Test Selection for Deep Learning Systems

WEI MA, MIKE PAPADAKIS, ANESTIS TSAKMALIS, MAXIME CORDY, and YVES LE TRAON, Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg

Testing of deep learning models is challenging due to the excessive number and complexity of the computations involved. As a result, test data selection is performed manually and in an ad hoc way. This raises the question of how we can automatically select candidate data to test deep learning models. Recent research has focused on defining metrics to measure the thoroughness of a test suite and to rely on such metrics to guide the generation of new tests. However, the problem of selecting/prioritising test inputs (e.g. to be labelled manually by humans) remains open. In this paper, we perform an in-depth empirical comparison of a set of test selection metrics based on the notion of model uncertainty (model confidence on specific inputs). Intuitively, the more uncertain we are about a candidate sample, the more likely it is that this sample triggers a misclassification. Similarly, we hypothesise that the samples for which we are the most uncertain, are the most informative and should be used in priority to improve the model by retraining. We evaluate these metrics on 5 models and 3 widely-used image classification problems involving real and artificial (adversarial) data produced by 5 generation algorithms. We show that uncertainty-based metrics have a strong ability to identify misclassified inputs, being 3 times stronger than surprise adequacy and outperforming coverage related metrics. We also show that these metrics lead to faster improvement in classification accuracy during retraining: up to 2 times faster than random selection and other state-of-the-art metrics, on all models we considered.

Additional Key Words and Phrases: Deep Learning Testing, Software Testing, Software Engineering

ACM Reference Format:
Wei Ma, Mike Papadakis, Anestis Tsakmalis, Maxime Cordy, and Yves Le Traon. 2019. Test Selection for Deep Learning Systems. *ACM Trans. Softw. Eng. Methodol.* , (November 2019), 23 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Deep Learning (DL) systems [18] are capable of solving complex tasks, in many cases equally well or even better than humans. Such systems are attractive because they learn features by themselves and thus require only minimum human knowledge. This property makes DL flexible and powerful. As a result, it is increasingly used and integrated with larger software systems and applications.

Naturally, the adoption of this technology introduces the need for its reliable assessment. In classical, code-based software engineering, this assessment is realised by means of testing. However, the testing of DL systems is challenging due to the complexity of the tasks they solve. In order to effectively test the DL system, we need to identify corner cases that challenge the learned properties. In essence, DL system testing should focus on identifying the incorrectly learned properties and lead to data that can make the systems deviate from their expected behaviour.

Authors' address: Wei Ma, wei.ma@uni.lu; Mike Papadakis, michail.papadakis@uni.lu; Anestis Tsakmalis, anestis.tsakmalis@gmail.com; Maxime Cordy, maxime.cordy@uni.lu; Yves Le Traon, yves.letraon@uni.lu, Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg.

To this end, recent research [14, 22, 28, 39] focuses on designing test coverage metrics to measure how thoroughly a set of inputs tests the model. Most of them (e.g. those inspired by neuron coverage [22, 28, 39]) focus on activating various combinations of neurons and on generating new test inputs to increase the proportion of those combinations. Others (e.g. [14]) argue that DL models reflect particular properties of the training data and their behaviour is determined based on the knowledge they acquired during the training phase. As such, they promote coverage metrics that measure the differences across the inputs rather than within the model.

In this paper, we focus on the problem of *selecting* test inputs. In DL, test input selection adresses a practical concern: which subset of unlabelled data one should label to discover faults in DL models. This goal differs from previous methods that either measure test thoroughness or generate (artificial) test inputs. Our aim is to help with the manual effort involved when labelling test data (deciding the class of an input). Evidently, data labelling involves extensive manual analysis (due to the large amounts of data required by DL systems), which could be reduced by using only the most likely fault revealing test data. Therefore, to reduce this burden, we aim to identify metrics that help selecting the most interesting (likely to trigger misclassifications) test data.

In the recent years, the scientific community has come up with metrics to support the testing of deep learning models (read more in Section 2). However, these metrics were studied in different contexts (e.g. adversarial example generation and detection) or testing scenarios (e.g. measuring test thoroughness). All in all, the capability of existing metrics to pinpoint misclassified inputs remains unclear. Thus, our contribution is the evaluation of these metrics from a new perspective; the test input selection. That is serving the purpose of selecting inputs that maximise the chances to trigger misclassifications. To our knowledge, our work is the only one to study this test selection goal.

We postulate that effective metrics should select inputs that are more likely to trigger misclassifications by the model. Experience has shown that classification mistakes are incorrectly learned properties that happen due to overlapping and closely located regions of the feature space. Therefore, the cases residing between the learned categories and their boundaries are the most likely to be the incorrectly learned ones. In view of this, rather than aiming at the coverage of specific neurons [28] or test data diversity [14], we aim at the data with properties that are close to the model boundaries. In other words, we argue that test selection should be directed towards the boundaries of the learned classes.

Accordingly, we consider test selection metrics based on the notion of model uncertainty (low confidence on specific inputs). Intuitively, the more uncertain a model is about a candidate sample, the more likely the sample will trigger misclassification. Similarly, the samples for which the model is the most uncertain are also the most informative ones for learning (should be used to improve the models by retraining). As suggested by Gal and Ghahramani [7], we approximate uncertainty by the variance in prediction probabilities observed by randomly dropping neurons in the network multiple times [37]. We also use the actual model's output probabilities as a certainty measure, which we also combine with dropout variance.

We evaluate these metrics using image classifiers on three datasets, i.e., MNIST, Fashion-MNIST and CIFAR-10, and compare them with respect to previously proposed metrics, i.e., the surprise adequacy metrics [14] and several metrics based on neuron coverage [22, 29]. In particular, we investigate the correlation between the metrics and misclassification on both real and artificial (adversarial) data. We show that uncertainty-based metrics have medium to strong correlations with misclassification when considering real data, and strong correlations when considering a mix of real and adversarial data. In contrast, metrics based on coverage have weak or no correlation, while surprise adequacy has weak correlation.

Interestingly, our results reveal that when considering misclassifications, the prediction probabilities (a simple certainty metric overlooked by previous work) is among the most effective metrics, significantly outperforming surprise

adequacy and coverage metrics. However, in the case of retraining, a combination of the dropout variance with prediction probabilities outruns the other metrics in terms of faster improvements in classification accuracy.

Our contributions can be summarised by the following points:

- We propose to perform test input selection based on a set of metrics measuring model uncertainty, i.e., the confidence in classifying unseen inputs correctly. We consider the variance caused by multiple dropouts (i.e. the distribution of the model's output under dropouts), the model's prediction probabilities, and metrics combining the two.
- We perform the first study on the fault revealing ability (misclassification triggering ability) of test selection metrics. We demonstrate that the uncertainty-based metrics challenge the DL models and have medium to strong correlations with misclassification (correlations of approximately 0.3 on real data and 0.6 on real plus adversarial ones). Furthermore, we show that these metrics are significantly stronger than the surprise adequacy and coverage related metrics.
- We also show that model uncertainty can guide the selection of informative input data, i.e., data that are capable of increasing classification accuracy. In particular, when retraining the DL model based on the selected data, the best performing uncertainty metrics achieve up to 2 times faster improvement over random selection and coverage metrics.

2 RELATED WORK

Testing of learning systems is typically performed by selecting a dedicated test set randomly from available labelled data [43]. When an explicit test dataset is not available, one can rely on cross-validation [15] to use part of the training set to anticipate how well the learning model will generalize to new data. These established procedures, however, often fail to cover many errors. For instance, research work in adversarial learning has shown that applying minor perturbations to data can make models give a wrong answer [10]. Nowadays, those adversarial samples remain hard to detect and bypass many state-of-the-art detection methods [4]. In order to achieve better testing, multiple approaches have been proposed in the recent years. We distinguish four categories of contributions: (i) metrics for measuring the coverage/thoroughness of a test set; (ii) generation of artificial inputs; (iii) metrics for selecting test data; (iv) detection of adversarial samples.

DeepXplore, proposed by Pei et al. [28], comprises both a coverage metric and a new input generation method. It introduces neuron coverage as the first white-box metric to assess how well a test set covers the decision logic of DL models. Leaning on this criterion, DeepXplore generates artificial inputs by solving a joint optimization problem with two objectives: maximizing the behavioural differences between multiple models and maximizing the number of activated neurons. Pei et al. report that DeepXplore is effective at revealing errors (misclassifications) that were undetected by conventional ML evaluation methods. Also, retraining with additional data generated by DeepXplore increases the accuracy of the models. On some models, the increase is superior (1 to 3%) to an increase obtained by retraining with data generated by some adversarial technique [10]. Pei et al. also show that randomly-selected test data and adversarial data achieve smaller neuron coverage than data generated by DeepXplore. While they assume that more neuron coverage leads to better testing, future research showed that this metric is inadequate [14, 23].

In a follow-up paper, Tian et al. [40] propose DeepTest as another method to generate new inputs for autonomous driving DL models. They leverage metamorphic relations that hold in this specific context. Like DeepXplore, DeepTest utilizes the assumption that maximising neuron coverage leads to more challenging test data. The authors show that

different image transformations lead to different neuron coverage values and infer that neuron coverage is an adequate metric to drive the generation of challenging test data. However, this claim was not supported by empirical evidence.

