Title
Is Neuron Coverage a Meaningful Measure for Testing Deep Neural Networks?

Author
Harel-Canada, Fabrice Yohann

Publication Date
2019

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Is Neuron Coverage a Meaningful Measure for Testing Deep Neural Networks?

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Computer Science

by

Fabrice Yohann Harel-Canada

2019

ABSTRACT OF THE THESIS

Is Neuron Coverage a Meaningful Measure
for Testing Deep Neural Networks?

by

Fabrice Yohann Harel-Canada
Master of Science in Computer Science
University of California, Los Angeles, 2019
Professor Miryung Kim, Co-Chair
Professor Quanquan Gu, Co-Chair

The safety and reliability of deep learning systems necessitate a compelling demonstration of their encoded behavior. Unfortunately, testing metrics developed for traditional software systems are poorly suited for measuring how well this behavior has been modeled. Recent work in this area has produced a popular solution in neuron coverage, which is inspired by the notion of traditional code coverage. Neuron coverage measures the proportion of neurons activated in a neural network as a proxy for the effectiveness of a test suite.

In this work, we systematically investigate the relationship between neuron coverage and three important test suite properties: defect detection, naturalness, and output diversity. We also propose a novel regularization technique to efficiently induce higher neuron coverage and use it to extend two well-known adversarial attack algorithms. Contrary to expectation, our study finds that increasing neuron coverage actually makes it harder to generate effective test suites. For most of our experimental configurations, higher neuron coverage meant fewer defects detected, less natural tests, and more biased class preferences that harmed output diversity. Our results invoke skepticism that neuron coverage is a meaningful measure for testing deep neural networks.

The thesis of Fabrice Yohann Harel-Canada is approved.

Kai-Wei Chang

Quanquan Gu, Committee Co-Chair

Miryung Kim, Committee Co-Chair

University of California, Los Angeles

2019

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank Dr. Miryung Kim for seeing the potential in me that I wasn't sure was there. I would also like to thank Muhammad Ali Gulzar, Lingxiao Wang, and Dr. Quanquan Gu for their feedback and encouragement. Lastly, I would like to thank friends and family who supported me by reading and at least pretending to understand what I do.

CHAPTER 1

Introduction

Traditional software systems are collections of explicitly encoded actions. Their construction requires a human to conceptualize and decompose the desired actions into a series of discrete instructions. However, many of the tasks we wish computers to perform - e.g., image recognition, speech recognition, etc. - are either too complex or too ill-defined for us to achieve the semantic granularity required to successfully program them. Fortunately, extensive progress into the research of machine learning (ML) algorithms have enabled computers to model much of the expected behavior with minimal human guidance. While many of these algorithms have existed in some form for decades, interest in them has surged in the last several years. Additionally, the increasing effectiveness of deep learning (DL) on a wide range of tasks has led to its integration into many safety-critical systems [2, 15].

Unfortunately, software systems of all types, including and perhaps especially those that learn their behaviors, are prone to exhibiting both unanticipated and undesirable defects. This problem is further compounded for deep neural networks (DNNs) due to their relatively low interpretability and the complex interplay of the neurons that compose them. The combination of these unique challenges, pressing safety concerns, and the prevalence of these defects raises the question: *How can we trust that deep learning systems have learned the desired behaviors?*

In traditional software systems, the creation of effective test suites is an important part of building confidence in observed behavior. However, some test suites are better at detecting faults than others and the coverage of a suite is often used as a measure of its efficacy. Intuitively, the higher the coverage, the better the suite. After all, a test cannot expose faulty logic in lines or paths of code that it never executes. While coverage remains a

widely used quality metric for both government and industry [14, 3], recent studies [13] have questioned the common wisdom that it is a meaningful measure of defect detection capabilities. We investigate this very question in the context of DL models.

However, directly applying the traditional notion of code coverage to the realm of deep learning is problematic. It is common for a single test case to exercise 100% code coverage and still cause the system to behave unexpectedly [27]. Additionally, many ML techniques can be compactly constructed and the size of their code footprint makes for a poor estimate of their behavioral capacity. This problem has led researchers like Pei *et al.* to propose neuron coverage (NC) as an alternative metric that builds on the intuition of code coverage whilst recognizing the unique challenges and structures of neural networks [27]. Neuron coverage describes the proportion of neurons activated beyond a parameterized threshold and is designed to measure the amount of model logic exercised by a set of test inputs. The underlying assumptions behind NC is that the neurons themselves are meaningful enough to track individually and that the magnitude of their activations independently express the encoded logic of the model. It further assumes that increasing NC can improve test suite quality by systematically guiding the discovery of faulty behavior. These assumptions have not been investigated comprehensively and previous work focuses primarily on assessing NC with respect to defect detection alone.

In addition to defect detection, there are at least two other critical properties expected of all successful DL test suites. The first is that the test cases reflect the naturalness of the expected input space. The second is output diversity, which is a newly proposed measure of the degree that model predictions are biased towards "preferred" class labels. We argue that the viability of any DL test metric can be tenably justified on the basis of these three properties. Finally, it is against these properties that we systematically evaluate the utility of neuron coverage as a meaningful testing metric.

We organize our study around three central hypotheses for both large and small deep neural networks trained for the task of image classification:

HYPOTHESIS 1. Defect Detection. Neuron coverage maximization is both strongly

and positively correlated with defect detection (i.e., adversarial attack success).

HYPOTHESIS 2. Naturalness. Neuron coverage maximization is both strongly and positively correlated with the realism and naturalness of the inputs.

HYPOTHESIS 3. Output Diversity. Neuron coverage maximization is both strongly and positively correlated with diverse output predictions.

In order to evaluate these hypotheses empirically, we conduct an extensive correlation analysis performed across 1,623 experimental configurations. From this effort, we make the following contributions:

- A novel adversarial attack regularizer that promotes neural activation diversity and increases NC during test suite creation.
- Formalization of three key properties expected of successful DNN test suites, including the newly proposed notion of output diversity.
- Extensive empirical evidence that neuron coverage is neither positively nor strongly correlated with attack success or input realism. We also demonstrate that certain classes have higher NC by default and the process of maximizing NC biases the test suite to adopt characteristics of the classes with higher default NC, thereby reducing output diversity.

Overall, our results invoke skepticism that neuron coverage is a meaningful measure for testing deep neural networks. The complete code, along with the 1,623 test suites generated for the empirical analysis, is available online.[1]

[1] `https://github.com/fabriceyhc/diversity_attacks`

CHAPTER 2

Related Work

In this section, we review some of the most relevant work along three dimensions: DL systems, DNN testing, and adversarial attacks. Related work relevant to our methodology are described in greater detail in Section 3.

It is important to note that our approach differs from prior work in several respects. First, we extend two standard adversarial attack algorithms - Carlini-Wagner (CW) [4] and Projected Gradient Descent (PGD) [22] - to induce higher neuron coverage whilst still producing test cases that expose faulty behavior in the DL system. Secondly, we incorporate the Inception Score (IS) [30] and Frèchet Inception Distance (FID) [8] as a generic, scalable, and automatic means of evaluating the naturalness of the our generated test cases. Thirdly, we investigate the previously under-investigated issue of output diversity characterizing the distribution of class label predictions for a test suite generated using NC maximization.

2.1 Deep Learning Systems

A deep learning (DL) system is any software system that includes at least one deep neural network (DNN). It may operate as a stand-alone model or be embedded into a larger collection of software components. In this paper, we focus on the DNNs directly.

DNNs are loosely based on biological neural networks like those observed in mammalian brains and have surpassed human performance in a number of application domains. At a high level, these models extract relevant low-level features directly from their inputs and increasingly abstract them into more semantically meaningful components that would be nearly impossible to describe manually.

While there are now many different kinds of DNNs, all of them have a few common architectural features. Each contains basic computational units called neurons, which are connected with one another via edges of varying importance or weight. Neurons apply a nonlinear activation function to the inner product of their inputs and weights to output a value that becomes the input to a subsequent neuron. Layers are used to organize the directed connections between neurons and there is always one or more hidden layers between one input and one output layer. Overall, a DNN can be viewed as a meta-function that aggregates the weighted contributions its neural sub-functions to transform some input into some target output.

