Guiding Deep Learning System Testing using Surprise Adequacy

Jinhan Kim
School of Computing
KAIST
Daejeon, Republic of Korea
jinhankim@kaist.ac.kr

Robert Feldt
Dept. of Computer Science and Engineering
Chalmers University
Gothenburg, Sweden
robert.feldt@chalmers.se

Shin Yoo
School of Computing
KAIST
Daejeon, Republic of Korea
shin.yoo@kaist.ac.kr

***Abstract*—Deep Learning (DL) systems are rapidly being adopted in safety and security critical domains, urgently calling for ways to test their correctness and robustness. Testing of DL systems has traditionally relied on manual collection and labelling of data. Recently, a number of coverage criteria based on neuron activation values have been proposed. These criteria essentially count the number of neurons whose activation during the execution of a DL system satisfied certain properties, such as being above predefined thresholds. However, existing coverage criteria are not sufficiently fine grained to capture subtle behaviours exhibited by DL systems. Moreover, evaluations have focused on showing correlation between adversarial examples and proposed criteria rather than evaluating and guiding their use for actual testing of DL systems. We propose a novel test adequacy criterion for testing of DL systems, called Surprise Adequacy for Deep Learning Systems (SADL), which is based on the behaviour of DL systems with respect to their training data. We measure the surprise of an input as the difference in DL system's behaviour between the input and the training data (i.e., what was learnt during training), and subsequently develop this as an adequacy criterion: a good test input should be sufficiently but not overtly surprising compared to training data. Empirical evaluation using a range of DL systems from simple image classifiers to autonomous driving car platforms shows that systematic sampling of inputs based on their surprise can improve classification accuracy of DL systems against adversarial examples by up to 77.5% via retraining.**

***Index Terms*—Test Adequacy, Deep Learning Systems**

I. Introduction

Deep Learning (DL) [24] systems have achieved significant progress in many domains including image recognition [13], [22], [38], speech recognition [17], and machine translation [20], [37]. Based on their capability to match or even surpass human performance, DL systems are increasingly being adopted as part of larger systems in both safety and security critical domains such as autonomous driving [6], [10], and malware detection [12].

Such adoption of DL systems calls for new challenges, as it is critically important that these larger systems are both correct and predictable. Despite their impressive experimental performances, DL systems are known to exhibit unexpected behaviours under certain circumstances. For example, in a reported incident, an autonomous driving vehicle expected another vehicle to yield in one of the rarer circumstances, and crashed into the other vehicle when the expectation proved incorrect [3]. There is an urgent need to verify and validate behaviours of DL systems. However, a significant part of existing software testing technique is not directly applicable to DL systems. Most notably, traditional white-box testing techniques that aim to increase structural coverage [4] is not very useful for DL systems, as their behaviour is not explicitly encoded in their control flow structures.

A number of novel approaches towards testing and verification of DL systems have been recently proposed to fill in the gap [19], [27], [34], [40]. Most of these techniques share two assumptions. The first assumption is essentially a generalisation of the essence of metamorphic testing [11]: if two inputs to a DL system are *similar* with respect to some human sense, the outputs should also be similar. For example, DeepTest [40] checks whether an autonomous driving system behaves in the same way when the input image is transformed as if the same scene is under a different weather condition. The second assumption, also based in more traditional software testing results [15], is that the more diverse a set of input is, the more effective testing of a DL system one can perform. For example, DeepXplore [34] presented the Neuron Coverage (the ratio of neurons whose activation values were above a predefined threshold) as the measure of diversity of neuron behaviour, and subsequently showed that inputs violating the first assumption will also increase the neuron coverage.

While the recently introduced techniques have made significant advances over manual *ad hoc* testing of DL systems, there is a major limitation. The coverage criteria proposed so far are not sufficiently fine grained, in a sense that all of them simply count neurons whose activation values satisfy certain conditions. While this aggregation by counting does allow the tester to quantify the test effectiveness of a given input set, it conveys little information about individual inputs. For example, it is not immediately clear when an input with higher NC should be considered *better* than another with lower NC, and why: certain inputs may naturally activate more neurons above the threshold than others, and vice versa. Another example is the k-Multisection Neuron Coverage [27], which partitions the ranges of activation values of neurons, observed during training, into k buckets, and count the number of total buckets covered by a set of inputs. When measured for a single input, the coverage will be either $\frac{1}{k}$ if the input activates each neuron with a value from one of the k buckets, or smaller than that if some neurons activate outside the range observed during training. Again, the information about how far such activations go beyond observed range is lost during aggregation, making it hard to evaluate the relative value of each input. For a test adequacy criterion to be practically useful, it should be able to guide the selection of individual inputs, eventually resulting in improvements of the accuracy of the DL system under investigation.

To overcome these limitations, we propose a new test adequacy for DL systems, called Surprise Adequacy for DL systems (SADL). Intuitively, a good test input set for a DL system should be systematically diversified to include inputs ranging from those similar to training data to those significantly different and adversarial.[1] At individual input granularity, SADL measures how *surprising* the input is to a DL system with respect to the data the system was trained with: the actual measure of surprise can be either based on the likelihood of the system having seen a similar input during training (here with respect to probability density distributions extrapolated from the training process using kernel density estimation [41]), or the distance between vectors representing the neuron activation traces of the given input and the training data (here simply using Euclidean distance). Subsequently, the Surprise Adequacy (SA) of a set of test inputs is measured by the range of individual surprise values the set covers. We show that SADL is sufficiently fine rained by training adversarial example classifiers based on SADL values that can produce higher accuracy compared to the state of the art. We also show that sampling inputs according to SADL for retraining DL systems can result in higher accuracy, thus showing that SADL is an independent variable that can positively affect the effectiveness of DL system testing.

The technical contributions of this paper are as follows:

- We propose SADL, a fine grained test adequacy metric that measures the surprise of an input, i.e., the difference in the behaviour of a DL system between a given input and the training data. Two concrete instances of SADL are proposed based on different ways to quantify surprise. Both are shown to be correlated with existing coverage criteria for DL systems.
- We show that SADL is sufficiently fine grained in capturing the behaviour of DL systems by training a highly accurate adversarial example classifier. Our adversarial example classifier shows as much as 100% and 94.53% ROC-AUC score when applied to MNIST [25] and CIFAR-10 [21] dataset, respectively.
- We show that SADL metrics can be used to sample effective test input sets. When retraining DL systems using additional adversarial examples, sampling additional inputs with broader SA values can improve the accuracy after retraining by up to 77.5%.
- We undertake all our experiments using publicly available DL systems ranging from small benchmarks (MNIST and CIFAR-10) to a large system for autonomous driving vehicles (Dave-2 [6] and Chauffeur [1]).

The remaining of this paper is organised as follows. Section II introduces Surprise Adequacy for DL systems, SADL: two variants of SADL are presented along with algorithms that measure them. Section III sets out the research questions and Section IV describes the experimental set-up of the empirical evaluations. Section V presents the results from empirical evaluations. Section VI addresses threats to validity. Section VII presents related work, and Section VIII concludes.

II. Surprise Adequacy for Deep Learning Systems

All existing test adequacy criteria for DL systems aim to measure the diversity of an input set. Neuron Coverage (NC) [34] posits that the higher the number of neurons that are activated above a predefined threshold, the more diverse input the DL system has been executed with. DeepGauge [27] proposed a range of finer grained adequacy criteria including k-Multisection Neuron Coverage, which measures the ratio of activation value *buckets* that have been covered across all neurons, and Neuron Boundary Coverage, which measures the ratio of neurons that are activated beyond the ranges observed during training.

We argue that diversity in testing of DL systems is more meaningful when it is measured with respect to the training data, as DL systems are likely to be more error prone for inputs that are unfamiliar, i.e., diverse. Furthermore, while neuron activation above thresholds, or beyond observed ranges, may be closely related to diversity of the given input, they do not measure to what degree the activations of the network for one input differs from the activations for another input. They are fundamentally discretisations and do not utilize the fact that neuron activations are continuous quantities. In contrast, our aim is to define an adequacy criterion that quantitatively measures behavioural differences observed in a given set of inputs, relative to the training data.

