2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)

On the Investigation of Essential Diversities for Deep Learning Testing Criteria

Zhiyi Zhang, Xiaoyuan Xie *
School of Computer Science
Wuhan University
Wuhan, China
{*zhyzhang, xxie*}*@whu.edu.cn*

***Abstract*—Recent years, more and more testing criteria for deep learning systems has been proposed to ensure system robustness and reliability. These criteria were defined based on different perspectives of diversity. However, there lacks comprehensive investigation on what are the most essential diversities that should be considered by a testing criteria for deep learning systems. Therefore, in this paper, we conduct an empirical study to investigate the relation between test diversities and erroneous behaviors of deep learning models. We define five metrics to reflect diversities in neuron activities, and leverage metamorphic testing to detect erroneous behaviors. We investigate the correlation between metrics and erroneous behaviors. We also go further step to measure the quality of test suites under the guidance of defined metrics. Our results provided comprehensive insights on the essential diversities for testing criteria to exhibit good fault detection ability.**

***Keywords*-metamorphic testing; deep learning testing; testing criteria; essential metrics;**

I. Introduction

In recent years, deep learning (DL) has been widely used in a variety of areas including image recognition [1], speech recognition [2], medical diagnostics [3], to improve efficiency. DL systems are also increasingly deployed in both safety and security critical domains such as autonomous driving [4] and malware detection [5]. With the tremendous progress of DL techniques (i.e. Deep Neural Networks (DNNs)) and the large deployment of DL systems, the predictability and reliability of these systems are of great significance. DL systems often exhibit erroneous or unexpected behaviors under certain circumstances which can lead to dangerous situations. Hence, it is imperative to test and validate DL systems in a systematic way. DNN-based software is fundamentally different from the traditional one as the logic of DNNs is not encoded using control flow, making automated testing of DL systems a challenging problem.

Efforts have been recently made on testing DL systems [9]–[12], [19], [20], to ensure their reliability and robustness and gain higher confidence towards test quality. Methodologies in deep learning testing are generally inspired by traditional testing techniques where different types of coverage criteria have been defined to measure how thoroughly a software is tested. For traditional software testing, coverage criteria partitions the input domain and builds up the relation of input domain and software internal states [12]. The more diverse states being covered, the higher the chance to detect faults. *Testers would find various diversities that strongly correlate with erroneous behaviors to define coverage criteria* [13]. Similar to traditional software testing criteria, deep learning testing criteria are also proposed based on the assumption that more diversities could lead to more effective testing of a DL system. Different features of DL models are taken into consideration to explore more diverse states inside neural networks. For instance, DeepXplore [11] proposed a coverage-guided criteria that tries to cover as many neurons as possible to maximize the chances of finding erroneous behaviors as the measure of neuron behavior diversity; DeepGuage [12] presented a set of testing criteria from neuron-level and layer-level to monitor neuron activity diversities and intrinsic network connectivity, where neuron output boundaries obtained from training data were leveraged to approximate erroneous behaviors. We argue that instead of continuously presenting new testing criteria, understanding the diversities of these criteria for guiding effective fault detection of DL systems is a primary task. However, *there is a lack of comprehensive investigation on what kind of diversity is the most essential in revealing erroneous behaviors.*

To fill this research gap, we conduct a large-scale investigation on the essential diversities of testing criteria for revealing erroneous behaviors of DL systems, aiming to provide a hint for defining more comprehensive criteria that guide better testing quality in future research. In order to investigate the correlation between neuron activity diversities and erroneous model behaviors, we first adopt metamorphic testing (MT) to detect erroneous behaviors. Metamorphic Relations (MRs) that define relationships between behaviors of a DNN-based image retrieval system across certain types of image transformations are constructed. A metric is then designed to measure model erroneous behaviors by measuring the dissimilarity of system outputs under MRs. We then define five metrics to quantify diversities in terms of neuron activities, where four are extracted from existing

* Xiaoyuan Xie is the corresponding author.

testing criteria (Neuron Coverage [9], k-multisection Neuron Coverage, Neuron Boundary Coverage, and Top-k Neuron Coverage [12]) and one is defined from the perspective of activated neuron distribution. The correlation analysis provides a general understanding of what are most essential internal state diversities for testing DL models. Finally, the quality of test suites under the guidance of different metrics are evaluated, to provide a further insight on the effectiveness of input domain diversities in revealing erroneous behaviors. Our contributions are summarized as follows:

- To the best of our knowledge, this is the first comprehensive investigation on the existing deep learning testing criteria in view of test diversities. Two publicly available datasets and three DL models are involved in our large-scale empirical analysis.
- We explore the essential diversities for testing criteria to better detect faulty behaviors, by evaluating the correlation between neuron activity diversities and erroneous behaviors of DL models. We define five metrics to quantify various diversities of neuron activities, and leverage metamorphic testing to automatically detect erroneous behaviors. We show that diversities measured by k-multisection Neuron Coverage and Top-k Neuron Coverage criteria, as well as diversities in activated neuron distribution, are more essential to effective fault detection.
- We investigate the effectiveness test suites which are generated under the guidance of five metrics in facilitating fault detection. Experimental results show that different metrics yield test suites with different input domain diversities, and test suites that cover more diversities in activated neuron distribution or output value distribution are demonstrated to trigger more erroneous behaviors.

II. Background

Instead of investigating standalone DL models, we will focus on a particular application of these models, namely, content-based image retrieval (CBIR). In this section, we will give a brief introduction to CBIR and testing criteria for deep learning systems.

A. Deep Learning for Content-based Image Retrieval

CBIR is the application of computer vision techniques to the image retrieval problem. Using a database of reference images each having a label corresponding to the particular object or scene and given a new unlabeled query image, the task is to find images containing the same object or scene as in the query. “Content-based” means that the search analyzes the content of the image rather than the meta-data like keywords, tags or descriptions associated with the image. The key issues in CBIR are: image representation and image similarity measurement. The image is first transformed to a feature space, and the similarity which reflects the relevance in semantics between images is measured. In content-based visual retrieval, there is still a key challenge: “semantic gap” between low-level image pixels captured by machines and high-level semantic concepts perceived by human [14]. To narrow the gap, great efforts have been made in recent years [15].

Deep Convolutional Neuron Network (CNN) have advanced the state-of-the-art in image classification dramatically and have consequently attracted a lot of interests in computer vision community. The main difference between CNN and ordinary neural network is the presence of a convolutional layer. A CNN architecture consists of multiple interconnected neurons from input layer, output layer, pooling layers, fully-connected layers and convolutional layers. Specifically, the neurons on a convolutional layer are connected only to some of the neurons on the next layer, and multiple connections among different neurons share the same weight. Qualitative evidence of CNN’s ability for image retrieval is provided in [16]. A comprehensive study is conducted to investigate deep learning methods for learning feature representations and similarity measures towards CBIR task [14]. The deep learning framework for CBIR task always consists of two stages: (1) training a deep learning model from a large collection of training data; (2) applying the trained model for learning feature representations for CBIR task. For the first stage, the deep learning model can be simply trained for classification task, or be trained directly for the retrieval task [15]. In this paper, we simply adopt the former method that training models for classification task which has also been adopted as in [14], [17]. For the second stage, we can extract the activations of an intermediate layer (e.g. fully-connected layer or a convolutional layer) of the trained CNN model as the feature representations for CBIR task, since the lower layers are unlikely to contain reach semantic representations [14]. The architecture of an image retrieval system is shown in Fig. 1.

