Evaluating Surprise Adequacy for Deep Learning System Testing

JINHAN KIM, KAIST, Republic of Korea
ROBERT FELDT, Chalmers University of Technology, Sweden
SHIN YOO, KAIST, Republic of Korea

The rapid adoption of Deep Learning (DL) systems in safety critical domains such as medical imaging and autonomous driving urgently calls for ways to test their correctness and robustness. Borrowing from the concept of test adequacy in traditional software testing, existing work on testing of DL systems initially investigated DL systems from structural point of view, leading to a number of coverage metrics. Our lack of understanding of the internal mechanism of Deep Neural Networks (DNNs), however, means that coverage metrics defined on the Boolean dichotomy of coverage are hard to intuitively interpret and understand. We propose the degree of out-of-distribution-ness of a given input as its adequacy for testing: the more *surprising* a given input is to the DNN under test, the more likely the system will show unexpected behaviour for the input. We develop the concept of surprise into a test adequacy criterion, called Surprise Adequacy (SA). Intuitively, SA measures the difference in the behaviour of the DNN for the given input and its behaviour for the training data. We posit that a good test input should be sufficiently, but not overtly, surprising compared to the training data set. This paper evaluates SA using a range of DL systems from simple image classifiers to autonomous driving car platforms, as well as both small and large data benchmarks ranging from MNIST to ImageNet. The results show that the SA value of an input can be a reliable predictor of the correctness of the mode behaviour. We also show that SA can be used to detect adversarial examples, and also be efficiently computed against large training dataset such as ImageNet using sampling.

CCS Concepts: • **Software and its engineering** → **Software creation and management**; • **Computing methodologies** → *Machine learning*.

Additional Key Words and Phrases: Test Adequacy, Deep Learning Systems

1 INTRODUCTION

Deep Learning (DL) [37] is increasingly being incorporated into larger software systems, including those traditionally considered to be safety critical, such as autonomous driving [5, 8] and malware detection [12]. The rapid adoption is based on significant progress made by DL in many different areas, such as image recognition [14, 34, 58], speech recognition [24], and machine translation [27, 57]. Such rapid adoption of DL systems has resulted in needs for validating their behaviour, as it is critically important that these larger systems are both correct and predictable. Unfortunately, despite their impressive performances, DL systems are also known to exhibit unexpected behaviour under certain circumstances, of which we have growing evidence such as the case of an autonomous driving vehicle that crashed into the other vehicle that did not yield as expected by the autonomous driving system. However, most of existing software testing techniques are simply not applicable to Deep Neural Networks (DNNs) inside the DL systems, due to their fundamental differences from human written code [69]. For example, structural coverage [3], the most widely used test adequacy metric in traditional software testing, is not meaningful for DL systems, as their behaviour is not encoded in any control flow structures.

Choosing effective inputs that can reveal unexpected behaviour of System Under Test (SUT) is a fundamental cornerstone of any software testing activity. Consequently, many novel approaches

Authors' addresses: Jinhan Kim, jinhankim@kaist.ac.kr, School of Computing, KAIST, 291 Daehak-ro, Yuseong-gu, Daejeon, Republic of Korea, 34141; Robert Feldt, robert.feldt@chalmers.se, Department of Computer Science and Engineering, Chalmers University of Technology, House Jupiter, Chalmers Campus Lindholmen, Gothenburg, Sweden, SE-412 96; Shin Yoo, shin.yoo@kaist.ac.kr, School of Computing, KAIST, 291 Daehak-ro, Yuseong-gu, Daejeon, Republic of Korea, 34141.

towards measuring the adequacy of a test input for a given DNN have been proposed [41, 51, 61]. Given the lack of control flow structures in DNNs, these approaches fall back to a more fundamental principle in software testing, which is that the more diverse a set of input is the more effective for fault detection it also is [16]. For example, DeepXplore, the seminal work of Pei et al. [51], presented the Neuron Coverage (NC, the ratio of neurons whose activation values were above a predefined threshold) as the measure of diversity of neuron behaviour. Tian et al. [61] showed that Neuron Coverage can be associated with unexpected behaviour observed in DNN models trained for autonomous driving tasks. Ma et al. [41], on the other hand, refined and extended NC in various ways.

These newly proposed test adequacy metrics, however, are not without limitations, despite having made significant advances over *ad hoc* testing of DL systems. A fundamental limitation lies not within the way these adequacy metrics are designed, but in our understanding of how DNNs work. While the explainability of DNNs is a critical research topic that has received a lot of attention [55], we are still nowhere near the level of comprehension we have in source code when it comes to understanding the functional role of an individual neuron in any DNN of practical complexity. Consequently, it is difficult to *interpret* count based aggregation metrics such as NC: what does it mean to cover an additional neuron by activating it above the given threshold? Higher NC would indeed correlate with higher diversity of the input set, but only indirectly, without understanding the specific relationship between inputs and model behaviour.

Another limitation is one of granularity, and this does lie in the way the existing adequacy metrics are designed. Many existing metrics are essentially count based aggregation of neuron activation, meaning that we can only measure them against *sets* of inputs. Consequently, these metrics do not work when we try to compare two individual inputs using them. Consider the example of k-Multisection Neuron Coverage [41], 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 typically be $\frac{1}{k}$, unless the input activates some neurons outside the range observed during training, and therefore outside any buckets. Considering that comparison of individual test cases is the foundation of many techniques that aim to improve the efficiency of testing [70], the fixed coarse granularity at the input set level may limit how these adequacy metrics can be used. For example, given a newly collected input data, we may want to prioritise them based on their likelihood of revealing undesired behaviour of the DNN model under test.

We propose a new test adequacy metric, called Surprise Adequacy (SA), for DL systems to overcome these limitations. Like existing adequacy metrics, SA also captures how neurons activate. However, instead of counting neurons that activate above a certain threshold, SA focuses on how similar the given input is to the data that the model under test has been trained with: the similarity is measured via the similarity between neuron activations. 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] The name, Surprise Adequacy, suggests that, for effective testing of DL systems, the inputs should deliver appropriate levels of surprise to the model. Too little, and we risk not finding any unexpected behaviour as the DL system will handle very familiar inputs without any problem; too high, and we risk wasting out effort for inputs that are unlikely to be found in real world scenarios.

This paper is an extended version of our conference paper that introduced Surprise Adequacy [29]. The conference version included the following technical contributions:

[1]Experiments show benefits of diversity for general testing [16] and benefits of a 'scale of distances' of test inputs for robustness testing introduced in [52].

- We propose SA, 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 SA, Likelihood-based SA (LSA) and Distance-based SA (DSA), are proposed based on different ways to quantify surprise. Corresponding two coverage criteria, Likelihood-based Surprise Coverage (LSC) and Distance-based Surprise Coverage (DSC), are shown to be correlated with existing coverage criteria for DL systems.
- We show that SA 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 [38] and CIFAR-10 [32] dataset, respectively.
- We conduct an experimentation to study whether SA metrics can guide retraining by prioritising inputs to be used in the retraining dataset.
- We undertake all our experiments using publicly available DL systems ranging from small benchmarks (MNIST and CIFAR-10) to a system for autonomous driving vehicles (Dave-2 [5] and Chauffeur [1]). All implementations are available online.[2]

The paper has been extended with the following technical contributions:

- The large size of the training dataset, and the resulting computational load, may limit the applicability of the SA analysis. We study the impact of sampling training data for the SA analysis to investigate the scalability of the approach. Our results show that SA values computed using only 10% of the training data are still very closely correlated with the values computed with the full training data, while achieving about five times average speed-up.
- The original work reduced the dimensionality of the AT vectors using manual thresholds for variance in order to make LSA analysis computationally affordable. We study the impact of this threshold-based dimensionality reduction by varying the threshold values and measuring the impact on the accuracy of the resulting SA based analysis. Our results show that DSA and MDSA (a recently proposed variant of SA; see the last item of this list), both of which do not require variance based dimensionality reduction, can still benefit from the reduction.
- We add a new study that investigates the correlation between different formulations of Surprise Adequacy, including the one based on Mahalanobis distance. The results show that they are largely correlated.
- We add three more image classification benchmarks to expand the empirical evaluation and gain more evidence for our claims: Street View House Numbers (SVHN) [47], CIFAR-100 [32], and ImageNet [13]. SVHN and CIFAR-100 benchmarks are used to evaluate SA with image classification. ImageNet is used to evaluate the impact of sampling when computing SA, due to its significant size.
- We present an empirical evaluation of Mahalanobis Distance based Surprise Adequacy (MDSA), which is a new type of SA metric based on Mahalanobis Distance [45] that has been recently introduced by authors of the original conference paper [30]. MDSA has been only evaluated with a semantic segmentation DNN model [30] as well as natural language processing DNN models [31]; this extension presents a theoretical analysis on its computational efficiency, and reports findings from its evaluation against a wide range of image classification benchmarks.

The remainder of this paper is organised as follows. Section 2 introduces our new test adequacy metric, Surprise Adequacy, and describes three concrete formulations: Likelihood-based SA, Distance-based SA, and Mahalanobis Distance-based SA. Section 3.1 presents our research

[2]Please refer to https://github.com/coinse/sadl.

questions, and Section 3 describes the experimental set-up of the empirical evaluations that we conduct to answer the research questions. Section 4 presents the results from empirical evaluations, and Section 5 addresses threats to validity. Section 6 discusses related work, and Section 7 concludes.

2 SURPRISE ADEQUACY FOR DEEP LEARNING SYSTEMS

Many existing test adequacy criteria for DL systems aim to measure the diversity of an input set. Neuron Coverage (NC) [51] 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 [41] 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.

2.1 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 the behaviours of the DL system under investigation when executed using X.[3]

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 three variants of SA, each with different way of measuring the similarity between x and $A_{\mathbf{N}}(\mathbf{T})$.[4]

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

[3]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) [25], it is possible to *unroll* the loops up to a predefined bound [61].

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

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.

2.2 Likelihood-based Surprise Adequacy

Kernel Density Estimation (KDE) [62] is a way of estimating the probability density function of a given random variable. Intuitively, it serves a similar purpose to that of a histogram, but with the help of kernel functions, the probability density is represented as a continuous function. 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 [15]. 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 $|N_L|$. Gaussian kernel function K is a basis of density estimation that determines the shape of distribution and a bandwidth matrix H is a free variable that controls bias and variance of estimates. Given the activation trace of the new input x, and $x_i \in \mathbf{T}$, KDE produces density function $\hat{f}$ as follows:

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

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 [40, 60], we define LSA to be the negative of the log of density:

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

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.

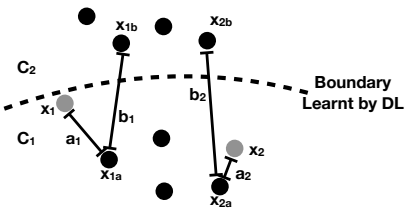


Fig. 1. An example of Distance-based SA. Black dots represent ATs of training data inputs, whereas grey dots represent ATs of new inputs, x_1 and x_2. Compared to distances from x_{1a} and x_{2a} to class c_2, AT of x_1 is farther out from class c_1 than that of x_2, i.e., $\frac{a_1}{b_1} > \frac{a_2}{b_2}$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

2.3 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 4.1 and 4.2 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 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:

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

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:

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

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 $dist_b$.

