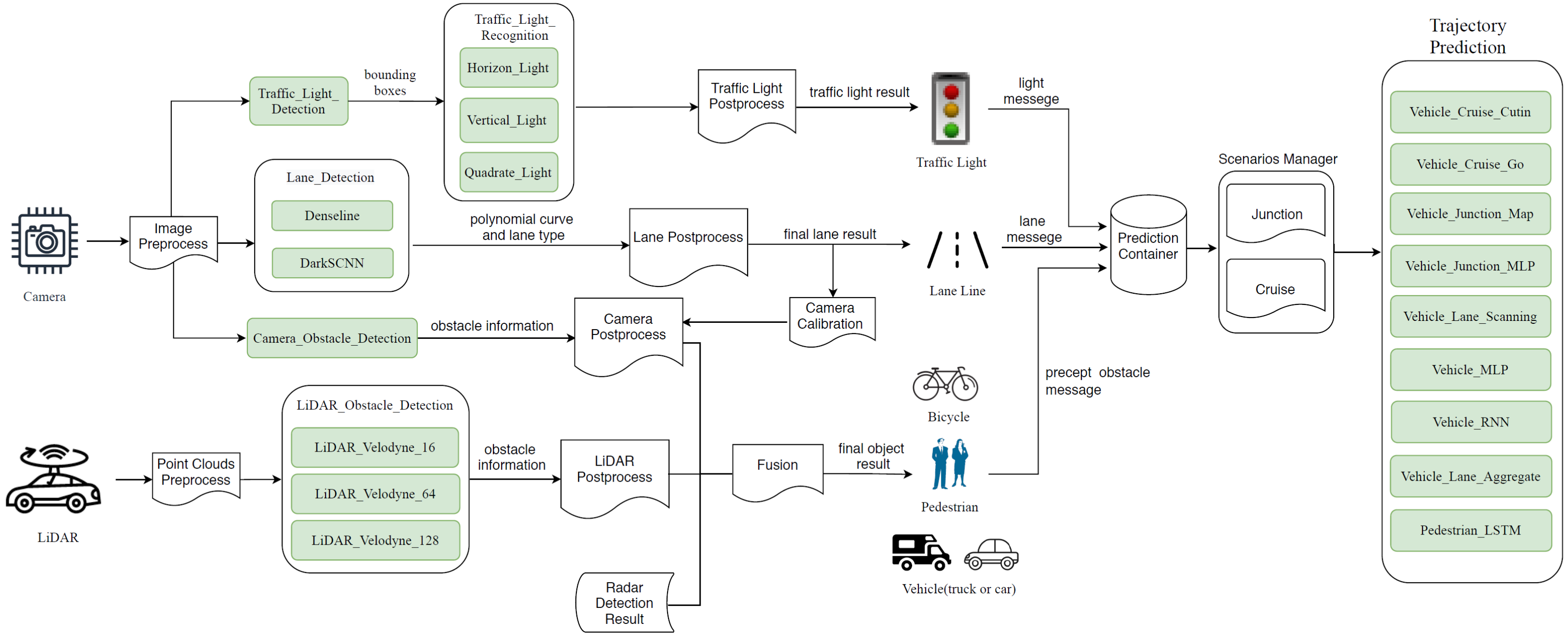


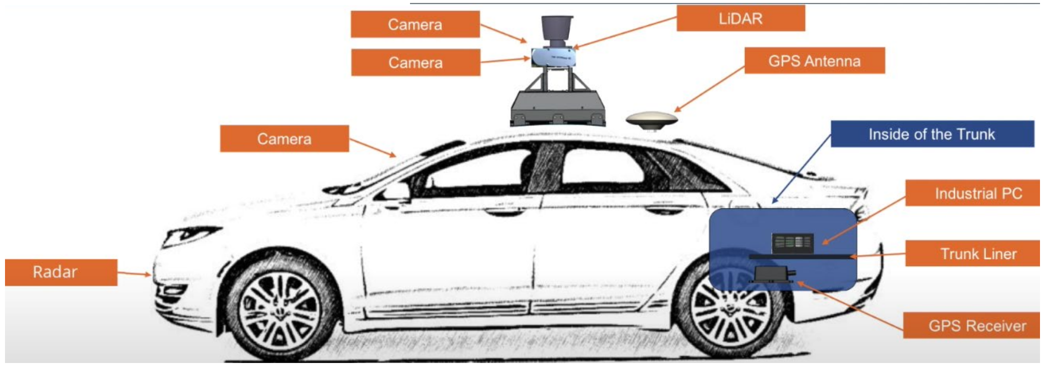


https://www.nature.com/articles/s42256-022-00520-5

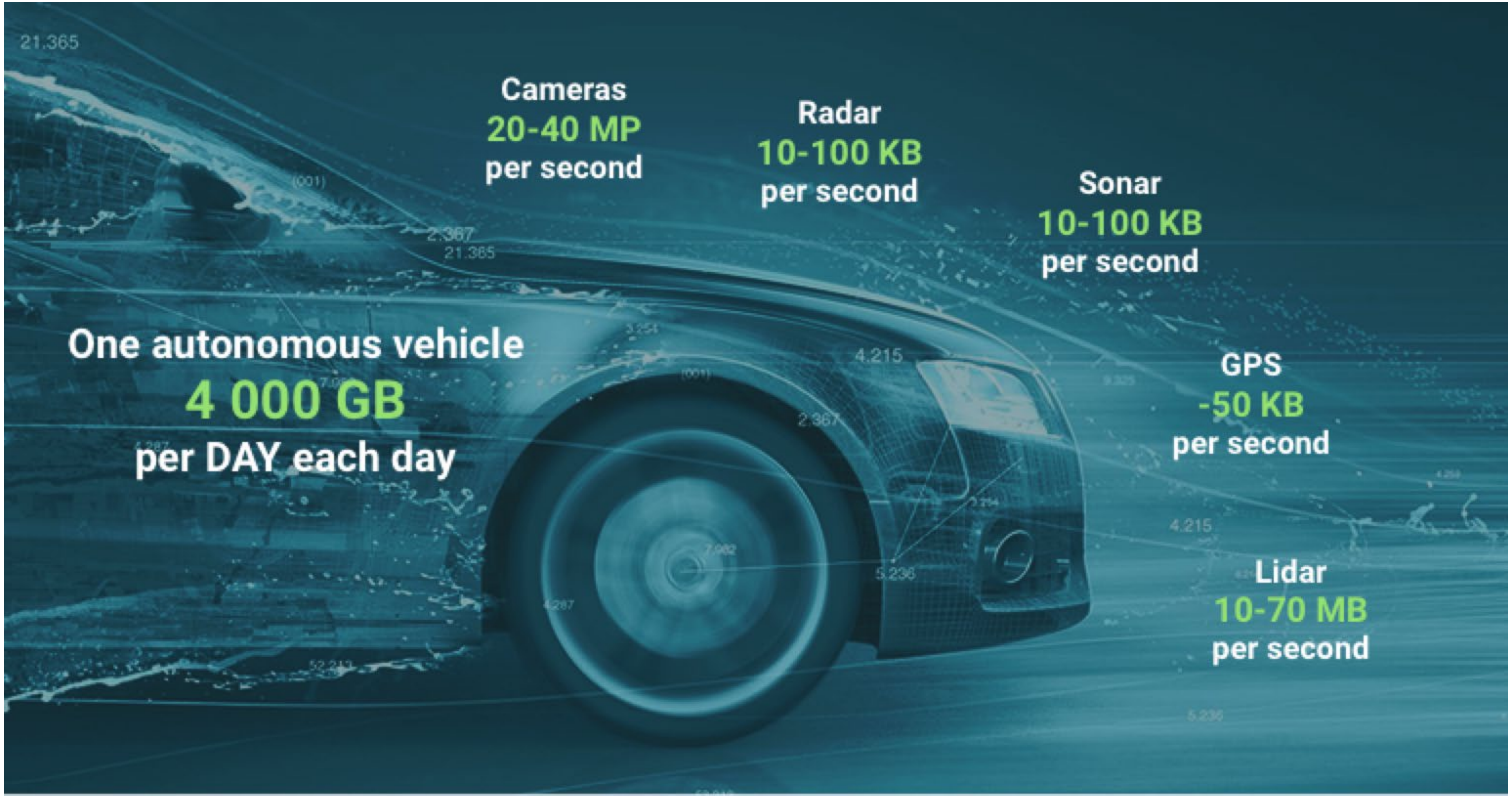

Adaptive Cruise Control
Emergency Braking
Pedestrian Detection
Collision Avoidance
Environment Mapping
Traffic Sign Recognition
Cross Traffic Alert
Lane Departure Warning
Park Assist
Environment Mapping
Surround View
Digital Side Mirror
Environment Mapping
Blind Spot Detection
Environment Mapping
Park Assistance
Surround View
Rear View Mirror
Rear Collision Warning
Environment Mapping
Digital Side Mirror
Surround View
Environment Mapping
Long-Range Radar
LIDAR
Camera
Short-/Medium-Range Radar

The Apollo Autonomous Driving System



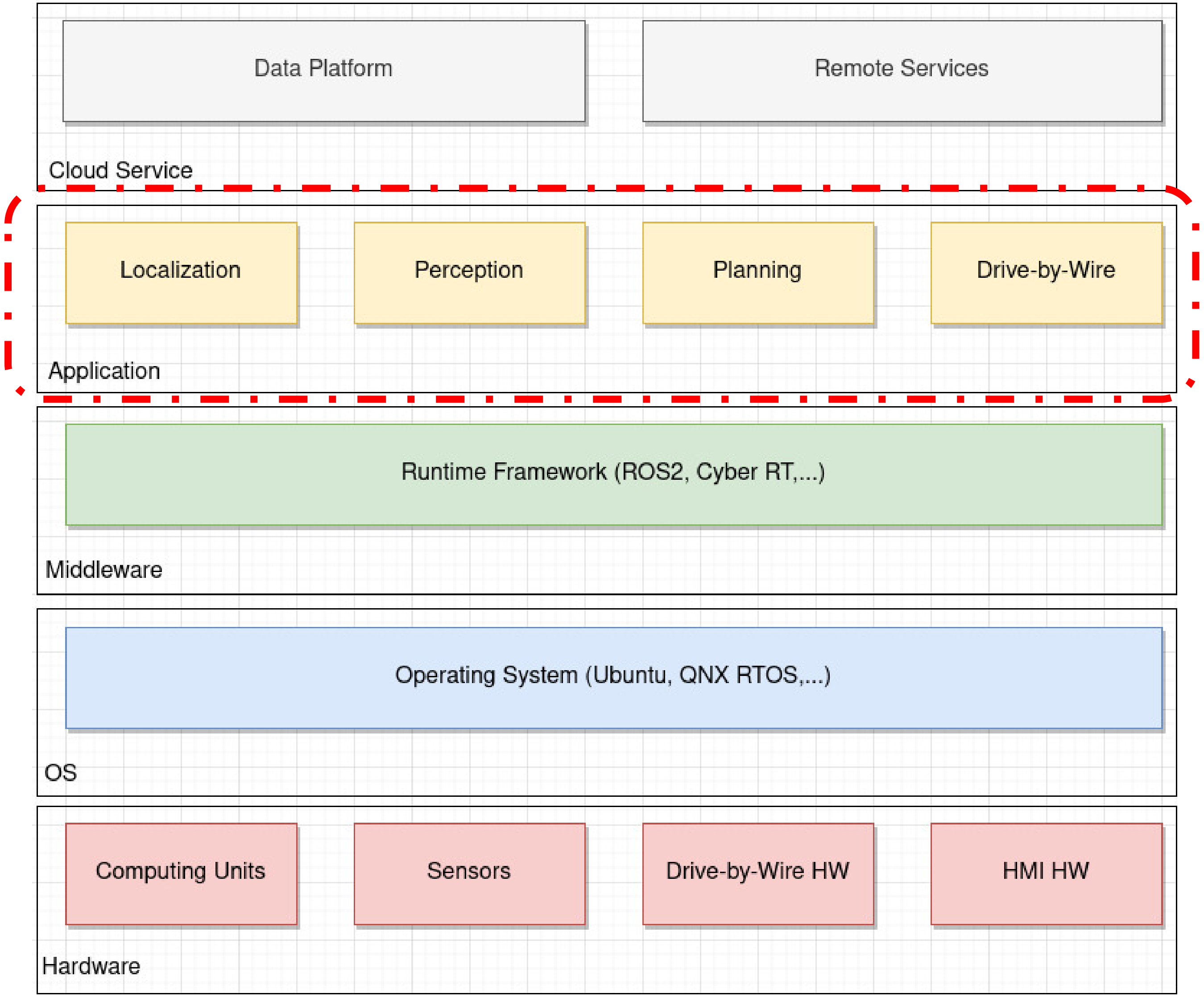


https://dl.acm.org/doi/pdf/10.1145/3368089.3417063



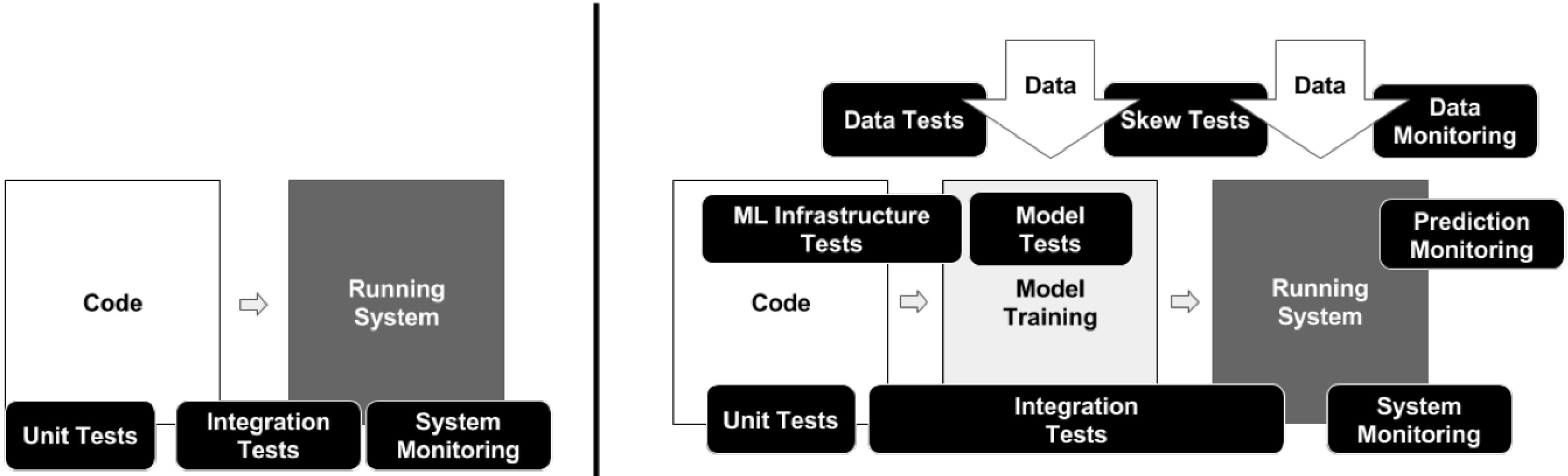
A single autonomous car will produce more data in a year than the roughly 320 million monthly users of Twitter create (Kastrenakes, 2019; Matthews, 2018)

Architecture of an Autonomous Driving system



https://channgo2203.github.io/av_software/

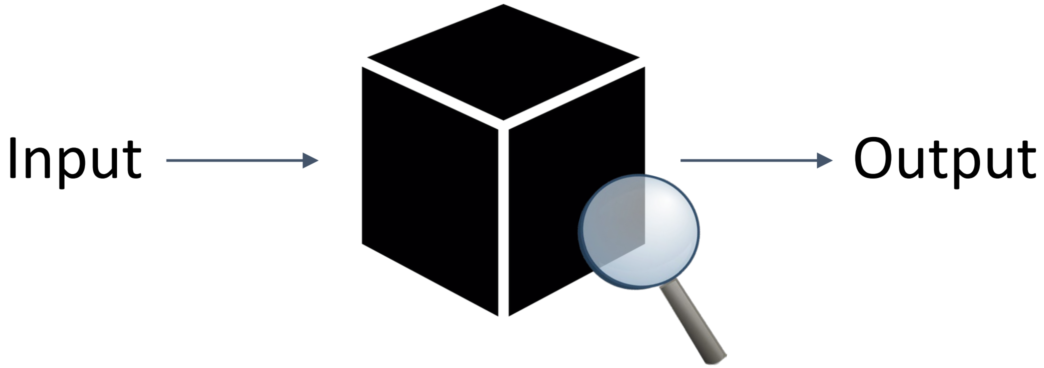
Ensuring the safety and auditability of ML-based components is challenging



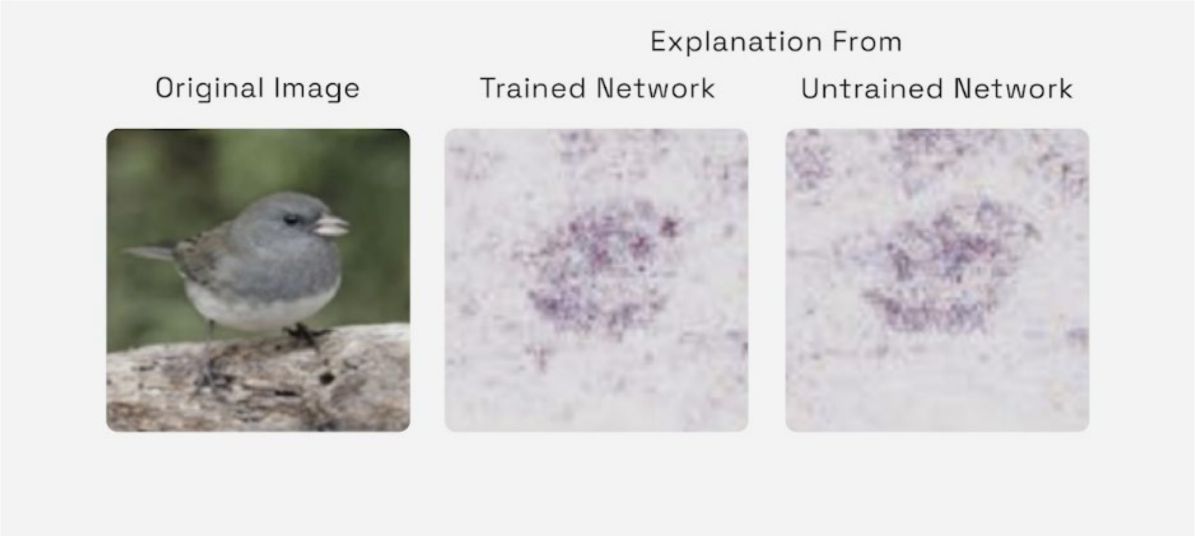
Eric Breck Shanqing Cai Eric Nielsen Michael Salib D. Sculley Proceedings of IEEE Big Data (2017)

Ensuring the auditability of ML-based components is challenging because…

- Current state-of-the-art models are hard to interpret (i.e., black box)

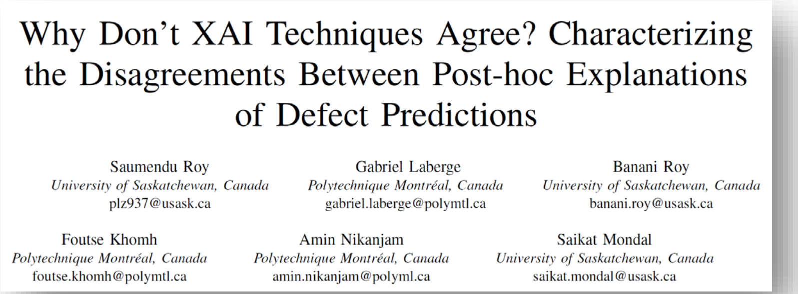


Moreover, current popular explanation methods are unfortunately not reliable!



https://medium.com/@beenkim/beyond-interpretability-4bf03bbd9394

Neither can we fully trust current post-hoc XAI techniques


Why Don't XAI Techniques Agree? Characterizing the Disagreements Between Post-hoc Explanations of Defect Predictions

Saumendu Roy
University of Saskatchewan, Canada
plz937@usask.ca

Gabriel Laberge
Polytechnique Montréal, Canada
gabriel.laberge@polymtl.ca

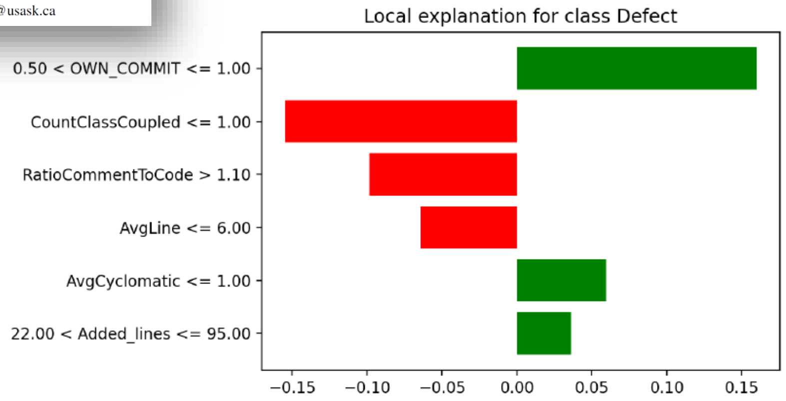
Banani Roy
University of Saskatchewan, Canada
banani.roy@usask.ca