B. Coverage-guided Criteria in Deep Learning Testing

Recently, various testing criteria that measure neuron activities of DNNs have been proposed. DeepXplore [11] presents Neuron Coverage (NC), the ratio of neurons whose output value is higher than a predefined threshold and the total number of neurons in a neural network, as a measure of diversity of neuron behaviors. This metric simply counts individual neurons whose value satisfy a certain condition. Inspired by traditional MC/DC coverage criteria, a set of new criteria are designed for distinct features of neural networks [18]. Later, a set of multi-granularity testing criteria are proposed to guide DL systems construction [12]. At neuron-level, neuron output range obtained from training stage are partitioned into k sections, and k-multisection Neuron Coverage is defined as the ratio of sections covered by a set of test inputs and overall sections. Different from NC, this metric mainly focus on measuring neuron out-

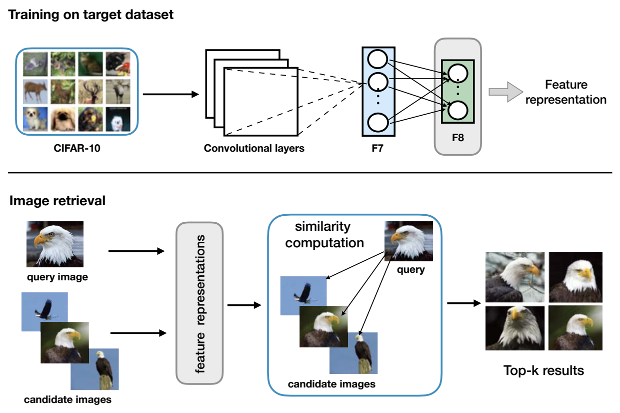


Figure 1. Overall architecture of our image retrieval system which consists of two components. The first is the training of neural network and the extraction of feature representations for a query image from one of the fully-connected layers. The second component is to retrieve similar images and rank images based on similarity scores.

put distribution. Another criteria ar neuron-level is Neuron Boundary Coverage, which measures the number of neurons whose value fall out of the output range. At layer-level, Top-k Neuron Coverage is defined to measure how many neurons have once been the top-k activated neurons on each layer based on the intuition that a test suit should uncover more top dominant neurons to characterize the major functionality of a DNN. Testing criteria inspired by traditional concolic testing, mutation testing and combinatorial testing are also presented [19]–[21].

III. Methodology

Testing criteria for DL systems are generally defined based on the assumption that larger diversity may lead to better fault detection ability of a DL system. Various diversities related to faulty behaviors could be considered to construct effective testing criteria by testers. What we are interested in this study is the correlation between different diversities and model erroneous behaviors. To this end, we adopt metamorphic testing to detect erroneous behaviors, and define five metrics to quantify diversities in terms of neuron activities.

A. Detection of erroneous behaviors of DL systems

Generally speaking, it is hard to test a software or a system in the area of Artificial Intelligence (AI) for lack of test oracle, which is known as "oracle probelm" [22]. Metamorphic Testing (MT) [7] is a promising technique that provides an alternative to alleviate oracle problem and automatically generate test cases to detect software bugs. The applications of MT to AI-driven systems has grown impressively with the tremendous progress in deep learning [9], [10], [23].

Considering a self-driving system, one can define relationships between the car's behaviors across different conditions. For example, the output steering angle should not change significantly for the same image under different weather conditions such as fog and rain. Let θ_s be the steering angle for an original input x (known as source input), θ_f be the steering angle for a newly generated input x^t (known as follow-up input) after apply a transformation t on x. One would define a metamorphic relation (MR) in MT where $\theta_s = \theta_f$. In this case, even it is hard to determine whether the output of a self-driving system is correct or not, if θ_s and θ_f are of significant difference, we can conclude that the system is wrongly behaved. In this paper, we leverage MRs to create test oracle for a DNN-based image retrieval system and then check whether the relationship between source and follow-up outputs violate the predefined MRs. A set of MRs for CBIR system are defined as follows.

Light. We adjust lightness of the source image by a random degree γ.

Shift. We randomly shift the image vertically and horizontally by s_w and s_h, where s_w and s_h denote the fraction of total width and height, respectively.

Rotate. Source image is rotated by a random degree r.

Zoom. We zoom the source image by a random degree z, and ensure the integrity of the digit on image.

Spnoise. Salt-and-pepper noise is a particular type of noise in which some randomly selected pixels are replaced by a white or a black pixel. This type of noise can occur in faulty communications when the value of some pixels is lost during the transmission, or occur in the image decoding stage. We randomly select n pixels and set the values to 0 or 255.

Erosion. Morphological erosion shrinks bright regions and enlarges dark regions. Erosion is conducted on the source image with a square the side of which is sq pixels.

Dilation. Morphological dilation enlarges bright regions and shrinks dark regions. Dilation is conducted on the source image with a square the side of which is sq pixels.

Line. We draw a random line across the central region of the image. (r_0, c_0) denote the starting coordinates, (r_1, c_1) denote the end coordinates.

Flip. We flip the images horizontally.

Next, we measure model output behaviors under the pre-defined MRs. For an image retrieval system I, let x be an input query, $db = \{c_1, c_2, \cdots, c_n\}$ be the database of candidate images to be retrieved. The retrieval system will return a set of images which are "similar" to x. Let $sim(x, c_i)$ be the cosine similarity of the query and i-th candidate image where $1 \leq i \leq n$. We use $top_k(x)$ to denote the top-k images in db that are most similar to x. In our experiments, we use top-900 images for evaluation. Given an input query x, we define the corresponding result returned by I as:

$$R = \{i \mid \exists c_i \in db : c_i \in top_k(x)\}$$

It is generally considered that the retrieval system should be robust to certain types of image transformations. For example, the system should be able to return similar results for the same image under small-angle rotation, because the image content remains unchanged.

Given a source input x_s, assuming that a follow-up input x_f is generated by applying a transformation t on x_s. Let $R_s = I(x_s)$ be the source output and $R_f = I(x_f)$ be the corresponding follow-up output. A metamorphic relation (MR) defines the expectation on the changing trend of I's behavior after the transformation on input. To further measure the inconsistence between R_s and R_f which estimate model behavior of I, we define a metric RD as follows.

Definition 3.1 (RD): We measure the dissimilarity between R_s and R_f by measuring their Jaccard distance:

$$RD = 1 - \frac{|R_s \cap R_f|}{|R_s \cup R_f|}$$

Obviously, RD=0 means follow-up output is consistent with the source one. The higher this value (i.e., closer to 1), the lower the similarity between source and follow-up outputs. As a reminder, using jaccard distance, we do not care about the order of the images returned by I.

B. Metrics for Neuron Activities

To investigate the diversities of the testing criteria for enhancing error-behavior detection, we identify four metrics from existing criteria in terms of intrinsic neuron activities, and define one metric from the perspective of activated neuron distribution. Before giving the definition of five metrics, we first define Metamorphic Neuron Slice, which is the basis of these metric definitions.