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

2.4 Mahalanobis Distance-based Surprise Adequacy

DSA is computed using the relative distance between *learnt boundaries* of all training inputs, which is only applicable to classification problems and it is hard to find the best formulation representing the surprise of the input. Recently, a more direct and general distance-based surprise adequacy named Mahalanobis Distance-based Surprise Adequacy (MDSA) has been introduced by the authors of the original conference paper [30]. MDSA exploits Mahalanobis Distance [45] that measures the distance between a point and a distribution. As a distance metric, MDSA aims to be increased

when the training inputs are far from the new input, and vice versa. Let the mean of $A_N(\mathbf{T})$ be μ, and covariance matrix of $A_N(\mathbf{T})$ be $\mathbf{C}$. Then, we calculate Mahalanobis distance between AT of the new input (a point) and a set of ATs of training inputs (a distribution) as follows:

$$MDSA(x) = \sqrt{(x-\mu)^\intercal \mathbf{C}(x-\mu)} \tag{6}$$

MDSA has been originally evaluated against the semantic segmentation task [30] as well as various Natural Language Processing (NLP) tasks [31]: this paper compares its performance to other types of SA metrics using a wide range of image classification benchmarks for which the existing SA metrics are evaluated.

Although MDSA and LSA share similar interpretations of measuring distance between the distribution of training inputs and new input, MDSA is computationally less costly than LSA and DSA, once its covariance matrix and mean vector have been pre-calculated, since it uses neither individual training inputs nor KDE. In that sense, MDSA generalizes well to large datasets such as ImageNet. More formally, the pre-calculation step of MDSA has an algorithmic complexity of $O(N * d^2 + d^{2.373})$ where N is the number of activation traces, and d their dimension (length). Which of the two factors dominate will depend on how N and d relate. Typically, for complex DNN architectures, the dimension of an activation trace (d) will be much larger than the number of inputs (N) available. Thus, if the best known algorithm for matrix inversion is used [65], the overall complexity will be $O(d^{2.373})$, given that typically $d^{0.373} > N$. Once the pre-calculation is done, calculating the distance to a new activation trace is dominated by matrix multiplication and has an algorithmic complexity of $O(d^2)$.

In addition to being faster to compute, MDSA can, similar to LSA, use extra information that computes MDSA per class if it targets classification problems. We therefore use per-class MDSA by default in the experiment.

2.5 Surprise Coverage

Given a *set* of inputs, we can also measure the range of SA values the set *covers*, called Surprise Coverage (SC). Since SA is defined in continuous spaces, we use bucketing to discretise the space of surprise and define both Likelihood-based Surprise Coverage (LSC), Distance-based Surprise Coverage (DSC), Mahalanobis Distance-based Surprise Coverage (MDSC). Given an lower bound L, upper bound U, and buckets $B = \{b_1, b_2, ..., b_n\}$ that divide $(L, U]$ into n segments where $b_i = (L + \frac{U-L}{n} \times (i-1), L + \frac{U-L}{n} \times i]$, SC for a set of inputs X is defined as follows:

$$SC(X) = \frac{|\{b_i \mid \exists x \in X : SA(x) \in b_i\}|}{n} \tag{7}$$

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 [52].

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 [71]. Second, SC does not render itself to a combinatorial set cover problem, which the test suite minimisation is often formulated into [70]. 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 criterion 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.

3 EXPERIMENTAL SETUP

We investigate a total of eight research questions listed below. To answer them, we evaluate SA on seven different DL systems using (a) the original test sets, (b) adversarial examples generated by five attack strategies, and (c) synthetic inputs generated by DeepXplore [51] and DeepTest [61]. This section describes the studied DL systems, input generation methods, and experimental configurations.

3.1 Research Questions

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

- **RQ1. Surprise:** Is SA capable of capturing the relative surprise of an input of a DL system?
- **RQ2. Layer Sensitivity:** Does the selection of the layer used for SA computation has any impact on how accurately SA reflects the behaviour of DL systems?
- **RQ3. Coverage:** How sensitive is SC to the new input sets? Is SC correlated to existing coverage criteria for DL systems?
- **RQ4. Sampling:** What is the impact of training set sampling to the effectiveness of capturing relative surprise of an input?
- **RQ5. Variance Threshold:** What is the effect of variance threshold, not only on LSA, but also on DSA and MDSA?
- **RQ6. Scalability:** Is SA scalable to a complex model trained on a large dataset?
- **RQ7. SA Correlation:** Are the ranks of LSA, DSA and MDSA correlated?
- **RQ8. Guidance:** Can SA guide retraining of DL systems to improve their accuracy against adversarial examples and synthetic test inputs generated by DeepXplore?

3.2 Datasets and DL Systems

Table 1 lists the subject datasets and models of DL systems.[5] MNIST [38], CIFAR-10, CIFAR-100 [33], and ImageNet [54] are widely used datasets for machine learning research. CIFAR-100 has 100 different classes, ImageNet has 1,000 different classes, and other datasets have ten different classes. For MNIST, we adopt the widely studied five layer Convolutional Neural Network (ConvNet) with max-pooling and dropout layers. It achieves 99.36% accuracy on the provided test set. Similarly, the adopted models for CIFAR-10 and SVHN are a 12-layer ConvNet and six layer ConvNet with max-pooling and dropout layers. They achieve 81.88% and 92.51% accuracy on the provided test set, respectively. To evaluate SA on larger DL systems, we use ResNet [23] and InceptionV3 [59] for CIFAR-100 and ImageNet, much more complex models that include 44 and 93 activation layers.

For evaluation of SA 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 [5], which

[5]Note that we use the name of a dataset on behalf of the name of a model for classifiers and vice versa for others (i.e., Dave-2 and Chauffeur).

Table 1. 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 testing.	Five layer ConvNet with max-pooling and dropout layers.	788	FGSM, BIM-A, BIM-B, JSMA, C&W.	99.36% (Accuracy)
SVHN	Real-world dataset of house numbers for object recognition, composed of 73,257 images for training and 26,032 images for testing.	Six layer ConvNet with max-pooling, batch normalization, and dropout layers.	3,092	FGSM, BIM-A, BIM-B, JSMA, C&W.	92.51% (Accuracy)
CIFAR-10	Object recognition dataset in ten different classes composed of 50,000 images for training and 10,000 images for testing.	12 layer ConvNet with max-pooling and dropout layers.	7,796	FGSM, BIM-A, BIM-B, JSMA, C&W.	81.88% (Accuracy)
CIFAR-100	Object recognition dataset having 100 different classes, composed of 50,000 images for training and 10,000 images for testing.	44 layer ResNet [23]	5,963	FGSM, BIM-A, BIM-B, C&W.	50.54% (Accuracy)
ImageNet	A large visual dataset for object recognition, composed of 1,281,167 images for training, 50,000 images for validation. We use validation set for testing.	93 layer InceptionV3 [59]	76,267	-	76.28% (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 testing. The goal of the challenge is to predict steering wheel angle.	Dave-2 [5] 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)

is a public artefact provided by DeepXplore[6], 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.

3.3 Adversarial Examples and Synthetic Inputs

SA 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 [21]. We rely on adversarial attacks to generate input images: 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 SA: Fast Gradient Sign Method (FGSM) [21], Basic Iterative Method (BIM-A, BIM-B) [35], Jacobian-based Saliency Map Attack (JSMA) [50], and Carlini&Wagner (C&W) [7]. Our implementation of these strategies is based on `cleverhans` [49] and a framework of Ma et al. [44].

For Dave-2 and Chauffeur, we use the state-of-the-art synthetic input generation algorithms, DeepXplore [51] and DeepTest[61]. 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 optimisation algorithm, whose aim

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

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.

Table 2. Configurations

DNN	NC	NLCs	Layer	LSC			DSC			MDSC		
Model	th	k		n	L	U	n	L	U	n	L	U
ConvNet (MNIST)	0.5	1,000	activation_3	1,000	-180.0	2,000.0	1,000	0.0	2.0	1,000	0.0	2,000.0
ConvNet (CIFAR-10)	0.5	1,000	activation_6	1,000	-200.0	400.0	1,000	0.0	2.5	1,000	0.0	2,000.0
ConvNet (SVHN)	0.5	1,000	activation_4	1,000	0.0	2,500.0	1,000	0.0	3.2	1,000	0.0	6,000.0
ResNet (CIFAR-100)	0.5	1,000	activation_43	1,000	-5.0	3,200.0	1,000	0.0	2.5	1,000	0.0	6,000.0
InceptionV3 (ImageNet)	0.5	1,000	activation_93	-	-	-	-	-	-	-	-	-
Dave-2	0.5	1,000	block1_conv2	1,000	0.0	80.0	N/A			1,000	0.0	150.0
Chauffeur	0.5	1,000	convolution2d_11	1,000	-70.0	0.0	N/A			1,000	0.0	130.0

3.4 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 [56]. Except for RQ2, we use the default layer listed in Table 2. For the computation of other coverage criteria used in RQ3, we use the configurations in Table 2. The threshold of NC is set to 0.5. For NLCs, we all set the number of sections (k) to 1,000. For LSC, DSC, and MDSC, we manually choose the number of buckets (n), the lower bound (L), and the upper bound (U). For RQ8, we perform 20 runs of retraining for each subject and report the statistics.

All experiments were performed on machines equipped with Nvidia GeForce GTX 1080 Ti, Intel i7-8700 CPU, 32GB RAM, running Ubuntu 16.04.4 LTS. All models and scripts are implemented using Keras v.2.3.1.

4 RESULTS

4.1 Input Surprise (RQ1)

We provide answers to RQ1 from two 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 choose three sets of synthesised inputs by DeepTest with high, middle, and low SA values, to visually confirm the expectation that the higher SA value is computed, the more the image is transformed (e.g., rotated).

Figure 2 shows how the classification accuracy changes when we classify sets of images of growing sizes from the test inputs included in each 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. Each row corresponds to the dataset and each column corresponds to the type of SA. It is clear that including images with higher SA values, i.e., more surprising images, leads to lower accuracy.