The edge weights encode the rules and behavior of the DNN and determining suitable values for them represents the primary learnable challenge. Suboptimally set weights make the DL system vulnerable to erroneous behaviors, but the opacity of these numerically-derived rules make them difficult to understand and debug. For this reason, much research into testing DNNs focus on automatically generating inputs that trigger erroneous behaviors as both a means of evaluating model robustness and as a benchmark against which training, architectural, and hyper-parameter tweaks can be measured. We turn to examples of such work in the next subsection.

2.2 DNN Testing

The vast majority of DNN testing consists of measuring the prediction accuracy on a set of randomly drawn inputs not included in the original training set. However, this is black-box approach is a relatively shallow appraisal of the model's behavioral robustness. Since the test inputs are drawn from the same data set as the training examples, they are likely to share similar characteristics and biases inherent to their collection and labeling, thus undermining generalizability.

Several other research efforts utilized data augmentation to generate new inputs specifically designed to expose faulty behaviors. Pei *et al.* proposed DeepXplore [27], a white-box differential testing algorithm that leverages NC to guide a systematic exploration of the

DNN's internal logic. Input images were modified by one of several domain-specific transformations, all of which were readily visible to observers, but were argued to reflect real-world scenarios. A particular transformed image would be selected for inclusion into a test suite if it fooled at least one of several similarly trained DNNs.

Tian *et al.* proposed DeepTest [33], a gray-box approach for NC-guided test suite generation using a single DNN and metamorphic relations. This effort introduced a wider range of domain-specific transformations to predict the steering angle of an autonomous vehicle from image data alone. At around the same time, Zhang et al. proposed DeepRoad [35] as a GAN-based metamorphic testing approach that utilized a shared latent space representation to perform a sophisticated style transfer of some target road condition - i.e. rain, snow, etc. - to a given source image. This approach made no attempt to systematically explore the possible input space via a metric like NC, but did demonstrate that significant transformations could expose faulty behaviors while still preserving image naturalness.

Lastly, Ma *et al.* [21] proposed DeepGauge to significantly expand on the idea of NC. They introduce three new neuron-level coverage criteria and two layer-level coverage criteria to produce a multi-granular set of insights into DNN robustness. Unlike the previously aforementioned work, Ma *et al.* embraced standard adversarial techniques to generate test suites and demonstrated that not only could they expose faulty behavior, but they generally exhibited higher coverage scores for most of the proposed criteria. This contrasted with DeepXplore's conception of NC, which did not change significantly for any of the adversarially generated test suites and therefore could not provide useful guidance in testing DNNs. In Section 6, we discuss some theoretical limitations with any coverage criteria predicated on the basis on individual neurons and in Section 7 propose the use of channel-based criteria as a promising future path of investigation.

2.3 Adversarial Attacks

The many successes of deep learning have made it important to understand the robustness of their performance against malevolent agents. One such area of research is that of adversarial

attacks. By adding a very small, often visually indistinguishable, perturbation to an input, it is possible to fool a DNN into misclassifying a previously known input with high confidence.

While adversarial attacks employ a variety of methods to induce the erroneous behavior, their effectiveness is largely measured by the post-perturbation accuracy of the attacked inputs and distance from the original inputs. In this last respect, most norm-minimizing attacks represent a small subset of semantically equivalent inputs that can cause a DNN to misclassify them. Some work has attempted to side-step the norm-minimizing constraint of adversarial attacks with mixed results. For the present evaluation of neuron coverage, we accept the standard norm-minimizing constraint because it eliminates the potentially spurious influences of more substantial transformations and simplifies the image naturalness analysis.

Lastly, daversarial attack algorithms offer both targeted and untargeted approaches for perturbing the inputs to be predicted as some other class. Targeted attacks allow you to specify the desired output distribution and are therefore of less interest to the evaluation of the role of NC in shaping the output distribution of the test suite. We exclusively use untargeted formulations of the attacks to give them the freedom to perturb the class relation in whichever way NC maximization incentivizes.

CHAPTER 3

Methodology

In this section, we describe the datasets, DNNs, and adversarial attack algorithms selected to provide a diverse and varied cross-section of potential use cases. Readers will also find other methodological details common to all experimental hypotheses, such as how NC is measured and how we designed a novel means of increasing NC via a diversity-promoting regularizer.

3.1 Datasets and DNNs

MNIST [20] is a large, well-studied dataset containing 28x28x1 pixel images representing handwritten digits from 0 to 9. The digits are normalized to fit within a 20x20 pixel box and are centrally aligned by computing the center of mass of the pixels. Instances of each class also appear against a blank background. These and other preprocessing alterations make this dataset ideal for many ML problem evaluations but make it difficult to assess the naturalness of their adversarial counterparts. The two realism metrics we employ - IS and FID - are tuned on the internal structures of natural images (i.e. those taken of the natural world) and generally have both foregrounds and backgrounds. Adversarial attacks generally add somewhat noticeable distortions to the backgrounds of the digits that result in higher realism scores from IS and FID. For this reason, we exclude MNIST from our evaluation of research question into whether NC is a useful guide to image realism.

We constructed and trained four different DNNs with a variety of architectures to explore their possible influences on our NC evaluation. The first pair of DNNs are both fully connected linear models where the primary difference between them is the number of layers.

The second pair of DNNs fix the number of layers, but vary the primary convolutional layer type from 1D to 2D. All MNIST DNNs were trained for 10 epochs using an Adam optimizer [17].

CIFAR10 [18] is a dataset containing 32x32x3 RGB pixel images representing ten mutually exclusive classes of naturally occurring entities that are suitable for IS and FID realism measurement. We test two well-known pre-trained DNNs: a 56-layer ResNet [7, 11] and a 121-layer DenseNet [9, 28], both of which achieve competitive performance on this dataset.

Lastly, all DNNs were implemented in Pytorch [26]. See Table 3.1 for more details on all the DNNs under evaluation.

DNNs	Task Dataset	Primary Layer Type	# Layers	# Neurons	Test Accuracy (%)
FCNet5	MNIST	Linear	5	478	97.96
FCNet10	MNIST	Linear	10	3,206	97.92
Conv1DNet	MNIST	Conv1D	4	35,410	98.17
Conv2DNet	MNIST	Conv2D	4	15,230	98.82
ResNet56	CIFAR10	Conv2D	56	532,490	93.39
DenseNet121	CIFAR10	Conv2D	121	563,210	95.04

Table 3.1: Relevant DNN Architectural Details

3.2 Measuring Neuron Coverage

Pei *et al.* formally define neuron coverage by the following:

$$neuron\_cov(T, \mathbf{x}, t) = \frac{|n|\forall \mathbf{x} \in T, out(n, \mathbf{x}) > t|}{|N|}$$

where $N = \{n_1, n_2, ...\}$ represents all the neurons in the DNN; $T = \{\mathbf{x}_1, \mathbf{x}_2, ...\}$ represents all the test inputs (i.e. those to be perturbed); $out(n, \mathbf{x})$ is a function that returns the output value of neuron n for a given test input $\mathbf{x}$ which is *scaled* to be between 0 and 1 based on the minimum and maximum neuron activations for the layer; and t is the user-set threshold for determining whether a neuron is sufficiently activated.

For illustration, Figure 3.1 depicts a simple neural network with a single hidden layer.

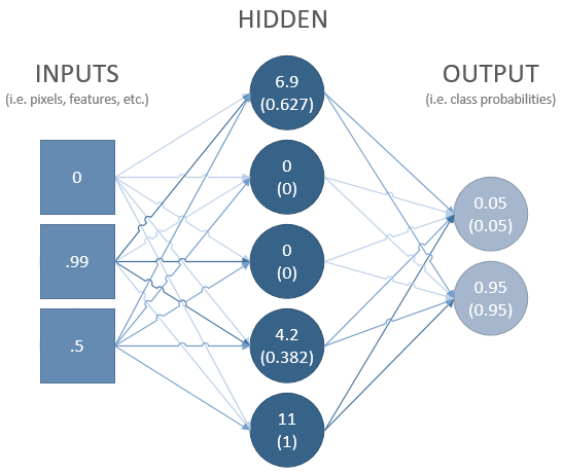

Figure 3.1: Neural Network with single hidden layer

Each circular node corresponds to neurons that are organized and color coded by layer. The hidden and output layer neurons also contain their layer-wise scaled activations in parentheses for comparison against a chosen threshold t. If $t = 0$, then $neuron\_cov(T, \mathbf{x}, t) = 5/7 = 0.714$, or if $t = 0.75$, then $neuron\_cov(T, \mathbf{x}, t) = 2/7 = 0.286$. Selecting an appropriate threshold t was an open issue in early NC research. In this paper, we track NC across four different thresholds $t \in \{\ 0,\ 0.2,\ 0.5,\ 0.75\ \}$ to ensure a comprehensive analysis and because prior work used similar values.