Definition 3.2 (Metamorphic Neuron Slice): Given an input x and an investigated DNN model M, let N denote the neuron set in M. For each neuron $n \in N$, let $\phi(x, n)$ be the output of neuron n under x. Let t be the threshold to determine whether a neuron is get activated. We define neuron slice $slice$ as:

$$slice{=}\{n \mid \exists n \in N : \phi(x, n) > t\}$$

For a metamorphic relation MR, suppose x_s and x_f are its respective source test input and follow-up test input. Let $slice_s$ and $slice_f$ be the respective neuron slice in source and follow-up execution with respect to the whole model M.

Let $mslice$ be the metamorphic neuron slice of a pair of test inputs which satisfy a certain MR. A set of metrics is defined as follows:

- Let $mslice^u$ be the union of $slice_s$ and $slice_f$.

$$mslice^u = slice_s \cup slice_f$$

- Let $mslice^c$ be the intersection of $slice_s$ and $slice_f$.

$$mslice^c = slice_s \cap slice_f$$

Next, we will define four metrics in terms of neuron activities from existing testing criteria, namely, ΔNC from Neuron Coverage criteria [9], [11], SD from k-multisection Neuron Coverage criteria [12], BD from Neuron Boundary Coverage criteria [12] and TD from Top-k Neuron Coverage criteria [12]. And we also define a metric ND from the perspective of activated neuron distribution.

Neuron Coverage was the very first testing criteria for DL systems, are was illustrated to be an effective indicator of input-output diversity and thus can guide test data generation. To perform our evaluation on Neuron Coverage, here we present metric ΔNC.

Definition 3.3 (Metric ΔNC): This metrics aims to measure the diversity with respect to the number of activated neurons, which is defined from *Neuron Coverage* [11]. Given an input x and an investigated DNN model M with $|N|$ neurons, neuron coverage NC denote the ratio of the number of neurons in $slice$ and the total number of neurons in M. And the neuron coverage difference ΔNC is denoted as the difference between source and follow-up neuron coverage.

$$\Delta NC = \frac{|slice_f| - |slice_s|}{|N|}$$

ΔNC>0 means that neuron coverage in follow-up execution is greater than the coverage in source execution, and vice versa.

DeepGuage [12] proposed a set of coverage criteria from multiple levels to gauge the testing adequacy of DNNs, which was demonstrated to be effective at revealing erroneous behaviors in DL systems. k-multisection Neuron Coverage is a neuron-level criteria which measure how throughly a test suit cover the neuron output range $[low_n, high_n]$, where n is the investigated neuron. Top-k Neuron Coverage is defined at layer-level, which measure the top dominant neurons in different layers. Here we extend their concepts and define three metrics including SD, BD and TD, from *k-multisection Neuron Coverage, Neuron Boundary Coverage and Top-k Neuron Coverage* [12]:

Definition 3.4 (Metric SD): This metric aims to measure the diversity in neuron output distribution with respect to major function regions. For a neuron n, the output range $[low_n, high_n]$ derived from training stage is divided into k equal sections. Let $sec(x, n)$ be the index of section that covered by a test input x where $1 \leq sec(n) \leq k$. We define the sequence of sections covered by x as: $\{sec(x, n) \mid \forall n \in N\}$

Given a source input x_s and its corresponding follow-up input x_f, and an investigated DNN model M, we say neuron n is covered if $sec(x_s, n)$=$sec(x_f, n)$. We define k-multisection Neuron Difference (denoted as SD) as the ratio of uncovered neurons and total neurons in M:

$$SD{=}1 - \frac{|\{n \mid \exists n \in N : sec(x_s, n) = sec(x_f, n)\}|}{|N|}$$

A higher SD indicates bigger difference of neurons that get covered in source and follow-up execution (cover the same section in this case).

Definition 3.5 (Metric BD): This metric aims to measure the diversity in neuron output distribution with respect to corner-case regions. For a neuron n, if $\phi(x, n)$ belongs to $(-\infty, low_n)$ or $(high_n, \infty)$, we say the corresponding corner-case region is covered. Let $bound(x, n)$ be the corner-case region of n that is covered by an input x.

Given a source input x_s and its corresponding follow-up input x_f, we say a neuron n is covered if $bound(x_s, n){=}bound(x_f, n)$. We define Neuron Boundary Difference (denoted as BD) as the ratio of uncovered neurons and total neurons in M:

$$BD{=}1 - \frac{|\{n \mid \exists n \in N : bound(x_s, n) = bound(x_f, n)\}|}{|N|}$$

Definition 3.6 (Metric TD): This metric aims to measure the diversity in most activated neurons. As defined in DeepGuage, for neurons n_1 and n_2 on the same layer, we say n_1 is more active than n_2 if $\phi(x, n_1) > \phi(x, n_2)$. For the i-th layer in M with l layers, let $top_k(x, i)$ be the most active k neurons. Given a source input x_s and its corresponding follow-up input x_f, we measure the difference of top-k neurons as follows:

$$TD{=}1 - \frac{|\{top_k(x_s, i) \cap top_k(x_f, i) \mid \forall i \in l\}|}{k \times l}$$

In this case, TD measures how many common neurons are in top-k dominant neurons in source and follow-up execution.

It can be seen that most of the criteria focus on evaluating output values of neurons, while the slice of neurons that get activated in the network also form a pattern which could deliver important information for detecting erroneous behaviors of a DNN. Considering Neuron Coverage (NC), it simply counts the number of activated neurons without considering their distribution in the network. However, we argue that the number of activated neurons alone is not enough for revealing neuron behaviors in a DNN. For example, NC_s and NC_f could be the same even when x_s and x_f activate totally different sets of neurons as long as the amount is identical. To complement the existing criteria, we define a metric based on the intuition that unique neurons that get activated might potentially lead to the difference in output behaviors of DL systems. That is, *the activated neuron distribution should be more informative to reveal erroneous behaviors*.

Definition 3.7 (Metric ND): This metrics is defined to measure the diversity with respect to activated neuron distribution. Given a source input x_s and its corresponding follow-up input x_f, and an investigated DNN model M, neuron pattern difference ND denote the dissimilarity between $slice_s$ and $slice_f$.

$$ND = 1 - \frac{|mslice^c|}{|mslice^u|}$$

Obviously, $ND{=}0$ implies that $slice_s$ and $slice_f$ are identical. The greater the ND, the bigger the difference in neuron slices of source and follow-up execution, that is, follow-up input tends to activate more unique neurons.

