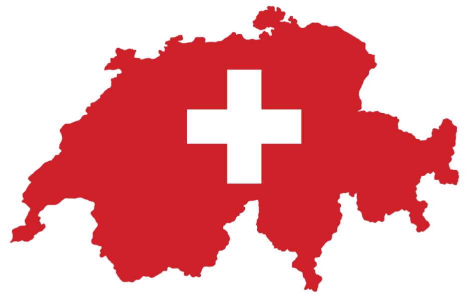
DEEPHYPERION

EXPLORING THE FEATURE SPACE OF DEEP LEARNING BASED SYSTEMS THROUGH ILLUMINATION SEARCH

VINCENZO
RICCIO

@p1ndsvin

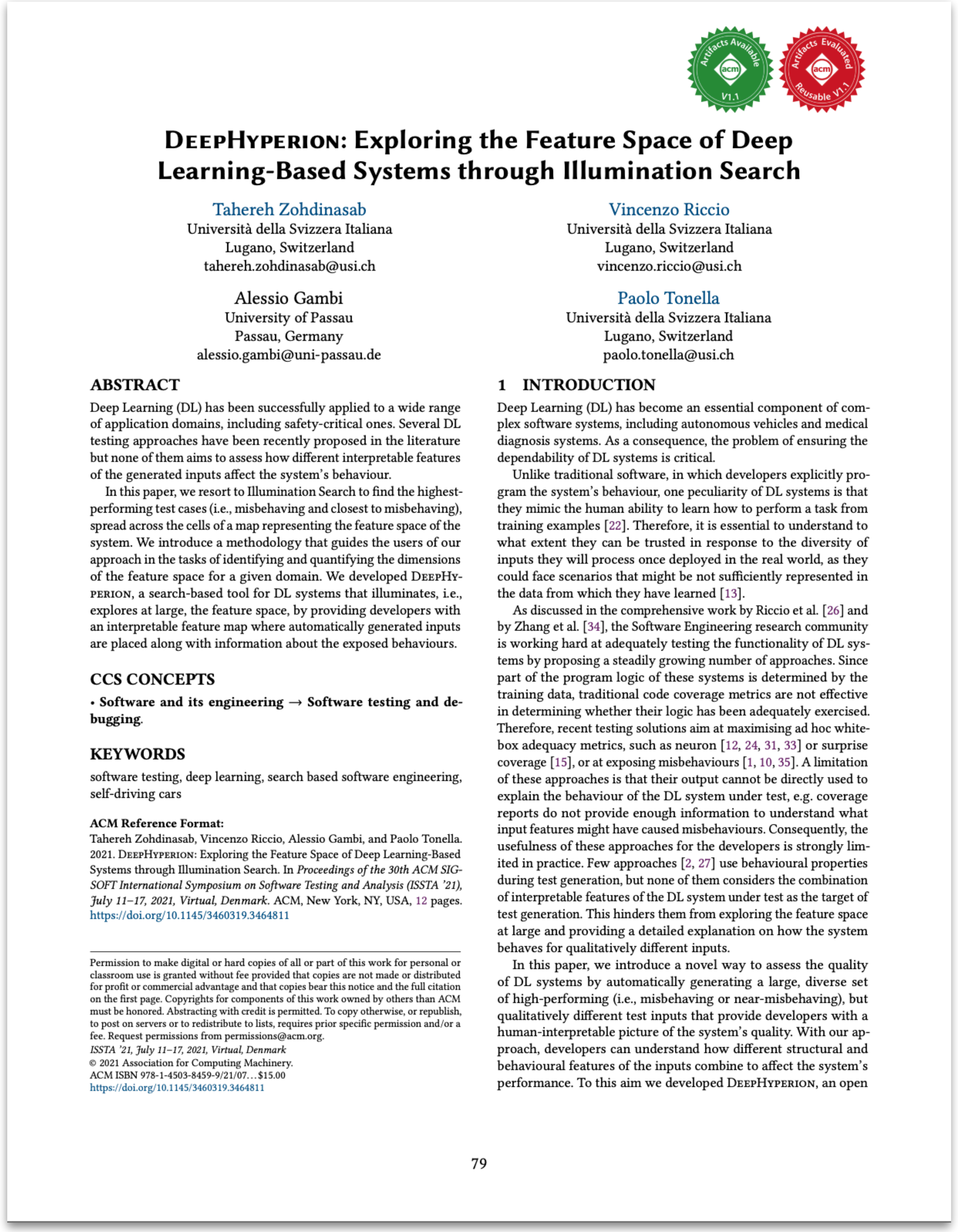
A COLLABORATION WITH



TAHEREH ZOHDINASAB
UNIVERSITÀ DELLA SVIZZERA ITALIANA, SWITZERLAND

PAOLO TONELLA
UNIVERSITÀ DELLA SVIZZERA ITALIANA, SWITZERLAND

ALESSIO GAMBI
UNIVERSITY OF PASSAU, GERMANY



DEEPHYPERION: Exploring the Feature Space of Deep Learning-Based Systems through Illumination Search

Tahereh Zohdinasab
Università della Svizzera Italiana
Lugano, Switzerland
tahereh.zohdinasab@usi.ch

Vincenzo Riccio
Università della Svizzera Italiana
Lugano, Switzerland
vincenzo.riccio@usi.ch

Alessio Gambi
University of Passau
Passau, Germany
alessio.gambi@uni-passau.de

Paolo Tonella
Università della Svizzera Italiana
Lugano, Switzerland
paolo.tonella@usi.ch

ABSTRACT

Deep Learning (DL) has been successfully applied to a wide range of application domains, including safety-critical ones. Several DL testing approaches have been recently proposed in the literature but none of them aims to assess how different interpretable features of the generated inputs affect the system's behaviour.

In this paper, we resort to Illumination Search to find the highest-performing test cases (i.e., misbehaving and closest to misbehaving), spread across the cells of a map representing the feature space of the system. We introduce a methodology that guides the users of our approach in the tasks of identifying and quantifying the dimensions of the feature space for a given domain. We developed DEEPHYPERION, a search-based tool for DL systems that illuminates, i.e., explores at large, the feature space, by providing developers with an interpretable feature map where automatically generated inputs are placed along with information about the exposed behaviours.

CCS CONCEPTS

- **Software and its engineering → Software testing and debugging.**

KEYWORDS

software testing, deep learning, search based software engineering, self-driving cars

ACM Reference Format:
Tahereh Zohdinasab, Vincenzo Riccio, Alessio Gambi, and Paolo Tonella. 2021. DEEPHYPERION: Exploring the Feature Space of Deep Learning-Based Systems through Illumination Search. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '21), July 11–17, 2021, Virtual, Denmark*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3460319.3464811

ISSTA '21, July 11–17, 2021, Virtual, Denmark
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8459-9/21/07...$15.00
https://doi.org/10.1145/3460319.3464811

1 INTRODUCTION

Deep Learning (DL) has become an essential component of complex software systems, including autonomous vehicles and medical diagnosis systems. As a consequence, the problem of ensuring the dependability of DL systems is critical.

Unlike traditional software, in which developers explicitly program the system's behaviour, one peculiarity of DL systems is that they mimic the human ability to learn how to perform a task from training examples [22]. Therefore, it is essential to understand to what extent they can be trusted in response to the diversity of inputs they will process once deployed in the real world, as they could face scenarios that might be not sufficiently represented in the data from which they have learned [13].

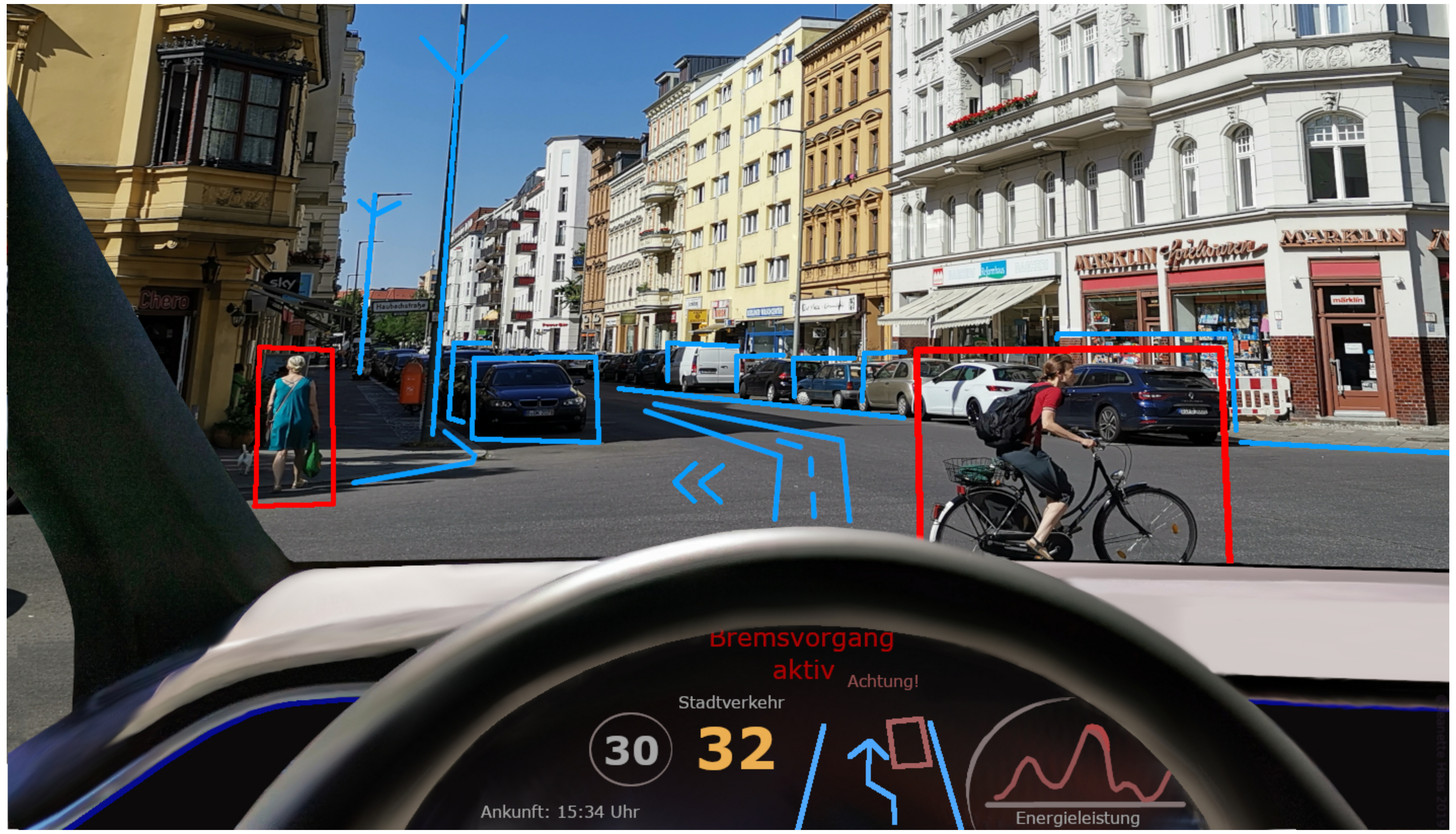
As discussed in the comprehensive work by Riccio et al. [26] and by Zhang et al. [34], the Software Engineering research community is working hard at adequately testing the functionality of DL systems by proposing a steadily growing number of approaches. Since part of the program logic of these systems is determined by the training data, traditional code coverage metrics are not effective in determining whether their logic has been adequately exercised. Therefore, recent testing solutions aim at maximising ad hoc white-box adequacy metrics, such as neuron [12, 24, 31, 33] or surprise coverage [15], or at exposing misbehaviours [1, 10, 35]. A limitation of these approaches is that their output cannot be directly used to explain the behaviour of the DL system under test, e.g. coverage reports do not provide enough information to understand what input features might have caused misbehaviours. Consequently, the usefulness of these approaches for the developers is strongly limited in practice. Few approaches [2, 27] use behavioural properties during test generation, but none of them considers the combination of interpretable features of the DL system under test as the target of test generation. This hinders them from exploring the feature space at large and providing a detailed explanation on how the system behaves for qualitatively different inputs.

In this paper, we introduce a novel way to assess the quality of DL systems by automatically generating a large, diverse set of high-performing (i.e., misbehaving or near-misbehaving), but qualitatively different test inputs that provide developers with a human-interpretable picture of the system's quality. With our approach, developers can understand how different structural and behavioural features of the inputs combine to affect the system's performance. To this aim we developed DEEPHYPERION, an open

ACM SIGSOFT INTERNATIONAL SYMPOSIUM ON SOFTWARE TESTING AND ANALYSIS (ISSTA) 2021

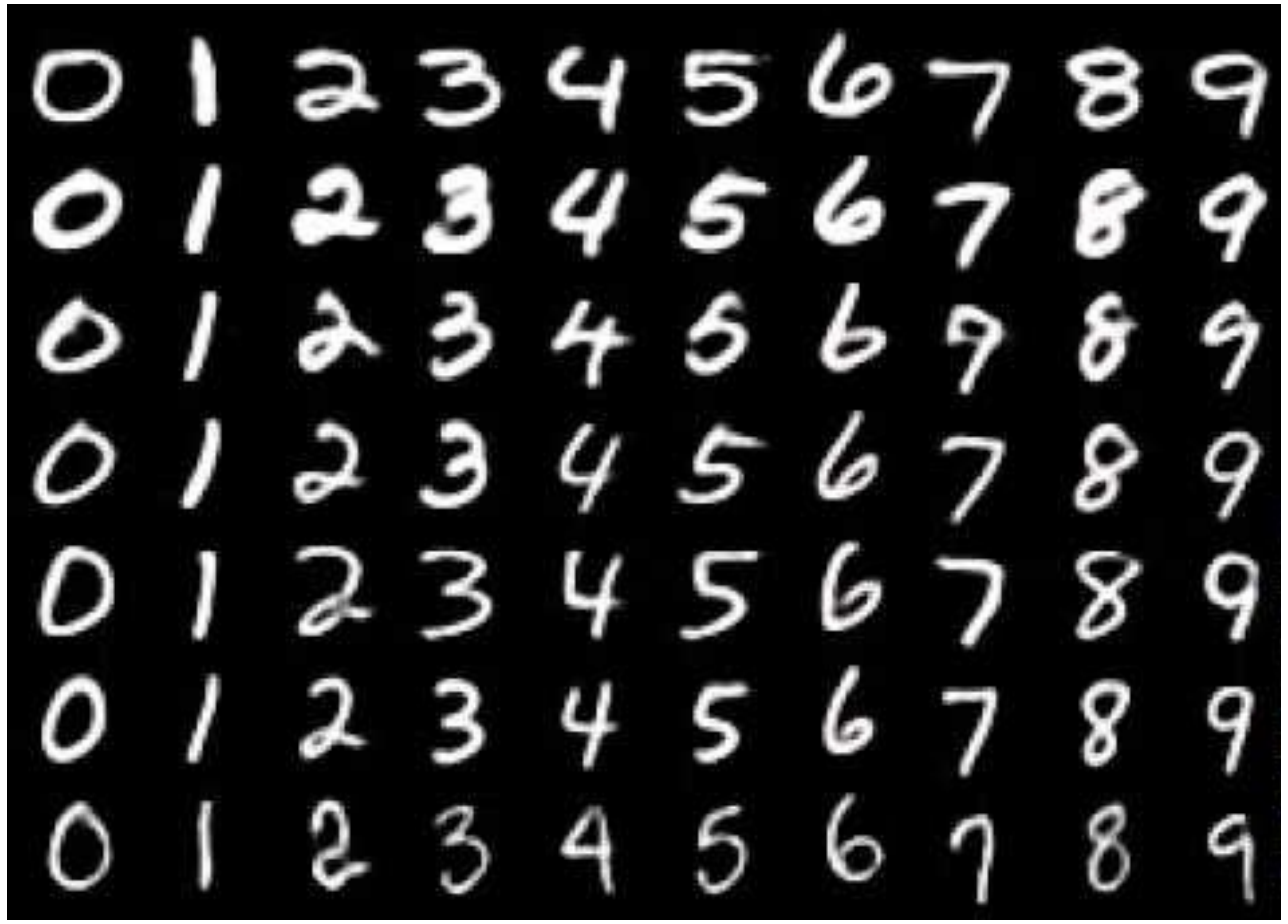
THE AGE OF DEEP LEARNING (DL)

CAN WE TRUST SOFTWARE THAT LEARNS?



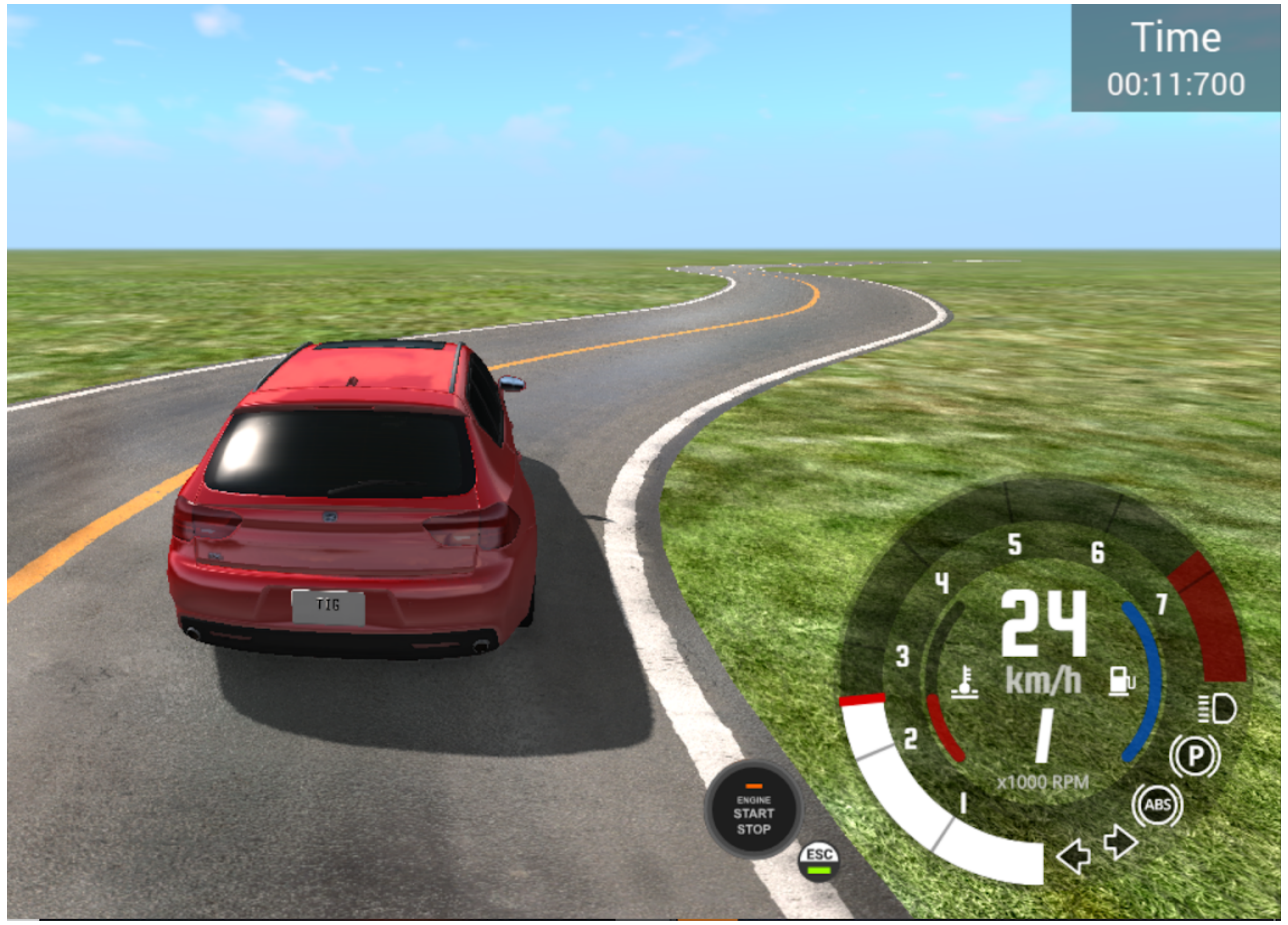
Hi, how can I help?

RESEARCH SUBJECTS



MNIST

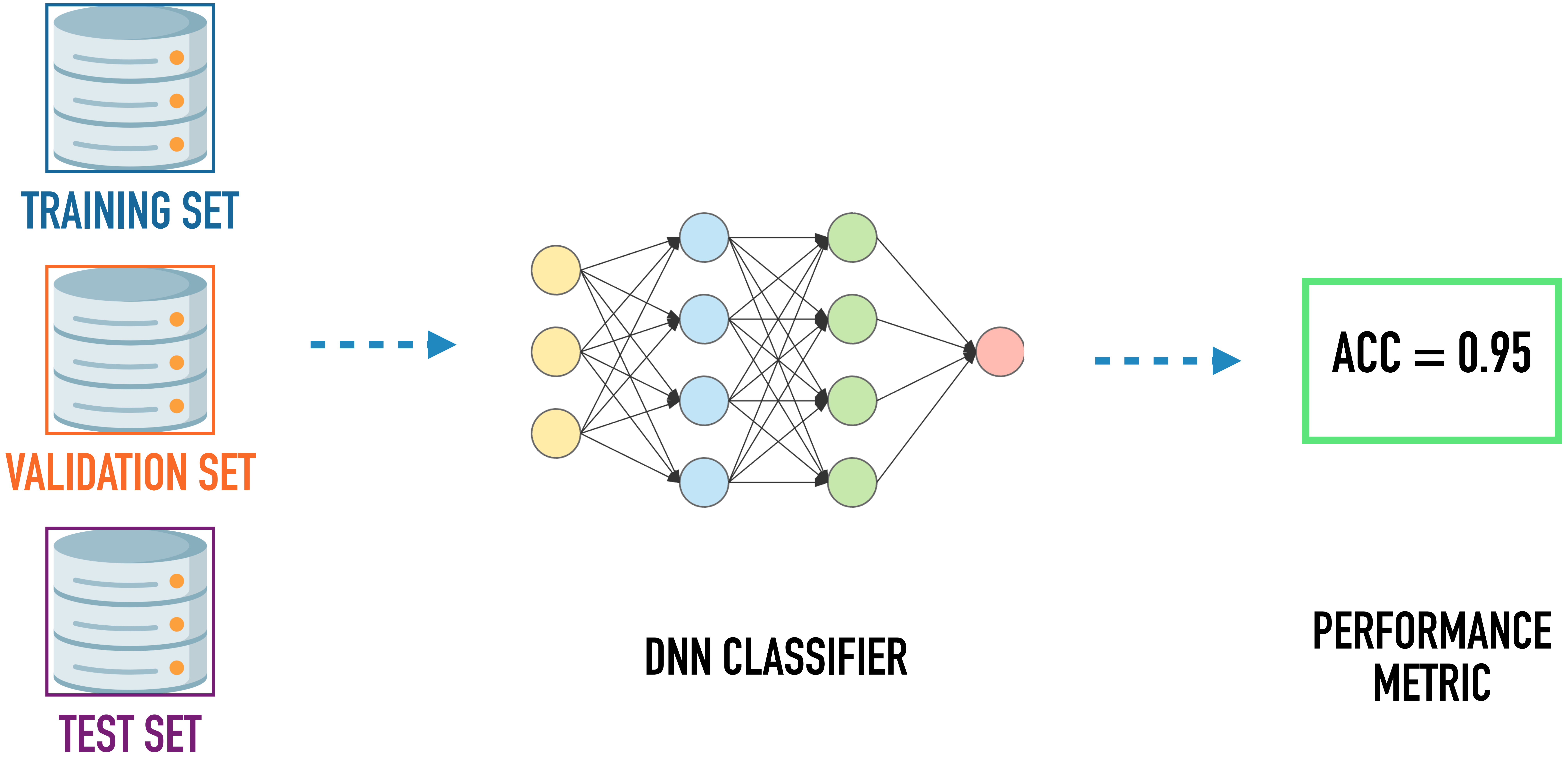
HAND-WRITTEN DIGIT CLASSIFICATION



BEAMNG

LANE-KEEPING FOR AUTONOMOUS CARS

TRADITIONAL DL ASSESSMENT



TRAINING SET

VALIDATION SET

TEST SET

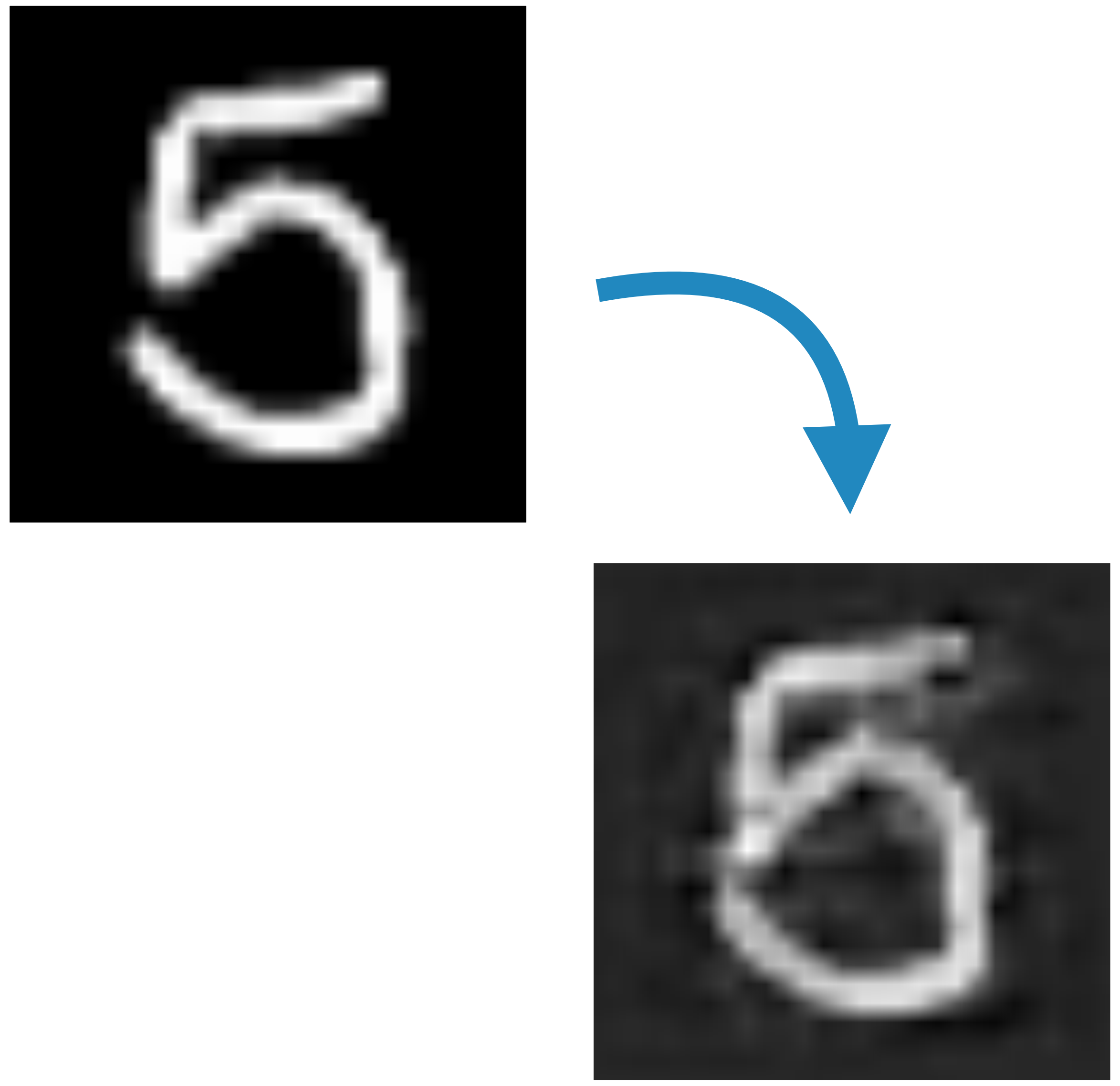
DNN CLASSIFIER

ACC = 0.95

PERFORMANCE METRIC

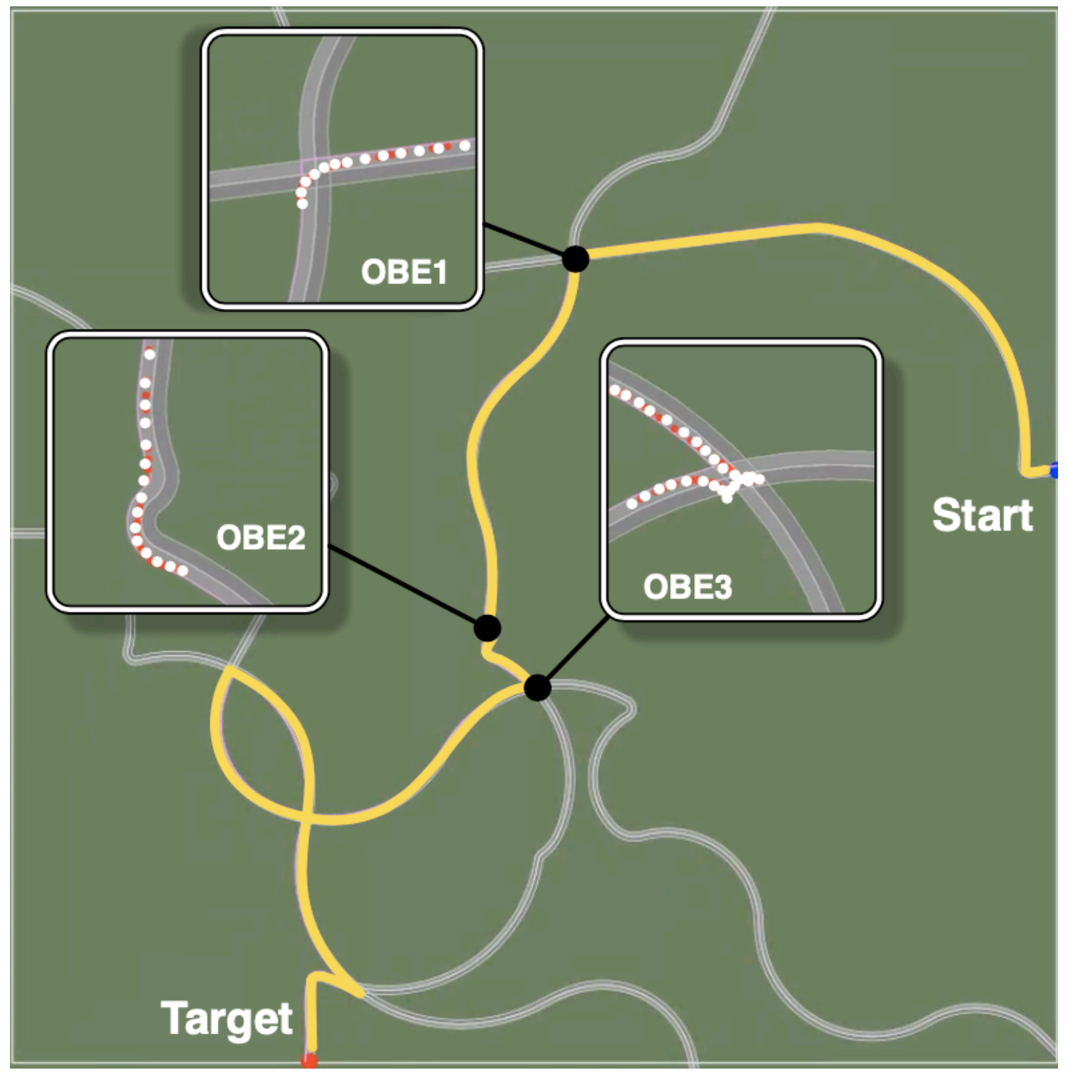
HOW WILL THE DL BASED SYSTEM BEHAVE FOR DATA BEYOND THE ORIGINAL DATASET?