As adversarial examples are hard to be recognised by even human, we alternatively chose sets of inputs synthesised for Chauffeur by DeepTest, as well as for Dave-2 by DeepXplore, to visually

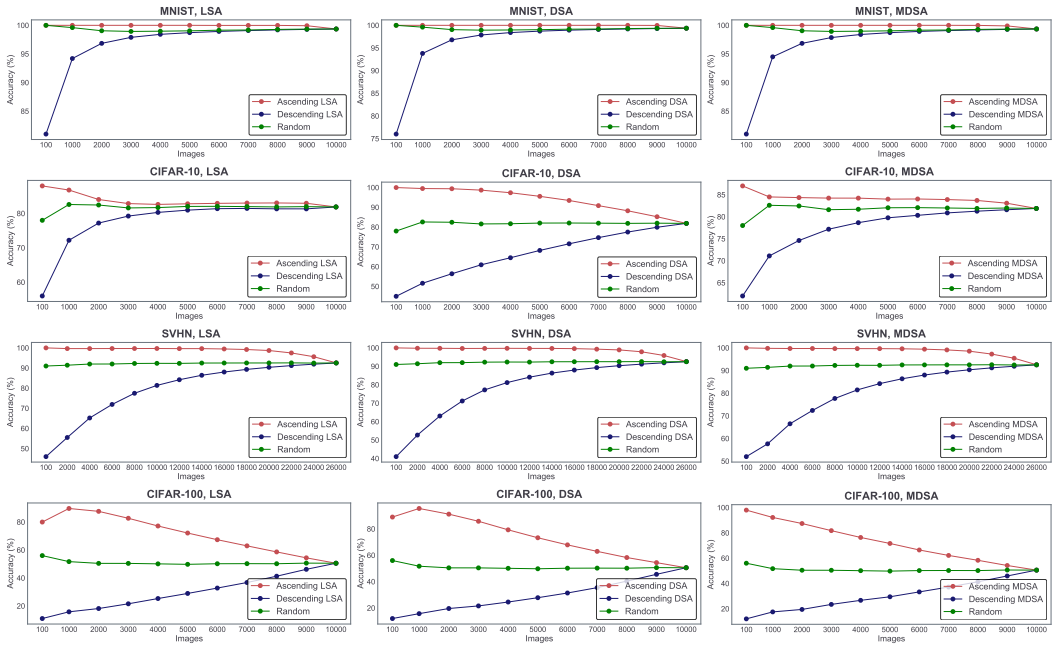


Fig. 2. Accuracy of test inputs for each dataset, selected from the input with the lowest SA, increasingly including inputs with higher SA, and vice versa (i.e., from the input with the highest SA to inputs with lower SA).

confirm our hypothesis. Figure 3 shows the synthetic images by DeepTest from three different SA levels: Low, Medium, and High, each of which consists of six images of which the first three images are picked by LSA and the rest three images are picked by MDSA. The results suggest that the higher the SA values are, the harder it is to recognise images visually.

On the same conditions that use three different SA levels, Figure 4 shows the synthetic images by DeepXplore. The columns represent three groups of images (i.e., groups of three consecutive columns) based on the modification strategy. Odd rows show images selected by LSA and even row show images selected by MDSA. The results are in line with the DeepTest-Chauffeur case. In addition, it shows that even if the images are not much distorted visually, they can have high SA values: some high LSA and MDSA images have a brownish tinge that can be surprising due to their rareness in the training set.

Both quantitatively and visually, the observed trends support our claim that SA 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.

Answer to RQ1: SA 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.

(a) Low LSA and Low MDSA

(b) Medium LSA and Medium MDSA

(c) High LSA and High MDSA

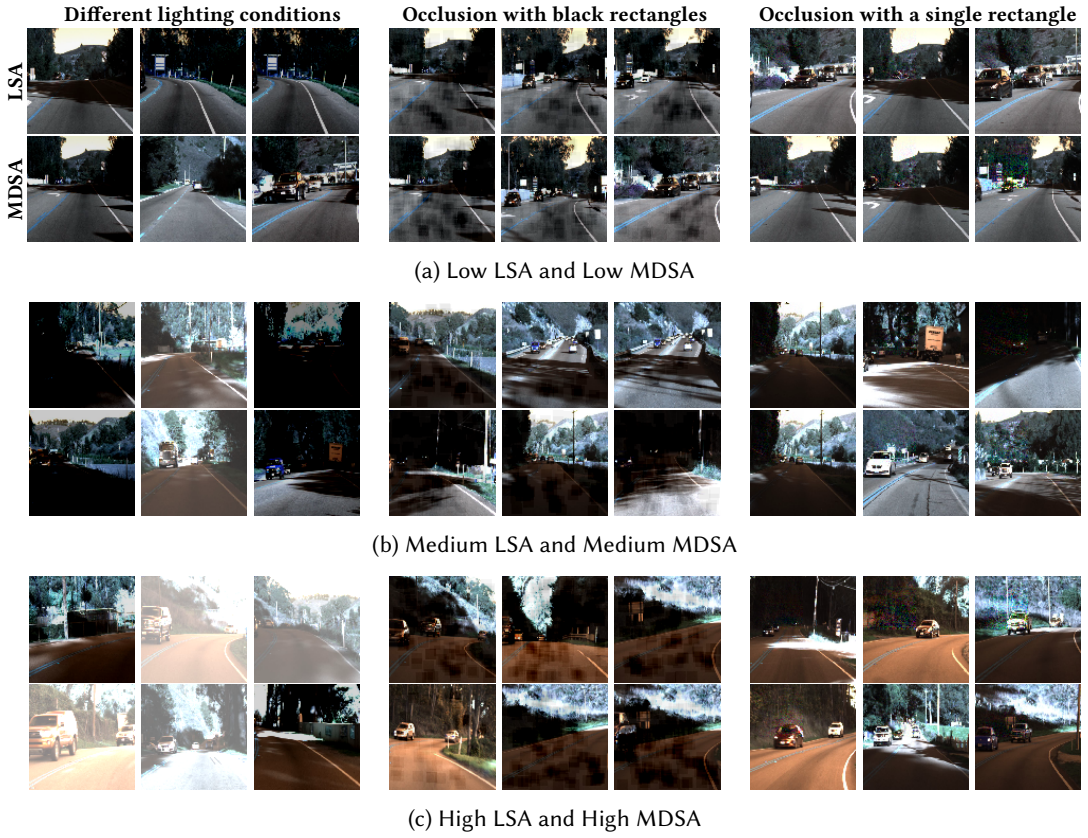
Fig. 3. Synthetic images for Chauffeur model generated by DeepTest. The three images on the left are picked based on LSA and three images on the right are picked based on MDSA. Images with higher LSA or MDSA values tend to be harder to recognise and interpret visually.

4.2 Impact of Layer Selection (RQ2)

Bengio et al. suggest that deeper layers represent higher level features of the input [4]: subsequent work that introduced KDE based adversarial example detection technique [15] assumes the deepest (i.e., the last hidden) layer to contain the most information helpful for detection. RQ2 evaluates this assumption in the context of SA by considering all individual layers. For each adversarial attack strategy, we generate the same number of adversarial examples using the original test images provided by each dataset. Using only one percent of both original test images and adversarial examples, all chosen randomly, we compute SA values for each layer and train the simple logistic regression classifier. Lastly, we evaluate the trained classifiers using the SA values from the remaining 99% of original test images and adversarial examples. The evaluation metric we use is Area Under Curve of Receiver Operator Characteristics (ROC-AUC) that captures both true and false positive rates [6].

Figure 5 shows ROC-AUC of classification of adversarial examples for each pair of SA and model. Each row corresponds to the model and each column corresponds to the SA used to train and test the classifier. The points in each figure represent the result of the classifier trained on the SA from the specific layer of the model; the x-axis represents the layers ordered by their depth, e.g., activation_3 (noted as 3) is the deepest and the last hidden layer in MNIST.

Overall, there is no strong evidence that the deepest layer produces the most accurate classifier: only for 17 out of 57 cases, the classifiers trained using the deepest layer show the best performance in terms of the detection of adversarial examples. The cases for which ROC-AUC scores are 100% (11 out of 736 cases) mean that SA values make a clear separation between the original test inputs and adversarial examples. However, for 15 out of 736 cases, ROC-AUC scores are lower than 50, meaning that there are no differences between SA values of the test set and adversarial examples, or the SA values of adversarial examples are lower than the SA values of the test set. For instance, LSA values of C&W from activation_3 of MNIST are lower than the original test set (i.e., C&W



(a) Low LSA and Low MDSA

(b) Medium LSA and Medium MDSA

(c) High LSA and High MDSA

Fig. 4. In addition to Figure 3, we showcase the synthetic images for Dave-2 model generated by DeepXplore. Each column group (three consecutive columns) represents different modification strategy. Images in odd rows are selected based on their LSA values and images in even rows are selected based on their MDSA values.

adversarial examples are less surprising than the original test data): this results in the low ROC-AUC value of 37.96%.

Answer to RQ2: All subject models are sensitive to the selection of layers it is computed from, and it is hard to tell that they benefit from choosing the deeper layer. The layer sensitivity varies across different adversarial example generation strategies.

4.3 Correlation between SC and Other Criteria (RQ3)

In addition to capturing input surprise, we want SC to be sensitive to the new input sets as well as 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, RQ3 checks whether SC is correlated with other criteria. We control the input diversity by cumulatively adding inputs generated by different methods (i.e., different adversarial example generation techniques or input synthesis techniques), execute the studied DL systems with these inputs, and compare the observed

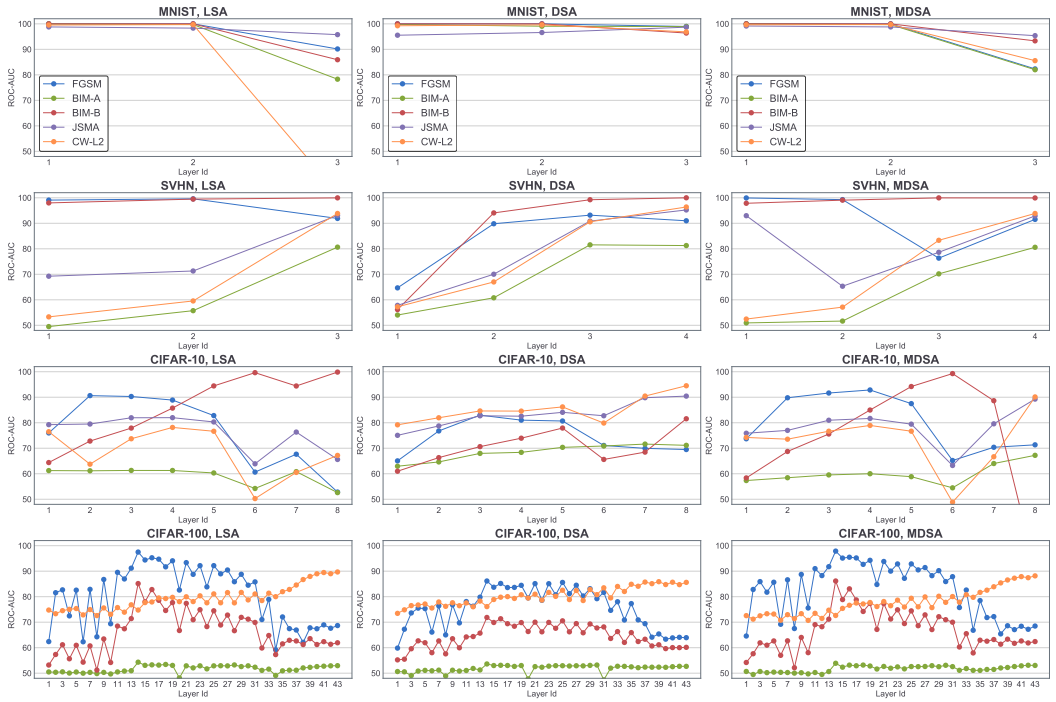


Fig. 5. ROC-AUC of classification of adversarial examples using SA per layers on the subject datasets. The x-axis represents activation layer number (e.g., activation_3 refers to 3).

changes of various coverage criteria including SC and four existing ones: DeepXplore's Neuron Coverage (NC) [61] and three Neuron-level Coverages (NLCs) introduced by DeepGauge [41]: k-Multisection Neuron Coverage (KMNC), Neuron Boundary Coverage (NBC), and Strong Neuron Activation Coverage (SNAC).