Algorithm 1: Greedy search for covergae-guided test cases generation

```
   Data: Transformation $T$, Seed (Source) Images $I$
   Result: Synthetically generated images
1  Push all seed images $\in I$ to Stack $S$
2  newImgs = $\emptyset$
3  while $S$ is not empty do
4      seedImg = S.pop()
5      while numTries <maxTries do
6          Randomly pick a transformation $t$ from $T$
7          Randomly pick parameter $p$ for $t$
8          newImg = Transform(seedImg, $T$, $p$)
9          if diffIncrease(seedImg, newImg) then
10             newImgs = newImgs $\cup$ newImg
11             updateDiff()
12         else
13             numTries $+=1$
14         end
15     end
16 end
```

C. Evaluation on Quality of Metric-guided Test Suites

The metrics defined above are intend to investigate the correlation to erroneous model behaviors (c.f. Section III-A), which would give us a preliminary understanding on what are the essential internal state diversities for a testing criteria to better detect erroneous behaviors. And we also want to investigate the capabilities of these diversities in guiding testing process to reveal various defects. Thus in this section, we generate test suites with different metric guidance to boost diversities over input domains, and go further step to evaluate the actual testing errors of these test suites, which help us obtain further confidence of testing quality.

We leverage a greedy strategy to generate defect candidates which would increase difference in neuron behaviors after applying specific transformations. Algorithm 1 describes the outline of such an algorithm for guiding test case generation. The algorithm takes a set of source images I, a list of transformations T and corresponding parameters p as inputs. For each metric, we select one source image from I for each time and randomly pick one transformation t from T. For a picked transformation t, parameter p is also randomly selected. By applying t on I, a new image is generated. We keep record of the neuron behavior difference between a newly generate image and the original image. Method $diffIncrease()$ returns whether each new generated image increase the difference in each iteration. If the returned value is true, the new image will be added to the candidate set. The difference value is then updated by applying method $updateDiff()$. In summary, new test cases will be iteratively generate by keeping track of the image that boosts neuron behavior difference and add it to the test queue.

We also evaluate the effectiveness of different metrics for guiding test case generation by comparing the Mean Squared Error (MSE) between the predictions of original test cases and transformed test cases with respect to their manual labels. Let $\{p_{s1}, p_{s2}, \cdots, p_{sn}\}$ be a set of outputs predicted by DNN model for source inputs and $\{p_{f1}, p_{f2}, \cdots, p_{fn}\}$ be the outputs for follow-up inputs. Let $\{p_{o1}, p_{o2}, \cdots, p_{0_n}\}$ be the manual labels for source inputs. Metrics for source and follow-up execution are defined as follows.

$$MSE_{source} = 1/n \sum_{i=1}^{n} (p_{si} - p_{oi})^2$$

$$MSE_{follow} = 1/n \sum_{i=1}^{n} (p_{fi} - p_{oi})^2$$

$MSE_{follow} > MSE_{source}$ means the newly generated images could trigger more erroneous behaviors of a DNN model than randomly selected test inputs, thus the corresponding metric can serve as an effective guidance for test case generation.

IV. Experiment Setup

To evaluate model behaviors and neuron activities with our defined metrics, we select two well-known datasets, i.e., MNIST [24] and CIFAR-10 [25], and three pre-trained DNNs, i.e., LeNet-5 [24], VGG-16 [26] and ResNet-20 [27] based on Keras framework [28] for analyses.

MNIST. MNIST is a large dataset of handwritten digits which comprises a training set of $60,000$ images and a testing set of $10,000$ images. Each image is a single channel (grayscale) data, and is 28 pixels in width and 28 pixels in height. We then select a pre-trained LeNet-5 model to conduct our evaluation. LeNet-5 is a convolutional neural network which comprises 9 layers (including input and output layers) and 268 neurons. To perform image retrieval task based on LeNet-5, we extract the outputs from layer F6 (the fully-connected layer before soft-max) as feature representations of the input image.

CIFAR-10. The CIFAR-10 dataset consists of 60000 general color images in 10 classes, where 50000 for training and 10000 for testing. Each image is 32 pixels in width and 32 pixels in height. To further evaluate model and neuron behaviors, we select two publicly available DNNs, i.e., VGG-16 and ResNet-20 for CIFAR-10 dataset. VGG-16 is a deep network that contains 22 layers (16 weight layers) and $14,888$ neurons, where we extract the outputs from layer FC-4096 as feature representations of the input image to perform image retrieval task. ResNet-20 is a deep residual network with 70 layers (20 weight layers) and 2570 neurons. We take the activations from the last activation layer as the feature representation of the input image.

Table I
PARAMETER CONFIGURATION FOR INVESTIGATED MRS.

Dataset	MR	Parameter	Value
MNIST CIFAR-10	Light	γ	[0.7, 1.5]
	Shift	s_w, s_h	[0, 0.2], [0, 0.2]
	Rotation	r	[0, 30]
	Zoom	z	[0.6, 1.3]
	Spnoise	n	[50, 100]
MNIST	Erosion	sq	2
	Dilation	sq	2
	Line	r_0, c_0, r_1, c_1	[1,27], [1,4], [1,27], [24,27]
CIFAR-10	Flip	-	-
	Line	r_0, c_0, r_1, c_1	[0,28], [0.28], r_0+4, c_0-4

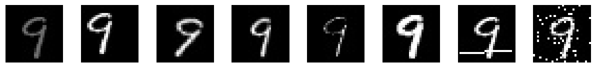
Table II
PARAMETER CONFIGURATION FOR EVALUATED METRICS.

Metric	Parameter Configuration		
ΔNC (from Neuron Coverage)	t=0.2	t=0.5	t=0.75
SD (from k-multisection Neuron Coverage)	k=2	k=10	k=50
BD (from Neuron Boundary Coverage)	-	-	-
TD (from Top-k Neuron Coverage)	k=1	k=2	k=3
ND	t=0.2	t=0.5	t=0.75

For MNIST dataset, we sample 1000 images from the testing set to be source inputs, each of which is transformed to generate new input by applying 8 MRs (see Table I). Totally we get 1000×8 pairs of source and follow-up inputs for LeNet-5 model. For CIFAR-10 dataset, we also randomly select 1000 images from testing set to be source inputs, each of which is transformed to generate follow-up input by applying 7 MRs (see Table I). Totally we get 1000×7 pairs of source and follow-up inputs for VGG-16 and ResNet-20, respectively. As a reminder, we do not perform Erosion, Dilation on CIFAR-10 because these MRs are designed for hand written digits according to their specific characteristics. And we do not perform Flip on MINIST because image semantic would be changed by flipping a digit. The parameters described in Table I are carefully configured to not corrupt image semantics. Fig. 2 and Fig. 3 show the examples of follow-up images for MNIST and Cifar-10 dataset, respectively. To further evaluate neuron behaviors, we construct three parameters for each evaluation metric (except BD) as shown in Table II.