A related method was proposed by [25]. It works under the assumption that there is a recurring defect in the DNN model, such that inputs from one particular class are often misclassified. The method is based on differential analysis to identify features/neurons responsible for this defect, so as to fix the model. On the contrary, the uncertainty metrics we propose to use are independent of the particular class of the inputs and are lightweight (they do not require more expensive computations/analyses). Overall, we see our work (lightweight metrics to detect and fix punctual errors) as complementary to [25] (in-depth analysis to fix recurring defects).

DeepGauge [23] and DeepMutation [24] are two test coverage metrics proposed by Ma et al. With DeepGauge, they push further the idea that higher coverage of the structure of DL models is a good indicator of the effectiveness of test data. They show, however, that the basic neuron coverage proposed previously is unable to differentiate adversarial attacks from legit test data, which tends to indicate the inadequacy of this metric. As a result, they propose alternative criteria with different granularities, i.e. at the neuron level and the layer level. Their experiments reveal that replacing original test inputs by adversarial ones increases the coverage of the model wrt. DeepGauge's criteria. However, they did not assess the capability of their criteria to prioritize test samples likely to trigger misclassifications.

Similarly, DeepMutation leverages the mutation score used in traditional mutation testing to DL models. From a given model, it generates multiple mutant models by applying different operators such as, e.g., neuron switch or layer removal. Then, they define the mutation score of a test input as the number of mutants that it killed (i.e. those that yield a different classification output for the test input than the original model). Thus, the mutation score assesses how sensitive the model is wrt. the test inputs rather than how these cover the neurons of the network.

Nevertheless, both DeepGauge and DeepMutation measure the coverage/thoroughness of a test set but do not aim at selecting individual inputs to undergo labelling. Moreover, a recent study [14] has shown that neuron coverage criteria do not necessarily increase when more misclassified/surprising inputs are added. Although our contribution starts from the idea of analyzing the sensitivity of the model by mutating it (using dropouts), our scope differs in that we examine how mutations can actually support the selection of data for testing and improving by retraining the model.

Later on, Ma et al. [21] proposed DeepCT, a test coverage metric suggesting that within a given layer, all tuples of neurons should be covered by at least one test input. They also propose an algorithm to generate artificial inputs to cover as many t-wise interactions as possible. They show, first, that random test selection cannot cover a large part ($> 65\%$) of the 2-wise neuron interactions. Second, they show that retraining on the inputs generated by their algorithm allows the detection of up to 10% of adversarial samples that could not be detected by retraining on randomly selected inputs. An alternative proposed by Xiaofei Xie et al. [44] is DeepHunter, a fuzzing-based test generation algorithm to hunt defects in DL models. The fuzzing is guided by the coverage metrics defined in DeepXplore [28] and DeepGauge [23]. Their evaluation shows that the fuzzing algorithm manages to increase the intended coverage metrics. Both DeepCT and DeepHunter focus on generating artificial inputs and are not directed towards the selection of available challenging data for testing and retraining.

Most recently, Kim et al. [14] proposed metrics for test coverage *and* test selection. They highlight the fact that coverage criteria fail to discriminate the added value of *individual* test inputs and are therefore impractical for test selection. They argued that test criteria should guide the selection of individual inputs and eventually help improving the DL models' performance. As a consequence, they propose *surprise adequacy* as a metric that measures how surprised the model is when confronted with a new input. More precisely, the degree of surprise measures the dissimilarity of the neurons' activation values when confronted to this new input wrt. the neurons' activation values when confronted to

the training data. They hypothesise that a set of test inputs is preferable for both testing and retraining when it covers a diverse range of surprise values. In other words, a good test set should range from very similar to very different inputs to those observed during training. Kim et al. show experimentally that (a) surprise coverage is sensitive to adversarial samples and (b) retraining on such samples yields better improvements in accuracy. In our paper, we show that model uncertainty is more effective at triggering misclassifications and improving the accuracy of the models than input diversity. Nevertheless, surprise adequacy is complementary to our work since it aims for a diversity of surprise degrees and thus better applies to models that are not yet well-trained, while uncertainty metrics aim at reinforcing well-trained models against inputs that remain challenging.

Uncertainty of DL systems was theoretically studied by a number of authors. Gal and Ghahramani [7] proved that the variance of the softmax function resulting from neuron dropout is a good estimate for model uncertainty. Kendall and Gal [13] propose a model to capture jointly *aleatoric* uncertainty (originating from noise inherent to the observations) and *epistemic* uncertainty (induced by the fact that the model is not trained on all possible data). The latter type is what is commonly referred to as *model uncertainty*, which can be captured by dropout variance [13].

In [36], Smith and Gal provide evidence that uncertainty metrics can be used to detect adversarial samples, although this does not hold for data that are far away from the training set. Compared with this line of work, our contribution is to study uncertainty from a new perspective and how well it can achieve the purpose of selecting inputs that trigger misclassifications. Akin notions were used by Feinman et al. [6] to detect adversarial samples. This method uses Kernel density estimate of neuron activation (similar to likelihood-based surprise adequacy [14]) and Bayesian uncertainty based on dropout variance (similar to what we use here). Wang et al. [42] proposed computing how much the labels predicted by a model change when (after training) this model is slightly altered. They showed that adversarial inputs are more likely to increase the label change rate. A purely Bayesian generative adversarial method is proposed in [33], where the adversarial sample generator and the discriminator are Bayesian neural networks trained with stochastic gradient Hamiltonian Monte Carlo sampling. Specifically, the discriminator network is capable of efficiently detecting adversarial samples because of the competition-based structure, which forces learning to be a repeated contest between the generator and the discriminator. Another detection method based on uncertainty is that of Sheikholeslami and Giannakis [34], which promotes scalability by measuring uncertainty on sampled hidden layers. Pinder's master thesis [30] reports other experiments demonstrating that adversarial images yield a significantly greater uncertainty than original images. In other settings, though, Grosse et al. [11] show that there exist adversarial examples which do not affect the uncertainty of the model.

Overall, the aforementioned studies aim at detecting adversarial examples. Compared to them our goal is to select examples that are most likely to be misclassified, be these real or adversarial (studied separately and together). Another key difference is that we consider both well-classified and misclassified adversarial examples. Doing so, we demonstrate that the metrics are even more sensitive to the noise introduced by adversarial algorithms than they are to the classification results, which is in line with the results of [30, 34, 36, 42]. Also, we consider a broad range of metrics, including (but not limited to) multiple metrics that approximate uncertainty. Specifically, compared to [42], the dropout variance we use is more fine-grained than label change rate as it is computed over classification probabilities.

3 MOTIVATION AND PROBLEM DEFINITION

DL systems are known for their capability to solve problems with large input space, by learning statistical patterns from the available data. This is typically the case in computer vision applications (the use case we consider in our experiments), where the goal is to classify images correctly between two classes or more. An interesting characteristics

of such problems is that data (i.e. images) usually proliferate. However, to be useful these data also need to be labelled (associated with their correct class). They can then be used either to test the model (check that the DL model predicts the correct label of the image) to (re-)train it (feed new labelled data into the model to improve its predictions). Data labelling is typically performed by manual or non-systematic procedures. This means that DL system developers have to put a lot of effort to produce a DL model of acceptable quality. Our key motivation is to support them in this task by optimizing the effort-reward ratio.

We formulate this problem as follows. Let us assume that developers have access to an arbitrarily large number of inputs (i.e. data without label) and that they can afford to label only an arbitrary number of k inputs. We name *test input selection* the problem of selecting the k most effective inputs to label. Here, effectiveness is measured differently depending on the considered DL development phase:

- When *testing* the trained DL model, effectiveness is measured in terms of fault-revealing ability. Misclassifications being the most obvious defects that occur in DL models, selection methods should maximize the *number of inputs misclassified by the model.*
- When *re-training* the model, the inputs to label should be selected in order to *maximize the performance gain* (e.g. maximize accuracy).

Our goal is to address the test input selection problem by providing objective and measurable ways of identifying effective test inputs. Thus, We thus aim at answering the following two questions:

- *How can we select test inputs to challenge (trigger misclassifications in) a Deep Learning model?*
- *How can we select additional training inputs to improve the performance (increase classification accuracy) of a Deep Learning model?*

We answer these questions by conducting an empirical study evaluating the adequacy of different selection metrics. Based on our results, practitioners can identify which metrics they should use given their goal (testing or retraining their model) and their context (e.g. with or without adversarial data).

4 TEST SELECTION METRICS

Our overall goal is to consider a range of metrics related to misclassification and study how effective they are in selecting misclassified inputs. Particularly, we hypothesize that model uncertainty is strongly correlated to misclassification, that is, the more uncertain a model is towards a certain input, the more it is likely to misclassify this input. Dropout variance is the most concrete and simple way to estimate the prediction uncertainty [7], instead of dealing with Bayesian models whose training can often be quite demanding. Other metrics we consider come from the machine learning testing literature (e.g. neuron coverage [29], surprise adequacy [14] and etc.) – please refer to Section 4.1 for details. Although they were not meant for test input selection (as we define in this paper), they relate to the general concept of test adequacy and remain commonly used to drive test generation. As such, they appear as natural baselines worth of consideration. Since we aim at experimenting with the test input selection problem (our new perspective), we should use the most relevant metrics. The remaining metrics represent complementary properties: neuron boundary coverage is a generalization of neuron coverage; silhouette coefficient is an alternative to surprise adequacy; Kullback-Leiber divergence is another way of measuring dropout variance that estimates the uncertainty of the model, therefore we consider them as well.