A. Activation Trace and Surprise Adequacy

Let $\mathbf{N} = \{n_1, n_2, \ldots\}$ be a set of neurons that constitutes a DL system $\mathbf{D}$, and let $X = \{x_1, x_2, \ldots\}$ be a set of inputs. We denote the activation value of a single neuron n with respect to an input x as $\alpha_n(x)$. For an ordered (sub)set of neurons, let $N \subseteq \mathbf{N}$, $\alpha_N(x)$ denote a vector of activation values, each element corresponding to an individual neuron in N: the cardinality of $\alpha_N(x)$ is equal to $|N|$. We call $\alpha_N(x)$ the Activation Trace (AT) of x over neurons in N. Similarly, let $A_N(X)$ be a set of activation traces, observed over neurons in N, for a set of inputs X: $A_N(X) = \{\alpha_N(x) \mid x \in X\}$. We note that the activation trace is trivially available after each execution of the network for a given input.

Since behaviours of DL systems are driven along the data-flow and not control-flow, we assume that activation traces observed over all $\mathbf{N}$ with respect to X, $A_{\mathbf{N}}(X)$, fully captures

[1] Experiments show benefits of diversity for general testing [15] and benefits of a 'scale of distances' of test inputs for robustness testing introduced in [35].

the behaviours of the DL system under investigation when executed using X.[2]

Surprise Adequacy (SA) aims to measure the relative novelty (i.e., surprise) of a given new input with respect to the inputs used for training. Given a training set $\mathbf{T}$, we first compute $A_{\mathbf{N}}(\mathbf{T})$ by recording activation values of all neurons using every input in the training data set. Subsequently, given a new input x, we measure how surprising x is when compared to $\mathbf{T}$ by comparing the activation trace of x to $A_{\mathbf{N}}(\mathbf{T})$. This quantitative similarity measure is called Surprise Adequacy (SA). We introduce two variants of SA, each with different way of measuring the similarity between x and $A_{\mathbf{N}}(\mathbf{T})$.[3]

Note that certain types of DL tasks allow us to focus on parts of the training set $\mathbf{T}$ to get more precise and meaningful measurement of SA. For example, suppose we are testing a classifier with a new input x, which is classified by the DL system under investigation as the class c. In this case, the surprise of x is more meaningfully measured against $A_{\mathbf{N}}(T_c)$, in which T_c is the subset of $\mathbf{T}$ where members are classified as c. Basically, the input might be surprising as an example of class c even if not surprising in relation to the full set of training examples.

B. Likelihood-based Surprise Adequacy

Kernel Density Estimation (KDE) [41] is a way of estimating the probability density function of a given random variable. The resulting density function allows the estimation of relative likelihood of a specific value of the random variable. Likelihood-based SA (LSA) uses KDE to estimate the probability density of each activation value in $A_{\mathbf{N}}(\mathbf{T})$, and obtains the surprise of a new input with respect to the estimated density. This is an extension of existing work that uses KDE to detect adversarial examples [14]. To reduce dimensionality and computational cost, we only consider the neurons in a selected layer $N_L \subseteq \mathbf{N}$, which yields a set of activation traces, $A_{N_L}(\mathbf{X})$. To further reduce the computational cost, we filter out neurons whose activation values show variance lower than a pre-defined threshold, t, as these neurons will not contribute much information to KDE. The cardinality of each trace will be $|N_L|$. Given a bandwidth matrix H and Gaussian kernel function K, the activation trace of the new input x, and $x_i \in \mathbf{T}$, KDE produces density function $\hat{f}$ as follows:

$$\hat{f}(x) = \frac{1}{|A_{N_L}(\mathbf{T})|} \sum_{x_i \in \mathbf{T}} K_H(\alpha_{N_L}(x) - \alpha_{N_L}(x_i)) \quad (1)$$

Since we want to measure the *surprise* of the input x, we need a metric that increases when probability density decreases (i.e., the input is rarer compared to the training data), and vice versa (i.e., the input is similar to the training data). Adopting common approach of converting probability density to a measure of rareness [26], [39], we define LSA to be the negative of the log of density:

$$LSA(x) = -log(\hat{f}(x)) \quad (2)$$

Note that extra information about input types can be used to make LSA more precise. For example, given a DL classifier $\mathbf{D}$, we expect inputs that share the same class label will have similar ATs. We can exploit this by computing LSA per class, replacing $\mathbf{T}$ with $\{x \in \mathbf{T} \mid \mathbf{D}(x) = c\}$ for class c. We use per-class LSA for DL classifiers in our empirical evaluation.

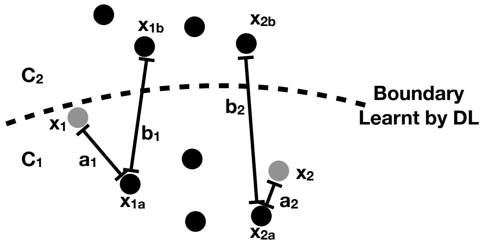


Fig. 1: An example of Distance-based SA. Black dots represent ATs of training data inputs, whereas grey dots represent ATs of new inputs, x_1 and x_2. Compared to distances from x_{1a} and x_{2a} to class c_2, AT of x_1 is farther out from class c_1 than that of x_2, i.e., $\frac{a_1}{b_1} > \frac{a_2}{b_2}$ (see Equations 3, 4, and 5). Consequently, we decide that x_1 is more surprising than x_2 w.r.t. class c_1.

C. Distance-based Surprise Adequacy

An alternative to LSA is simply to use the distance between ATs as the measure of surprise. Here, we define Distance-based Surprise Adequacy (DSA) using the Euclidean distance between the AT of a new input x and ATs observed during training. Being a distance metric, DSA is ideally suited to exploit the boundaries between inputs, as can be seen in the classification example in Figure 1. By comparing the distances a_1 and a_2 (i.e., distance between the AT of a new input and the reference point, which is the nearest AT of training data in c_1) to distances b_1 and b_2 (i.e., distance to c_2 measured from the reference point), we get a sense of how close to the class boundary the new inputs are. We posit that, for classification problems, inputs that are closer to class boundaries are more surprising and valuable in terms of test input diversity. On the other hand, for tasks without any boundaries between inputs, such as prediction of appropriate steering angle for autonomous driving car, DSA may not be easily applicable. With no class boundaries, an AT of a new input being far from that of another training input does not guarantee that the new input is surprising, as the AT may still be located in crowded parts of the AT space. Consequently, we only apply DSA for classification tasks, for which it can be more effective than LSA (see Section V-A and V-B for more details).

Let us assume that a DL system $\mathbf{D}$, which consists of a set of neurons $\mathbf{N}$, is trained for a classification task with a set of classes C, using a training dataset $\mathbf{T}$. Given the set of

[2] For the sake of simplicity, we assume that it is possible to get the complete activation traces from all the neurons in a DL system. For network architectures with loops, such as Recurrent Neural Nets (RNNs) [18], it is possible to *unroll* the loops up to a predefined bound [40].