We start from the original test data provided by the dataset 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 a number of DeepTest transformations.

Table 3 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 cumulatively. In addition, Figure 6 visualises the results. 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, MDSC, as well as KMNC, as all four use bucketing with $k = 1,000$. We acknowledge that the threshold is arbitrary, and provide it only as a supporting aid. Note that DSC cannot be computed for Dave-2 and Chauffeur, since they are not classifiers (see Section 2.3).

Overall, most of the studied criteria increase more than 0.1 percentage point as additional inputs are added at each step (147 out of 169 cases). The notable exception is NC, which plateaus against many steps. This is in line with results in existing work [41]. However, LSC, DSC, and MDSC show

a clear increase to most of the additional inputs, while there is only one exception in MDSC when adding C&W inputs generated for CIFAR-10.

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.

Answer to RQ3: Overall, with the exception of NC, SC is correlated with other coverage criteria, such as SNAC, KMNC, and NBC. SC is also more sensitive to the additional inputs than the other studied criteria.

Table 3. Changes in various coverage criteria against increasing input diversity. We put additional inputs into the original test inputs and observe changes in coverage values.

DNN Model	Criteria	Test	Step 1 + FGSM	Step 2 + BIM-A	Step 3 + BIM-B	Step 4 + JSMA	Step 5 + C&W
MNIST	NC	66.24	67.51	67.77	67.77	67.77	69.92
	KMNC	70.59	72.66	73.41	77.63	78.65	78.69
	NBC	8.31	12.63	14.78	37.50	40.86	48.48
	SNAC	13.20	17.64	20.56	63.07	65.36	69.80
	LSC	34.10	37.50	40.70	57.20	70.60	71.00
	DSC	46.00	61.80	67.40	72.50	74.00	74.20
	MDSC	53.50	59.50	63.90	73.10	81.60	82.20
SVHN	NC	88.45	88.45	88.45	88.45	88.45	88.45
	KMNC	60.41	61.53	61.82	67.70	67.90	67.99
	NBC	17.82	19.19	20.42	51.65	51.71	51.71
	SNAC	27.23	28.98	31.08	89.62	89.62	89.62
	LSC	43.30	48.60	49.60	73.80	75.60	78.00
	DSC	48.10	52.10	52.80	56.60	60.00	63.70
	MDSC	44.00	47.50	48.80	81.30	82.40	85.20
CIFAR-10	NC	63.88	64.07	64.12	64.14	64.28	64.65
	KMNC	33.42	33.96	34.28	42.66	42.95	43.09
	NBC	5.76	6.56	6.70	37.14	37.84	38.40
	SNAC	8.44	9.93	10.21	52.62	53.94	54.68
	LSC	34.30	38.80	39.80	78.90	79.50	80.00
	DSC	64.80	68.00	68.80	78.60	79.90	80.30
	MDSC	31.10	35.20	37.00	72.00	72.60	72.60
CIFAR-100	NC	78.24	78.34	78.37	78.41	78.42	
	KMNC	60.28	61.99	62.61	63.10	63.44	
	NBC	13.62	16.73	16.94	17.10	17.15	
	SNAC	14.11	17.85	18.12	18.32	18.36	
	LSC	15.90	16.70	17.10	17.70	22.20	
	DSC	62.60	63.20	64.30	65.10	67.80	
	MDSC	16.40	16.80	17.20	18.20	23.00	

DNN Model	Criteria	Test	+ Single-Occ	+ Multi-Occ	+ Light
Dave-2	NC	79.55	80.90	81.67	83.46
	KMNC	36.20	38.39	40.18	42.27
	NBC	1.79	7.37	8.14	10.06
	SNAC	3.53	14.68	16.22	19.62
	LSC	38.80	48.30	55.40	61.00
	MDSC	45.20	56.10	64.40	73.50

DNN Model	Criteria	Test	+ Set 1	+ Set 2	+ Set 3
Chauffeur	NC	27.22	27.94	28.00	28.10
	KMNC	51.76	54.61	56.74	58.50
	NBC	31.20	40.91	44.81	46.57
	SNAC	33.16	42.82	45.61	47.73
	LSC	47.20	49.50	54.10	55.80
	MDSC	49.50	52.10	56.00	57.70

4.4 Training Set Sampling (RQ4)

Training deep neural networks typically suffers from processing large and high-dimensional datasets, which motivates GPU-accelerated computing these days. Computing SA has the same problem since it needs a large number of forward-passes for all training inputs to extract their ATs. However, one of the aims of computing SA for testing is to get the *ranks* of the inputs by their SA values: from images with high SA values that may be marked as important ones to be considered for further testing to images with low SA values that could be ignored. Therefore, to alleviate the cost of the large training set and to see how much trade-off occurs between correct ranks (i.e., same ranks as using all training inputs) and execution time, we sample training set with sample rates 0.1, 0.3, 0.5, and 0.7, and compute SA using the sampled training set. Then, we compute Spearman's rank correlation between the relative ranks from the original training set and the ranks from the sampled training set. If the correlation is strong and positive, we can decrease the computational

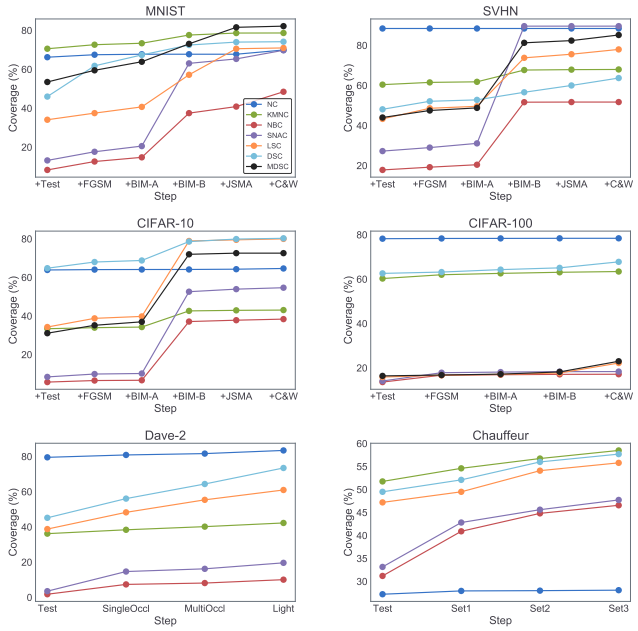


Fig. 6. Visualisation of Table 3. As additional sets of inputs (x-axis) are added to the original test set, various coverage criteria (y-axis) increase.

cost without adverse effects on the SA ranks. Note that we use uniform random sampling, as we do not make any assumptions about the model or the training data at the time of sampling. With a sufficiently large sample size, uniform random sampling can represent any underlying distribution.

Table 4 presents the results of Spearman's rank correlation between SA values computed from the original training set and sampled set. The column "ρ" represents the average correlation coefficient, "p" represents the average two-sided p-value whose null hypothesis is that two sets are uncorrelated, and "speed-up" represents the average speed-up when SA values are computed from the sampled training set.

All sample rates show a strong positive correlation with no sampling as high ρ values (0.96 on average) and low p-values (0.00 on average) suggest, meaning that only using 10% of the training set, we can have similar ranks of SA values. As expected, the highest average speed-up is 5.28x with a sample rate of 0.1, followed by 0.3 (2.39x), 0.5 (1.65x), and 0.7 (1.30x). DSA gets more benefits from sampling than LSA and MDSA since DSA computes distances between training inputs and new input each time. When we sample the training set of SVHN with a sample rate of 0.1, DSA becomes 13.41 times faster. On the other hand, the average speed-up of MDSA is relatively low: the highest speed-up of MDSA is 2.52x in the case of CIFAR-10. This is because MDSA makes use of training set to compute mean and covariance only once, which is much faster than computing KDE or distances for DSA.

Answer to RQ4: The ranks from the sampled training set are aligned with the original SA ranks from the full training set. The average speed-up with a sample rate of 0.1 is on average 5.28x and DSA is shown to take advantage of training set sampling more than LSA and MDSA.

Table 4. Spearman's rank correlation between SA values computed from the original training set and sampled training set averaged across 20 repetitions. For all cases, correlation coefficient (ρ) is high (> 0.80), which means that, only using 10% of training set, we can rank inputs by SA values like using the original training set.

DNN Model	Sample Rate	LSA					DSA					MDSA				
		ρ	p	t_{all}	t_{sample}	speed-up	ρ	p	t_{all}	t_{sample}	speed-up	ρ	p	t_{all}	t_{sample}	speed-up
MNIST	0.1	0.94	0.00	84.38	12.08	6.99	0.88	0.00	494.33	44.06	11.22	0.98	0.00	7.76	5.59	1.39
	0.3	0.98	0.00	84.38	29.78	2.83	0.92	0.00	494.33	144.95	3.41	1.00	0.00	7.76	6.31	1.23
	0.5	0.99	0.00	84.38	43.07	1.96	0.95	0.00	494.33	250.70	1.97	1.00	0.00	7.76	6.87	1.13
	0.7	1.00	0.00	84.38	57.12	1.48	0.97	0.00	494.33	354.81	1.39	1.00	0.00	7.76	7.16	1.09
SVHN	0.1	0.98	0.00	324.80	49.84	6.52	0.92	0.00	1804.53	134.56	13.41	0.99	0.00	15.55	9.94	1.56
	0.3	0.99	0.00	324.80	99.54	3.26	0.95	0.00	1804.53	470.97	3.83	1.00	0.00	15.55	11.16	1.39
	0.5	1.00	0.00	324.80	158.50	2.05	0.97	0.00	1804.53	803.54	2.25	1.00	0.00	15.55	12.70	1.22
	0.7	1.00	0.00	324.80	219.67	1.48	0.98	0.00	1804.53	1128.83	1.60	1.00	0.00	15.55	13.95	1.11
CIFAR-10	0.1	0.95	0.00	88.51	16.85	5.25	0.80	0.00	432.23	41.96	10.30	0.98	0.00	25.57	10.14	2.52
	0.3	0.98	0.00	88.51	36.22	2.44	0.86	0.00	432.23	125.83	3.44	1.00	0.00	25.57	13.74	1.86
	0.5	0.99	0.00	88.51	50.48	1.75	0.90	0.00	432.23	218.18	1.98	1.00	0.00	25.57	17.29	1.48
	0.7	0.99	0.00	88.51	65.07	1.36	0.94	0.00	432.23	310.61	1.39	1.00	0.00	25.57	20.61	1.24
CIFAR-100	0.1	nan	nan	26.89	13.90	1.93	0.83	0.00	332.65	38.56	8.63	0.64	0.00	20.29	11.09	1.83
	0.3	0.95	0.00	26.89	16.95	1.59	0.88	0.00	332.65	95.14	3.50	0.96	0.00	20.29	13.25	1.53
	0.5	0.98	0.00	26.89	19.62	1.37	0.91	0.00	332.65	169.39	1.96	0.99	0.00	20.29	15.47	1.31
	0.7	0.99	0.00	26.89	22.12	1.22	0.95	0.00	332.65	240.33	1.38	0.99	0.00	20.29	17.67	1.15
Dave-2	0.1	0.92	0.00	67.97	8.62	7.89	-	-	-	-	-	0.99	0.00	0.08	0.07	1.14
	0.3	0.97	0.00	67.97	18.66	3.64	-	-	-	-	-	1.00	0.00	0.08	0.07	1.14
	0.5	0.98	0.00	67.97	30.42	2.23	-	-	-	-	-	1.00	0.00	0.08	0.07	1.14
	0.7	0.99	0.00	67.97	45.39	1.50	-	-	-	-	-	1.00	0.00	0.08	0.07	1.13
Chauffeur	0.1	0.97	0.00	91.81	34.49	2.66	-	-	-	-	-	1.00	0.00	0.14	0.11	1.28
	0.3	0.99	0.00	91.81	47.53	1.93	-	-	-	-	-	1.00	0.00	0.14	0.12	1.20
	0.5	1.00	0.00	91.81	65.43	1.40	-	-	-	-	-	1.00	0.00	0.14	0.12	1.13
	0.7	1.00	0.00	91.81	78.80	1.17	-	-	-	-	-	1.00	0.00	0.14	0.12	1.14