4.1 Metrics Derived from the Machine Learning Testing Literature

The metrics we retain from the literature were initially directed to measuring test thoroughness ("coverage") rather than test input selection. Still, they can be used for the latter purpose by selecting the test inputs that reach the highest coverage.

4.1.1 Neuron Coverage. Neuron coverage was first proposed in [29] to drive the generation of artificial inputs. It is simply defined as the percentage of neurons that were activated by at least one input of the test set. Accordingly, we define the coverage of a single input as follows.

Definition 4.1. Let D be a trained DL model composed of a set N of neurons. The **Neuron Coverage** (NC) of the input x wrt. D is given by

$$NC(x) = \frac{|\{n \in N \mid activate(n, x)\}|}{|N|}$$

where $activate(n, x)$ holds true if and only if n is activated when passing x into D.

The above definition determines the coverage of an input independently of the other inputs. One can instead define the *additional* neurons covered by x that were not covered during training.

Definition 4.2. Let D be a DL model composed of a set N of neurons and trained on a set T of inputs . The **Additional Neuron Coverage** (ANC) of the input x wrt. D is given by

$$ANC(x) = \frac{|\{n \in N \mid activate(n, x) \wedge \forall x' \in T : \neg activate(n, x')\}|}{|N|}.$$

We also consider the other test thoroughness metrics that extend the concept of NC: **K-Multisection Neuron Coverage** (KMNC), **Neuron Boundary Coverage** (NBC) and **Strong Neuron Activation Coverage** (SNAC). A detailed description of those metrics is omitted and can be found in their original paper [22].

4.1.2 Surprise Metrics. The next two test selection methods are based on surprise adequacy [14]. In their recent paper, Kim et al. proposed two metrics to measure the surprise of a DL model D when confronted to a new input x. The first one is based on kernel density estimation and aims at estimating the relative likelihood of x wrt. the training set in terms of the activation values of D's neurons. To reduce computational cost, only the neurons of a specified layer are considered [14].

Definition 4.3. Let D be a DL model trained on a set T of inputs. The **Likelihood-based Surprise Adequacy** (LSA) of the input x wrt. D is given by

$$LSA(x) = \frac{1}{|A_{N_L}(T_{D(x)})|} \sum_{x_i \in T_{D(x)}} K_H(\alpha_{N_L}(x) - \alpha_{N_L}(x_i))$$

where $\alpha_{N_L}(x)$ is the vector recording the activation values of the neurons in layer L of D when confronted to x, $T_{D(x)}$ is the subset of T composed of all inputs of the same class as x, $A_{N_L}(T_{D(x)}) = \{\alpha_{N_L}(x_i) \mid x_i \in T_{D(x)}\}$, and K_H is the Gaussian kernel function with bandwidth matrix H.

As an alternative, Kim et al. proposed a second metric that relies on Euclidean distance instead of kernel density estimation. The idea is that inputs that are closer to inputs of other classes and farther from inputs of their own class are considered as more surprising. This degree of surprise is measured as the quotient between the distance of the closest input x_a of the same class as x and the distance of the closest input x_b from any other class. Like the LSA metric, all these distances are considered in the activation value space of the inputs.

Definition 4.4. Let D be a DL model trained on a set T of inputs. The **Distance-based Surprise Adequacy** (DSA) of the input x wrt. D is given by

$$DSA(x) = \frac{||\alpha_N(x) - \alpha_N(x_a)||}{||\alpha_N(x_a) - \alpha_N(x_b)||}$$

where

$$x_a = \underset{\{x_i \in T_{D(x)}\}}{\operatorname{argmin}} ||\alpha_N(x) - \alpha_N(x_i)||$$

$$x_b = \underset{\{x_j \in T \setminus T_{D(x)}\}}{\operatorname{argmin}} ||\alpha_N(x_a) - \alpha_N(x_j)||$$

and where $D(x)$ is the predicted class of x by D and $\alpha_N(x)$ is the activation value vector of all neurons of D when confronted to x.

LSA and DSA rely on measuring the density or distance of the clusters formed by the different classes. In essence, any metrics that can discriminate the consistency of clusters might be also candidate metrics for test selection. For instance, we propose to use **Silhouette Coefficient** [32] as another metric. Its advantages are its stability and a limited range of output, i.e. $[-1, 1]$ (whereas LSA and DSA have a priori no upper bound).

Definition 4.5. Let D be a DL model trained on a set T of inputs. **Silhouette Coefficient** (Si) of the input x wrt. D is given by

$$Si(x) = \begin{cases} 1 - \dfrac{a_x}{b_x}, & a_x < b_x \\ 0, & a_x = b_x \\ \dfrac{b_x}{a_x} - 1, & a_x > b_x \end{cases}$$

where

$$a_x = \frac{1}{|T_{D(x)} \setminus \{x\}|} \sum_{x_i \in T_{D(x) \setminus \{x\}}} ||\alpha_N(x) - \alpha_N(x_i)||,$$

$$b_x = \min_{D(x) \neq D(x_i)} \frac{1}{|T_{D(x_i)}|} \sum_{x_i \in T_{D(x_i)}} ||\alpha_N(x) - \alpha_N(x_i)||.$$

4.2 Model Uncertainty Metrics

The starting point for suggesting the use of other selection metrics lies in the hypothesis that test inputs are more challenging (i.e. more likely to be misclassified) as they engender more uncertainty (as opposed to surprise) in the considered DL model.

The prediction probabilities of the classes returned by the model are immediate metrics that can indicate how challenging a particular input is. Indeed, one can intuitively state that more challenging inputs are classified with lower probability, that is, the highest prediction probability output by the model is low. Using prediction probabilities as metrics is mostly overlooked by the literature but remains efficient, as our experiments show.

Definition 4.6. Let D be a trained DL model. The **maximum probability score** of the input x wrt. D is given by

$$MaxP(x) = \max_{i=1:C} p_i(x)$$

where C is the number of classes and $p_i(x)$ is the prediction probability of x to class i according to D.

Recently, it has been mathematically proven that neuron dropout [38] can be used to model uncertainty [7, 13]. Dropout was initially proposed as a technique to avoid overfitting in neural networks by randomly dropping (i.e. deactivating) neurons during training [38]. This is achieved by incorporating so-called *dropout* layers into the network, which dynamically simulate the deactivation of neurons during a forward pass.

Dropout can be used to estimate the uncertainty of a trained model wrt. a new input x [7]. More precisely, the uncertainty is estimated by passing k times the input x into the model (wherein dropout layers were added) and computing the variance of the resulting prediction probabilities over x. Intuitively, while prediction probabilities can be visualized as the distances of x from the class boundaries estimated by the model, dropout variance represents the variance of these distances induced by the uncertainty of our knowledge about the exact locations of the class boundaries. The motivation towards using dropout variance rather than classification probabilities stems from the observation that some modern deep neural networks are poorly *calibrated* [12], i.e. their prediction probabilities do not correlate with their likelihood of correct classification.

In addition to being a good estimate of model uncertainty [7, 36], dropouts are cheap to compute thanks to their implementation as dropout layers (as opposed to really generating k altered models from the original one, which would be more expensive given the high number of neurons that models include). Formally, let D be an original, well-trained model equipped with dropout layers to simulate random dropping, such that each neuron is dropped out with probability (i.e. dropout rate) r. Given an input x, we pass x into the network k times and denote by $p_i^j(x)$ the prediction probability of x to class i output on the j-th pass. We also denote by $P_i(x) = \{p_i^j(x) | 1 \leq j \leq k\}$ the multiset of prediction probabilities of x assigned to class i resulting from the k passes. Then, the variance of $P_i(x)$ is a good estimate of the uncertainty of D when classifying x in class i [7].

Following our hypothesis that uncertain inputs are more likely to be misclassified, we define a metric derived from dropout variance to assess how much an input x is challenging to D. This *variance score* is a macroscopic view of dropout variance in that it averages the uncertainty of D wrt. x over all classes.

Definition 4.7. The **variance score** of the input x is given by

$$Var(x) = \frac{1}{C} \sum_{i=1}^{C} \text{var}(P_i(x))$$

where C is the number of classes and var denotes the variance function.

A drawback of this metric is that it does not consider the prediction probabilities (thus, the actual distance to class boundaries). To overcome this, we propose a relative metric that normalizes the variance score with the highest probability output by D.

Definition 4.8. The **weighted variance score** of the input x is

$$Var_w(x) = \Big(MaxP(x)\Big)^{-1} \cdot Var(x)$$

While variance and weighted variance scores of x can be regarded as quantitative measures of the uncertainty of the model wrt. x, we also propose a nominal alternative. Instead of the variance of prediction probabilities, we focus on the actual class predictions produced by the different mutant models, that is, the classes with the highest probability scores. We construct a normalized histogram of these k class predictions and we compare their distribution with that of a theoretical, worst-case, completely uncertain model, where the class predictions are uniformly distributed over all

Table 1. Datasets and DNN models used in our experiments.

