European Research Council
Established by the European Commission

Self-assessment Oracles for Anticipatory Testing

TECHNICAL REPORT: TR-Precrime-2020-01

Vincenzo Riccio, Paolo Tonella
Università della Svizzera italiana

Model-based Exploration of the Frontier of Behaviours for Deep Learning System Testing

Project no.: 787703
Funding scheme: ERC-2017-ADG
Start date of the project: January 1, 2019
Duration: 60 months

Technical report num.: TR-Precrime-2020-01
Date: January, 2020
Organization: Università della Svizzera italiana
Authors: Vincenzo Riccio, Paolo Tonella
Università della Svizzera italiana
Dissemination level: Public
Revision: 1.0

Università della Svizzera Italiana (USI)

Principal investigator: Prof. Paolo Tonella
E-mail: paolo.tonella@usi.ch
Address: Via Buffi, 13 – 6900 Lugano – Switzerland
Tel: +41 58 666 4848
Project website: https://www.pre-crime.eu/

Abstract

With the increasing adoption of Deep Learning (DL) for critical tasks, such as autonomous driving, the evaluation of the quality of systems that rely on DL has become crucial. Once trained, DL systems produce an output for any arbitrary numeric vector provided as input, regardless of whether it is within or outside the validity domain of the system under test. Hence, the quality of such systems is determined by the intersection between their validity domain and the regions where their outputs exhibit a misbehaviour.
In this paper, we introduce the notion of frontier of behaviours, i.e., the inputs at which the DL system starts to misbehave. If the frontier of misbehaviours is outside the validity domain of the system, the quality check is passed. Otherwise, the inputs at the intersection represent quality deficiencies of the system. We developed DEEPJANUS, a search-based tool that generates frontier inputs for DL systems. The experimental results obtained for the lane keeping component of a self-driving car show that the frontier of a well trained system contains almost exclusively unrealistic roads that violate the best practices of civil engineering, while the frontier of a poorly trained one includes many valid inputs that point to serious deficiencies of the system.

Contents

1 Introduction

Deep Neural Networks (DNNs) have been applied successfully to complex tasks such as image processing, speech recognition and natural language analysis. The range of applications of DNNs is huge and includes autonomous driving, medical diagnosis, financial trading and automated customer services. As a consequence, the problem of testing Deep Learning (DL) systems to ensure their dependability has become critical.

Existing approaches to generate tests for DL systems can be split into two groups: (1) techniques that directly manipulate the raw input data [43, 51, 48] and (2) techniques that derive input data from a model of the input domain [1, 16]. In the former case the result is a so called *adversarial example*, while in the latter case it can be regarded as a *critical test scenario*. In both cases, test generation is guided by the criticality of the produced inputs, measured either directly as a misclassification/inconsistent behaviour [1, 16, 48] or mediated by a proxy such as surprise [24] or neuron coverage [32, 43]. Adversarial examples, e.g. images obtained by manipulation of the pixels of an image taken from the camera of an autonomous car, may produce very unlikely (or even impossible) cases, whose resolution might have no impact on the system's reliability. Critical test scenarios obtained by model-based input generation tend to be more realistic. However, the existing model-based approaches do not aim at covering thoroughly and characterising the region where the DL system misbehaves.

The ISO/PAS standard 21448 [14] on safety of autonomous and assisted driving prescribes that unsafe situations should be identified and be demonstrated to be sufficiently implausible. When unsafe situations are plausible, countermeasures must be adopted. Manual identification of unsafe conditions for DNNs is challenging, because their behaviour cannot be decomposed via logical conditions, as done e.g. with root cause analysis [23]. This motivates our work on the automated identification of the frontier of behaviours of DL systems: we aim to support engineers in identifying and checking the plausibility of the frontier of behaviours.

In this paper, we introduce a novel way to assess the quality of DL systems, based on a new notion: the *frontier of behaviours*. The frontier of behaviours of a DL system is a set of pairs of inputs that are similar to each other and that trigger different behaviours of the DL system. It represents the border of the input region where the DL system behaves as expected. For instance, the frontier of a classifier of hand-written digits consists of the pairs of similar digits that are classified differently (one correctly and the other incorrectly). The frontier of behaviours of a low quality DL system may include pairs of inputs that intersect the validity domain, being similar to nominal cases for which the system is expected to behave correctly according to the requirements. On the contrary, a DL system of high quality will start to misbehave only on inputs that deviate substantially from the nominal ones, with small or no intersection with the validity domain (e.g. a digit "5" is misclassified only when it becomes unrecognisable or indistinguishable from another digit, such as "6").

We have adopted a model-based input generation approach to produce realistic inputs, under the assumption that a high fidelity model of the input is available for the DL system under test. There are several domains in which the development of input models is standard practice, among which safety-critical domains such as automotive and aerospace engineering [27]. In other domains, such as image classification, input models can be constructed (e.g. in Unity [47]) or reverse engineered. Our tool DEEPJANUS implements a multi-objective evolutionary algorithm to manipulate the input model, with the overall goal of achieving thorough exploration of the frontier of behaviours. To this aim, one of its two fitness functions promotes diversity, so as to spread the solutions along the entire frontier, and minimises the distance between the elements in each pair. The other fitness function pushes the solutions to the frontier. The output of DEEPJANUS provides developers with a thorough and human-interpretable picture of the system's quality. In fact, the elements of each pair in the frontier may be deemed as within or outside the validity domain of the system (in the latter case, they are irrelevant for the reliability of the DL system). When used to compare alternative DL systems that solve the same problem, metrics of the frontier size (e.g. its radius) are useful to show quantitatively if the region contained in one frontier is substantially smaller/larger than the region contained in the other.

The proposed technique was evaluated on both a classification problem (hand-written digit recognition) and a regression problem (steering angle prediction in a self-driving car). The frontier of the digit classifier was evaluated by 20 human assessors recruited on a crowdsourcing platform. Results show that a high quality classifier has a smaller intersection with the validity domain with respect to a poorly trained one. The frontier of the self-driving car was evaluated by assessing the conformance of the shapes of the roads at the frontier to the guidelines for the design of American highways [39]. Frontier roads obtained for a high quality system violate such guidelines, showing that the system misbehaves only in extreme cases.

Such results were confirmed by quantitative measures of the frontier radius, which was larger for the high quality than for the low quality DL system, and qualitative assessment of the frontier images/roads, which are more challenging for humans when taken from the high quality system frontier.

We compared our results with those produced by DLFuzz [18], a tool that generates boundary adversarial inputs by pixel manipulation, and found that it generates corner cases that are more concentrated and less realistic than those of DEEPJANUS.

2 Background

2.1 Deep Learning Systems

In this work, we refer to software systems that include one or more DNNs as DL systems [32]. Their behaviour is defined both by the code that implements them and by the data used to train their DNN components.

A DNN can be considered as a black-box component that transforms a numeric input vector into a numeric output. It can accomplish various tasks, such as the prediction of the steering angle of a self-driving car starting from the image captured by a camera sensor [7]. In a regression problem the output is a continuous value, whereas in a classification problem the output is a discrete class.

A DNN consists of a collection of computation units, called *neurons*, organised into *layers* that are connected sequentially (i.e. neurons of layer n are only connected to neurons of layer $n + 1$). Each connection of the network has a *weight*, which determines the propagation of a neuron's output to the next neuron. Among the layers of a DNN, the input layer receives external data, the output layer produces the final result, while internal, hidden layers perform intermediate processing (e.g. feature extraction). Each neuron computes its output by applying an activation function (e.g., sigmoid) to the weighted sum of its inputs.