3.3 Adversarial Attack Algorithms

We select two effective attack algorithms with widespread usage in the related literature.

Carlini-Wagner (CW) attacks [4] are a set optimization-based techniques for three different norms: L_∞, L_0, L_2. In principle, the CW attack is designed to approximate the solution to the following:

$$\min c \cdot f(x + \delta) + ||\delta||_p \quad \text{such that} \quad x + \delta \in [0, 1]^n$$

where c is a scaling constant, f is a suitable objective function, x are the original inputs to attack, and δ is the applied perturbation tensor. The $||\cdot||_p$ is the L_p distance between

the original inputs x and the perturbed attack images $x + \delta$. In this effort, we use the L_∞ norm where distance is measured by the pixel with the greatest magnitude change from its original value.

Projected Gradient Descent (PGD) attacks [22] are another first-order technique designed to exploit areas of highest loss within a given norm-ball of an original input. From many random starting points, gradient descent is performed to converge to a local maxima representing the most adversarial form of a given input. To maintain a similar threat model to the CW attack, we use their L_∞ norm-box attack to generate the PGD adversaries. We further specify three different perturbation limits $\epsilon \in \{ 0.1, 0.2, 0.3 \}$ for the norm bounds to explore its possible effects on NC.

3.4 Extending Attacks to Increase Neuron Coverage

Standard adversarial attacks aim at creating perturbed inputs that achieve two primary objectives - maximizing misclassification losses and minimizing L_p norm distance from the original inputs. Previous research [27, 21] demonstrated that the standard formulations of these algorithms are not able to produce any significant variation in NC and cited this as a deficiency with adversarial attacks in general.

In order to address this concern, we devised a novel adversarial attack regularizer to incorporate the maximization of NC as a third objective. Like many regularizers, ours works by penalizing an undesirable condition within the optimization landscape, and here we target the diversity of layer-wise activations within the DNNs. In this context, diversity characterizes the degree of uniformity in the neural activation distributions and low diversity (i.e. low uniformity) is penalized. Figure 3.2 illustrates the effects of regularization with a cross-section from one of the DNNs under test. It also highlights the inverse relationship between NC and divergence from diversity.

The inherently targeted penalization is a consequence of the relative infrequency of inactive neurons in DNNs - approximately 5-30% for our models. Since the vast majority of neurons are activated, the promotion of diversity has the effect gravitating all neurons

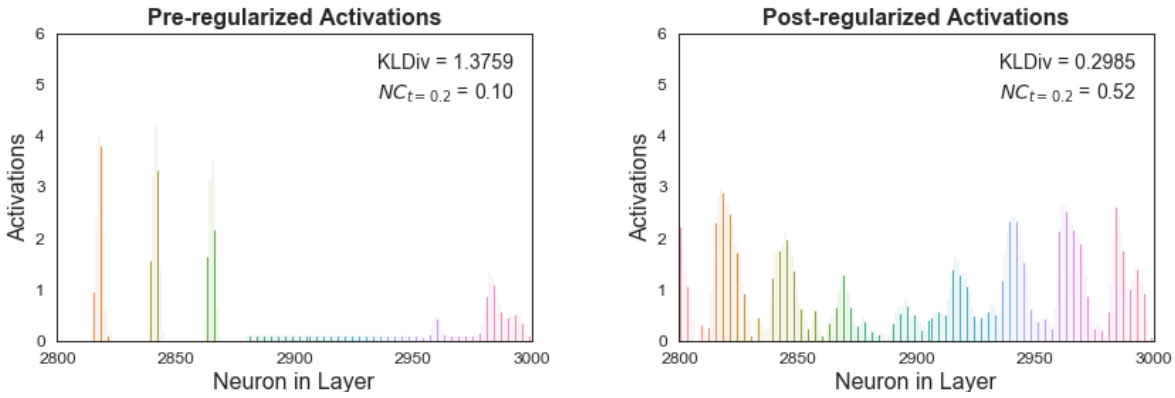

Figure 3.2: Conv2DNet Neural Activations Before and After Regularization

toward the average magnitude of activation. With a sufficiently high weight placed on this objective, the regularization for diversity can be an effective means of inducing previously inactive neurons to fire and thereby increase NC. For this evaluation, we use a logarithmic scale to explore the diversity regularization weights: $\{0, 10^0, 10^1, 10^2, 10^3, 10^4, 10^5\}$.

Below we show the extended CW attack as an example application of the new regularizer:

$$\min c \cdot f(x+\delta) + ||\delta||_p + \lambda \cdot \sum_l div(out_l(x+\delta), U)$$

$$\text{such that} \quad x + \delta \in [0,1]^n$$

where λ is a user-set diversity weight to control how strongly we wish to induce higher NC; $div(\cdot)$ is a divergence function; $out_l(\cdot)$ is a function that returns the neural activations from the l^{th} layer of the DNN for the perturbed inputs $x + \delta$; and U represents a uniform distribution. We use the Kullback-Leibler (KL) divergence [19] to implement our $div(\cdot)$ function, but others could be suitable.

Table 3.2 demonstrates of the regularizer's effectiveness. It shows a breakdown of the average and maximum percent increases in NC over the baseline NC of the original test suite images for MNIST and CIFAR10, respectively. Naturally, DNNs that are already highly activated are more difficult to activate further, making the $\text{NC}_{t=0}$ undesirable for comparison purposes. On the other hand, $\text{NC}_{t=0.5}$ and $\text{NC}_{t=0.75}$ represent significantly smaller portions

of the network and the significance / utility of NC at this level is less clear. Therefore, we report primarily on the $\text{NC}_{t=0.2}$ for visual figures in the experimental results.

	$\textbf{NC}_{t=0}$ (%)			$\textbf{NC}_{t=0.2}$ (%)			$\textbf{NC}_{t=0.5}$ (%)			$\textbf{NC}_{t=0.75}$ (%)		
DNNs	Orig	Avg ↑	Max ↑	Orig	Avg ↑	Max ↑	Orig	Avg ↑	Max ↑	Orig	Avg ↑	Max ↑
FCNet5	96.09	0.56	0.66	61.21	22.59	61.72	15.01	45.75	173.98	4.33	38.72	171.88
FCNet10	78.52	7.65	18.63	16.79	54.68	98.90	3.70	38.27	71.63	0.89	63.77	123.91
Conv1DNet	68.08	4.82	14.50	12.67	8.77	45.33	1.26	12.85	139.58	0.48	15.16	63.38
Conv2DNet	94.96	1.77	4.18	23.89	9.11	36.47	6.66	25.48	55.69	1.23	67.88	122.04
ResNet56	95.07	0.22	0.40	26.87	4.31	11.77	5.42	6.17	21.76	1.29	6.20	15.71
DenseNet121	96.46	0.04	0.06	12.88	7.00	14.49	1.20	6.59	15.85	0.16	11.77	31.87
Average	90.09	1.91	4.86	24.26	14.72	36.87	4.98	18.48	64.51	1.23	27.68	72.05

Table 3.2: Original NC, Average % Increase from Original NC, and Maximum % Increase from Original NC

As an implementation note, our regularizer can target a specific layer, contiguous and non-contiguous layer subsets, or all layers simultaneously. In our experiments, we primarily focus on targeting one specific layer at a time, primarily to evaluate the sensitivity of NC to the regularization. The smaller DNNs trained for MNIST have each of their layers targeted for diversity promotion in turn. However, the larger DNNs trained for CIFAR10 have too many layers and so we sample six evenly distributed layers, starting from the first hidden layer and ending at the output layer. We also conducted an experiment where we targeted all layers simultaneously and are content to merely mention that it was also successful in increasing NC with smaller diversity weights. Ultimately, targeting layers individually provides more varied results useful to evaluating our experimental hypotheses.