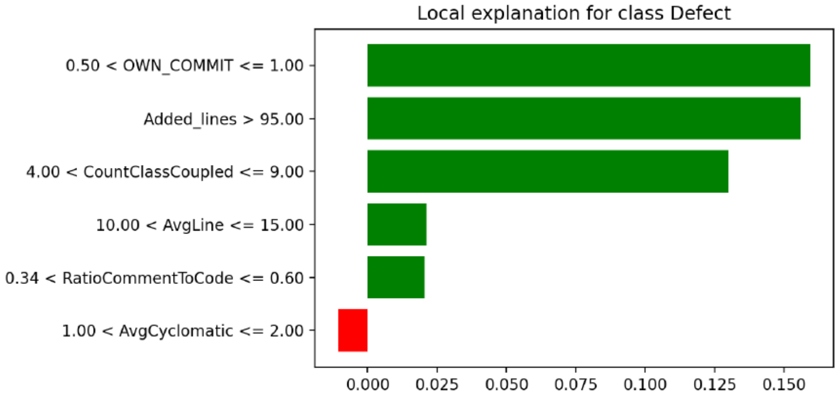
Foutse Khomh
Polytechnique Montréal, Canada
foutse.khomh@polymtl.ca

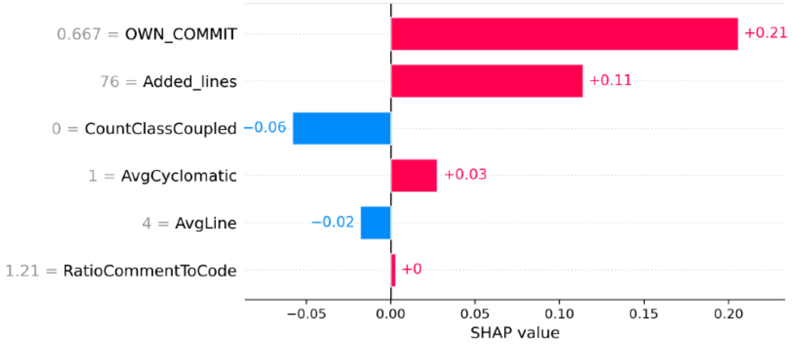
Amin Nikanjam
Polytechnique Montréal, Canada
amin.nikanjam@polyml.ca

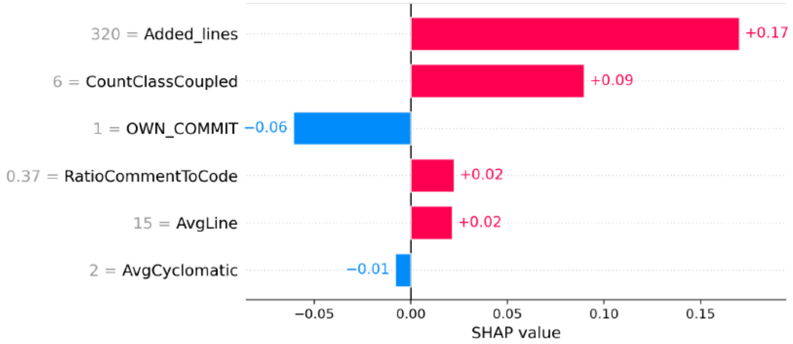
Saikat Mondal
University of Saskatchewan, Canada
saikat.mondal@usask.ca

ICSME'22



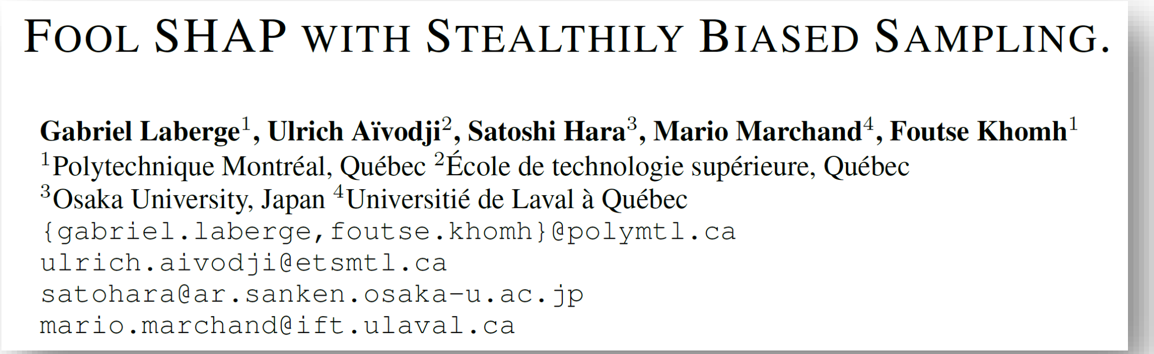






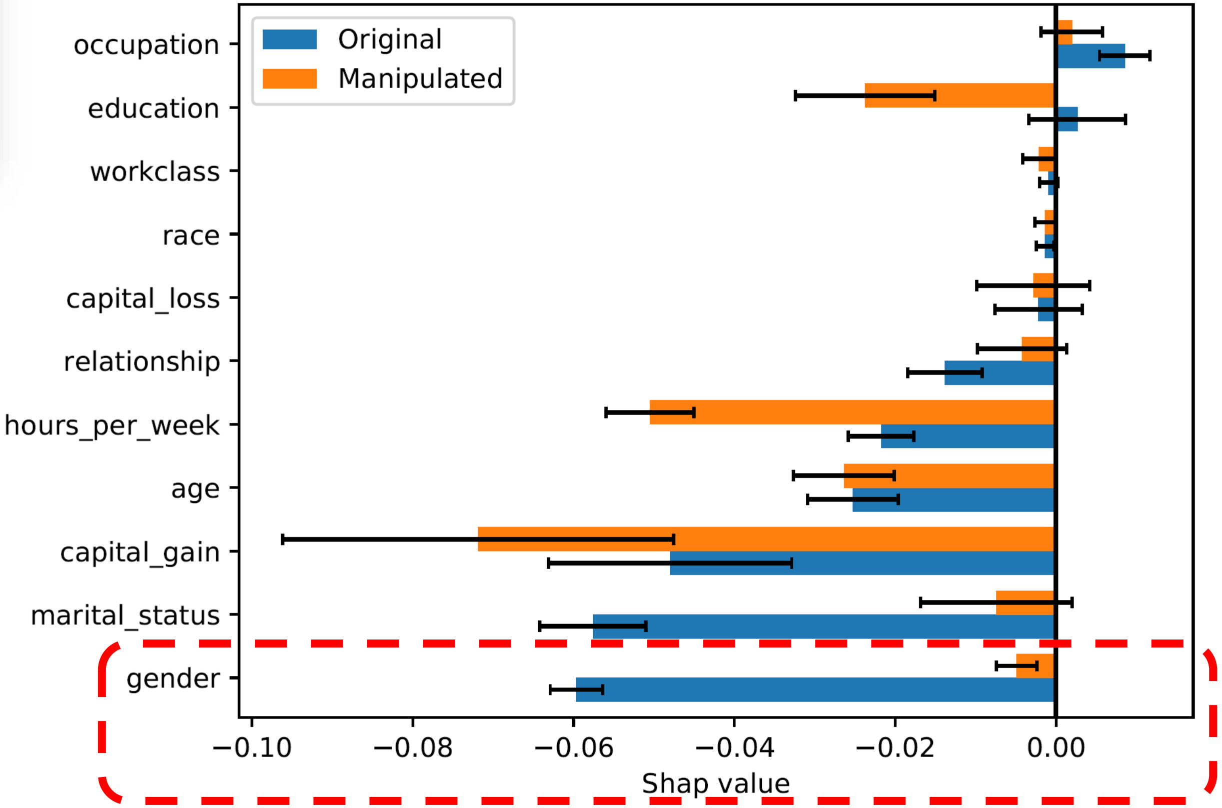
They often disagree!

Even worse, current post-hoc XAI techniques …


FOOL SHAP WITH STEALTHILY BIASED SAMPLING.

Gabriel Laberge[1], Ulrich Aïvodji[2], Satoshi Hara[3], Mario Marchand[4], Foutse Khomh[1]
[1]Polytechnique Montréal, Québec [2]École de technologie supérieure, Québec
[3]Osaka University, Japan [4]Universitié de Laval à Québec
{gabriel.laberge,foutse.khomh}@polymtl.ca
ulrich.aivodji@etsmtl.ca
satohara@ar.sanken.osaka-u.ac.jp
mario.marchand@ift.ulaval.ca

ICLR’23



…can be manipulated easily!

ML models are vulnerable to carefully crafted perturbations (adversarial robustness).

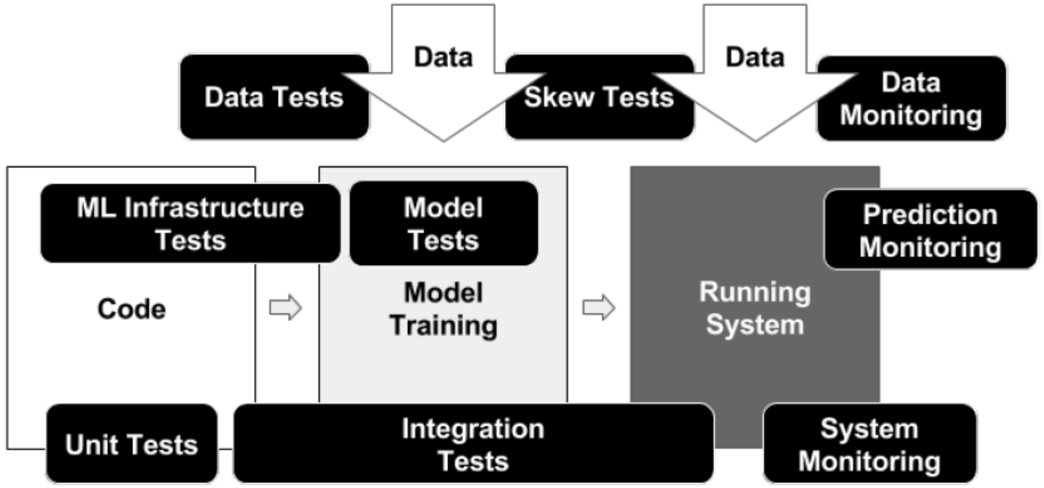
Stop | Yield | Speed Limit

(a) Normal | (b) Attack

https://portswigger.net/daily-swig/trojannet-a-simple-yet-effective-attack-on-machine-learning-models

Moreover, they hardly generalize out-of-distribution.

How can we provide safety guarantees that are required to reach Level 4/5?

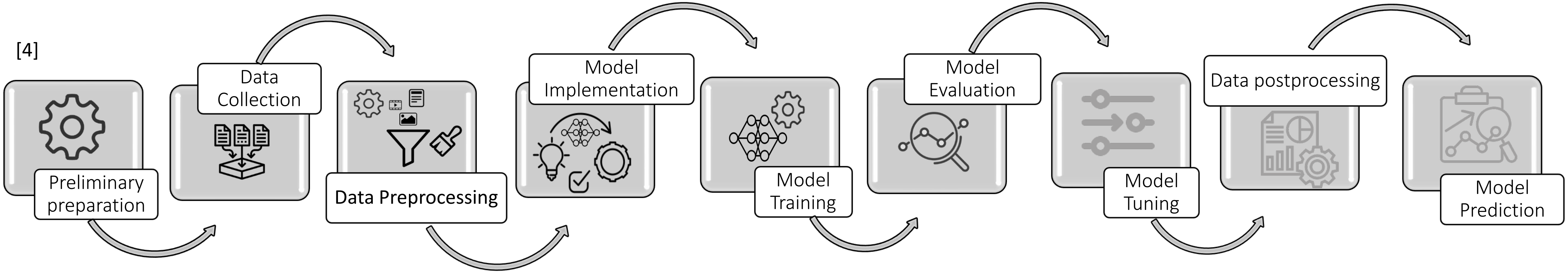


Extensive testing!

https://pubs.rsna.org/doi/10.1148/rg.305105013

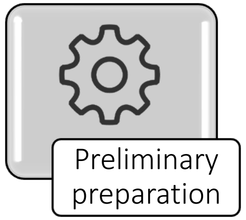
ML Development Phases

[4]



[4]Han et al., What do Programmers Discuss about Deep Learning Frameworks

ML Development Phases



Environment Preparation

Resolve Frameworks/libraries versions

CPU, GPU management

```
def initModel(self):
  model = LunaModel()
  if self.use_cuda:
    log.info("Using CUDA; {} devices.".format(torch.cuda.device_count()))
    if torch.cuda.device_count() > 1:
      model = nn.DataParallel(model)
    model = model.to(self.device)
  return model
```

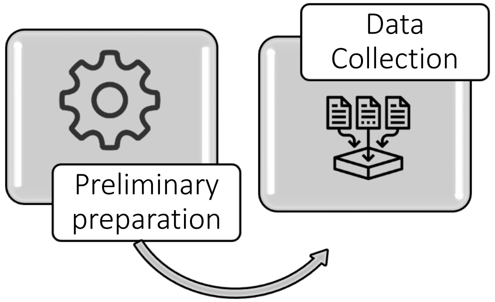
Detects multiple GPUs

Wraps the model

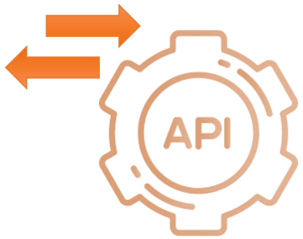
Sends model parameters to the GPU

[5] ELI STEVENS, LUCA ANTIGA, AND THOMAS VIEHMANN Deep Learning With Pytorch p286

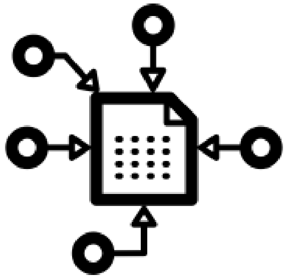
ML Development Phases



Load File from Disk



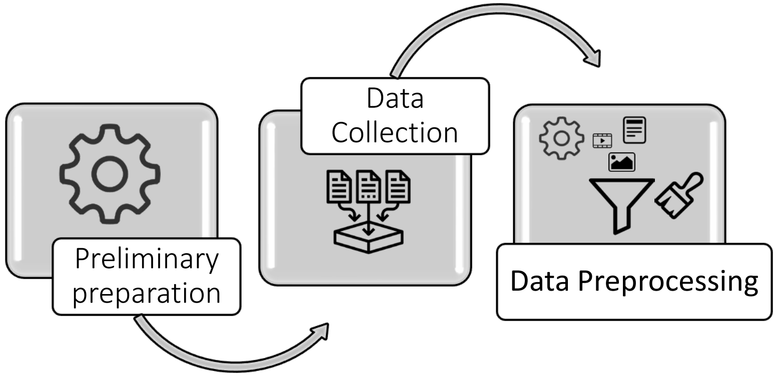
Call REST API

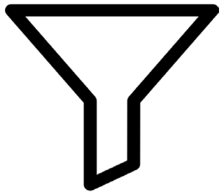


Using Data Collector Functionalities Provided by DL Frameworks

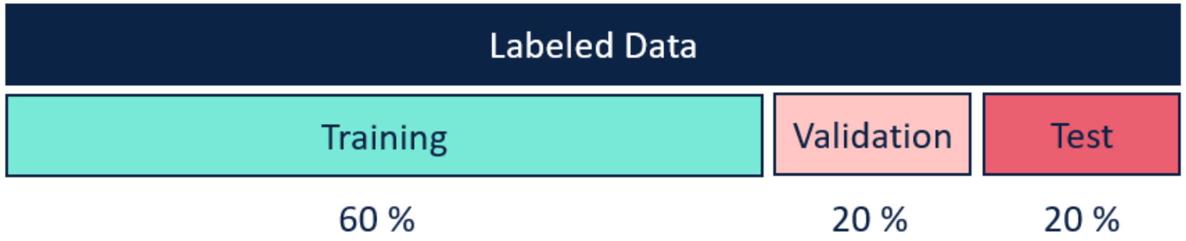
[4]Han et al., What do Programmers Discuss about Deep Learning Frameworks

ML Development Phases



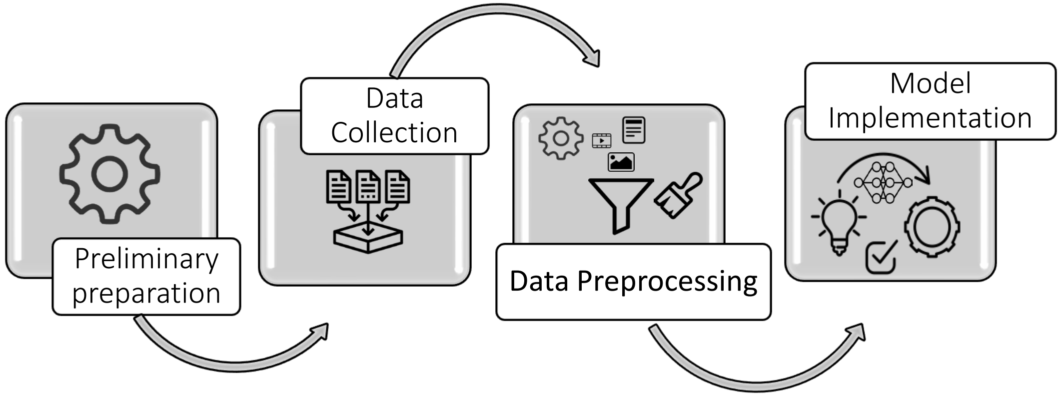


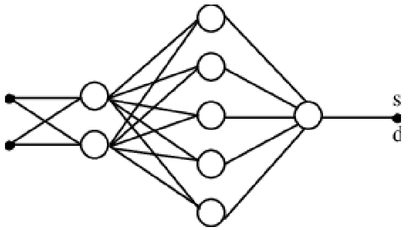
- Shape
- Size
- Format
- Data Type



[4]Han et al., What do Programmers Discuss about Deep Learning Frameworks

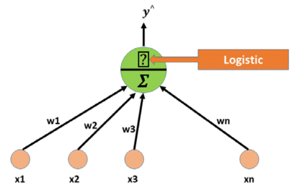
ML Development Phases

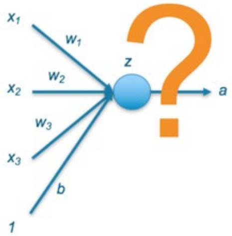




(Hyper)parameters
- Learning rate
- Batch size

[5]



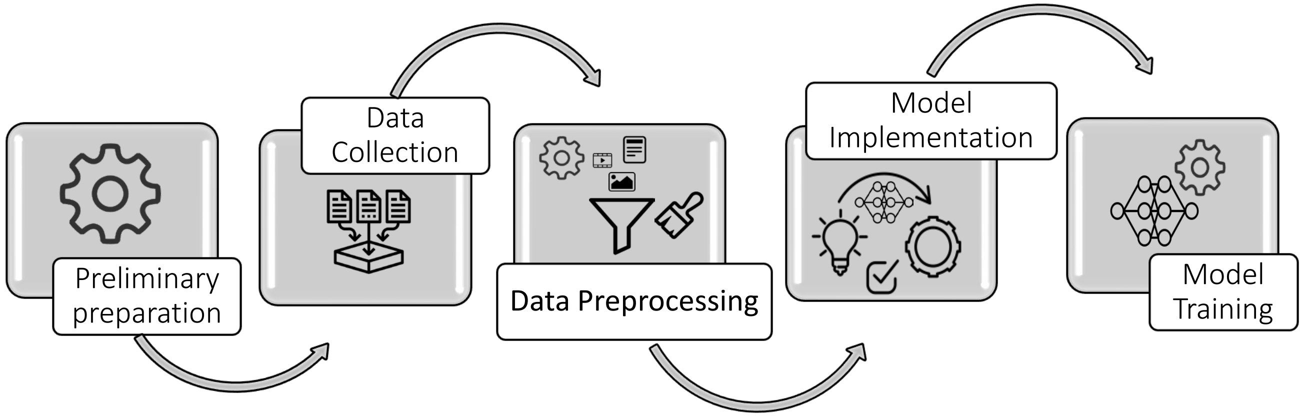


Optimizers
- Adam
- Momentum
- RMSProp

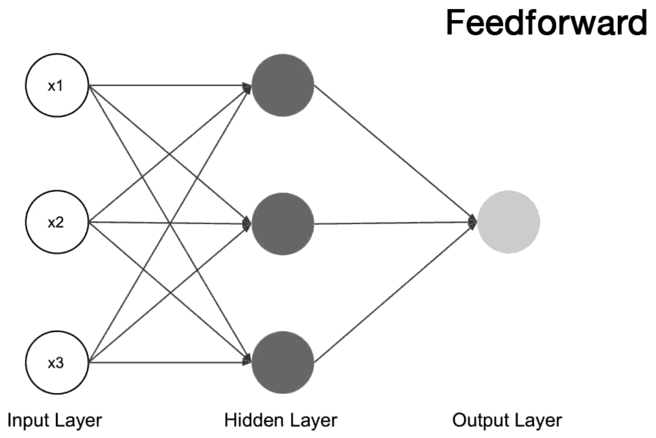
Choice of the architecture (Hyper)parameters Set Up Activation Function Loss Function Model Optimizers

[5] vikashraj luhaniwal., Analyzing different types of activation functions in neural networks — which one to prefer?