To accomplish a task, DNNs are iteratively trained through a large set of labelled training data. During training, a DNN learns how to predict the expected label by adjusting the weights of the network. The number of training iterations in which the whole training set is processed by the network is a hyper-parameter called *epochs*. The number of epochs influences how the network fits the training data and how it will be able to generalise to unseen inputs. Another fundamental hyper-parameter is the *learning rate*, which defines the amount of corrections that are applied to the weights at each training iteration.

2.2 Evolutionary Search and Novelty Search

Evolutionary algorithms are a family of meta-heuristic optimisation algorithms that evolve a population of *individuals* (i.e. candidate solutions to an optimisation problem) by means of genetic operators such as *mutation* and *crossover*. A *fitness function* provides an approximate, heuristic distance of each candidate solution from the searched optimum. During evolution, the best individuals are selected for the next population based on the values of the fitness function.

Multi and many objective evolutionary algorithms generalise the basic evolutionary algorithms to multiple fitness functions. Since, in such a case, there is no single dimension on which to compare individuals during selection, the best ones are obtained by Pareto front analysis as those that are not dominated by any other individual. Multi and many objective genetic algorithms have proved to be particularly effective in test case generation [40, 34].

The solutions found by search algorithms might be concentrated in a small portion of the input space, especially when the search landscape includes local optima with a large basin of attraction. If the goal is not only to find good solutions according to the fitness functions, but also to find solutions spread across the entire input space (as in our case), evolutionary algorithms can be combined with *novelty search*. Novelty search algorithms reward individuals that exhibit diversity of behaviours, instead of promoting only those that contribute to progress toward the optimum [29, 35]. They trade off a lower pressure toward optimal fitness values with a higher diversity in the population being evolved.

3 Motivating Example

In this Section, we provide a motivating example that shows how the frontiers of behaviours can help characterising the quality of DL systems. Let us consider the frontiers of behaviours of two DL systems

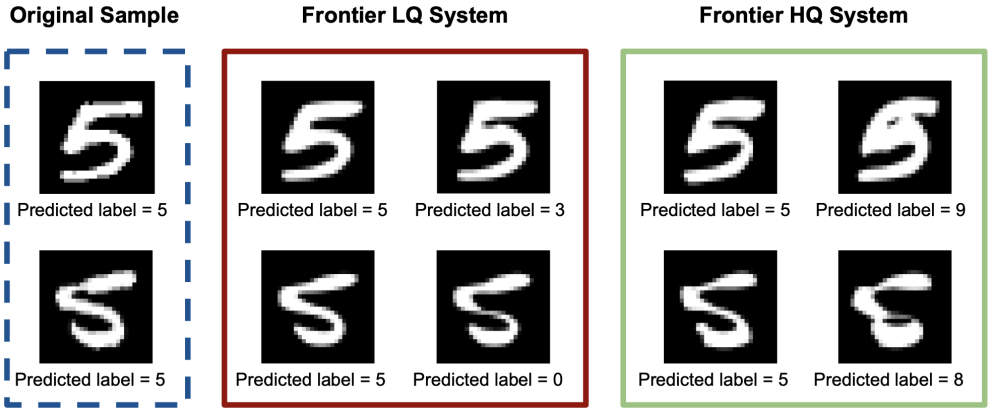

Figure 1: Four pairs of elements in the frontiers of the considered DL systems. The first column shows two original samples from the MNIST data set. The second and third columns show the pairs in the frontier of system LQ; fourth and fifth columns show the frontier pairs of system HQ.

that perform the same task but exhibit different levels of quality in terms of test accuracy (namely HQ: High Quality: LQ: Low Quality). Both systems consist of a classifier of handwritten digits that predicts which digit is represented by an input image. In a classification problem such as this one, the frontier is represented by pairs of similar inputs that are classified differently (one correctly, the other incorrectly).

To assess the difference between the frontiers of these two systems, we consider two images of handwritten digits taken from the MNIST [28] dataset that are labeled correctly (i.e. as number "5") by both systems. They are shown in the first column of Fig. 1. Then, we apply slight changes to the shape of the two inputs. Technically, this is achieved by first extracting a vector model of the digits and then manipulating the control points of such model. The result consists of two pairs of samples in the frontier of each considered system, i.e. LQ (second and third column) and HQ (fourth and fifth column).

We can notice that the inputs in the frontier of LQ are very similar to the original samples. Moreover, all the misclassified inputs in its frontier are still clearly recognisable as digit "5". Instead, the frontier of HQ contains inputs that are probably challenging to classify even for humans. In particular, the first element of the fifth column has the general shape of a five, but it could also be considered as a nine, since the upper part of the figure forms a circle. The second element of the fifth column does not look like any reasonably classifiable digit, despite its similarity with the corresponding member of the pair on the other side of the frontier.

To summarise, the frontier of a low quality DL system is expected to contain samples that are quite close to those that the system is supposed to classify correctly, indicating a poor generalisation capability. Differently, the frontier of a high quality DL system includes cases that are difficult or impossible to handle even for humans, being outside the validity domain.

4 Model-based Input Representation

We aim at generating inputs at the behavioural frontier of a DL system and we want them to be realistic and representative. Therefore, we adopt a model-based approach that produces test inputs starting from a model representation of the input domain and enforces the compliance with domain-specific constraints. This may require the transformation of a concrete input into an abstract model that can be manipulated by the exploration algorithm, in case no domain specific model of the input is available. The transformation from models to concrete inputs is instead always required.

To illustrate how our approach works in practice, we consider both an exemplary classification problem and a regression problem. The classification problem consists of handwritten digit recognition, while the regression problem is steering angle prediction for self-driving cars. In the latter case, we focus on systems that perform behavioural cloning, i.e. the DL component learns the lane keeping behaviour from a human driver [7]. In detail, the DL system is able to autonomously keep the lane since it contains a DNN that is trained with images captured by the camera sensors of the car, paired with the steering angles provided by a driver.

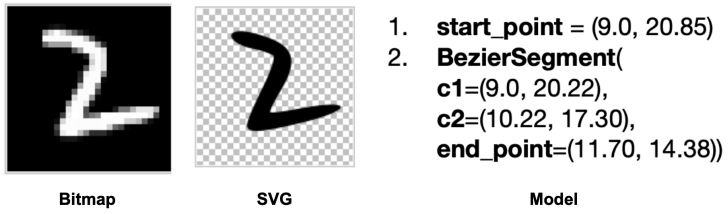

Figure 2: Bitmap and vector image; model representation of the image returned by Potrace

4.1 Image Classification

We use the inputs available from the MNIST database [28] and originally encoded as 28 x 28 images [28], with greyscale levels that range from 0 to 255. We adopt Scalable Vector Graphics (SVG)[1] as their model representation. SVG is an XML-based vector image format for two-dimensional graphics that can represent shapes as the combination of cubic and quadratic Bézier curves [20]. By modelling handwritten digits as a combination of Bézier curves, we ensure that the smoothness and curvature of handwritten shapes is preserved and that images remain realistic even after (minor) manipulation of the Bézier curve parameters.

To transform an original input image into its SVG model representation, we use the Potrace algorithm [44]. This algorithm performs a sequence of operations, including binarisation, despeckling and smoothing, to produce a smooth vector image starting from a bitmap. Figure 2 shows an MNIST image paired with its SVG model and its description. The control parameters that determine the shape of the modelled digit are: the start point, the end point and the control points *c1* and *c2* that define each Bézier segment.

In the other direction, we use rasterisation to transform a vector model into a 28 x 28 grayscale image. This operation exploits the functionality offered by LibRsvg[2] and Cairo[3], two popular open source graphic libraries.

4.2 Steering Angle Prediction

We consider a self-driving car that is trained and tested in the BeamNG [5] simulation environment. It features an accurate driving physics engine and it is freely available and research-oriented.