[3] However, the main idea is general and other, specific variants would result if using other similarity functions.

activation traces $A_{\mathbf{N}}(\mathbf{T})$, a new input x, and a predicted class of the new input $c_x \in C$, we define the reference point x_a to be the closest neighbour of x that shares the same class. The distance between x and x_a follows from the definition:

$$\begin{aligned} x_a &= \operatorname*{argmin}_{\mathbf{D}(x_i)=c_x} \|\alpha_{\mathbf{N}}(x) - \alpha_{\mathbf{N}}(x_i)\|, \\ dist_a &= \|\alpha_{\mathbf{N}}(x) - \alpha_{\mathbf{N}}(x_a)\| \end{aligned} \tag{3}$$

Subsequently, from x_a, we find the closest neighbour of x_a in a class other than c_x, x_b, and the distance $dist_b$, as follows:

$$\begin{aligned} x_b &= \operatorname*{argmin}_{\mathbf{D}(x_i)\in C\setminus\{c_x\}} \|\alpha_{\mathbf{N}}(x_a) - \alpha_{\mathbf{N}}(x_i)\|, \\ dist_b &= \|\alpha_{\mathbf{N}}(x_a) - \alpha_{\mathbf{N}}(x_b)\| \end{aligned} \tag{4}$$

Intuitively, DSA aims to compare the distance from the AT of a new input x to known ATs belonging to its own class, c_x, to the known distance between ATs in class c_x and ATs in other classes in $C \setminus \{c_x\}$. If the former is relatively larger than the latter, x would be a surprising input for class c_x to the classifying DL system $\mathbf{D}$. While there are multiple ways to formalise this we select a simple one and calculate DSA as the ratio between $dist_a$ and $dist_b$. Investigation of more complicated formulations is left as future work.

$$DSA(x) = \frac{dist_a}{dist_b} \tag{5}$$

D. Surprise Coverage

Given a *set* of inputs, we can also measure the range of SA values the set *covers*, called Surprise Coverage (SC). Since both LSA and DSA are defined in continuous spaces, we use bucketing to discretise the space of surprise and define both Likelihood-based Surprise Coverage (LSC) and Distance-based Surprise Coverage (DSC). Given an upper bound of U, and buckets $B = \{b_1, b_2, ..., b_n\}$ that divide $(0, U]$ into n SA segments, SC for a set of inputs X is defined as follows:

$$SC(X) = \frac{|\{b_i \mid \exists x \in X : SA(x) \in (U \cdot \frac{i-1}{n}, U \cdot \frac{i}{n}]\}|}{n} \tag{6}$$

A set of inputs with high SC is a diverse set of inputs ranging from similar to those seen during training (i.e., low SA) to very different from what was seen during training (i.e., high SA). We argue that an input set for a DL system should not only be diversified, but *systematically* diversified considering SA. Recent results also validate this notion by showing that more distant test inputs were more likely to lead to exceptions but might not be as relevant for testing [35].

While we use the term *cover* and *coverage*, the implications of SA based coverage is different from the traditional structural coverage. First, unlike most of the structural coverage criteria, there is no finite set of targets to cover, as in statement or branch coverage: an input can, at least in theory, be arbitrarily surprising. However, an input with arbitrarily high SA value may simply be irrelevant, or at least less interesting, to the problem domain (e.g., an image of a traffic sign will be irrelevant to the testing of animal photo classifiers). As such, SC can only be measured with respect to pre-defined upper bound, in the same way the theoretically infinite path coverage is bounded by a parameter [44]. Second, SC does not render itself to a combinatorial set cover problem, which the test suite minimisation is often formulated into [43]. This is because a single input yields only a single SA value and cannot belong to multiple SA buckets. The sense of redundancy with respect to SC as a coverage criteria is weaker than that of structural coverage, for which a single input can cover multiple targets. While we aim to show that SA can guide the better selection of inputs, rigorous study of optimisation of test suites for DL systems remains a future work. However, as we show with our empirical studies, SC can still guide test input selection.

III. Research Questions

Our empirical evaluation is designed to answer the following research questions.

RQ1. Surprise: Is SADL capable of capturing the relative surprise of an input of a DL system?

We provide answers to RQ1 from different angles. First, we compute the SA of each test input included in the original dataset, and see if a DL classifier finds inputs with higher surprise more difficult to correctly classify. We expect more surprising input to be harder to correctly classify. Second, we evaluate whether it is possible to detect adversarial examples based on SA values, as we expect adversarial examples to be more surprising as well as to cause different behaviours of DL systems. Using different techniques, multiple sets of adversarial examples are generated and compared by their SA values. Finally, we train adversarial example classifiers using logistic regression on SA values. For each adversarial attack strategy, we generate 10,000 adversarial examples using 10,000 original test images provided by MNIST and CIFAR-10. Using 1,000 original test images and 1,000 adversarial examples, all chosen randomly, we train the logistic regression classifiers. Finally, we evaluate the trained classifiers using the remaining 9,000 original test images and 9,000 adversarial examples. If SA values correctly capture the behaviour of DL systems, we expect the SA based classifiers to successfully detect adversarial examples. We use Area Under Curve of Receiver Operator Characteristics (ROC-AUC) for evaluation as it captures both true and false positive rates [8].

RQ2. Layer Sensitivity: Does the selection of layers of neurons used for SA computation have any impact on how accurately SA reflects the behaviour of DL systems?

Bengio et al. suggest that deeper layers represent higher level features of the input [5]: subsequent work that introduced KDE based adversarial example detection technique [14] assumes the deepest (i.e., the last hidden) layer to contain the most information helpful for detection. We evaluate this assumption in the context of SA by calculating LSA and DSA of all individual layers, and subsequently by comparing adversarial example classifiers trained on SA from each layer.

TABLE I: List of datasets and models used in the study.

Dataset	Description	DNN Model	# of Neuron	Synthetic Inputs	Performance
MNIST	Handwritten digit images composed of 50,000 images for training and 10,000 images for test.	A five layer ConvNet with max-pooling and dropout layers.	320	FGSM, BIM-A, BIM-B, JSMA, C&W.	99.31% (Accuracy)
CIFAR-10	Object recognition dataset in ten different classes composed of 50,000 images for training and 10,000 images for test.	A 12 layer ConvNet with max-pooling and dropout layers.	2,208	FGSM, BIM-A, BIM-B, JSMA, C&W.	82.27% (Accuracy)
Udacity Self-driving Car Challenge	Self-driving car dataset that contains camera images from the vehicle, composed of 101,396 images for training and 5,614 images for test. The goal of the challenge is to predict steering wheel angle.	Dave-2 [6] architecture from Nvidia.	1,560	DeepXplore's test input generation via joint optimization.	0.09 (MSE)
		Chauffeur [1] architecture with CNN and LSTM.	1,940	DeepTest's combined transformation.	0.10 (MSE)

RQ3. Correlation: Is SC correlated to existing coverage criteria for DL systems?

In addition to capturing input surprise, we want SC to be consistent with existing coverage criteria based on counting aggregation. If not, there is a risk that SC is in fact measuring something other than input diversity. For this, we check whether SC is correlated with other criteria. We control the input diversity by cumulatively adding inputs generated by different method (i.e., different adversarial example generation techniques or input synthesis techniques), execute the studied DL systems with these input, and compare the observed changes of various coverage criteria including SC and four existing ones: DeepXplore's Neuron Coverage (NC) [40] and three Neuron-level Coverages (NLCs) introduced by DeepGauge [27]: k-Multisection Neuron Coverage (KMNC), Neuron Boundary Coverage (NBC), and Strong Neuron Activation Coverage (SNAC).

For MNIST and CIFAR-10, we start from the original test data provided by the dataset (10,000 images), and add 1,000 adversarial examples, generated by FGSM, BIM-A, BIM-B, JSMA, and C&W, at each step. For Dave-2, we start from the original test data (5,614 images) and add 700 synthetic images generated by DeepXplore at each step. For Chauffeur, each step adds 1,000 synthetic images (Set1 to Set3), each produced by applying random number of DeepTest transformations.

RQ4. Guidance: Can SA guide retraining of DL systems to improve their accuracy against adversarial examples and synthetic test inputs generated by DeepXplore?