ML Development Phases

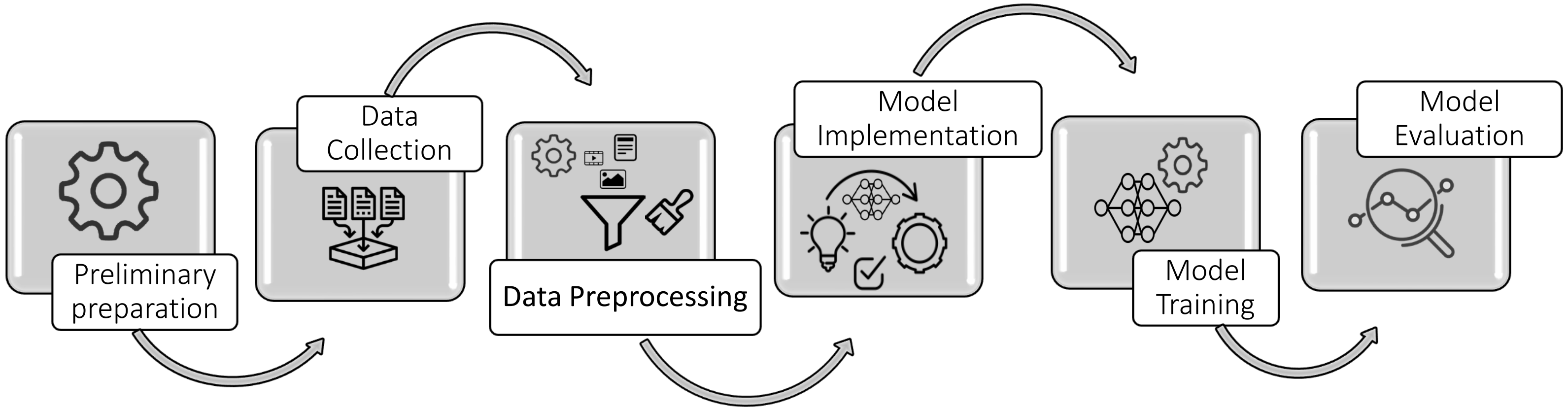


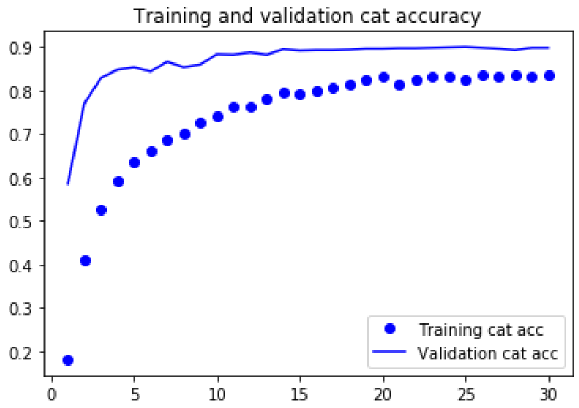
[6]

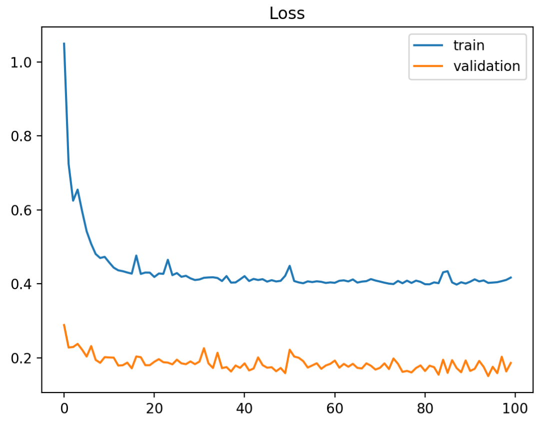


[6]Adarsh Menon., Neural Networks from Scratch in Python

ML Development Phases

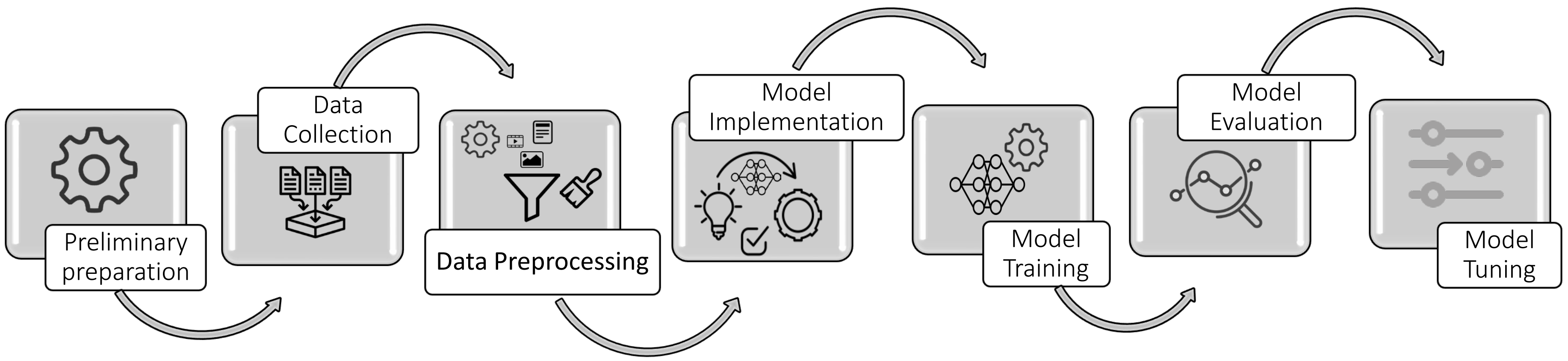


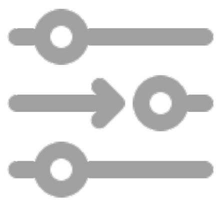




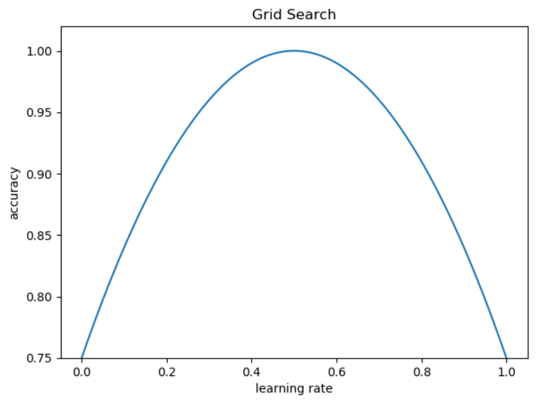
[4]Han et al., What do Programmers Discuss about Deep Learning Frameworks

ML Development Phases



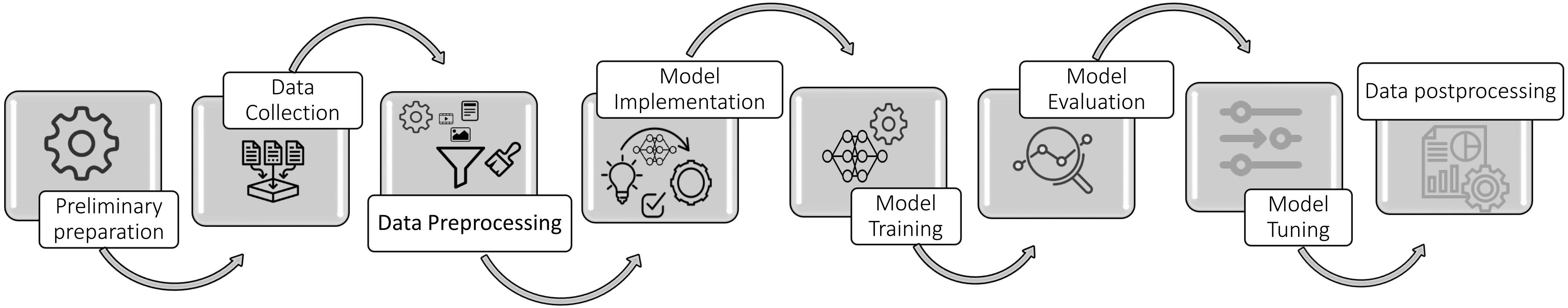


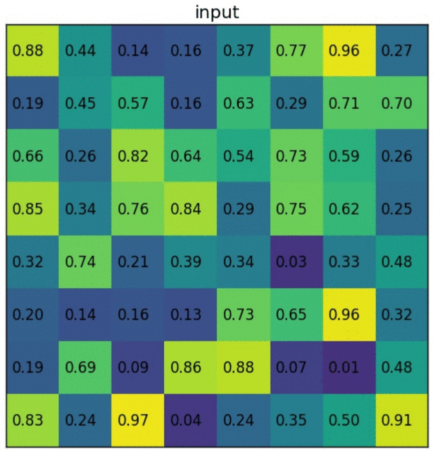
[7]

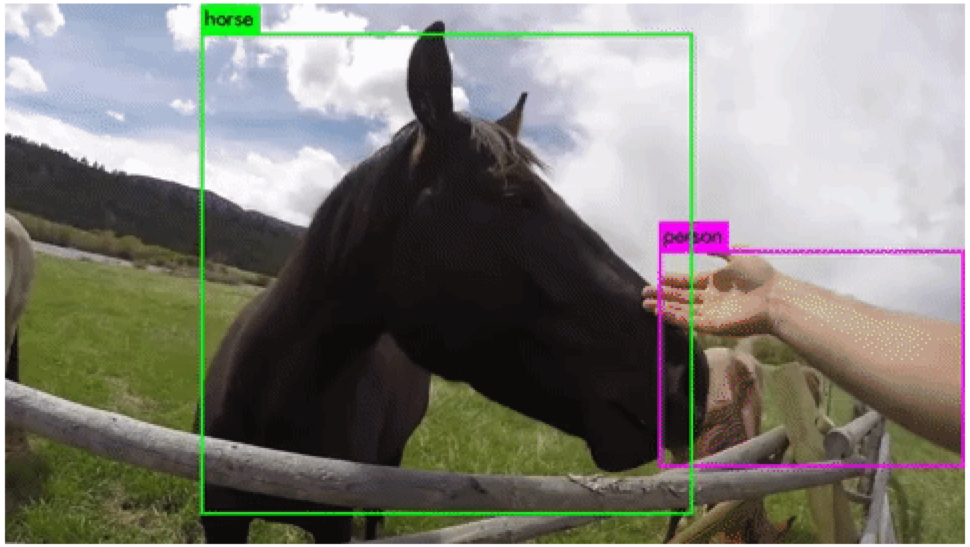


[7] David Morán, Hyperparameters Optimization

ML Development Phases

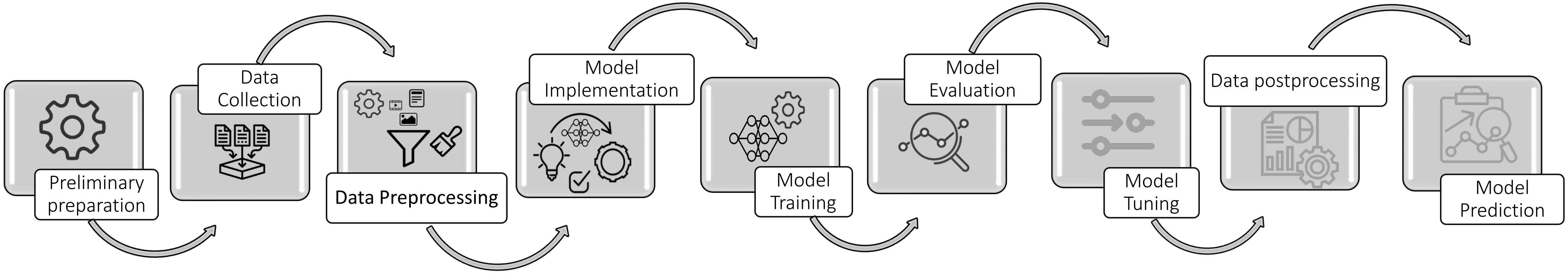


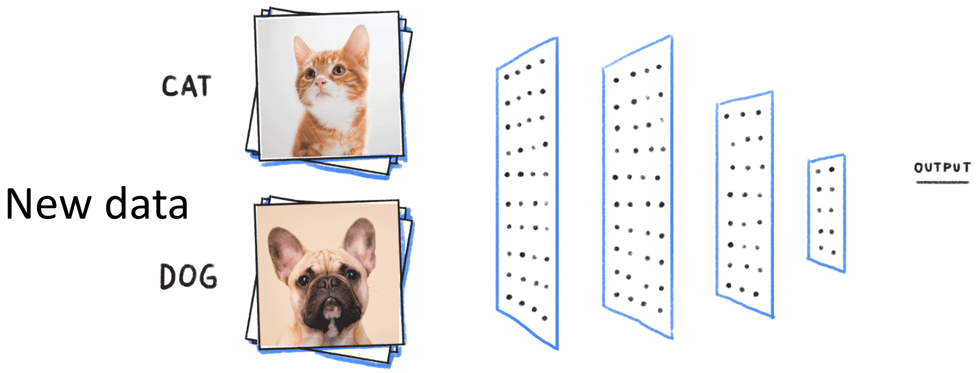




[4]Han et al., What do Programmers Discuss about Deep Learning Frameworks

ML Development Phases

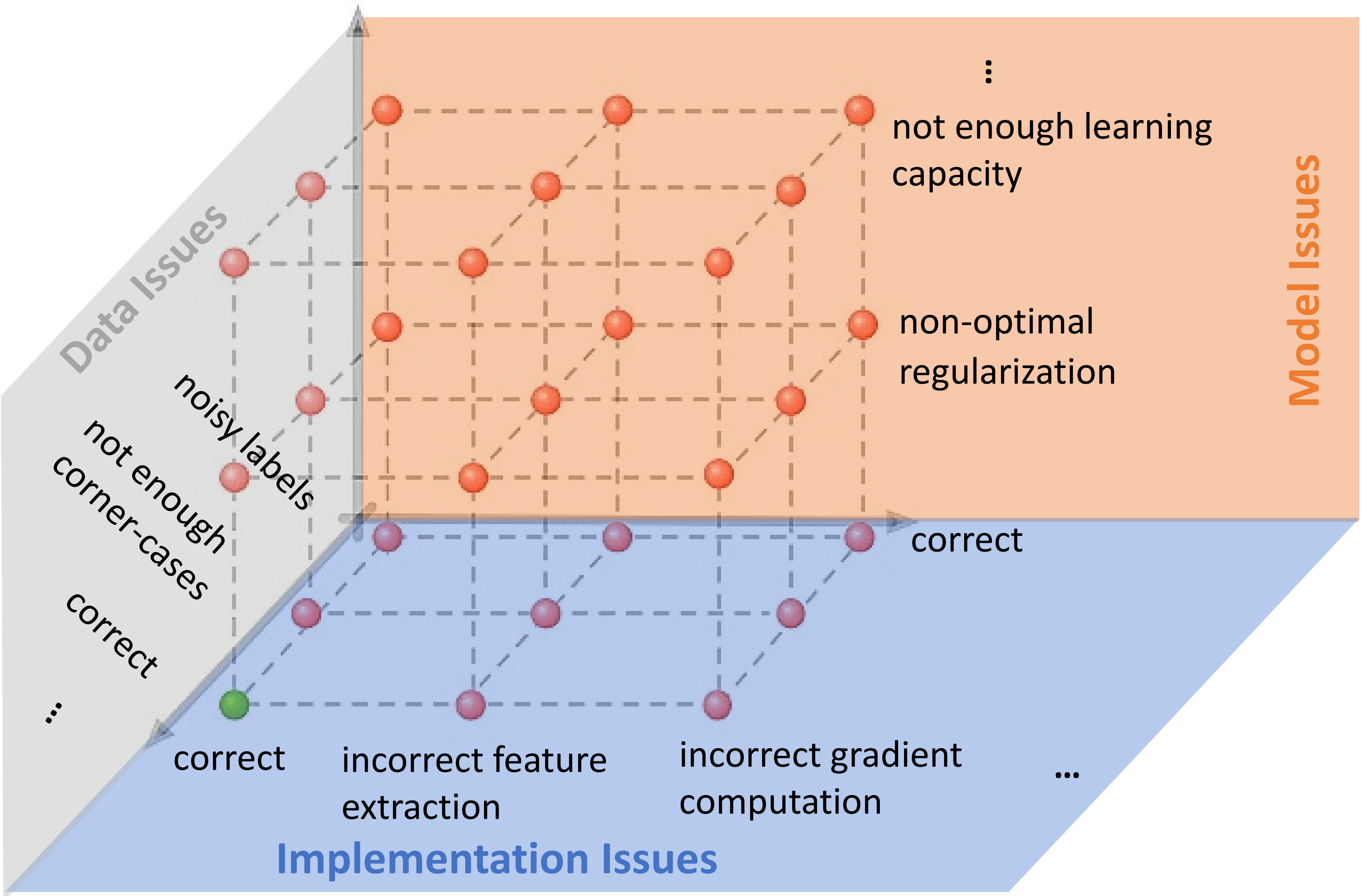




Prediction

[4]Han et al., What do Programmers Discuss about Deep Learning Frameworks

Multi-dimensional space of ML faults



Finding bugs in ML programs is hard

Common sentiment among practitioners

- **80-90% of time is spent debugging and tuning.**
- 10-20% is spent on figuring the mathematics and implementing the code for training.

Why is finding bugs in ML programs hard?

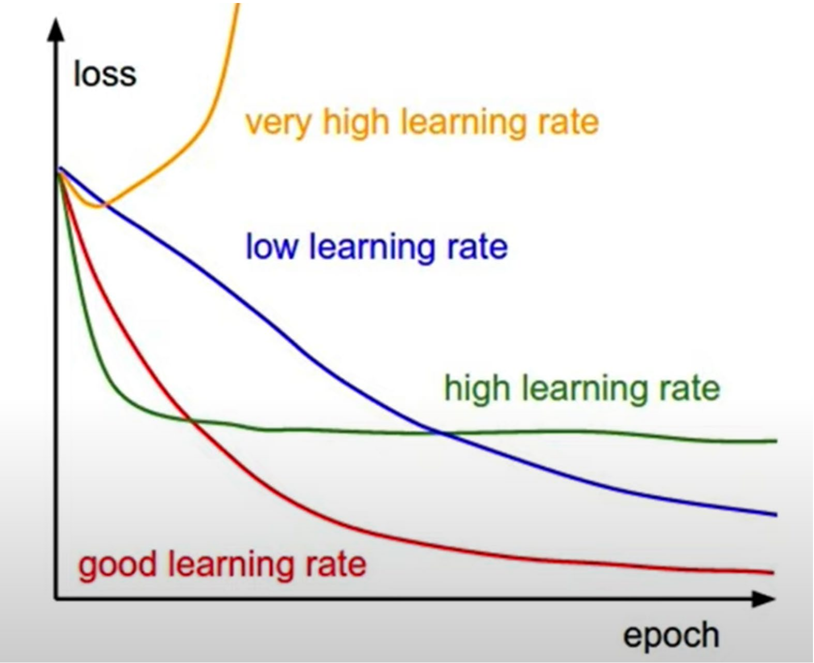
Most ML bugs are invisible

Labels out of order!

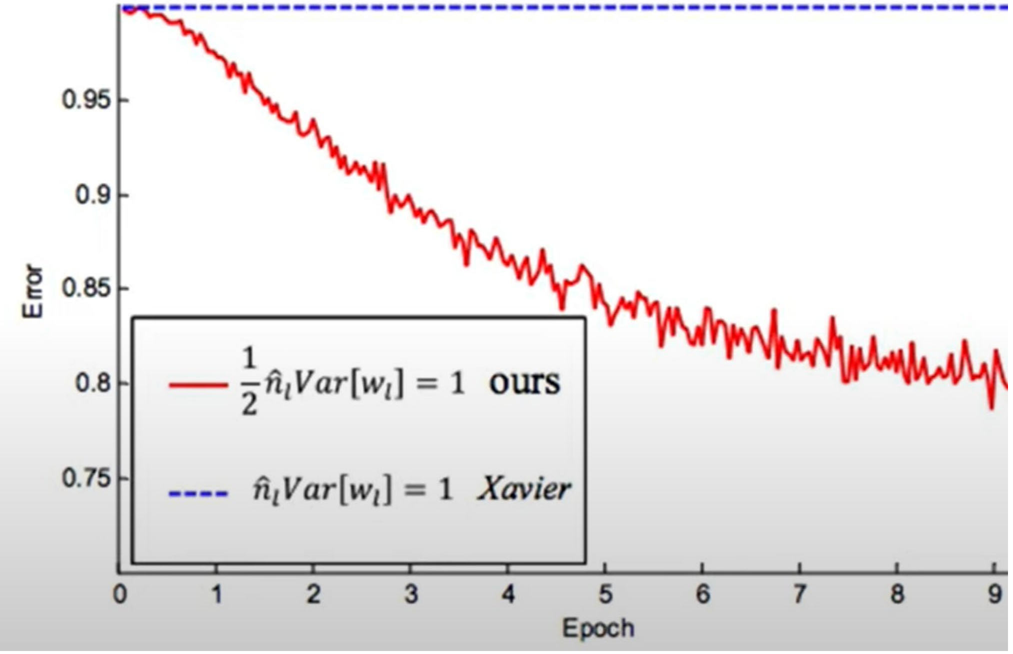
```
features = glob.glob('path/to/features/*')
labels = glob.glob('path/to/labels/*')
train(features, labels)
```

Full Stack Deep Learning, UC Berkeley, 2021

Why is finding bugs in ML programs hard?



Andrej Karpathy, CS231n course notes



He, Kaiming et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification." *2015 IEEE International Conference on Computer Vision (ICCV)* (2015): 1026-1034.

Models can be very sensitive to small differences in hyperparameters!