The input to the steering angle predictor is an image captured by the onboard sensor camera in the simulated environment. Therefore, the test input is determined by the scenario in which the car drives. Such simulated scenario can be modelled as the composition of the roads, the driving task (i.e., start point, end point and lane to keep), and the environment, which includes the weather and lightness conditions.

For the sake of simplicity, let us consider scenarios consisting of single plain asphalt roads surrounded by green grass on which the car has to drive keeping the right lane. The environment is always set to a clear day without fog. The roads are composed of two lanes with fixed width in which there is a yellow center line plus two white lines that separate each lane from the non-drivable area.

Abstractedly, a road can be represented as a sequence of contiguous points in a bi-dimensional space (assuming constant elevation), constrained to fall within a square bounding box of fixed size. To produce a smooth and realistic shape for the road being modelled, we use Catmull-Rom cubic splines [8] and then we interpolate such curves to obtain the 2D point sequence. Figure 3 shows the splines that define a road as well as its interpolated 2D points (marked as grey dots). The control parameters that determine the shape of the splines in Figure 3 are the coordinates of the control points of the center line spline (marked as larger red dots).

The concrete representation of the driving scenario is strictly dependent on the simulator. BeamNG exposes an intuitive API for programmatically configuring virtual roads and controlling the simulations[4]. In BeamNG, a scenario is described by a JSON file that contains the set of points to render the roads. The simulation engine renders the road by creating polygons starting from the points provided in the scenario description.

To transform the abstract model into the road to be rendered in the simulator, we calculate its points by exploiting the recursive algorithm for the evaluation of Catmull-Rom cubic splines proposed by Barry

[1] https://www.w3.org/Graphics/SVG/
[2] https://wiki.gnome.org/Projects/LibRsvg
[3] https://www.cairographics.org
[4] https://github.com/BeamNG/BeamNGpy

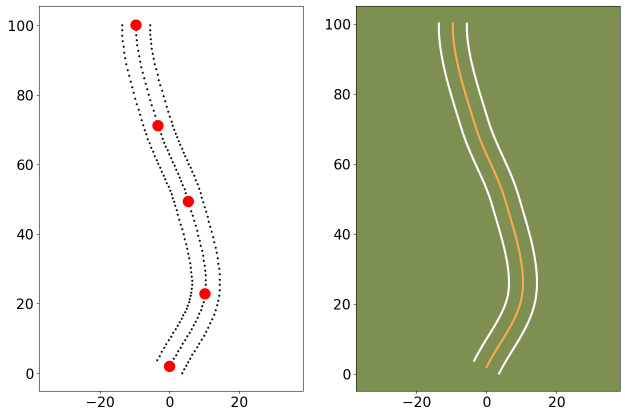

Figure 3: The model of a road and the corresponding road

and Goldman [4] and the functionality offered by the Shapely library for manipulation and analysis of planar geometric objects[5]. We also enforce the domain specific constraint of avoiding the generation of self-intersecting roads.

5 The DEEPJANUS Technique

DEEPJANUS explores the behavioural space of a DL system to find pairs of inputs at its frontier: one input on which the DL system behaves as expected, and another similar input on which it misbehaves. DEEPJANUS aims at exploring the frontier at large, i.e., as thoroughly as possible, so as to report a broad picture of the boundary behaviours to developers.

To perform such exploration, it aims at producing inputs at the frontier of behaviours and at maximising the diversity among the elements that are moved toward the frontier, so as to achieve thorough frontier exploration. At the same time, it also maintains high similarity within each pair of inputs crossing the frontier. Therefore, the problem solved by DEEPJANUS can be cast as a multi-objective search problem [19]. To obtain a diverse set of solutions, we hybridise traditional multi-objective search-based algorithms [12] with novelty search [36]. The idea is to measure the diversity between the population being evolved and the archive of the best individuals.

Algorithm 1 outlines the top level steps implemented in DEEPJANUS. Our algorithm is based on NSGA-II [12], a multi-objective evolutionary search algorithm quite popular in search-based software testing research [40, 49, 50, 34, 26], extended with: (1) hybridisation with novelty search, achieved by defining a fitness function that includes a measure of sparseness of the solutions (see Section 5.1.1); (2) use of an archive, to avoid cycling and to promote frontier exploration at large (lines 5 and 15 of Algorithm 1); (3) use of re-population, to escape from stagnation (line 13 of Algorithm 1). Moreover, we defined domain specific mutation operators to evolve the candidate solutions.

We implemented DEEPJANUS in Python on top of the DEAP evolutionary computation framework (v. 1.3.0) [15]. The code of DEEPJANUS is available online as open source [46].

5.1 Fitness Functions

The algorithm optimises two fitness functions, which measure respectively the quality of an individual (consisting of its cross-pair diversity and within pair similarity) and the closeness of the inputs to the frontier of behaviours.

[5] https://github.com/Toblerity/Shapely

Algorithm 1: Overall algorithm of DEEPJANUS

Input : S: set of input seeds
g_{max}: max number of generations
$popsize$: population size
Output: A: archive of best individuals at the frontier

generation $g \leftarrow 0$;
$A \leftarrow \emptyset$;
population $P \leftarrow$ INITIALISEPOPULATION(S, $popsize$);
EVALUATE(P);
$A \leftarrow$ UPDATEARCHIVE(P);
`/* assign crowding distance to individuals */`
$P \leftarrow$ SELECT(P, $popsize$);
while $g < g_{max}$ **do**
  $g \leftarrow g + 1$;
  `/* Tournament selection based on dominance and crowding distance */`
  offspring $Q \leftarrow$ SELTOURDCD(P, $popsize$) ;
  foreach $q \in Q$ **do**
    $q \leftarrow$ MUTATE(q) ;
  end
  `// substitute the most dominated individuals`
  $P \leftarrow$ REPOPULATION(P, S, A);
  EVALUATE($P \cup Q$);
  $A \leftarrow$ UPDATEARCHIVE($P \cup Q$);
  $P \leftarrow$ SELECT($P \cup Q$, $popsize$);
end
return (A)

5.1.1 Quality of an individual

The quality of an individual is measured by two factors, namely (1) the distance between the two members of a pair and (2) the sparseness of an individual with respect to the individuals in the archive measure the quality of a given individual. Since both are distances between inputs, we can combine these two measures additively into the fitness function f_1, to be maximised:

$$\max\ f_1(x) = \text{spars}(x, A) - k\ \text{dist}(x.m_1, x.m_2) \tag{1}$$

where A is the archive and $x.m_1, x.m_2$ are the members of the pair in the individual x. Both functions *spars* and *dist* report a measure of distance between inputs. Hence, the constant k is a pure number that can be safely set to 1. It can also be experimentally tuned (by decreasing it to values less than 1), to give more importance to the sparseness component of f_1, especially when preliminary runs of the algorithm show that the final archive contains a small number of individuals.

Function *dist* measures the similarity of the inputs within an individual as the distance between its members. This distance is computed on the input instances and is domain-specific. For the image classification problem, we compute the Euclidean distance between pixel matrices [43, 18]. For the regression problem, we use a weighted Levenshtein distance [31] that takes into account the edit operations on the sequences of angles and points sampled on the spines of the roads being compared. This distance metric is suitable for the comparison of road shapes, since it takes into account the order of the points along the curves as well as the relative angle between consecutive points. Function *spars* measures the sparseness of an individual, i.e., its minimum distance from the solutions in the archive A. spars(x) is defined as the distance from the closest individual in the archive A: $\min_{y \in A}$ dist(x, y). *dist*(x, y) is computed from the distances between individual members, as the minimum between (*dist*$(x.m_1, y.m_1)$ + *dist*$(x.m_2, y.m_2)$)/2 and (*dist*$(x.m_1, y.m_2)$ + *dist*$(x.m_2, y.m_1)$)/2.