AUTOMATED TEST GENERATION FOR DL BASED SYSTEMS



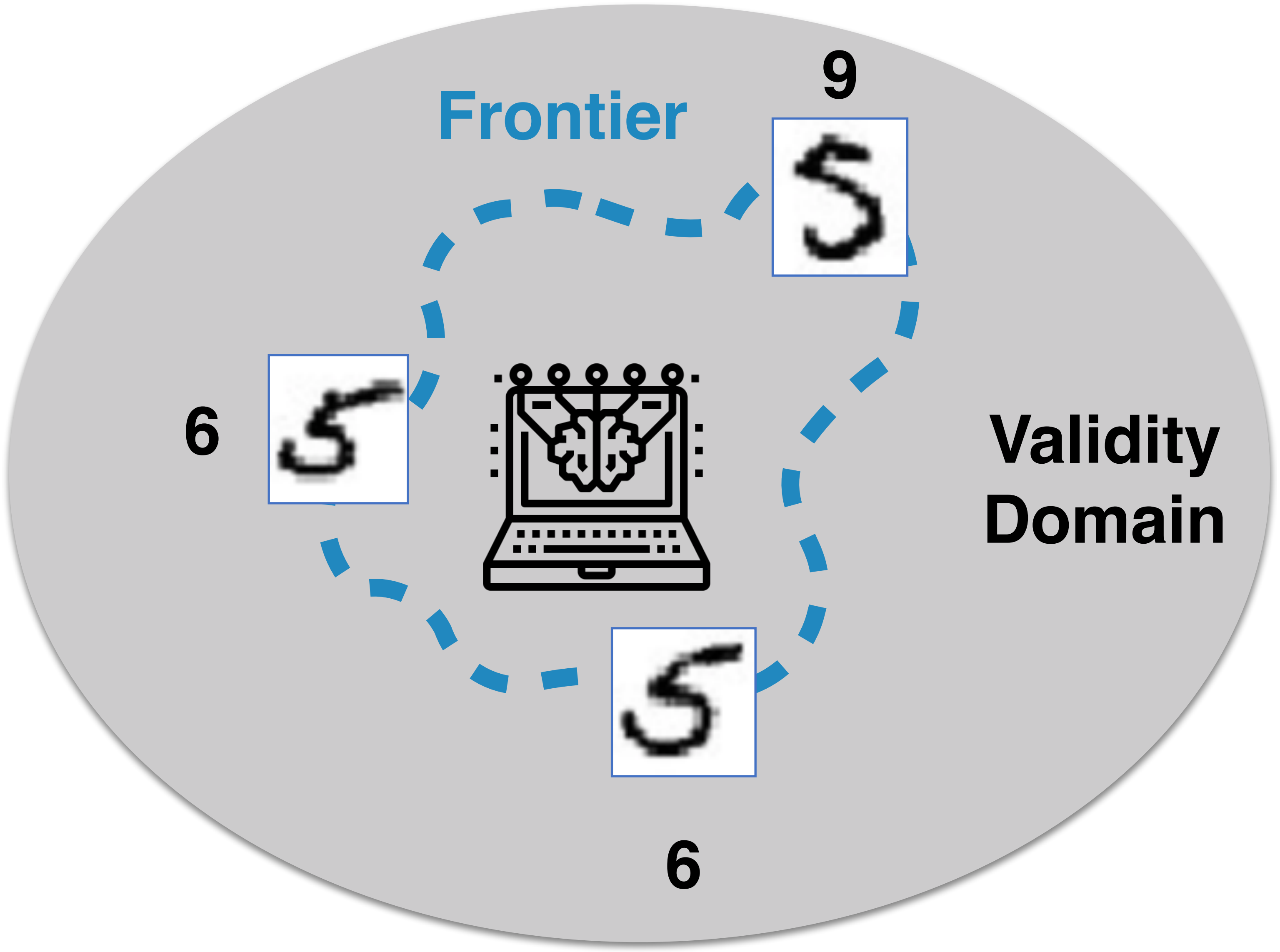
DLFUZZ

ADVERSARIAL IMAGES



ASFAULT

ADVERSARIAL ROAD NETWORKS

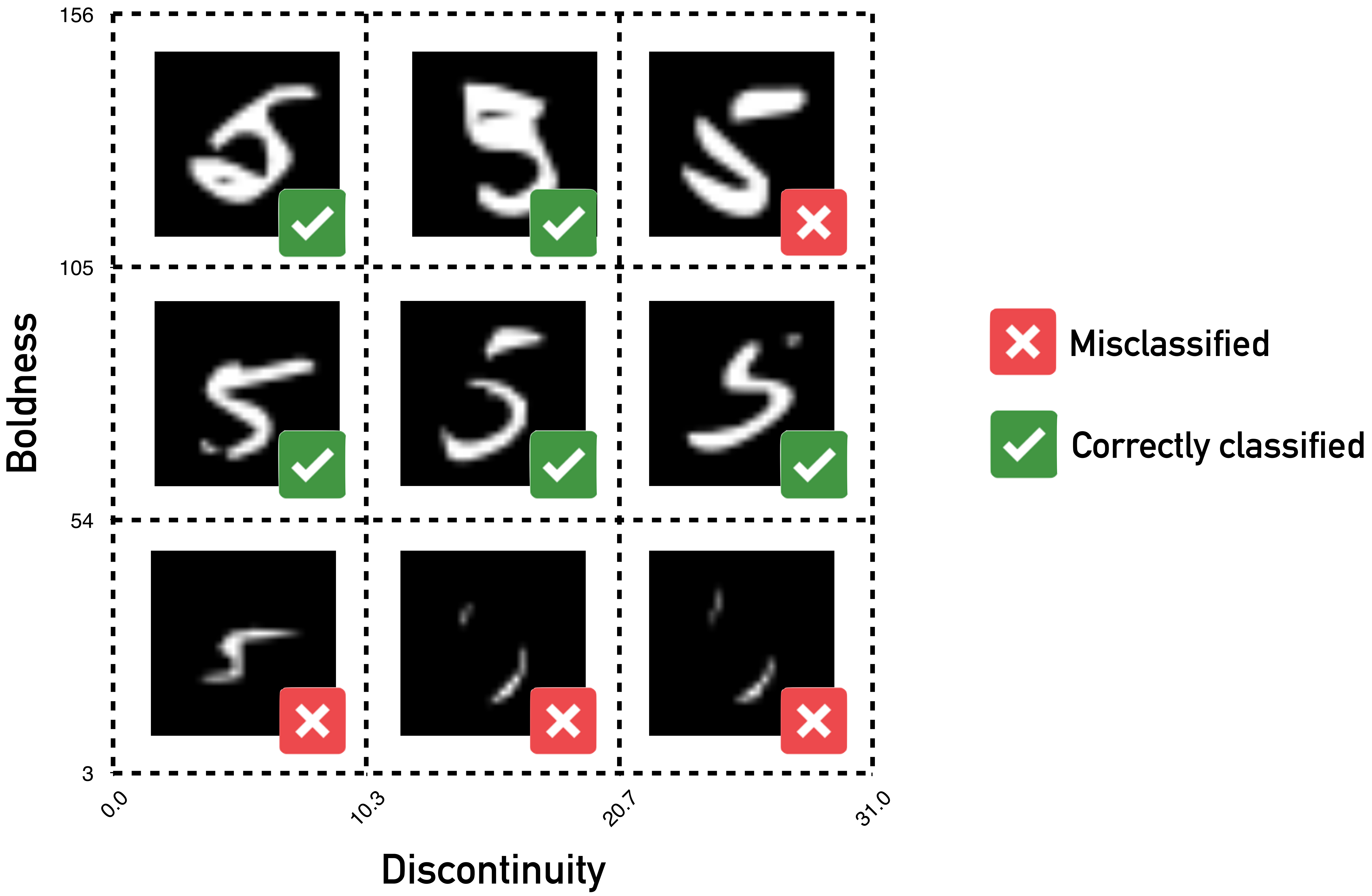


DEEPJANUS

FRONTIER OF BEHAVIOURS

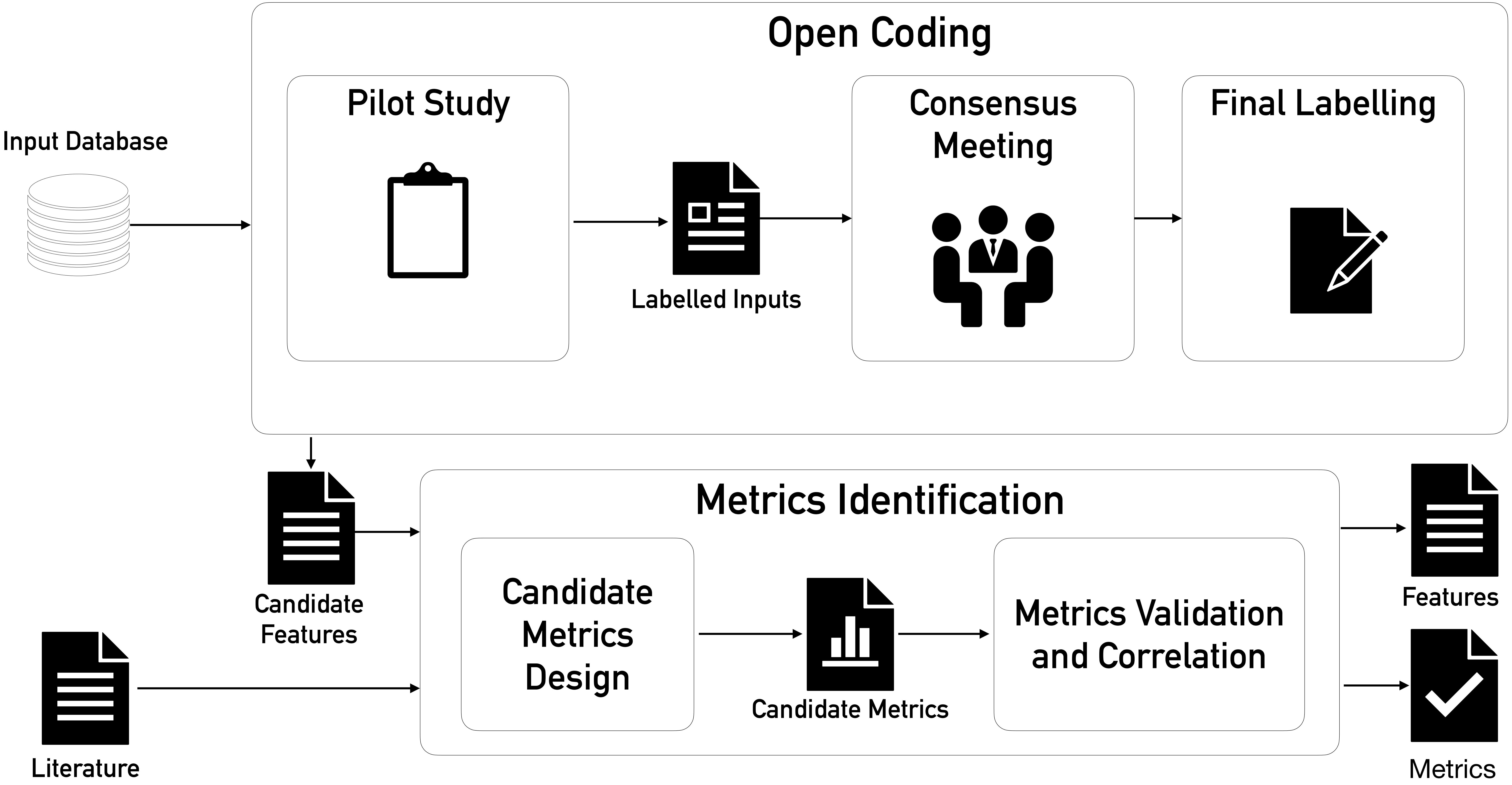
NEED FOR A HUMAN-INTERPRETABLE CHARACTERISATION OF THE MISBEHAVIOURS

FEATURE MAPS

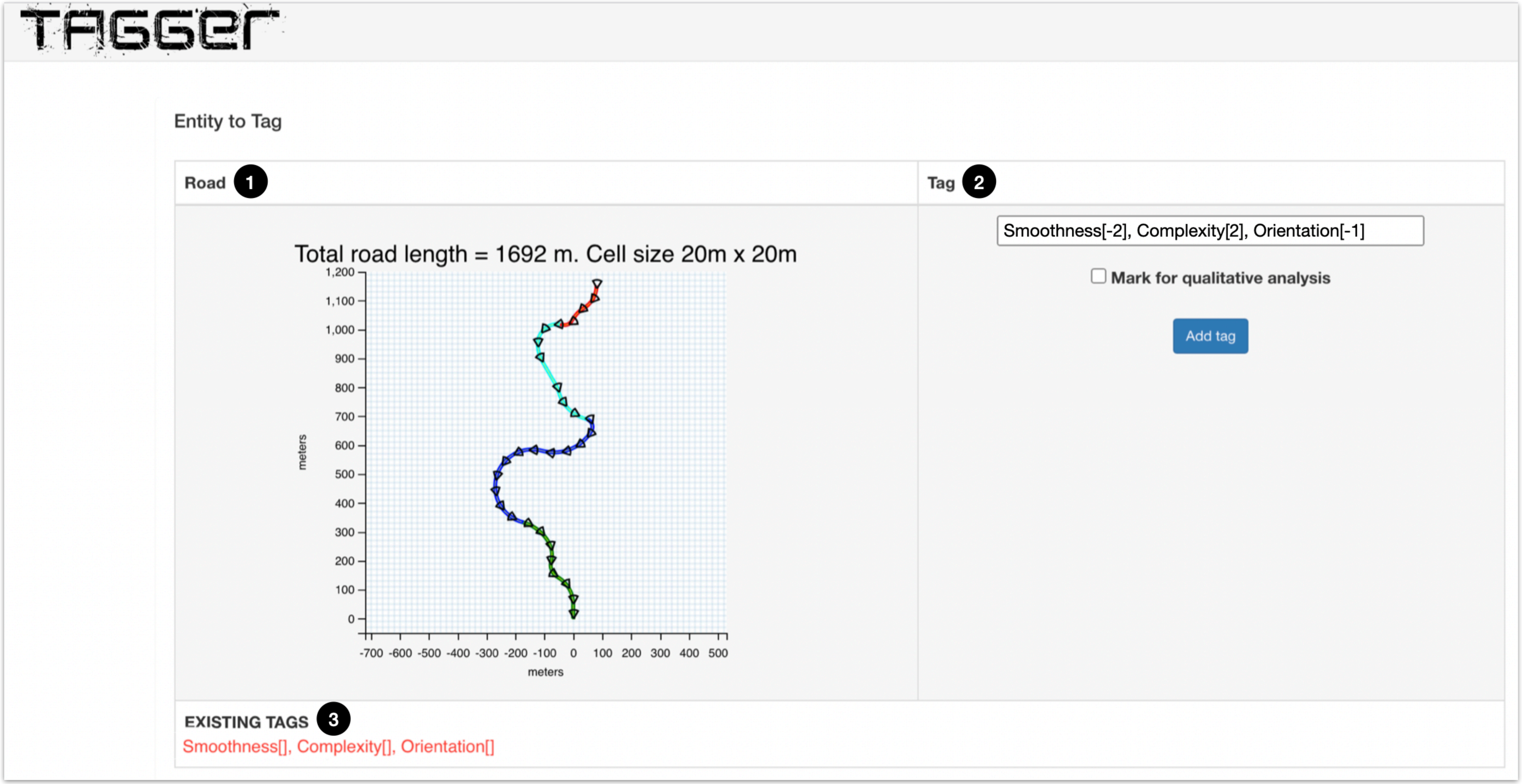


FEATURE SELECTION

FEATURE SELECTION METHODOLOGY



FEATURE SELECTION METHODOLOGY: OPEN CODING



Entity to Tag

Road 1

Tag 2

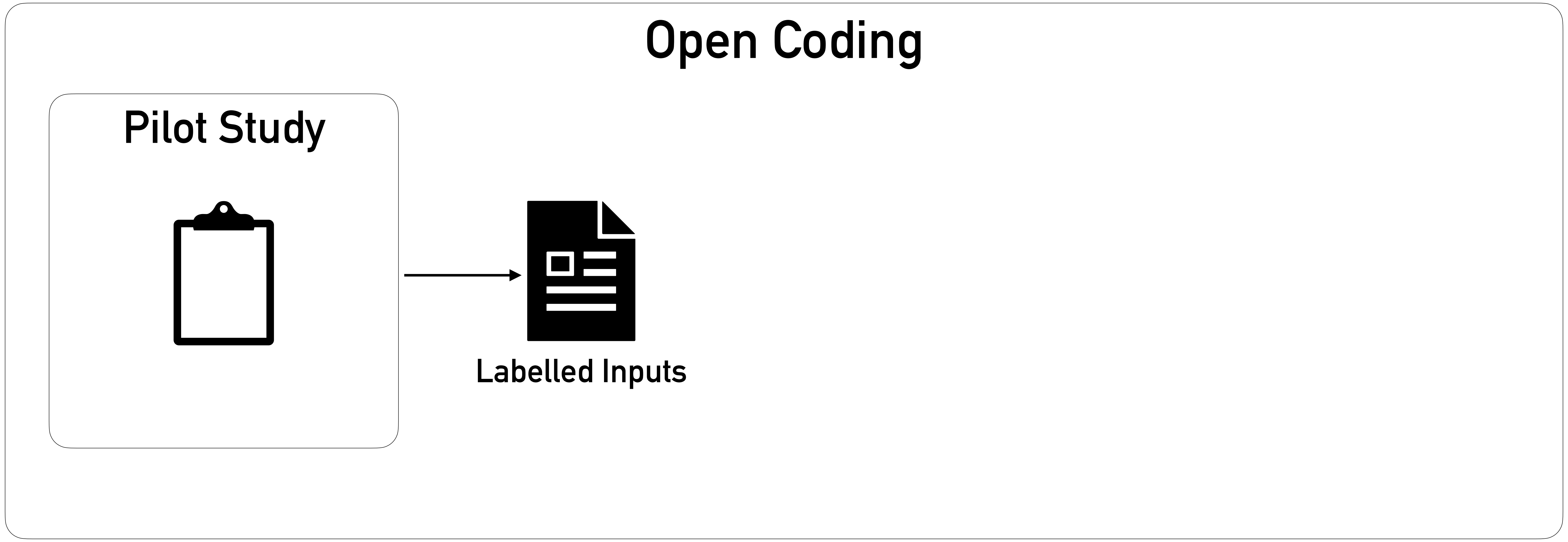
Smoothness[-2], Complexity[2], Orientation[-1]

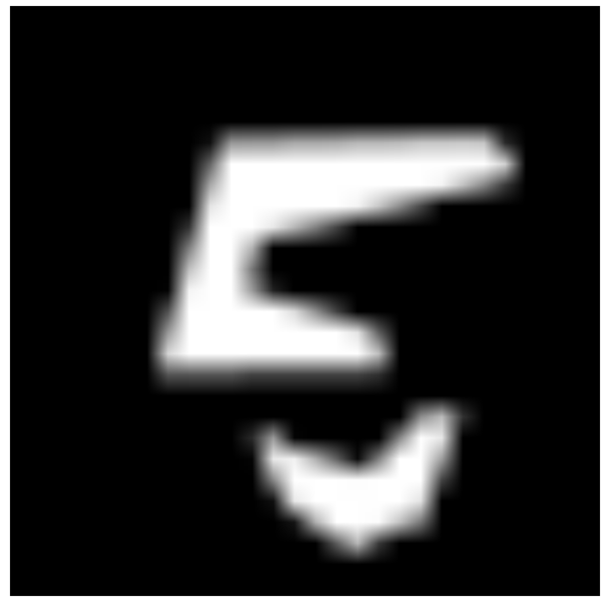
□ Mark for qualitative analysis

EXISTING TAGS 3

Smoothness[], Complexity[], Orientation[]

FEATURE SELECTION METHODOLOGY: OPEN CODING





Feature
Boldness
Smoothness
Discontinuity

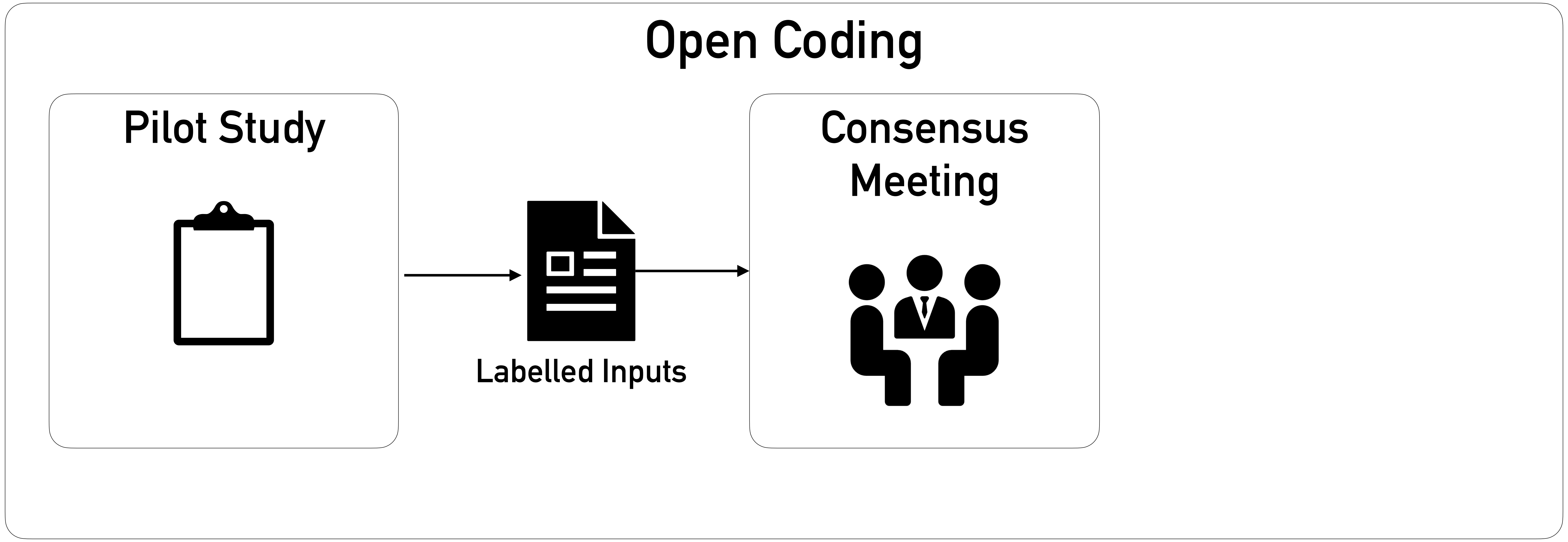
Assessor 1

Boldness:[0],
Smoothness:[-2],
Discontinuity:[1]

Assessor 2

Boldness:[2],
Smoothness:[-2],
Discontinuity:[1]