At this point, it may be objected that the regularizer ought to have incorporated measures of NC directly rather than implicitly. The time required to calculate NC increases as a function of network capacity (i.e. number of layers and neurons) and primary layer type - convolutional layers take significantly longer than linear layers. In our evaluation we observed calculation times between 1 and 94 seconds depending on the target network. Since the attack algorithms we selected are highly iterative, involving thousands of cycles tuning the perturbation vectors, calculating NC in each iteration is not feasible. While it may have been strictly preferable to incorporate NC directly, the competing time and usability concerns called for a nearly instantaneous formulation like the one we've proposed.

CHAPTER 4

Experiments

For each combination of experimental variables described in Section 3, we evaluate the relationship between NC and our quality metrics for DNN test suites - defect detection, naturalness, and output diversity. The variables are summarized in Table 4.1 below:

Variable	**Values**
Adversarial Attacks	CW, PGD
DNNs	FCNet5, FCNet10, Conv1DNet, Conv2DNet, ResNet56, DenseNet121
Datasets	MNIST, CIFAR10
Target Layers	Varies
Diversity Weights	$0, 10^0, 10^1, 10^2, 10^3, 10^4, 10^5$
c **Confidence (CW)**	$0, 20, 40$
ϵ **Limit (PGD)**	$0.1, 0.2, 0.3$

Table 4.1: Experimental Variables

For each dataset, we randomly select a test suite of one hundred input images for adversarial perturbation. Every class is equally represented and the DNNs have not seen any of these instances during training. Collectively, the test suite and the specific combination of variables form an experimental configuration. For each configuration, we use an adversarial attack algorithm to automatically generate a DL test suite and then calculate it's $\text{NC}_{t=0}$, $\text{NC}_{t=0.2}$, $\text{NC}_{t=0.5}$, $\text{NC}_{t=0.75}$, IS, FID, and output diversity. Lastly, we perform a broad correlation analysis of 1,623 test suites to measure the strength, direction, and significance of the relationship between NC and our three critical test properties.

CHAPTER 5

Findings

In this Section, all the results from the experiments outlined in Section 4 are presented and examined in detail. The strength and directionality of each relationship between NC and our three critical test suite properties are explained in turn and various implications discussed.

Correlations are computed from data grouped by adversarial attack and DNN for each NC threshold. These groupings reflect the two most high-level variables in the investigation and help to position the results in a way most relevant to practitioners deliberating upon methods and models. All correlations are presented in tabular form and we visualize a sample of the $NC_{t=0.2}$ results for CW to highlight the outcomes of the most prevalent threshold used in the related literature. We adopt a standardized delineation of correlative significance laid out by Ratner [29] to characterize values between 0 and ± 0.3 as weak, ± 0.3 to ± 0.7 as moderate, and ± 0.7 to ± 1.0 as strong. Correlation coefficients are also color coded according to whether or not they are statistically significant. Gray indicates a p-value > 0.05 and such values are discounted in our subsequent analyses. **Emboldened** values indicate that the results support the associated hypothesis and all others undermine it.

5.1 Defect Detection

The primary purpose of a test suite is to exercise system logic and thereby expose potentially defective behaviors. Its value increases with its defect detection rate (DDR), which describes the proportion of constituent test cases that result in undesirable behavior. Once detected, those behaviors can be tweaked to better handle the associated region of the input space and then compared against the test suite to gauge the degree of improvement. In this way,

test suites with a high DDR become an indispensable reference point for subsequent cycles of bug fixing, model retraining, and other robustness improvements.

5.1.1 Measuring Defect Detection

Since our approach relies on adversarial attacks to generate test suites, we are able to equate the attack success rate (ASR) with DDR and use both interchangeably. If we allow *pert_acc* to represent the DNN's classification accuracy on T, or the adversarialy perturbed suite of test inputs, then the ASR is defined simply as $1 - pert\_acc$.

5.1.2 Hypothesis & Results

H1. Defect Detection Neuron coverage maximization is both strongly and positively correlated with defect detection (i.e. adversarial attack success).

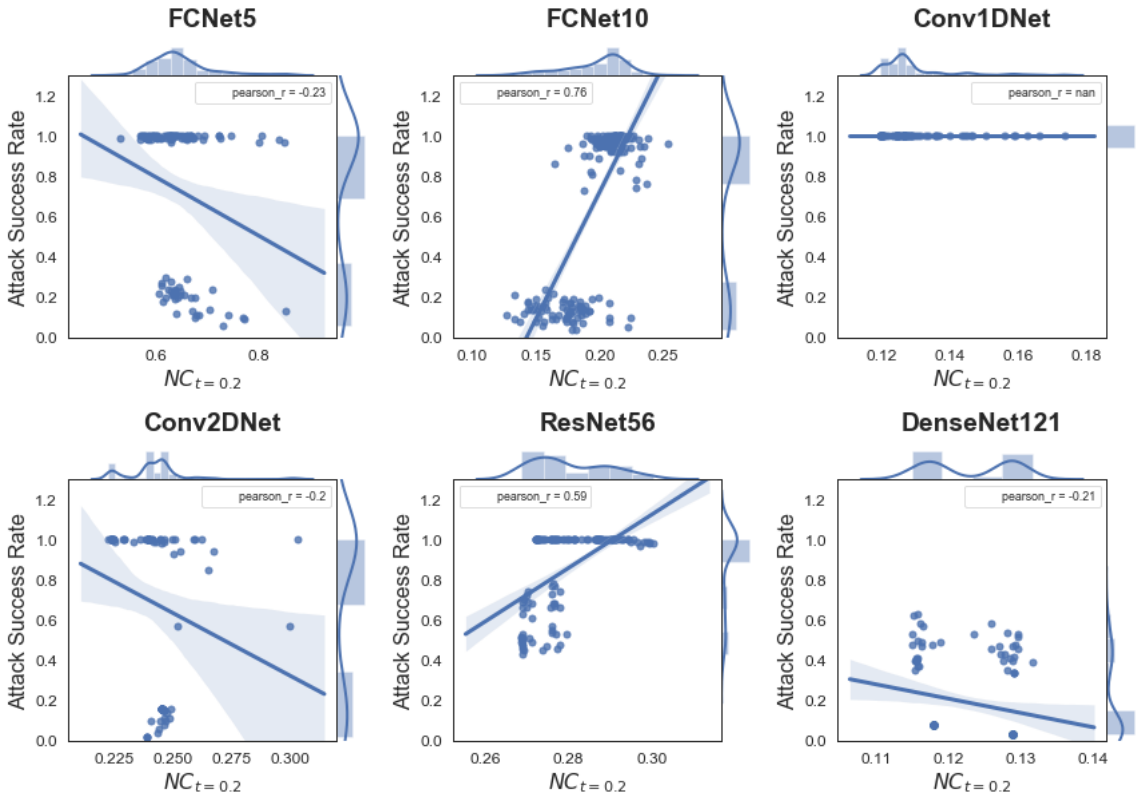

Figure 5.1: $NC_{t=0.2}$ vs ASR

This hypothesis addresses whether the maximization of NC is useful for creating test suites that expose more defective behavior in the DNN under test. Figure 5.1 visualizes this relationship broken down by DNN for the CW attack. The various subfigures portray NC as varied and volatile guide for anyone hoping to improve the defect detection capability of their test suites. Even for models that share a large degree of architectural similarity, like the FCNet5 and FCNet10 models, the correlations differ in both strength and direction, reinforcing the unpredictability of the metric.

Table 5.1 shows the Pearson correlations between NC and ASR broken down by attack algorithm, network, and t threshold. As can be seen from the table, a mere 2 (4%) of the 48 correlations satisfy the hypothesis that NC is both positively and strongly correlated with defect detection. The two strongly positive results were also observed from a single DNN - FCNet10 at $\text{NC}_{t\in\{0.2,0.5\}}$ - suggesting a limited generalizability to the zoo of models currently in existence. Independent of direction, a 58% majority of scenarios show a weak correlation while another 27% are merely moderate. Independent of strength, the correlation is positive in only 42% of measurements, negative in 50%, and non-existent in 8%.