5.1.2 Closeness to the frontier

The fitness function f_2 measures the closeness of an individual to the frontier:

$$\min\ f_2(x) = \begin{cases} \text{eval}(x.m_1) \cdot \text{eval}(x.m_2), & \text{if } > 0 \\ -1, & \text{otherwise} \end{cases} \tag{2}$$

Computation of f_2 requires the execution of the DL system under test on the two members belonging to the individual. This is represented by the invocation of function *eval* on each member. The quality of the behaviour exhibited by the DL system under test is measured during its execution. We design f_2 so that such quality is a positive number if the system exhibits the expected behaviour; a negative number otherwise.

The definition of function *eval* is domain/problem specific. For the image classification problem, we exploit the confidence level provided by the output layer of the DNN as *eval* function. In fact, the output of a classification DNN is usually the array returned by the softmax activation function containing the confidence levels assigned to each of the possible classes [17]. More specifically, *eval* is calculated as the difference between the confidence level associated with the expected label and the maximum confidence level associated to any other class.

For the steering angle prediction problem, we use a metric similar to that proposed by Gambi et al. [16]. The behaviour of the DNN is characterised by the distance of the car from the center of the lane during the simulation of the corresponding input scenario. More specifically, *eval* is calculated as $\min(w/2 - d)$, where w is the width of the lane and d the distance of the car from the lane centre. The position of the car is approximated by its centre of mass. Function *eval* returns its maximum value ($w/2$) when the car has distance zero from the center of the lane; it returns a negative number when an out of bound episode occurs.

5.2 Initial Population

Function INITIALISEPOPULATION (line 3 in Algorithm 1) returns the initial population given a set of seeds and the population size. Seeds are inputs on which the DL system under test exhibits a correct behaviour. The two members of an individual are obtained by copying the same seed twice and applying the mutation operator to one of the two copies.

More specifically, for image classification, seeds are chosen from the MNIST samples that are correctly classified by the system under test. For steering angle prediction, we generate valid roads and evaluate the system on them. The ones on which the car does not depart from the lane are considered as seeds (positive *eval*).

5.3 Archive of Solutions

The best (non dominated) individuals encountered during the search are kept in the archive [11]. This prevents the search for novelty from *cycling*, a phenomenon where the population moves from one area of the behavioural space to another and back again, without any memory of the areas it has already explored [37]. At the end of the exploration, the archive contains the final solution.

The archive is managed by the UPDATEARCHIVE function (lines 5 and 15 of Algorithm 1). An individual of the population is considered as a candidate to be included in the archive if it is at the frontier of behaviours. When a new input pair at the frontier is found, it is compared with its nearest neighbour in the archive. If the distance from the nearest neighbour in the archive is higher than a threshold t_a, the new individual is kept in the archive. Instead, if the distance from its nearest neighbour is lower than the threshold, the new candidate competes locally with its nearest neighbour in the archive. The local competition rewards individuals outperforming the most similar ones in the behavioural space [30]. In our case, local competition is based on the distance between the members of each pair: only the individual that has the closest members is kept in the archive.

The threshold t_a is a parameter that determines the granularity of the final frontier, so it can be adjusted by the tester so as to obtain a frontier with the desired size: a high value of t_a makes it difficult for new individuals at the frontier to enter the archive, because they must be extremely different from those already included. Lowering the value of t_a increases the granularity of the frontier and a higher number of similar individuals are added.

5.4 Selection Operator

We use the SELECT operator from NSGA-II (lines 6, 9 and 16 in Algorithm 1) [12]. This selection operator favours individuals with smaller non-domination rank and, when the rank is the same, i.e., the individuals belong to the same Pareto front, it favours the one with higher crowding distance (less dense regions) to promote diversity. Since we use tournament selection, the offspring of the current population is obtained

by choosing the winner among the (two) individuals being compared in each tournament (SELTOURDCD at line 9 in Algorithm 1).

5.5 Mutation

The individuals selected for the offspring are mutated by the MUTATE operator (lines 10:12 in Algorithm 1). This operator manipulates the control parameters of the model representation of each input: it chooses one of the two members of the individual and it applies a perturbation to the model parameters representing the input. The extent of the perturbation is uniformly sampled in a customisable range.

After applying the operator, we verify if the mutant complies with the constraints of the input domain. Moreover, we also verify that, once concretised into an actual input for the DL system, the mutant is different from its parent and from the other member of the pair. If any of these checks fails, the operator is applied repeatedly, until a valid input is obtained.

For the classification problem, the mutation operator randomly chooses a start point, an end point or a control point of the SVG model and applies a displacement to it in the two-dimensional space. The mutation operator for the regression problem is similar and it is applied to the control points that define the road shape.

5.6 Repopulation Operator

The exploration could get stuck in local optima, despite the mechanisms used to promote diversity (e.g., sparseness, in fitness function f_1). To mitigate this undesirable situation and further vary the population, DEEPJANUS uses the REPOPULATION operator (line 13 in Algorithm 1) inspired by the Shotgun hill climbing meta-heuristic algorithm [19]. It replaces n of the most dominated individuals in the current population with individuals newly generated from the seeds. The aggressiveness of this operator can be tuned by setting the range from which the value of n is uniformly sampled.

When repopulation is applied, each new individual is generated starting from one of the seeds that have not yet produced any solutions in the archive. If all the starting seeds have produced at least one solution, the new individuals are generated starting from a randomly chosen seed.

6 Experimental Evaluation

6.1 Subject Systems

We evaluate DEEPJANUS on two DL systems, addressing different tasks and domains. The first system performs a classification task, which consists of recognising handwritten digits from the MNIST dataset [28]. The second system solves a regression problem. Specifically, it predicts the steering angle of a self-driving car given the image of its onboard camera [7], using the BeamNG simulator [5]. Hereafter, we refer to such objects of study simply as *MNIST* and *BeamNG*. We chose these problems because of their representativeness, and because they have been widely used in the literature to evaluate testing techniques for DNN systems [18, 24, 43, 16]. Moreover, self-driving cars are an example of safety-critical usage of DNNs. To assess the usefulness of the frontier when computed for a high quality (HQ) vs a low quality (LQ) DL system, we trained two versions of each of the two considered systems.

The **MNIST** case study consists of a DNN model that predicts which digit is represented by an input image. We considered the deep convolutional network provided by Keras,[6] because of its popularity, simplicity and effectiveness. DEEPJANUS crosses the frontier of MNIST when the input image is misclassified. The HQ version of MNIST has 99.11% test accuracy and was obtained by training the DNN on the $60\,000$ images of the MNIST training set with the default settings provided by Keras, i.e., 12 epochs, with batches of size 128, and with a learning rate equal to 1. The LQ version was trained on the same training set and with the same hyper-parameters, with the exception of the learning rate that was set to 0.001. In fact, such a learning rate is associated with an accuracy drop, i.e., the model's test accuracy goes down to 84.34% .

The **BeamNG** case study is a self-driving car equipped with a Lane Keeping Assist System (LKAS) running in the BeamNG simulator [5]. It adopts a *behavioural reflex* approach, i.e., the DL component learns a direct mapping from the sensor camera input to the steering angle value to be passed to the actuators [9].

[6] https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py

The DNN driving the BeamNG ego-car utilises the DAVE-2 architecture designed by Bojarski et al. at NVIDIA, consisting of three CNNs, followed by five fully-connected layers [7]. The DNN was trained with images captured by the camera sensors of the ego-car, paired with the steering angles provided by the simulator's autopilot, which takes advantage of global knowledge and computes the optimal steering angle geometrically. DEEPJANUS crosses the frontier of BeamNG if the ego-car goes out of bound when driving in the input road. For BeamNG we also produced two versions, HQ and LQ. Both versions were trained for $4,600$ epochs, with batches of size 128 and with a learning rate equal to 0.001. To train the LQ version, we used a training dataset obtained by letting the autopilot drive up to 15 mph on a sinusoidal road ($y = sin(x/10) \times 10$, where 1 unit corresponds to 1 meter). The HQ version was instead trained on an enriched training set, including 30 diverse types of roads made of 20 control points that were automatically generated. Unrepresentative training data is a common fault in DL systems [22].