FEATURE SELECTION METHODOLOGY: OPEN CODING

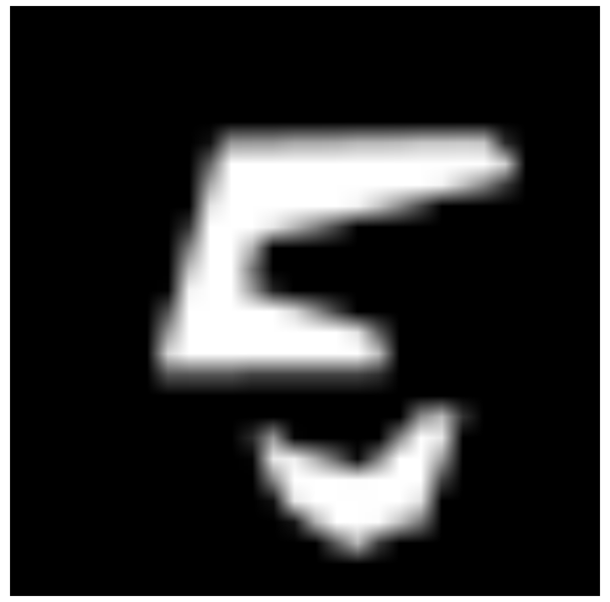


Open Coding

Pilot Study

Labelled Inputs

Consensus Meeting



Feature
Boldness
Smoothness
Discontinuity

Assessor 1

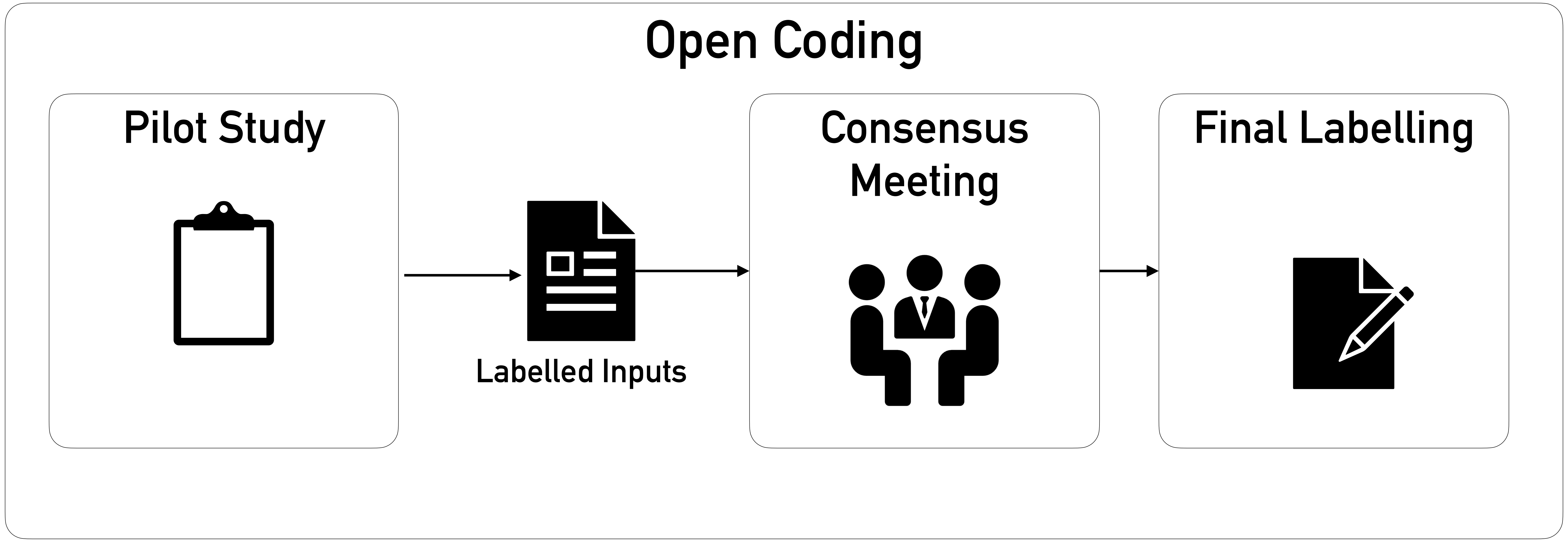
Boldness:[0], Smoothness:[-2], Discontinuity:[1]

Assessor 2

Boldness:[2], Smoothness:[-2], Discontinuity:[1]

Feature
Boldness
Smoothness
Discontinuity
Rotation

FEATURE SELECTION METHODOLOGY: OPEN CODING



Feature

Boldness
Smoothness
Discontinuity

Assessor 1

Boldness:[0],
Smoothness:[-2],
Discontinuity:[1]

Assessor 2

Boldness:[2],
Smoothness:[-2],
Discontinuity:[1]

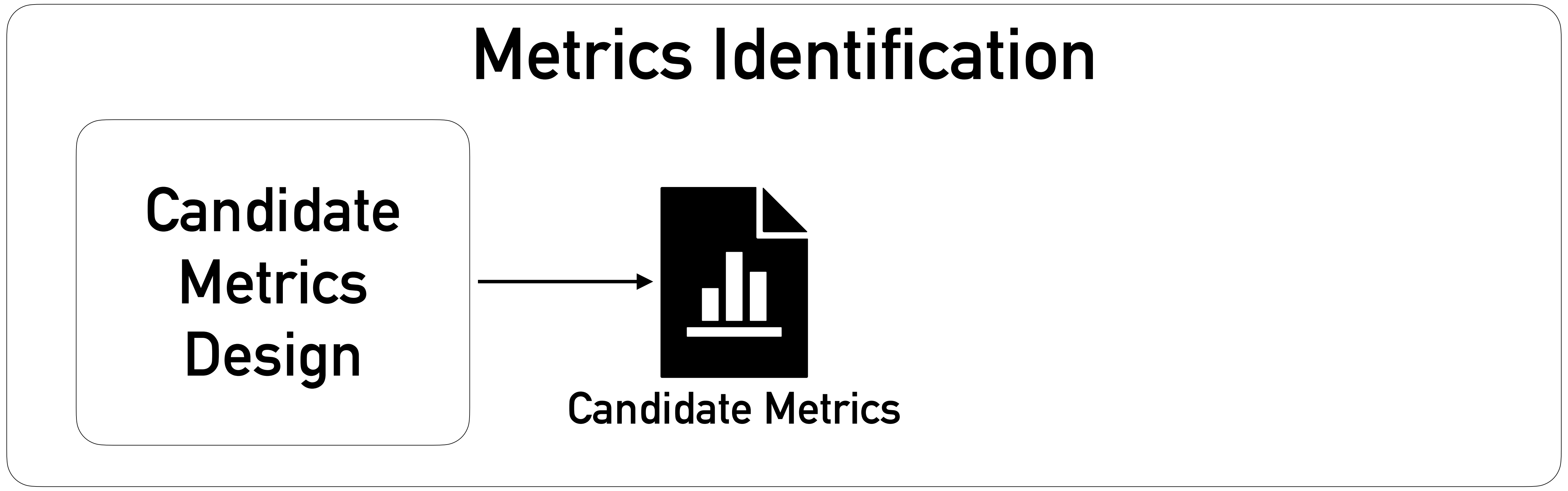
Feature

Boldness
Smoothness
Discontinuity
Rotation

Final Label

Boldness:[2],
Smoothness:[-2],
Discontinuity:[1],
Rotation:[0]

FEATURE SELECTION METHODOLOGY: METRICS IDENTIFICATION

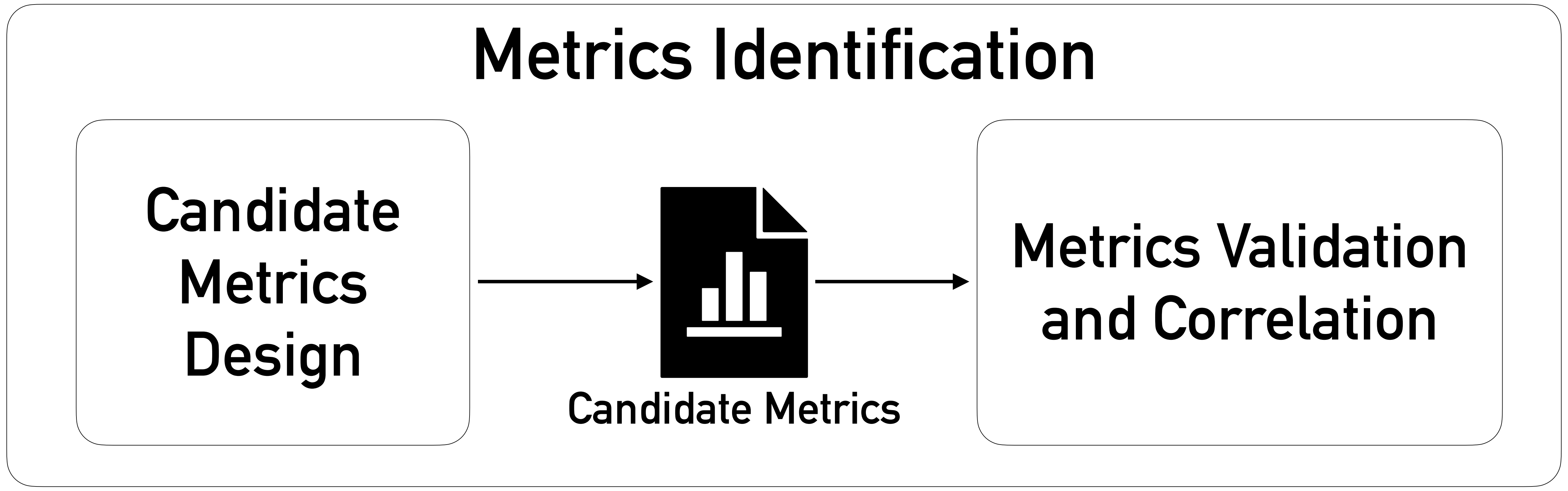

Metrics Identification

Candidate Metrics Design

Candidate Metrics

Case study	Feature	Metric
MNIST	Boldness	Lum
	Smoothness	AvgAng
	Discontinuity	Mov
	Rotation	Or

FEATURE SELECTION METHODOLOGY: METRICS IDENTIFICATION



Metrics Identification

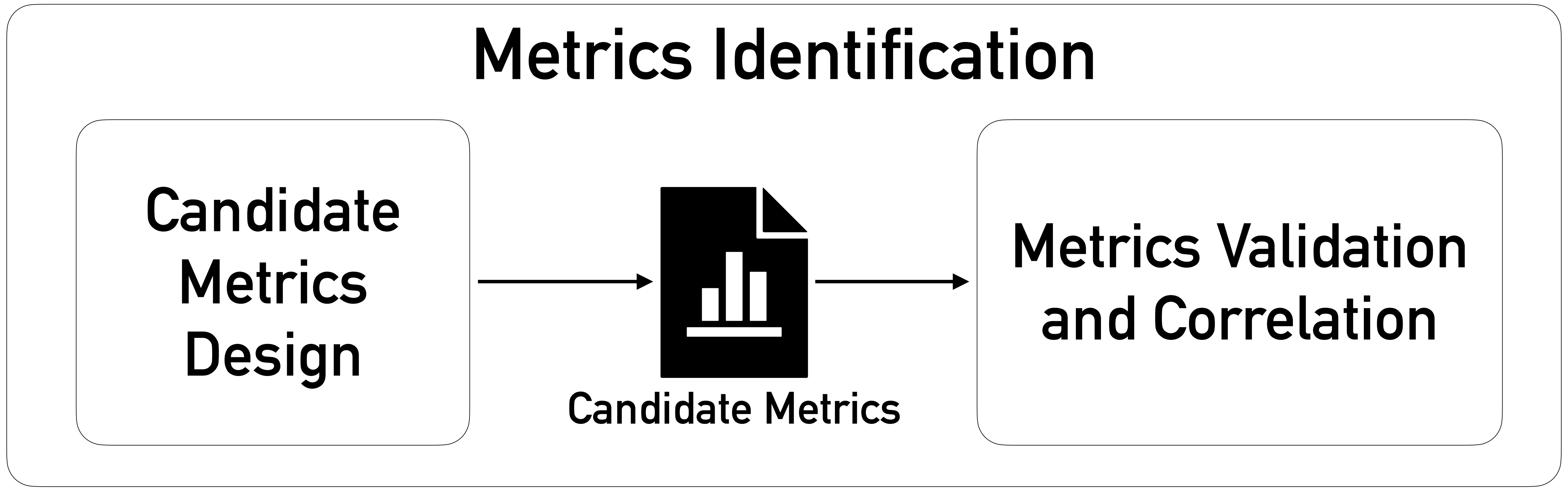
Candidate Metrics Design

Candidate Metrics

Metrics Validation and Correlation

Case study	Feature	Metric	Correlation	P-value
MNIST	Boldness	Lum	0.67	< 0.002
	Smoothness	AvgAng	0.05	0.241
	Discontinuity	Mov	0.9	< 0.002
	Rotation	Or	0.43	< 0.002

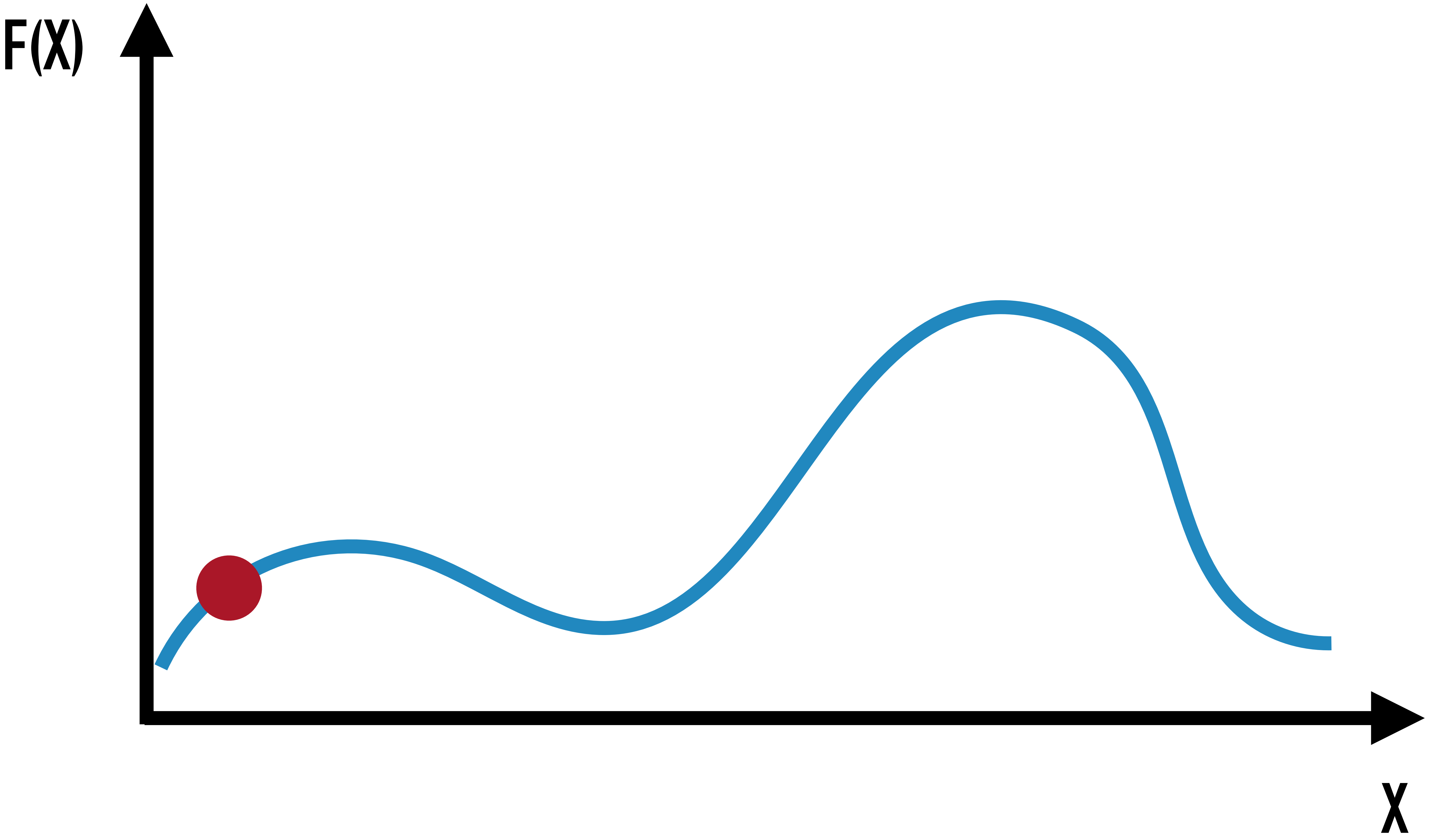
FEATURE SELECTION METHODOLOGY: METRICS IDENTIFICATION



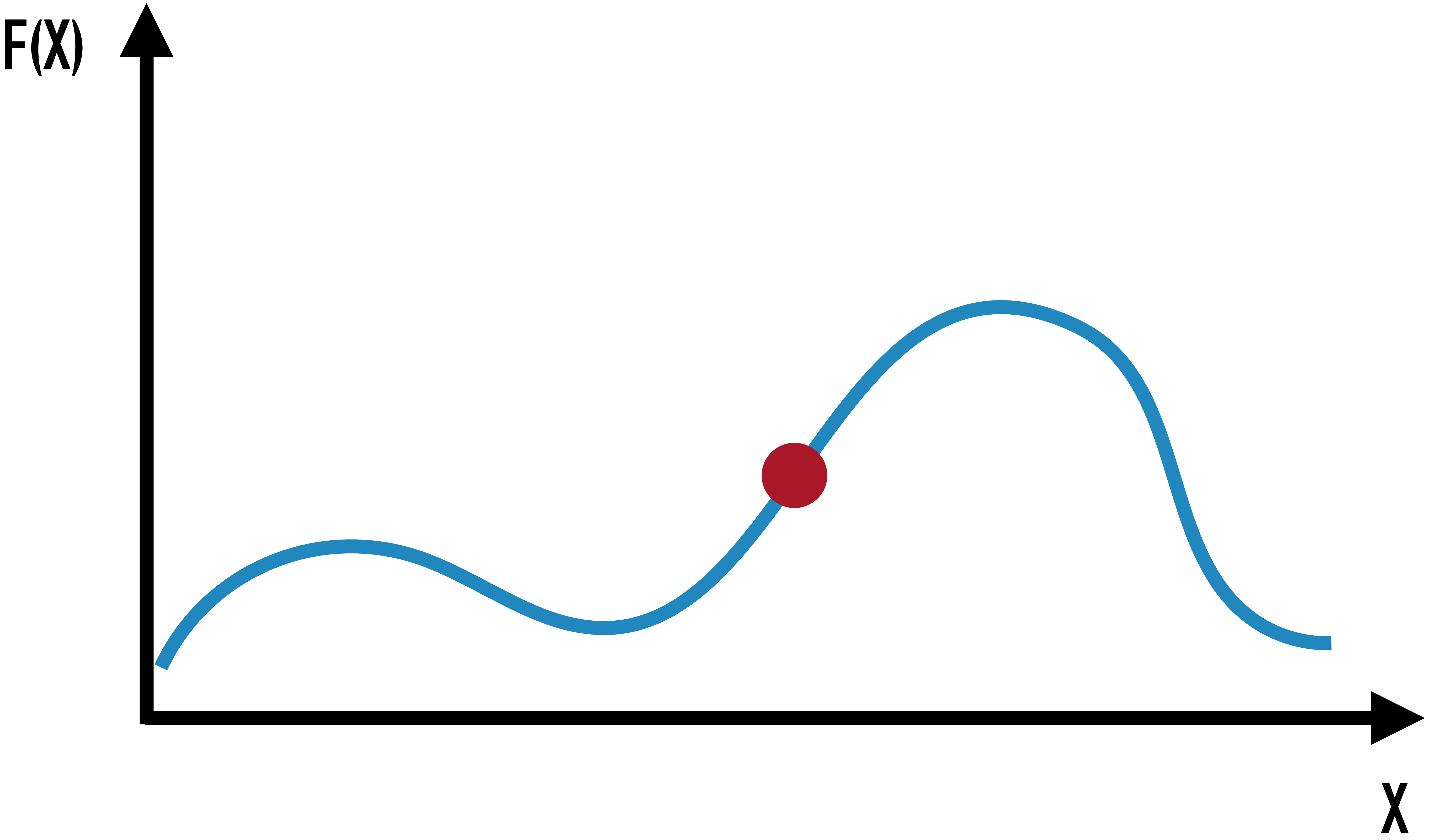
Case study	Feature	Metric	Correlation	P-value
MNIST	Boldness	Lum	0.67	< 0.002
	~~Smoothness~~	~~AvgAng~~	~~0.05~~	~~0.241~~
	Discontinuity	Mov	0.9	< 0.002
	Rotation	Or	0.43	< 0.002