To evaluate whether SADL can guide additional training of existing DL systems with the aim of improved accuracy against adversarial examples, we ask whether SA can guide the selection of input for additional training. From the adversarial examples and synthesised inputs for these models[4], we choose four sets of 100 images from four different SA ranges. Given U as the upper bound used in RQ3 to compute the SC, we divide the range of SA $[0, U]$ into four overlapping subsets: the first subset including the low 25% SA values ($[0, \frac{U}{4}]$), the second including the lower half ($[0, \frac{2U}{4}]$), the third including the lower 75% ($[0, \frac{3U}{4}]$), and finally the entire range, $[0, U]$. These four subsets are expected to represent increasingly more diverse sets of inputs. We set the range R to one of these four, randomly sample 100 images from each R, and train existing models for five additional epochs. Finally, we measure each model's performance (accuracy for MNIST and CIFAR-10, MSE for Dave-2) against the entire adversarial and synthetic inputs, respectively. We expect retraining with more diverse subset will result in higher performance.

IV. Experimental Setup

We evaluate SADL on four different DL systems using (a) the original test sets, (b) adversarial examples generated by five attack strategies, and (c) synthetic inputs generated by DeepXplore [34] and DeepTest [40]. This section describes the studied DL systems and the input generation methods.

A. Datasets and DL Systems

Table I lists the subject datasets and models of DL systems. MNIST [25] and CIFAR-10 [21] are widely used datasets for machine learning research, each of which is a collection of images in ten different classes. For MNIST, we adopt the widely studied five layer Convolutional Neural Network (ConvNet) with max-pooling and dropout layers and train it to achieve 99.31% accuracy on the provided test set. Similarly, the adopted model for CIFAR is a 12-layer ConvNet with max-pooling and dropout layers, trained to achieve 82.27% accuracy on the provided test set.

For evaluation of SADL for DL systems in safety critical domains, we use the Udacity self-driving car challenge dataset [2], which contains a collection of camera images from the driving car. As its aim is to predict steering wheel angle, the model accuracy is measured using Mean Squared Error (MSE) between actual and predicted steering angles. We use a pre-trained Dave-2 model [6], which is a public artefact provided by DeepXplore[5], and a pre-trained Chauffeur model [1], made publicly available by the Udacity self-driving car challenge. Dave-2 consists of nine layers including five convolutional layers, and achieves 0.09 in MSE. Chauffeur consists of both a ConvNet and an LSTM sub-model, and achieves 0.10 in MSE.

[4] We could not resume training of Chauffeur model for additional five epochs, which is why it is absent from RQ4.

[5] DeepXplore is available from: https://github.com/peikexin9/deepxplore.

B. Adversarial Examples and Synthetic Inputs

SADL is evaluated using both adversarial examples and synthetic test inputs. Adversarial examples are crafted by applying, to the original input, small perturbations imperceptible to humans, until the DL system under investigation behaves incorrectly [16]. We rely on adversarial attacks to generate input images for MNIST and CIFAR-10: these generated images are more likely to reveal robustness issues in the DL systems than the test inputs provided by the original datasets. We use five widely studied attack strategies to evaluate SADL: Fast Gradient Sign Method (FGSM) [16], Basic Iterative Method (BIM-A, BIM-B) [23], Jacobian-based Saliency Map Attack (JSMA) [33], and Carlini&Wagner (C&W) [9]. Our implementation of these strategies is based on `cleverhans` [32] and a framework of Ma et al. [30].

For Dave-2 and Chauffeur, we use the state-of-the-art synthetic input generation algorithms, DeepXplore [34] and DeepTest [40]. Both algorithms are designed to synthesise new test input from existing ones with the aim of detecting erroneous behaviours in autonomous driving vehicle. For Dave-2, we use DeepXplore's input generation via joint optimization algorithm, whose aim is to generate inputs that lead multiple DL systems trained independently, but using the same training data, to disagree with each other. Using Dave-2 and its two variants, Dave-dropout and Dave-norminit, we collect synthetic inputs generated by lighting effect (Light), occlusion by a single black rectangle (SingleOcc), and occlusion by multiple black rectangles (MultiOcc). For Chauffeur, we synthesise new inputs by iteratively applying random transformations provided by DeepTest to original input images: translation, scale, shear, rotation, contrast, brightness, and blur.[6]

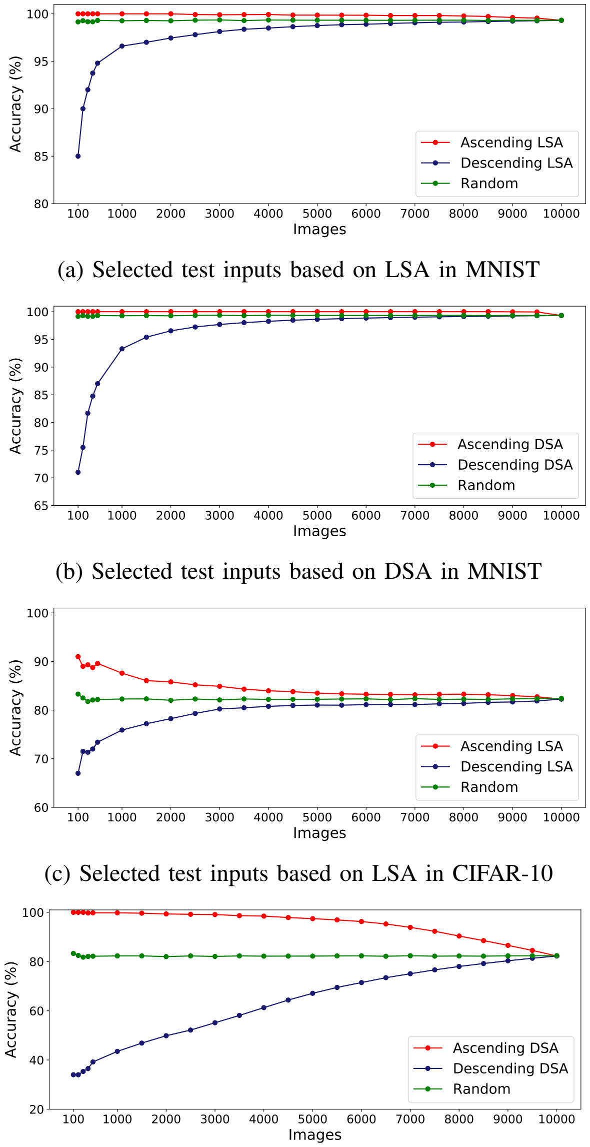
TABLE II: Configurations for RQ3.

DNN	NC	NLCs	LSC			DSC	
Model	*th*	*k*	layer	*n*	*ub*	*n*	*ub*
MNIST	0.5	1,000	activation_3	1,000	2,000	1,000	2.0
CIFAR-10	0.5	1,000	activation_3	1,000	100	1,000	2.0
Dave-2	0.5	1,000	block1_conv2	1,000	150	N/A	
Chauffeur	0.5	1,000	convolution2d_11	1,000	5	N/A	

C. Configurations

For all research questions, the default activation variance threshold for LSA is set to 10^{-5}, and the bandwidth for KDE is set using Scott's Rule [36]. The remaining of this Section details RQ specific configurations. For RQ1, we use the activation_2 layer for MNIST, and activation_6 for CIFAR-10, when computing LSA values. Computation of LSA based on all neurons is computationally infeasible due to precision loss. For RQ2, we set the activation variance threshold for layers activation_7 and activation_8 of CIFAR-10 to 10^{-4}, which reduces the number of neurons used for the computation of LSA and, consequently, the computational cost. For computation of other coverage criteria in RQ3, we use the configurations in Table II. The threshold of NC is set to 0.5. For NLCs, we all set the number of sections (k) to 1,000. For LSC and DSC, we manually choose the layer, the number of buckets (n), and the upper bound (ub). For RQ4, the layers chosen for MNIST and CIFAR-10 are activation_3 and activation_5 respectively. We perform 20 runs of retraining for each subject and report the statistics.

All experiments were performed on machines equipped with Intel i7-8700 CPU, 32GB RAM, running Ubuntu 16.04.4 LTS. MNIST and CIFAR-10 are implemented using Keras v.2.2.0.