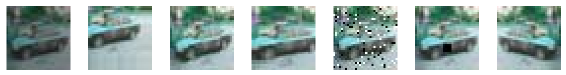
V. Results

In this section, we evaluate five metrics by investigating the following three research questions, aiming to provide a further insight on the effectiveness of existing testing criteria as indicators to quantify fault detection ability in DL systems:



(a) Light (b) Shift (c) Rotation (d) Zoom (e) Erosion (f) Dilation (g) Line (h) Spnoise

Figure 2. Examples of investigated MRs for MNIST dataset



(a) Light (b) Shift (c) Rotation (d) Zoom (e) Spnoise (f) Line (g) Flip

Figure 3. Examples of investigated MRs for CIFAR-10 dataset

- **RQ1:** Do different MRs enable diverse model output behaviors ?
  Our first research question is if we can detect erroneous behaviors using MRs as described in Section III-A, and to what extent different MRs reveal error-behaviors. To address this issue, we will reveal the fault detection ability of MRs by comparing RD across different MRs.
- **RQ2:** How effective are different metrics in revealing erroneous behaviors ?
  Next, in order to explore the essential diversities for better fault detection, we will measure the correlation between different metrics (c.f. Section III-B) and model erroneous behaviors, aiming provide a preliminary understanding of how internal state diversities contribute to fault detection.
- **RQ3:** How well do different metrics guide test case generation ?
  Finally, in order to investigate the capabilities of these diversities in guiding testing process to reveal various defects, test suites are generated with metric guidance to boost diversities over input domains, and test errors between the predictions of test suites and their manual labels will be measured to quantify the effectiveness.

A. RQ1: Evaluation on MR Fault Detection Ability

First, we check whether different MRs could trigger diverse behaviors of investigated DL models. Commonly said, good MRs are the MRs with higher chance of detecting failures [29]. To quantify the fault detection ability of MRs, we adopt the Jaccard distance to measure the dissimilarity between source output and follow-up output (c.f. Section III-A). We compute RD for each pair of test inputs. Obviously, the higher the RD, the more effective of an MR in revealing erroneous behaviors of DL models. Fig. 4 shows the distribution of RD under different MRs for three investigated models.

It can be seen from the leftmost plot in Fig. 4 that MR-Shift on MNIST dataset could trigger more erroneous behaviors of LeNet-5 than other MRs, with a widest RD distribution where the median value is over 0.8. MR-Rotate and MR-Zoom also have relatively high capability of revealing erroneous model behaviors with median RD value close to 0.4. While MR-Erosion, MR-Dilation and MR-Light do not show much effectiveness on fault detection, which implies that LeNet-5 model may have good robustness to the change in image brightness, or to morphological variation. For VGG-16, MR-Light, MR-Rotate and MR-Zoom are shown to have relatively better performance at revealing erroneous model behaviors, where the maximum value of RD is close to 1. While MR-Line and MR-Spnoise are invalid to trigger different logic of DNNs wirh RD close to 0, which means the investigated model could stay robust to simple pollution on inputs such as adding specific amount of salt and pepper noise or drawing a single line across the image. Moreover, the investigated model also show robustness to transformations such as shifting or flipping. As for ResNet-20, it also stay quite robust to MR-Line and MR-Spnoise with RD close to 0, while MR-Light, MR-Rotate and MR-Zoom could trigger more erroneous behaviors of Investigated model with RD around 0.4. We also notice that in comparison to VGG-16, the investigated MRs have higher chance of detecting erroneous behaviors of ResNet-20 model.

Among the three DNNs under test, LeNet-5 show greatest sensitiveness to image disturbation, this may caused by the shallow architecture of LeNet-5, while VGG-16 and ResNet-20 are larger in size and complexity which may exhibit better robustness to small disturbation.

To address RQ1: Different MRs exhibit different capabilities of revealing erroneous behaviors of DL models. The effectiveness of proposed MRs also partially rely on specific datasets and DNNs.

B. RQ2: Correlation Between Metrics and Model Erroneous Behaviors

On testing DL systems, previous works always leverage coverage-guided strategy to boost the coverage with the guidance of different criteria, however, there still lacks of comprehensive evaluation on what are most essential diversities that guide better fault detection. We address this issue in our second research question by evaluating the correlation between five metrics (c.f. Section III-B) and model output behaviors (c.f. Section III-A).

Neuron coverage (NC) was initially proposed in DeepXplore [11] which was maximized to induce erroneous behaviors in DL system. NC was shown to be a good metric for DNN testing comprehensiveness. To evaluate Neuron Coverage Difference (ΔNC), we set the activation threshold t to 0.2, 0.5 and 0.75, respectively. Fig. 5 shows the distribution of ΔNC with respect to all input pairs on three investigated models. It can be seen from Fig. 5 that for three investigated

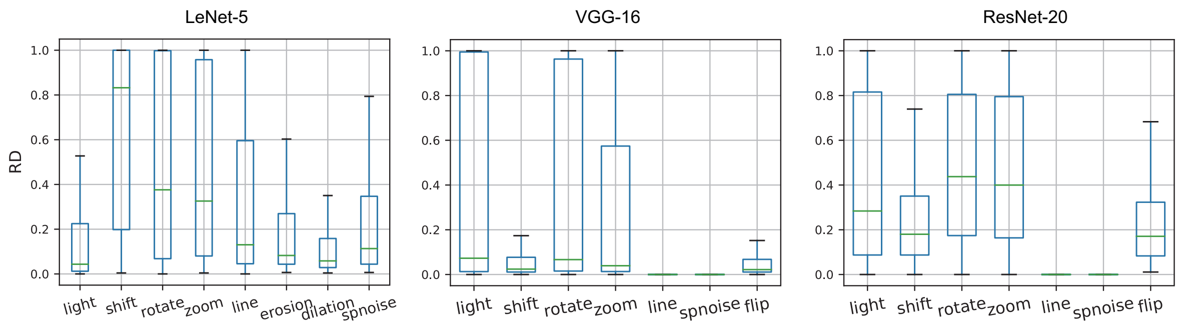


Figure 4. RD distribution on LeNet-5, VGG-16, and ResNet-20 w.r.t each MR.

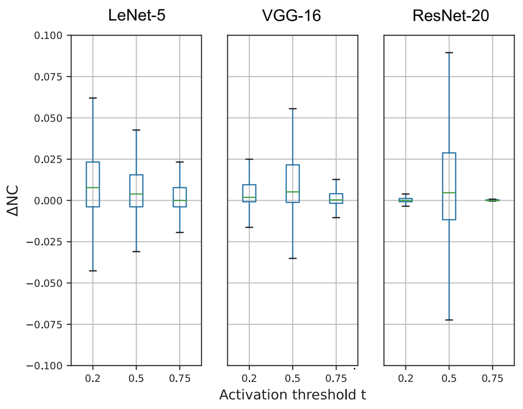


Figure 5. ΔNC distribution on LeNet-5, VGG-16, and ResNet-20 w.r.t all MRs.

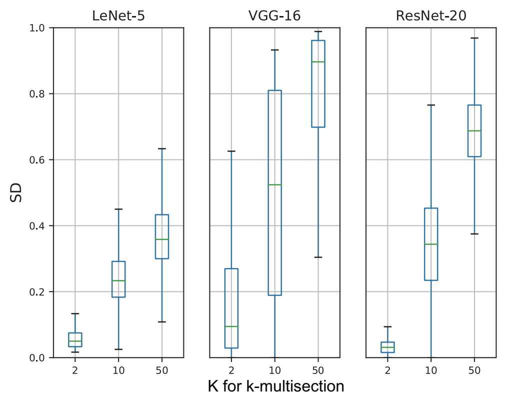


Figure 6. SD distribution on LeNet-5, VGG-16, and ResNet-20 w.r.t all MRs.