Dataset	Model	Optim. method	# layers (convolution, dense)	# neurons (kernels)	Accuracy (%)
MNIST / Fashion-MNIST	MLP	RMSprop	3	1034	98.51 / 89.33
	LeNet	SGD + Nesterov	5	236	99.05 / 89.99
	WLeNet	Adam	5	310	99.57 / 92.88
CIFAR-10	NetInNet	SGD + Nesterov	9	1418	90.77
	VGG10	SGD + Momentum	10	1674	91.99

classes. Thus, in this worst case, the number of mutants predicting that an input x belongs to class i is approximately given by $\frac{k}{C}$.

To compare the actual class prediction distribution with the worst-case distribution, we rely on the discrete version of Kullback-Leibler (KL) divergence. When the uncertainty of D is high (i.e. the mutants often disagree), the KL divergence is low.

Definition 4.9. The **Kullback-Leibler score** of the input x is

$$KL(x) = \sum_{i=1}^{C} H_i \ln \frac{H_i}{Q_i}$$

where i is the class label, H is the normalized histogram, or frequencies, of the class predictions for x resulting from the k dropouts and Q is the uniform distribution (i.e. $Q_i = \frac{1}{C}$).

5 EXPERIMENTAL SETUP

5.1 Datasets and Models

We consider three image recognition datasets. *MNIST* [19] contains handwriting number data of 10 classes and is composed of 70,000 images (60,000 for training and 10,000 for testing). *Fashion-MNIST* [1] has clothing images classified into 10 classes and is also composed of 70,000 images, 60,000 and 10,000 for training and testing. *CIFAR-10* [2] has 10 categories of images (cats, dogs, trucks etc.). The dataset has 50,000 images for training and 10,000 for testing.

The three datasets are widely used in research and considered as a good baseline to observe key trends, in addition to requiring affordable computation cost. Furthermore, the diversity of these datasets (in terms of classes and domain concepts) and the used models provides confidence about the generality of our results.

Thanks to the efforts of the research community, these classification problems can today be solved with high accuracy. This characteristic makes these datasets challenging and relevant for us; triggering misclassifications in accurate models is much harder than in inaccurate ones. Indeed, test selection is more beneficial as interesting tests (i.e. misclassified inputs) are rarer within the set of test candidates (thus, when the model has high accuracy). Considering models with low accuracy is not relevant, as in this case it is more likely to select misclassified examples.

Table 1 shows the characteristics of the models we use in our experiment. For MNIST and Fashion-MNIST, we use three simple networks, Multi-Layer Perceptron (MLP), LeNet [19] and a modified version LeNet with more kernels (WLeNet). For CIFAR-10, we use tow complex networks, NetInNet[20] and 10-layer *VGG16* [35] – named VGG10 – obtained by removing the top layers and inserting a batch normalization layer after each convolutional layer. The models were trained for 50 epochs (MNIST), 150 epochs (Fashion-MNIST) and 300 epochs (CIFAR-10). The last column in Table 1 shows the best accuracy of the models (over the epochs) when trained on the whole training set.

5.2 Objectives and Methodology

5.2.1 Test Selection with Real Data. Our first step is to assess the abilities of the studied metrics to select test inputs that can challenge a given DL model D. To achieve this, we determine the correlation between the metrics and misclassification. We encode the 'correctness' of the prediction of D for one particular input x as a binary variable b_x (well- or miss-classified). For each metric, we compute the Kendall correlation between the score given by the metric to all test inputs and their corresponding binary variables. We used Kendall correlation because, being an ordinal association metric, it focuses on how well the metrics rank the misclassified inputs first (irrespective of the actual score values). Thus, if one has to select a limited number of inputs to label and test, one should select the inputs of higher ranking (irrespective of their score). This is important for the test input selection problem, as the metrics should allow an effective selection regardless of the actual budget of inputs to label. Given that we study the correlation between the numeric values returned by the selection metrics and misclassification, random selection is not a relevant baseline in this experiment, as it is not a metric. Actually, since we consider high-accuracy models which yield few misclassifications, random selection is inherently ineffective for test input selection.

5.2.2 Test Selection with Adversarial Data. To investigate the fault revealing ability with a larger number of data, we augment our test data with adversarial samples. Adversarial data result from the successive application of minor perturbations to original data with the aim of deceiving a classifier. Adversarial samples have been of major concern [17] and test selection metrics should be robust against them. Moreover, previous research [14, 22] also used adversarial data. To craft adversarial data, we use five established adversarial data generation algorithms: Fast Gradient Sign Method (FGSM) [10], Jacobian-based Saliency Map Attack (JSMA) [27], DeepFool (DF) [26], Basic Iterative Method(BIM) [16] and Carlini-Wagner (CW) [5]. We apply each algorithm separately and add its generated images to the original test set. Thus, we obtain five new datasets. All algorithms except CW generated 10,000 images and thus doubled the size of the test set. Regarding CW, we made it generate 1,000 adversarial images as it is much slower, which necessitated more than one day of computing on our HPC infrastructure. Still, CW remains interesting to study as it is known to apply less perturbation to the original image. Nevertheless, we use the procedure mentioned previously to compute the Kendall correlation between test selection metrics and misclassification, in the five datasets using both original and adversarial images generated by five algorithms.

To further investigate the sensitivity of the metrics on adversarial data, we apply FGSM and CW on 600 images randomly picked from the datasets. As these algorithms iteratively generate adversarial images (by altering them, introducing noise), until they succeed, most (66% to 100%) of the intermediate images are well-classified. Thus, we store 3,603 intermediate images generated by FGSM and 18,148 generated by CW over the iterations and compute their score according to the studied metrics. Since we start from well-classified images and the adversarial generation algorithms work incrementally (at each iteration they generate images that are closer to misclassification) the number of the iteration at which an input was generated reflects its distance from the starting point (a later iteration step signifies a higher chance for misclassification). Therefore, a monotonic relation between the metrics and the iterations signifies a good capability to quantify the likelihood of misclassification (caused by the adversarial images ultimately produced at the end of the process). Hence, we compute the Spearman correlation (statistical test measuring the monotonicity between the studied variables) between the score of the metrics, of these intermediate images, and the iteration number that they were produced.

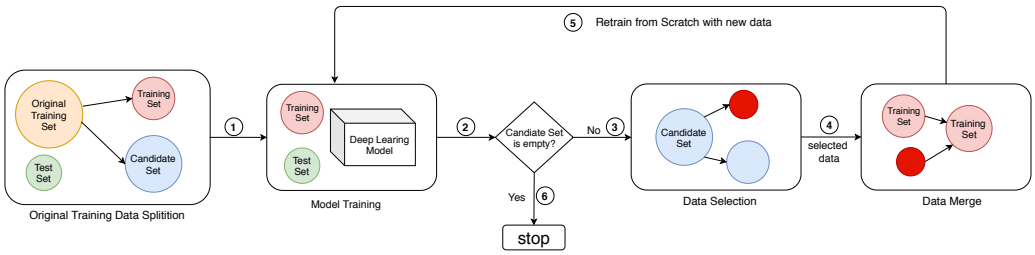


Fig. 1. Flow process of Data Selection for Retraining

We also study the correlation between the metrics and misclassification when using only adversarial inputs. We pick the intermediate images and the final adversarial images and associate them with a binary variable (well- or miss-classified) and compute the Kendall correlations between the binary variables and the metrics.

5.2.3 Data Selection for Retraining. Having studied the adequacy of the metrics to select challenging test inputs, we focus next on how much the metrics can help selecting additional training data effectively. That is, we study whether augmenting the training set with data selected based on the metrics can lead to faster improvement. To do this, we set up an iterative retraining process as shown in Figure 1.

At first, we randomly split the original training set into an initial training set of 10,000 images and a candidate set that contains the remaining images. The test set remains untouched. In the first round, we train the model using only the initial training set and compute its accuracy on the test set. After finishing training, we use the best model that we get (over the training epochs) to compute the test selection metrics on the remaining candidate data.

Then, we add (without replacement) a batch of 5, 000 new images (selected by the metrics) from the candidate set to the current training set. The selected images are those that have the highest uncertainty (i.e. lowest score for Si, KL and MaxP, highest score for Var and Var_w) or surprise (LSA or DSA) or coverage (i.e. higher NC, ANC, KBNC, NBC and SNAC). We retrain the models from scratch using the whole augmented training set for a sufficient number of epochs to guarantee convergence (150 epochs for MNIST and Fashion-MNIST, 300 epochs for CIFAR-10) so that we can fairly analyze the different methods. Although incremental training (which re-applies the training algorithms on the current model using the new data) is more efficient computation-wise, current implementations (e.g. within scikit-learn[1]) have shown that incremental training creates biases towards the oldest data, as training algorithms (like stochastic gradient descent) give less importance to new examples over time (due to a decreasing learning rate). This difference can be significant if the new data follow a different distribution than the old data. Thus, incremental training is used when assuming minor concept drift, while training from scratch is used in cases where such assumption cannot be made (or may not hold). This is actually the reason why many companies retrain from scratch [8]. Nevertheless, the purpose of our experiments is not to find the computationally optimal way to incorporate additional training samples, but to make sure that by incorporating additional training samples in the most exhaustive way (to make sure that the model has been trained well enough) yields the best possible (even by a small difference) results. Thus, to avoid making such assumptions, we followed the conservative approach of retraining from scratch to make sure that the old and new training data are treated equally.