(a) Selected test inputs based on LSA in MNIST

(b) Selected test inputs based on DSA in MNIST

(c) Selected test inputs based on LSA in CIFAR-10

(d) Selected test inputs based on DSA in CIFAR-10

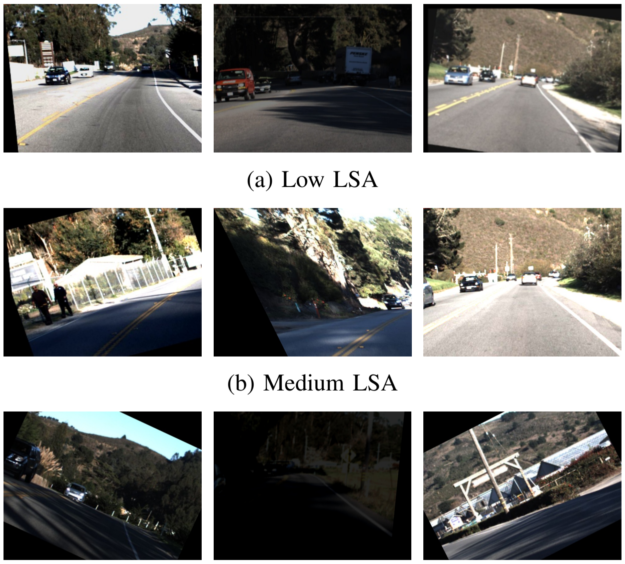
Fig. 2: Accuracy of test inputs in MNIST and CIFAR-10 dataset, selected from the input with the lowest SA, increasingly including inputs with higher SA, and vice versa (i.e., from the input with the highest SA to inputs with lower SA).

[6] At the time of our experiments, the publicly available version of DeepTest did not internally support realistic image transformations such as fog and rain effects.

V. Result

A. Input Surprise (RQ1)

Figure 2 shows how the classification accuracy changes when we classify sets of images of growing sizes from the test inputs included in the MNIST and CIFAR-10 dataset. The sets of images corresponding to the red dots (Ascending SA) start with images with the lowest SA, and increasingly include images with higher SA in the ascending order of SA; the sets of images corresponding to the blue dots grow in the opposite direction (i.e., from images with the highest SA to lower SA). As a reference, the green dots show the mean accuracy of randomly growing sets across 20 repetitions. It is clear that including images with higher LSA values, i.e., more surprising images, leads to lower accuracy. For visual confirmation on another dataset, we also chose sets of inputs synthesised for Chauffeur by DeepTest, from three distinct levels of LSA values: Figure 3 shows that the higher the LSA values are, the harder it is to recognise images visually. Both quantitatively and visually, the observed trend supports our claim that SADL captures input surprise: even for unseen inputs, SA can measure how surprising the given input is, which is directly related to the performance of the DL system.



(a) Low LSA

(b) Medium LSA

(c) High LSA

Fig. 3: Synthetic images for Chauffeur model generated by DeepTest. Images with higher LSA values tend to be harder to recognise and interpret visually.

Figure 4 shows plots of sorted DSA values of 10,000 adversarial examples, generated by each of the five techniques, as well as the original test inputs. Figure 5 contains similar plots based on LSA values of 2,000 randomly selected adversarial examples and the original test set, from different layers of MNIST and CIFAR-10. For both MNIST and CIFAR-10, the test inputs provided with the datasets (represented in blue colour) tend to be the least surprising, whereas the majority of adversarial examples are clearly separated from the test inputs by their higher SA values. This supports our claim that SADL can capture the differences in DL system's behaviours for adversarial examples.

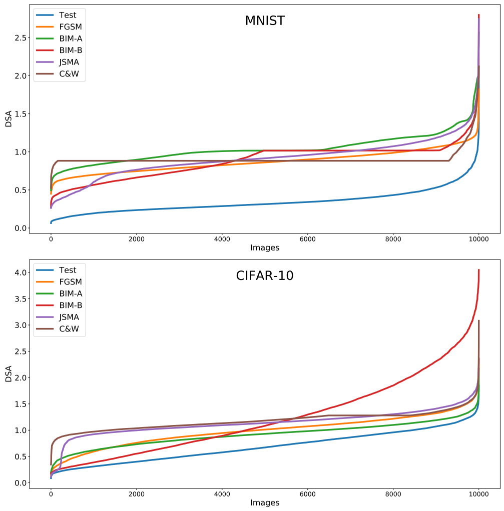


Fig. 4: Sorted DSA values of adversarial examples for MNIST and CIFAR-10.

Finally, Table III shows the ROC-AUC results of DSA-based classification using all neurons in MNIST and CIFAR-10.[7] The results show that the gap in DSA values observed in Figure 4 can be used to classify adversarial examples with high accuracy. For the relatively simpler MNIST model, the DSA-based classifier can detect adversarial examples with ROC-AUC ranging from 96.97% to 99.38%. The DSA-based classification for the more complicated CIFAR-10 model shows lower ROC-AUC values, but answers to RQ2 suggest that DSA from specific layers can produce significantly higher accuracy (see Section V-B).

TABLE III: ROC-AUC of DSA-based classification of adversarial examples for MNIST and CIFAR-10

Dataset	FGSM	BIM-A	BIM-B	JSMA	C&W
MNIST	98.34%	99.38%	96.97%	97.10%	99.04%
CIFAR-10	76.81%	72.93%	71.66%	88.96%	92.84%

Based on three different analyses, the answer to RQ1 is that **SADL can capture the relative surprise of inputs**. Inputs with higher SA are harder to correctly classify; adversarial examples show higher SA values and can be classified based on SA accordingly.

B. Impact of Layer Selection (RQ2)

Table IV shows the ROC-AUC of classification of adversarial examples, resulting in each row corresponding to a classifier trained on LSA and DSA from a specific layer in MNIST, respectively. Rows are ordered by their depth, i.e.,

[7] LSA-based classification is only possible for subsets of neurons due to the computational cost of KDE; hence we introduce the results of LSA-based classification when answering the impact of layer selection for RQ2.

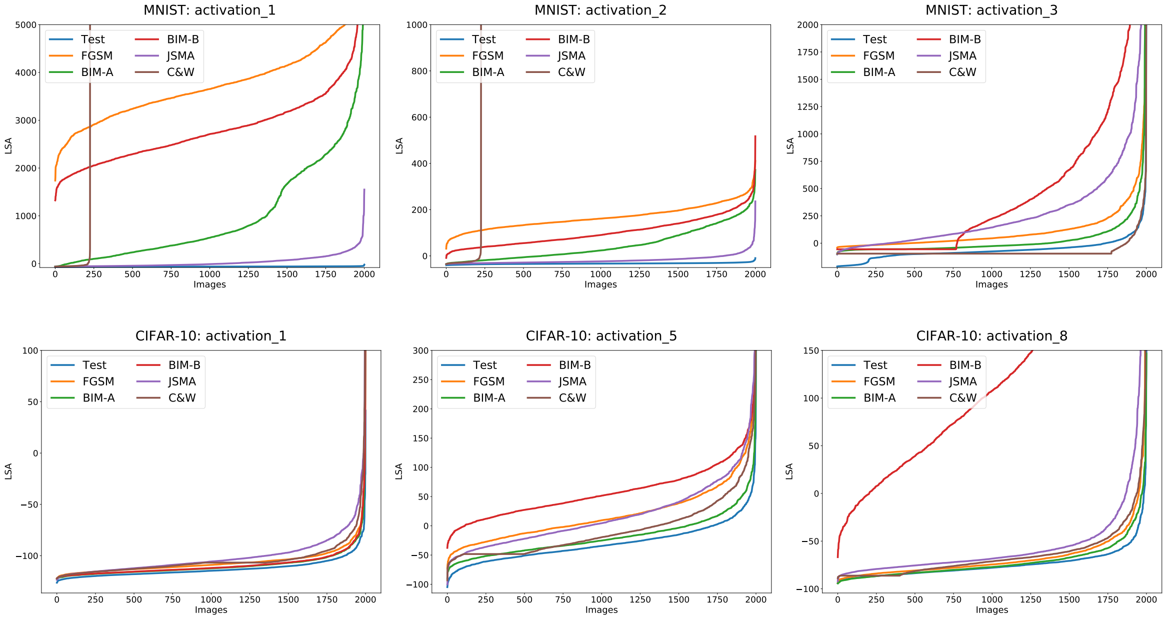


Fig. 5: Sorted LSA of randomly selected 2,000 adversarial examples for MNIST and CIFAR-10 from different layers

activation_3 is the deepest and the last hidden layer in MNIST. The highest ROC-AUC values for each attack strategy are typeset in bold. For MNIST, there is no clear evidence that the deepest layer is the most effective.