	CW Correlations				**PGD Correlations**			
DNNs	$\textbf{NC}_{t=0}$	$\textbf{NC}_{t=0.2}$	$\textbf{NC}_{t=0.5}$	$\textbf{NC}_{t=0.75}$	$\textbf{NC}_{t=0}$	$\textbf{NC}_{t=0.2}$	$\textbf{NC}_{t=0.5}$	$\textbf{NC}_{t=0.75}$
FCNet5	-0.2033	-0.2295	-0.1754	0.0667	-0.0999	-0.5242	-0.5218	-0.3249
FCNet10	-0.6673	**0.7632**	**0.7515**	0.0382	-0.1819	-0.1572	-0.0974	0.1418
Conv1DNet	NA	NA	NA	NA	0.5844	-0.3687	0.1000	0.0507
Conv2DNet	-0.1611	-0.2025	-0.2943	-0.2334	0.0807	-0.0394	-0.1558	-0.3586
ResNet56	-0.4569	0.5880	0.5849	0.5698	-0.1141	0.5153	0.5293	0.2135
DenseNet121	-0.8269	-0.2121	-0.0610	0.1286	0.1943	0.1759	0.2016	0.1087
All	-0.3077	0.1679	0.1403	0.1623	0.3413	-0.1157	-0.2063	-0.1876

Table 5.1: Pearson Correlation between NC & ASR

Across all CW experimental results, we observed a *moderately negative* correlation for $\text{NC}_{t=0}$ and *weakly positive* correlations for the remaining NC thresholds. The reverse held for the PGD correlations: *moderately positive* for $\text{NC}_{t=0}$ and *weakly negative* for the remaining NC thresholds. There was also no single t threshold that consistently correlated with attack success - either positively or negatively - for all DNNs. The wide degree of variation and the lack of any consistent results suggest a sensitivity and volatility to NC that undermine its

reliability, utility, and meaningfulness.

Answer 1. Our findings reject the hypothesis that maximizing NC is both strongly and positively correlated with defect detection. Only 4% of all experimental results supported the hypothesis, a further 50% were actually negatively correlated, and another 8% were uncorrelated. This means that maximizing NC is most likely to harm the defect detection capabilities of your DL test suites.

5.2 Naturalness

Naturally, the value of a test suite is also proportional to how realistic it appears. After all, DL systems are designed to solve real-world problems. Even if a test reliably induces faulty behavior, doing so by breaching the bounds of realism necessarily undercuts its validity. We now turn our attention to the question of whether maximizing NC can help generate valid test suites that reflect the naturalness of the expected input space.

5.2.1 Measuring Naturalness

While many kinds of tests have either written or unwritten rules defining what underlying variations in them are permissible, the image space we study is much more complex and open ended. Unfortunately, appraising the visual quality of an image is highly subjective and there is still no definitive solution on how to formalize its measurement for research or application purposes. This is not to say that there has been no progress in attempting to measure naturalness. In fact, research into generative adversarial networks (GANs) [6] has produced several popular metrics for this very purpose. We selected the two most highly cited metrics from GAN literature to objectively measure naturalness in our evaluation.

The **Inception Score (IS)** [30, 1] attempts to formalize the concept of naturalness for a set of images by breaking the concept into two criteria:

- *Salience.* Of the possible class labels that could be applied to an individual image,

only one has a high probability and the others are very low. This corresponds to the image being highly recognizable.

- *Diversity.* There are many different kinds of classes present across all images in the set.

Normally, the second criteria would make the cross-field application of this GAN naturalness metric to adversarial attacks problematic as there exists no requirement to balance the post-attack class prediction distribution. However, since we actually are interested in evaluating the relationship between NC and the diversity of the DNN's output space, the IS for naturalness arguably provides another point of reference for both hypotheses.

The **Frèchet Inception Distance (FID)** [8, 23] is a measure of similarity between two datasets of images. It is calculated by computing the Frèchet distance between two Gaussians fitted to feature representations of the final average pooling layer within the InceptionV3 network [32]. The inventors, Heusel *et al.*, provide evidence that the metric captures the similarities of generated images better than IS and that it correlates well with human judgement of visual quality. Unlike IS, a lower FID value indicates that the images are more realistic since the distance from the original images is smaller. Therefore, only a strong *negative* correlation for FID can be interpreted favorably for **H2**.

Lastly, recall that only the CIFAR10 dataset is considered for this hypothesis due to a "naturalistic" assumption built into the metrics we employed for this analysis. Both metrics use components of the InceptionV3 network, which is trained to recognize ImageNet classes [5]. CIFAR10 contains a subset of these classes, but ImageNet *does not* overlap with the highly pre-processed MNIST digits against a blank background.

5.2.2 Hypothesis & Results

H2. Naturalness Neuron coverage maximization is both strongly and positively correlated with the realism and naturalness of the inputs.

This hypothesis seeks to address the test suite's degree of naturalness in a more objective

manner than the subjective manual spot-checks performed in most related work. Figure 5.2 depicts the relationship for IS broken down by metric and DNN for the CW attack. Once again, the wide fluctuation of strongly negative and strongly positive correlations underscore the volatility of NC. Table 5.2 shows the Pearson correlations between NC and IS broken down by attack algorithm, network, and t threshold. As can be seen from the table, only 1 out of 16 correlations satisfy the hypothesis that NC is both positively and strongly correlated with improving input naturalness. Independent of direction, 38% of scenarios show a weak correlation while another 44% are merely moderate. Independent of strength, the correlation is positive in only 56% of measurements. When measured across all CW and PGD experimental configurations, the relationship is weakly positive for $\text{NC}_{t=0}$ and weakly negative for the remaining NC thresholds.

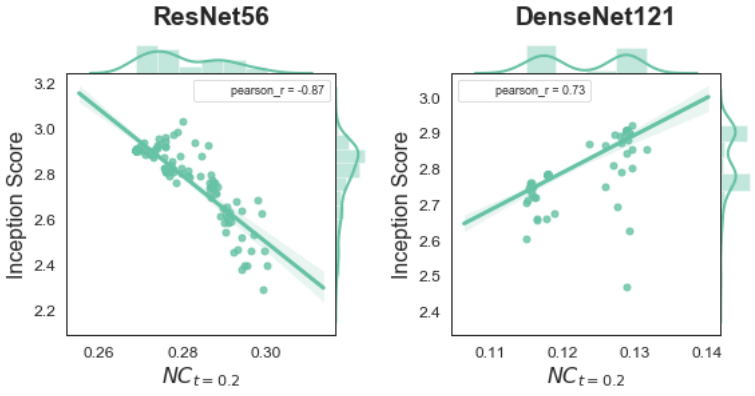

Figure 5.2: $\text{NC}_{t=0.2}$ vs IS

	CW Correlations				PGD Correlations			
DNNs	$\textbf{NC}_{t=0}$	$\textbf{NC}_{t=0.2}$	$\textbf{NC}_{t=0.5}$	$\textbf{NC}_{t=0.75}$	$\textbf{NC}_{t=0}$	$\textbf{NC}_{t=0.2}$	$\textbf{NC}_{t=0.5}$	$\textbf{NC}_{t=0.75}$
ResNet56	0.0876	-0.8708	-0.8093	-0.5868	0.3385	-0.3809	-0.5187	0.0565
DenseNet121	0.5683	**0.7311**	0.6272	0.4624	-0.1497	0.1557	-0.0785	0.1860
All	0.1349	-0.1654	-0.1817	-0.1486	0.0245	-0.1265	-0.1649	-0.0893

Table 5.2: Pearson Correlation between NC & IS

Figure 5.3 shows the relationship for FID broken down by metric and DNN for the CW attack. Unlike the more mixed results for IS, these subfigures reflect the general finding that increasing NC almost invariably increases the distance of the test suite from its source

material. In fact Table 5.3 confirms that not a single configuration yielded evidence in favor of our hypothesis.

It is worth noting that more than half of the PGD results across both IS and FID are grayed out, indicating statistical insignificance. This can likely be explained by different strategies the attacks use to control distortion. PGD attacks enforce a more strict ϵ perturbation limit while the perturbations of CW attacks are theoretically unbounded and merely aim at minimizing the distortion as much as possible. Since the limit more tightly constrains the range of measurements, it makes it more difficult to assess the correlation with NC. Relaxing the perturbation limits might bring the PGD into closer alignment with CW, but the fundamental conclusion remains the same: increasing NC makes it more difficult to create realistic test suites.