[1] https://scikit-learn.org/0.15/modules/scaling_strategies.html#id2

We repeat the process for multiple rounds, until the candidate set is empty. We ensure that test data are never used during training or retraining. To account for random variations in the training process, we repeat the experiments three times and report, for each obtained model, the average (over the three repetitions) of the best accuracy obtained (over all epochs).

To assess the effectiveness of each metric, we observe the evolution of the validation loss and accuracy wrt. the independent test data over the retraining process. Effective metrics should yield fast increases in validation accuracy and fast decreases in validation loss. Although validation loss has usually a strong negative correlation with validation accuracy, it is still important to study both, e.g. to detect overfitting models.

Previous studies have shown that increasing the accuracy of models that already have a high accuracy (> 90%) may result in decreasing their robustness to adversarial attacks [41]. Thus, it is possible that some test selection metrics increase the accuracy during retraining but reduce robustness. To assess this, we also compute the empirical robustness [26] (based on FGSM and 100 randomly picked images) of all models at all retraining rounds. This allows us to check whether there exists a compromise between the metrics (i.e. privileging accuracy or robustness).

5.3 Implementation

We tooled our approach on top of Keras and Tensorflow, and used the library Foolbox [31] to generate adversarial images. Our tool, together with our replication package, is available online.[2] The Model training phase was performed on GPU K80 and GPU Volta V100.

When the considered DL model (e.g. VGG16) does not use standard dropout for training, we implement it as Lambda layers. When the DL model includes Dropout layers for training (e.g. WLeNet), we simply keep those Dropout layers working during testing. Thus, we do not alter the model computations per se but rather alternating the model behaviour through the dropout layers.

In every case, the hyperparameters of our method are the dropout rate r (probability of dropping-out neurons) and the number k of forward passes of any input into the network (while randomly dropping neurons on the fly). If r is too large, the original model competence will be significantly degraded, which will result in poor quality. On the contrary, a small r results in too small variations for our method to perform well. For VGG10, we experimentally set the drop rate to 0.25. On the other models, we keep the drop rate as 0.35. We also set $k = 50$ as it appeared as a good trade-off between the estimation of the variance and computation cost.

For Surprise Adequacy and Neuron Coverage, we use the source code available on Github[3]. We re-implemented NC and Neuron-Level Coverage (NLC) based on the source code in an efficient batch-computing way. Finally, we use the IBM robustness framework[4] to compute the empirical robustness of the retrained models.

6 RESULTS

6.1 Test Selection with Real Data

Table 2 shows the Kendall correlation between the metrics and misclassification. We observe that KL, Var, Var_w, MaxP and DSA have a medium degree of correlation, meaning that they can lead to valuable test data, i.e., those causing misclassification. Conversely, we observe that both LSA and Si have a weak correlation to misclassification. All these correlations are statistically significant with a p-value lower than 10^{-05}. Metrics based on neuron coverage have weak

[2] https://github.com/TestSelection/TestSelection
[3] https://github.com/coinse/sadl, https://github.com/ARiSE-Lab/deepTest
[4] https://github.com/IBM/adversarial-robustness-toolbox/

Table 2. Kendall correlation between misclassification and the metrics on the *original (real) test data*. Overall, KL and MaxP achieve the strongest correlations.

Model (Dataset)	KL	Var	Var_w	MaxP	DSA	LSA	Si	NC	ANC	KMNC	SANC	BNC
MLP (MNIST)	**0.3556**	0.1626	0.1627	0.2804	0.1606	0.1396	0.1326	0.0454	N/A	-0.0181	0.0006	0.0006
LeNet (MNIST)	0.1253	0.1263	0.1282	**0.2076**	0.1273	0.1157	0.1143	-0.0430	0.0017	-0.0464	0.0144	0.0105
WLeNet (MNIST)	**0.2774**	0.0903	0.0905	0.0990	0.0900	0.0766	0.0872	0.0722	N/A	0.0308	0.0047	0.0068
MLP (Fashion)	**0.4107**	0.3222	0.3339	0.3519	0.322	0.0414	0.1639	0.2112	0.0113	0.1876	0.0171	0.0177
LeNet (Fashion)	0.3048	0.2784	0.3103	**0.3369**	0.3059	0.1542	0.2601	-0.0234	0.0058	0.0598	0.0057	0.0067
WLeNet (Fashion)	**0.4156**	0.2896	0.2962	0.3133	0.2941	0.0949	0.2551	0.2429	-0.0139	0.1760	0.0205	0.0188
VGG10 (CIFAR)	0.3404	0.2818	0.2911	**0.3647**	0.2661	0.1964	0.2560	-0.0101	-0.0087	-0.1092	0.0404	0.0329
NetInNet (CIFAR)	**0.4366**	0.3337	0.3371	0.339	0.3208	0.2076	0.3302	-0.0236	N/A	-0.0228	0.0132	0.0228

to very weak correlations. In particular, we could not compute the correlation of ANC for three models because, in these models, the test set does not cover new neurons that the training set did not cover already.

Overall, these results indicate that KL, Var, Var_w, $MaxP$ and DSA correlate better with misclassifications. More precisely, KL and $MaxP$ appear as the best metrics for test selection, being up to 3 times more correlated to misclassification than all the other metrics. In the particular case of the MNIST dataset models, these two metrics are the only ones to achieve a medium correlation, while the other metrics reach only weak or very weak correlations. Nevertheless, the best correlations we found are only moderate, meaning that none of the metrics can perfectly distinguish between well-classified and misclassified inputs.

> KL and $MaxP$ are the best metrics to discriminate misclassified real data from well-classified ones. They correlate with misclassification up to 3 times more than the others.

6.2 Test Selection with Adversarial Data

6.2.1 Mix of Real and Adversarial Data. Figure 2 records the Kendall correlation between the metrics and misclassification when the original test data set is augmented with each adversarial dataset (separately). Interestingly, the correlations of all metrics are stronger than they were with real data only. KL, Var, Var_w, MaxP, Si and DSA now have a strong degree of correlation with misclassifications in most cases, while the correlations of LSA reach only moderate levels. Overall, MaxP achieves the stronger correlations regardless of the algorithm used, except for the WLeNet-MNIST where KL gets even stronger correlations. As for metrics based on neuron coverage, their correlations remain weak overall and can be positive or negative. Even on a model-by-model basis, no general tendency tends to appear. Quite surprisingly, NC performs better than KMNC, NBC and SNAC, and even achieves moderate/strong correlations on the two WLeNet models.

Nonetheless, the overall strengthening of the correlations can be explained by the fact that adversarial images have some form of artificial noise that the classifier never experienced during training. This noise makes the classifier less confident on how to deal with them, a fact reflected by the metrics. We also infer that adversarial data do not form a challenging scenario to evaluate test selection methods (as performed by related work [14, 22, 28]). Given that the adversarial images are misclassified, it is possible that the metrics are even more appropriate to distinguish adversarial and real data than they are to differentiate well- and miss-classified data.