TABLE IV: ROC-AUC results of SA per layers on MNIST.

SA	Layer	FGSM	BIM-A	BIM-B	JSMA	C&W
LSA	activation_1	**100.00%**	**99.94%**	**100.00%**	98.17%	99.48%
	activation_2	**100.00%**	99.46%	**100.00%**	94.42%	99.23%
	pool_1	**100.00%**	99.73%	**100.00%**	**99.08%**	**99.61%**
	activation_3	93.29%	81.70%	86.73%	94.45%	37.96%
DSA	activation_1	**100.00%**	**99.85%**	**100.00%**	97.79%	99.39%
	activation_2	**100.00%**	99.39%	99.99%	97.59%	**99.69%**
	pool_1	**100.00%**	99.32%	99.99%	**98.21%**	**99.69%**
	activation_3	98.45%	99.43%	97.40%	97.07%	99.10%

The cases for which ROC-AUC is 100% can be explained by Figure 5: LSA values from activation_1 of MNIST, for example, show a clear separation between the original test inputs and FGSM, BIM-A, or BIM-B: by choosing an appropriate threshold, it is possible to completely separate test inputs from adversarial examples. Similarly, the plot of LSA from activation_3 of MNIST shows that C&W LSA line crossing with that of the original test data (i.e., C&W adversarial examples are less surprising than the original test data): this results in the low ROC-AUC value of 37.96%.

Table V contains the ROC-AUC values of LSA- and DSA-based classifiers, trained on each layer of the CIFAR-10 model: for each attack strategy, the highest ROC-AUC values are typeset in bold. Interestingly, LSA and DSA show different trends with CIFAR-10. With LSA, there is no strong evidence that the deepest layer produces the most accurate classifiers. However, with DSA, the deepest layer produces the most accurate classifiers for three out of five attack strategies (BIM-B, JSMA, and C&W), while the second deepest layer produces the most accurate classifier for BIM-A. More importantly, per-layer DSA values produce much more accurate classification results than all neuron DSA values, as can be seen in the comparison between Table III and Table IV & V. Identical models have been used to produce results in Tables above.

TABLE V: ROC-AUC results of SA per layers on CIFAR-10.

SA	Layer	FGSM	BIM-A	BIM-B	JSMA	C&W
LSA	activation_1	72.91%	61.59%	63.30%	76.85%	74.01%
	activation_2	89.59%	62.17%	73.20%	80.33%	75.98%
	pool_1	**93.31%**	61.79%	78.89%	82.64%	73.48%
	activation_3	86.75%	**62.69%**	76.93%	80.33%	79.02%
	activation_4	83.31%	62.73%	86.15%	80.86%	**80.42%**
	pool_2	82.82%	61.16%	89.69%	80.61%	73.85%
	activation_5	83.80%	60.64%	96.31%	79.56%	64.60%
	activation_6	63.85%	51.90%	99.74%	66.99%	60.40%
	pool_3	63.46%	51.86%	**99.77%**	67.62%	56.21%
	activation_7	67.96%	61.09%	92.18%	**83.02%**	76.85%
	activation_8	59.28%	52.66%	99.60%	73.26%	62.15%
DSA	activation_1	65.00%	62.25%	61.57%	73.85%	79.09%
	activation_2	77.63%	64.73%	67.95%	78.16%	81.59%
	pool_1	80.22%	64.89%	70.94%	78.96%	82.03%
	activation_3	**83.25%**	68.48%	73.49%	79.89%	84.16%
	activation_4	81.77%	68.94%	77.94%	80.55%	84.62%
	pool_2	82.51%	69.28%	81.43%	80.92%	84.81%
	activation_5	81.45%	70.29%	83.28%	82.15%	85.15%
	activation_6	71.71%	70.92%	71.15%	84.05%	85.42%
	pool_3	71.75%	70.35%	74.65%	83.57%	85.17%
	activation_7	71.04%	**71.44%**	81.46%	89.94%	92.98%
	activation_8	70.35%	70.65%	**90.47%**	**90.46%**	**94.53%**

Based on these results, we answer RQ2 that **DSA is sensitive to the selection of layers it is computed from, and benefits from choosing the deeper layer**. However, for LSA, there is no clear evidence supporting the deeper layer assumption. The layer sensitivity varies across different adversarial example generation strategies.

C. Correlation between SC and Other Criteria (RQ3)

Table VI shows how different coverage criteria respond to increasing diversity levels. Columns represent steps, at each of which more inputs are added to the original test set. If the increase in coverage at a step is less than 0.1 percentage point when compared to the previous step, the value is underlined. The threshold of 0.1 percentage point is based on the finest step change possible for LSC, DSC, as well as KMNC, as all three use bucketing with $k = 1,000$. We acknowledge that the threshold is arbitrary, and provide it only as a supporting aid. Figure 6 presents visualisation of results from CIFAR-10 and Chauffeur. Note that DSC cannot be computed for these two DL systems, as they are not classifiers (see Section II-C).

Overall, most of the studied criteria increase as additional inputs are added at each step. The notable exception is NC, which plateaus against many steps. This is in line with results in existing work [27]. There exists an interplay between the type of added inputs and how different criteria respond: SNAC, KMNC, and NBC show significant increases with the addition of BIM-B examples to CIFAR-10, but change little when C&W inputs are added. However, only SNAC and NBC exhibit a similar increase with the addition of input Set 1 for Chauffeur, while KMNC increases more steadily. Overall, with the exception of NC, we answer RQ3 that **SC is correlated with other coverage criteria introduced so far**.

DNN	Criteria	Test	Step 1 + FGSM	Step 2 + BIM-A	Step 3 + BIM-B	Step 4 + JSMA	Step 5 + C&W
MNIST	LSC	29.50	34.90	37.10	56.30	61.90	62.00
	DSC	46.00	56.10	65.00	67.20	70.90	72.30
	NC	42.73	42.73	43.03	43.03	43.03	45.45
	KMNC	68.42	70.96	72.24	75.82	77.31	77.37
	NBC	6.52	14.55	16.36	36.06	38.03	43.48
	SNAC	10.91	19.39	19.39	53.33	57.27	57.27
CIFAR-10	LSC	46.20	54.70	55.8	57.70	61.10	63.20
	DSC	66.20	70.10	70.6	80.90	83.40	84.10
	NC	26.15	26.28	26.28	26.28	26.33	27.01
	KMNC	28.77	29.30	29.51	34.09	34.31	34.41
	NBC	6.56	7.26	7.30	23.96	24.01	24.84
	SNAC	12.58	13.71	13.8	47.11	47.2	47.70
DNN	Criteria	Test	+ SingleOcc	+ MultiOcc	+ Light		
Dave-2	LSC	30.00	42.00	42.00	76.00		
	NC	79.55	80.26	80.45	83.14		
	KMNC	33.53	35.15	35.91	37.94		
	NBC	0.51	5.29	5.32	6.60		
	SNAC	1.03	10.58	10.64	13.21		
DNN	Criteria	Test	+ Set 1	+ Set 2	+ Set 3		
Chauffeur	LSC	48.90	53.50	56.10	58.40		
	NC	22.14	22.65	22.70	22.83		
	KMNC	48.08	50.79	52.20	53.21		
	NBC	3.05	16.88	17.96	19.13		
	SNAC	3.93	18.37	19.41	20.93		

TABLE VI: Changes in various coverage criteria against increasing input diversity. We put additional inputs into the original test inputs and observe changes in coverage values.