DNN models, there is no significant change in neuron coverage after image transformation, where ΔNC mainly varies from $[-0.1, 0.1]$. Moreover, we learn that NC does not always increase in follow-up execution, which however contradict the conclusion in previous work [9] that "different image transformation increase neuron coverage at different rates". We can also learn that different configurations of t result in different coverage variability. For example, greater variations of NC are found under $t = 0.2$ for LeNet-5, while VGG-16 and ResNet-20 show more variability of NC under $t = 0.5$. Overall, it can be concluded that Neuron Coverage exhibit weak capability in revealing erroneous behaviors of DL models.

k-multisection Neuron Coverage and Neuron Boundary Coverage are proposed as neuron-level criteria in DeepGauge [12], where neuron output boundaries from training stage are leveraged to approximate major functionality and corner-case behaviors. To evaluate k-multisection Neuron Difference (SD), we set parameter k to $2, 10, 50$, respectively. As shown in Fig. 6, in comparison with ΔNC, SD has relatively larger variation in range $[0, 1]$, which implies that neuron output tends to fall in different sections in source execution and follow-up execution. Thus, SD is effective in distinguishing neuron activities under different inputs. In general, SD becomes higher as k increases for all three investigated models. We also measure the Neuron Boundary Difference (BD) to explore the corner-case behaviors of

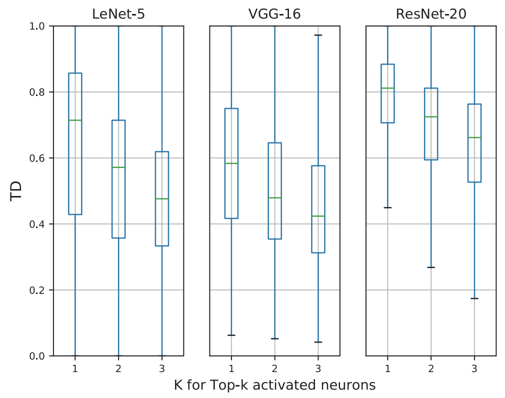


Figure 7. TD distribution on LeNet-5, VGG-16, and ResNet-20 w.r.t all MRs.

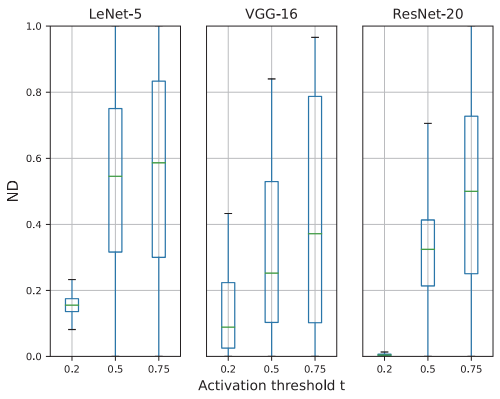


Figure 8. ND distribution on LeNet-5, VGG-16, and ResNet-20 w.r.t all MRs.

neurons. However in our experiment, test inputs mainly trigger the difference in major function region, only few neurons cover the corner-case regions, hence the plot of BD deliver little information. We would leave the analysis of this criteria in our future work that take account of generating effective MRs to cover more corner-case regions.

Top-k Neuron Coverage is proposed as a layer-level criteria in DeepGauge [12], based on the intuition that a test suit should uncover more top dominant neurons to characterize the major functionality of a DNN. To evaluate metric Top-k Neuron Difference (TD), we set parameter k to $1, 2, 3$, respectively. Fig. 7 show the distribution of TD for three investigated models. It can be observed that all three DNN models show great variation over TD and the median values are all over 0.4, which verify the intuition that neurons that get activated most in each layer could characterize major functionality of a DNN. In general, the larger the k, the less increment on TD.

The above testing metrics exhibit different capabilities of revealing erroneous model behaviors, among which SD and TD are of greater effectiveness. However, these metrics mainly take account of the amount of neurons that satisfy a specific constraint, while leaving out the network-wide neuron activation pattern. Although DeepGauge [12] has defined a layer-level criteria named Top-k Neuron Patterns, it only take the sequence of top-k neurons on each layer. Therefore, we are encouraged to define a new metric named ND (see Definition 3.7 in Section III-B) to measure the activated neuron distribution between inputs, based on the intuition that neurons on different parts of the network tend to learn different features which may contribute to diverse output behaviors.

Fig. 8 shows the distribution of ND, where ND increases as threshold t increases. It can be observed that under $t = 0.75$, ND approximates 0.6 for LeNet-5 model and ResNet-20 model, while VGG-16 approximates 0.4. It should be notice that for ResNet-20 under $t = 0.2$, ND mainly distribution around 0, which implies there is no significant difference between neuron activities in source and follow-up execution. This might caused by the relatively high neuron output boundaries with respect to ResNet-20. To be summarized, ND can effectively distinguish different neuron activities.

Table III
CORRELATION BETWEEN TESTING METRICS AND MODEL OUTPUT BEHAVIORS.

Metrics	Parameters	Models		
		LeNet-5	VGG-16	ResNet-20
ΔNC	t=0.2	0.35	0.41	-0.05
	t=0.5	0.23	0.43	0.21
	t=0.75	0.18	0.12	0.05
SD	k=2	0.31	0.95	0.40
	k=10	0.56	0.93	0.77
	k=50	0.40	0.77	0.72
BD	-	0.14	0.30	0.21
TD	k=1	0.54	0.80	0.34
	k=2	0.55	0.77	0.28
	k=3	0.53	0.74	0.24
ND	t=0.2	0.77	0.96	0.30
	t=0.50	0.72	0.95	0.17
	t=0.75	0.71	0.97	0.55

Spearman correlations are of statistical significance with p-value < 0.01

For further analysis, we evaluate the effectiveness of testing metrics in revealing erroneous behaviors by checking the Spearman rank correlation between each metric and RD (i.e. model output behavior difference) considering all pairs of inputs. Table III summarizes the correlation coefficient for three investigated models under different configurations. Among the five metamorphic metrics, BD shows the lowest correlation with model output behaviors. As discussed above, only few neurons cover the corner-case region in our experiment, thus the result is acceptable. For ΔNC under three configurations, the correlations are low especially for LeNet-5 and ResNet-20. In comparison with ΔNC, TD performs much better on revealing model behaviors with stronger Spearman correlations. As for ND and SD, three investigated models show an strong association, that is, ND or SD changes significantly with changes in RD. ND show fairly strong correlation with output behaviors with Spearman correlations over 0.9 for VGG-16 model and over 0.7 for LeNet-5 model, while ResNet-20 model show relatively lower correlations. And SD also show fairly strong correlation with Spearman correlations over 0.9 for VGG-16 model, while for LeNet-5 and ResNet-20, correlations are weaker. The results indicate that testing criteria which measure neuron activation pattern and major functionality could effectively reveal erroneous behaviors of a DL model with high confidence.

Table IV
MSE FOR SOURCE AND FOLLOW-UP EXECUTION.