6.2 Research Questions

RQ1 (Effectiveness): *What is the intersection between the frontier reported by* DEEPJANUS *and the input validity domain of the DL system under test?*

This is the main research question of our empirical evaluation, since it focuses on the use of DEEPJANUS to check if the frontier behaviours of the DL system under test are outside the validity domain, which indicates the system has reached an adequate quality level, or within the validity domain, which points to issues that might affect the DL system in real executions.

Metrics: Assessing whether a frontier input is or is not part of the valid inputs is in general a domain dependent task, which requires human judgment and deep knowledge of the requirements behind the DL system. For MNIST, we resort to a human study in order to understand if and to what extent a given digit is recognisable correctly by a human. When this is not the case, the input image is deemed as outside the validity domain of the classifier. For BeamNG, we refer to the guidelines from the American Association of State Highway and Transportation Officials (AASHTO) [39], which prescribes among other things the minimum recommended radius of curvature by speed limit (in particular, 47 ft at 15 mph, the car speed in the simulations run on BeamNG). When the generated inputs violate the road design guidelines on the minimum curvature radius we deem the frontier input as outside the validity domain.

Human assessment: To determine whether the frontier of MNIST intersects its validity domain, we asked humans to recognise the digit images taken from the frontier and to declare their confidence in such recognition. We created 20 surveys made of 10 questions to be presented to human assessors: 9 assessment questions (ASQ) and 1 attention check question (ACQ). The difference between ASQ and ACQ is that the former involves a frontier image while the latter is by construction a nominal image whose classification is trivial and unambiguous for humans. To this aim, we used the digits delivered with the Freestyle Script font,[7] a computer font that resembles handwritten characters. We double checked manually that the ASQs were indeed unambiguous to classify. To obtain the images featured in the ASQs, we ran DEEPJANUS on both versions of MNIST (HQ and LQ), once for each digit class. Then, we considered the member of the pair outside the frontier (i.e., the misclassified image). We took 90 images for each of the two MNIST frontiers. We sorted the elements in the frontier based on their distance from the corresponding nominal Freestyle-font digit (the same image used in the ACQ) and divided them into 9 buckets of the same size. We selected the same number of samples (9) for each digit by randomly selecting one sample from each bucket. In total, we selected 180 inputs from the obtained frontiers. Each of the selected inputs appeared in a single survey. The order of appearance of HQ ASQs, LQ ASQs and the ACQ was randomised in the surveys. In total, this assessment involved 20 different human assessors, one for each survey.

For each image, we asked humans to answer the following questions: (1) *"What digit does the image represent?" (0, 1, 2, . . ., 9)*; (2) *"How confident are you in your answer?" (-2: not at all, -1: not much, 0: borderline, 1: quite confident, 2: very confident)*.

RQ2 (Discrimination): *Does* DEEPJANUS *provide discriminative information about the DL systems under test?*

In this research question, we compare the frontiers of two DL systems, one exhibiting good performance (HQ) and one having poor performance (LQ). We measured the size of the region identified by each frontier, to assess whether HQ systems have a larger frontier than LQ ones.

Metrics: To answer this research question quantitatively, we defined the metric *radius*: let us consider the archive A at the end of the execution of Algorithm 1 and let us assume that each individual $x \in A$ stores the input on which the DL system misbehaves in its second member $x.m_2$. We define the outer frontier of misbehaviours as $S_{out} = \{x.m_2 | x \in A\}$ and the inner one as $S_{in} = \{x.m_1 | x \in A\}$. The radius measures the

[7] https://www.fonts.com/font/itc/freestyle-script

average distance of inputs in the frontier from the reference input Ω, an elementary, nominal input that the system is expected to handle correctly by the requirements:

$$\text{radius}(S) = \frac{\sum_{m \in S} \text{dist}(m, \Omega)}{|S|} \tag{3}$$

For MNIST, we considered as reference image Ω the corresponding digit in Freestyle Script font, converted to the same format as the MNIST dataset. For BeamNG, the reference sample is a straight road with no curves.

We also evaluated HQ's vs LQ's frontiers qualitatively, by involving humans in a survey, in which they performed pairwise comparisons between images taken from the two frontiers (i.e., one image from HQ's and one from LQ's frontier).

Human assessment: For the qualitative assessment of HQ's vs LQ's frontiers, we provided human evaluators with two images taken respectively from each of the two frontiers. For MNIST, we asked them to decide which of the two digit images is easier to recognise. For BeamNG, we asked them to decide which of the two roads is easier to drive. Each pair of inputs was assessed by two human evaluators. Then, we measured the number of answers in which both subjects agreed in considering as easier to recognise/drive the element on the frontier of HQ (resp. LQ). This would support our conjecture that manual inspection of the frontier is an effective way to discriminate between good and poor performance DL systems. For each system, we published 10 surveys made of 10 questions: 9 discriminative questions (DSQ) and 1 attention check question (ACQ), consisting of a pair of inputs for which the human choice is obvious and completely predictable. Each survey, made of 10 questions, was answered by 2 evaluators. In total, this assessment involved 40 different evaluators, i.e., 20 for each system.

RQ3 (Comparison): *Is* DEEPJANUS *able to characterise the frontier of the behaviours better than the state of the art tool* DLFUZZ*?*

DLFUZZ [18] is a state of the art tool for the generation of boundary values by means of fuzzing. Among the techniques proposed in the literature, DLFuzz is the most related to DEEPJANUS since it produces boundary inputs by manipulation of existing seeds. DLFuzz uses gradient ascent optimisation to maximise a custom loss function that takes also into account the distance between the new input and the seed. It should be noticed that DLFuzz is not a model-based input generator as it operates directly on the raw input (i.e., image pixels). Hence, its boundary inputs are supposedly less realistic and less representative than DEEPJANUS'.

Metrics: We compare the radius, as defined above, of DEEPJANUS' frontier w.r.t. DLFuzz's boundary inputs.

6.3 Experimental Procedure

Our experimental procedure consists of: (1) generation of the frontiers for the DL systems under test (MNIST HQ/LQ; BeamNG HQ/LQ) and computation of the radius for the generated frontiers; (2) generation of boundary inputs using DLFuzz and comparison with DEEPJANUS's frontiers; (3) human assessment of the inputs in the frontiers, by means of two surveys: a digit recognition survey (for RQ1) and a pairwise image comparison survey (for RQ2).

Generation of the frontier with DEEPJANUS: We ran DEEPJANUS 10 times on each version of each system under test. At the end of the runs, we collected the values of the radius metric, as well as the representation of the input pairs that belong to the frontiers. The configurations of DEEPJANUS were obtained in a few preliminary runs and are reported in Table 1.

Before each run, we obtained a different set of initial seeds by sampling the inputs under the constraint that they have to produce a correct behaviour of the systems under test. For MNIST, each set of seeds was obtained by randomly selecting 100 correctly classified inputs from the MNIST test set, all belonging to the same class (digit "5"). Similar results have been obtained for digits other than five, but we do not report them for space reasons. For BeamNG, each initial set consisted of 12 valid seed roads on which the considered model was able to keep the lane. A seed road was defined by 10 control points in which the initial point was always at a fixed position whereas the others were placed at a random position 25 meters away from the previous one.