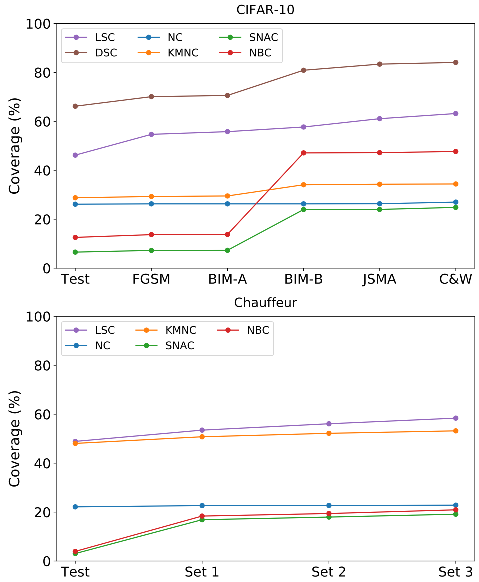


Fig. 6: Visualisation of CIFAR-10 and Chauffeur in Table VI. As additional sets of inputs (x-axis) are added to the original test set, various coverage criteria (y-axis) increase.

D. Retraining Guidance (RQ4)

Table VII shows the impact of SA-based guidance for retraining of MNIST, CIFAR-10, and Dave-2 models. The column R from $\frac{1}{4}$ to $\frac{4}{4}$ represents the increasingly wider ranges of SA from which the inputs for additional training are sampled; rows with $R = \emptyset$ show performance of the DL system before retraining. Overall, there are 23 retraining configurations (2 SA types × 2 DL systems × 5 adversarial attack strategies, and 1 SA type × 1 DL system × three input synthesis methods), each of which is evaluated against four SA ranges with 20 repetitions. Columns μ and σ contain the mean and standard deviation of observed performance metric (i.e., the highest accuracy for MNIST and CIFAR-10, the lowest MSE for Dave-2). The best performance is typeset in bold.

The full range, $\frac{4}{4}$, produces the best retraining performance for 13 configurations, followed by $\frac{2}{4}$ (5 configurations), $\frac{3}{4}$ (3 configurations), and $\frac{1}{4}$ (3 configurations). Note that for the configuration of CIFAR-10 and BIM-B, both ranges $\frac{2}{4}$ and $\frac{2}{4}$ produces the same and the best retraining performance. The largest improvement is observed when retraining MNIST against FGSM using DSA: the accuracy of the $\frac{4}{4}$ range shows 77.5% increase from that of $\frac{1}{4}$ (i.e., from 15.60% to 28.69%). While retraining MNIST against BIM-B using DSA shows even greater improvement (from 9.40% to 40.94%), we suspect this is an outlier as the accuracy for ranges $\frac{1}{4}$ and $\frac{2}{4}$ are significantly smaller when compared to other configurations.

While our observations are limited to the DL systems and input generation techniques studied here, we answer RQ4 that **SA can provide guidance for more effective retraining against adversarial examples based on our interpretation of the observed trend**.

VI. Threats to Validity

The primary threat to internal validity of this study is the correctness of implementation of the studied DL systems, as well as the computation of SA values. We have used publicly available architectures and pre-trained models as our subjects to avoid incorrect implementation. SA computation depends on a widely used computation library, `SciPy`, which has

DNN Model	SA	R	FGSM		BIM-A		BIM-B		JSMA		C&W	
			μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
MNIST		$\emptyset$	11.65	-	9.38	-	9.38	-	18.88	-	8.92	-
	LSA	1/4	25.81	1.95	95.14	0.69	**41.00**	0.01	72.67	3.09	92.51	0.51
		2/4	28.45	2.91	95.71	0.41	40.98	0.12	75.03	2.68	92.55	0.67
		3/4	**29.66**	3.63	95.87	0.98	40.97	0.10	75.48	2.60	92.41	1.03
		4/4	23.70	4.98	**95.90**	0.79	40.93	0.18	**77.37**	1.75	**92.56**	0.77
	DSA	1/4	15.60	2.12	93.67	3.42	9.90	1.05	74.56	2.62	12.80	0.96
		2/4	19.67	4.32	**95.78**	0.70	9.40	0.05	76.16	2.69	12.46	1.00
		3/4	26.37	6.15	95.37	0.93	40.81	0.22	**78.01**	1.87	12.37	1.14
		4/4	**27.69**	5.59	95.31	0.98	**40.94**	0.04	76.60	2.38	**13.61**	1.19
CIFAR-10		$\emptyset$	6.13	-	0.00	-	0.00	-	2.68	-	0.31	-
	LSA	1/4	11.07	1.20	32.34	1.70	0.59	1.76	32.80	2.05	34.38	2.83
		2/4	**12.96**	2.18	32.68	2.07	**0.89**	2.10	33.84	2.52	42.99	2.78
		3/4	12.79	2.17	32.14	2.40	**0.89**	2.10	35.81	2.81	45.58	2.23
		4/4	12.53	1.19	**32.79**	2.29	0.60	1.76	**35.83**	2.54	**45.74**	2.04
	DSA	1/4	**14.86**	2.16	25.94	2.99	0.01	0.00	34.92	2.01	44.21	2.02
		2/4	14.64	1.95	29.59	3.52	0.01	0.00	34.49	1.89	44.79	2.32
		3/4	13.81	1.85	31.93	2.77	0.01	0.00	35.61	2.40	46.16	2.45
		4/4	13.12	1.41	**32.17**	2.36	**0.60**	1.76	**37.32**	1.58	**46.21**	2.72

(a) MNIST and CIFAR-10

DNN Model	SA	R	SingleOcc		MultiOcc		Light	
			μ	σ	μ	σ	μ	σ
Dave-2		$\emptyset$	0.4212	-	0.0964	-	0.3822	-
	LSA	1/4	0.0586	0.0142	**0.0539**	0.0003	0.0573	0.0057
		2/4	**0.0540**	0.0012	0.0562	0.0060	**0.0560**	0.0042
		3/4	0.0554	0.0041	0.0544	0.0009	0.0570	0.0133
		4/4	0.0553	0.0028	0.0561	0.0042	0.0601	0.0111

(b) Dave-2

TABLE VII: Retraining guided by SA: we sample 100 inputs from four increasingly wider ranges of SA: $[0, \frac{U}{4}]$, $[0, \frac{2U}{4}]$, $[0, \frac{3U}{4}]$, and $[0, U]$, and retrain for five additional epochs using the samples as the training data, and measure the accuracy and MSE against the entire adversarial and synthetic inputs. Sampling from wider ranges improves the retraining accuracy.

stood the public scrutiny. Threats to external validity mostly concerns the number of the models and input generation techniques we study here. It is possible that SADL is less effective against other DL systems. While we believe the core principle of measuring input surprise is universally applicable, only further experimentations can reduce this particular risk. Finally, threats to construct validity asks whether we are measuring the correct factors to draw our conclusion. For all studied DL systems, activation traces are immediate artefacts of their executions and the meaning of output accuracy is well established, minimising the risk of this threat.

VII. Related Work

Adversarial examples pose significant threats to the performance of DL systems [7]. There are existing work in the machine learning community on detection of such inputs. Feinman et al. [14] first introduced the KDE as a means of similarity measurement, with the aim of detecting adversarial examples. SADL improves upon the existing work by a number of different ways. First, we generalise the concept of Surprise Adequacy (SA) and introduce Distance-based SA. Second, our evaluation is in the context of DL system testing. Third, our evaluation of SADL includes more complicated and practical DL systems, as well as testing techniques such as DeepXplore and DeepTest. Finally, we show that the choice of neurons has limited impact on LSA.