AUTOMATED TEST GENERATION

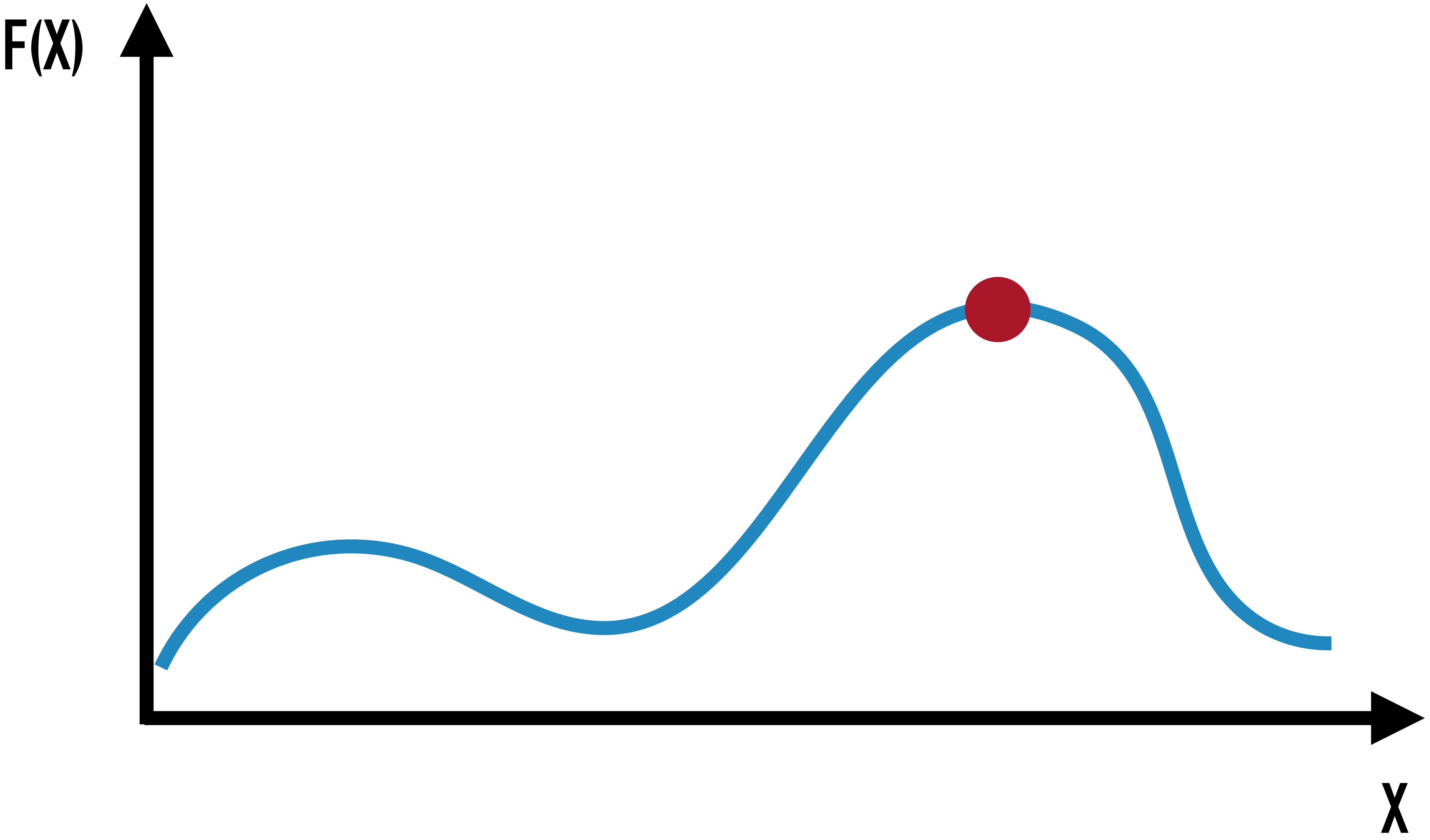
EVOLUTIONARY ALGORITHMS



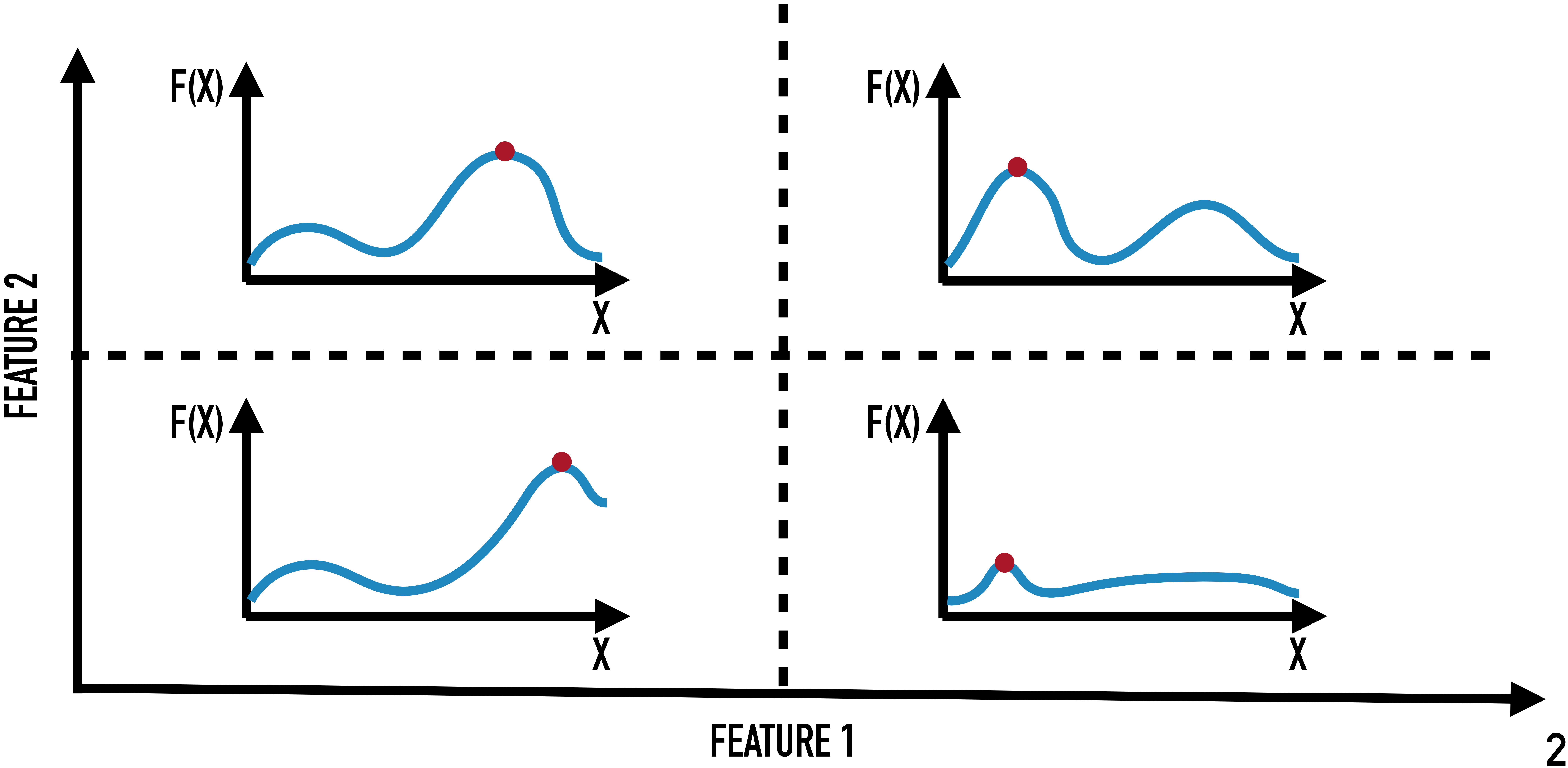
EVOLUTIONARY ALGORITHMS



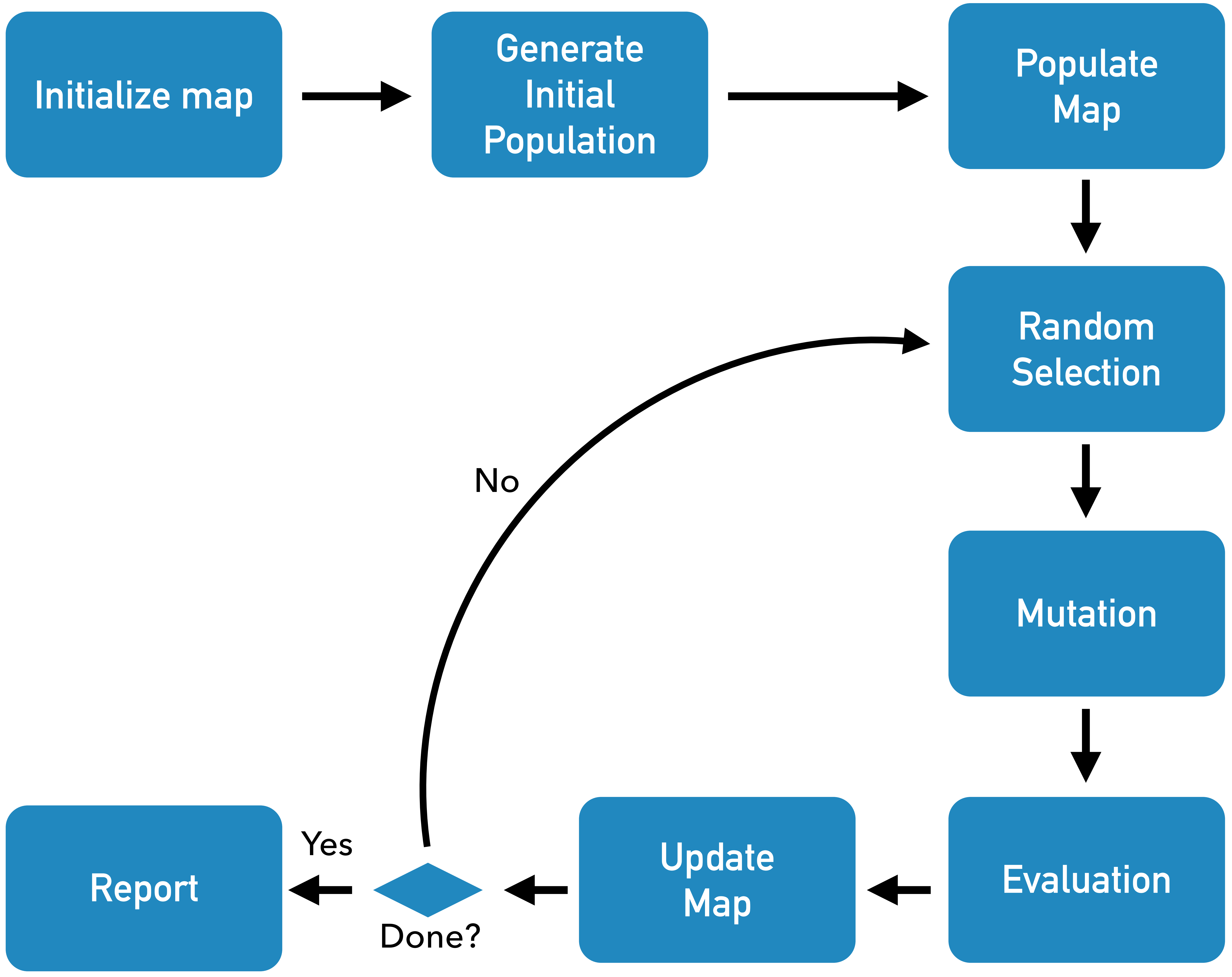
EVOLUTIONARY ALGORITHMS



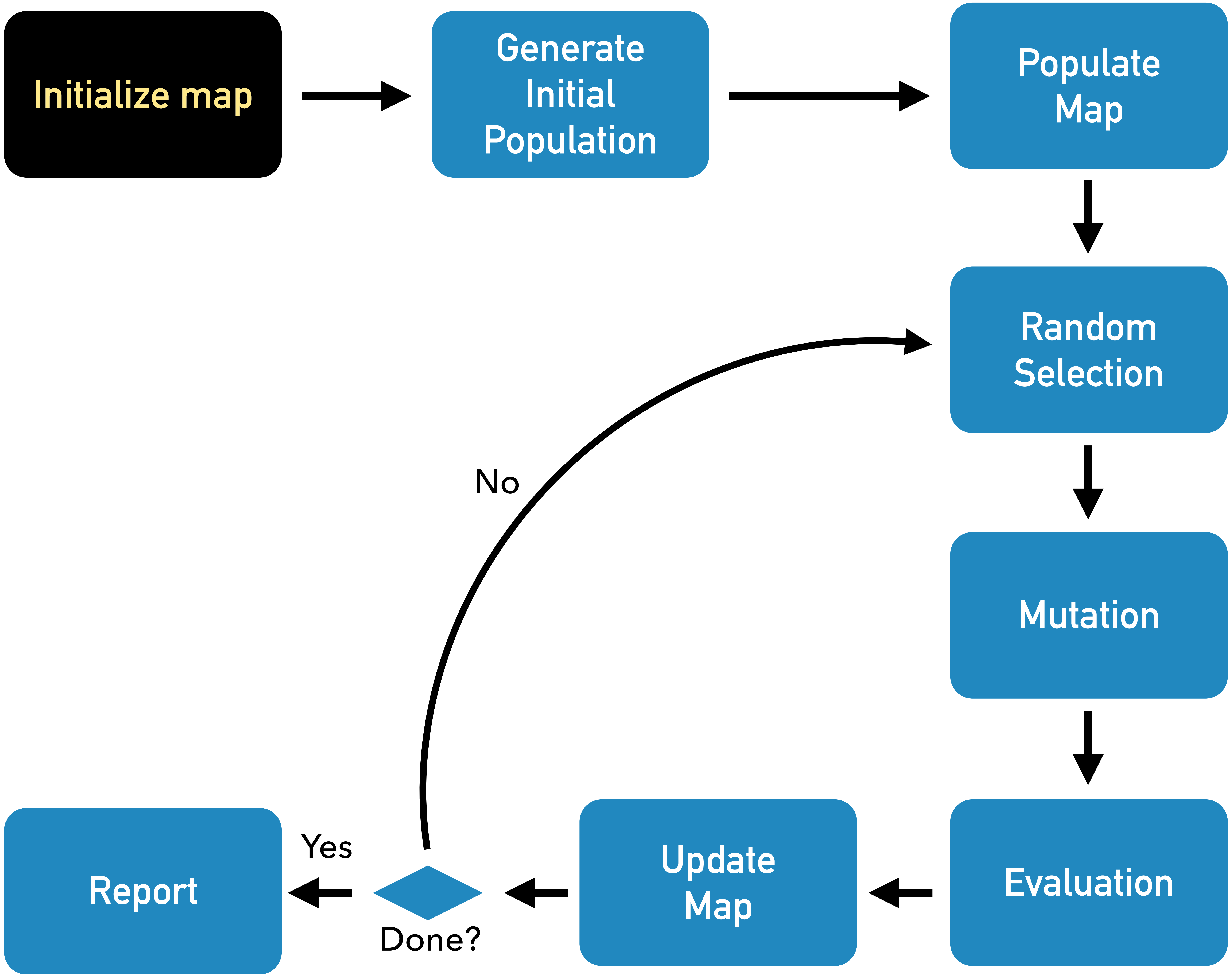
ILLUMINATION SEARCH



DEEPHYPERION



DEEPHYPERION



Initialize map

Generate Initial Population

Populate Map

Random Selection

Mutation

Evaluation

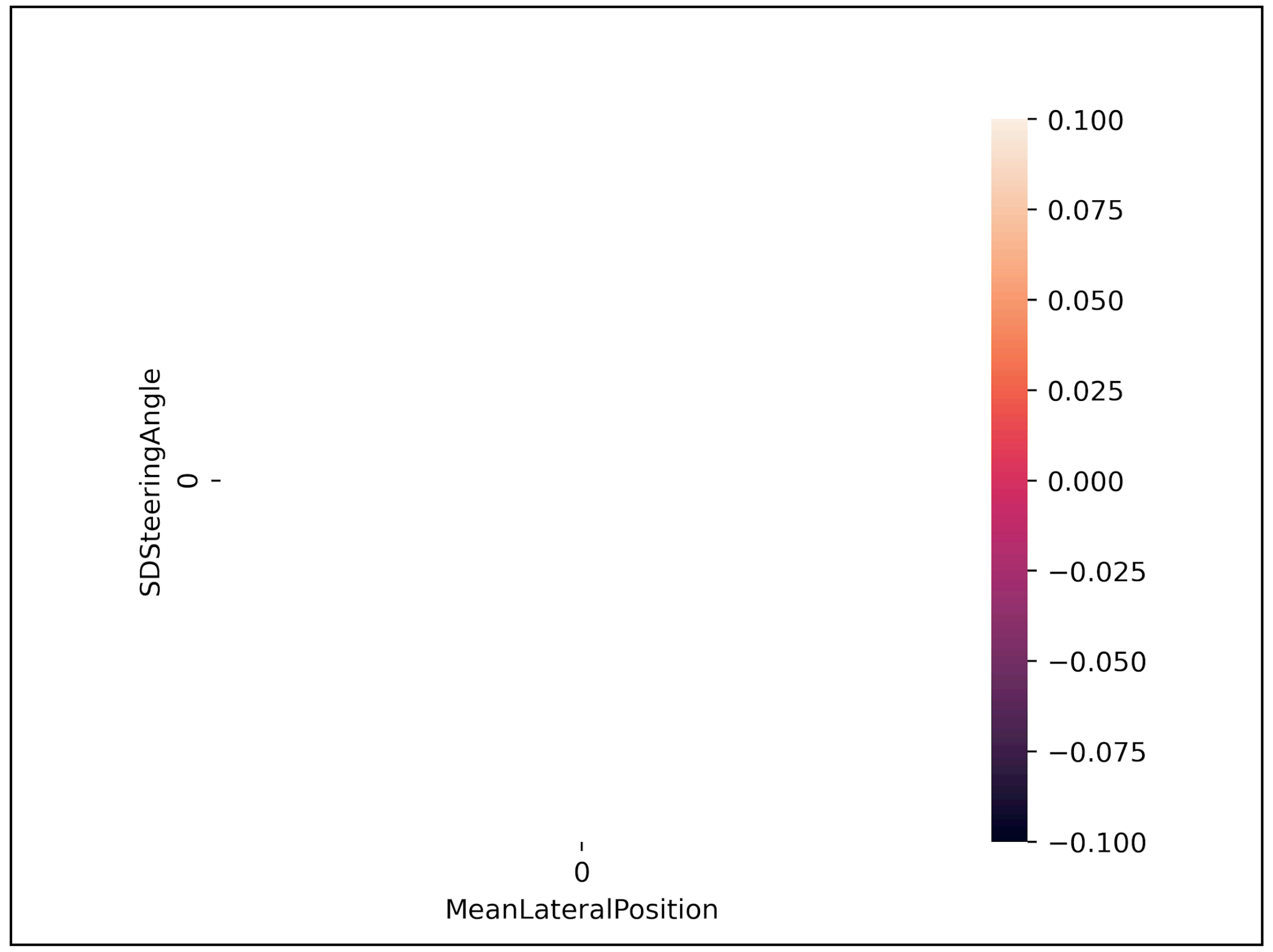
Update Map

Done?