Example of Bugs and Design Issues in a CNN

① is a bug:

- Incompatibility between *softmax* as output activation and *binary_crossentropy* as loss function

② and ③ are design issues:

- Decreasing filters count: 224 > 55 > 13
- Decreasing filtering spatial size: (11, 11) > (5, 5) > (3, 3)
- Both represent poor structural choices
- Violating design patterns of effective and optimal CNN architectures
- **Leading to bad performance**
  - **Low accuracy**
  - **Long training time**

```
#train data
data1 = DataFetch('orange', ...)
data1 = DataFetch('apple', ...)
...
#one-hot encode outputs
y_train = np_utils.to_categorical(y_train)
#number of classes is 2: {orange, apple}
number_classes = y_train.shape[1]
#create the model
model = Sequential()
model.add(Conv2D(224, (11, 11), ...))
model.add(Dropout(0.2))
model.add(Conv2D(55, (5, 5), ...))
model.add(MaxPooling2D(pool_size(2, 2)))
model.add(Conv2D(13, (3, 3), ...))
model.add(Dropout(0.5))
...
model.add(Dense(num_classes), activation='softmax'))
# compile model
model.compile(loss='binary_crossentropy', optimizer=SGD, ...)
```

Deep Learning Model Verification Using Graph Transformations

AMIN NIKANJAM*, K. N. Toosi University of Technology, Iran and SWAT Lab., Polytechnique Montreal, Canada
HOUSSEM BEN BRAIEK*, SWAT Lab., Polytechnique Montreal, Canada
MOHAMMADMEHDI MOROVATI, SWAT Lab., Polytechnique Montreal, Canada
FOUTSE KHOMH, SWAT Lab., Polytechnique Montreal, Canada

TOSEM'21

NeuraLint : A linter for DL programs

- ✓ Capture defects early, so saves rework cost.
- ✓ Less expensive, because it doesn't require execution.
- ✓ Find defects in seconds.
- ✓ …

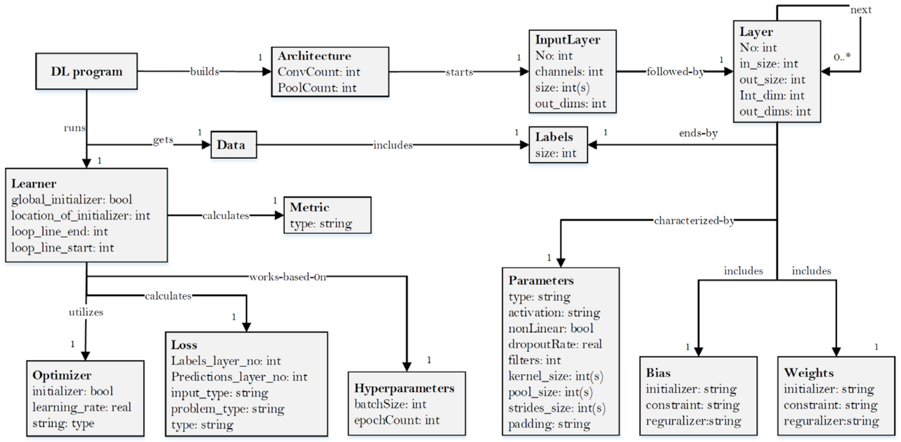
NeuraLint is fast and effective!

- ✓ It achieves an accuracy of 91.7 % .
- ✓ It correctly reported 18 additional bugs that were not found by developers.
- ✓ The average execution time of NeuraLint for the studied TensorFlow and Keras based programs are 2.892 and 3.197 seconds, respectively.

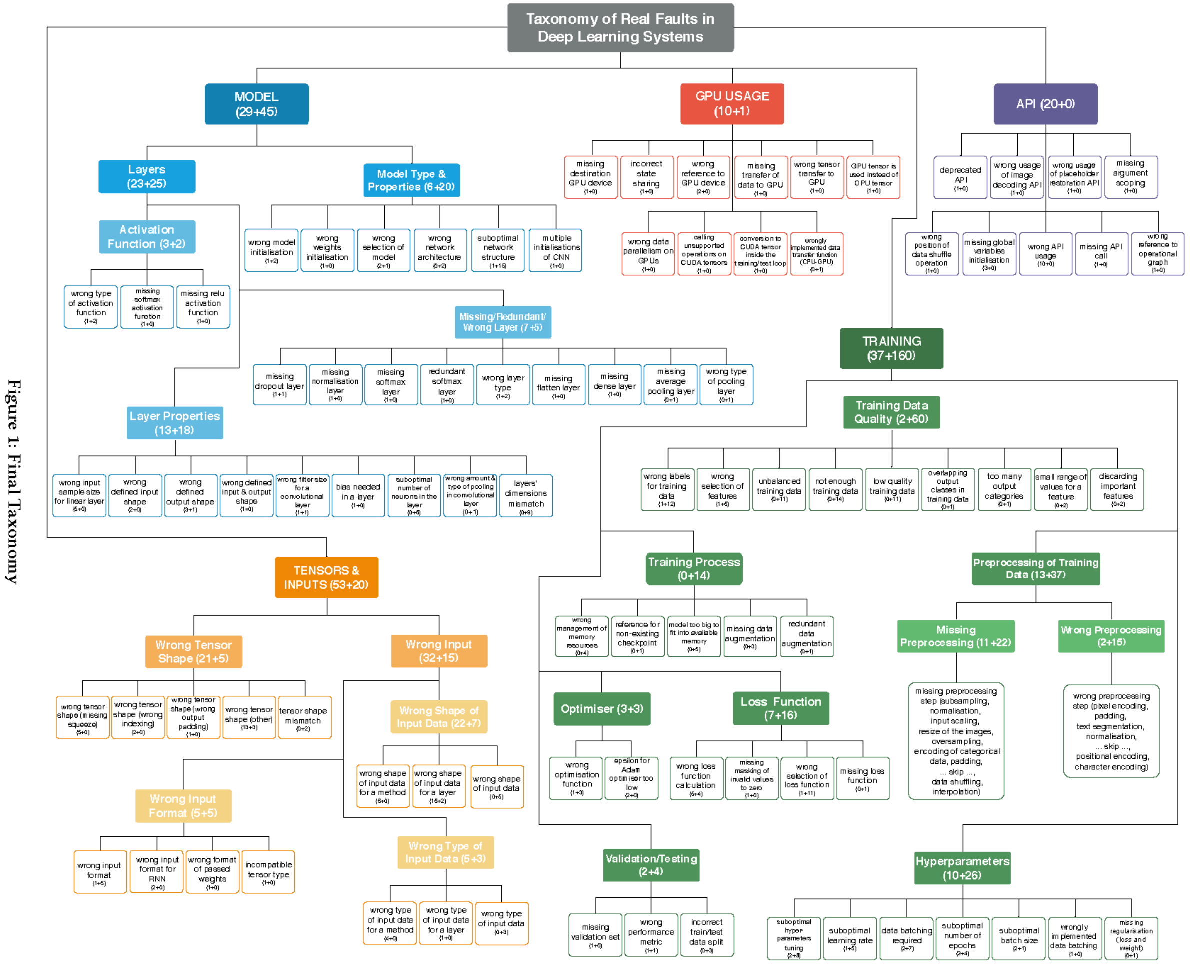
Try it out!

NeuraLint has two pillars

A meta-model of DL programs



Taxonomy of common DL faults



Gunel Jahangirova, Nargiz Humbatova, Gabriele Bavota, Vincenzo Riccio, Andrea Stocco, and Paolo Tonella. 2019. Taxonomy of Real Faults in Deep Learning Systems. arXiv preprint arXiv:1910.11015

Identification of Common DL Faults

Frameworks

PYTORCH

Keras

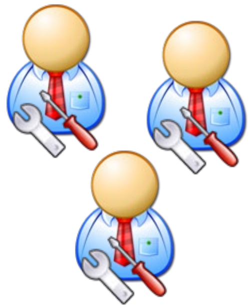
TensorFlow

Sources

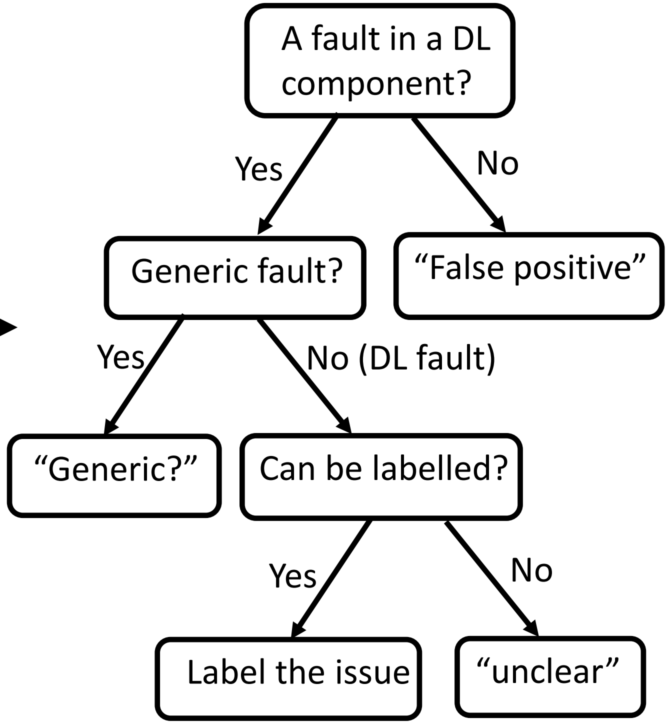
GitHub

stackoverflow

Artifact extraction



Open Coding



23 rules capturing common errors in DL programs (an excerpt)

- **Reshaped Data Retention**

  →A reshape layer should preserve the number of data elements. We verify that the product of original tensor dimensions equals to the product of reshaped tensor dimensions.

- **Unnecessary Activation Removal**

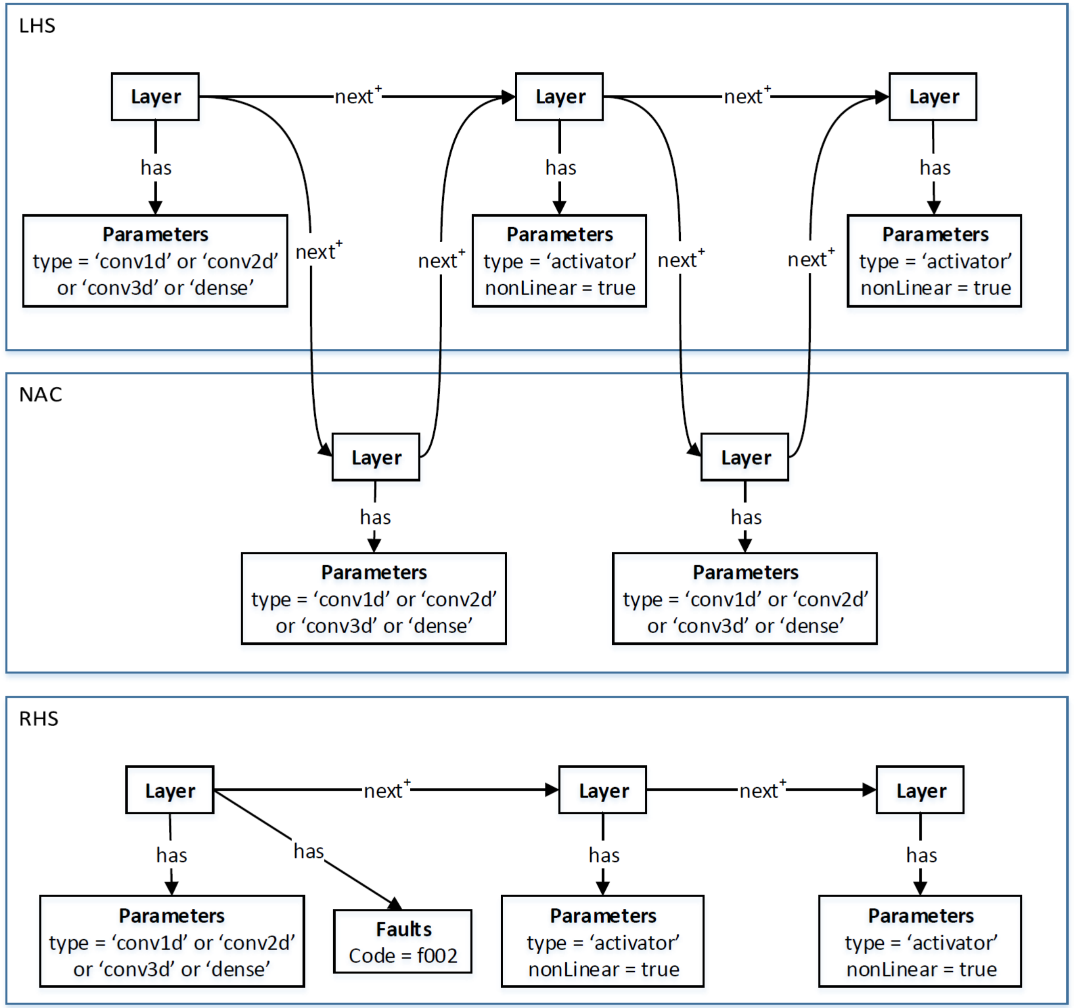
  →Multiple and redundant connected activations are not allowed. Since all activation functions are designed to transform real values into a restricted interval, **successive activations produce erroneous outputs.**

- **Zero Gradients Reset**

  →The gradients should be re-initialized after each training iteration. This clears old gradients from the last step; otherwise accumulating the gradients hinders the optimization process. **Some DL libraries (e.g., Pytorch) delegates this necessary reset step to their users.**

Graph transformations for ‘Unnecessary Activation Removal’

HG, (LHS, RHS, NAC)
HG: Host graph
LHS: Precondition of the rule
RHS: postcondition of the rule
NAC: Negative Application Condition, i.e., the rule can be applied only when NAC does not exist in the host graph

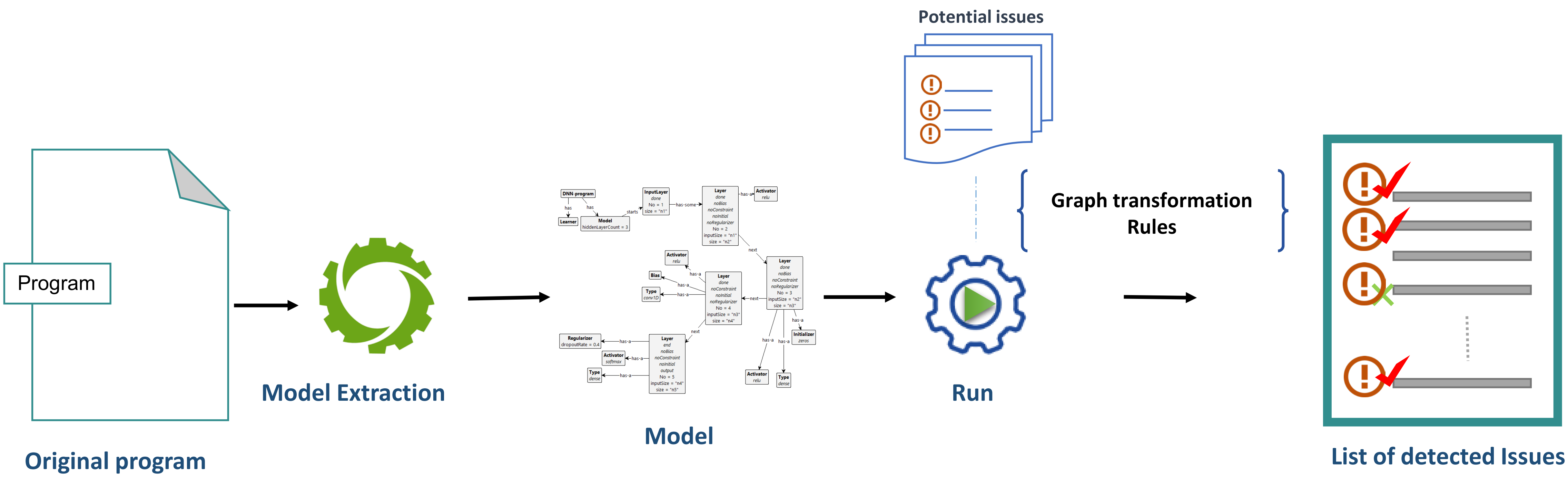


Application of the rule

(1) find a matching of LHS in HG,
(2) check NAC that forbid the presence of certain nodes and edges,
(3) remove a part of HG that can be mapped to LHS but not to RHS,
(4) a specific fault code is added to the node or edge in which the violation occurred.

Graph transformations are very efficient for finding violations of some conditions in a graph

NeuraLint: Execution Flow



NeuraLint: Model-based verification of DL programs

Algorithm 1: *NeuraLint*: Model-based verification of DL programs using graph transformations

Input: A DL program, *program*, and *rules* as a graph grammar
Output: List of bugs or warnings to improve the program
graph ← extractGraphFromProgram(*program*)
final ← graphChecker(*graph*, *rules*) :

(1) starting by *graph*, apply enables rules.
(2) apply enabled rules recursively.
(3) terminate when further application of rules becomes impossible.
(4) **return** *final*.

report ← extractReportFromGraph(*final*)
return *report*

Evaluation of NeuraLint

Keras

TensorFlow

18 Real-world DL programs with reported bugs

No.	SO #	Symptom	Recommended Fix	*NeuraLint*: violated rules
1	33969059	Bad Performance	Change the number of units for the output layer	Rules 9, 13
2	34311586	Bad Performance	Remove the last layer activation	Rules 9, 13, 19
3	38584268	Program Crash	Adding a flatten layer	Rules 1, 19, 21
4	44184091	Program Crash	Fix the limit size for input sequence data	Rules 19
5	44322611	Bad Performance	Prune the DNN, use RMSprop instead SGD	Rules 13, 20, 21
6	45120429	Program crash	Change the number of units for the output layer, Adding a flatten layer	Rules 1, 13, 19
7	45378493	Incorrect Functionality	Use a sigmoid for last layer activation	Rules 9, 11, 13, 19, 20
8	45711636	Program Crash	Use channels_last format for input data	Rule 2
9	49117607	Program Crash	Reduce spatial size of both Conv. filtering and pooling widows	Rules 2 ,11

- ✓ In total, **22 out of 24 bugs are detected correctly by NeuraLint (91.7 %)**. Moreover, NeuraLint correctly reported **18 additional bugs** that were not found by developers.
- ✓ The average execution time of NeuraLint for the studied TensorFlow and Keras based programs are 2.892 and 3.197 seconds respectively, **it is therefore quite efficient!**

Testing Neural Networks Training Programs

HOUSSEM BEN BRAIEK, SWAT Lab., Polytechnique Montreal, Canada
FOUTSE KHOMH, SWAT Lab., Polytechnique Montréal, Canada

TOSEM’22

TheDeepChecker : Dynamic testing of DL programs

- ✓ Capture defects during the training process.
- ✓ Less expensive than testing the resulting model.
- ✓ Some overhead on the training process.

…

TheDeepChecker outperforms AWS SMD

- ✓ DL coding bugs and misconfigurations are detected with (precision, recall), respectively, equal to (90%, 96.4%) and (77%, 83.3%).
- ✓ Finds 30% more defects than AWS SageMaker.

Try it out!

TheDeepChecker verification rules

Parameters-related Issues	Untrained Parameters
	Poor Weight Initialization
	Parameters' Values Divergence
	Parameters Unstable Learning
Activation-related Issues	Activations out of Range
	Neuron Saturation
	Dead ReLU
Optimization-related Issues	Unable to fit a small sample
	Zero Loss
	Diverging Loss
	Slow or Non decreasing Loss
	Loss Fluctuations
	Unstable Gradient: Exploding
	Unstable Gradient: Vanishing

TheDeepChecker verification rules

Parameters-related Issues

Untrained Parameters

Poor Weight Initialization

Parameters' Values Divergence

Parameters Unstable Learning

Issue

Given a layer i and N iterations

$$W_i^0 = W_i^1 , b_i^0 = b_i^1$$
$$W_i^1 = W_i^2 , b_i^1 = b_i^2$$
$$\dots$$
$$W_i^{N-1} = W_i^N , b_i^{N-1} = b_i^N$$

Verification Routine

Given a layer i and an iteration j

$$W_i^j \neq W_i^{j+1} b_i^j \neq b_i^{j+1}$$

$$\forall j \in [0, N-1]$$

TheDeepChecker verification rules

Activation-related Issues	Activations out of Range	Issue: Given a layer i, $A_i \notin [min, max]$
	Neuron Saturation	Verification Routine: Given a layer i, $min \leq A_i \leq max$
	Dead ReLU	

TheDeepChecker verification rules

Optimization-related Issues

Unable to fit a small sample

Zero Loss

Diverging Loss

Slow or Non decreasing Loss

Loss Fluctuations

Unstable Gradient: Exploding

Unstable Gradient: Vanishing

Issue

The DNN could not properly minimize the loss.

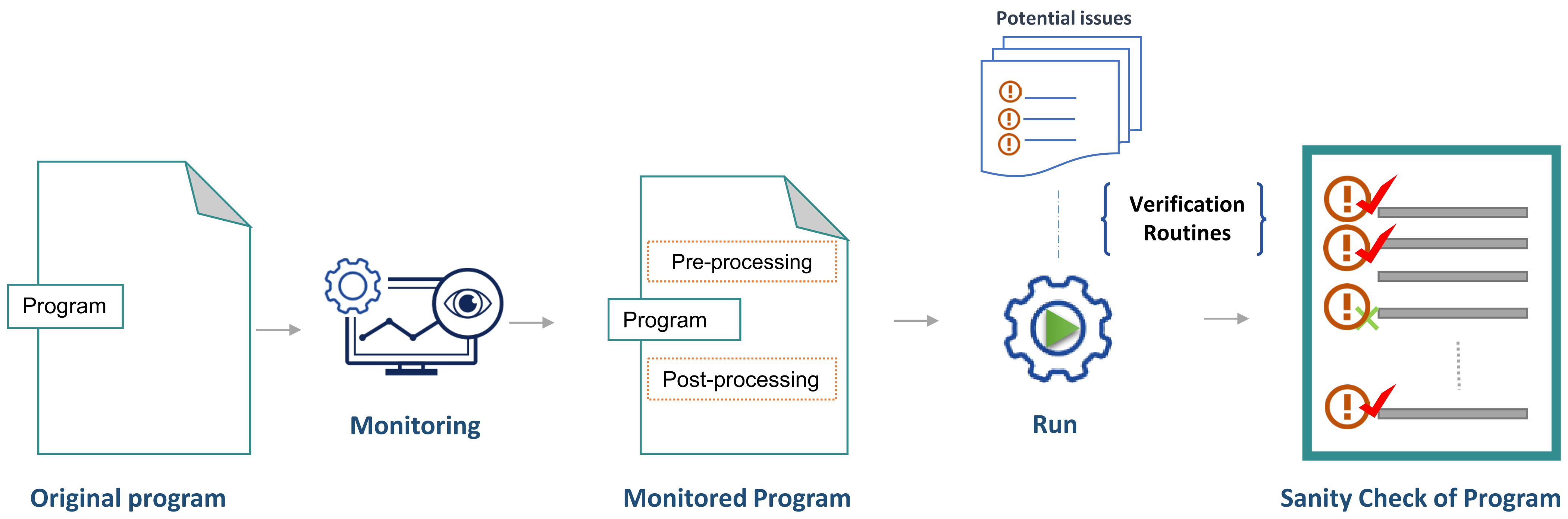
Verification Routine

The DNN (with regularization off) should overfit a tiny sample of data.