4.5 Effect of Variance Threshold (RQ5)

LSA computation is based on KDE that sometimes fails if the variances of activation values of some neurons are too low. Those neurons should be eliminated for KDE which motivates our use of a variance threshold in LSA. Moreover, we hypothesize that filtering out neurons whose variances are lower than the pre-defined threshold can help compute more stable and accurate SA values because such neurons may disturb the approximation of SA.[7]

RQ5 studies the effects of variance threshold: first, we select five variance thresholds: 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, and an experimental case that does not filter any neurons. In turn, we train SA-based classifiers and see the changes of ROC-AUC scores against variance thresholds. If we can find a tendency between ROC-AUC score and variance threshold, we can recommend using a specific variance threshold as a default even for DSA or MDSA. Note that selecting the variance threshold is a practical concern. One may want to find the threshold that just works (i.e., there are sufficient neurons left for KDE), or one may seek the threshold that shows the best performance.

Table 5 shows the changes of ROC-AUC against the variance threshold used for SA computation. The column "th" represents the variance threshold and "None" is a case that does not use the variance threshold. The column "before" shows the original number of neurons and the column "after" shows the number of remaining neurons after thresholding. The highest ROC-AUC scores for each attack strategy are typeset in bold.

[7] Actually, it is not possible to define the meaning of *accurate* SA value because there is no ground truth value. As an alternative, we approximate the accuracy of SA by its fit-for-purpose/usefulness, and compare the ROC-AUC scores of SA-based classifiers for different variance thresholds.

Table 5. ROC-AUC of SA-based classification of adversarial examples with different variance thresholds, *th*.

DNN	SA	*th*	before	after	FGSM	BIM-A	BIM-B	JSMA	C&W	DNN	SA	*th*	before	after	FGSM	BIM-A	BIM-B	JSMA	C&W
MNIST	LSA	10^{-1}	128	0	-	-	-	-	-	CIFAR-10	LSA	10^{-1}	128	0	-	-	-	-	-
		10^{-2}	128	4	83.78	57.15	73.41	76.54	61.47			10^{-2}	128	0	-	-	-	-	-
		10^{-3}	128	70	**91.57**	**80.49**	**90.59**	**95.97**	61.64			10^{-3}	128	71	**70.58**	53.87	**99.70**	**64.83**	46.95
		10^{-4}	128	111	90.13	78.48	86.21	95.78	64.45			10^{-4}	128	126	60.82	54.15	99.64	64.00	**50.30**
		10^{-5}	128	120	90.15	78.30	85.93	95.77	33.64			10^{-5}	128	127	60.65	**54.22**	99.63	63.90	50.26
		None	128	128	89.86	78.89	85.43	95.74	**68.88**			None	128	128	60.65	**54.22**	99.63	63.90	50.26
	DSA	10^{-1}	128	0	-	-	-	-	-		DSA	10^{-1}	128	0	-	-	-	-	-
		10^{-2}	128	4	-	-	-	-	-			10^{-2}	128	0	-	-	-	-	-
		10^{-3}	128	70	98.95	98.81	**96.52**	98.48	**97.15**			10^{-3}	128	71	70.26	69.10	**67.51**	81.18	76.22
		10^{-4}	128	111	**99.04**	98.86	96.31	98.70	96.33			10^{-4}	128	126	**71.13**	**70.89**	65.93	**82.77**	**79.92**
		10^{-5}	128	120	98.98	**98.91**	96.38	**98.71**	96.72			10^{-5}	128	127	71.09	70.88	65.61	82.74	79.88
		None	128	128	98.95	**98.91**	96.37	98.69	96.77			None	128	128	71.09	70.88	65.61	82.74	79.88
	MDSA	10^{-1}	128	0	-	-	-	-	-		MDSA	10^{-1}	128	0	-	-	-	-	-
		10^{-2}	128	4	81.97	73.12	82.90	79.88	32.10			10^{-2}	128	0	-	-	-	-	-
		10^{-3}	128	70	**86.06**	**82.78**	**94.19**	**95.46**	**90.07**			10^{-3}	128	71	**72.49**	46.32	**99.39**	**65.06**	46.73
		10^{-4}	128	111	82.85	81.44	93.59	95.35	86.15			10^{-4}	128	126	65.48	**54.47**	99.32	63.34	48.87
		10^{-5}	128	120	83.05	81.63	93.64	95.39	86.24			10^{-5}	128	127	65.28	**54.47**	99.30	63.26	**48.91**
		None	128	128	82.64	81.81	93.48	95.41	85.85			None	128	128	65.28	**54.47**	99.30	63.26	**48.91**
SVHN	LSA	10^{-1}	128	42	90.38	79.19	**99.99**	92.29	92.57	CIFAR-100	LSA	10^{-1}	64	30	65.77	52.85	61.31	-	88.83
		10^{-2}	128	127	**91.92**	**80.76**	99.97	**93.15**	**93.90**			10^{-2}	64	64	**68.68**	**52.95**	**61.93**	-	**89.69**
		10^{-3}	128	128	91.90	80.65	99.97	93.08	93.85			10^{-3}	64	64	**68.68**	**52.95**	**61.93**	-	**89.69**
		10^{-4}	128	128	91.90	80.65	99.97	93.08	93.85			10^{-4}	64	64	**68.68**	**52.95**	**61.93**	-	**89.69**
		10^{-5}	128	128	91.90	80.65	99.97	93.08	93.85			10^{-5}	64	64	**68.68**	**52.95**	**61.93**	-	**89.69**
		None	128	128	91.90	80.65	99.97	93.08	93.85			None	64	64	**68.68**	**52.95**	**61.93**	-	**89.69**
	DSA	10^{-1}	128	42	90.97	81.12	**100.00**	95.09	96.20		DSA	10^{-1}	64	30	62.33	47.33	59.12	-	84.74
		10^{-2}	128	127	90.98	**81.27**	**100.00**	95.22	96.38			10^{-2}	64	64	**63.93**	**52.68**	**60.14**	-	**85.61**
		10^{-3}	128	128	**90.99**	**81.27**	**100.00**	**95.23**	**96.39**			10^{-3}	64	64	**63.93**	**52.68**	**60.14**	-	**85.61**
		10^{-4}	128	128	**90.99**	**81.27**	**100.00**	**95.23**	**96.39**			10^{-4}	64	64	**63.93**	**52.68**	**60.14**	-	**85.61**
		10^{-5}	128	128	**90.99**	**81.27**	**100.00**	**95.23**	**96.39**			10^{-5}	64	64	**63.93**	**52.68**	**60.14**	-	**85.61**
		None	128	128	**90.99**	**81.27**	**100.00**	**95.23**	**96.39**			None	64	64	**63.93**	**52.68**	**60.14**	-	**85.61**
	MDSA	10^{-1}	128	42	88.87	78.89	**99.99**	91.51	92.18		MDSA	10^{-1}	64	30	66.09	52.93	62.02	-	87.12
		10^{-2}	128	127	91.52	**80.61**	99.94	**92.96**	**93.92**			10^{-2}	64	64	**68.49**	**53.04**	**62.40**	-	**88.16**
		10^{-3}	128	128	**91.53**	80.59	99.94	92.89	93.85			10^{-3}	64	64	**68.49**	**53.04**	**62.40**	-	**88.16**
		10^{-4}	128	128	**91.53**	80.59	99.94	92.89	93.85			10^{-4}	64	64	**68.49**	**53.04**	**62.40**	-	**88.16**
		10^{-5}	128	128	**91.53**	80.59	99.94	92.89	93.85			10^{-5}	64	64	**68.49**	**53.04**	**62.40**	-	**88.16**
		None	128	128	**91.53**	80.59	99.94	92.89	93.85			None	64	64	**68.49**	**53.04**	**62.40**	-	**88.16**

The results indicate that, for all the subjects, there is no such case in which the variance threshold has to be used to make LSA computation possible, as ROC-AUC values in "None" represent.[8] Also, SVHN and CIFAR-100 are shown to have ATs whose variances are high. Thus, they are insensitive to the changes of variation threshold: from 10^{-2} to None of CIFAR-100 and from 10^{-3} to None of SVHN, neurons are not eliminated at all and classifiers produce the same and the highest ROC-AUC scores. On the other hand, MNIST and CIFAR-10 take advantage of using the variance threshold: for 34 out of 45 cases, ROC-AUC scores are higher than the scores of not using the variance threshold. The threshold 10^{-3} shows the highest ROC-AUC score in 18 cases, followed by 6 cases of 10^{-4}, 5 cases of 10^{-5}, and 5 cases of None.

Answer to RQ5: Even if the variance threshold is not a requisite for DSA and MDSA, removing the neurons whose activation variance is lower than the threshold can improve the performance of SA-based classifiers.

4.6 Scalability (RQ6)

A major computational bottleneck of SA comes from the number of training inputs and their dimensions. In this RQ, we provide a closer look at the execution time of running SA and find out whether SA is scalable to a large dataset by using ImageNet (ILSVRC-2012) [54], which contains

[8]We note that LSA computation of ImageNet in RQ7 (see Section 4.7) is not possible when the variance threshold is 10^{-5}.

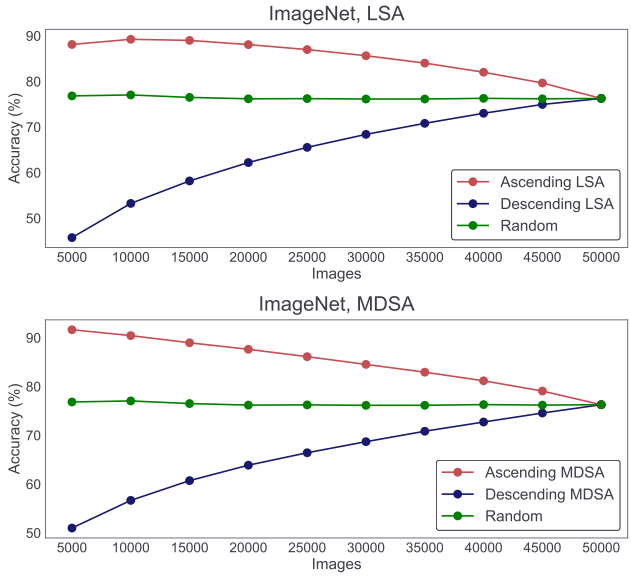


Fig. 7. Changes of accuracy when inputs are selected by their SA values like Figure 2.