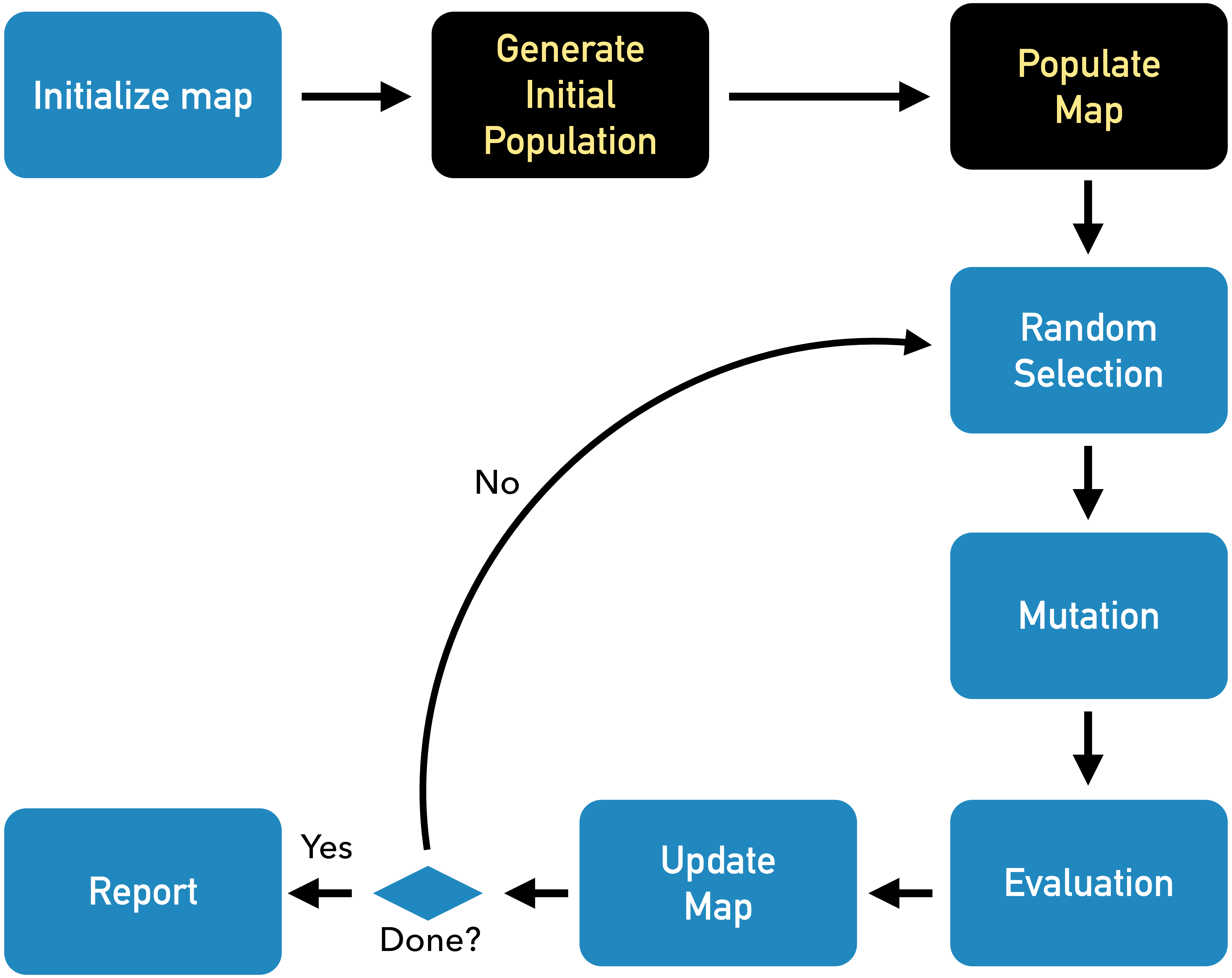
No

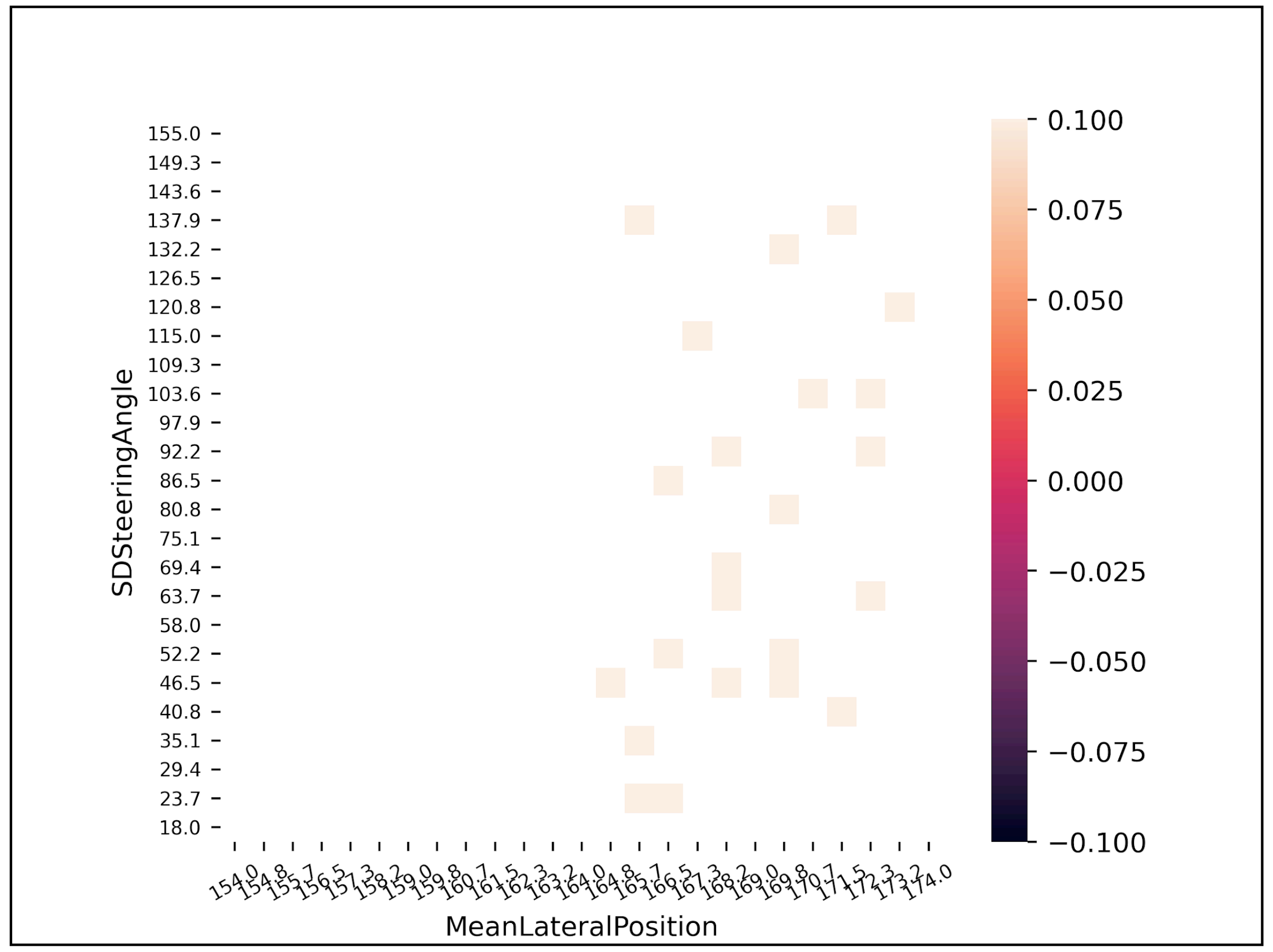
Yes

Report

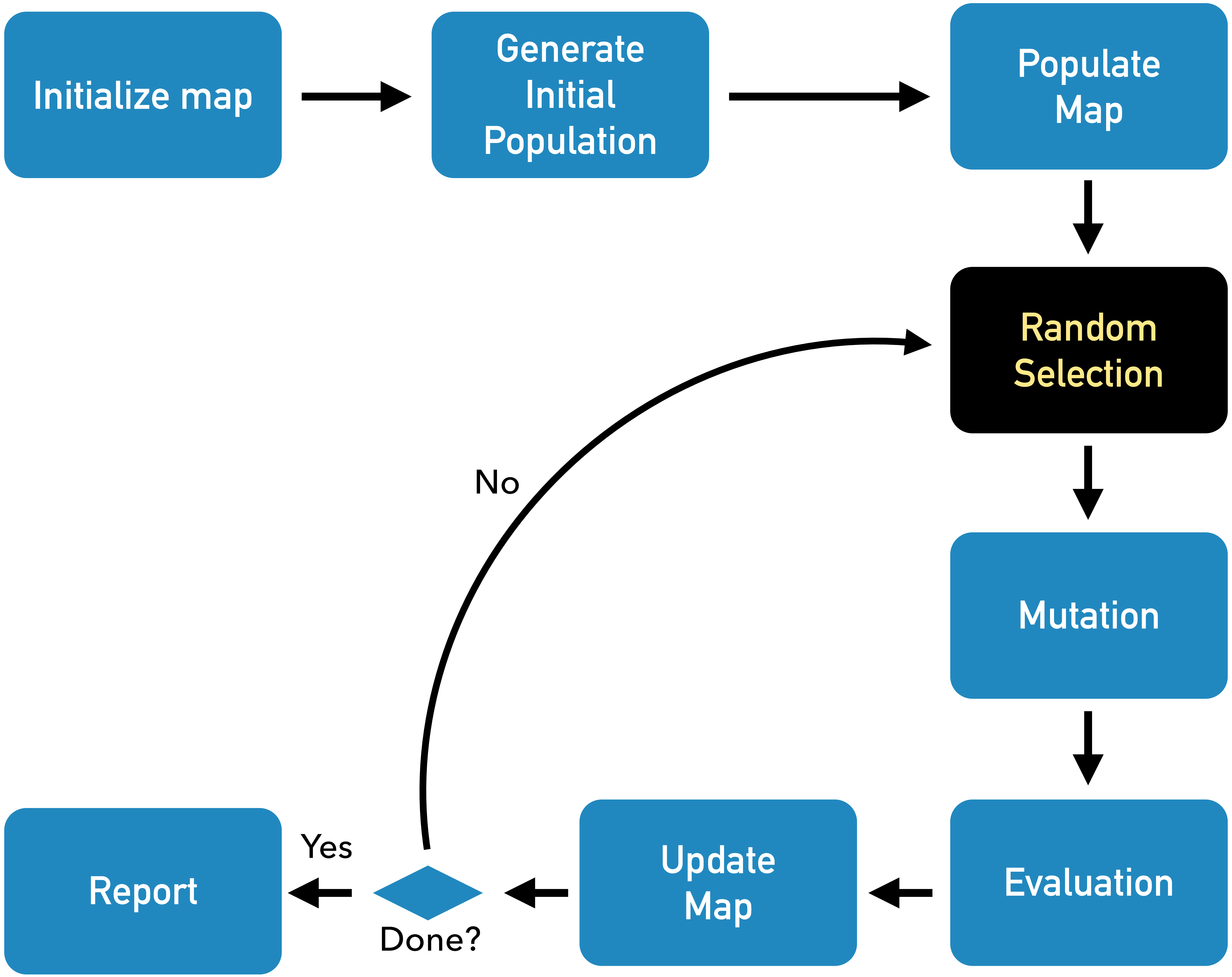


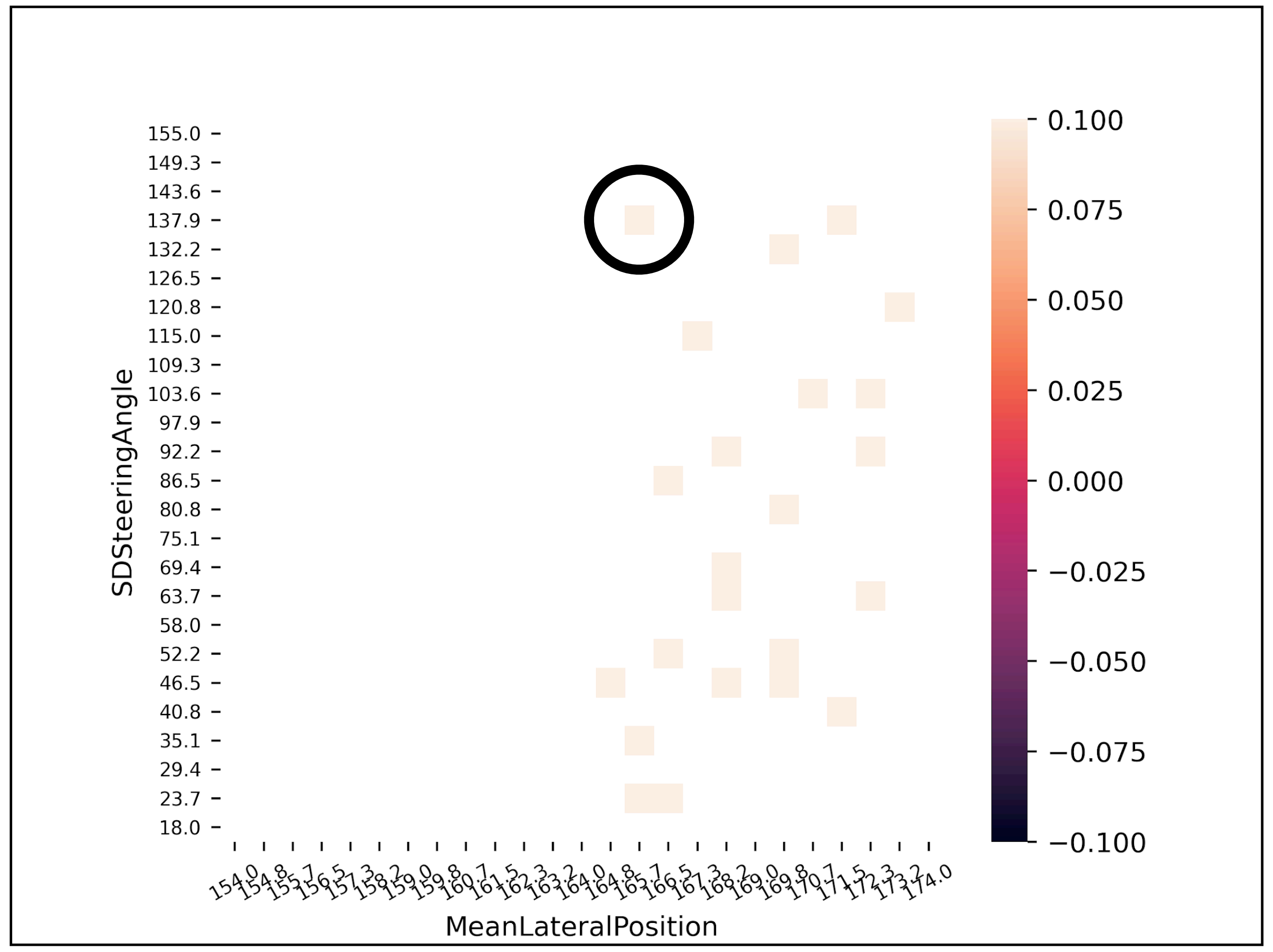
DEEPHYPERION



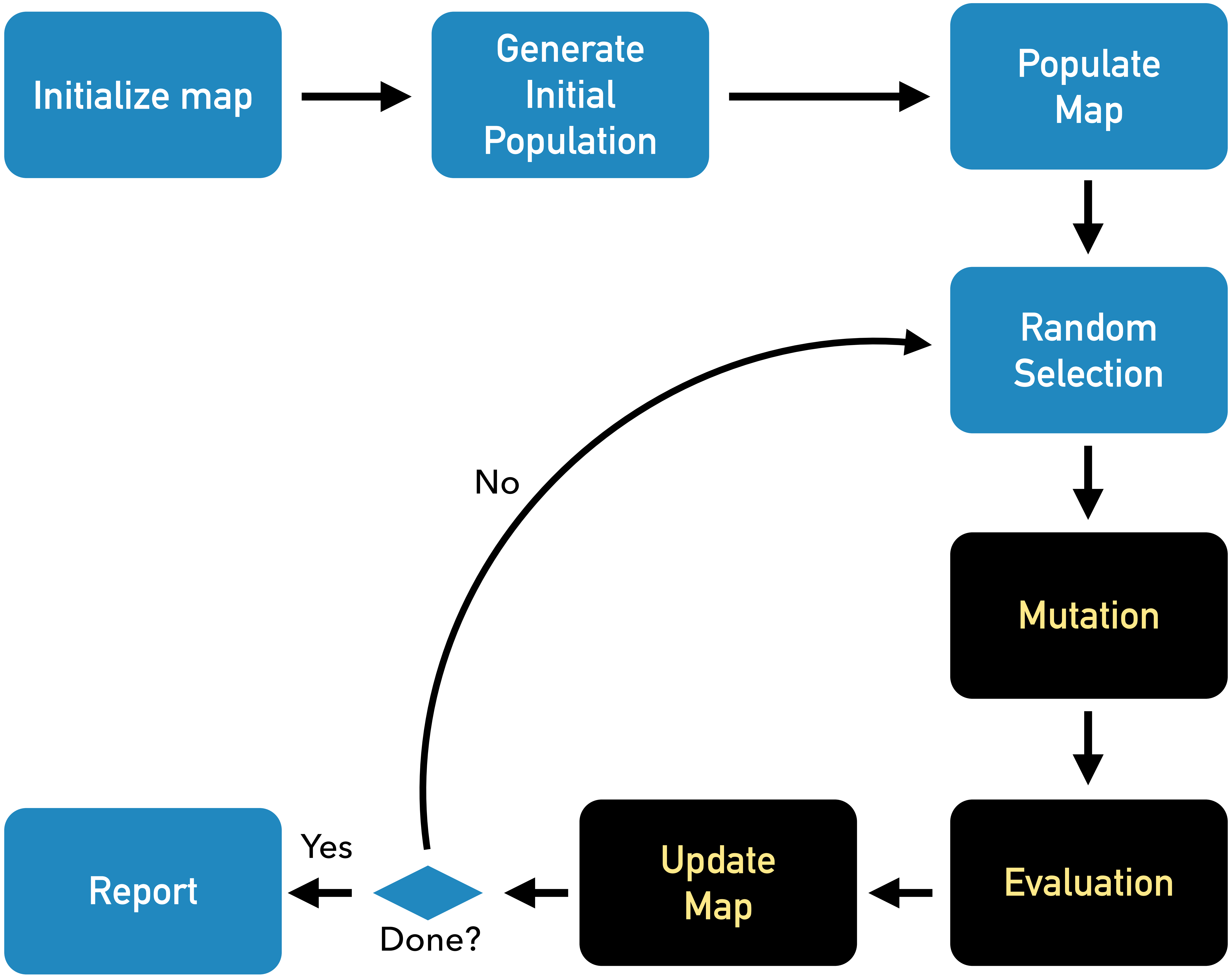


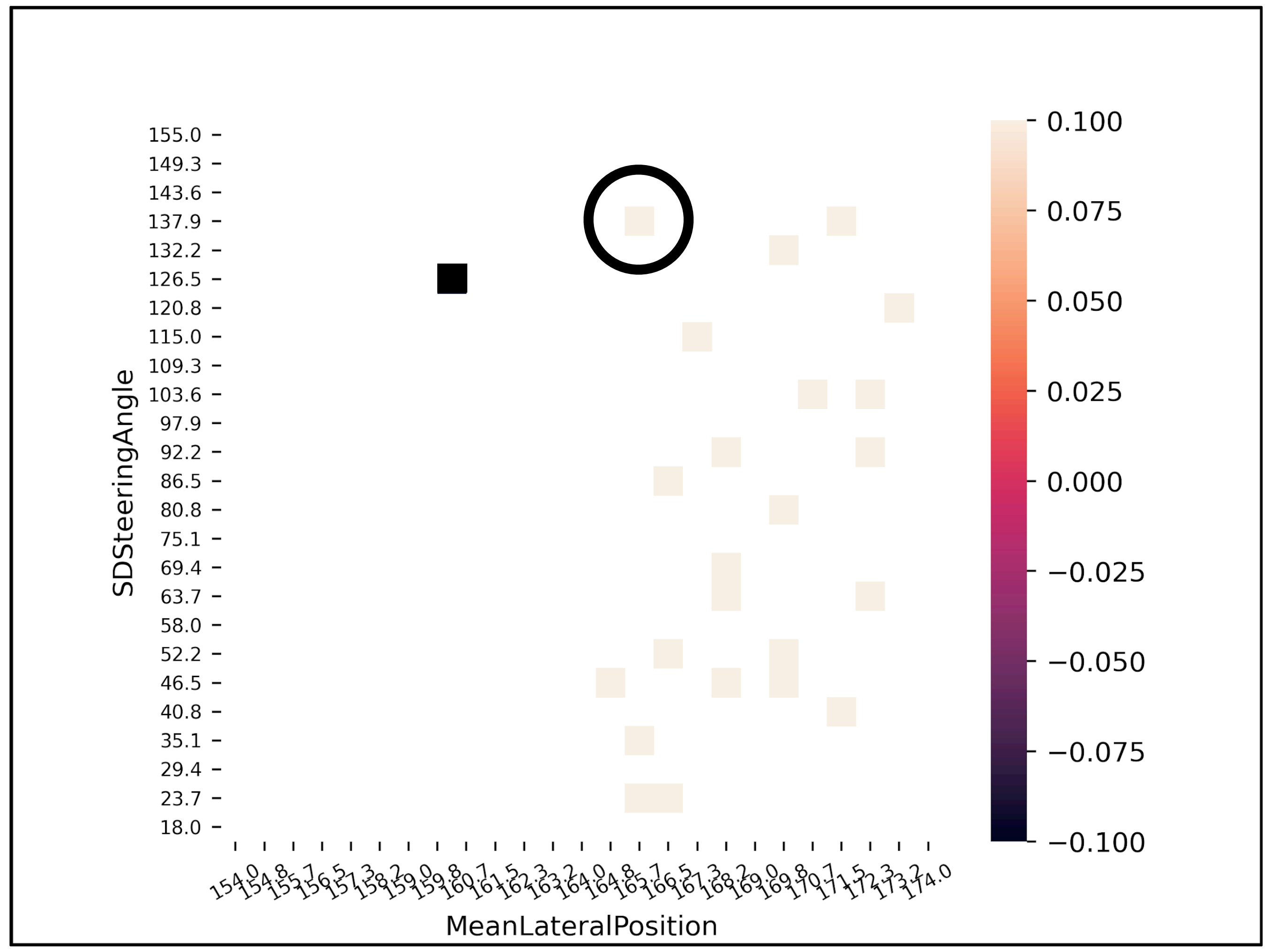
DEEPHYPERION



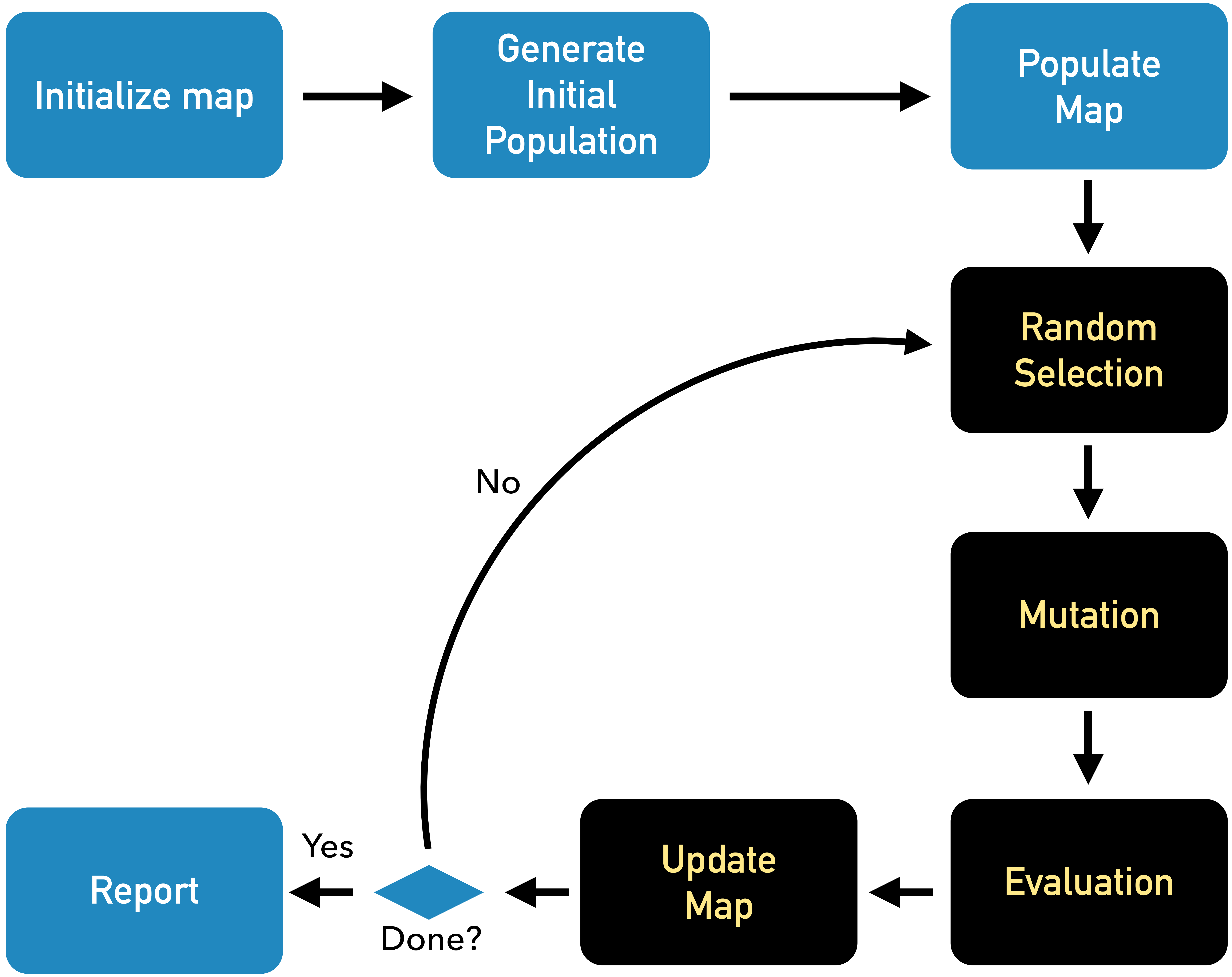


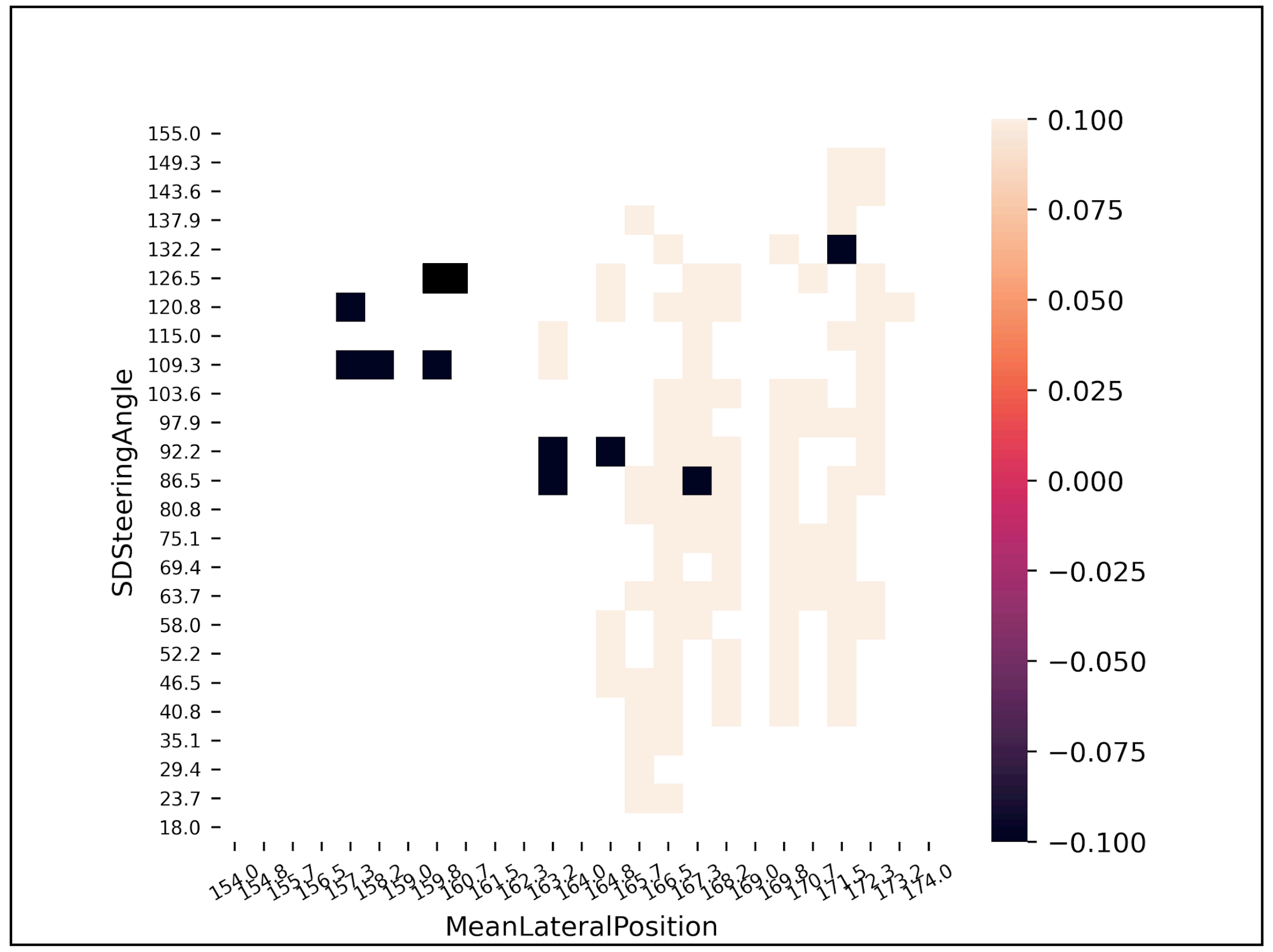
DEEPHYPERION



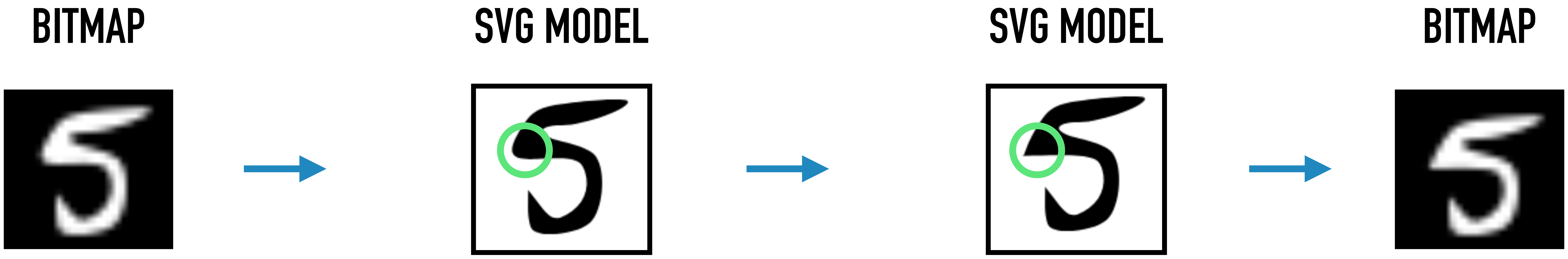


DEEPHYPERION





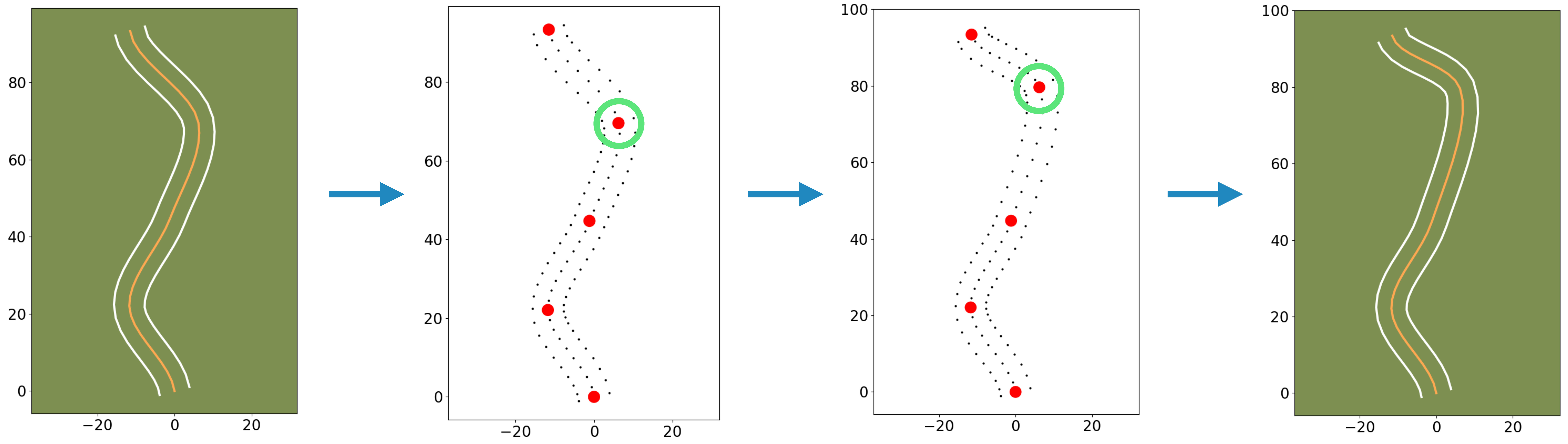
MODEL-BASED INPUT PERTURBATIONS - DIGITS



start_point = (9.0, 20.85)
BezierSegment(
c1 = (9.0, 20.22),
c2 = (10.22, 17.30),
end_point = (11.70, 14.38)
)

start_point = (9.0, 20.85)
BezierSegment(
c1 = (9.0, 20.22),
c2 = (**8.10**, 17.30),
end_point = (11.70, 14.38)
)