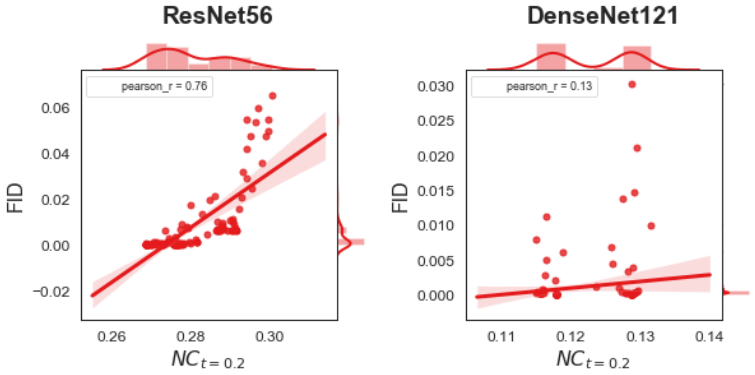

Figure 5.3: $NC_{t=0.2}$ vs FID

	CW Correlations				PGD Correlations			
DNNs	**$NC_{t=0}$**	**$NC_{t=0.2}$**	**$NC_{t=0.5}$**	**$NC_{t=0.75}$**	**$NC_{t=0}$**	**$NC_{t=0.2}$**	**$NC_{t=0.5}$**	**$NC_{t=0.75}$**
ResNet56	0.2664	0.7558	0.7525	0.5899	-0.0323	0.4179	0.4599	-0.1400
DenseNet121	-0.2250	0.1350	0.2441	0.3492	0.2637	-0.0556	0.1564	-0.1227
All	-0.2811	0.3576	0.3635	0.3326	0.1566	-0.1190	-0.0935	-0.1701

Table 5.3: Pearson Correlation between NC & FID

Figures 5.4 and 5.5 provide an opportunity to manually inspect a sample of two test suites with an approximately 14% difference in $NC_{t=0.2}$. While both sets of images are noticeably distorted, test suite # 140 is clearly more severe. Test suite # 33 has an IS about 33% higher and an FID about 29x smaller, both confirming the intuition that it is substantially more

natural. Here the additional NC made for noisier, more noticeably perturbed test cases and therefore a less valuable test suite overall.

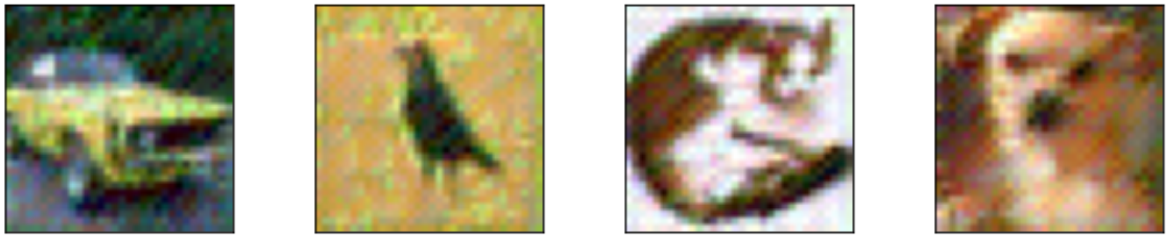

Figure 5.4: ResNet56 Test Suite #33. $NC_{t=0.2}$: 0.289 - IS: 1.972 - FID: 0.100

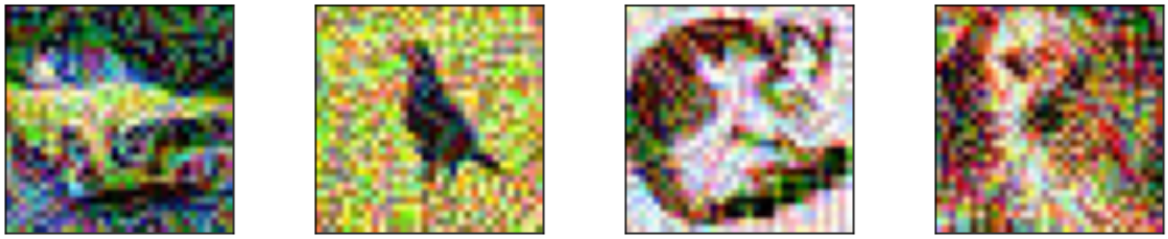

Figure 5.5: ResNet56 Test Suite #140. $NC_{t=0.2}$: 0.330 - IS: 1.480 - FID: 2.935

Answer 2. Our findings reject the hypothesis that maximizing NC is both strongly and positively correlated with naturalness. Only 3% of all experimental results supported the hypothesis and a further 56% were actually negatively correlated. This means that maximizing neuron coverage is most likely to undermine the naturalness of your DL test suites.

5.3 Output Diversity

The final dimension of our investigation probes the relationship between NC and the diversity of DNN predictions for different class labels. Placing the focus on the outputs underscores the idea that, regardless of the true input class, the actual logic being exercised reflects the class the model predicts. We find support for this view in several recent works. Firstly, Pei *et al.* established that different classes activate distinctive patterns of neurons [27] and

therefore misclassifications can be interpreted as an instance of some class exhibiting a neural pattern that more closely overlaps with some other class. Secondly, recent work by Ilyas *et al.* [12] has demonstrated that adversarial examples work by incorporating largely unnoticeable, non-robust features of other classes to confuse the DL system.

As an illustrative example, consider a well-trained DNN and a balanced test suite comprised of instances evenly drawn from multiple classes. If every test in this suite was predicted to belong to one single class, this strongly suggests that a) the misclassifications occurred because the tests incorporated features of the predicted class and b) that all of their neural patterns are generally similar and overlapping. In terms of coverage, this outcome is strictly worse than if the predictions are split between two classes and therefore a broader set of neural patterns. Extending this idea to its logical conclusion, the best situation is one where the predictions are evenly split between all classes as this maximizes the variety of cross-class features and exercises the widest range of neural patterns. Since the ultimate goal of test coverage is to exercise as much behavior as possible, we argue that maximizing output diversity is an indispensable property expected of any DL test suite.

5.3.1 Measuring Output Diversity

We characterize the output diversity of a DNN test suite by the distribution of class predictions when the input diversity is uniform (i.e. the test suite contains an equal number of inputs from each class). A test suite with the lowest possible diversity reflects that all inputs are predicted to belong to only one class. The highest possible diversity reflects that each class is equally represented in the predictions of the DNN.

This formulation makes it possible to use Shannon Entropy [31] to succinctly score the diversity of the output class distribution. Low entropy entails low diversity and vice versa. For convenience, we further normalize the entropy to be on a scale between a low of 0 and a high of 1 by dividing each test suite by the maximum entropy possible given the total number of classes. Thus, we formally define the output diversity for a test suite, T, with C possible classes - indexed by k - below:

$$output\_div(T) = \frac{\sum_{t \in C_k} P_{t=C_k} log P_{t=C_k}}{\frac{1}{|C|} log \frac{1}{|C|}}$$

where $P_{t=C_k}$ represents the percentage of the test cases t predicted to belong to class C_k; and $|C|$ is set notion indicating the total number of classes.

5.3.2 Hypothesis & Results

H3. Output Diversity. Neuron coverage maximization is both strongly and positively correlated with diverse output predictions.

This hypothesis addresses whether the maximization of NC has an adverse affect on the distribution of predicted classed by the DNN under test. Figure 5.6 visualizes this relationship broken down by DNN for the CW attack. While some of the subfigures serve to reinforce the takeaway that NC is volatile metric, half exhibit flat relationships. This highlights that, for many slices of the data, NC is probably not relevant to promoting output diversity.