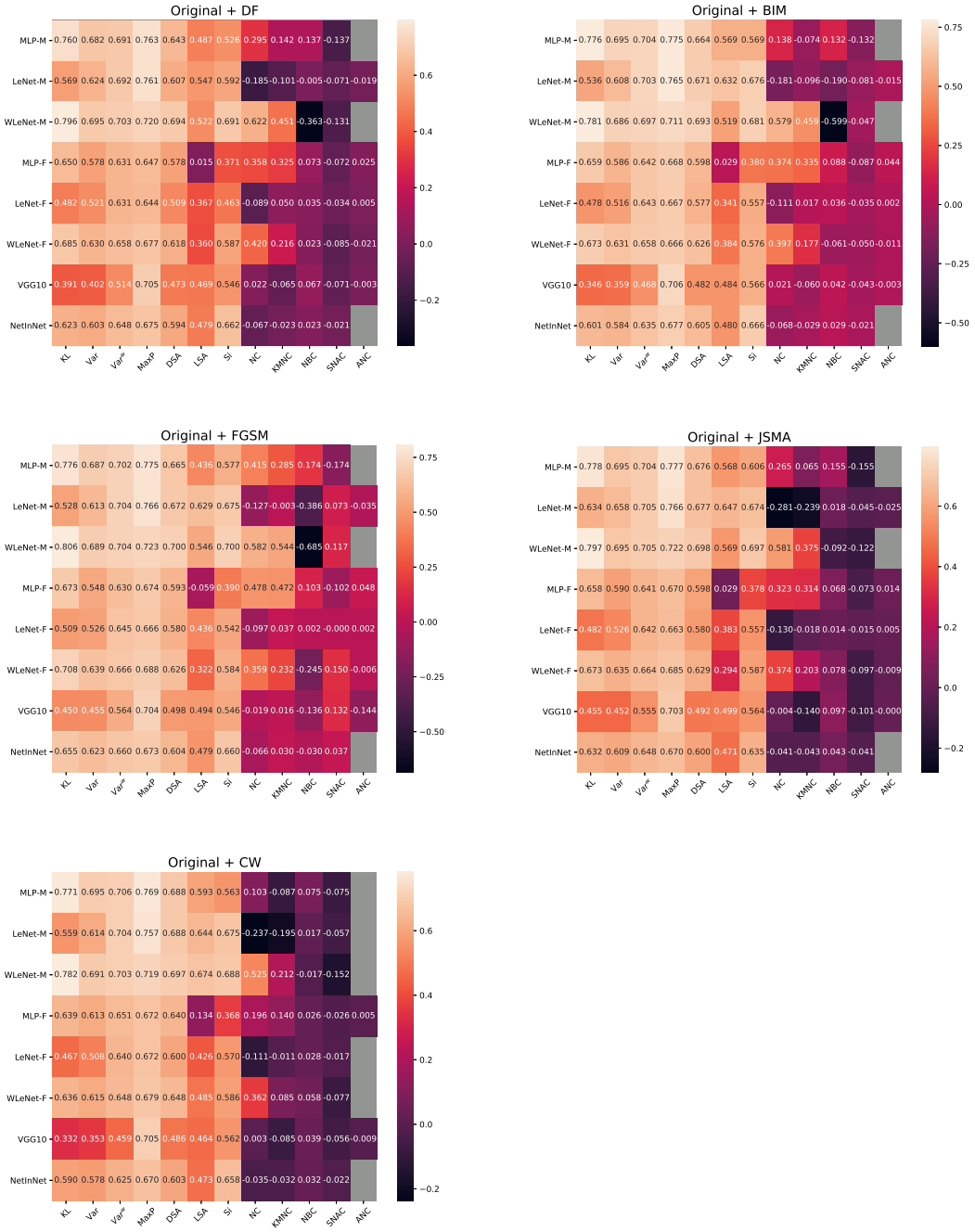
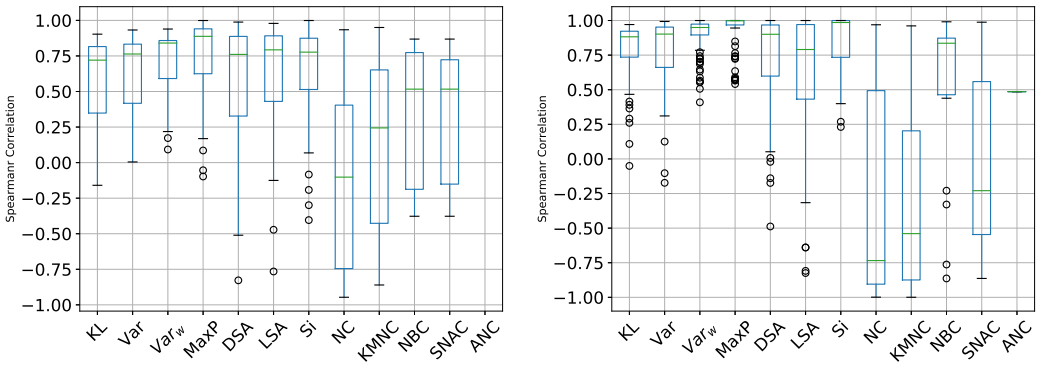


Fig. 2. Heatmap showing the Kendall correlation between misclassification and the metrics on a *mix of real and adversarial (misclassified) data*, obtained using five different algorithms. The lighter the color the better. Grey parts in ANC correspond to no increase in neuron coverage. Overall, MaxP achieves the strongest correlations.



(a) using 18,418 intermediate images from CW attacking

(b) using 3,603 intermediate images from FGSM attacking

Fig. 3. Spearman rank-order correlation between the metrics and the iteration number of the images generated by the adversarial algorithms. The iteration number reflects the distance from a misclassification and, thus, high correlation suggests that the metrics reflect well the likelihood of misclassification.

Uncertainty- and surprise-based metrics can discriminate well-classified real inputs from all (real and adversarial) misclassified inputs. They can achieve this with more ease as misclassified adversarial inputs are added. Overall, MaxP reaches the strongest correlations (between 0.64 and 0.78).

6.2.2 Well- and Miss-Classified Adversarial Data. Figure 3 shows, for each metric, boxplots representing the statistical distribution (over all models and images) of the Spearman correlation between the number of the iteration at which the image was produced and the metric value for this image. Metrics based on uncertainty and surprise achieve strong correlations (Var_w being the best in this regard), meaning that they are close to being monotonous over the iterations and thus capture well the adversarial generation process. On the contrary, the metrics based on neuron coverage reach very weak or irregular correlations.

Table 3 and Table 4 demonstrates the Kendall correlations between misclassification and the metrics computed on the intermediate (mostly well-classified) and final (misclassified) images generated by CW and FGSM, respectively. When considering FGSM, the correlations are similar to what they were when mixing real data with adversarial (misclassified) data. In the case of CW, however, they get weaker, although they remain medium to strong for some metrics (Var_w, MaxP, LSA and DSA). In particular, the correlations of KL are disappointing although this metric performed well in the previous experiments. These regressions can be explained by the fact that CW is known to generate smaller perturbations than the other adversarial algorithms. Thus, the difference between the intermediate images and the final images are smaller than what they are in the FGSM case.

When confronted with adversarial inputs only, the test selection metrics lose part of their capability. This is due to the inherent noise introduced by the adversarial generation algorithms. MaxP still achieves the strongest correlations overall, outperforming the other metrics in 13/16 cases.

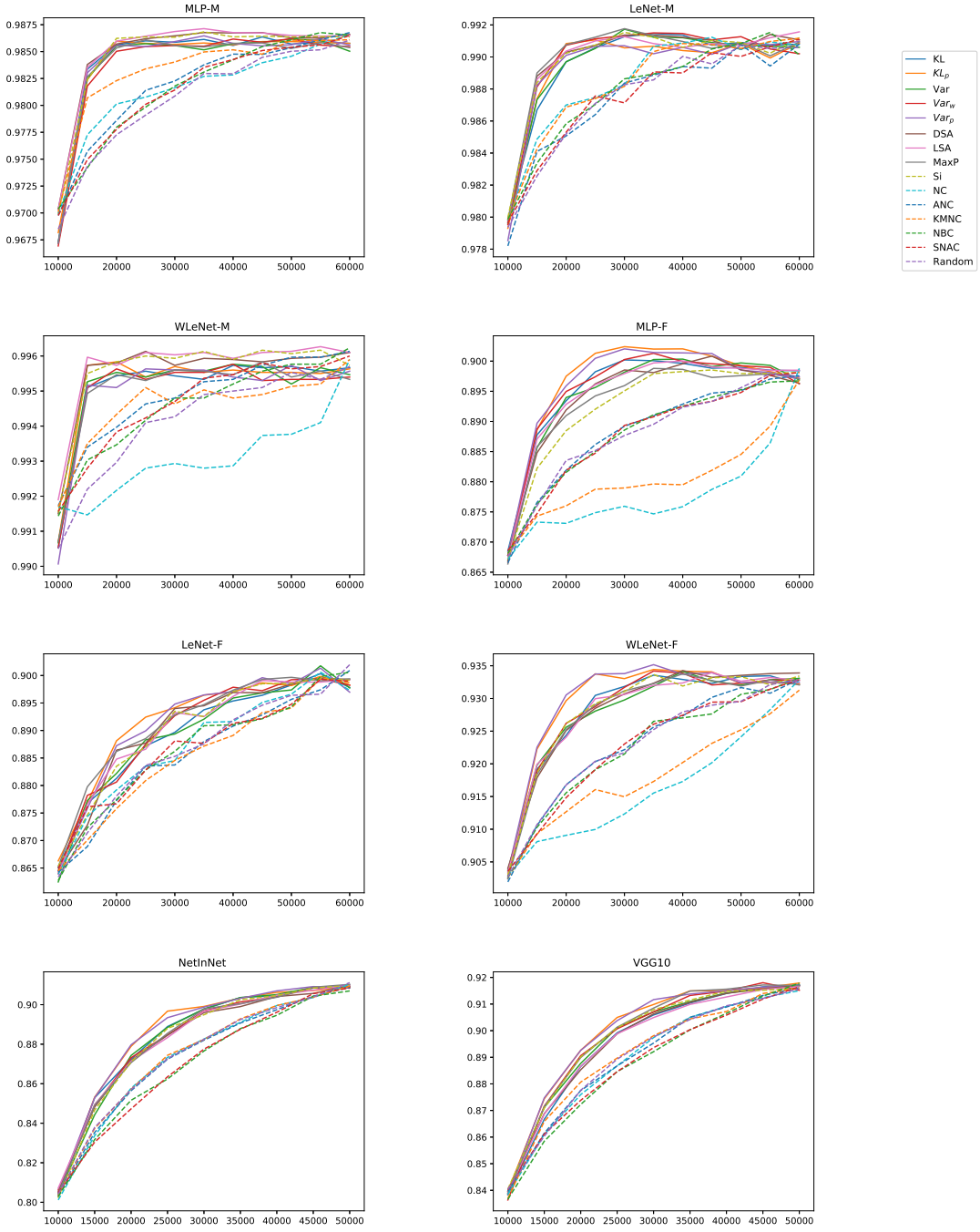


Fig. 4. Validation accuracy over a fixed set of 10,000 original test data and achieved by successively augmenting the training data with 5,000 data (at each retraining round) selected by the different metrics. X-axis denotes the size of the training set at each round, while Y-axis shows the average accuracy over three repetitions (variance is less than 10^{-5}).

Table 3. Kendall Correlation between misclassification and the metrics on adversarial data generated by CW (*mix of well-classified (intermediate) and misclassified (final) adversarial images*).