Table 6. Spearman's rank correlation between SA values of ImageNet: we sample the training set with 20 repetitions and compare the rank from the sampled training set to the rank from the original training set.

Sample Rate	LSA					MDSA				
	ρ	p	t_{all}	t_{sample}	speed-up	ρ	p	t_{all}	t_{sample}	speed-up
0.1	-	-	-	-	-	0.75	0.00	15,304.25	1,617.41	9.46
0.3	0.92	0.00	16,276.06	5,024.07	3.24	0.96	0.00	15,304.25	4,689.12	3.26
0.5	0.98	0.00	16,276.06	8,236.69	1.98	0.99	0.00	15,304.25	7,736.77	1.98
0.7	0.99	0.00	16,276.06	11,445.60	1.42	1.00	0.00	15,304.25	10,784.34	1.42

about 1.2 million training images and has a large size of images ($224 \times 224 \times 3$). We use pre-trained model, InceptionV3 [59] implemented in Keras that includes 93 activation layers.

To answer RQ6,we conduct the same experiment of Section 4.1 (RQ1) and Section 4.4 (RQ4) using a large dataset (ImageNet) and a corresponding complex model (InceptionV3). Figure 7 shows the changes of accuracy when the validation inputs are sorted by their SA values and selected to be fed into the model. ImageNet clearly shows the same trends with others as shown in Figure 2: it is hard to correctly classify the images with high SA values (blue dots), and vice versa (red dots).

Next, we compare the trade-off between correct ranks and execution time by sampling the training set. Table 6 shows the results: column t_{all} represents the execution time in seconds when SA values are computed from the original training set, and column t_{sample} represents the execution time in seconds from the sampled training set. Note that the results of LSA with sample rate of 0.1 are omitted since sampling 10% of the training set leads to a failure of KDE. SA computation using the original training set takes more than 4 hours (> 14,400 seconds) in both LSA and MDSA. Further analysis reveals that the forward-passes of the training inputs to extract ATs account for most of the computational load: 67% and 72% of execution time are responsible for forward-passes. The sample rate of 0.1 achieves the highest speed-up 9.46x as expected, and in this case, MDSA

computation only takes 26 minutes (1,617 seconds). However, the correlation is less positive (0.75) than sample rates of 0.3, 0.5, and 0.7 whose correlation coefficients are greater than 0.96.

Answer to RQ6: Average speed-ups have similar trends with sample rate of (i.e., if we use 50% of the training set, then we achieve 50% of speed-up). When we use 10% of the training set in ImageNet, MDSA only takes 26 minutes, suggesting that SA is applicable and scalable to the complex model with large dataset.

4.7 Correlation between SAs (RQ7)

Table 7. Spearman's rank correlation between LSA, DSA, and MDSA.

DNN Model	*LSA − DSA*		*LSA − MDSA*		*DSA − MDSA*	
	ρ	p	ρ	p	ρ	p
MNIST	0.75	0.00	0.67	0.00	0.71	0.00
SVHN	0.86	0.00	0.94	0.00	0.85	0.00
CIFAR-10	0.10	0.00	0.85	0.00	0.23	0.00
CIFAR-100	0.79	0.00	0.94	0.00	0.76	0.00
Dave-2	-	-	0.73	0.00	-	-
Chauffeur	-	-	0.88	0.00	-	-

If the three types of SA are interchangeable, we can choose to use any type of SA to get the ranks, or, as MDSA is shown to be faster than LSA and MDSA, we can pick MDSA as a default option. We provide an answer to this question using Spearman's rank correlation between two types of SA on the same input set. There are three pairs of SAs: LSA-DSA, LSA-MDSA, and DSA-MDSA. To answer RQ7, we report the correlation coefficient and p-value of those three pairs using the default setting specified in Table 2. Note that DSA is not computed on Dave-2 and Chauffeur, so the results of LSA-DSA and DSA-MDSA are omitted for those two models.

The results in Table 7 show that all correlations are positive: average ρ is 0.72 and average p-value is 0.00. The relationship of DSA of CIFAR-10 with LSA and MDSA has a relatively weaker positive correlation than other cases (i.e., p-value is low), though LSA-MDSA of CIFAR-10 has a strong positive correlation. Figure 5 explains this: Layer Id 6 (activation_6) of CIFAR-10's LSA and MDSA have similar ROC-AUC scores (e.g., scores of BIM-B are 99.63% and 99.27% respectively, and C&W are 50.26% and 48.86%), but DSA has different patterns of ROC-AUC scores (e.g., the score of BIM-B is 65.61% and C&W is 79.88%). In this case, unlike LSA and MDSA, DSA interprets the input surprises differently. But, it does not mean that DSA fails to capture the relative surprise of inputs.

Answer to RQ7: All SAs have positive rank correlation for each other. LSA and MDSA show similar trends, but for CIFAR-10, DSA alone shows different aspects of input surprise. Given the results of the correlation analysis, we recommend the use of MDSA in a general application scenario as it is compuationally most efficient (as can be seen in the results for RQ6).

4.8 Retraining Guidance (RQ8)

To evaluate whether SA can guide additional training of existing DL systems with the aim of improved accuracy against adversarial examples, RQ8 asks whether SA can guide the selection

Table 8. Retraining guided by SA: we sample 100 inputs from four increasingly wider ranges of SA: $[L, L+\frac{U-L}{4}]$, $[L, L+\frac{2(U-L)}{4}]$, $[L, L+\frac{3(U-L)}{4}]$, and $[L, U]$, 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 tends to improve the retraining accuracy.

DNN Model	SA	R	FGSM		BIM-A		BIM-B		JSMA		C&W	
			μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2
MNIST		$\emptyset$	12.29	-	0.87	-	0.87	-	9.75	-	9.66	-
	LSA	1/4	23.37	7.81	96.14	0.27	**21.36**	5.73	73.43	14.23	11.95	0.26
		2/4	26.83	3.84	95.91	0.17	18.51	20.37	73.90	11.38	11.84	0.19
		3/4	**28.45**	3.78	96.12	0.27	16.21	4.89	73.57	8.70	11.91	0.19
		4/4	26.87	7.44	**96.15**	0.70	14.89	17.85	**75.87**	5.89	**15.17**	0.43
	DSA	1/4	24.57	10.03	95.63	0.98	10.88	5.59	73.52	6.17	13.06	1.65
		2/4	28.03	5.55	96.05	0.56	**23.11**	20.16	74.02	8.40	12.46	1.00
		3/4	27.67	9.28	**96.31**	0.19	17.06	23.93	75.58	6.93	12.46	0.91
		4/4	**28.86**	7.17	96.07	0.47	14.39	12.24	**75.93**	5.63	**15.63**	0.54
	MDSA	1/4	24.16	4.88	95.79	0.33	**21.82**	4.78	73.78	9.72	**16.41**	0.30
		2/4	26.24	3.91	**96.05**	0.21	19.83	9.79	74.67	7.14	15.73	0.33
		3/4	27.51	5.81	96.05	0.27	18.19	22.30	74.39	14.25	15.10	0.63
		4/4	**28.61**	7.74	95.97	0.78	14.67	17.24	**75.87**	6.56	15.87	0.60
SVHN		$\emptyset$	3.70	-	0.00	-	0.00	-	0.34	-	0.00	-
	LSA	1/4	6.91	1.47	47.02	7.94	0.78	0.28	60.35	8.13	85.85	0.82
		2/4	6.43	0.73	**48.10**	14.17	1.23	0.51	60.42	6.87	86.01	0.72
		3/4	6.42	0.78	47.76	3.08	1.35	0.78	60.20	14.68	**86.16**	0.65
		4/4	**7.07**	2.81	46.62	5.34	**1.40**	0.31	**62.52**	5.29	86.03	0.89
	DSA	1/4	7.09	2.12	46.99	10.81	0.56	0.30	**62.93**	5.84	85.86	1.08
		2/4	**7.61**	1.86	48.46	7.26	1.08	0.37	62.17	4.29	85.79	0.71
		3/4	7.03	2.99	48.14	6.01	1.49	1.57	61.78	8.22	85.92	1.17
		4/4	6.49	0.74	**48.47**	5.81	**1.52**	0.98	62.23	12.92	**86.10**	0.41
	MDSA	1/4	6.87	1.36	47.81	11.58	0.95	0.35	60.77	8.88	85.85	1.04
		2/4	6.71	1.23	**48.63**	5.28	1.04	0.41	61.25	6.40	**86.43**	0.38
		3/4	6.35	0.94	47.92	5.54	1.27	0.38	61.19	13.01	85.90	0.82
		4/4	**7.01**	4.23	48.40	6.20	**1.29**	0.19	**61.40**	9.45	85.74	1.47
CIFAR-10		$\emptyset$	8.01	-	0.00	-	0.00	-	2.61	-	2.32	-
	LSA	1/4	12.77	0.87	32.33	6.55	0.23	1.02	34.93	5.75	36.23	16.10
		2/4	13.05	2.14	**33.71**	9.50	0.00	0.00	34.56	7.37	44.01	8.55
		3/4	13.35	2.39	32.86	7.39	0.00	0.00	**35.13**	7.28	**46.15**	12.20
		4/4	**14.45**	3.31	33.45	7.48	**0.64**	2.67	34.55	9.19	45.69	4.53
	DSA	1/4	**15.63**	2.30	30.58	7.88	0.00	0.00	34.62	5.03	36.19	10.96
		2/4	14.78	5.92	31.69	6.27	0.00	0.00	35.70	3.86	44.11	5.06
		3/4	13.98	2.72	**34.52**	8.05	**0.26**	1.30	36.22	5.49	**46.05**	8.55
		4/4	14.02	2.00	33.92	4.46	0.00	0.00	**36.40**	5.35	45.99	6.26
	MDSA	1/4	13.65	1.25	31.74	4.86	0.00	0.00	34.28	4.83	34.70	13.66
		2/4	13.82	2.05	32.83	3.78	0.00	0.00	34.57	5.13	43.51	7.06
		3/4	13.57	1.32	33.44	4.53	**0.51**	2.46	35.73	2.86	45.24	8.06
		4/4	**14.18**	2.63	**34.30**	4.11	0.26	1.30	**36.47**	8.13	**46.30**	2.24
CIFAR-100		$\emptyset$	13.14	-	44.36	-	17.55	-	-	-	0.00	-
	LSA	1/4	12.15	1.48	**41.01**	1.55	17.27	1.44	-	-	26.75	5.98
		2/4	12.88	1.33	40.47	2.93	16.93	1.64	-	-	27.71	7.41
		3/4	12.57	1.79	39.70	1.06	17.25	1.22	-	-	28.83	10.40
		4/4	**13.02**	1.75	40.59	1.53	**17.55**	1.17	-	-	**29.47**	8.14
	DSA	1/4	12.79	0.77	**41.30**	1.22	16.96	1.04	-	-	26.86	11.73
		2/4	**12.96**	1.74	40.67	1.38	17.41	2.13	-	-	27.54	8.71
		3/4	12.22	2.50	40.05	2.61	**17.47**	1.17	-	-	28.08	8.63
		4/4	12.61	1.74	39.71	2.06	17.12	1.12	-	-	**28.35**	8.92
	MDSA	1/4	12.34	1.04	**41.34**	1.66	16.86	1.30	-	-	27.09	7.18
		2/4	12.52	1.29	40.04	2.99	17.21	1.51	-	-	27.34	9.43
		3/4	12.59	0.76	39.42	2.55	17.23	1.21	-	-	**29.03**	8.84
		4/4	**12.94**	1.70	39.86	2.53	**17.31**	1.92	-	-	28.87	6.72