MODEL-BASED INPUT PERTURBATIONS – ROADS



ROAD

CATMULL-ROM MODEL

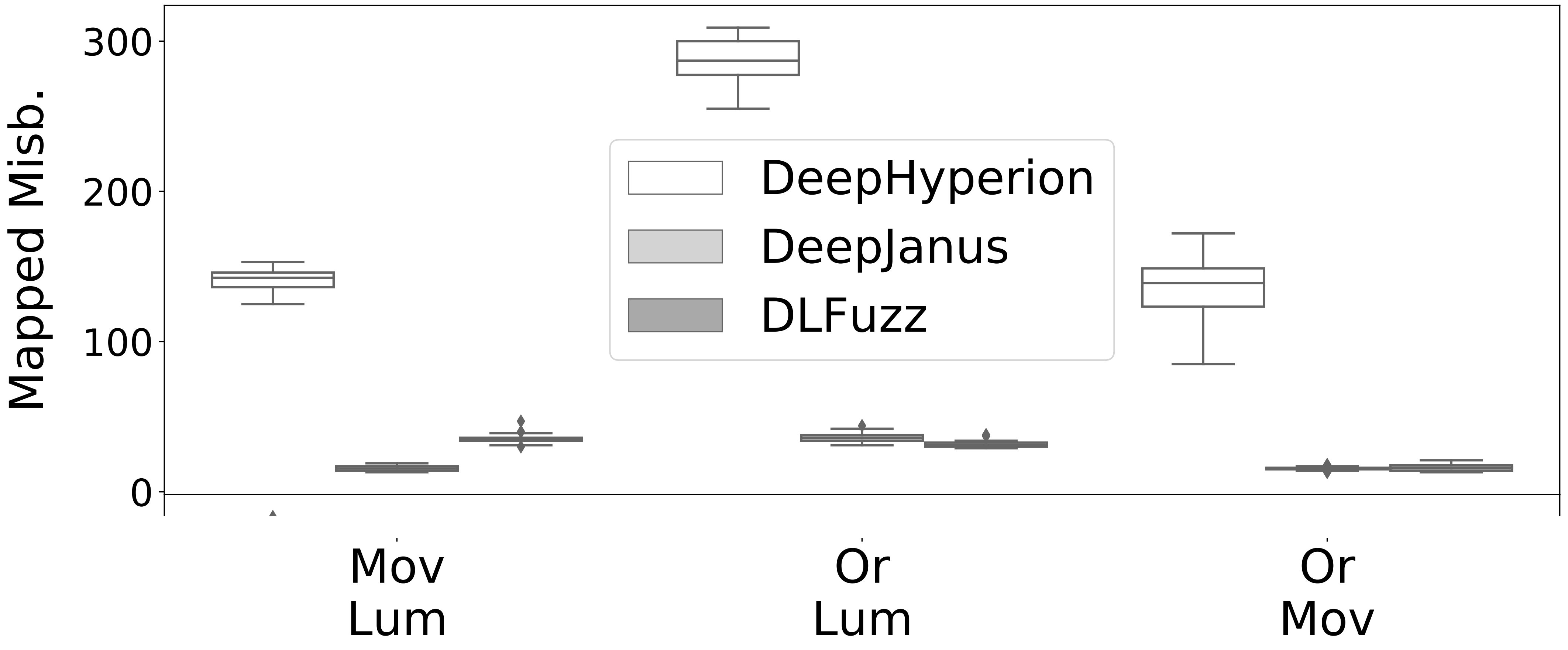
CATMULL-ROM MODEL

ROAD

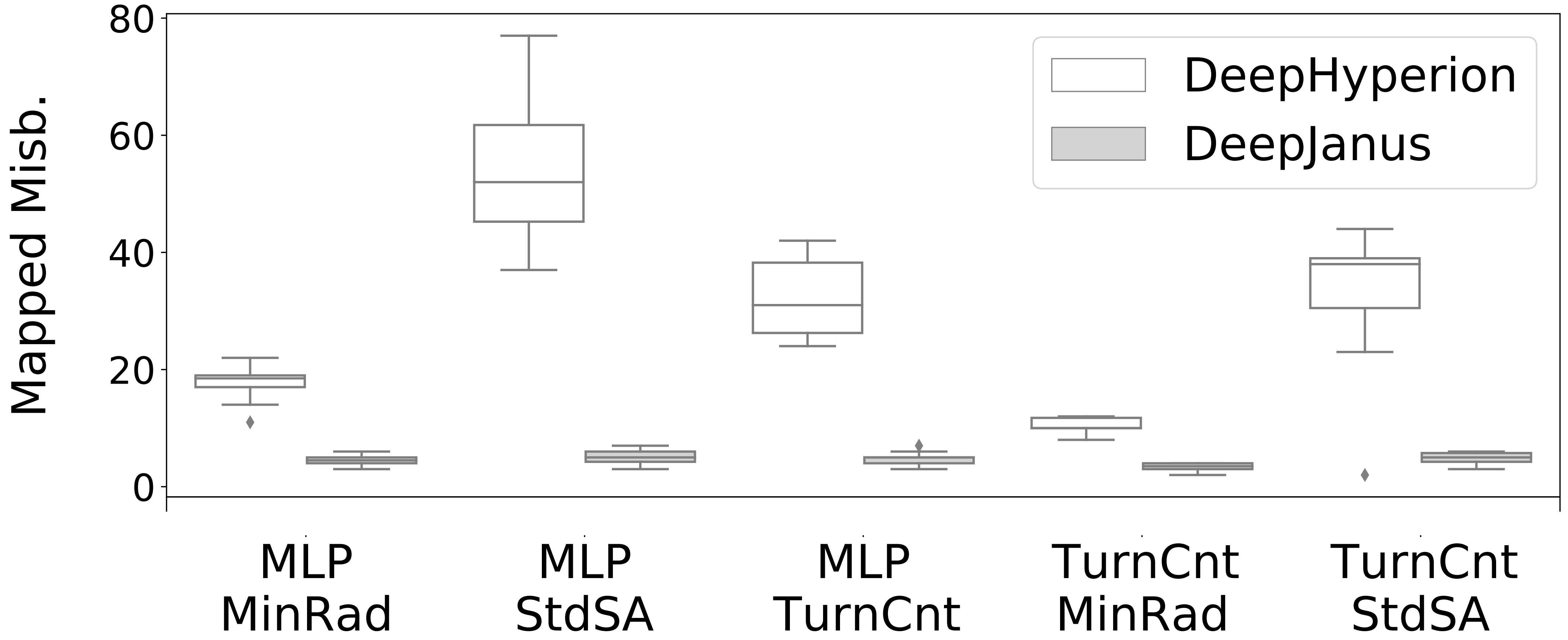
MISBEHAVIOUR DETECTION

MISBEHAVIOUR DETECTION

MNIST



BEAMNG



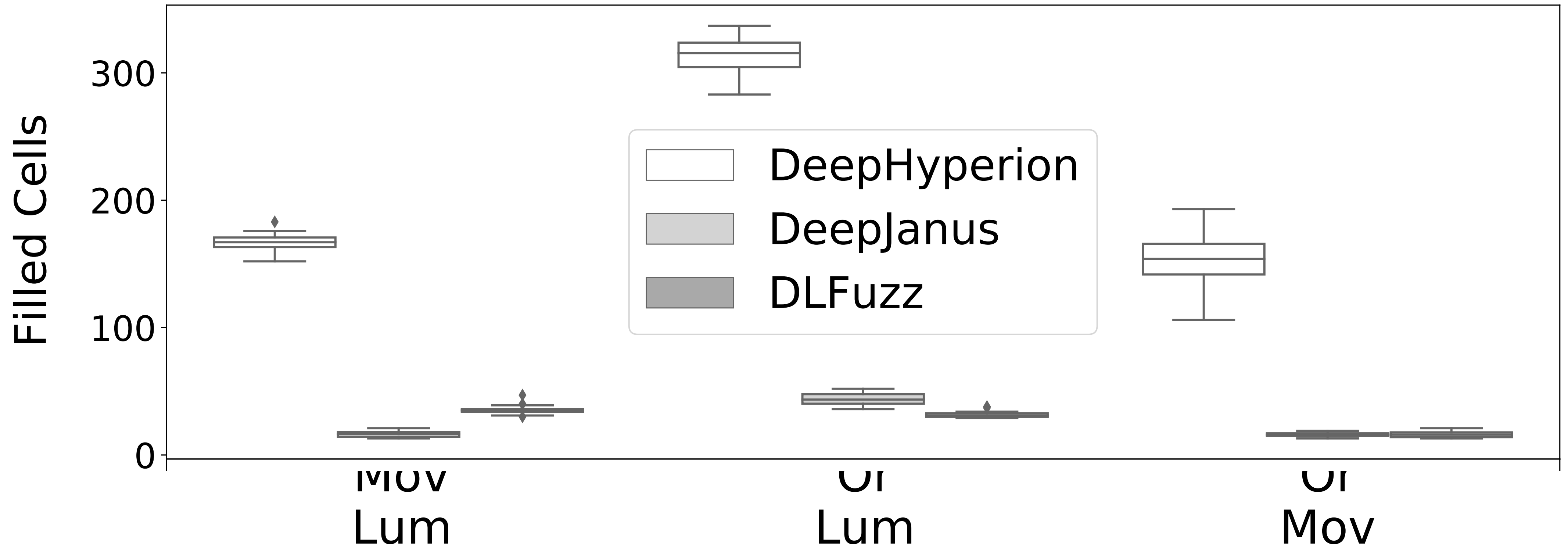
UP TO 10X

MORE MISBEHAVIOURS

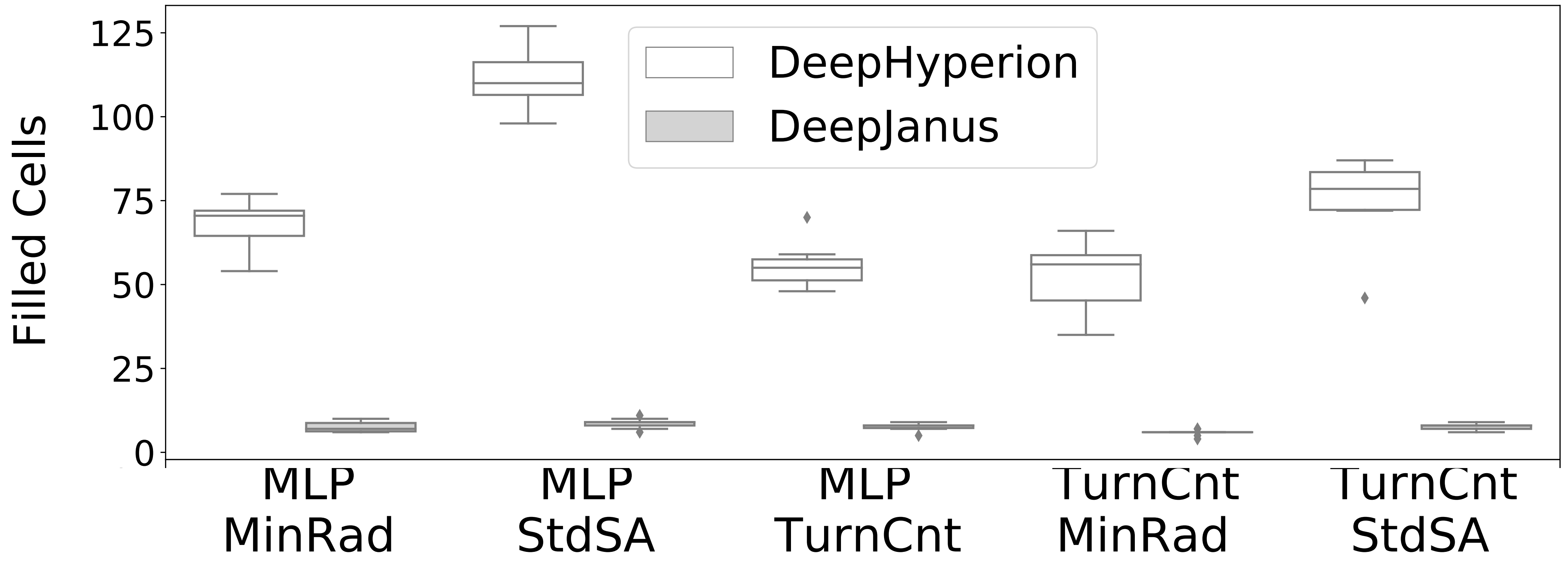
EXPLORATION EFFECTIVENESS

EXPLORATION EFFECTIVENESS

MNIST



BEAMNG



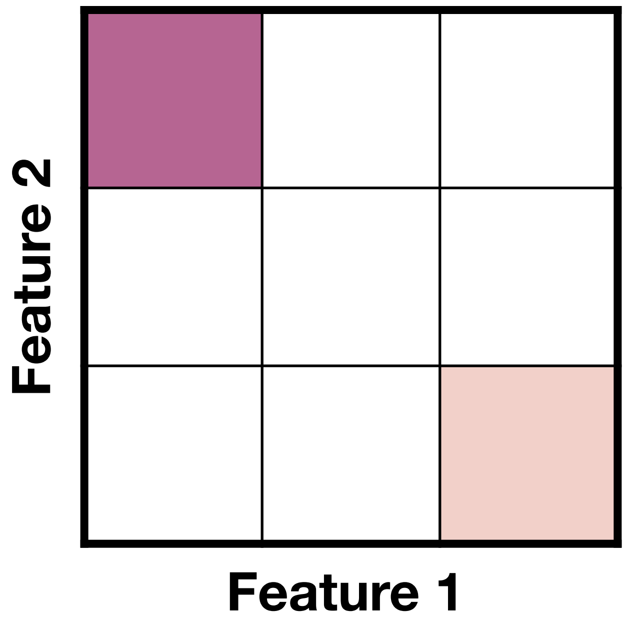
UP TO 8X

MAP EXPLORATION

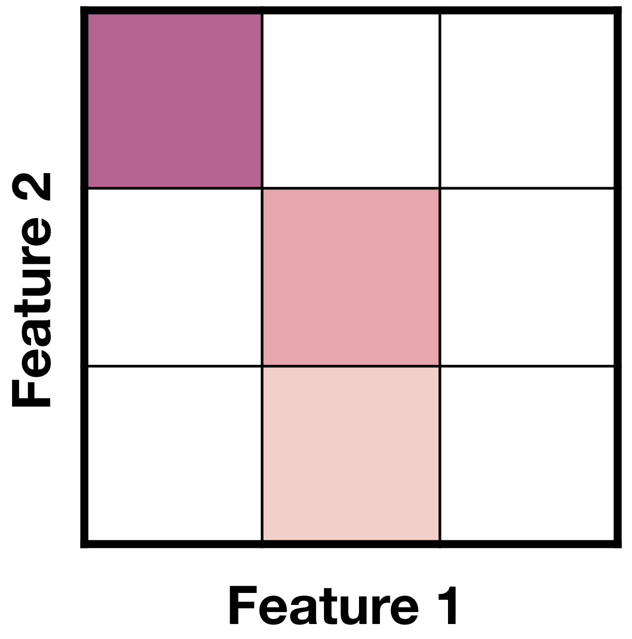
MISBEHAVIOUR CHARACTERISATION

MISBEHAVIOUR CHARACTERISATION

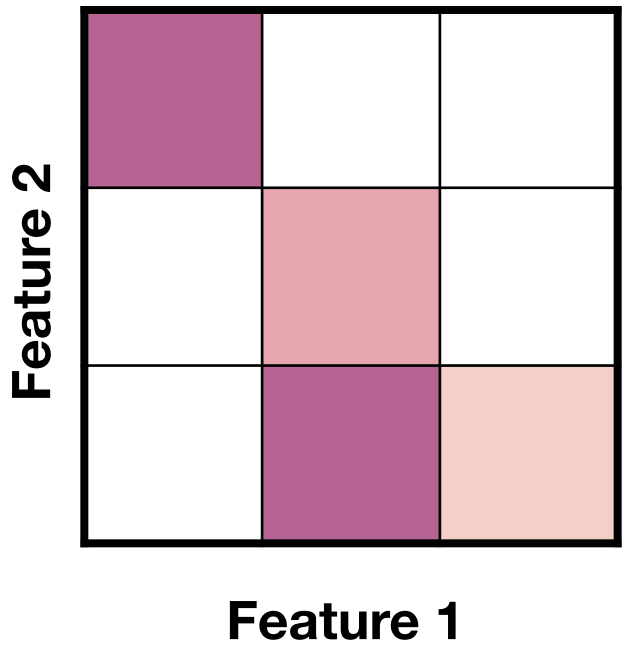
Misbehaviour Probability Map (Run 1)



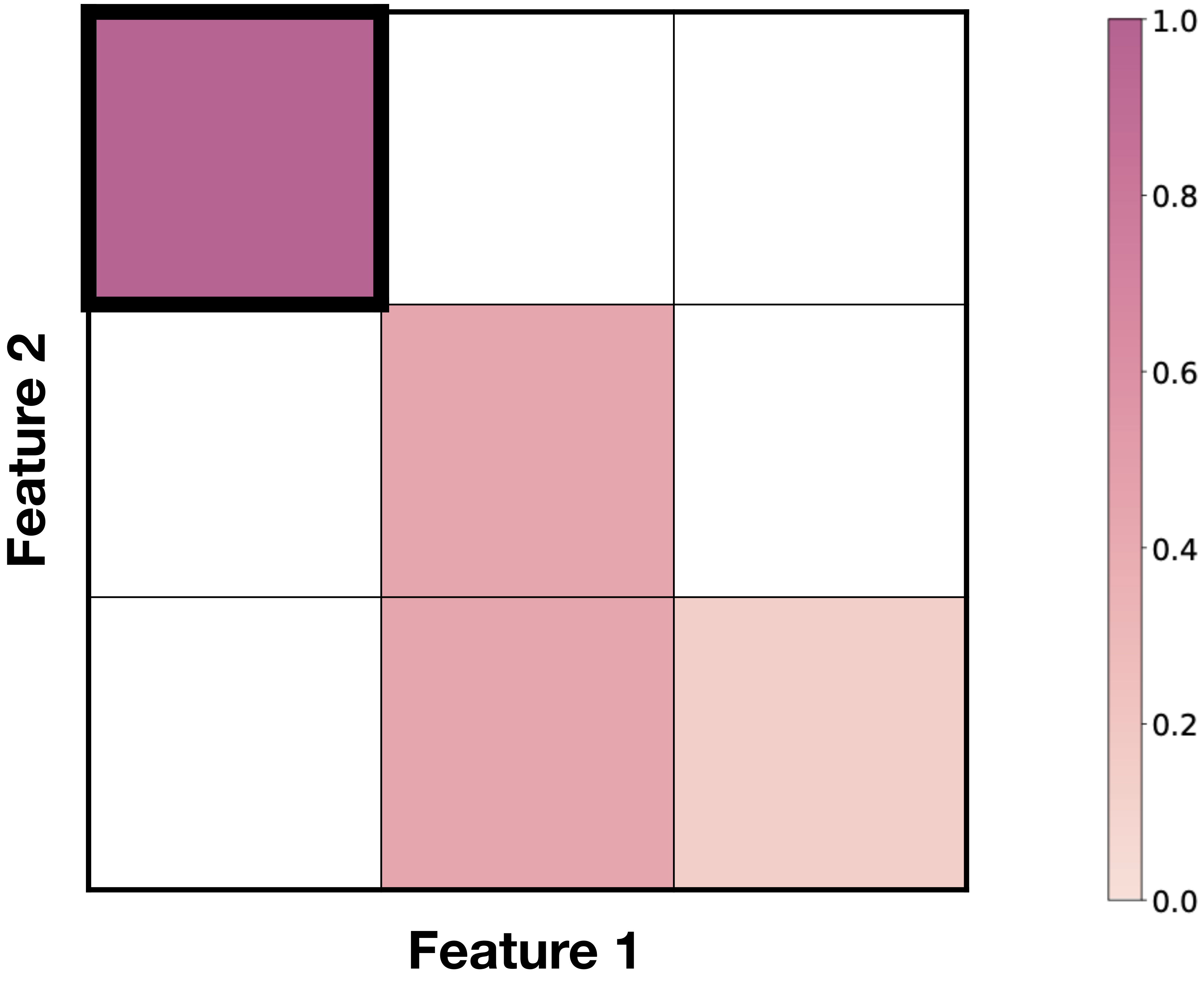
Misbehaviour Probability Map (Run 2)



Misbehaviour Probability Map (Run 3)



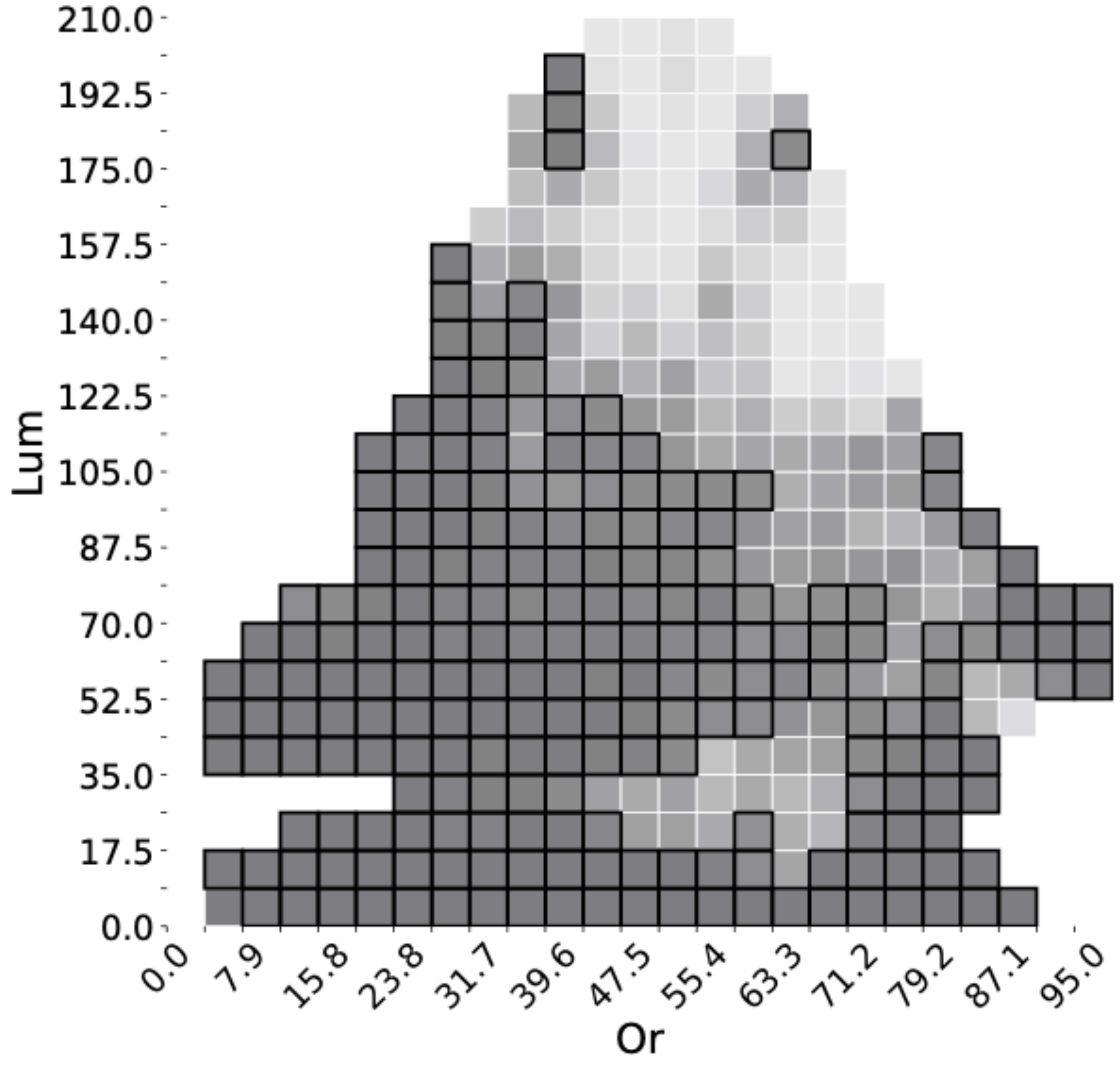
Final Misbehaviour Probability Map

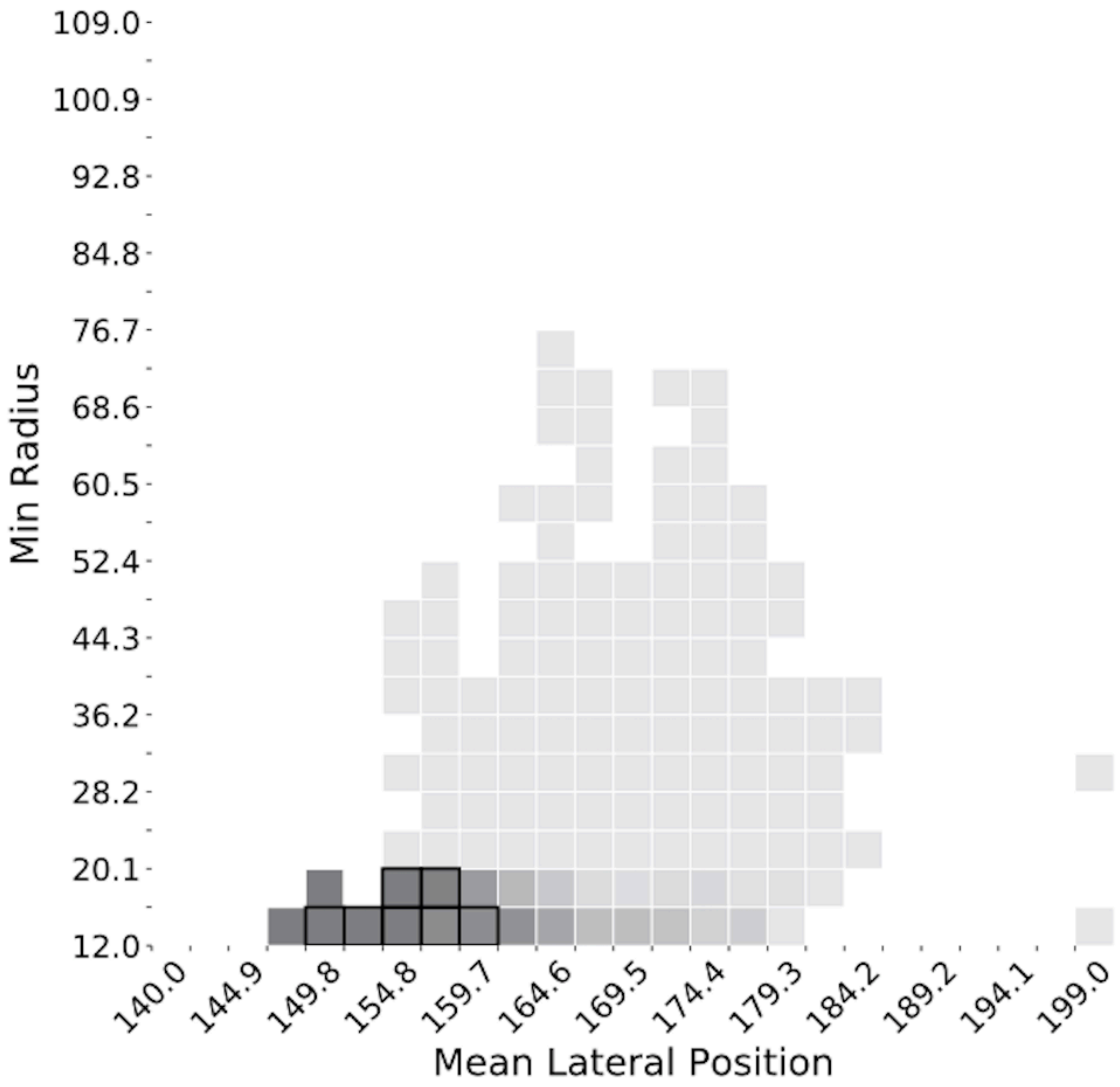


MISBEHAVIOUR CHARACTERISATION

MNIST

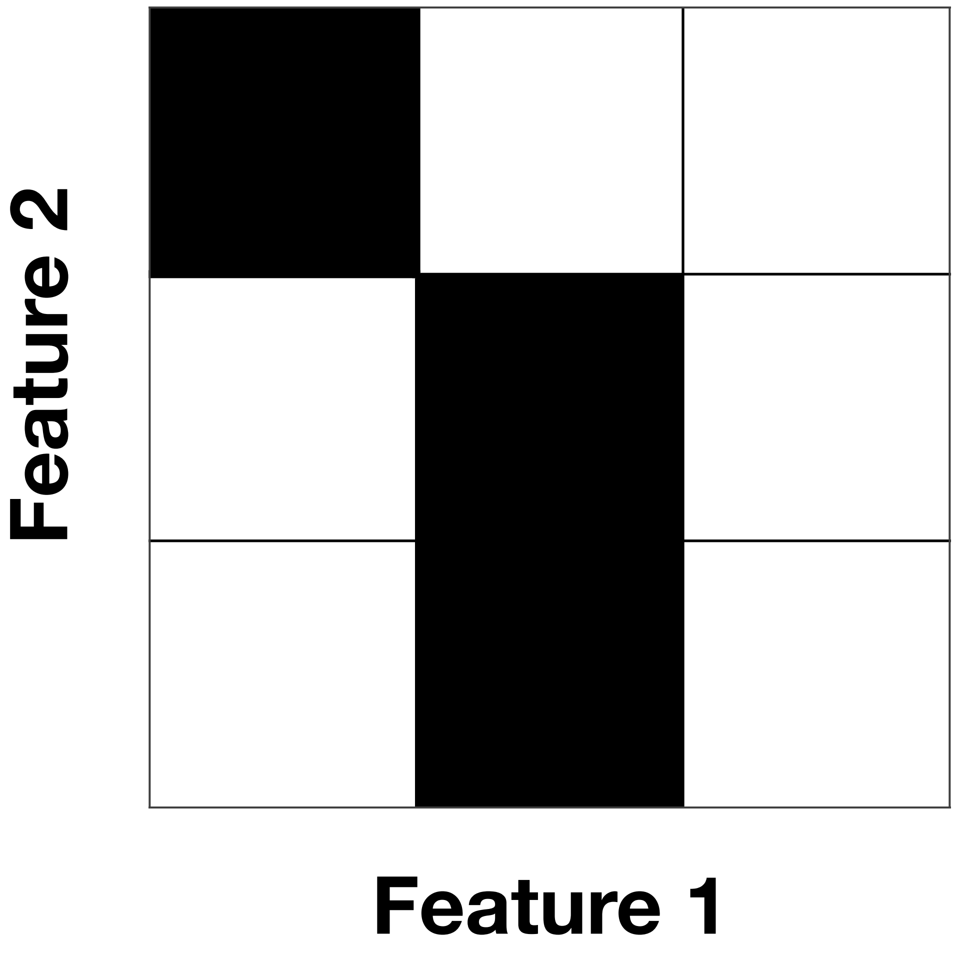
BEAMNG



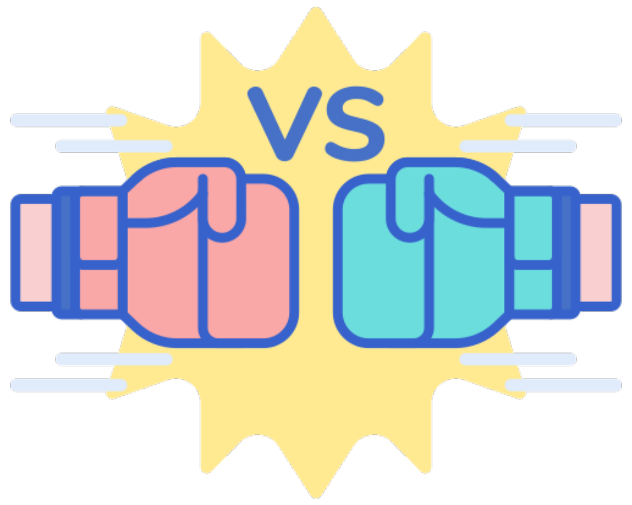


USAGE: TEST ADEQUACY CRITERION

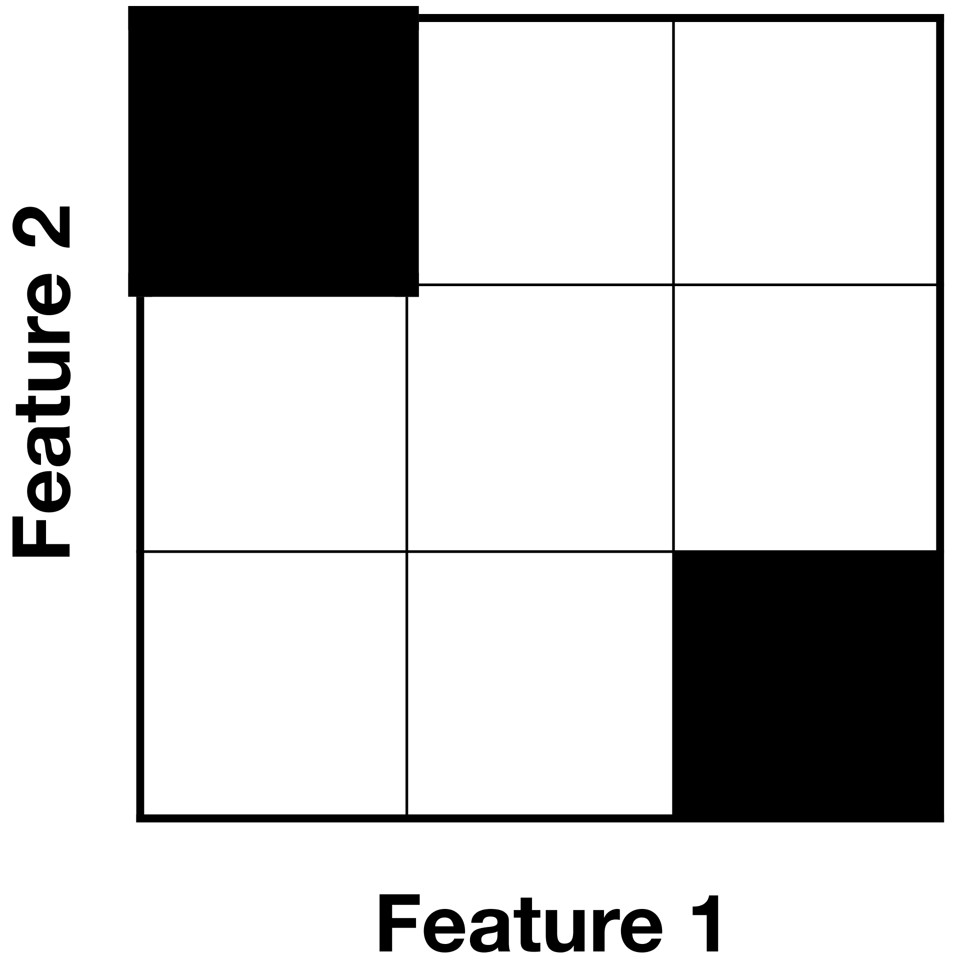
Test Generator 1



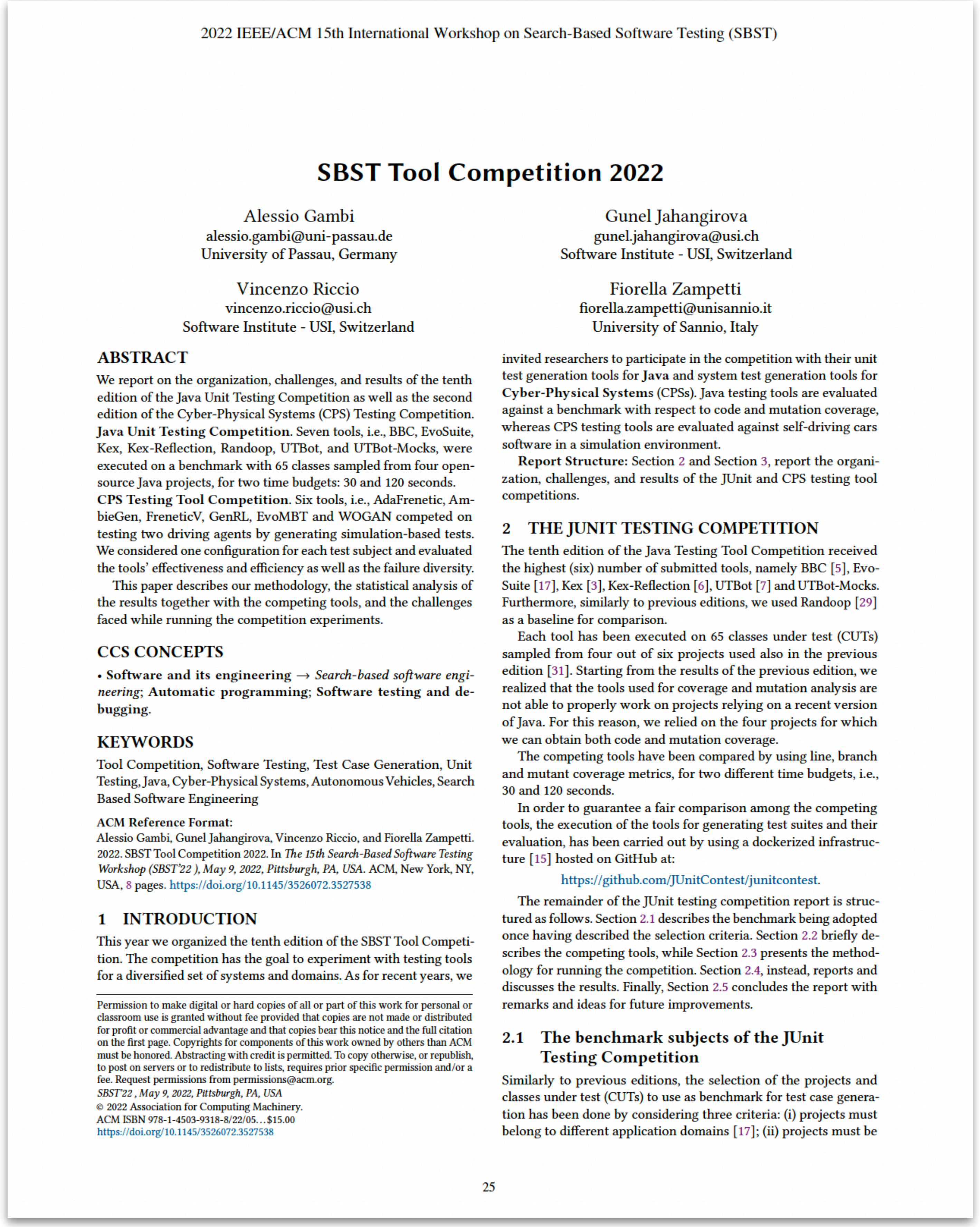
■		
	■	
	■	



Test Generator 2



■		
		■



2022 IEEE/ACM 15th International Workshop on Search-Based Software Testing (SBST)

SBST Tool Competition 2022

Alessio Gambi
alessio.gambi@uni-passau.de
University of Passau, Germany

Gunel Jahangirova
gunel.jahangirova@usi.ch
Software Institute - USI, Switzerland

Vincenzo Riccio
vincenzo.riccio@usi.ch
Software Institute - USI, Switzerland

Fiorella Zampetti
fiorella.zampetti@unisannio.it
University of Sannio, Italy

ABSTRACT

We report on the organization, challenges, and results of the tenth edition of the Java Unit Testing Competition as well as the second edition of the Cyber-Physical Systems (CPS) Testing Competition. **Java Unit Testing Competition**. Seven tools, i.e., BBC, EvoSuite, Kex, Kex-Reflection, Randoop, UTBot, and UTBot-Mocks, were executed on a benchmark with 65 classes sampled from four open-source Java projects, for two time budgets: 30 and 120 seconds. **CPS Testing Tool Competition**. Six tools, i.e., AdaFrenetic, AmbieGen, FreneticV, GenRL, EvoMBT and WOGAN competed on testing two driving agents by generating simulation-based tests. We considered one configuration for each test subject and evaluated the tools' effectiveness and efficiency as well as the failure diversity.