Given N iterations

$$loss_N = 0$$

TheDeepChecker: Execution Flow



TheDeepChecker vs Amazon SageMaker (SMD)

Faults	Base NN	Perf.	SMD Rule(s)	Fired Check(s)	TP	FP	FN
missing input normalization	Regr	24.20	-	**Uns-Inps**[1], PI-Loss[2] Un-Fit-Batch[3], Uns-Act-HS[4]	1+3	0	0
	Shallow	11.35%	R_1,R_8,R_{14}	**Uns-Inps**, PI-Loss, Un-Fit-Batch Div-Loss[5], Div-W[6], Div-B[7], Div-Grad[8]	1+6	0	0
	Deep	85%	R_1,R_8,R_{10}	**Uns-Inps**, PI-Loss, Uns-Act-HS, ~~NR-Loss~~[9]	1+2	1	0
over-scaled outputs	Regr	20.14	R_2, R_{12}	**Uns-Outs**[10], SD-Loss[11], Dead-ReLU[12], Uns-Act-HS	1+3	0	0
redundant input normalization	Regr	2.86	-	**Uns-Inps**, SD-Loss, Uns-Act-LS[13], Un-Fit-Batch	1+3	0	0
	Shallow	33.75%	R_8, R_{14}	**Uns-Inps**, SD-Loss, W-Up-Slow[14], Uns-Act-LS	1+3	0	0
	Deep	77.5%	-,R_8,R_{10}	**Uns-Inps**, Uns-Act-LS	1+1	0	0
gradients with flipped sign	Regr	$1.72e7$	-	Un-Fit-Batch, **Div-Loss**, Uns-Act-HS	1+2	0	0
	Shallow	9.8%	R_{11},R_{14}	Un-Fit-Batch, **Div-Loss**, Div-W, Div-B, Uns-Act-HS, Van-Grad[15]	1+5	0	0
	Deep	10%	R_{11},R_{14}	Un-Fit-Batch, **Div-Loss**, Uns-Act-HS, NR-Loss[16]	1+2	0	0
missing softmax activation	Shallow	9.8%	R_{14}	PI-Loss, **Inv-Outs**[17], SD-Loss W-Up-Slow, Van-Grad, Un-Fit-Batch, ~~Over-Reg-Loss~~[18]	1+5	1	0
	Deep	11.48%	R_{14},R_8,R_{10}	PI-Loss, **Inv-Outs**, Van-Grad	1+2	0	0
softmax out-and in-the loss	Shallow	99.29%	-	SD-Loss, W-Up-Slow(Dense)	0+2	0	1
	Deep	83.24%	-,R_8,R_{10}	SD-Loss, HF-Loss[19], W-Up-Slow(Dense), ~~NR-Loss~~[20]	0+2	1	1
softmax over wrong axis	Shallow	99.45%	R_{14}	PI-Loss, **Inv-Outs**, **Inv-Out-Dep**[21], Inv-Loss-Dep[22]	2+2	0	0
	Deep	85.86%	R_{14},R_8,R_{10}	PI-Loss, **Inv-Outs**, **Inv-Out-Dep**, Inv-Loss-Dep	2+2	0	0
CE over wrong axis	Shallow	8.92%	R_2,R_7	**PI-Loss**, **Inv-Loss-Dep**	2+0	0	0
	Deep	86.79%	-,R_8,R_{10}	**PI-Loss**, **Inv-Loss-Dep**	2+0	0	0
MSE with wrong broadcasting	Regr	7.02	R_2	Un-Fit-Batch, SD-Loss, Van-Grad	0+3	0	1
inverted CE's mean and sum	Shallow	11.34%	R_{14}	**PI-Loss**	1+0	0	0
	Deep	87.08%	-,R_8,R_{10}	**PI-Loss**	1+0	0	0
shuffle only the features	Regr	7.27	-	**Corrupted Labels**	1+0	0	0
	Shallow	11.35%	-	**Corrupted Labels**	1+0	0	0
	Deep	10.09%	-,R_8,R_{10}	**Corrupted Labels**	1+0	0	0
invalid data transformation	Shallow	99.24%	-	**Shifted-Augmented-Data**	1+0	0	0
	Deep	86.28%	-,R_8,R_{10}	**Shifted-Augmented-Data**	1+0	0	0

- ✓ DL coding bugs and misconfigurations are detected with (precision, recall), **respectively, equal to (90%, 96.4%) and (77%, 83.3%).**
- ✓ TheDeepChecker outperforms SMD **by detecting 75% rather than 60% of** the total of reported bugs.

Testing Neural Networks Training Programs

HOUSSEM BEN BRAIEK, SWAT Lab., Polytechnique Montreal, Canada
FOUTSE KHOMH, SWAT Lab., Polytechnique Montréal, Canada

TOSEM'22

TheDeepChecker : Dynamic testing of DL programs

- ✓ Capture defects during the training process.
- ✓ Less expensive than testing the resulting model.
- ✓ Some overhead on the training process.

…

TheDeepChecker outperforms AWS SMD

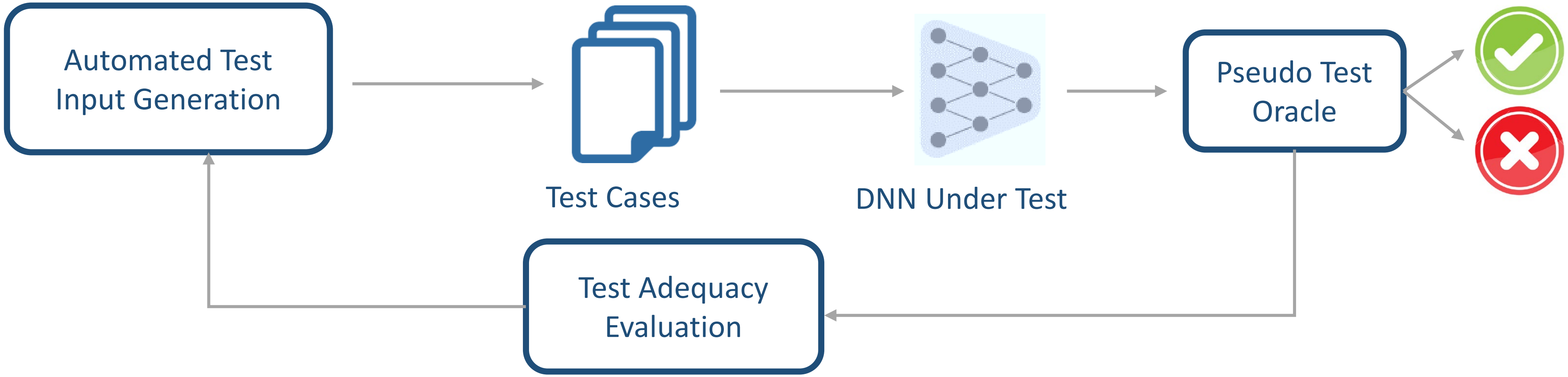
- ✓ DL coding bugs and misconfigurations are detected with (precision, recall), respectively, equal to (90%, 96.4%) and (77%, 83.3%).
- ✓ Finds 30% more defects than AWS SageMaker.

Try it out!

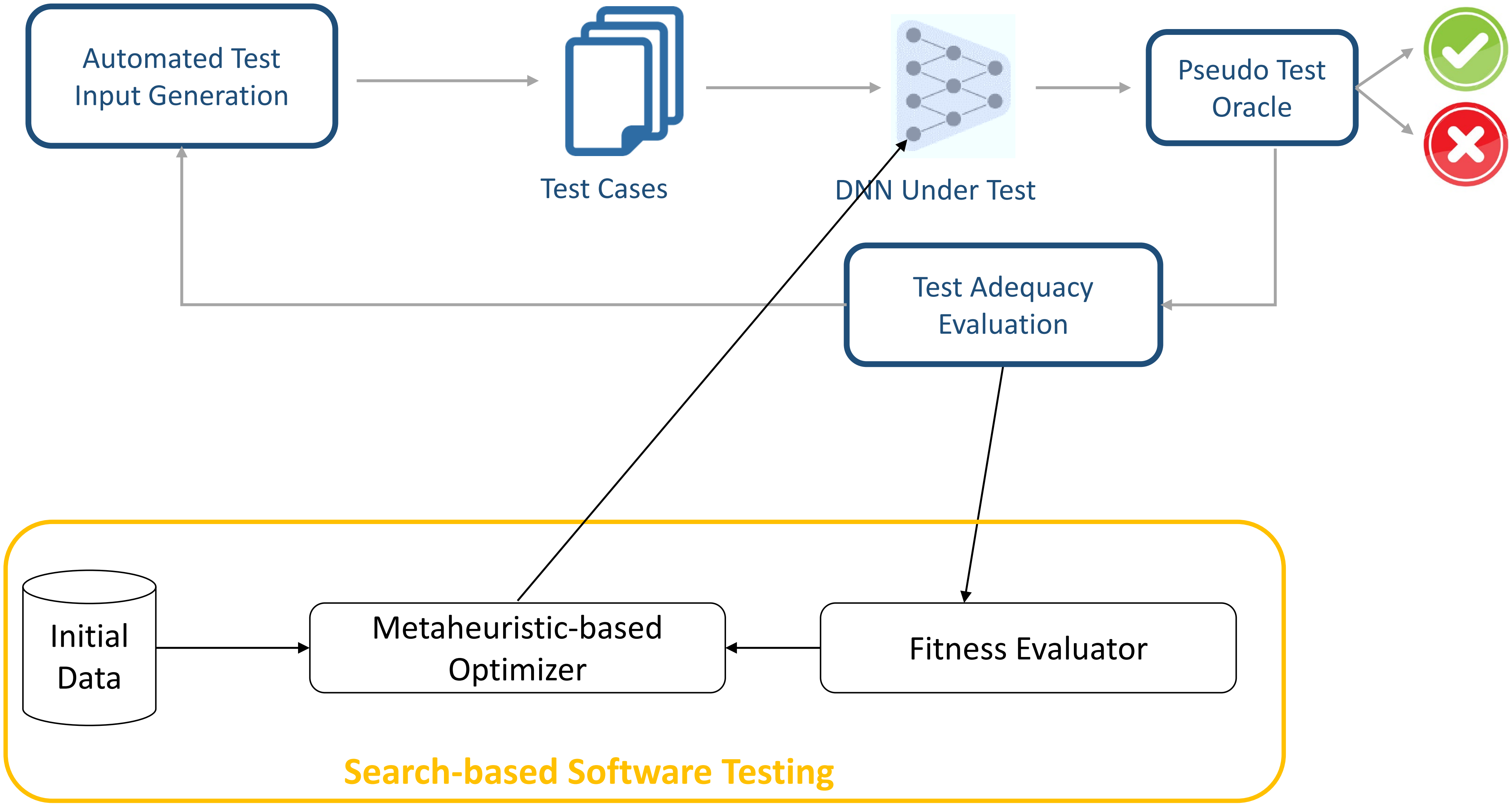
ICSME’19, TOSEM’23

DeepEvolution: A Search-Based Testing Approach for Deep Neural Networks

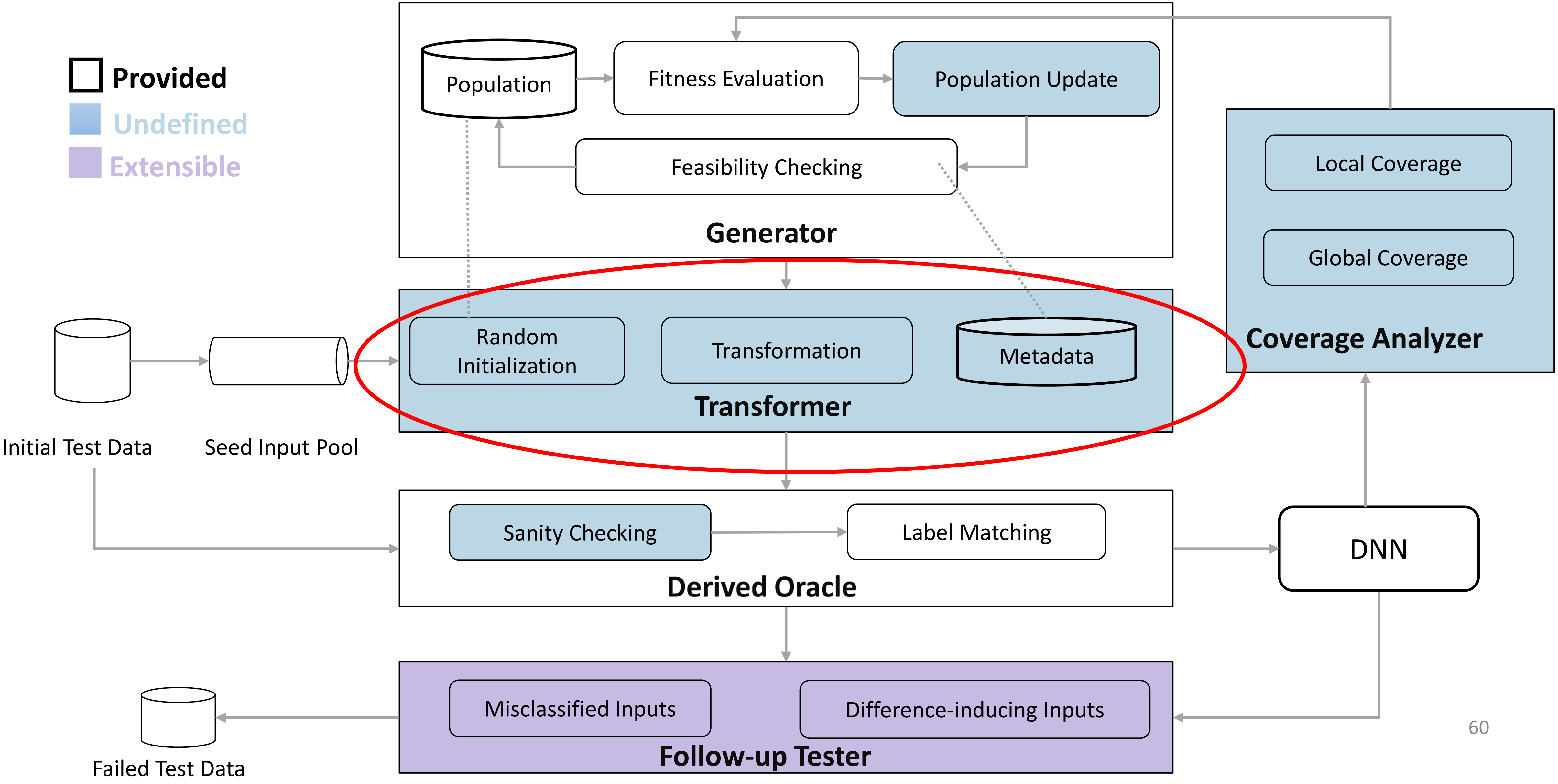
Houssem Ben Braiek and Foutse Khomh
SWAT Lab., Polytechnique Montréal, Montréal, Canada
{houssem.ben-braiek, foutse.khomh}@polymtl.ca



DeepEvolution: Search-based Test Input Generation



DeepEvolution: DL-based Software Testing Workflow



Semantically-Preserving Metamorphic Image Transformation

- Brightness
- Contrast
- Pixel Perturbation
- Blurring

Pixel-value Transformations

- Translation
- Scaling
- Shearing
- Rotation

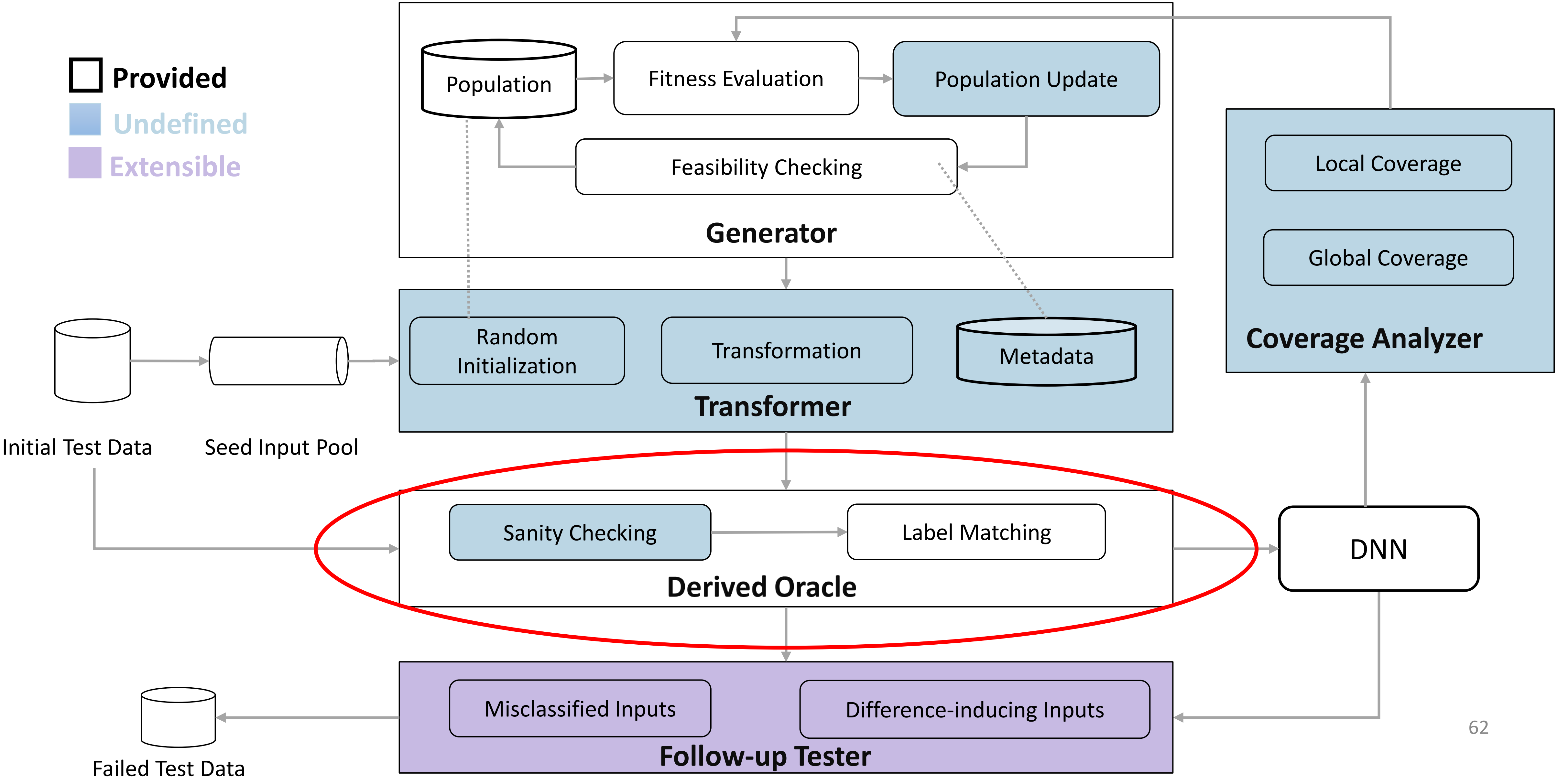
Affine Transformations

Tuning the interval domain of each transformation's parameters,
e.g.
[th_R_min, th_R_max] ➔ [tun_R_min, tun_R_max]

th_R_{min, max} : {min, max} theoretical rotation angle

tun_R_{min, max} : {min, max} tuned rotation angle

DeepEvolution: DL-based Software Testing Workflow



Semantically-Preserving Metamorphic Image Transformation

Brightness

Contrast

Pixel Perturbation

Blurring

Pixel-value Transformations

+

Sanity Check:

$$SSIM(mutated, original) > threshold$$

Translation

Scaling

Shearing

Rotation

Affine Transformations

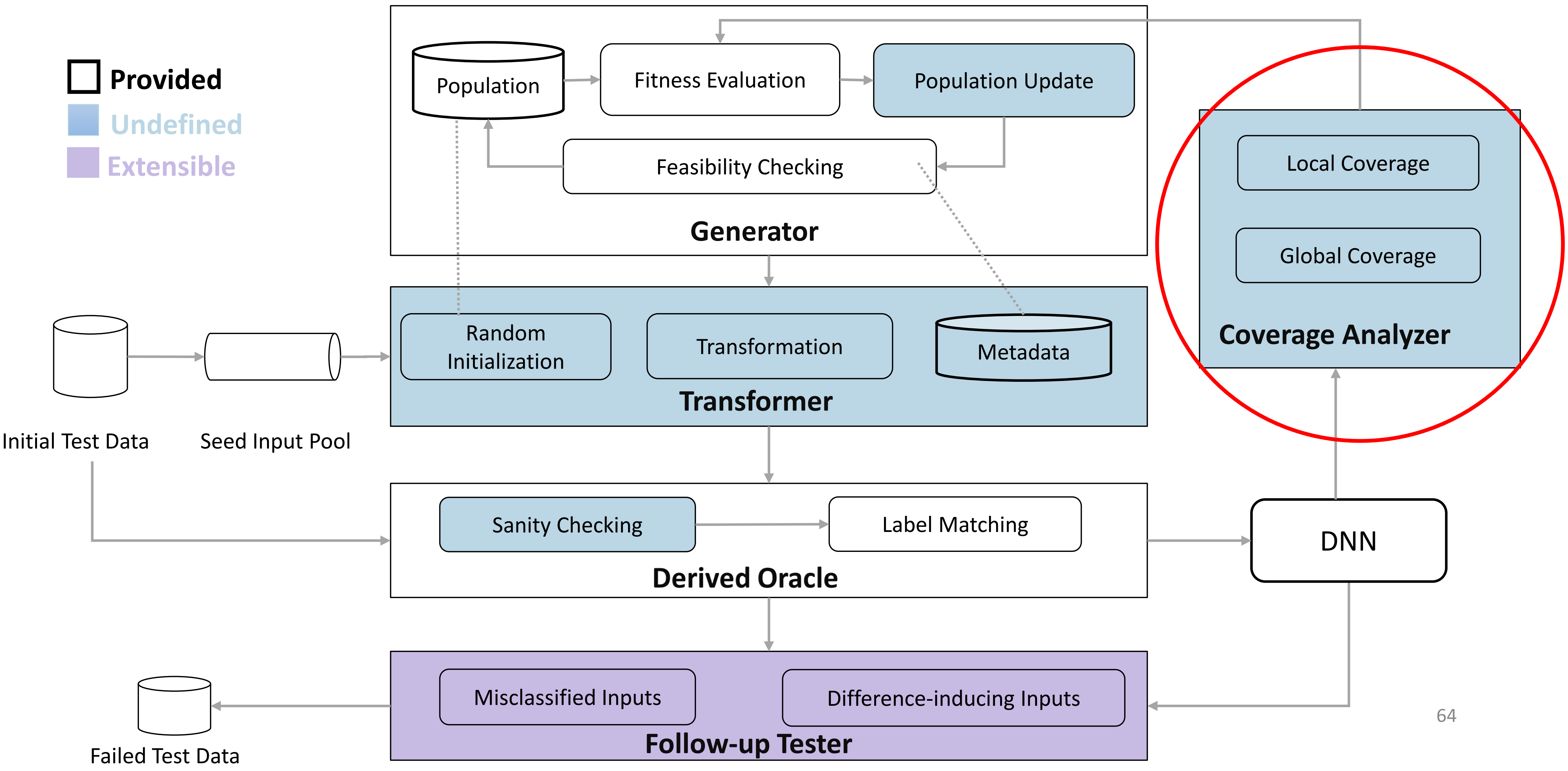
+

Conservative Strategy:

They should be exclusively applied

SSIM : Structural Similarity Index Metric

DeepEvolution: DL-based Software Testing Workflow

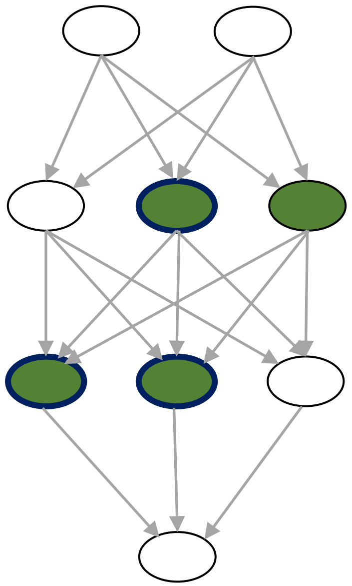


Neuron Coverage-based Fitness Function

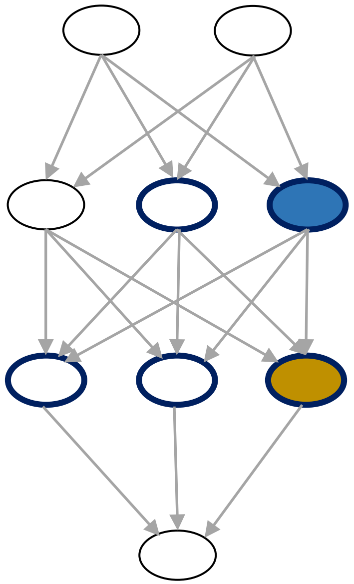
$$Fitness_1 = \alpha_1 \times NLNC + \beta_1 \times NGNC$$

$\boldsymbol{NLNC}$: *Novel Local Neuron Coverage*
$\boldsymbol{NGNC}$: *Novel Global Neuron Coverage*
$\boldsymbol{\alpha_1}, \boldsymbol{\beta_1}$: *Weights assigned to NLNC, NGNC*

Original Input (Ancestor)



Synthetic Input (Descendant)



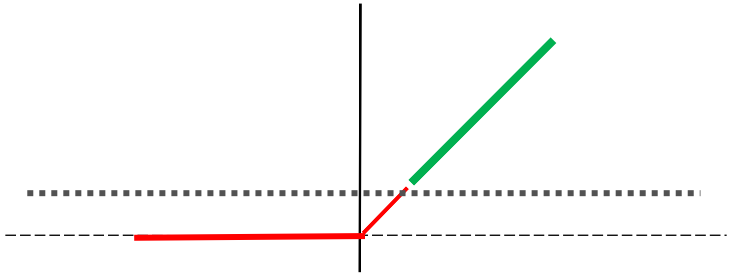
activated

covered

new globally covered

new locally covered

Rectified Liner Unit(ReLU)

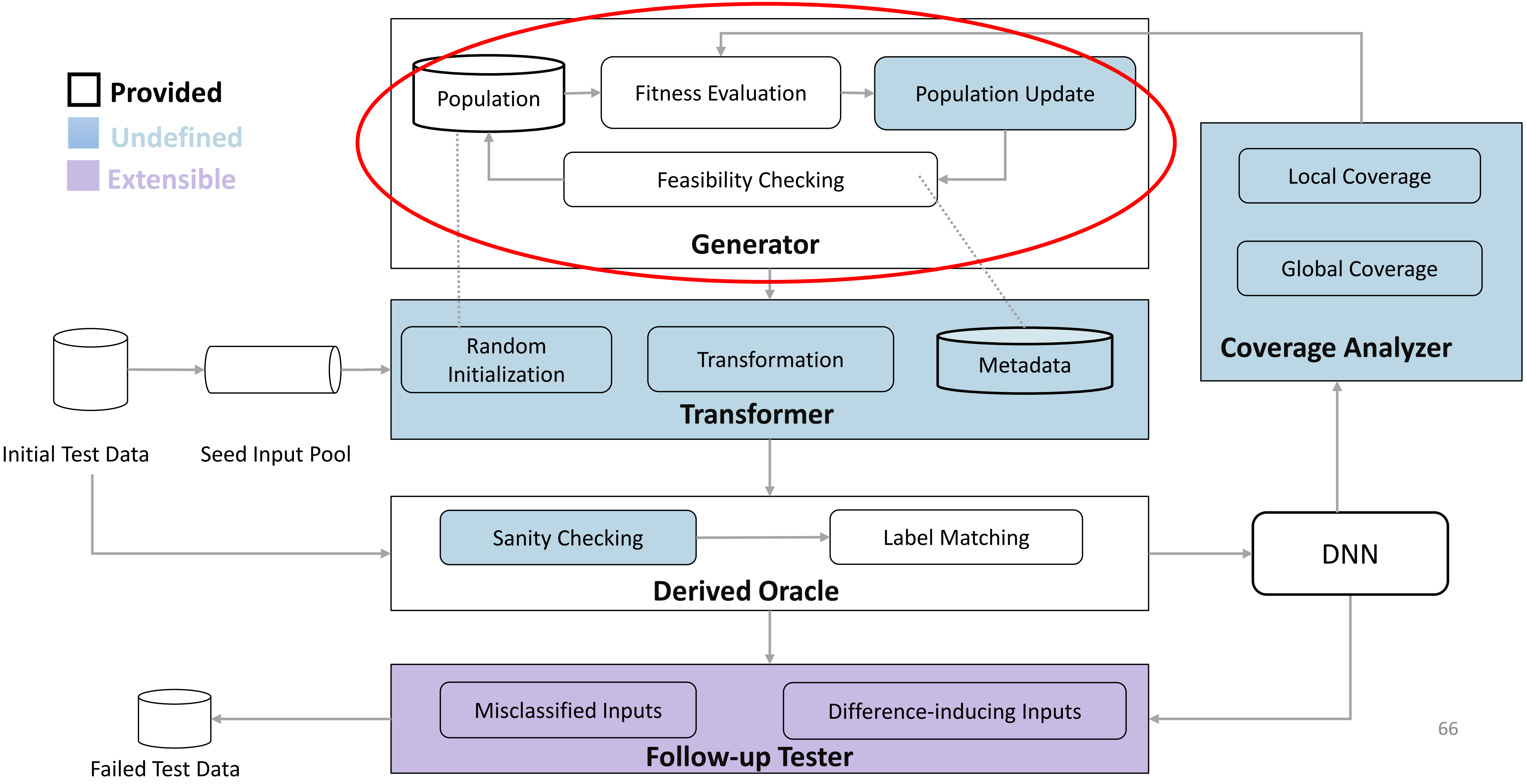


Activated

Deactivated

Threshold(pre-defined)

DeepEvolution: DL-based Software Testing Workflow

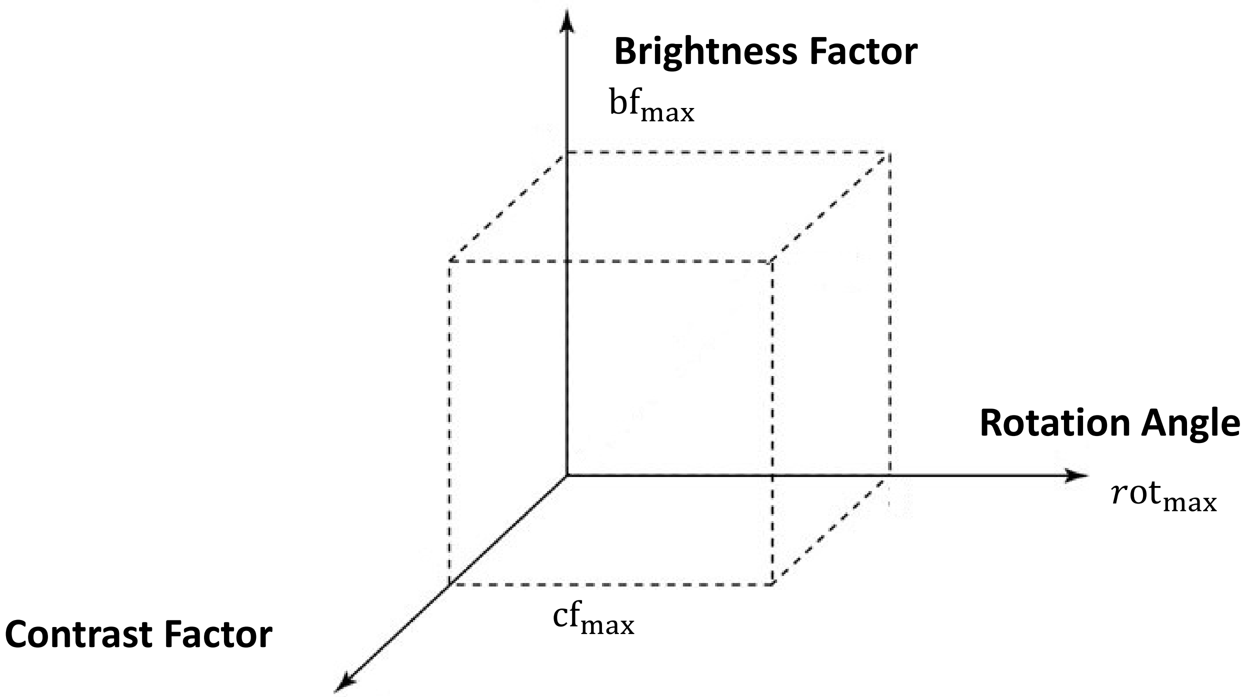


Vectorization of our metamorphic image-based transformations

The **vector encoding** of the **compound metamorphic transformation**:

Brightness Factor	Contrast Factor	…	Translation X	Translation Y	Rotation Angle	…

$\underbrace{\text{Brightness Factor} \quad \text{Contrast Factor} \quad \dots}_{\text{Parameters of pixel transformations}}$ $\underbrace{\text{Translation X} \quad \text{Translation Y} \quad \text{Rotation Angle} \quad \dots}_{\text{Parameters of affine transformations}}$

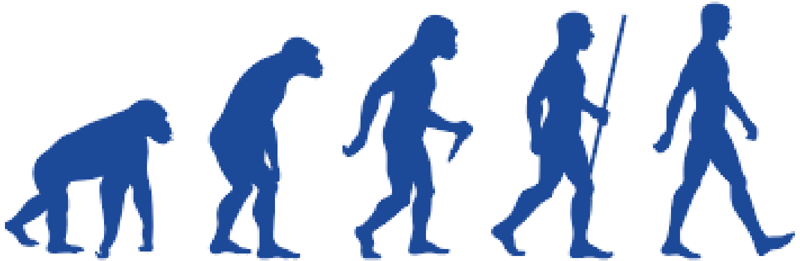


Constrained Multidimensional Space

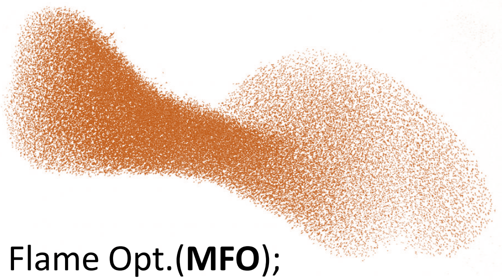
Nature-Inspired Metaheuristic for exploring the transformations' space

No Free Lunch Theorem

Evolution-Based metaheuristics: Genetic Algorithm(GA).



Swarm-Based metaheuristics: PSO, CSA,BAT, GWO, MFO, WOA, MVO, FFA, and SSA.



Particle Swarm Opt. (**PSO**);
Cuckoo Search Algo. (**CSA**);
Bat Algo. (**BAT**);

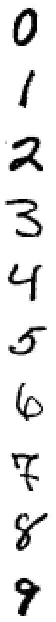
Gray Wolf Opt. (**GWO**);

Moth Flame Opt.(**MFO**);
Whale Opt. Algo. (**WOA**);

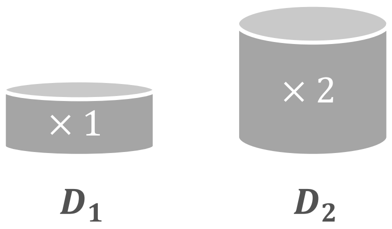
Multi-Verse Opt. (**MVO**);
Firefly Algo. **(FFA**);

Salp Swarm Algo. (**SSA**)

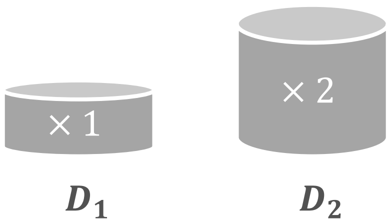
DeepEvolution outperformed TensorFuzz in finding defects introduced during model quantization!



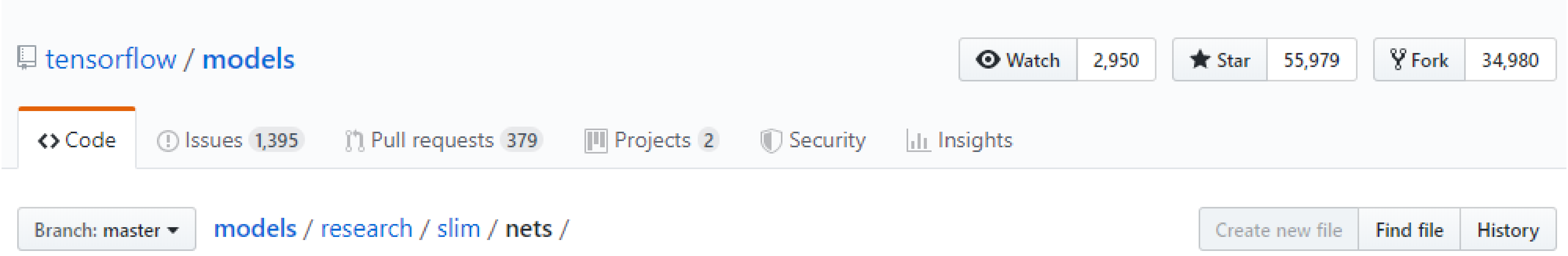
MNIST dataset :
- 28x28 grayscale images
- 10 classes



CIFAR-10 dataset :
- 32x32 color images
- 10 classes



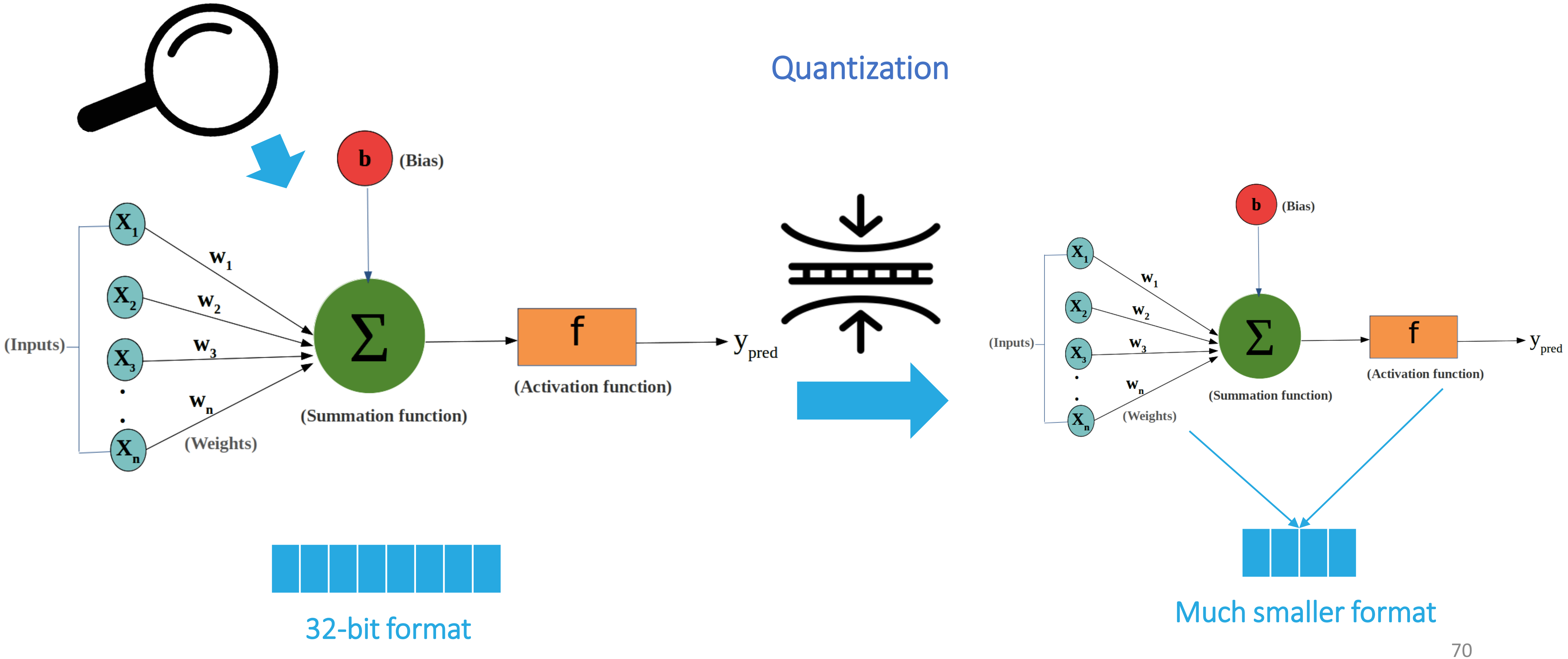
TensorFlow



DiverGet: a search-based software testing approach for Deep Neural Network quantization assessment

Ahmed Haj Yahmed[1] · Houssem Ben Braiek[1] · Foutse Khomh[1] · Sonia Bouzidi[2]
Rania Zaatour[3]