		LeNet	VGG-16	ResNet-20
MSE_{orig}		0.00153	0.01387	0.01202
MSE_{new}	NC	0.00722	0.05753	0.07105
	SD	0.08360	0.13280	0.12515
	BD	0.06337	0.13401	0.12468
	TD	0.04407	0.13376	0.12320
	ND	0.06459	0.13297	0.12811

To address RQ2: Testing criteria that approximate more diversities in neuron space distribution, major function region cover and most activated neurons may contribute to greater fault detection ability. k-multisection Neuron Coverage criteria and Top-k Neuron Coverage criteria are more effective in revealing erroneous behaviors of DL systems; while Neuron Coverage criteria contributes less to effective fault detection.

C. *RQ3: Evaluation on Quality of Test Suites with Different Metric Guidance*

To answer RQ3, we perform a greedy search algorithm (see 1) to generate test cases with the guidance of five metrics, aiming at boosting the diversities over input domains. Then Mean Squared Error (MSE) between the predictions of test cases and their manual labels are computed. Obviously, a higher MSE value indicates a higher potential of test cases in triggering erroneous behaviors of DNNs. Table IV show the results for five investigated metrics.

In Table IV, mean squared error of original test cases (MSE_{orig}) for three models are quite low; while mean squared error of synthetic test cases (MSE_{new}) are much higher, which means the newly generated test cases have higher chance to trigger erroneous behaviors of DNNs. Furthermore, we notice that synthetic test cases generated with the guidance of different metrics also show different capabilities in revealing erroneous behaviors. Among the five metrics, test cases generated with NC guidance exhibit the weakest capability of triggering erroneous behaviors of the DNNs under test. Similarly, we have discussed in RQ2 that neuron coverage does not change significantly after image transformations. Thus test cases which boost neuron coverage may not be able to trigger more erroneous behaviors of DNNs than randomly selected test cases. In comparison to the ineffectiveness of NC, BD and TD show effectiveness in guiding test case generation with a higher mean squared error. Test case with SD and ND guidance are shown to have the highest chance to trigger more erroneous behaviors of DNNs under test. The observation is consistent with our conclusion in RQ2 that SD and ND exhibit strong fault detection ability, such that synthetic test cases which aim at covering unique neurons or covering unique neuron function regions may have larger chance to detect buggy model behaviors.

To address RQ3: Test suites which boost diversities in neuron coverage show less effectiveness at revealing erroneous behaviors of DNNs; while test suites which aim at covering more unique neurons or diverse neuron function regions are able to trigger more different states of DNNs, thus exhibit greater fault detection abilities. That is, k-multisection Neuron Coverage criteria and Top-k Neuron Coverage criteria can be effective guidance of test case generation; while Neuron Coverage show less effectiveness in guiding test case generation.

VI. THREATS TO VALIDITY

We construct eight and seven MRs for MNIST dataset and CIFAR-10 dataset, respectively. However, the constructed MRs may not cover all possible scenes and some of them may not be too effective at triggering buggy model behaviors. We also carefully configure the parameters of transformations to not break image semantics. This might cause a limited chance to trigger corner-case behaviors of DNNs. The selection of evaluation subjects such as datasets and DNNs could also be a threat to validity. In this study, a commonly studied MNIST dataset and a practical dataset CIFAR-10 are used. We also use the well-known pre-trained models with different sizes. However, some of our results might not generalize to other datasets or models.

VII. RELATED WORK

A. *Testing Deep Learning Systems*

DL systems often exhibit erroneous or unexpected behaviors under certain circumstances which can lead to dangerous situations. Hence, it is imperative to test DL systems in a systematic way. While DNN-based software is fundamentally different from the traditional one as the logic of DNNs is not encoded using control flow, making automated testing of DL systems a challenging problem. Researchers have paid much effort towards this issue. The very first testing criteria Neuron Coverage [11] was proposed to measure the triggering extent of DL system's logic by test inputs. A set of testing criteria inspired by traditional MC/DC was proposed [18] and were used to guide concolic testing of DL systems [19]. Later, a set of multi-granularity testing criteria from neuron-level and layer-level was proposed [12]. Mutation testing techniques were also applied to deep learning testing by asserting faults into DL models [20], [30]. An open-source library TensorFuzz [31] was released, where coverage-guided fuzzing method for neural networks was developed.

B. Metamorphic Testing

Metamorphic Testing (MT) [6], [7] has been successfully applied in many domains for alleviating oracle problem [8] since its first introduction in 1998 [32] by verifying the expected relationships between multiple inputs and outputs. These relationships are defined as “Metamorphic Relations (MRs)”. Murphy et.al [33] applied MT to several machine learning applications for the purpose of verification. Later study has successfully demonstrated that MT can be extended to support validation of machine learning classifiers [34] and search engines [35]. In practice, MT has successfully detected real bugs in GCC and LLVM compiler [36], NASA system [37], web APIs of YouTube and Spotify [38] and Adobe [39]. MT has recently been leveraged to create test oracle for deep learning systems, that is, the behaviors of DL systems should satisfy a predefined relationship (i.e., metamorphic relation) across certain types of transformations on the input. For example, DeepTest [9] checks whether the autonomous car’s steering angles are identical for the same images under different weather conditions or other affine transformations. DeepRoad [10] leveraged MT to test autonomous driving models with GAN-generated images, where metamorphic relations were defined that model behaviors should be consistent under certain driving scenes. Thousands of erroneous behaviors have been successfully detected by DeepTest and DeepRoad, which demonstrates that MT is a promising technique for automatic testing of DL systems. MT has also been leverage to detect implementation bugs in image classifier [40], and to validate a deep learning framework for automatically classifying biology cell images [41].

VIII. Conclusion and Future Work

We perform an large-scale empirical study on essential diversities for deep learning testing criteria to guide effective fault detection. In order to reveal the correlation between diversities and erroneous behaviors of DL systems, we define five metrics to quantify internal state diversities, where four are extracted from existing testing criteria in terms of intrinsic neuron activities, and one is defined from the perspective of neuron slice, and leverage MT to detect erroneous behaviors of DL systems. Next, in order to investigate the capabilities of state diversities in guiding testing process to reveal various defects, we generate test suites with different metric guidance to boost diversities over input domains, and go further step to evaluate the actual testing errors of these test suites with respect to their manual labels. We show that k-multisection Neuron Coverage criteria and Top-k Neuron Coverage criteria can be effective guidance of testing process; while Neuron Coverage show less effectiveness in guiding effective testing. Furthermore, apart from diversities measured by these criteria, diverse distribution of activated neurons could also trigger more erroneous behaviors. Im summary, our analysis provides a comprehensive insight on what are essential diversities for a testing criteria to exhibit better fault detection ability.

To the best of our knowledge, our work is the first comprehensive empirical study on what are the most essential diversities that should be considered by a testing criteria, in order to better detect erroneous behaviors of a DL system. Our work could facilitate more reliable test design and better test quality. Due to the limited computation resources, we mainly select two small-scale datasets and three commonly used DNNs. In the next step, we will continue to investigate more testing criteria on complex architectures. And we are also interested in how can our evaluation metrics reveal the effectiveness of some testing techniques in physical world scenes [42]. Furthermore, we intend to make effort to metamorphic relation selection and construction, as well as a systematic guideline for testing deep learning systems.

Acknowledgments

This work was supported by the National Key R&D Program of China (2018YFB1003901) and the National Natural Science Foundation of China (61572375, 61832009 and 61772263).