This paper describes our methodology, the statistical analysis of the results together with the competing tools, and the challenges faced while running the competition experiments.

CCS CONCEPTS

• **Software and its engineering** → *Search-based software engineering*; **Automatic programming**; **Software testing and debugging**.

KEYWORDS

Tool Competition, Software Testing, Test Case Generation, Unit Testing, Java, Cyber-Physical Systems, Autonomous Vehicles, Search Based Software Engineering

ACM Reference Format:
Alessio Gambi, Gunel Jahangirova, Vincenzo Riccio, and Fiorella Zampetti. 2022. SBST Tool Competition 2022. In *The 15th Search-Based Software Testing Workshop (SBST'22), May 9, 2022, Pittsburgh, PA, USA*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3526072.3527538

1 INTRODUCTION

This year we organized the tenth edition of the SBST Tool Competition. The competition has the goal to experiment with testing tools for a diversified set of systems and domains. As for recent years, we invited researchers to participate in the competition with their unit test generation tools for **Java** and system test generation tools for **Cyber-Physical Systems** (CPSs). Java testing tools are evaluated against a benchmark with respect to code and mutation coverage, whereas CPS testing tools are evaluated against self-driving cars software in a simulation environment.

Report Structure: Section 2 and Section 3, report the organization, challenges, and results of the JUnit and CPS testing tool competitions.

2 THE JUNIT TESTING COMPETITION

The tenth edition of the Java Testing Tool Competition received the highest (six) number of submitted tools, namely BBC [5], EvoSuite [17], Kex [3], Kex-Reflection [6], UTBot [7] and UTBot-Mocks. Furthermore, similarly to previous editions, we used Randoop [29] as a baseline for comparison.

Each tool has been executed on 65 classes under test (CUTs) sampled from four out of six projects used also in the previous edition [31]. Starting from the results of the previous edition, we realized that the tools used for coverage and mutation analysis are not able to properly work on projects relying on a recent version of Java. For this reason, we relied on the four projects for which we can obtain both code and mutation coverage.

The competing tools have been compared by using line, branch and mutant coverage metrics, for two different time budgets, i.e., 30 and 120 seconds.

In order to guarantee a fair comparison among the competing tools, the execution of the tools for generating test suites and their evaluation, has been carried out by using a dockerized infrastructure [15] hosted on GitHub at:

https://github.com/JUnitContest/junitcontest.

The remainder of the JUnit testing competition report is structured as follows. Section 2.1 describes the benchmark being adopted once having described the selection criteria. Section 2.2 briefly describes the competing tools, while Section 2.3 presents the methodology for running the competition. Section 2.4, instead, reports and discusses the results. Finally, Section 2.5 concludes the report with remarks and ideas for future improvements.

2.1 The benchmark subjects of the JUnit Testing Competition

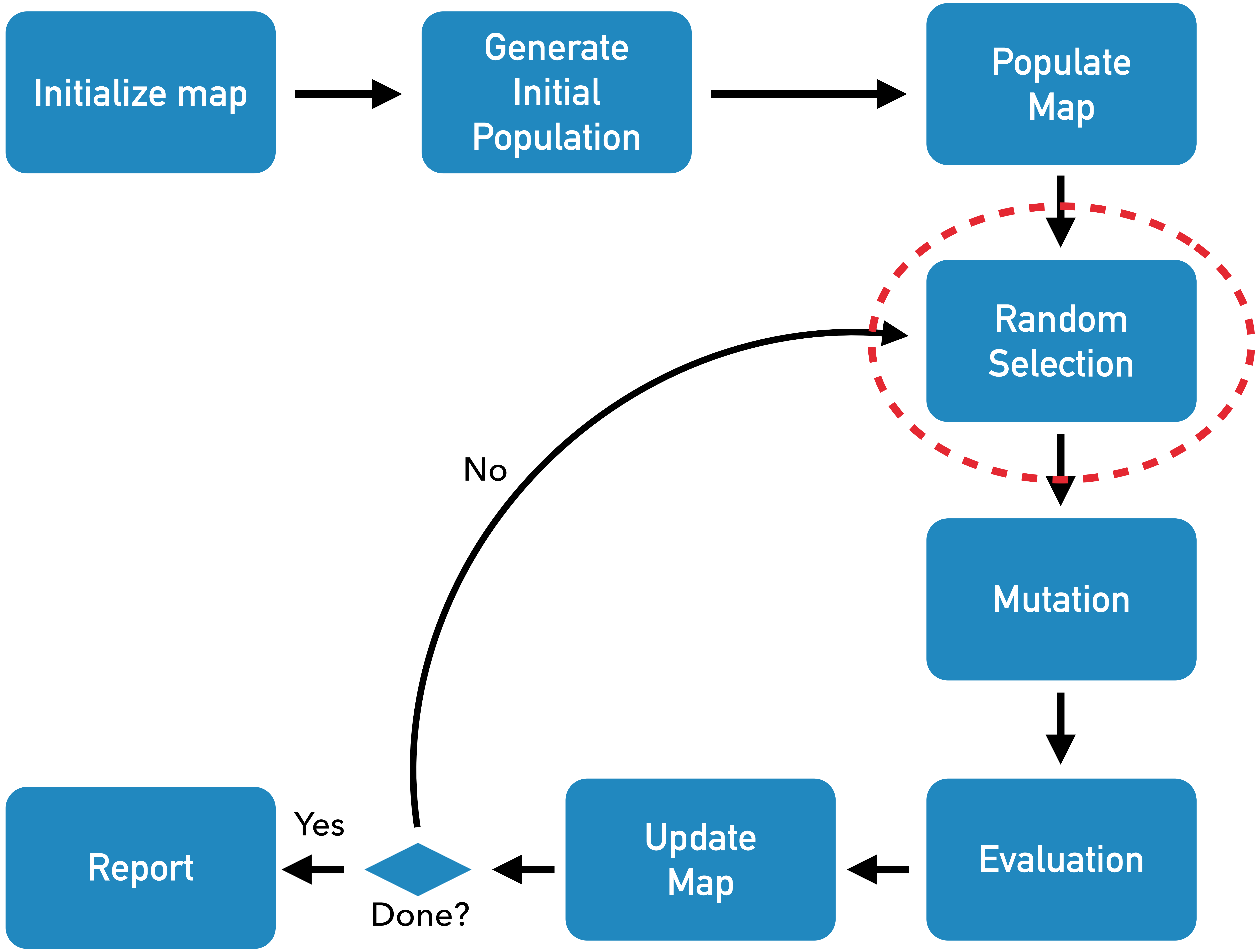
Similarly to previous editions, the selection of the projects and classes under test (CUTs) to use as benchmark for test case generation has been done by considering three criteria: (i) projects must belong to different application domains [17]; (ii) projects must be

SBST'22 , May 9, 2022, Pittsburgh, PA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9318-8/22/05…$15.00
https://doi.org/10.1145/3526072.3527538

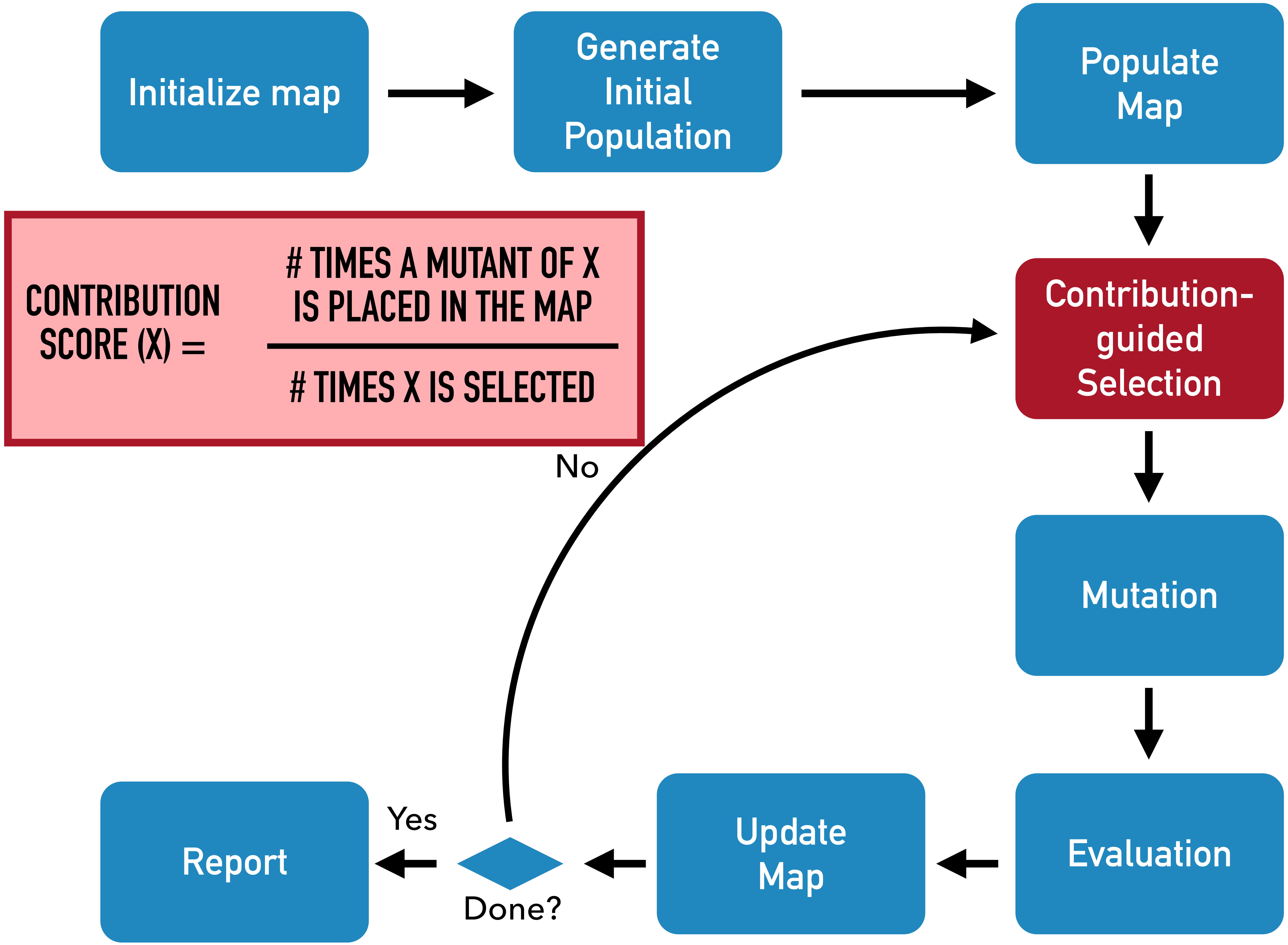
25

IEEE/ACM SBST TOOL COMPETITION 2022

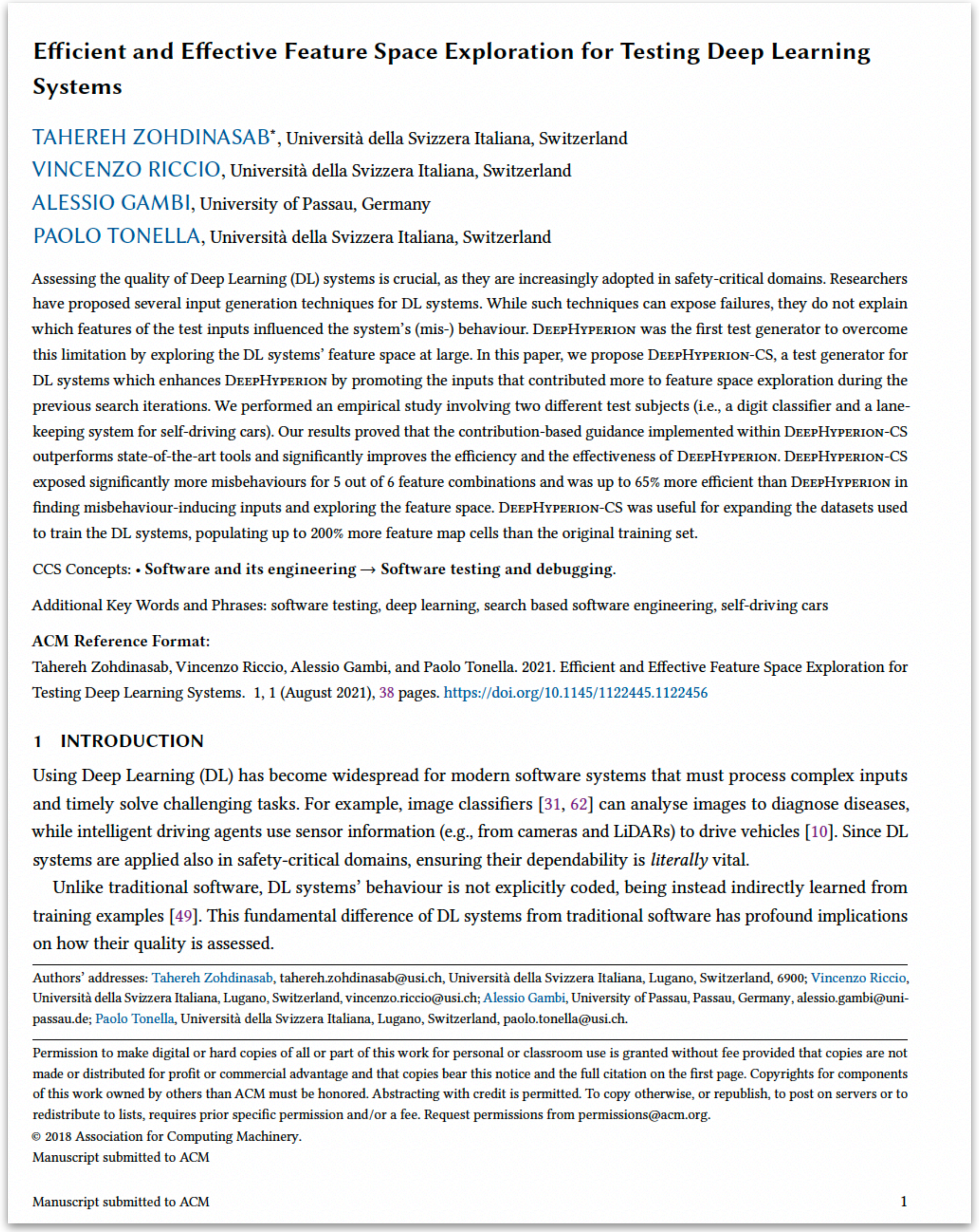
EXTENSION: DEEPHYPERION-CS



EXTENSION: DEEPHYPERION-CS



$$\text{CONTRIBUTION SCORE (X)} = \frac{\text{\# TIMES A MUTANT OF X IS PLACED IN THE MAP}}{\text{\# TIMES X IS SELECTED}}$$



Efficient and Effective Feature Space Exploration for Testing Deep Learning Systems

TAHEREH ZOHDINASAB*, Università della Svizzera Italiana, Switzerland
VINCENZO RICCIO, Università della Svizzera Italiana, Switzerland
ALESSIO GAMBI, University of Passau, Germany
PAOLO TONELLA, Università della Svizzera Italiana, Switzerland

Assessing the quality of Deep Learning (DL) systems is crucial, as they are increasingly adopted in safety-critical domains. Researchers have proposed several input generation techniques for DL systems. While such techniques can expose failures, they do not explain which features of the test inputs influenced the system's (mis-) behaviour. DeepHyperion was the first test generator to overcome this limitation by exploring the DL systems' feature space at large. In this paper, we propose DeepHyperion-CS, a test generator for DL systems which enhances DeepHyperion by promoting the inputs that contributed more to feature space exploration during the previous search iterations. We performed an empirical study involving two different test subjects (i.e., a digit classifier and a lane-keeping system for self-driving cars). Our results proved that the contribution-based guidance implemented within DeepHyperion-CS outperforms state-of-the-art tools and significantly improves the efficiency and the effectiveness of DeepHyperion. DeepHyperion-CS exposed significantly more misbehaviours for 5 out of 6 feature combinations and was up to 65% more efficient than DeepHyperion in finding misbehaviour-inducing inputs and exploring the feature space. DeepHyperion-CS was useful for expanding the datasets used to train the DL systems, populating up to 200% more feature map cells than the original training set.

CCS Concepts: • **Software and its engineering → Software testing and debugging.**

Additional Key Words and Phrases: software testing, deep learning, search based software engineering, self-driving cars

ACM Reference Format:
Tahereh Zohdinasab, Vincenzo Riccio, Alessio Gambi, and Paolo Tonella. 2021. Efficient and Effective Feature Space Exploration for Testing Deep Learning Systems. 1, 1 (August 2021), 38 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

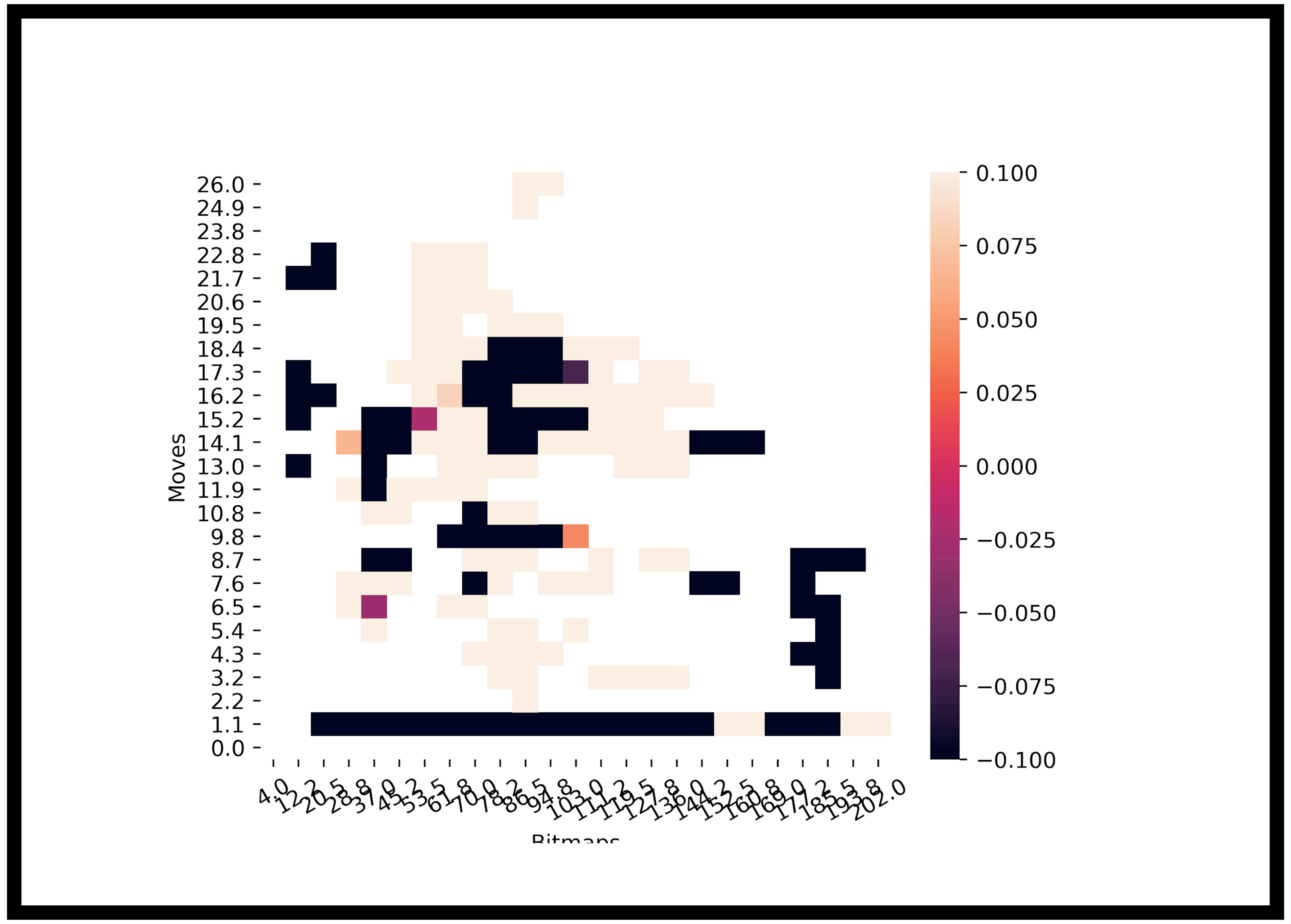
Using Deep Learning (DL) has become widespread for modern software systems that must process complex inputs and timely solve challenging tasks. For example, image classifiers [31, 62] can analyse images to diagnose diseases, while intelligent driving agents use sensor information (e.g., from cameras and LiDARs) to drive vehicles [10]. Since DL systems are applied also in safety-critical domains, ensuring their dependability is *literally* vital.

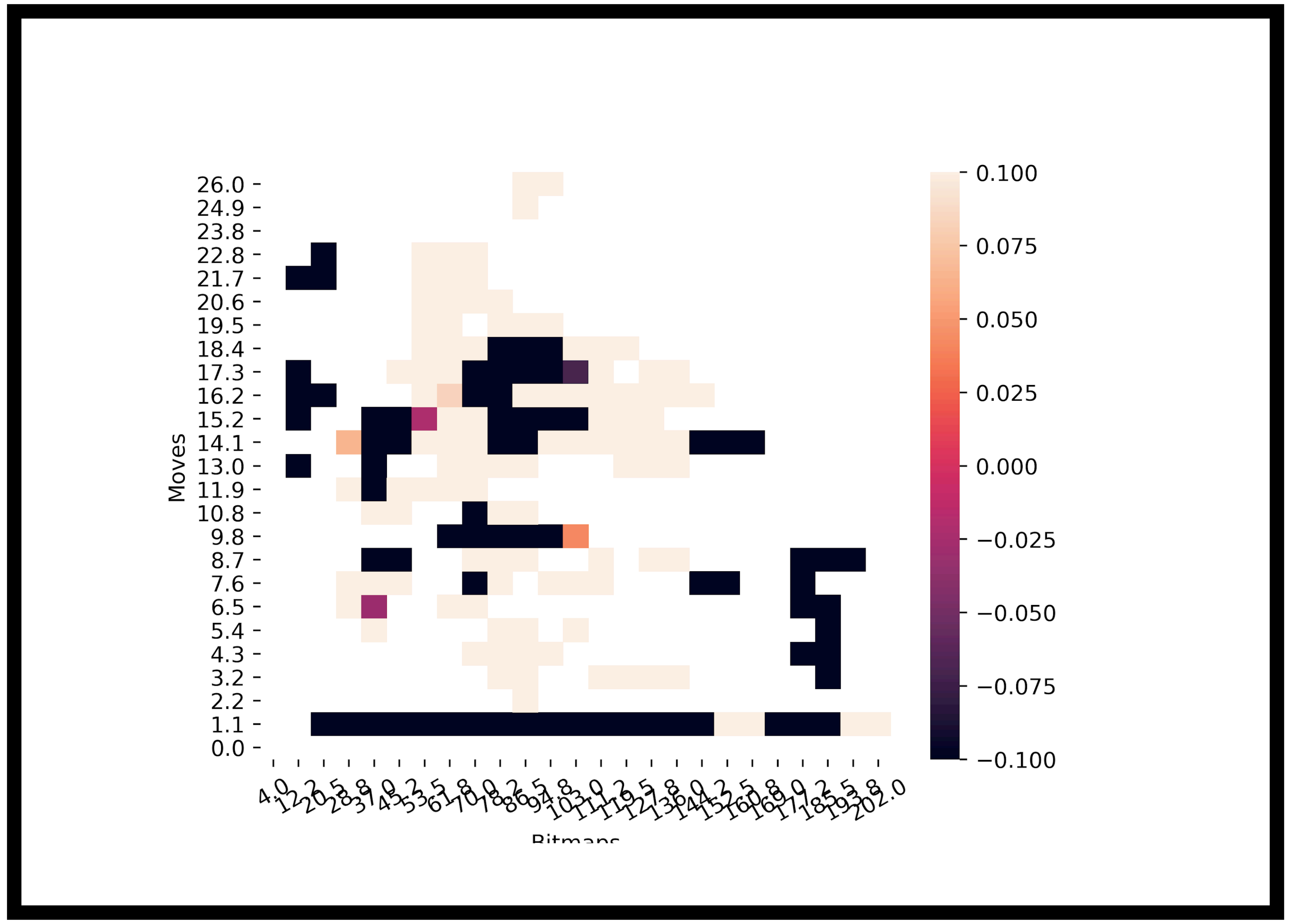
Unlike traditional software, DL systems' behaviour is not explicitly coded, being instead indirectly learned from training examples [49]. This fundamental difference of DL systems from traditional software has profound implications on how their quality is assessed.