Generation of boundary inputs with DLFuzz: We generated boundary inputs for MNIST using DLFUZZ [18] starting from the same seeds used by DEEPJANUS. However, we could not consider BeamNG in the comparison, because DLFUZZ is not able to test a system in the simulation loop, on a sequence of images. It can only evaluate the output of the DNN component on a single, statically collected image,

Table 1: DeepJanus Configurations

Parameter	MNIST	BeamNG
population size	100	12
generations	4000	100
mutation lower bound	0.01	1
mutation upper bound	0.6	6
archive threshold t_a	4	35
repopulation upper bound	10	2
parameter k of fitness function f_1	0.1	0.01

Table 2: RQ1: Invalid inputs found at the frontier

Object	Validity	Number	Confidence
MNIST HQ	Valid	69	0.463 ± 1.255
	Invalid	21	-0.095 ± 1.338
MNIST LQ	Valid	82	1.524 ± 0.835
	Invalid	8	-0.875 ± 1.356
p-value		1.394E-2	2.48E-4
odds ratio		0.322	-
effect size		-	0.81 (large)
Object	**Validity**	**Number**	**Curv. radius (ft)**
BeamNG HQ	Valid	1	47.371
	Invalid	141	37.359 ± 4.262
BeamNG LQ	Valid	56	49.283 ± 1.966
	Invalid	173	40.419 ± 4.488
p-value		3.787E-12	2.52E-12
odds ratio		0.022	-
effect size		-	1.009 (large)

which is fuzzed by the tool. This means that it is not possible to simulate out of bound episodes, which would require a sequence of images to be fuzzed dynamically. We adopted the DLFUZZ configuration that is reported as the one achieving the best performance [18].

Human assessment of the frontier: We outsourced our surveys to a crowdsourcing platform in order to have a diverse pool of respondents [6]. *Crowdsourcing* has recently become quite popular in software engineering [33] to automate tasks that can only be performed by humans. A problem is specified in the form of small Human Intelligence Tasks (HITs) and made available in a crowdsourcing platform, where registered workers can choose to complete HITs for a small remuneration [41]. We selected the Amazon Mechanical Turk platform[8] for our two surveys (resp. for RQ1 and RQ2), because it is well known, well documented and widely used to gather qualitative feedbacks [25, 21].

We applied two methods to ensure the quality of the answers: (1) added an attention check question (ACQ) to each survey (see above); and (2) restricted the participation to workers with high reputation (above 95% approval rate) [42]. We only accepted answers from users that passed the ACQ.

7 Results

7.1 RQ1 (Effectiveness)

Table 2 reports the intersection between the input validity domain and the outer frontier of behaviours for each version of the considered systems. The upper part of the table reports the results for the MNIST system. As shown in the first two rows, out of the 90 images generated by DEEPJANUS on MNIST HQ, 69 are recognised by the crowdworkers as the digit classes to which the corresponding seeds belong, whereas

[8] https://www.mturk.com

21 are recognised incorrectly. On the other hand, on MNIST LQ DEEPJANUS generated 82 frontier inputs that are recognised correctly by crowdworkers. This indicates that the frontier of MNIST LQ has a larger intersection with the set of valid inputs than MNIST HQ. The classification performed by the crowdworkers was subjected to the Fisher's exact test [13] to determine the statistical significance of the effect of the version (HQ vs LQ) on the validity of the frontier inputs. The p-value lower than the usual threshold $\alpha = 0.05$ indicates statistical significance of the difference between MNIST HQ and MNIST LQ. The odds ratio indicates that the expected relative proportion of valid vs invalid is much lower in the MNIST HQ system.

The last column contains the confidence (mean $\pm$ standard deviation) expressed by the crowdworkers when classifying the images. The scale is between -2:+2 (with -2 = min confidence and +2 max confidence). Human assessors had a significantly higher confidence in classifying the valid inputs belonging to the frontier of MNIST LQ and were more uncertain when recognising valid inputs belonging to the frontier of MNIST HQ. This confirms that the frontier of a high quality DL system contains elements that are difficult to classify confidently even for a human. We assessed the statistical significance of the confidence comparison by applying Generalised Linear Modelling (GLM) [38]. The dependent variable of the GLM model is confidence, whereas the independent variable is a numeric encoding of the system (HQ = 0; LQ = 1). The results of the statistical test are a small p-value (way below $\alpha = 0.05$) and a large Cohen-d effect size (i.e., a large difference between the means, normalised by the pooled standard deviation; conventionally, the threshold for a large Cohen-d effect size is set to 0.8).

The lower part of the table shows the results for the BeamNG system. As reported in the third column, the self-driving car equipped with the HQ lane keeping assist system goes out of bound (misbehaves) only on one valid road from the frontier (indeed the minimum radius of curvature of this road is greater that the AASHTO threshold by just 0.371 feet). Instead, there is a significant number of valid frontier roads when the car is equipped with the LQ system. The fourth column reports the minimum radius of curvature (mean $\pm$ standard deviation). Its values for the roads on the HQ frontier are significantly lower than the radius values on the LQ frontier. This confirms that roads in the HQ frontier are substantially more difficult or even impossible to drive than those in the LQ frontier. Also for the results of BeamNG, we applied the Fisher's exact test and GLM to assess the statistical significance of the results. P-values indicate statistical significance of the difference between HQ and LQ, with a low odds ratio (valid/invalid proportion in HQ vs LQ) and a large Cohen-d effect size (large normalised difference between minimum curvature radii).

Summary: *Many elements at the frontier of behaviours identified by* DEEPJANUS *intersect the input validity domain when the quality of the system under test is low. For a high quality system, the valid inputs at the frontier are challenging to handle even for humans (MNIST case study) or are very close to being invalid (BeamNG case study).*

7.2 RQ2 (Discrimination)

The top rows of Table 3 show the inner/outer frontier radii returned by DEEPJANUS for the systems under test (mean $\pm$ standard deviation).

For each system under test, we compare the values of the frontier radii obtained for HQ vs LQ, to understand if the frontier volume can help discriminate between different quality levels. We assessed the statistical significance of this comparison by applying GLM, with radius as dependent variable and the quality of the system (HQ = 0; LQ = 1) as independent variable.

For MNIST, the radius for the LQ version of the system is significantly smaller than the one of the HQ version (with low p-value and large Cohen-d effect size). This means that, on average, the higher quality system tolerates larger changes to input images before exhibiting a misbehaviour. Similarly, for BeamNG the radius for the LQ version of the system is significantly smaller than the one of the HQ version (with low p-value and large/medium Cohen-d effect size resp. for inner/outer radius), showing again that the frontier can discriminate the HQ version from the LQ one.

Table 4 shows the results obtained from the crowdsourced survey on the discrimination of easier to recognise digits and easier to drive roads. In the LQ column, it reports the number of answers in which both crowdworkers comparing the same pair of images agreed in considering as qualitatively easier to classify/drive the input on the frontier produced by the LQ system. The HQ column reports the number of answers in which both subjects agreed in considering as qualitatively easier to classify/drive the input on the HQ frontier. The next column reports the number of cases in which the two subjects did not agree with each other.