EMSE’22

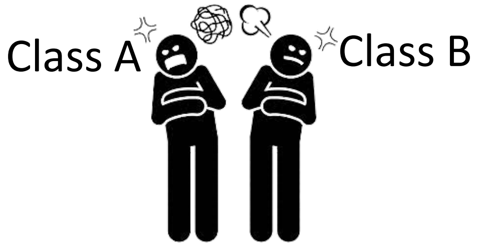


Search-based Software Testing Framework

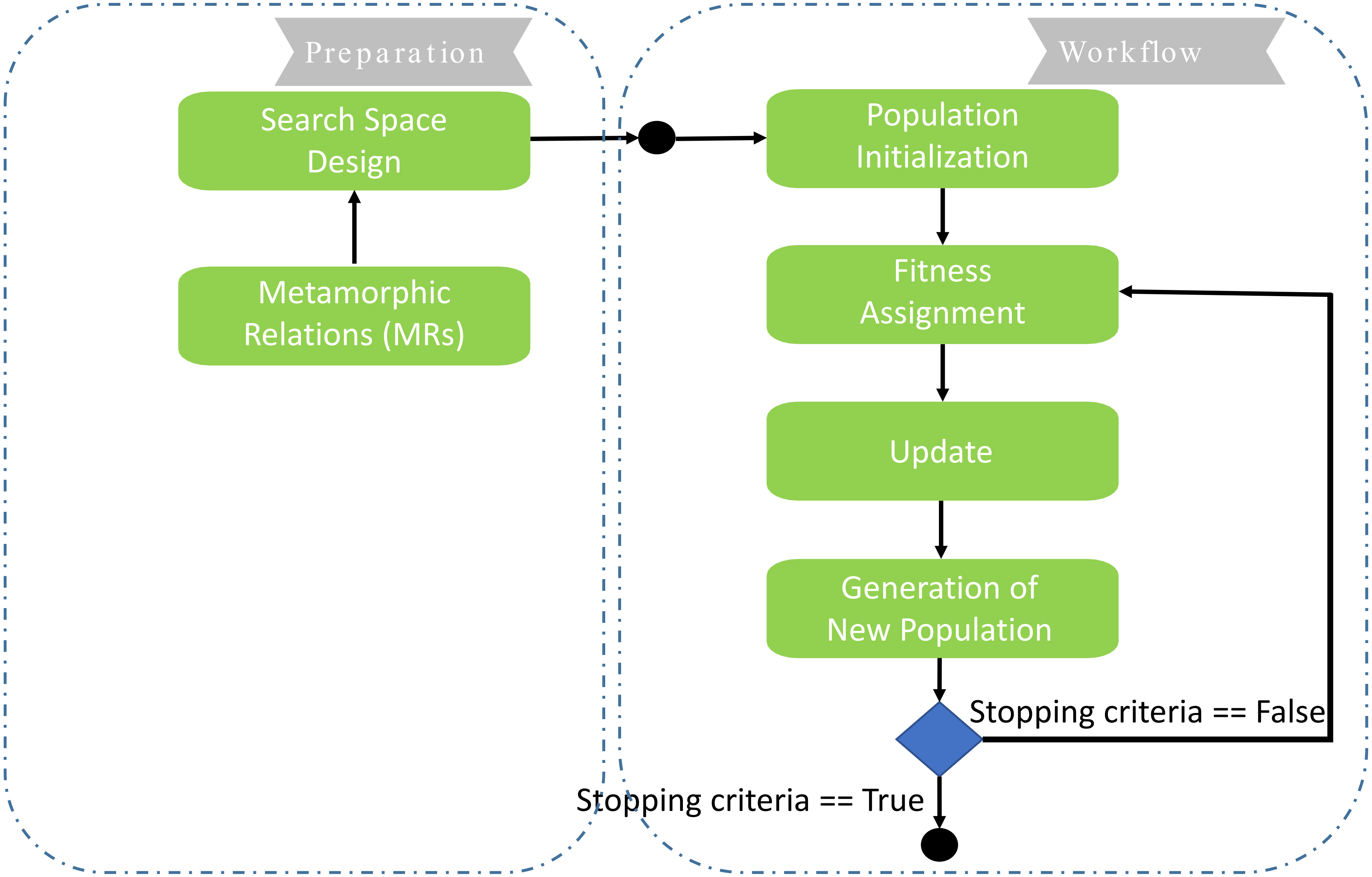
Dedicated to Quantization Assessment

DiverGet

Detecting Difference-Inducing Inputs



Behavioral Disagreements between DNN versions


Preparation
Search Space Design
Metamorphic Relations (MRs)
Workflow
Population Initialization
Fitness Assignment
Update
Generation of New Population
Stopping criteria == False
Stopping criteria == True

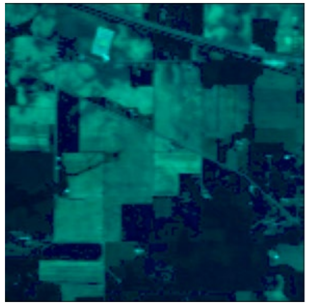
MRs: Metamorphic Relation Formulation

$$\boxed{(m_o(T_r(X_i)) = m_q(T_r(X_i))) \wedge (m_o(X_i) = y_i), \quad \forall i \in \{1 \,..\, N\}}$$

$$\begin{cases} m_o : \text{Original Model} \\ m_q : \text{Quantized Model} \\ T_r : \text{Naturally-Occurring Distortion} \\ (X_i, y_i) : \text{Data point and its Ground Truth} \end{cases}$$

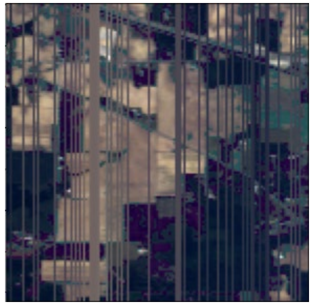
MRs: Naturally-Occurring Radiometric Distortions

Original Image



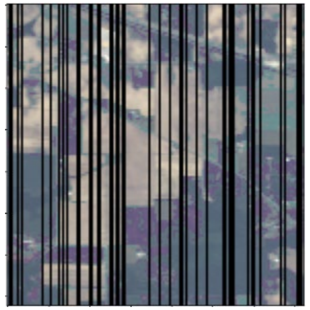
Spectral Band Loss

Line Stripping



Column Stripping

Continuous Line Dropout



Continuous Column Dropout

Region Dropout

Discontinuous Line Dropout

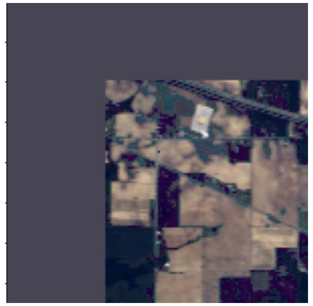
Discontinuous Column Dropout

Salt and Pepper Noise

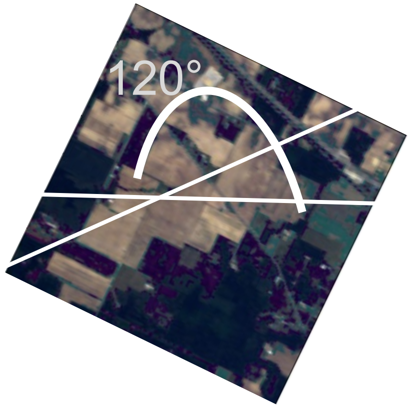
MRs: Naturally-Occurring Geometric Distortions

Original Image

Zoom In

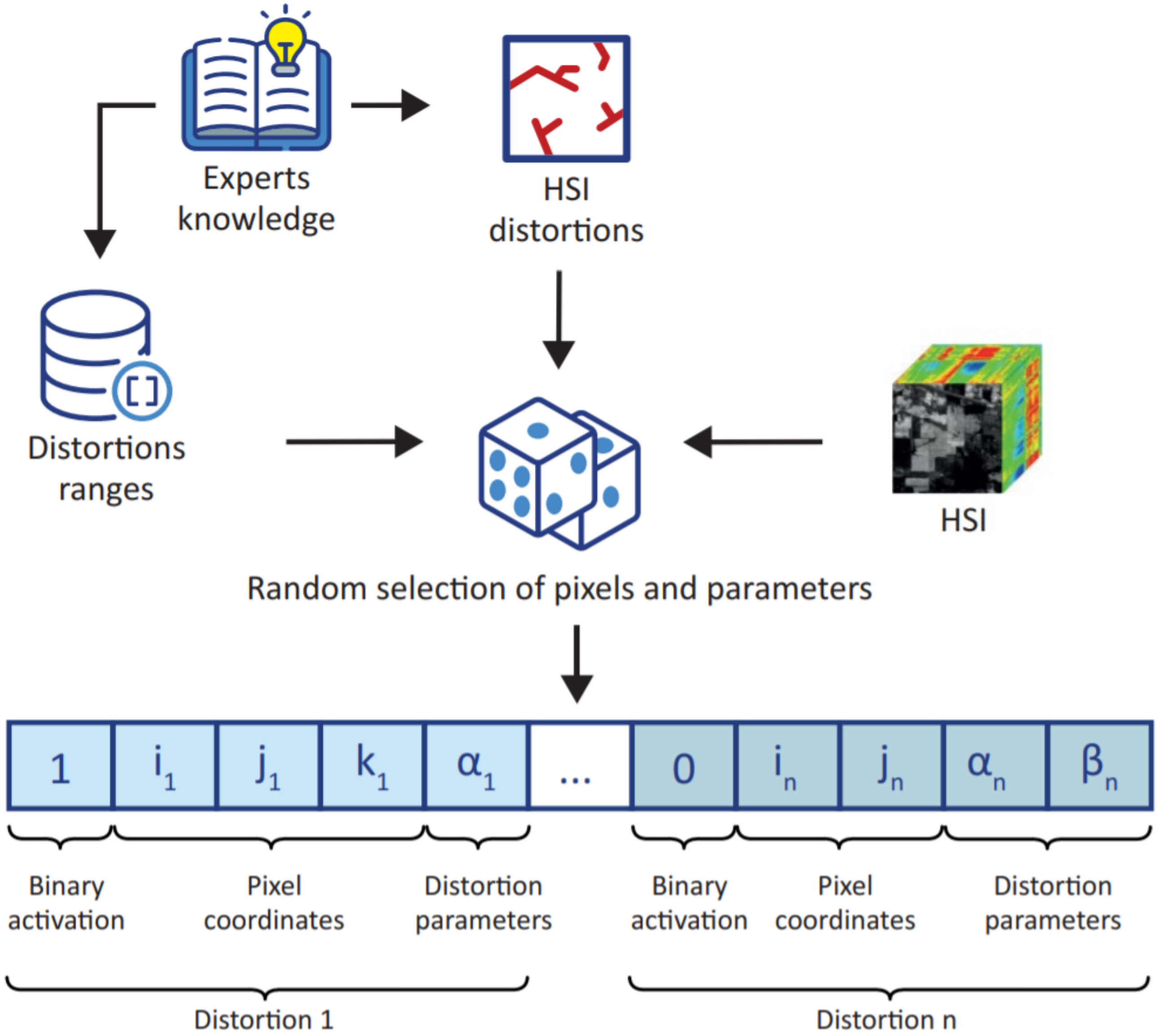


Zoom Out

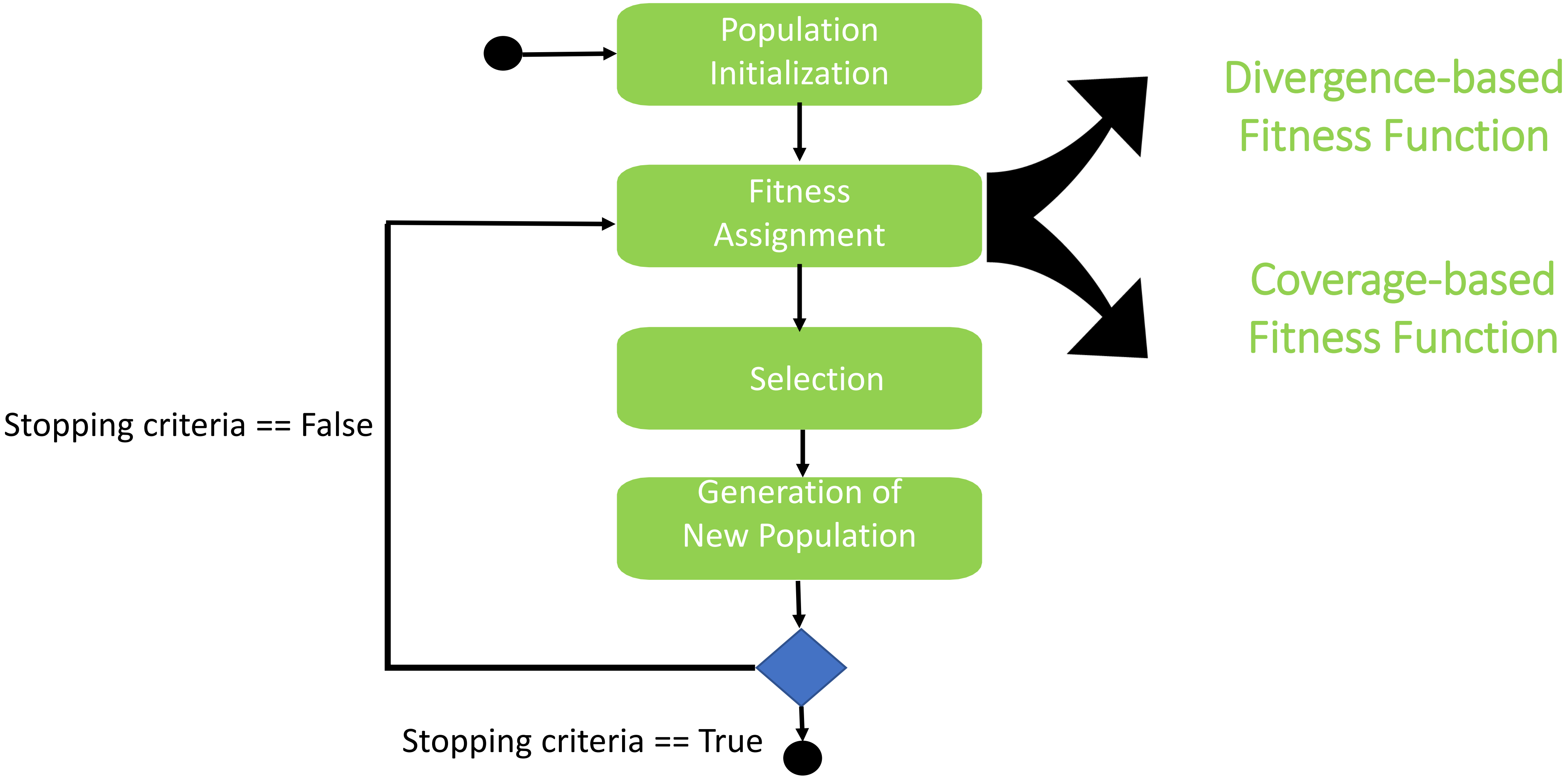


Rotation

Vectorization of our metamorphic image-based transformations



Fitness Function Design



Fitness Function Design

Divergence-based Fitness Function

$$f_X^{div}(\hat{x}) = J(\boldsymbol{s}_o(\hat{x}), \boldsymbol{s}_q(\hat{x}))$$

$$J(Q||R) = \frac{1}{2}(D(Q||M) + D(R||M))$$

$$where\ D(Q||R) = \sum_{i=1}^{c} Q(i) \ln(\frac{Q(i)}{R(i)})$$

$$and\ M = \frac{1}{2}(Q+R).$$

$$s_i = \sigma(l_i) = \frac{e^{l_i}}{\sum_{j=1}^{c} e^{l_j}} \quad for\ i = 1, ..., c$$

$$where\ e^x\ is\ the\ exponential\ function$$

Coverage-based Fitness Function

$$f_X^{cov}(\hat{x}) = -J(S_o^{\hat{x}}, S_q^{\hat{x}})$$

$$J(S_o^x, S_q^x) = \frac{|S_o^x \cap S_q^x|}{|S_o^x| + |S_q^x| + |S_o^x \cap S_q^x|}$$

$$S^x = \{S_i^{n_m} | \phi(\mathbf{x}, n) \in S_i^n\}, \quad \forall m \in [1, M]\}$$

Evaluation of DiverGet

- RQ1: How effective is DiverGet's main feature (i.e., the domain-specific metamorphic relations and the search-based data transformation) at finding difference-inducing inputs?

- RQ2: How does DiverGet compare to DiffChaser?

Evaluation Subjects

Dataset

Pavia University (PU)

Salinas (SA)

Models

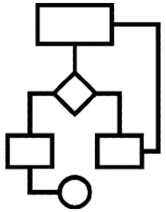
Spectral-Spatial Residual Network (SSRN)

Hybrid Spectral Neural Network (HybridSN)

Metaheuristics

Particle Swarm Optimization (PSO)

Genetic Algorithm (GA)



Quantization methods

Post Training Quantization (PTQ)

Quantization Aware Training (QAT)

TensorFlow Lite

RQ1: The effectiveness of DiverGet as a novel quantization assessment framework

Naturally-occurring synthetic inputs **vs** original test inputs:

Model	Dataset	Quantization	# DII - Original Test Data	# DII - Random sampling (RS)
SSRN	PU	PTQ	136	1609
		QAT	1	763
	SA	PTQ	0	132
		QAT	40	498
hybridSN	PU	PTQ	0	133
		QAT	1	522
	SA	PTQ	0	110
		QAT	10	506

 # DII: number of Difference-Inducing Inputs

Finding 1: the designed domain-specific metamorphic relations **expose uncovered divergences** caused by quantization that original test data **failed to highlight**.

RQ1: The effectiveness of DiverGet as a novel quantization assessment framework

Population-based metaheuristic algorithms **vs** Random Sampling

Model	Dataset	Quantization	RS		DiverGet	
			DiR	VR	DiR	VR
SSRN	PU	PTQ	1.07	3.75	24.05	75.82
		QAT	0.48	3.89	15.68	70.28
	SA	PTQ	0.03	3.38	5.05	70.43
		QAT	0.33	3.45	18.27	70.72
hybridSN	PU	PTQ	0.08	3.66	8.43	67.77
		QAT	0.43	3.84	10.92	67.52
	SA	PTQ	0.02	2.[illegible]	3.07	67.[illegible]6
		QAT	0.25	2.85	9.96	6[illegible].18

DiR: Divergence Rate
VR: Validation Rate

Finding 2: DiverGet's searching strategy using **population-based metaheuristic** succeed in **outperforming** the **Random Sampling** strategy into steering the generation into prominent regions.

RQ2: DiverGet vs. DiffChaser

Framework	Model	PU				SA				Average	
		PTQ		QAT		PTQ		QAT			
		DiR	SR	DiR	SR	DiR	SR	DiR	SR	DiR	SR
DiffChaser	SSRN	16.66	49.38	0.31	10.63	0.35	9.69	3.68	16.58	2.78	11.25
	Hybrid -SN	0.001	0.31	0.002	0.31	0.001	0.63	1.22	2.50		
DiverGet (PSO)	SSRN	24.96	71.25	16.42	61.25	3.60	43.75	13.92	63.44	14.59	40.98
	Hybrid -SN	16.97	24.38	20.08 (**)	28.75	9.35	14.06	11.42	20.94 (*)		
DiverGet (GA)	SSRN	35.90	58.75	28.86	37.50	14.47	20.63	31.93	43.75	20.40	27.27
	Hybrid -SN	12.06	13.75	19.22 (**)	20.94	5.32	5.63	15.40	17.19 (*)		

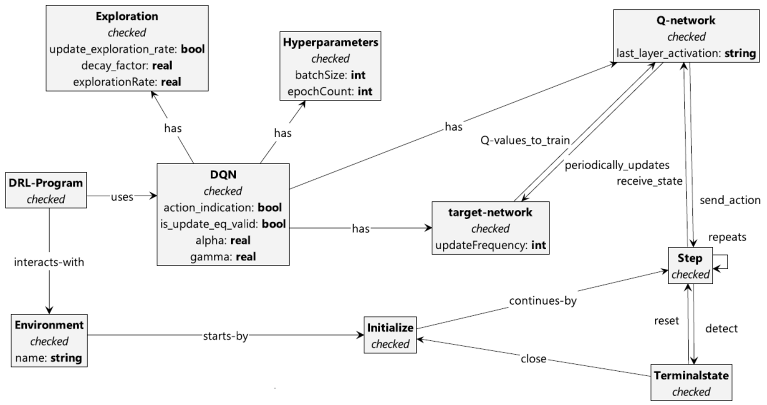
DiverGet **outperforms** DiffChaser in terms of **number of revealed disagreements** with a higher **success rate!**

Faults in Deep Reinforcement Learning Programs: A Taxonomy and A Detection Approach

Amin Nikanjam · Mohammad Mehdi Morovati · Foutse Khomh · Houssem Ben Braiek

A probabilistic framework for mutation testing in deep neural networks

Florian Tambon *, Foutse Khomh, Giuliano Antoniol
Department of Software Engineering - Polytechnique Montreal, 2500, chemin de Polytechnique, Montreal, H3T1J4, Quebec, Canada

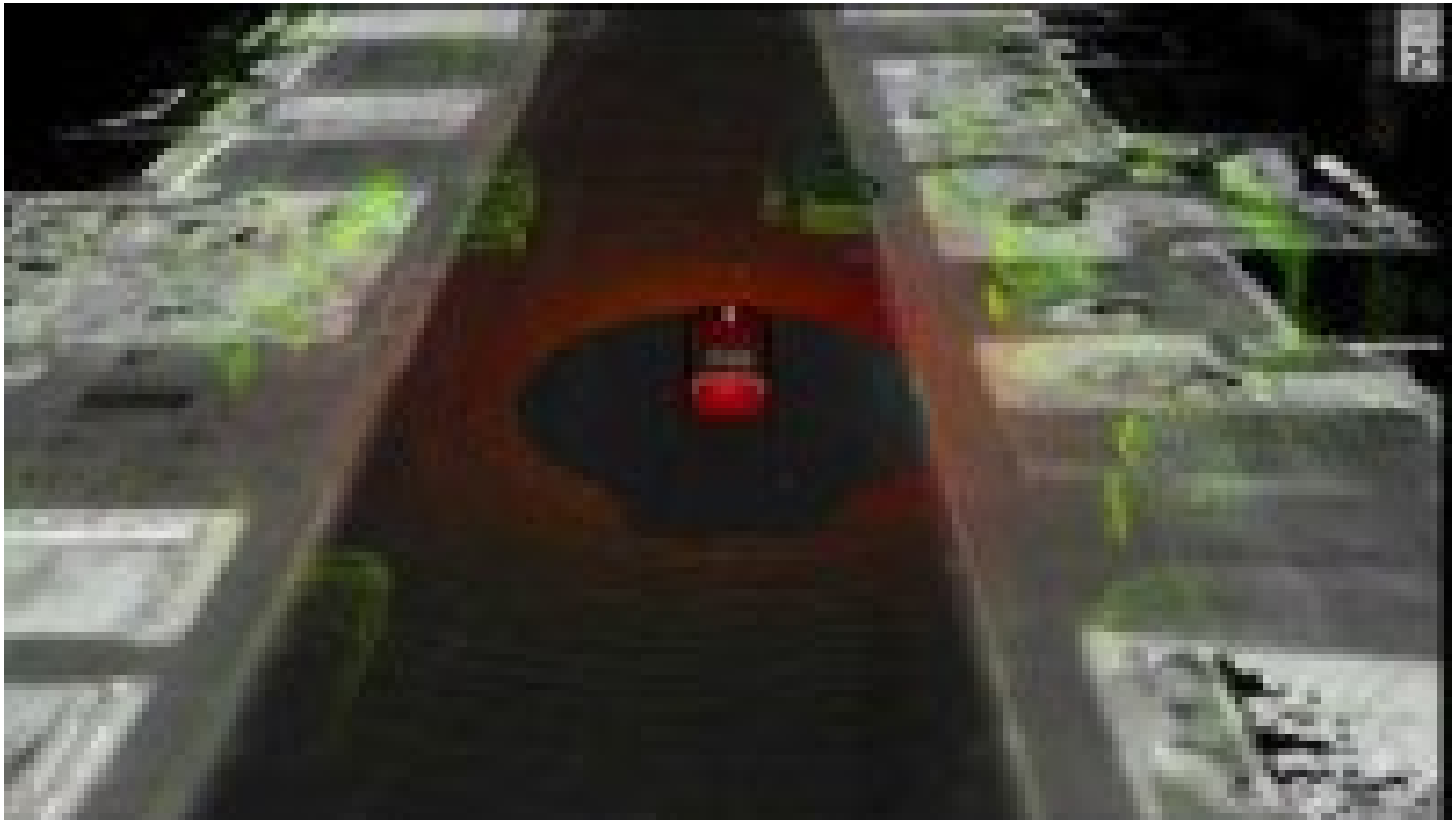


Mutation Testing of Deep Reinforcement Learning Based on Real Faults

Automated Quality Assurance Tools are essential!