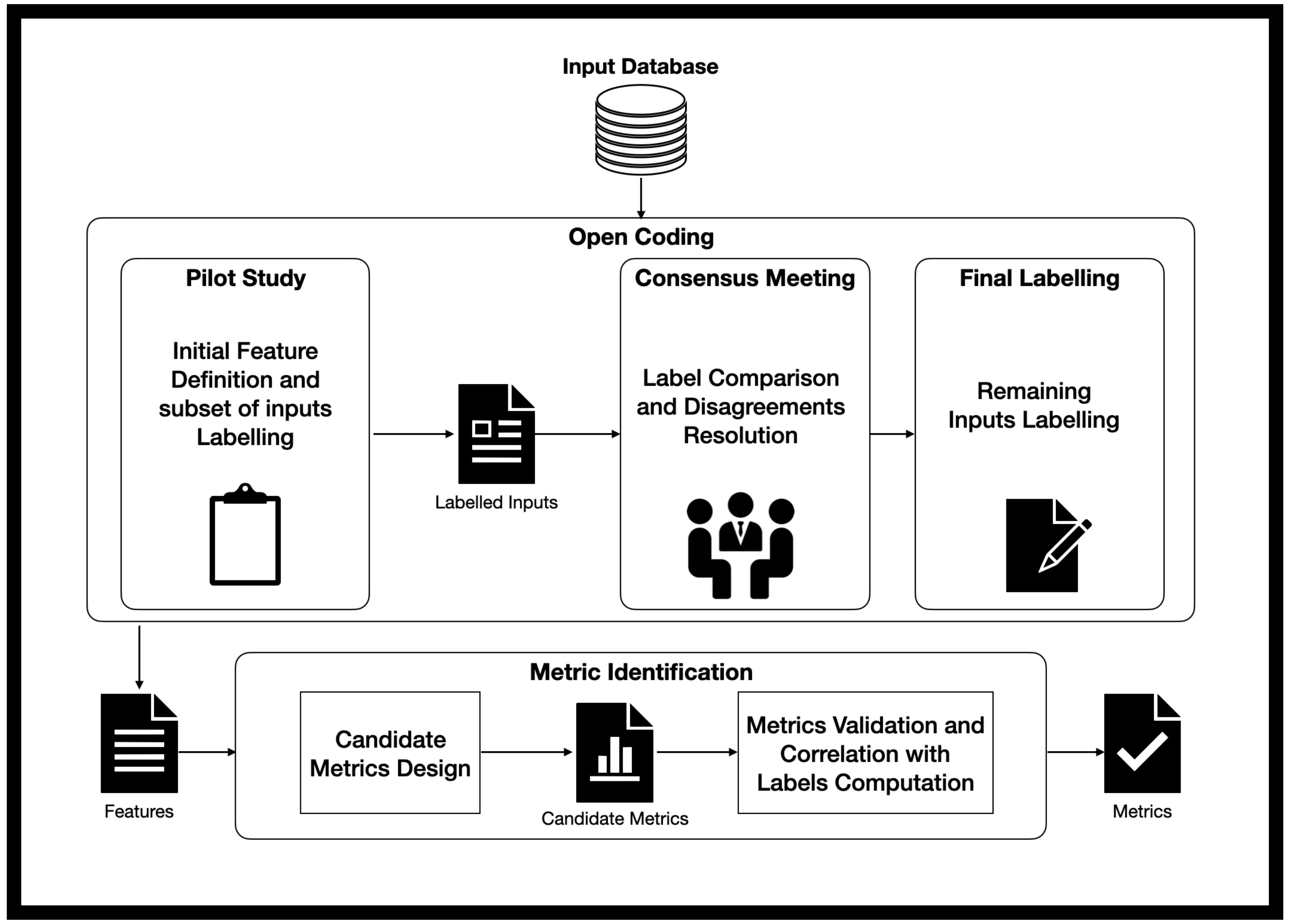
Authors' addresses: Tahereh Zohdinasab, tahereh.zohdinasab@usi.ch, Università della Svizzera Italiana, Lugano, Switzerland, 6900; Vincenzo Riccio, Università della Svizzera Italiana, Lugano, Switzerland, vincenzo.riccio@usi.ch; Alessio Gambi, University of Passau, Passau, Germany, alessio.gambi@uni-passau.de; Paolo Tonella, Università della Svizzera Italiana, Lugano, Switzerland, paolo.tonella@usi.ch.

Manuscript submitted to ACM

ACM TRANSACTIONS ON SOFTWARE ENGINEERING AND METHODOLOGY (TOSEM) 2022


Moves
26.0
24.9
23.8
22.8
21.7
20.6
19.5
18.4
17.3
16.2
15.2
14.1
13.0
11.9
10.8
9.8
8.7
7.6
6.5
5.4
4.3
3.2
2.2
1.1
0.0
4.0
12.2
20.5
28.8
37.0
45.2
53.5
61.8
70.0
78.2
86.5
94.8
103.0
111.2
119.5
127.8
136.0
144.2
152.5
160.8
169.0
177.2
185.5
193.8
202.0
Bitmaps
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100




Input Database

Open Coding

Pilot Study

Initial Feature Definition and subset of inputs Labelling

Labelled Inputs

Consensus Meeting

Label Comparison and Disagreements Resolution

Final Labelling

Remaining Inputs Labelling

Features

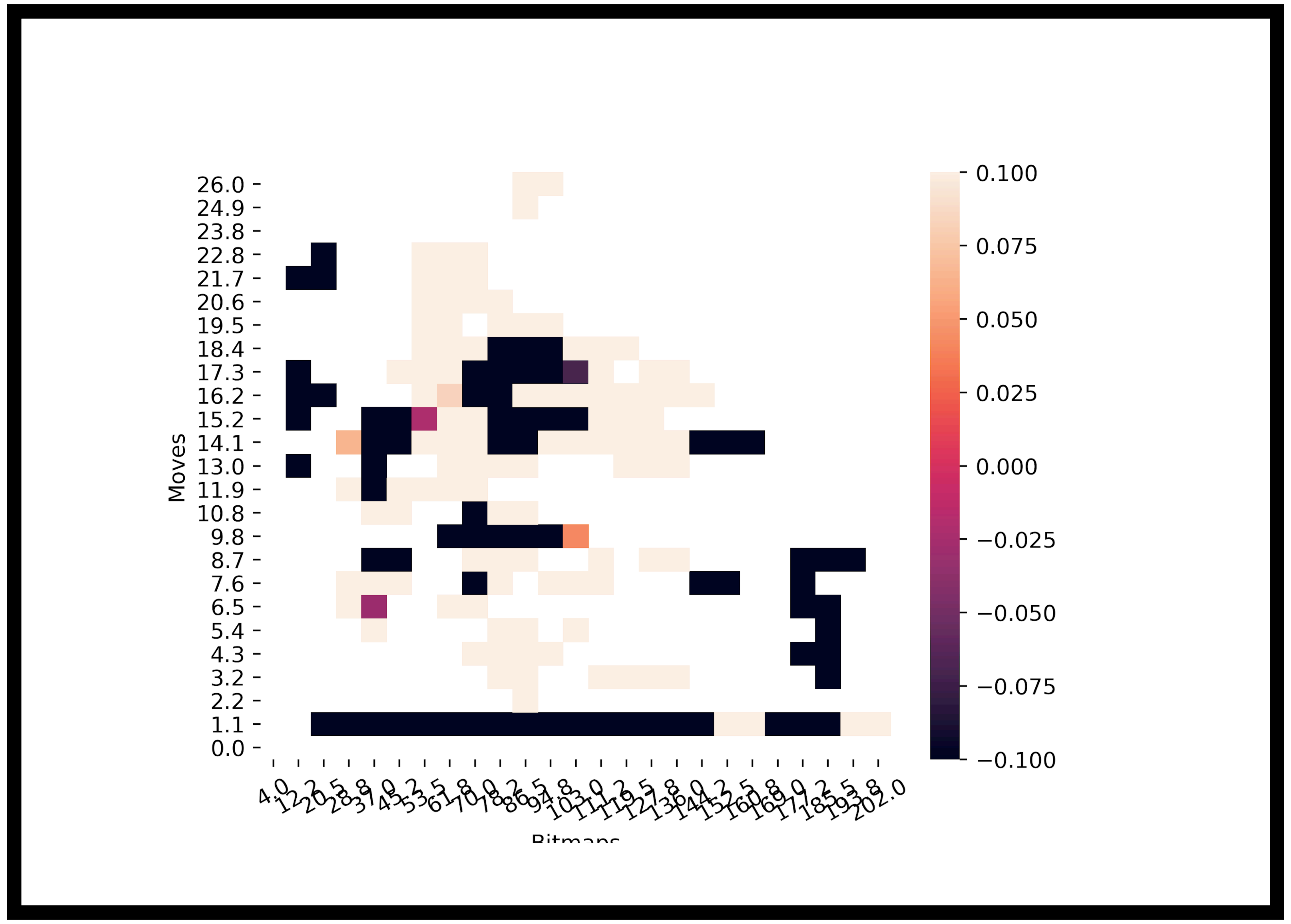
Metric Identification

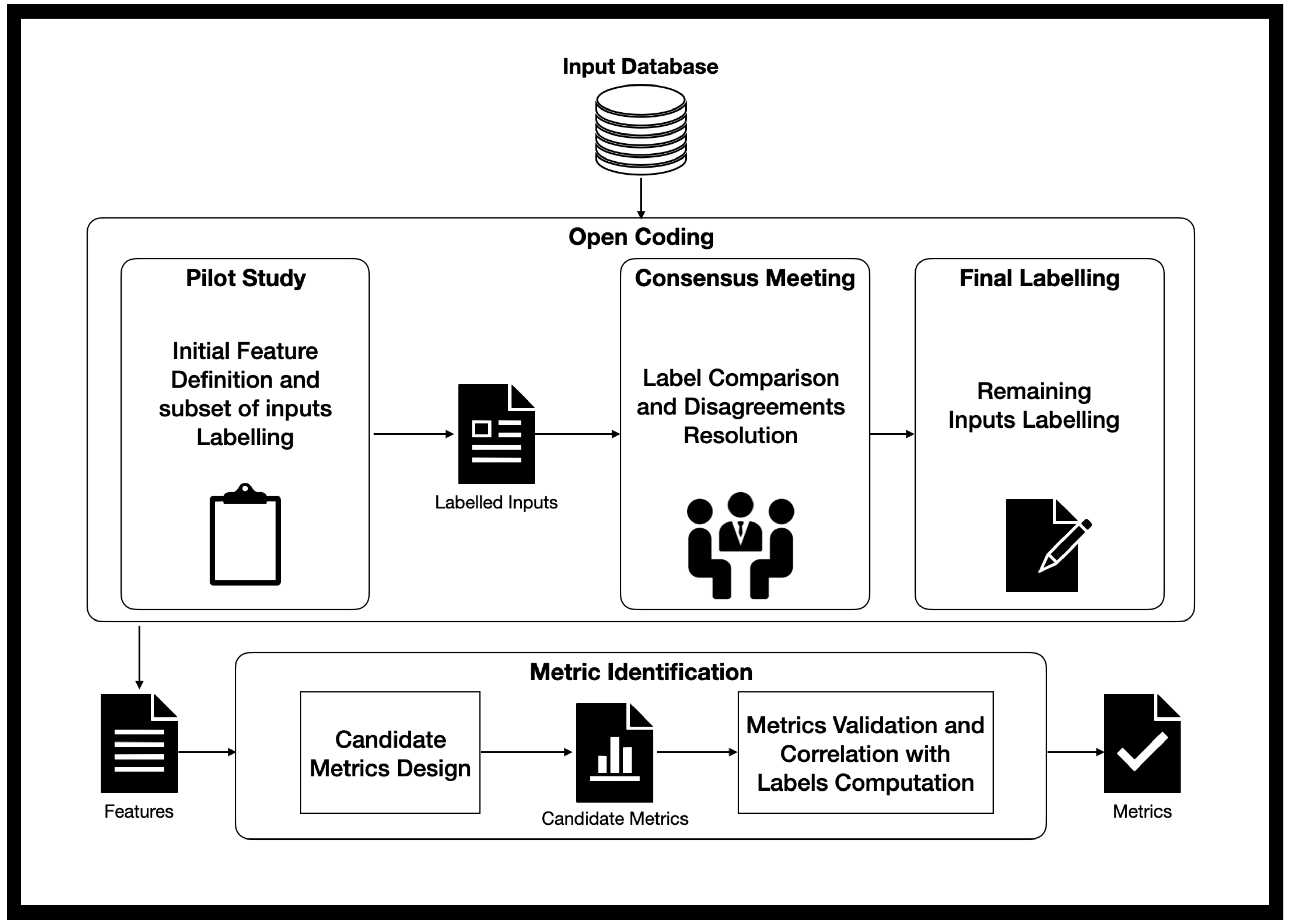
Candidate Metrics Design

Candidate Metrics

Metrics Validation and Correlation with Labels Computation

Metrics





Input Database

Open Coding

Pilot Study

Initial Feature Definition and subset of inputs Labelling

Labelled Inputs

Consensus Meeting

Label Comparison and Disagreements Resolution

Final Labelling

Remaining Inputs Labelling

Features

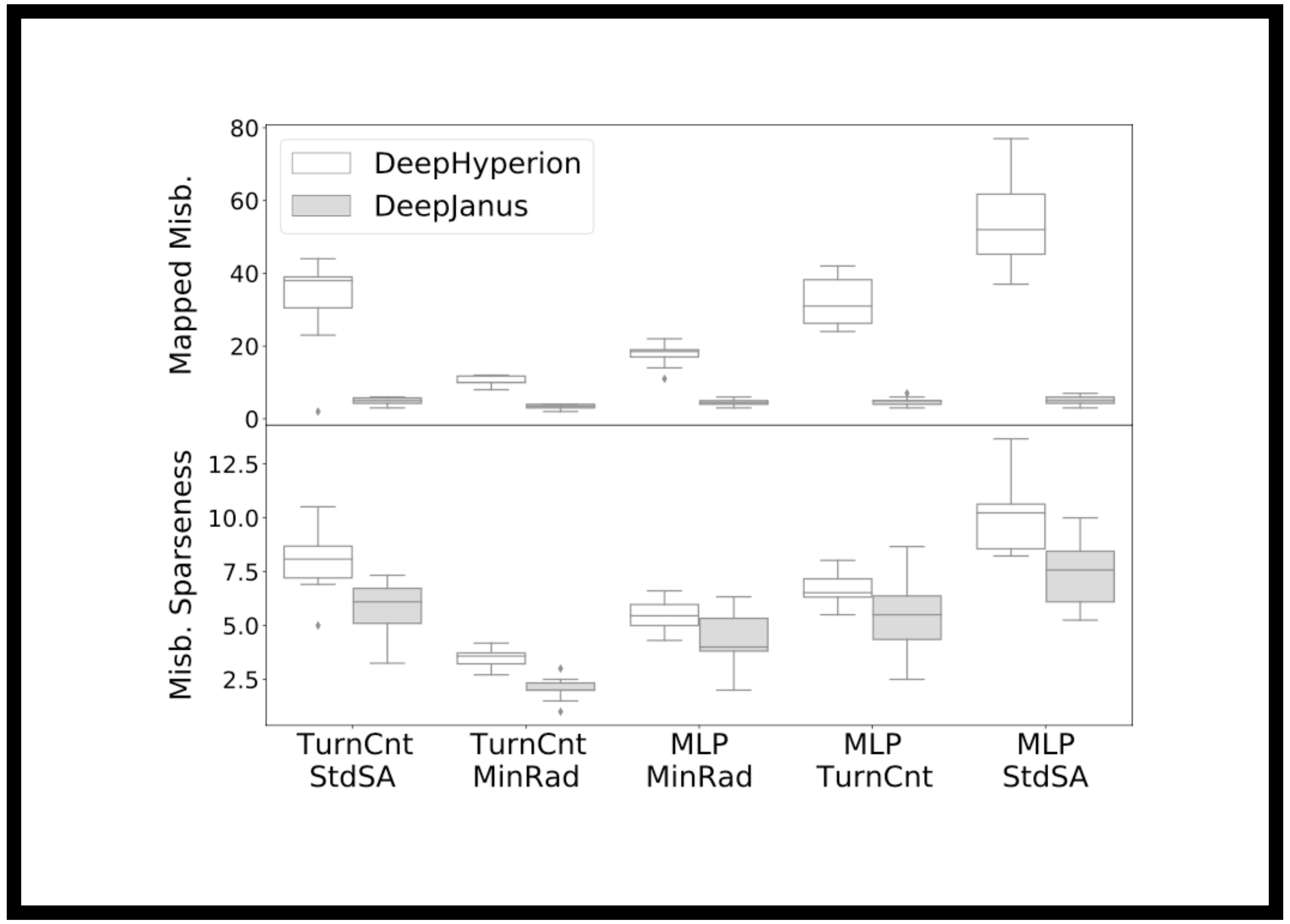
Metric Identification

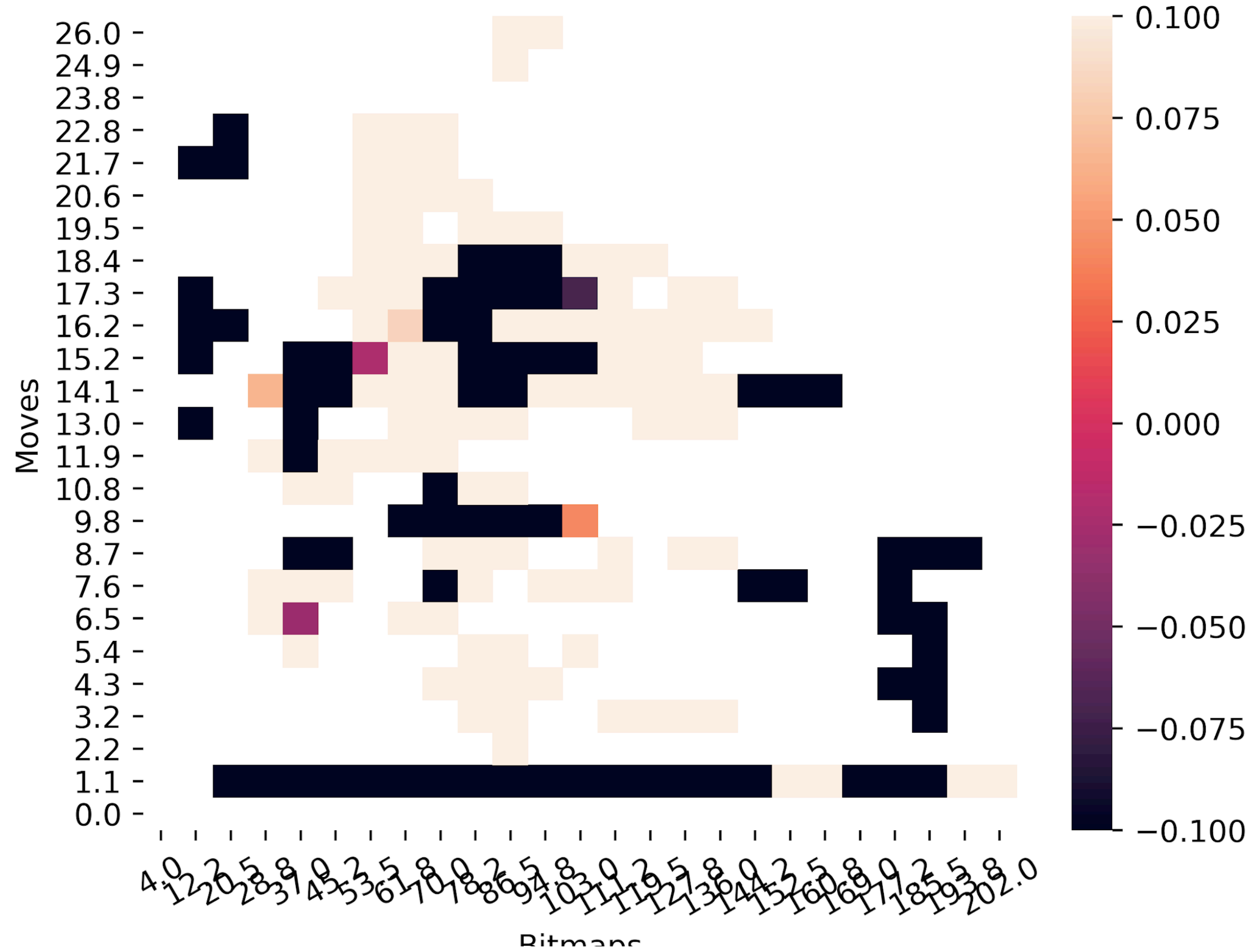
Candidate Metrics Design

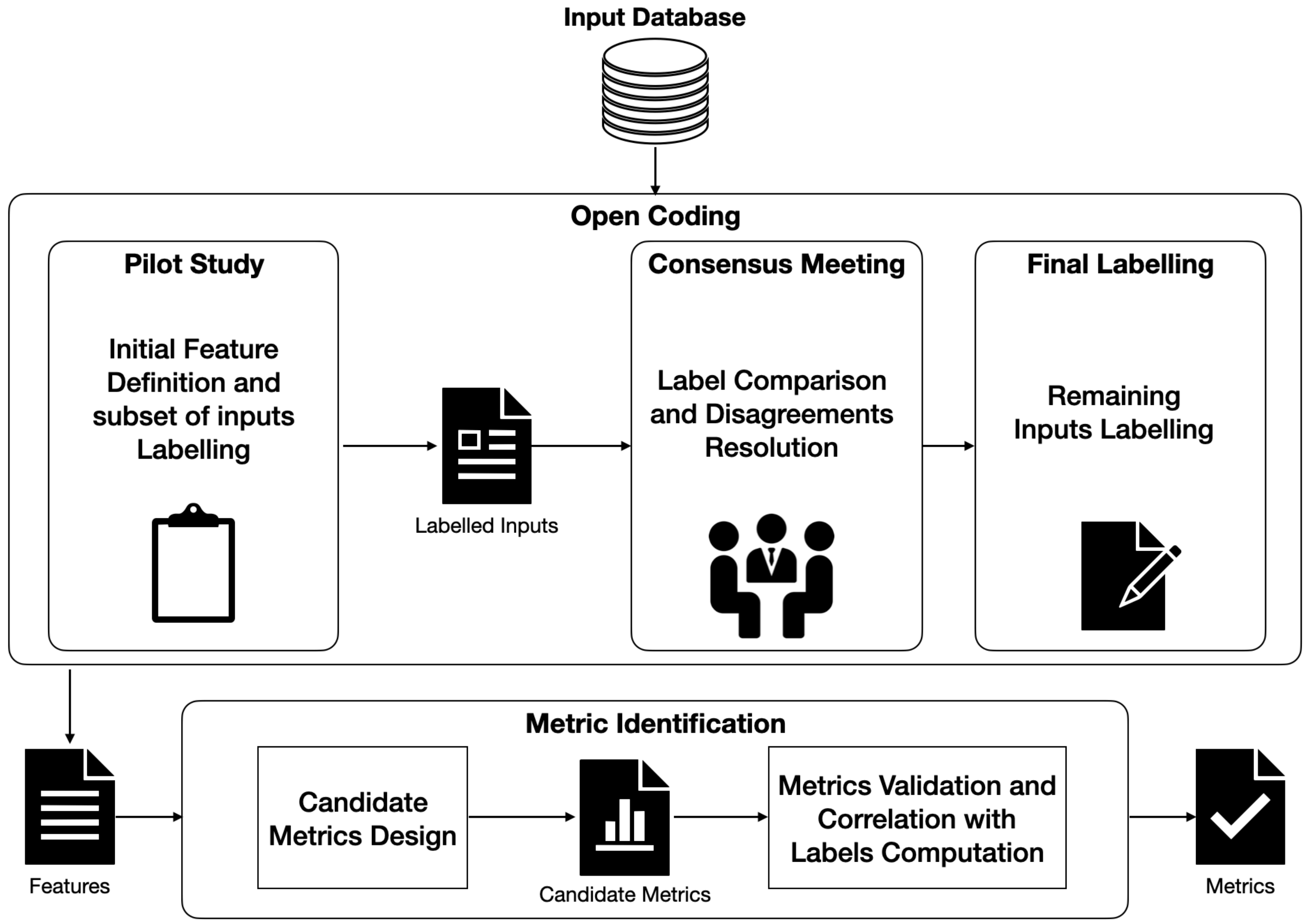
Candidate Metrics

Metrics Validation and Correlation with Labels Computation

Metrics







Input Database

Open Coding

Pilot Study

Initial Feature Definition and subset of inputs Labelling

Labelled Inputs

Consensus Meeting

Label Comparison and Disagreements Resolution

Final Labelling

Remaining Inputs Labelling

Features

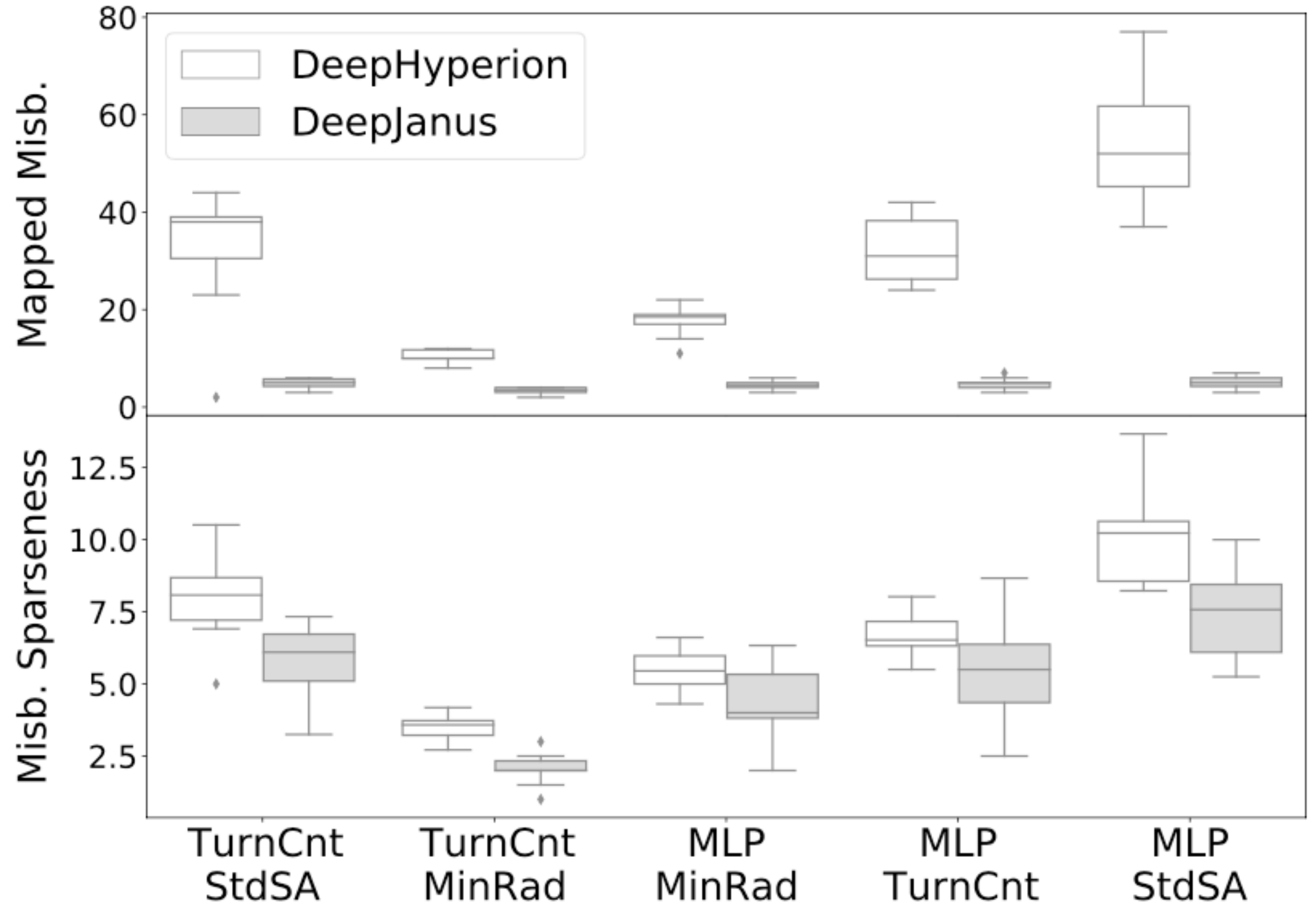
Metric Identification

Candidate Metrics Design

Candidate Metrics

Metrics Validation and Correlation with Labels Computation

Metrics


Mapped Misb.
DeepHyperion
DeepJanus
Misb. Sparseness
TurnCnt StdSA
TurnCnt MinRad
MLP MinRad
MLP TurnCnt
MLP StdSA