A range of techniques has been recently proposed to test and verify DL systems. The existing techniques are largely based on two assumptions. The first assumption is a variation of metamorphic testing [11], [31], [42]. Suppose a DL system N produces an output o when given i as the input, i.e., $N(i) = o$. Then we expect $N(i') \simeq o$ when $i' \simeq i$. Huang et al. [19] proposed a verification technique that can automatically generate counter-examples that violate this assumption. Pei et al. introduced DeepXplore [34], a white-box technique that generates test inputs that cause disagreement among a set of DL systems, i.e., $N_m(i) \neq N_n(i)$ for independently trained DL systems N_m and N_n. Tian et al. presented DeepTest, whose metamorphic relations include both simple geometric perturbations as well as realistic weather effects [40]. The second assumption is that the more diverse a set of input is, the more effective it will be for testing and validating DL systems. Pei et al. proposed Neuron Coverage (NC), which measures the ratio of neurons whose activation values are above a predefined threshold [34]. It has been shown that adding test inputs that violate the first assumption increases the diversity measured through NC. Similarly, DeepGauge introduced a set of multi-granularity coverage criteria that are thought to reflect behaviours of DL systems in finer granularity [27]. While these criteria capture input diversity, all of them are essentially count of neurons unlike SA, and therefore cannot be directly linked to behaviours of DL systems. We show that SA is closely related to the behaviours by training accurate adversarial example classifiers based on SA.

Apart from coverage criteria, other concepts in traditional software testing have been reformulated and applied to testing of DL systems. Ma et al. proposed DeepCT, which views ranges of neuron activation values as parameter choices and applies Combinatorial Interaction Testing (CIT) to measure interaction coverage [29]. SC is different from DeepCT as SADL aims to quantify the amount of surprise, rather than simply to detect surprise via increase in coverage. DeepMutation applies the principle of mutation testing to DL systems by mutating training data, test data, as well as the DL system itself, based on source and model level mutation operators [28].

VIII. Conclusion

We propose SADL, a surprise adequacy framework for DL systems that can quantitatively measure relative surprise of each input with respect to the training data, which we call Surprise Adequacy (SA). Using SA, we also develop Surprise Coverage (SC), which measures the coverage of discretised input surprise ranges, rather than the count of neurons with specific activation traits. Our empirical evaluation shows that SA and SC can capture the surprise of inputs accurately and are good indicators of how DL systems will react to unknown inputs. SA is correlated with how difficult a DL system finds an input, and can be used to accurately classify adversarial examples. SC can be used to guide selection of inputs for more effective retraining of DL systems for adversarial examples as well as inputs synthesised by DeepXplore.

REFERENCES

[1] Autonomous driving model: Chauffeur. https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/chauffeur.

[2] The udacity open source self-driving car project. https://github.com/udacity/self-driving-car.

[3] Google accident 2016: A google self-driving car caused a crash for the first time http://www.theverge.com/2016/2/29/11134344/google-self-driving-car-crash-report, 2016.

[4] Paul Ammann and Jeff Offutt. *Introduction to Software Testing*. Cambridge University Press, 2016.

[5] Yoshua Bengio, Grégoire Mesnil, Yann Dauphin, and Salah Rifai. Better mixing via deep representations. *CoRR*, abs/1207.4404, 2012.

[6] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[7] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected. *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security - AISec '17*, 2017.

[8] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017.

[9] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016.

[10] 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.

[11] T. Y. Chen, F.-C. Kuo, T. H. Tse, and Zhi Quan Zhou. Metamorphic testing and beyond. In *Proceedings of the International Workshop on Software Technology and Engineering Practice (STEP 2003)*, pages 94–100, September 2004.

[12] Zhihua Cui, Fei Xue, Xingjuan Cai, Yang Cao, Gai-ge Wang, and Jinjun Chen. Detection of malicious code variants based on deep learning. *IEEE Transactions on Industrial Informatics*, 14(7):3187–3196, 2018.

[13] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2013.

[14] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

[15] Robert Feldt, Simon Poulding, David Clark, and Shin Yoo. Test set diameter: Quantifying the diversity of sets of test cases. In *Proceedings of the IEEE International Conference on Software Testing, Verification, and Validation*, ICST 2016, pages 223–233, 2016.

[16] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[17] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[19] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In Rupak Majumdar and Viktor Kunčak, editors, *Computer Aided Verification*, pages 3–29, Cham, 2017. Springer International Publishing.

[20] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1–10, 2015.

[21] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The CIFAR-10 dataset. *online: http://www.cs.toronto.edu/kriz/cifar.html*, 2014.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[23] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[24] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[25] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

[26] Stijn Luca, Peter Karsmakers, Kris Cuppens, Tom Croonenborghs, Anouk Van de Vel, Berten Ceulemans, Lieven Lagae, Sabine Van Huffel, and Bart Vanrumste. Detecting rare events using extreme value statistics applied to epileptic convulsions in children. *Artificial Intelligence in Medicine*, 60(2):89 – 96, 2014.

[27] Lei Ma, Felix Juefei-Xu, Jiyuan Sun, Chunyang Chen, Ting Su, Fuyuan Zhang, Minhui Xue, Bo Li, Li Li, Yang Liu, Jianjun Zhao, and Yadong Wang. Deepgauge: Comprehensive and multi-granularity testing criteria for gauging the robustness of deep learning systems. *CoRR*, abs/1803.07519, 2018.

[28] Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu, Chao Xie, Li Li, Yang Liu, Jianjun Zhao, et al. Deepmutation: Mutation testing of deep learning systems. *arXiv preprint arXiv:1805.05206*, 2018.

[29] Lei Ma, Fuyuan Zhang, Minhui Xue, Bo Li, Yang Liu, Jianjun Zhao, and Yadong Wang. Combinatorial testing for deep learning systems. *arXiv preprint arXiv:1806.07723*, 2018.

[30] Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Michael E Houle, Grant Schoenebeck, Dawn Song, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613*, 2018.

[31] Christian Murphy, Kuang Shen, and Gail Kaiser. Automatic system testing of programs without test oracles. In *Proceedings of the 18th International Symposium on Software Testing and Analysis*, ISSTA 2009, pages 189–200. ACM Press, 2009.

[32] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, and Rujun Long. Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2018.

[33] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. *CoRR*, abs/1511.07528, 2015.

[34] 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*, SOSP '17, pages 1–18, New York, NY, USA, 2017. ACM.

[35] Simon Poulding and Robert Feldt. Generating controllably invalid and atypical inputs for robustness testing. In *Software Testing, Verification and Validation Workshops (ICSTW), 2017 IEEE International Conference on*, pages 81–84. IEEE, 2017.

[36] David W Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[37] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[38] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[39] L Tarassenko, A Hann, A Patterson, E Braithwaite, K Davidson, V Barber, and D Young. Biosign™: Multi-parameter monitoring for early warning of patient deterioration. 2005.

[40] 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*, pages 303–314. ACM, 2018.

[41] Matt P Wand and M Chris Jones. *Kernel smoothing*. Chapman and Hall/CRC, 1994.

[42] Shin Yoo. Metamorphic testing of stochastic optimisation. In *Proceedings of the 3rd International Workshop on Search-Based Software Testing*, SBST 2010, pages 192–201, 2010.

[43] Shin Yoo and Mark Harman. Regression testing minimisation, selection and prioritisation: A survey. *Software Testing, Verification, and Reliability*, 22(2):67–120, March 2012.

[44] Hong Zhu, Patrick A. V. Hall, and John H. R. May. Software unit test coverage and adequacy. *ACM Comput. Surv.*, 29(4):366–427, December 1997.