Adversarial weather conditions

Complex corner cases



Software in the loop testing!

Realistic simulator (CARLA, LGSVL, BeamNG)

Test scenario specification → 

→ Output traces of system behavior

We aim to generate fault-revealing scenarios!

Challenges:

- Vast search space
- Evaluating test scenarios is expensive
- The need for diverse test scenarios

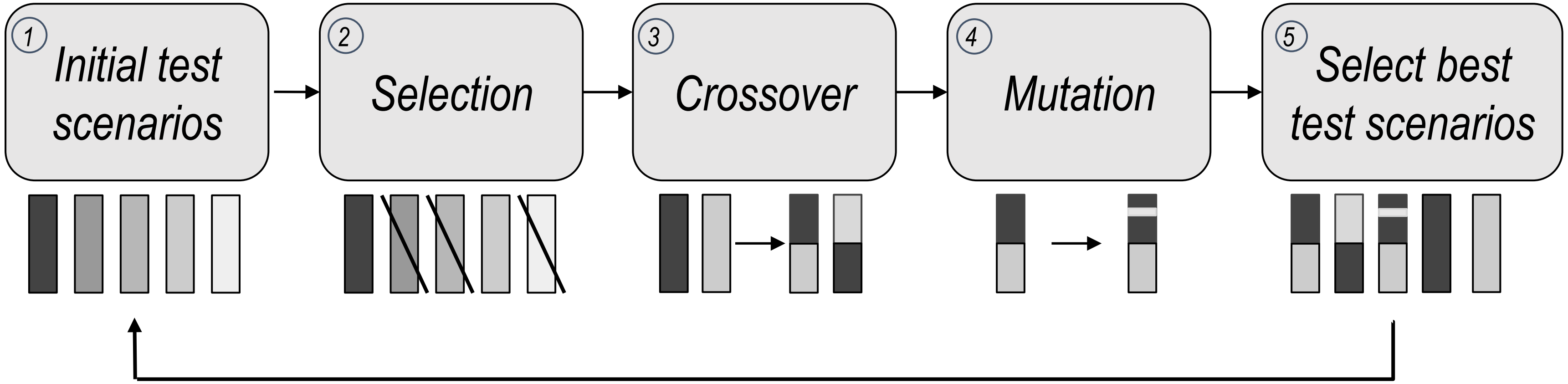
AmbieGen: A Search-based Framework for Autonomous Systems Testing

Dmytro Humeniuk, Foutse Khomh, Giuliano Antoniol

IST'21

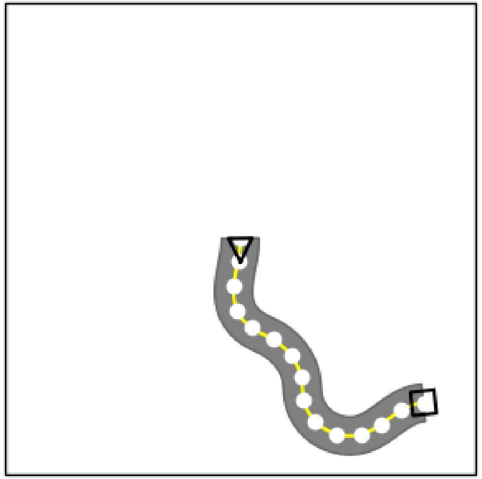
Multi-objective search algorithm (NSGA-II) with 2 objectives:

- Maximize the difficulty of test scenarios, respecting the constraints
- Maximize the diversity of test scenarios

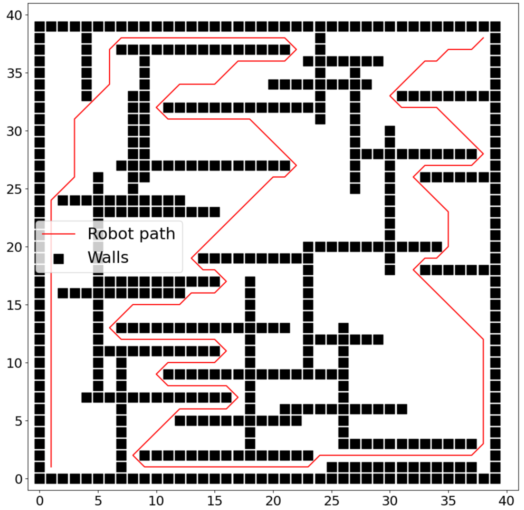


Flexible representation, applicable to different test problems

	Element 1	*Element 2*	*Element N*
Element type	Straight segment	Curved segment	Curved segment
Parameter 1	Segment length 10		
Parameter 2		Turning angle 60	Turning angle 30



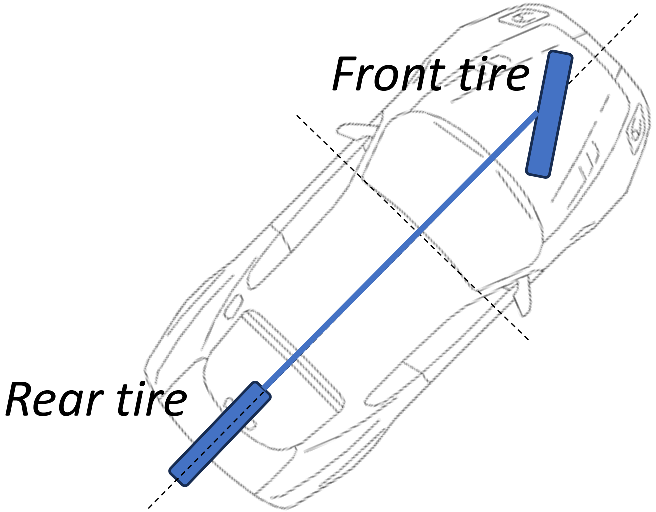
Lane keeping system testing



Mobile robots testing

Using a simplified model of the system to guide the search

Vehicle kinematic bicycle model



Try it out!

Quite effective, achieving the 1st place in SBST 2022 competition

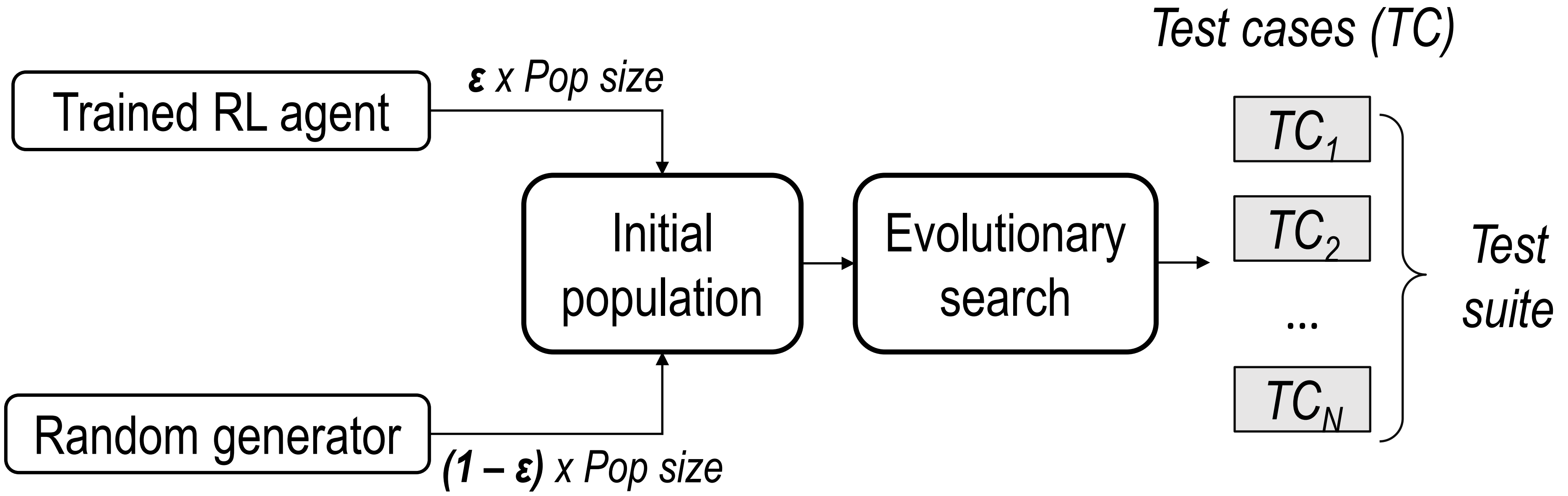
TOOL	BEAMNG.AI	DAVE2
ADAFRENETIC	0.183	0.044
AMBIEGEN	0.544	0.333
FRENETICV	0.447	0.302
GENRL	0.237	0.211
EVOMBT	0.216	0.200
WOGAN	0.514	0.262

Reinforcement learning informed evolutionary search for autonomous system testing

DMYTRO HUMENIUK, Polytechnique Montréal, Canada
FOUTSE KHOMH, Polytechnique Montréal, Canada
GIULIANO ANTONIOL, Polytechnique Montréal, Canada

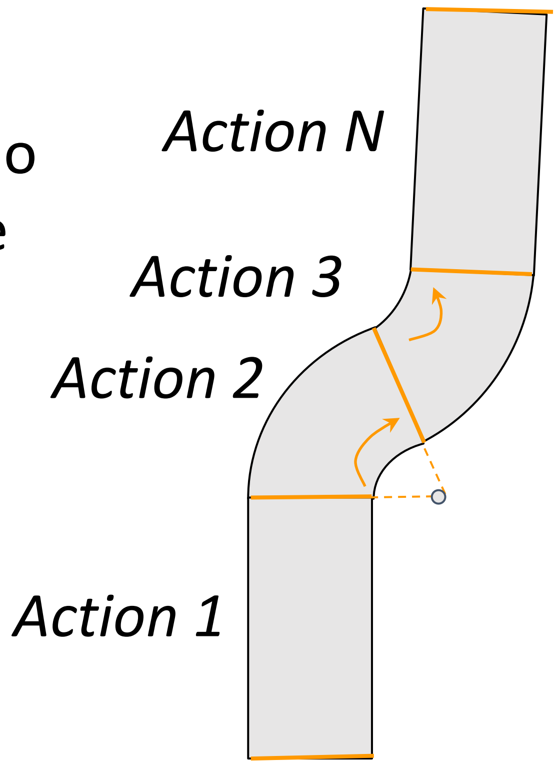
TOSEM'23

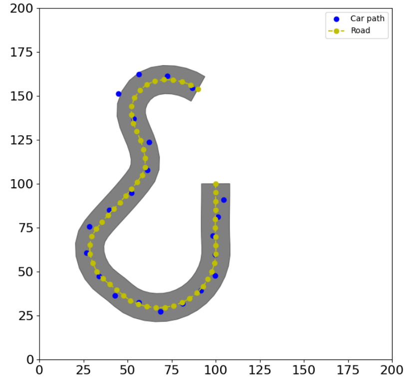
Using gradient based algorithms for smart initialization



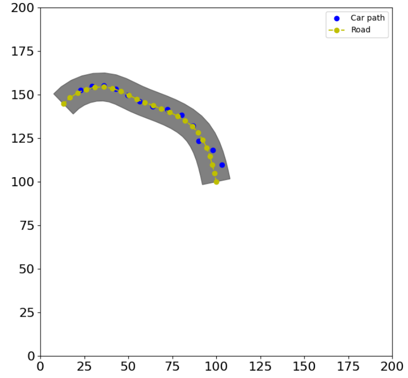
Training the RL agent to generate challenging scenarios with domain knowledge-based rewards

- State: 2D array defining the test scenario
- Actions: new element to add to the scenario
- Reward: using simplified model to estimate the reward
- PPO algorithm





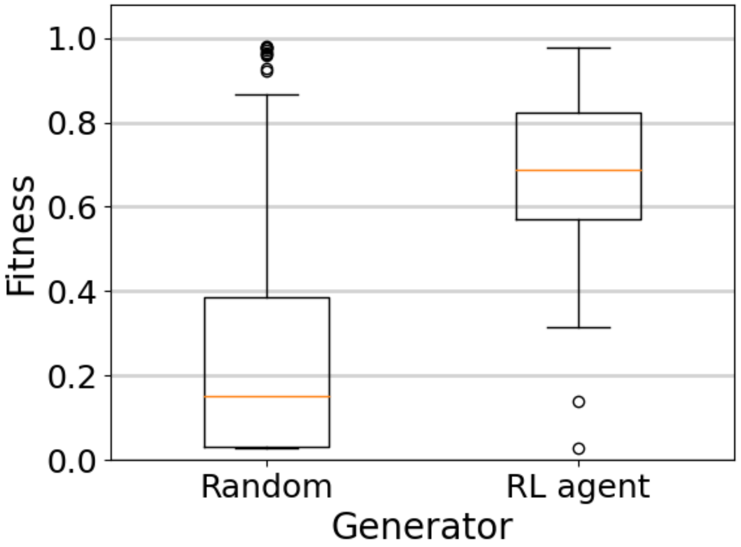
High reward



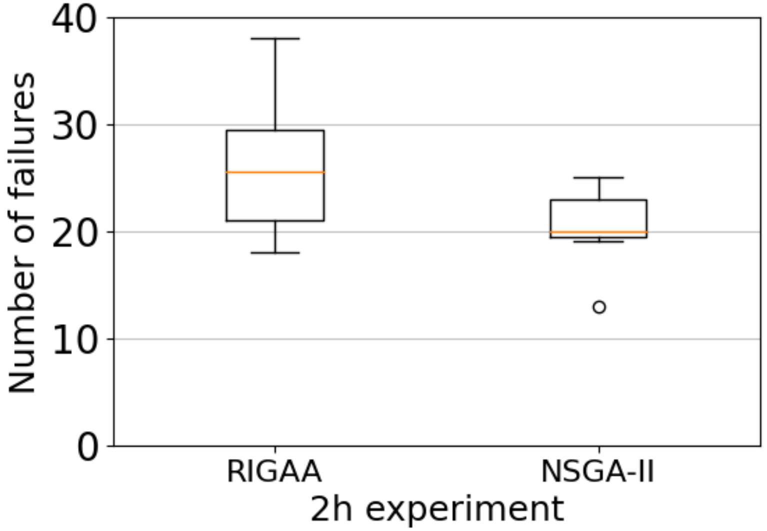
Low reward

RIGAA outperforms MOEA with random initialization

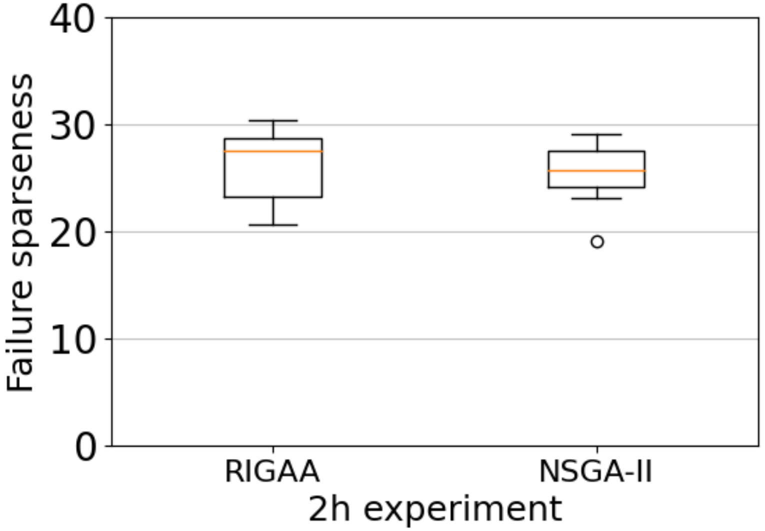
RL vs Random generator



Number of failures

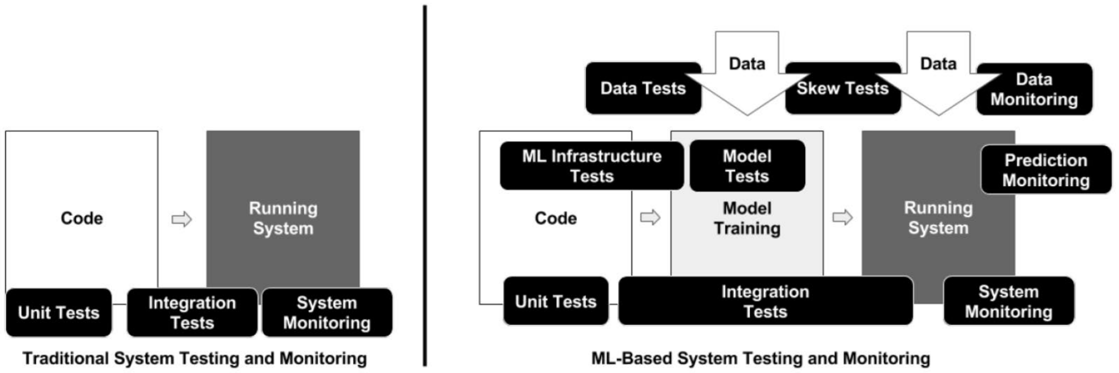


Diversity of failures



Try it out!

Ensuring the safety and auditability of ML-based components is challenging



Deep Learning Model Verification Using Graph Transformations

AMIN NIKANJAM*, K. N. Toosi University of Technology, Iran and SWAT Lab., Polytechnique Montreal, Canada
HOUSSEM BEN BRAIEK*, SWAT Lab., Polytechnique Montreal, Canada
MOHAMMADMEHDI MOROVATI, SWAT Lab., Polytechnique Montreal, Canada
FOUTSE KHOMH, SWAT Lab., Polytechnique Montreal, Canada

Testing Neural Networks Training Programs

HOUSSEM BEN BRAIEK, SWAT Lab., Polytechnique Montreal, Canada
FOUTSE KHOMH, SWAT Lab., Polytechnique Montréal, Canada

NeuraLint : A linter for DL programs

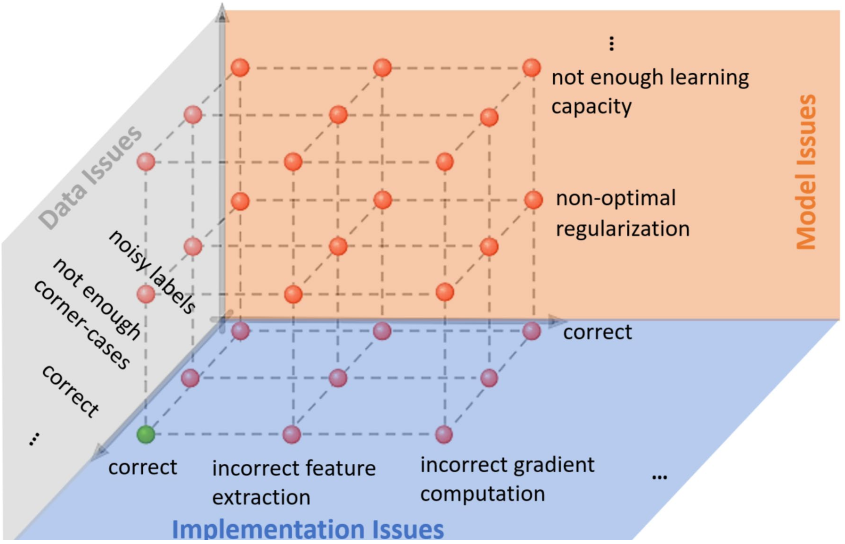
- ✓ Capture defects early, so saves rework cost
- ✓ Less expensive, because it doesn't require execution
- ✓ Find defects in seconds
- ✓ …

TheDeepChecker : Dynamic testing of DL programs

- ✓ Capture defects during the training process
- ✓ Less expensive than testing the resulting model
- ✓ Finds 30% more defects than AWS SageMaker
- ✓ …

Automated Quality Assurance tools are needed!

Multi-dimensional space of DL bugs



Realistic simulator (CARLA, LGSVL, BeamNG)

Test scenario specification →

→ Output traces of system behavior

We aim to generate fault-revealing scenarios!

Challenges:

- Vast search space
- Evaluating test scenarios is expensive
- The need for diverse test scenarios