Model (Dataset)	KL	Var	Var_w	MaxP	DSA	LSA	Si	NC	ANC	KMNC	NBC	SNAC
MLP(MNIST)	0.5070	0.5282	0.6261	**0.6589**	0.5265	0.4419	0.2983	-0.0458	N/A	-0.1617	-0.0013	-0.0013
LeNet (MNIST)	0.1011	0.132	0.4474	0.5784	0.5917	0.5465	**0.5925**	-0.0958	N/A	-0.0631	-0.0559	-0.0284
WLeNet (MNIST)	0.4417	0.474	0.5255	**0.538**	0.4600	0.3041	0.3018	0.0568	N/A	-0.1545	-0.0837	N/A
MLP(Fashion)	0.2505	0.2428	0.3631	**0.4628**	0.3213	0.1394	0.1059	-0.0553	0.0103	-0.0336	-0.0547	-0.0715
LeNet (Fashion)	0.0994	0.1233	0.3126	0.4370	**0.4425**	0.2777	0.4381	0.0191	0.0201	0.0199	0.0545	0.0545
WLeNet (Fashion)	0.2818	0.2908	0.3855	**0.5244**	0.2618	0.2109	-0.0528	0.0597	N/A	-0.0324	-0.0059	-0.0145
VGG10 (CIFAR)	0.0606	0.0599	0.153	**0.5031**	0.2643	0.2106	0.2720	0.0434	0.0585	0.0640	-0.0093	0.0117
NetInNet (CIFAR)	0.1380	0.1094	0.2027	**0.4560**	0.1751	0.0453	0.2968	0.0055	N/A	-0.0150	0.0150	-0.001

Table 4. Kendall Correlation between misclassification and the metrics on adversarial data generated by FGSM (*mix of well-classified (intermediate) and misclassified (final) adversarial images*).

Model (Dataset)	KL	Var	Var_w	MaxP	DSA	LSA	Si	NC	ANC	KMNC	NBC	SNAC
MLP(MNIST)	0.6984	0.6459	0.6989	**0.7125**	0.5902	0.3736	0.3242	0.1828	N/A	0.1301	0.0237	0.0237
LeNet (MNIST)	0.3009	0.4176	0.7078	**0.7081**	0.6327	0.5779	0.6326	-0.1206	N/A	-0.0361	-0.2202	-0.1007
WLeNet (MNIST)	**0.6951**	0.5840	0.6927	0.6905	0.5802	0.3364	0.5418	0.2901	N/A	0.3282	-0.3831	-0.1956
MLP(Fashion)	0.576	0.2429	0.5541	**0.6645**	0.4527	0.0595	0.1444	0.1051	N/A	0.2732	N/A	N/A
LeNet (Fashion)	0.2848	0.3315	0.6410	**0.6848**	0.5292	0.3563	0.4802	-0.0458	-0.0002	0.0112	-0.0514	-0.0514
WLeNet (Fashion)	0.6178	0.4869	0.6064	**0.6817**	0.4469	0.2845	0.0651	0.1059	N/A	0.0568	-0.1779	-0.1703
VGG10 (CIFAR)	0.2325	0.2480	0.4526	**0.7014**	0.4346	0.3610	0.3969	-0.0346	-0.0273	0.0364	-0.1121	-0.1138
NetInNet (CIFAR)	0.538	0.5332	0.6236	**0.6732**	0.4989	0.3411	0.6047	-0.0168	N/A	0.0108	-0.0108	0.0069

6.3 Data Selection for Retraining

Figure 4 shows the best accuracy achieved of each retraining round by augmenting, iteratively, the training data with 5,000 data selected according to the different metrics. Here it must be noted that, while the raw accuracy values may seem to have small differences, they are due to the high initial accuracy of the model. Improving beyond this level is challenging.

Overall, we see that uncertainty and surprise metrics outperform those based on neuron coverage, which are comparable to random selection. For example, at the 3rd training augmentation round and for model WLeNet applied on Fashion-MNIST, Var achieves a gain in accuracy (compared to the accuracy of the initial training set) more than 45% higher than the best coverage-based metric – ANC – (+2.2% vs +1.5%) on WLeNet (Fashion), while the accuracy increases by +3% from the initial training set to the final (whole) training set.

On NetInNet applied to CIFAR-10 and at the 3rd round, Var achieves a gain in accuracy more than 20% than ANC does (+8.4% vs +6.9%), while the accuracy increases by +10.4% from the initial training set to the final training set. We observe similar conclusions when validation loss is considered. Indeed, metrics based on neuron coverage lead to slower decreases (similar to random selection), which reveals the inappropriateness of these metrics to select data for retraining.

In addition to the metrics considered so far, we augment Var and KL with a tie-breaking method: when two inputs have the same Var or KL scores, we select the input that has the lowest $MaxP$ score. Interestingly, those two new metrics (denoted by Var_p and KL_p) further improve the increase in accuracy of five models out of eight, and the decrease in validation loss in four models. Overall, those new metrics increase the accuracy up to two times faster than coverage metrics and random selection. Compared with the other uncertainty metrics, the additional gain is not

significant, though it keeps the merit to exist. Thus, should one require the use of a single metric, KL_p and Var_p would appear as effective choices.

For each model and metric, we computed the evolution of the empirical robustness over the retraining rounds. Overall, we observed that the robustness score barely varies over the rounds, regardless of the considered model and metric. Indeed, the largest gap across all models and metrics is 0.022, which is insignificant. Moreover, we cannot infer that any of the metric is comparatively best or worst than the others in this regard, as the variations are not monotonous. Thus, uncertainty and surprise-based metrics can increase accuracy faster than coverage-based metrics without compromising robustness.

> Uncertainty-based and surprise-based metrics, in particular the tie-breaking metrics KL_p and Var_p, are the best at selecting retraining inputs and lead to improvements up to 2 times faster than random selection. They achieve this without significant variations of the robustness (< 0.022).

6.4 Threats to Validity

Threats to internal validity concern the implementation of the software constituents of our study. Some are addressed by the fact that we reuse existing model architectures with typical parameterizations. The resulting models obtain a high accuracy on state-of-the-art datasets used as is (including their splitting into training and test sets), which indicates that our setup was appropriate.

We implemented dropout "from scratch" (i.e. as Lambda layers) in one case and, in the other cases, we reused the implementation natively embedded in the training process. The use of these two alternatives increases our confidence in the validity of our results. Finally, the implementation of the different metrics was tested manually and through various experiments. Moreover, we reused available implementations of the surprise-based and coverage-based metrics. Regarding LSA, it has been shown that the choice of the layers has an impact on the adequacy of the metric [14]. However, Kim et al. could find no correlation with the depth of the layer. As such, we make the same choice as Kim et al. and compute LSA on the deepest hidden layer.

The threats to external validity originate from the number of datasets, models and adversarial generation algorithms we considered. The settings we used are established in the scientific literature and allow the comparison of our approach with the related work. Performing well on such established and generic datasets is a prerequisite for real-world applications, which generally exhibit biases inherent to their application domain. The replication and the complementation of our study are further facilitated by the black-box nature of uncertainty metrics: all of them necessitate only the prediction probabilities to be computed.

Construct validity threats originate from the measurements we consider. We consider the correlation between the studied metrics and misclassification, which is a natural metric to use (and is in some sense equivalent to the fault detection and test criteria relations studied by software engineering literature [3, 9]).We also compare with surprise adequacy [14] and coverage metrics [22, 28], which are the current state-of-the-art methods.

6.5 Discussion and Lessons Learned

Our experimental results shed some light on the ability of existing metrics (coverage- and uncertainty-based) to drive the selection of test inputs.

Starting with DeepXplore [28], previous research advocates that increasing neuron coverage is a good way to perform "better" testing and has been using this criterion for test input generation. Additionally, in traditional (code-based) software engineering, coverage metrics (like statement and branch coverage) are commonly used to guide test generation/selection. It is therefore natural for software engineers to consider neuron coverage for test selection in DL systems as well. Another advantage of neuron coverage is that it provides a natural end-point when testing DL systems, viz. reaching 100% coverage. Yet, as in traditional (code-based) software, finding an adequate stopping criterion for testing DL systems remains an open problem. Our results confirm that achieving 100% of neuron coverage does not guarantee the absence of bugs, just like achieving 100% of statement coverage in traditional software does not. Even worse, coverage-based metrics exhibit weak correlations to misclassification, sometimes weaker than random selection. This brings an important message to the community: the misclassified inputs are not necessarily those that cover new neurons. Overall, while coverage-based metrics are convenient driving criteria for test input generation, different metrics should be used for test selection.

Regarding the remaining metrics (i.e. those based on uncertainty and surprise adequacy), our results provide new significant findings. When selecting test inputs to trigger misclassifications, the highest class probability – a simple metric often overlooked in the literature – performs the best regardless of the nature of the inputs (real or adversarial). Thus, we show that this simple metric forms a strong baseline for future research and that developers can rely on it as all-rounder test selection metrics. Another lesson is that dropout variance, the state-of-the-art metric to estimate model uncertainty, can be improved by normalizing the variance score with the highest class probability (yielding the weighted variance score). This is revealed by the fact that weighted variance has a stronger correlation than dropout variance in all our experiments.