	CW Correlations				**PGD Correlations**			
DNNs	$\mathbf{NC}_{t=0}$	$\mathbf{NC}_{t=0.2}$	$\mathbf{NC}_{t=0.5}$	$\mathbf{NC}_{t=0.75}$	$\mathbf{NC}_{t=0}$	$\mathbf{NC}_{t=0.2}$	$\mathbf{NC}_{t=0.5}$	$\mathbf{NC}_{t=0.75}$
FCNet5	0.3263	0.4301	0.3593	0.1059	0.3723	**0.7378**	0.5176	0.2604
FCNet10	**0.7707**	-0.6962	-0.7558	-0.0189	0.4204	0.3382	0.2504	0.0165
Conv1DNet	0.3861	0.0786	0.1000	0.1455	-0.5669	0.1775	-0.2825	-0.1515
Conv2DNet	-0.2201	-0.0217	0.2887	0.3394	0.3350	0.1062	-0.0438	-0.0826
ResNet56	-0.2679	0.4513	0.4324	0.4307	-0.0213	0.0740	0.0892	0.0760
DenseNet121	**0.7873**	0.1086	-0.0380	-0.1793	-0.0888	-0.1801	-0.0607	0.0796
All	0.5252	-0.1301	-0.1472	-0.1911	0.3700	0.0549	0.0923	0.0451

Table 5.4: Pearson Correlation between NC & Output Diversity

To discern a more complete view of the situation, consider Table 5.4, which shows the Pearson correlations broken down by attack algorithm and t threshold. As can be seen from the table, a mere 3 (6%) of the 48 configurations satisfy the hypothesis that NC is both positively and strongly correlated with output diversity. Independent of direction, a

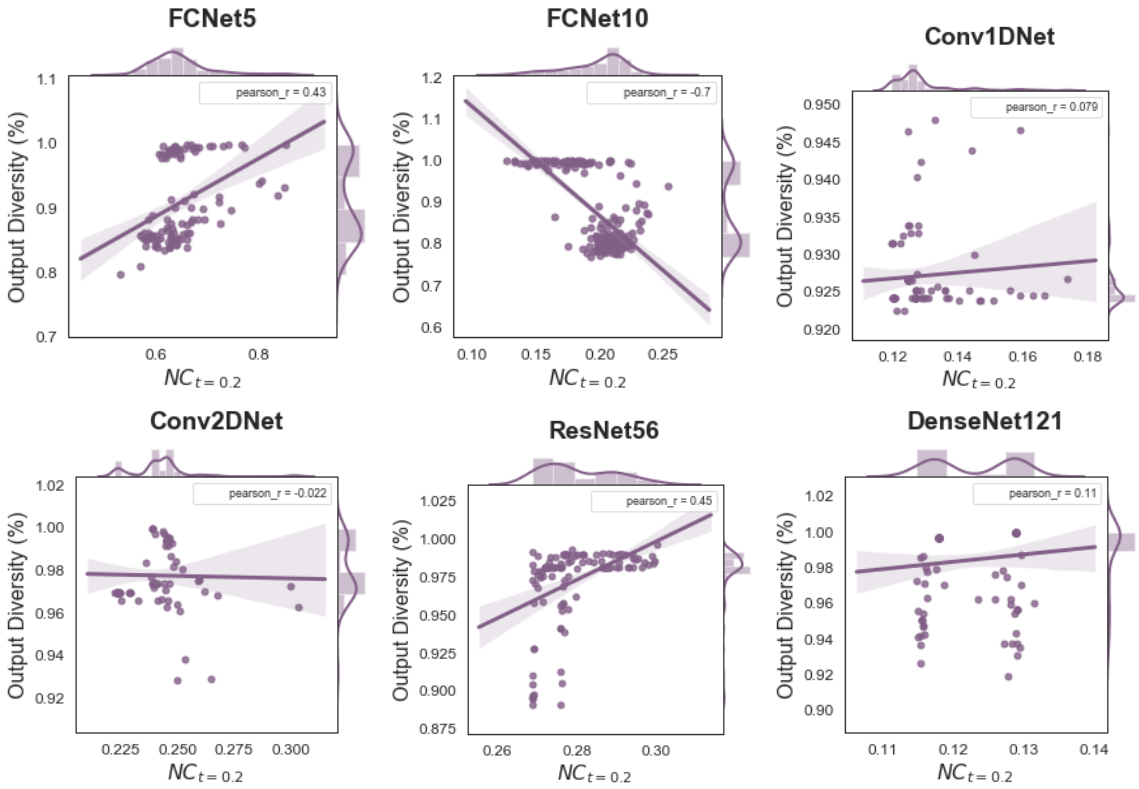

Figure 5.6: $NC_{t=0.2}$ vs Output Diversity

60% majority of scenarios show a weak correlation while another 30% are merely moderate. Independent of strength, the correlation is positive in 65% of correlations, making this the most favorable finding for NC in our evaluation. Unfortunately, over half of the positive correlations were weak and statistically insignificant at $p > 0.05$, undercutting the support these results might have contributed to the hypothesis.

5.3.3 Predicting Output Diversity Bias Caused by NC

In addition to the standard correlation analysis, we devised another experiment to explore another potential issue: does NC maximization biases test cases? This experiment consists of taking an original test suite of unperturbed images and calculating the class-specific NC using a range of ten different t thresholds between 0 and 0.9 in stepped increments of 0.1. For each threshold, we calculate the rank order of the classes with the highest NC activations. Finally, we average the ranks for each class to create a robust measure of the relationship

between class and NC. Following this approach, Table 5.5 shows the class rankings of a randomly generated test suite for a DNN trained to recognize 10 different classes between 0-9. The closer the rank is to 1, the more complete is that DNN's preference for the associated class. Here we can see that class 8 is the most preferred based on consistently having high NC across all thresholds and that 6 is the least preferred for the opposite reason.

Class	**8**	**5**	**2**	**0**	**7**	**3**	**4**	**9**	**1**	**6**
Average Rank	2.1	2.9	3.1	4.1	5.0	6.2	7.0	7.4	8.2	8.6

Table 5.5: MNIST Class and Avg Rank Prior to Attack

If the ranks of the classes are relatively close to the midpoint, e.g. the # of classes/2, then the DNN can be said to have little "preference" for any particular class. Otherwise, the DNN would be activating more strongly for particular classes. Any class preference would make using NC problematic for guiding the creation of test suites because the maximization is most easily satisfied by incorporating features of the preferred class(es) that already exhibit higher NC by default. Since the maximization of NC is the actual goal, we predict that inducing higher NC is likely to encourage the perturbations to incorporate characteristics of the preferred classes with higher robust NC and therefore skew the output diversity of the test suite - namely to class 8.

Our results indicate that it is feasible to anticipate which classes are likely to dominate the output distribution. This is true even when the underlying image set used to calculate the rankings is not the same as the ones used to build the test suite. This is probably the case because the "preference" for certain classes, manifested by higher robust NC coverage, is an inherent property of the DNN. Figure 5.7 shows the output diversity of a sampled experimental configuration consisting of 300 images (30 per class) performed on the Conv2DNet DNN traineed for the MNIST dataset. As we increase the diversity weight, we observe higher NC and lower output diversity. This test suite had a $NC_{t=0.2}$ of 0.3357 - about 40% higher than the original images - but an output diversity of 0.7394 because 45% of all predictions were now predicted for class 8.

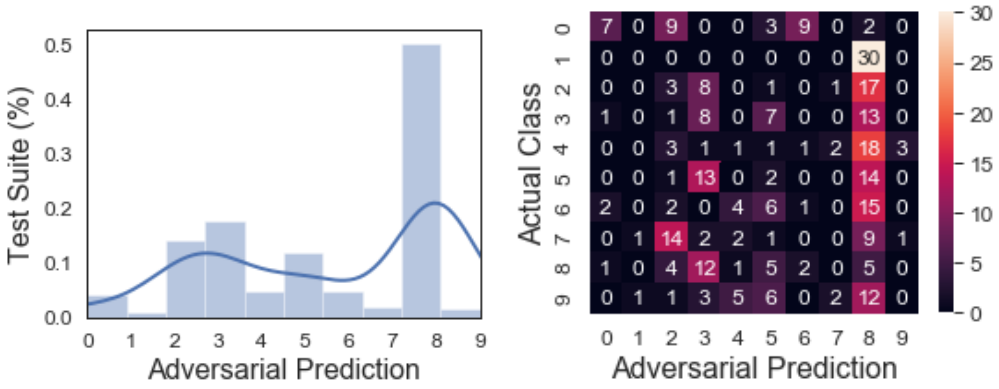

Figure 5.7: Class Distribution Histogram and Heatmap

Answer 3. Our findings reject the hypothesis that maximizing NC is both strongly and positively correlated with output diversity. Only 6% of all experimental results supported the hypothesis and a further 35% were actually negatively correlated. However, both theoretically and empirically, NC appears to reflect a preferred subset of classes and maximization of NC biases the test suite to predominantly incorporate features of this preferred subset.