Table 3: RQ2 (top, middle): discriminating HQ from LQ by DEEPJANUS (DJ)'s inner/outer radius and by DLFuzz (DLF); RQ3 (bottom): comparing DEEPJANUS vs DLFuzz

			Inner Radius	Outer Radius
DJ	MNIST	HQ	10.575 ± 0.188	10.597 ± 0.188
		LQ	10.284 ± 0.083	10.328 ± 0.078
		p-value	2.96E-4	5.61E-4
		effect size	1.99 (large)	1.87 (large)
	BeamNG	HQ	55.968 ± 1.522	57.236 ± 1.753
		LQ	52.853 ± 2.22	55.565 ± 2.412
		p-value	1.8E-2	9.34E-2
		effect size	1.636 (large)	0.792 (medium)
DLF	MNIST	HQ	-	9.889 ± 0.275
		LQ	-	9.976 ± 0.115
		p-value	-	0.385
		effect size	-	-0.398 (small)
DJ vs DLF (HQ)		p-value	-	4.14E-6
		effect size	-	2.905 (large)
DJ vs DLF (LQ)		p-value	-	4.25E-7
		effect size	-	3.441 (large)

Table 4: RQ2: human discrimination of LQ from HQ based on frontier inputs (MNIST: easier to recognise digit; BeamNG: easier to drive road)

Object	LQ	HQ	disagree	p-value	p-success
$MNIST$	77	4	19	¡ 2.2E-16	0.95
$BeamNG$	79	9	12	4.152E-15	0.89

Crowdworkers were able to determine the relative quality of the systems by looking at the inputs on their frontiers with very high accuracy. In fact, the large majority (95% for MNIST and 89% for BeamNG) of inputs found by DEEPJANUS on the outer frontier of the LQ system are perceived as qualitatively easier to classify/drive than those of the HQ systems. Statistical significance of the classification performed by the users was assessed by applying the Binomial exact test [10]. The returned p-values for MNIST and BeamNG are very low, showing that the choice between LQ frontier input and HQ frontier input as the easier to classify/drive is very unlikely to be random and uniform (i.e., 50%, 50%).

Summary: *The frontier of behaviours allows developers to discriminate a system with higher from one with lower quality: the radius is significantly larger for the former. A larger frontier means that the system is able to generalise correctly the learned behaviour to a larger set of valid input data. This is confirmed by the human study, where inputs from the larger frontier were deemed to be more difficult to handle than those from the smaller frontier.*

7.3 RQ3 (Comparison)

In order to address RQ_3, we compare the frontier identified by our approach with the boundary inputs returned by DLFuzz [18]. The middle part of Table 3 shows the radius of the outer boundary inputs returned by DLFuzz for the MNIST classifier (inner boundaries are not returned by DLFuzz). We can notice that, differently from the frontier produced by DEEPJANUS, the boundary inputs of DLFuzz do not exhibit any statistically significant difference between HQ and LQ's frontier radius (high p-value, > 0.05). Actually, the difference between the radii seems to go in the opposite direction than in the case of DEEPJANUS (negative, small effect size). This indicates that DLFuzz cannot reliably discriminate HQ from LQ by generating a larger frontier for the former than for the latter.

As reported in the lower part of Table 3, for both LQ and HQ the radius of the frontiers identified by DLFuzz is always significantly smaller than the radius of DEEPJANUS. This indicates that DEEPJANUS can explore

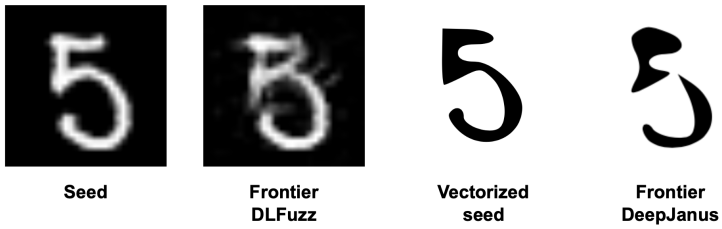

Figure 4: Seed and frontier images generated respectively by DEEPJANUS and DLFuzz (the starting seed is the same)

the frontier more thoroughly than DLFuzz.

Indeed, DLFuzz applies small perturbations to the pixels of the input image that cause the classifier to fail, but it does not attempt to explore the input space thoroughly by promoting diversity when new inputs are generated (the only way to promote diversity in DLFuzz is by selecting different initial seeds at each run, which we did in our experiments). On the contrary, DEEPJANUS is guided by a fitness function (f_1) that accounts explicitly for the sparseness of the generated inputs, hence exploring the behavioural space more thoroughly. As a consequence, the inputs generated by DLFuzz are closer to the reference than the ones produced by DEEPJANUS.

From a qualitative point of view, the boundary inputs found by DLFuzz are quite different from those found by DEEPJANUS, as apparent from Fig. 4. In fact, the same seed is manipulated as a shape of calligraphic traits modelled in SVG by DEEPJANUS (two rightmost pictures in Fig. 4), while raw pixel manipulation is performed by DLFuzz (leftmost pictures). The images generated by DEEPJANUS simulate quite realistically the traits that can be found in a hand-written digit, while DLFuzz's images look like blurred, noisy versions of the original ones.

Summary: DEEPJANUS *explores a significantly larger frontier than DLFuzz. The images produced by* DEEPJANUS *for MNIST look more realistic than those of DLFuzz.*

7.4 Threats to Validity

Construct Validity: The choice of the reference input, Ω, for the computation of the radius in a given domain is conventional. We have tried different choices of Ω in both domains (MNIST and BeamNG) and found that the experimental results reported in Section 7 are not sensitive to the specific choice of Ω: the same conclusions were drawn when a different choice of Ω was made.

External Validity: The choice of subject DL systems is a possible threat to the *external validity*. To mitigate this threat, we chose two diverse DL systems. One is a DNN that solves a classification problem, while the other is a self-driving car equipped with a DNN component that solves a regression problem. However, our results might not generalise to other DL systems and further studies with a wider set of systems should be carried out to fully assess the generalisability of our findings.

To ensure **Reproducibility** of our results, the source code of DEEPJANUS, our objects, and the experimental data are available online [46], making the evaluation repeatable.

8 Related Work

Raw Input Data Manipulation: Several works generate test inputs for DL systems by applying perturbations to the available training/test data. These approaches aim at generating inputs that trigger inconsistencies between multiple DL systems [43], or between the original and a transformed test input [18, 45, 51]. They require white-box access to the activation levels of the DNN, if they are guided by coverage criteria such as neuron coverage [43] or the more fine grained k-multisection neuron coverage [32].

One of the limitations of these approaches is the lack of realism of the generated inputs. While these corrupted images are useful for security testing as adversarial attacks, they are not representative of data captured by sensors of a real DL system. To generate realistic inputs we adopted a model-based approach, which ensures that data are generated only within the constraints of the model. Another limitation of the existing input generators is that they do not attempt to delimit the region of misbehaviour but they

just provide a way to sample it, regardless of the distance from the region of nominal behaviour. On the contrary, we sample the entire frontier of misbehaviours as thoroughly as possible.

Kim et al. [24] designed a test adequacy criterion, named *surprise adequacy*, to capture the novelty of the input with respect to the training data. However, surprise adequacy has been used for test case selection and retraining but not for test input generation. Moreover, a surprise adequate test set is not necessarily one that covers also our frontier of misbehaviours, so we view the two approaches as complementary.

Model-based Input Generation: Abdessalem et al. [1, 3, 2] combine genetic algorithms and machine learning to test advanced driver-assistance systems in an industrial setting. Gambi et al. [16] propose ASFAULT, a search-based approach to test the lane-keeping system of self-driving cars. The goal of these techniques is to generate extreme and challenging scenarios, maximising the number of detected system failures. DEEPJANUS is also model-based, but it differs from existing approaches because it aims at reaching the frontier without surpassing it and because it spreads the generated inputs along the entire frontier of behaviours. Its output is not just a set of critical inputs: it is also a thorough characterisation of the frontier of behaviours.

9 Conclusions and Future Work

DEEPJANUS characterises the quality of a DL system as its frontier of behaviours, i.e., pairs of similar inputs that trigger different behaviours of the system and are far from each other. Experimental results show that frontier inputs provide the developers with both quantitative and qualitative information, useful to assess the quality of a DL system and to identify valid inputs that it cannot handle properly. Our empirical study shows that DEEPJANUS is more effective in characterising the frontier of behaviours than the state of the art tool, DLFuzz. In our future work, we plan to extend the validity of our results by considering a wider sample of DL systems with increasingly complex input domain, including industrial ones.