Table 9. RQ8 results of Dave-2 (extension of Table 8)

DNN Model	SA	R	SingleOcc μ	SingleOcc σ^2	MultiOcc μ	MultiOcc σ^2	Light μ	Light σ^2
Dave-2		$\emptyset$	0.1520	-	0.1290	-	0.2490	-
	LSA	1/4	0.0708	0.0007	0.0649	0.0002	0.0763	0.0002
		2/4	0.0804	0.0005	0.0623	0.0001	0.0668	0.0002
		3/4	0.0759	0.0010	0.0615	0.0000	0.0616	0.0001
		4/4	**0.0589**	0.0001	**0.0600**	0.0000	**0.0602**	0.0000
	MDSA	1/4	0.0801	0.0005	0.0628	0.0002	0.0678	0.0001
		2/4	0.0770	0.0003	0.0636	0.0001	0.0690	0.0002
		3/4	0.0691	0.0002	0.0653	0.0001	0.0609	0.0001
		4/4	**0.0622**	0.0007	**0.0597**	0.0000	**0.0579**	0.0000

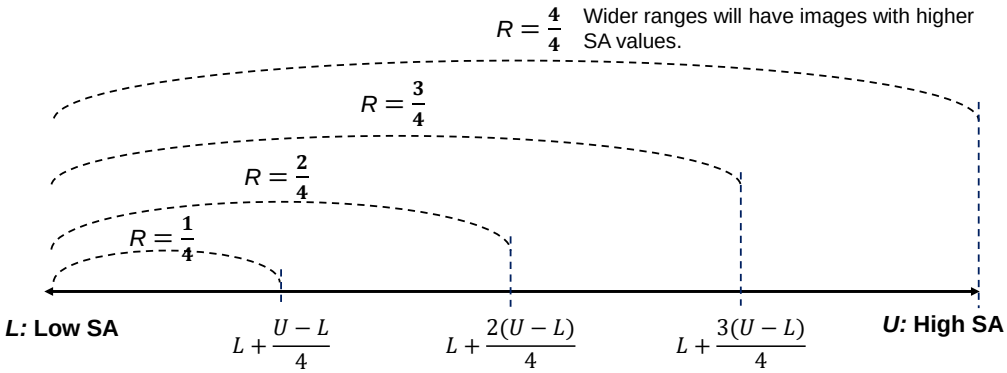


Fig. 8. We divide the SA space into four ranges and randomly select 100 images for each range. We expect that wider range will contain diverse inputs.

of input for additional training. From the adversarial examples and synthesised inputs for these models[9], we choose four sets of 100 images from four different SA ranges.

As shown in Figure 8, we divide the range of SA $[L, U]$ into four overlapping subsets: the first subset including the low 25% SA values ($[L, L + \frac{U-L}{4}]$), the second including the lower half ($[L, L + \frac{2(U-L)}{4}]$), the third including the lower 75% ($[L, L + \frac{3(U-L)}{4}]$), and finally the entire range, $[L, U]$. 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 classifiers, MSE for Dave-2) against the entire adversarial and synthetic inputs, respectively. We expect retraining with more diverse subset will result in higher performance.

Tables 8 and 9 show the impact of SA-based guidance for retraining of the 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 the performance of the DL system before retraining.

Overall, there are 66 retraining configurations (3 SA types × 4 DL systems × 5 adversarial attack strategies, and 2 SA type × 1 DL system × three input synthesis methods), each of which is evaluated against four SA ranges with 20 repetitions to cope with the stochastic nature of the training process. The columns μ and σ^2 contain the mean and variance of the observed performance metric (i.e.,

[9]We could not resume training of Chauffeur model for additional five epochs, which is why it is absent from RQ8.

the highest accuracy for classifiers, the lowest MSE for Dave-2). The best average performance is typeset in bold.

The full range, $\frac{4}{4}$, produces the best retraining performance for 39 configurations, followed by $\frac{3}{4}$ (11 configurations), $\frac{1}{4}$ (8 configurations), and $\frac{2}{4}$ (8 configurations). In particular, the $\frac{4}{4}$ range is effective against all attack strategies applied to Dave-2, regardless of the type of SA. However, we would like to nuance our conclusion about retraining from the previous work [29] and more cautiously state that SA alone likely is not enough to effectively guide retraining. Despite the overall count-based aggregation which suggests $\frac{4}{4}$ to be the best range, it is difficult to find a consistent pattern between the SA ranges, attack strategies, and SA types. Further, even when the full range outperforms others in terms of retraining performance, the effect sizes tend to remain small.

Taken altogether, the new results force us to focus on what SA works best for, which is to capture the out-of-boundness of a new input: the capacity of SA to do so has been demonstrated not only with RQ1, but also in its application to other domains [30, 31]. However, the set of inputs SA detects to be out-of-bound with respect to the training data may not necessarily be the ideal set of inputs to be added for further training of the model. There is no guarantee that these inputs represent what the training data lack to result in a better trained DNN model. The fact that the input has a high SA value is necessary for it to belong to the set of inputs needed for further retraining (simply because it suggests that this input was not part of the training), but not sufficient (because there may be other inputs that are more important for retraining). At least, our experiments to date have not found a large benefit and more research will be needed to fully understand the effects. For example, the amount and level of initial training, before retraining, might be a strong co-dependent factor, either because for an untrained model any additional training might be beneficial or for a well-trained model, there is a little differential benefit regardless of how training examples are chosen. Further work is needed on the quantification of the exact contribution made by a specific set of training data in order to both precisely measure the retraining impact and define the clear baseline.

Answer to RQ8: While using input sampled from the full SA range results in the largest number of retraining results that outperform the original model accuracy, the results lack a consistent trend that would support the effectiveness of SA guided retraining. We thus caution against only using SA to guide retraining: it provides some guidance but there are additional factors and further research on SA-guided retraining is needed before more detailed guidance can be provided.

5 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 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 SA 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.

6 RELATED WORK

We structure related work into four different categories: test adequacy and testing techniques for DL systems, related work based on SA, the general techniques that aim to quantify uncertainly in DL systems, and active learning.

6.1 Testing DL Systems

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 [10, 46, 68]. 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. [26] proposed a verification technique that can automatically generate counter-examples that violate this assumption. Pei et al. introduced DeepXplore [51], 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 [61]. Guided by several coverage criteria, DeepHunter [66] proposed fine grained mutation strategy to generate semantic-preserving tests and outperformed DeepTest by achieving higher coverages.

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 [51]. 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 [41]. 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 [43]. SC is different from DeepCT as SA 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 [42].

6.2 Surprise Adequacy in the Literature

SA has been widely adopted as a test adequacy criterion in the literature since its introduction. We briefly present some of the relevant work here, based on the recent work of Weiss et al. [64] that contains a comprehensive review of SA. Out of 105 papers included in the review paper, 15 directly study SA. We choose three representative papers [9, 20, 67] and discuss them below. Further, to augment the review paper with more up-to-date related work, we considered citations of our conference paper that appeared after January 2021, and choose five out of 52 papers that directly studies SA [22, 31, 39, 48, 63]. In both cases, the remaining work simply mentions SA as background or related work.

6.2.1 SA as a baseline metric. SA has been frequently compared with newly proposed test adequacy metrics or testing techniques in several studies as a baseline. PRIMA [63] adopted mutation testing to prioritise a test set that can kill (i.e., force them to produce different model outputs) more mutated models. LSC and DSC were compared to PRIMA and the results suggested that PRIMA

performed better than them. On the same problem, TestRank [39] used DSA as a baseline to evaluate bug-revealing capabilities of test inputs. DeepImportance [20] introduced an importance driven test adequacy that assesses the semantics of neuron influence to enable layer-wise functional understanding of DL models. Their results showed that SA is complementary to DeepImportance. The wide adoption of SA as a baseline shows its importance as a standard test adequacy metrics.

6.2.2 SA for validating existing approaches. Some existing work use SA to validate newly proposed DNN robustness metrics via their relationship to SA. Chen et al. [9] conducted an empirical study that shows SC is correlated with the ratio of the number of adversarial test inputs. Among the considered coverage criteria, LSC performed the best in terms of both effectiveness and efficiency. Yan et al. [67] investigated whether the coverage criteria are correlated with model robustness when SA is used to select adversarial examples. The results show that the coverage criteria do not have a monotonic relationship between the coverage improvement and model robustness.

6.2.3 SA as a part of the proposed approach. SA has also been used as an essential component of other approaches. Kim et al. [31] evaluated the efficacy of SA on the Natural Language Processing (NLP) tasks such as text classification or Named Entity Recognition (NER). In that work, SA is extended to consider the multimodality of the NLP tasks and used to prioritise the inputs that can reveal the incorrect behaviour of the model. The results showed that multimodal variants of SA successfully applied for NER tagging task. Ouyang et al. [48] used DSA to detect corner cases of DL models. The results demonstrated that DSA performs well on detecting corner cases in both MNIST and industrial dataset. Guerriero et al. [22] introduced DeepEST, an operational testing method to find failing tests efficiently. LSA and DSA were adopted to make two variants of DeepEST, all of which provided accurate estimates compared to existing sampling-based techniques. SINVAD [28] is a test data generation technique for DNN. It first constructs a latent semantic space of the given input domain using Variational AutoEncoders (VAEs), and performs a search of the space guided by SA metrics: any datapoint in the latent space can be evaluated for its SA value after being processed by the decoder of the VAE. Consequently, SINVAD can successfully synthesise new test inputs with high SA values.

6.3 Uncertainty Quantification for DL Systems

Technically, SA can be regarded as one of the uncertainty quantifications for DL systems. We therefore summarise two commonly used metrics in the ML literature [19]. First, Monte Carlo Dropout (MC Dropout) [17] utilises the dropout layers in DNN to formulate them as Bernoulli random variables. By activating the dropout layers in a testing time with several forward passes of the given input, MC Dropout builds an output distribution that enables us to compute the uncertainty of the input. One of the advantages of using MC Dropout is that it does not need any modifications to the model. However, it requires the model to have dropout layers for regularisation. Compared to SA that demands pre-calculation such as KDE or covariance matrix, its computational costs entirely occurs at testing time. Second, having training several models with the same structure but different initial seeds, Deep Ensembles [36] combines the predictions of several models to capture the uncertainty. Despite its simple and effective methodology, the need to train multiple models from the scratch results in substantial amounts of computational cost, which may hinder practical applications.

6.4 Active Learning

Active learning (AL) aims at selecting minimal and valuable data to update the model [53]. With the recent advances of DL that need a large labeled dataset, AL for DL has emerged. As the goal of AL is to prioritise the inputs by their values to label them earlier, common approaches have been to

measure their uncertainties by MC Dropout [18] or to use a small proxy model for efficiency [11]. SA can substitute the procedure of uncertainly quantification like MC Dropout. But if there are frequent model updates, SA computation will be inefficient since every time model is updated, it requires a pre-computation such as AT extraction and KDE. Future research directions include a focus on the utility of SA in the AL setting.