When selecting inputs to retrain the model, we observe that combining KL divergence or weighted variance with the highest class probability yields consistently better results than the other metrics, although by a small margin. Thus, the difference between the uncertainty (and surprise adequacy) metrics is rather observed when selecting inputs for testing.

Another important finding is that some metrics (like KL divergence) are particularly sensitive to the noise introduced by adversarial data and significantly lose their capability (to distinguish well-classified and misclassified data) when confronted to adversarial data only. Indeed, the results of Section 6.2 indicate that introducing misclassified adversarial data into the test set yields a stronger correlation between the uncertainty metrics and misclassification. This means that the adversarial examples engender more uncertainty than real ones. This is because most adversarial algorithms aim at achieving misclassification while minimizing input perturbation. Making the model misclassify those examples with high confidence (low uncertainty) model is not part of the objective function of those algorithms, although some studies (e.g. [10]) have shown that this may happen incidentally. Our results in Section 6.2.1 confirm that the uncertainty of the model increases over the iteration of the adversarial algorithm, due to the increasing noise it introduces over the iterations.

7 CONCLUSION

We considered test selection metrics for deep learning systems based on the concept of model uncertainty. We experimented with these metrics and compared them with surprise adequacy and coverage related metrics wrt to their fault revealing ability, i.e., ability to trigger missclassifications.

Overall, our findings can be summarised by the following points:

- When dealing with original data, uncertainty metrics (in particular, KL and MaxP) perform best, significantly better than previously proposed metrics (coverage based and surprise adequacy).
- When dealing with a mix of original and adversarial data, MaxP – a simple certainty metric often overlooked by the literature – is the most effective. Additionally, uncertainty-based metrics are also effective at test selection, independent from the training set (coverage based and surprise adequacy metrics fall behind).
- Our results also revealed that the use of adversarial data in testing-related experiments should be performed with caution. All the studied metrics experience significant performance differences when considering original, adversarial or a mix of them.
- We also demonstrated that the metrics and particularly KL_p and Var_p, lead to major classification accuracy improvement (when selecting data for retraining), achieving a gain in accuracy of up to 80% higher than the previously proposed metrics and random selection.

Our work forms an essential step towards a long-term goal of equipping researchers and practitioners with test assessment metrics for DL systems. These automatic data selection metrics pave the way for the systematic and objective selection of test data, which may lead to standardised ways of measuring test effectiveness.

ACKNOWLEDGMENT:

This work is mainly supported by the Luxembourg National Research Funds (FNR) through the CORE project grant C17/IS/11686509/CODEMATES/Papadakis.

REFERENCES

[1] [n.d.]. *Fashion-MNIST.* Retrieved January 25, 2019 from https://github.com/zalandoresearch/fashion-mnist

[2] Alex Krizhevsky and Vinod Nair and Geoffrey Hinton. [n.d.]. *The CIFAR-10 dataset.* Retrieved January 25, 2019 from https://www.cs.toronto.edu/~kriz/cifar.html

[3] James H. Andrews, Lionel C. Briand, and Yvan Labiche. 2005. Is mutation an appropriate tool for testing experiments?. In *27th International Conference on Software Engineering (ICSE 2005), 15-21 May 2005, St. Louis, Missouri, USA*. 402–411. https://doi.org/10.1145/1062455.1062530

[4] 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 (AISec '17)*. ACM, New York, NY, USA, 3–14. https://doi.org/10.1145/3128572.3140444

[5] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 39–57.

[6] Reuben Feinman, Ryan R. Curtin, Saurabh Shintre, and Andrew Gardner. 2017. Detecting Adversarial Samples from Artifacts. *arXiv preprint arXiv:1703.00410* (2017).

[7] 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.

[8] Salah Ghamizi, Maxime Cordy, Martin Gubri, Mike Papadakis, Andrey Boystov, Yves Le Traon, and Anne Goujon. 2020. Search-Based Adversarial Testing and Improvement of Constrained Credit Scoring Systems. In *Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*.

[9] Milos Gligoric, Alex Groce, Chaoqiang Zhang, Rohan Sharma, Mohammad Amin Alipour, and Darko Marinov. 2013. Comparing non-adequate test suites using coverage criteria. In *International Symposium on Software Testing and Analysis, ISSTA '13, Lugano, Switzerland, July 15-20, 2013*. 302–313. https://doi.org/10.1145/2483760.2483769

[10] 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

[11] Kathrin Grosse, David Pfaff, Michael T. Smith, and Michael Backes. 2018. The Limitations of Model Uncertainty in Adversarial Settings. *CoRR* abs/1812.02606 (2018). arXiv:1812.02606 http://arxiv.org/abs/1812.02606

[12] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1321–1330.

[13] Alex Kendall and Yarin Gal. 2017. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?. In *NIPS*.

[14] Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding Deep Learning System testing using Surprise Adequacy. In *ICSE '19 (to appear)*.

[15] Ron Kohavi. 1995. A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection. In *IJCAI '95 - Vol. 2*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1137–1143. http://dl.acm.org/citation.cfm?id=1643031.1643047

[16] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533* (2016).

[17] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2016. Adversarial Machine Learning at Scale. *CoRR* abs/1611.01236 (2016). arXiv:1611.01236 http://arxiv.org/abs/1611.01236

[18] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444. https://doi.org/10.1038/nature14539

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

[20] Min Lin, Qiang Chen, and Shuicheng Yan. 2013. Network in network. *arXiv preprint arXiv:1312.4400* (2013).

[21] Lei Ma, Felix Juefei-Xu, Minhui Xue, Bo Li, Li Li, Yang Liu, and Jianjun Zhao. 2019. DeepCT: Tomographic Combinatorial Testing for Deep Learning Systems. In *SANER '19 (to appear)*.

[22] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. 2018. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. ACM, 120–131.

[23] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, Jianjun Zhao, and Yadong Wang. 2018. DeepGauge: Multi-granularity Testing Criteria for Deep Learning Systems. In *ASE '18*. ACM, New York, NY, USA, 120–131. https://doi.org/10.1145/3238147.3238202

[24] Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu, Chao Xie, Li Li, Yang Liu, Jianjun Zhao, and Yadong Wang. 2018. DeepMutation: Mutation Testing of Deep Learning Systems. In *ISSRE '18*. 100–111. https://doi.org/10.1109/ISSRE.2018.00021

[25] Shiqing Ma, Yingqi Liu, Wen-Chuan Lee, Xiangyu Zhang, and Ananth Grama. 2018. MODE: Automated Neural Network Model Debugging via State Differential Analysis and Input Selection. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018)*. Association for Computing Machinery, New York, NY, USA, 175âĂŞ186. https://doi.org/10.1145/3236024.3236082

[26] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *CVPR*. IEEE Computer Society, 2574–2582.

[27] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. 2016. The Limitations of Deep Learning in Adversarial Settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS P)*. 372–387.

[28] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In *SOSP '17*. ACM, New York, NY, USA, 1–18. https://doi.org/10.1145/3132747.3132785

[29] 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*. ACM, 1–18.

[30] Thomas Pinder. 2018. Adversarial Detection Through Bayesian Approximations In Deep Learning.

[31] Jonas Rauber, Wieland Brendel, and Matthias Bethge. 2017. Foolbox v0.8.0: A Python toolbox to benchmark the robustness of machine learning models. *CoRR* abs/1707.04131 (2017). arXiv:1707.04131 http://arxiv.org/abs/1707.04131

[32] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987), 53–65.

[33] Yunus Saatchi and Andrew Gordon Wilson. 2017. Bayesian GAN. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 3622–3631. http://papers.nips.cc/paper/6953-bayesian-gan.pdf

[34] Fatemeh Sheikholeslami, Swayambhoo Jain, and Georgios B. Giannakis. 2019. Minimum Uncertainty Based Detection of Adversaries in Deep Neural Networks. *CoRR* abs/1904.02841 (2019). arXiv:1904.02841 http://arxiv.org/abs/1904.02841

[35] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[36] Lewis Smith and Yarin Gal. 2018. Understanding Measures of Uncertainty for Adversarial Example Detection. In *UAI '18*. 560–569.

[37] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958. http://dl.acm.org/citation.cfm?id=2670313

[38] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[39] 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, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*. 303–314. https://doi.org/10.1145/3180155.3180220

[40] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. DeepTest: Automated Testing of Deep-neural-network-driven Autonomous Cars. In *ICSE '18*. ACM, New York, NY, USA, 303–314. https://doi.org/10.1145/3180155.3180220

[41] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. 2019. Robustness May Be at Odds with Accuracy. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

[42] Jingyi Wang, Guoliang Dong, Jun Sun, Xinyu Wang, and Peixin Zhang. 2019. Adversarial Sample Detection for Deep Neural Network through Model Mutation Testing. In *ICSE '19 (to appear)*.

[43] Ian H. Witten, Eibe Frank, and Mark A. Hall. 2011. *Data Mining: Practical Machine Learning Tools and Techniques* (3rd ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[44] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Hongxu Chen, Minhui Xue, Bo Li, Yang Liu, Jianjun Zhao, Jianxiong Yin, and Simon See. 2018. DeepHunter: Hunting Deep Neural Network Defects via Coverage-Guided Fuzzing. *arXiv preprint arXiv:1809.01266v3* (11 2018).