CHAPTER 6

Discussion

6.1 How Meaningful is a Neuron?

The viability of NC as a good metric for testing DNNs is underpinned by the idea that each neuron independently extracts a specific feature of the input rather than collaborating with other neurons. However, previous research into DNN visualization techniques [34, 25, 24] have demonstrated that neuronal independence and local feature extraction do not accurately characterize the behavior of much of the DNN. Instead, the neurons in a layer interact with one another to pass information to subsequent layers, and thus each neuron's contribution to the input-output transformation depends on its specific context in a given layer of the DNN. Furthermore, the representations encoded by the neuron's weights frequently do not reflect the natural hierarchies of our semantic understandings. The probability that a neuron distinctly encodes a specific feature increases the deeper it is situated in the DNN, but many of the neurons represent an amalgam of very different abstract concepts, like those in Figure 6.1 [25]. This tendency raises serious doubts about whether neurons are the right semantic units for understanding DNNs, and therefore undermining any DNN testing metric based upon them.

Figure 6.1: Mixed concept neurons - cats, foxes, and cars

6.2 Does NC maximization make sense for DNNs?

Assuming the best case scenario of neurons independently encoding specific features, is maximization of NC actually a desirable goal? The idea of a neuron activating in response to some portion of the input suggests that some concept it's been trained to recognize is present in the input to some degree. If we think of each neuron as binary classifier checking for the presence or absence of some specific feature within the input, then it's activation is one possible state and it being dormant is the other. For n number of neurons in a DNN, there are 2^n possible activation patterns, many of which will correspond to realistic portions of the input space. On the other hand, NC merely targets *one* possible pattern of all neurons being activated rather than exploring the complex interplay of neural states.

As an extremely simplified example, consider two neurons in a DNN trained for autonomous driving. One detects the presence of vehicles and the other detects stop signs. Maximizing NC will encourage test suites to contain images of both a vehicle and a stop sign together - only 1 of the 4 possible permutations for this neuron pair - whereas other real world possibilities exist. Scenarios involving a sign and no vehicle, a vehicle and no sign, or the absence of both would go unexplored, much to the peril of drivers, pedestrians, and insurance companies.

Lastly, as we saw from our third hypothesis, the strength of neural activations represented by NC can be interpreted as a "preference" in the DNN towards certain input classes. Thus, the maximization of NC in the creation of new test suites implicitly encourages features of that class to be grafted into the other classes and ultimately skews the output diversity. It is therefore unlikely that any metric which implicitly possesses "preferred" classes will be suitable for testing DNNs.

CHAPTER 7

Future Work

Our next step is to evaluate other well regarded DNN testing metrics against the three test suite properties we've outlined - defect detection, realism, and output diversity. We believe that the metrics possessing the most promise are those that operate at a higher granularity than individual neuron activations. Thus, the top-k neuron pattern criteria proposed by Ma *et al.* [21] would be an especially good candidate for evaluation.

Concurrently, we plan to develop testing techniques along two different theoretical paths. The first is that of previously unexplored channel-level coverage criteria. Critical to the success of any such metric is that they permit for the exploration of a larger portions of the DNN's internal logic whilst still being constrained enough to focus on the most relevant features and avoid computational resource limitations. However, any kind of coverage-based criteria cannot reveal logic or behavior that was never encoded into the DNN, only measure to what extent it has been exercised by a test suite. For this reason, it is important to further research efforts into formal verification of DNNs. To this end, Huang *et al.* [10] and Katz *et al.* [16] provide exemplary work we hope to extend and improve upon.

CHAPTER 8

Conclusion

In this paper, we devised a novel NC-promoting regularizer and formalized three key properties required for successful DL test suites - defect detection, naturalness, and output diversity. We also studied the relationship between these properties and the most widely cited DL testing metric: neuron coverage. From our evaluation of 1,623 experimental configurations involving six DNNs, two datasets, and two adversarial attack algorithms, we conclude that neuron coverage is *not* a meaningful measure for testing neural networks. On average, only about 5% of correlations supported the three hypotheses relating NC and our key test suite properties and the vast majority of outcomes were either uncorrelated or negatively correlated.

The future applicability, safety, and success of deep learning techniques require - at minimum - a compelling demonstration of their behavior and robustness. The idea of neuron coverage is an important, early step in making this possible, but ultimately, we continue to need more sophisticated metrics capable of capturing the complex interplay of larger neural components and distilling their behavior into a meaningful signal of their reliability.

REFERENCES

[1] Barratt, S. *Inception Score for GANs in Pytorch*, 2018 (accessed August 8, 2019).

[2] Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. End to end learning for self-driving cars. *CoRR abs/1604.07316* (2016).

[3] Brosgol, B. Do-178c: The next avionics safety standard. In *Proceedings of the 2011 ACM Annual International Conference on Special Interest Group on the Ada Programming Language* (New York, NY, USA, 2011), SIGAda '11, ACM, pp. 5–6.

[4] Carlini, N., and Wagner, D. A. Towards evaluating the robustness of neural networks. *2017 IEEE Symposium on Security and Privacy (SP)* (2016), 39–57.

[5] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09* (2009).

[6] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2* (Cambridge, MA, USA, 2014), NIPS'14, MIT Press, pp. 2672–2680.

[7] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), 770–778.

[8] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS* (2017).

[9] Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 2261–2269.

[10] Huang, X., Kwiatkowska, M., Wang, S., and Wu, M. Safety verification of deep neural networks. *CoRR abs/1610.06940* (2016).

[11] Idelbayev, Y. *Proper ResNet Implementation for CIFAR10/CIFAR100 in pytorch*, 2018 (accessed August 8, 2019).

[12] Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. Adversarial examples are not bugs, they are features. *ArXiv abs/1905.02175* (2019).

[13] Inozemtseva, L., and Holmes, R. Coverage is not strongly correlated with test suite effectiveness. In *Proceedings of the 36th International Conference on Software Engineering* (New York, NY, USA, 2014), ICSE 2014, ACM, pp. 435–445.

[14] Road vehicles — Functional safety — Part 6: Product development at the software level. Standard, International Organization for Standardization, Geneva, CH, 2011.

[15] Julian, K., Kochenderfer, M., and Owen, M. Deep neural network compression for aircraft collision avoidance systems. *Journal of Guidance, Control, and Dynamics 42* (11 2018), 1–11.

[16] Katz, G., Barrett, C. W., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient SMT solver for verifying deep neural networks. *CoRR abs/1702.01135* (2017).

[17] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *CoRR abs/1412.6980* (2014).

[18] Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 (canadian institute for advanced research).

[19] Kullback, S., and Leibler, R. A. On information and sufficiency. *Ann. Math. Statist. 22*, 1 (03 1951), 79–86.

[20] LeCun, Y., and Cortes, C. MNIST handwritten digit database.

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

[22] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *ArXiv abs/1706.06083* (2017).

[23] mseitzer. *Frèchet Inception Distance (FID score) in PyTorch*, 2018 (accessed August 9, 2019).

[24] Nguyen, A. M., Yosinski, J., and Clune, J. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *ArXiv abs/1602.03616* (2016).

[25] Olah, C., Mordvintsev, A., and Schubert, L. Feature visualization. *Distill* (2017). https://distill.pub/2017/feature-visualization.

[26] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop* (2017).

[27] Pei, K., Cao, Y., Yang, J., and Jana, S. Deepxplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles* (New York, NY, USA, 2017), SOSP '17, ACM, pp. 1–18.

[28] Phan, H. *PyTorch models trained on CIFAR-10 dataset*, 2019 (accessed October 4, 2019).

[29] Ratner, B. The correlation coefficient: Its values range between +1/1, or do they? *Journal of Targeting, Measurement and Analysis for Marketing 17*, 2 (Jun 2009), 139–142.

[30] Salimans, T., Goodfellow, I. J., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. *ArXiv abs/1606.03498* (2016).

[31] Shannon, C. E. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev. 5*, 1 (Jan. 2001), 3–55.

[32] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), 2818–2826.

[33] Tian, Y., Pei, K., Jana, S., and Ray, B. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th International Conference on Software Engineering* (New York, NY, USA, 2018), ICSE '18, ACM, pp. 303–314.

[34] Yosinski, J., Clune, J., Nguyen, A. M., Fuchs, T. J., and Lipson, H. Understanding neural networks through deep visualization. *ArXiv abs/1506.06579* (2015).

[35] Zhang, M., Zhang, Y., Zhang, L., Liu, C., and Khurshid, S. 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* (New York, NY, USA, 2018), ASE 2018, ACM, pp. 132–142.