References

[1] Raja Ben Abdessalem, Shiva Nejati, Lionel C. Briand, and Thomas Stifter. Testing advanced driver assistance systems using multi-objective search and neural networks. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, ASE*, pages 63–74, 2016.

[2] Raja Ben Abdessalem, Shiva Nejati, Lionel C. Briand, and Thomas Stifter. Testing vision-based control systems using learnable evolutionary algorithms. In *Proceedings of the 40th International Conference on Software Engineering*, ICSE '18, pages 1016–1026, New York, NY, USA, 2018. ACM.

[3] Raja Ben Abdessalem, Annibale Panichella, Shiva Nejati, Lionel C. Briand, and Thomas Stifter. Testing autonomous cars for feature interaction failures using many-objective search. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE 2018, pages 143–154, New York, NY, USA, 2018. ACM.

[4] Phillip J. Barry and Ronald N. Goldman. A recursive evaluation algorithm for a class of catmull-rom splines. *SIGGRAPH Comput. Graph.*, 22(4):199–204, June 1988.

[5] BeamNG GmbH. BeamNG.research.

[6] Tara S. Behrend, David J. Sharek, Adam W. Meade, and Eric N. Wiebe. The viability of crowdsourcing for survey research. *Behavior Research Methods*, 43(3):800, Mar 2011.

[7] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016.

[8] Edwin Catmull and Raphael Rom. A class of local interpolating splines. In R. E. Barnhill and R. F. Riesenfeld, editors, *Computer Aided Geometric Design*, pages 317 – 326. Academic Press, 1974.

[9] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015.

[10] C. J. Clopper and E. S. Pearson. The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial. *Biometrika*, 26(4):404–413, 12 1934.

[11] Edwin D. de Jong. The incremental pareto-coevolution archive. In Kalyanmoy Deb, editor, *Genetic and Evolutionary Computation – GECCO 2004*, pages 525–536, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[13] Ronald Aylmer Fisher. *Statistical methods for research workers*. Genesis Publishing Pvt Ltd, 2006.

[14] International Organization for Standardization (ISO). Iso/pas 21448: Road vehicles – safety of the intended functionality, 2019.

[15] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner Gardner, Marc Parizeau, and Christian Gagné. Deap: Evolutionary algorithms made easy. *J. Mach. Learn. Res.*, 13(1):2171–2175, July 2012.

[16] Alessio Gambi, Marc Müller, and Gordon Fraser. Automatically testing self-driving cars with search-based procedural content generation. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA*, pages 318–328, 2019.

[17] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

[18] Jianmin Guo, Yu Jiang, Yue Zhao, Quan Chen, and Jiaguang Sun. Dlfuzz: differential fuzzing testing of deep learning systems. In *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE*, pages 739–743, 2018.

[19] Mark Harman, S. Afshin Mansouri, and Yuanyuan Zhang. Search-based software engineering: Trends, techniques and applications. *ACM Comput. Surv.*, 45(1):11:1–11:61, December 2012.

[20] M. Hazewinkel. *Encyclopaedia of Mathematics: Supplement*. Number v. 1 in Encyclopaedia of Mathematics. Springer Netherlands, 1997.

[21] Jeffrey Heer and Michael Bostock. Crowdsourcing graphical perception: Using mechanical turk to assess visualization design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 203–212, New York, NY, USA, 2010. ACM.

[22] Nargiz Humbatova, Gunel Jahangirova, Gabriele Bavota, Vincenzo Riccio, Andrea Stocco, and Paolo Tonella. Taxonomy of real faults in deep learning systems. In *Proceedings of 42nd International Conference on Software Engineering*, ICSE '20, page 12 pages. ACM, 2020.

[23] ISO. Road vehicles – Functional safety, 2011.

[24] Jinhan Kim, Robert Feldt, and Shin Yoo. Guiding deep learning system testing using surprise adequacy. In *Proceedings of the 41st International Conference on Software Engineering, ICSE*, pages 1039–1049, 2019.

[25] Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 453–456, New York, NY, USA, 2008. ACM.

[26] Kiran Lakhotia, Mark Harman, and Phil McMinn. A multi-objective approach to search-based test data generation. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, pages 1098–1105, New York, NY, USA, 2007. ACM.

[27] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design*. Prentice Hall, 1997.

[28] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[29] Joel Lehman and Kenneth O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223, 2011.

[30] Joel Lehman and Kenneth O. Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, pages 211–218, New York, NY, USA, 2011. ACM.

[31] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[32] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, Jianjun Zhao, and Yadong Wang. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE 2018, pages 120–131, New York, NY, USA, 2018. ACM.

[33] Ke Mao, Licia Capra, Mark Harman, and Yue Jia. A survey of the use of crowdsourcing in software engineering. *Journal of Systems and Software*, 126:57–84, 2017.

[34] Ke Mao, Mark Harman, and Yue Jia. Sapienz: Multi-objective automated testing for android applications. In *Proceedings of the 25th International Symposium on Software Testing and Analysis*, ISSTA 2016, pages 94–105, New York, NY, USA, 2016. ACM.

[35] B. Marculescu, R. Feldt, and R. Torkar. Using exploration focused techniques to augment search-based software testing: An experimental evaluation. In *2016 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, pages 69–79, April 2016.

[36] Jean-Baptiste Mouret. Novelty-based multiobjectivization. In Stéphane Doncieux, Nicolas Bredèche, and Jean-Baptiste Mouret, editors, *New Horizons in Evolutionary Robotics*, pages 139–154, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[37] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites, 2015.

[38] J. A. Nelder and R. W. M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384, 1972.

[39] American Association of State Highway and Transportation Officials. *AASHTO Green Book (GDHS-7) - A Policy on Geometric Design of Highways and Streets*. The Green Book. American Association of State Highway and Transportation Officials, 2018.

[40] Annibale Panichella, Fitsum Meshesha Kifetew, and Paolo Tonella. Automated test case generation as a many-objective optimisation problem with dynamic selection of the targets. *IEEE Transactions on Software Engineering*, 44(2):122–158, 2018.

[41] F. Pastore, L. Mariani, and G. Fraser. Crowdoracles: Can the crowd solve the oracle problem? In *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*, pages 342–351, March 2013.

[42] Eyal Peer, Joachim Vosgerau, and Alessandro Acquisti. Reputation as a sufficient condition for data quality on amazon mechanical turk. *Behavior Research Methods*, 46(4):1023–1031, Dec 2014.

[43] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 1–18, 2017.

[44] P. Selinger. Potrace: a polygon-based tracing algorithm. 2003.

[45] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th International Conference on Software Engineering*, ICSE '18, pages 303–314, New York, NY, USA, 2018. ACM.

[46] DEEPJANUS: A tool for model-based exploration of the frontier of behaviours for deep learning systems testing. `https://github.com/fse2020submission/deepjanus`, 2019.

[47] Unity Technologies. Unity, 2019.

[48] Matthew Wicker, Xiaowei Huang, and Marta Kwiatkowska. Feature-guided black-box safety testing of deep neural networks. In *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS*, pages 408–426, 2018.

[49] Shin Yoo and Mark Harman. Pareto efficient multi-objective test case selection. In *Proceedings of the 2007 International Symposium on Software Testing and Analysis*, ISSTA '07, pages 140–150, New York, NY, USA, 2007. ACM.

[50] Shin Yoo and Mark Harman. Using hybrid algorithm for pareto efficient multi-objective test suite minimisation. *Journal of Systems and Software*, 83(4):689 – 701, 2010.

[51] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE*, pages 132–142, 2018.