7 CONCLUSION

We propose Surprise Adequacy (SA), which is a test adequacy metric for DL systems. It aims to quantitatively measure how *surprising* each new test input is when compared to the training data. The intuition is that, if a new input is surprising, i.e., more out-of-bound with respect to the distribution of the training data, the input is also more likely to cause unexpected model behaviour. SA realises this intuition by representing each input with the resulting model behaviour: given an input, we capture the Activation Trace, the collection of neuron outputs produced by the model under test. Consequently, if the AT vector of a new input is far away from the AT vectors of the training data, the model is likely to behave in a different and unexpected way.

Our empirical evaluation shows that SA can capture the degree of out-of-distribution of inputs accurately, and can perform as a good indicator of how DL systems will behave against new, previously unseen inputs. This paper extends the original conference paper with additional research questions that present more comprehensive empirical evaluations of SA using one of the largest image classification benchmarks, ImageNet. Further, we consider the scalability of SA, and evaluate its sensitivity to its internal hyperparameters.

8 ACKNOWLEDGMENT

Jinhan Kim and Shin Yoo have been supported by the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921), as well as the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2018-0-00769, Neuromorphic Computing Software Platform for Artificial Intelligence Systems).

REFERENCES

[1] [n.d.]. Autonomous driving model: Chauffeur. https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/chauffeur.
[2] [n.d.]. The Udacity open source self-driving car project. https://github.com/udacity/self-driving-car.
[3] Paul Ammann and Jeff Offutt. 2016. *Introduction to Software Testing.* Cambridge University Press.
[4] Yoshua Bengio, Grégoire Mesnil, Yann Dauphin, and Salah Rifai. 2012. Better Mixing via Deep Representations. *CoRR* abs/1207.4404 (2012). arXiv:1207.4404 http://arxiv.org/abs/1207.4404
[5] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).
[6] Nicholas Carlini and David Wagner. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security.* ACM, 3–14.
[7] Nicholas Carlini and David A. Wagner. 2016. Towards Evaluating the Robustness of Neural Networks. *CoRR* abs/1608.04644 (2016). arXiv:1608.04644 http://arxiv.org/abs/1608.04644
[8] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. 2015. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision.* 2722–2730.
[9] Junjie Chen, Ming Yan, Zan Wang, Yuning Kang, and Zhuo Wu. 2020. Deep neural network test coverage: How far are we? *arXiv preprint arXiv:2010.04946* (2020).
[10] T. Y. Chen, F.-C. Kuo, T. H. Tse, and Zhi Quan Zhou. 2004. Metamorphic Testing and Beyond. In *Proceedings of the International Workshop on Software Technology and Engineering Practice (STEP 2003).* 94–100.
[11] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. 2019. Selection via proxy: Efficient data selection for deep learning. *arXiv preprint arXiv:1906.11829*

(2019).
[12] Zhihua Cui, Fei Xue, Xingjuan Cai, Yang Cao, Gai-ge Wang, and Jinjun Chen. 2018. Detection of Malicious Code Variants Based on Deep Learning. *IEEE Transactions on Industrial Informatics* 14, 7 (2018), 3187–3196.
[13] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 248–255.
[14] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. 2013. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1915–1929.
[15] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. 2017. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410* (2017).
[16] Robert Feldt, Simon Poulding, David Clark, and Shin Yoo. 2016. 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)*. 223–233.
[17] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*. 1050–1059.
[18] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep bayesian active learning with image data. In *International Conference on Machine Learning*. PMLR, 1183–1192.
[19] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. 2021. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342* (2021).
[20] Simos Gerasimou, Hasan Ferit Eniser, Alper Sen, and Alper Cakan. 2020. Importance-driven deep learning system testing. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 702–713.
[21] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*. http://arxiv.org/abs/1412.6572
[22] Antonio Guerriero, Roberto Pietrantuono, and Stefano Russo. 2021. Operation is the hardest teacher: estimating DNN accuracy looking for mispredictions. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 348–358.
[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[24] 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. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine* 29, 6 (2012), 82–97.
[25] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 arXiv:https://doi.org/10.1162/neco.1997.9.8.1735
[26] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. 2017. Safety Verification of Deep Neural Networks. In *Computer Aided Verification*, Rupak Majumdar and Viktor Kunčak (Eds.). Springer International Publishing, Cham, 3–29.
[27] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. 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)*, Vol. 1. 1–10.
[28] Sungmin Kang, Robert Feldt, and Shin Yoo. 2020. SINVAD: Search-based Image Space Navigation for DNN Image Classifier Test Input Generation. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (SBST 2020)*. 521–528.
[29] Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding Deep Learning System Testing using Surprise Adequacy. In *Proceedings of the 41th International Conference on Software Engineering (ICSE 2019)*. IEEE Press, 1039–1049.
[30] Jinhan Kim, Jeongil Ju, Robert Feldt, and Shin Yoo. 2020. Reducing DNN Labelling Cost using Surprise Adequacy: An Industrial Case Study for Autonomous Driving. In *Proceedings of ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE Industry Track) (ESEC/FSE 2020)*. 1466–1476.
[31] Seah Kim and Shin Yoo. 2021. Multimodal Surprise Adequacy Analysis of Inputs for Natural Language Processing DNN Models. In *2021 IEEE/ACM International Conference on Automation of Software Test (AST)*. IEEE, 80–89.
[32] Alex Krizhevsky. 2009. *Learning Multiple Layers of Features from Tiny Images*. Technical Report. University of Toronto.
[33] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2014. The CIFAR-10 dataset. *online: http://www.cs.toronto.edu/kriz/cifar.html* (2014).
[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.
[35] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *CoRR* abs/1607.02533 (2016). arXiv:1607.02533 http://arxiv.org/abs/1607.02533

[36] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2016. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474* (2016).

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

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

[39] Yu Li, Min Li, Qiuxia Lai, Yannan Liu, and Qiang Xu. 2021. TestRank: Bringing Order into Unlabeled Test Instances for Deep Learning Tasks. *arXiv preprint arXiv:2105.10113* (2021).

[40] Stijn Luca, Peter Karsmakers, Kris Cuppens, Tom Croonenborghs, Anouk Van de Vel, Berten Ceulemans, Lieven Lagae, Sabine Van Huffel, and Bart Vanrumste. 2014. Detecting rare events using extreme value statistics applied to epileptic convulsions in children. *Artificial Intelligence in Medicine* 60, 2 (2014), 89 – 96. https://doi.org/10.1016/j.artmed.2013.11.007

[41] 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. 2018. DeepGauge: Comprehensive and Multi-Granularity Testing Criteria for Gauging the Robustness of Deep Learning Systems. *CoRR* abs/1803.07519 (2018). arXiv:1803.07519 http://arxiv.org/abs/1803.07519

[42] Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu, Chao Xie, Li Li, Yang Liu, Jianjun Zhao, et al. 2018. DeepMutation: Mutation Testing of Deep Learning Systems. *arXiv preprint arXiv:1805.05206* (2018).

[43] Lei Ma, Fuyuan Zhang, Minhui Xue, Bo Li, Yang Liu, Jianjun Zhao, and Yadong Wang. 2018. Combinatorial Testing for Deep Learning Systems. *arXiv preprint arXiv:1806.07723* (2018).

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

[45] Prasanta Chandra Mahalanobis. 2018. Reprint of: Mahalanobis, P.C. (1936) "On the Generalised Distance in Statistics.". *Sankhya A* 80, 1 (2018), 1–7.

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

[47] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. 2011. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

[48] Tinghui Ouyang, Yoshinao Isobe, Vicent Sanz Marco, Jun Ogata, Yoshiki Seo, and Yutaka Oiwa. 2021. AI robustness analysis with consideration of corner cases. In *2021 IEEE International Conference on Artificial Intelligence Testing (AITest)*. IEEE, 29–36.

[49] 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. 2018. Technical Report on the CleverHans v2.1.0 Adversarial Examples Library. *arXiv preprint arXiv:1610.00768* (2018).

[50] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. 2015. The Limitations of Deep Learning in Adversarial Settings. *CoRR* abs/1511.07528 (2015). arXiv:1511.07528 http://arxiv.org/abs/1511.07528

[51] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17)*. ACM, New York, NY, USA, 1–18. https://doi.org/10.1145/3132747.3132785

[52] Simon Poulding and Robert Feldt. 2017. Generating Controllably Invalid and Atypical Inputs for Robustness Testing. In *Software Testing, Verification and Validation Workshops (ICSTW), 2017 IEEE International Conference on*. IEEE, 81–84.

[53] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. 2021. A survey of deep active learning. *ACM Computing Surveys (CSUR)* 54, 9 (2021), 1–40.

[54] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115, 3 (2015), 211–252.

[55] Wojciech Samek, Grégoire Montavon, Sebastian Lapuschkin, Christopher J. Anders, and Klaus-Robert Müller. 2021. Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications. *Proc. IEEE* 109, 3 (2021), 247–278. https://doi.org/10.1109/JPROC.2021.3060483

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

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

[58] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on*

computer vision and pattern recognition. 1–9.

[59] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2818–2826.

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

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

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

[63] Zan Wang, Hanmo You, Junjie Chen, Yingyi Zhang, Xuyuan Dong, and Wenbin Zhang. 2021. Prioritizing Test Inputs for Deep Neural Networks via Mutation Analysis. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 397–409.

[64] M. Weiss, R. Chakraborty, and P. Tonella. 2021. A Review and Refinement of Surprise Adequacy. In *2021 IEEE/ACM Third International Workshop on Deep Learning for Testing and Testing for Deep Learning (DeepTest)*. IEEE Computer Society, Los Alamitos, CA, USA, 17–24. https://doi.org/10.1109/DeepTest52559.2021.00009

[65] Wikipedia. 2021. Computational complexity of mathematical operations. https://en.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations Online; accessed 22-October-2021.

[66] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. 2019. Deephunter: a coverage-guided fuzz testing framework for deep neural networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 146–157.

[67] Shenao Yan, Guanhong Tao, Xuwei Liu, Juan Zhai, Shiqing Ma, Lei Xu, and Xiangyu Zhang. 2020. Correlations between deep neural network model coverage criteria and model quality. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 775–787.

[68] Shin Yoo. 2010. Metamorphic Testing of Stochastic Optimisation. In *Proceedings of the 3rd International Workshop on Search-Based Software Testing (SBST 2010)*. 192–201.

[69] Shin Yoo. 2019. SBST in the Age of Machine Learning Systems: Challenges Ahead. In *Proceedings of the 12th International Workshop on Search-Based Software Testing (SBST '19)*. IEEE Press, Piscataway, NJ, USA, 2–2. https://doi.org/10.1109/SBST.2019.000-2

[70] Shin Yoo and Mark Harman. 2012. Regression Testing Minimisation, Selection and Prioritisation: A Survey. *Software Testing, Verification, and Reliability* 22, 2 (March 2012), 67–120.

[71] Hong Zhu, Patrick A. V. Hall, and John H. R. May. 1997. Software Unit Test Coverage and Adequacy. *ACM Comput. Surv.* 29, 4 (Dec. 1997), 366–427. https://doi.org/10.1145/267580.267590