References

[1] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *ArXiv preprint*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[3] H. Greenspan, B. van Ginneken, and R. M. Summers, “Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique,” *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1153–1159, 2016.

[4] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, “End to end learning for self-driving cars,” *ArXiv preprint*, vol. abs/1604.07316, 2016. [Online]. Available: http://arxiv.org/abs/1604.07316

[5] J. Saxe and K. Berlin, “Deep neural network based malware detection using two dimensional binary program features,” in *2015 10th International Conference on Malicious and Unwanted Software*, 2015, pp. 11–20.

[6] S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortés, “A survey on metamorphic testing,” *IEEE Transactions on Software Engineering*, vol. 42, no. 9, pp. 805–824, 2016.

[7] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. Tse, and Z. Q. Zhou, “Metamorphic testing: A review of challenges and opportunities,” *ACM Computing Surveys*, vol. 51, no. 1, pp. 4:1–4:27, 2018.

[8] E. J. Weyuker, “On testing non-testable programs,” *The Computer Journal*, vol. 25, no. 4, pp. 465–470, 1982.

[9] Y. Tian, K. Pei, S. Jana, and B. Ray, “Deeptest: Automated testing of deep-neural-network-driven autonomous cars,” in *Proceedings of the 40th International Conference on Software Engineering*, 2018, pp. 303–314.

[10] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 132–142.

[11] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 1–18.

[12] L. Ma, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, T. Su, L. Li, Y. Liu, J. Zhao, and Y. Wang, "Deepgauge: Multi-granularity testing criteria for deep learning systems," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 120–131.

[13] T. Y. Chen, F.-C. Kuo, R. G. Merkel, and T. Tse, "Adaptive random testing: The art of test case diversity," *Journal of Systems and Software*, vol. 83, no. 1, pp. 60 – 66, 2010, sI: Top Scholars.

[14] J. Wan, D. Wang, S. C. H. Hoi, P. Wu, J. Zhu, Y. Zhang, and J. Li, "Deep Learning for Content-Based Image Retrieval: A Comprehensive Study," in *Proceedings of the 22nd ACM International Conference on Multimedia*, 2014, pp. 157–166.

[15] W. Zhou, H. Li, and Q. Tian, "Recent Advance in Content-based Image Retrieval: A Literature Survey," *ArXiv preprint*, 2017, arXiv: 1706.06064. [Online]. Available: http://arxiv.org/abs/1706.06064

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[17] Y. Jing, D. Liu, D. Kislyuk, A. Zhai, J. Xu, J. Donahue, and S. Tavel, "Visual search at pinterest," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 1889–1898.

[18] Y. Sun, X. Huang, and D. Kroening, "Testing deep neural networks," *Arxiv preprint*, vol. abs/1803.04792, 2018.

[19] Y. Sun, M. Wu, W. Ruan, X. Huang, M. Kwiatkowska, and D. Kroening, "Concolic testing for deep neural networks," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 109–119.

[20] L. Ma, F. Zhang, J. Sun, M. Xue, B. Li, F. Juefei-Xu, C. Xie, L. Li, Y. Liu, J. Zhao, and Y. Wang, "Deepmutation: Mutation testing of deep learning systems," in *Proceedings of the 29th IEEE International Symposium on Software Reliability Engineering*, 2018, pp. 100–111.

[21] L. Ma, F. Zhang, M. Xue, B. Li, Y. Liu, J. Zhao, and Y. Wang, "Combinatorial testing for deep learning systems," *Arxiv preprint*, vol. abs/1806.07723, 2018.

[22] E. J. Weyuker, "On testing non-testable programs," *The computer journal*, vol. 25, no. 4, pp. 465–470, 1982.

[23] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *The IEEE International Conference on Computer Vision*, 2015.

[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[25] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ArXiv preprint*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[28] F. Chollet. (2015) Keras. [Online]. Available: http://github.com/fchollet/keras

[29] T. Chen, D. Huang, Z. Zhou, and T. Tse, "Case studies on the selection of useful relations in metamorphic testing," in *Proceedings of the 4th Ibero-American Symposium on Software Engineering and Knowledge Engineering*, 2004, pp. 569–583.

[30] W. Shen, J. Wan, and Z. Chen, "Munn: Mutation analysis of neural networks," in *2018 IEEE International Conference on Software Quality, Reliability and Security Companion*, 2018, pp. 108–115.

[31] A. Odena and I. Goodfellow, "Tensorfuzz: Debugging neural networks with coverage-guided fuzzing," *Arxiv preprint*, Jul. 2018, arXiv: 1807.10875. [Online]. Available: http://arxiv.org/abs/1807.10875

[32] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: a new approach for generating next test cases," Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, Tech. Rep., 1998.

[33] C. Murphy, G. E. Kaiser, L. Hu, and L. Wu, "Properties of machine learning applications for use in metamorphic testing." in *Proceedings of the 20th International Conference on Software Engineering and Knowlege Engineering*, 2008, pp. 867–872.

[34] X. Xie, J. W. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Testing and validating machine learning classifiers by metamorphic testing," *Journal of Systems and Software*, vol. 84, no. 4, pp. 544–558, 2011.

[35] Z. Q. Zhou, S. Xiang, and T. Y. Chen, "Metamorphic testing for software quality assessment: A study of search engines," *IEEE Transactions on Software Engineering*, vol. 42, no. 3, pp. 264–284, 2016.

[36] V. Le, M. Afshari, and Z. Su, "Compiler validation via equivalence modulo inputs," *SIGPLAN Not.*, vol. 49, no. 6, pp. 216–226, Jun. 2014.

[37] M. Lindvall, D. Ganesan, R. Árdal, and R. E. Wiegand, "Metamorphic model-based testing applied on nasa dat: An experience report," in *Proceedings of the 37th International Conference on Software Engineering - Volume 2*, 2015, pp. 129–138.

[38] S. Segura, J. A. Parejo, J. Troya, and A. Ruiz-Cortés, "Metamorphic testing of restful web apis," *IEEE Transactions on Software Engineering*, vol. 44, no. 11, pp. 1083–1099, 2018.

[39] D. C. Jarman, Z. Q. Zhou, and T. Y. Chen, "Metamorphic testing for adobe data analytics software," in *Proceedings of the 2Nd International Workshop on Metamorphic Testing*, 2017, pp. 21–27.

[40] A. Dwarakanath, M. Ahuja, S. Sikand, R. M. Rao, R. P. J. C. Bose, N. Dubash, and S. Podder, "Identifying implementation bugs in machine learning based image classifiers using metamorphic testing," in *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2018, pp. 118–128.

[41] J. Ding, X. Kang, and X. Hu, "Validating a deep learning framework by metamorphic testing," in *Proceedings of the 2nd International Workshop on Metamorphic Testing*. 2017, pp. 28–34.

[42] H. Zhou, W. Li, Y. Zhu, B. Yu, L. Zhang, and C. Liu, "DeepBillboard: Systematic physical-world testing of autonomous driving systems", *ArXiv preprint*, vol. abs/1812.10812, 2018. [Online]. Available: http://arxiv.org/abs/1812.10812