2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)

Structural Test Coverage Criteria for Deep Neural Networks

Youcheng Sun*, Xiaowei Huang†, Daniel Kroening*, James Sharp‡, Matthew Hill‡, Rob Ashmore‡

*University of Oxford, UK
{youcheng.sun, kroening}@cs.ox.ac.uk
†University of Liverpool, UK
xiaowei.huang@liverpool.ac.uk
‡Defence Science and Technology Laboratory (Dstl)
{jsharp1, mhill2, rdashmore}@dstl.gov.uk

***Abstract*—Deep Neural Networks (DNNs) have a wide range of applications, and software employing them must be thoroughly tested, especially in safety-critical domains. However, traditional software test coverage metrics cannot be applied directly to DNNs. In this paper, inspired by the MC/DC coverage criterion, we propose four novel test criteria that are tailored to structural features of DNNs and their semantics. We validate the criteria by demonstrating that the generated test inputs, guided by our coverage criteria, are able to capture undesirable behaviours in DNNs. Test cases are generated using both a symbolic approach and a gradient-based heuristic. Our experiments are conducted on state-of-the-art DNNs, obtained using the MNIST and ImageNet datasets.**

I. Introduction

AI systems that use Deep Neural Networks, or DNNs, are typically implemented in software. However, (white-box) testing for traditional software cannot be directly applied to DNNs. In particular, the flow of control in DNNs is not sufficient to represent the knowledge that is learned during the training phase, and thus it is not obvious how to define structural coverage criteria for DNNs [1]. Additionally, DNNs exhibit different "bugs" from traditional software. Notably, *adversarial examples* [2], whereby two apparently indistinguishable (to a human) inputs cause contradictory decisions, are one of the most prominent safety concerns for DNNs.

Technically, DNNs contain not only an architecture, which bears some similarity to traditional software, but also a large set of parameters, whose values are learned during a training procedure. Any approach for testing DNNs needs to consider the unique properties of DNNs, such as: the syntactic connections between neurons in adjacent layers, where neurons in a given layer interact with each other and then pass information to higher layers; ReLU (Rectified Linear Unit) activation functions; and the semantic relationship between layers.

In this paper, we propose a family of four test coverage criteria, inspired by the MC/DC (modified condition/decision coverage) test criterion from traditional software testing, that examine the unique properties of DNNs, mentioned above. The test coverage criteria can be applied to identify adversarial behaviours in DNNs and to evaluate the quality of test suites.

II. Deep Neural Networks (DNNs)

A (feedforward and deep) neural network, or DNN, is a tuple $\mathcal{N} = (L, T, \Phi)$, where $L = \{L_k \mid k \in \{1..K\}\}$ is a set of layers, $T \subseteq L \times L$ is a set of connections between layers, and $\Phi = \{\varphi_k \mid k \in \{2..K\}\}$ is a set of functions, one for each non-input layer. In a DNN, L_1 is the *input* layer, L_K is the *output* layer, and layers other than input and output layers are called *hidden layers*. Each layer L_k consists of s_k *neurons*. The l-th neuron of layer k is denoted by $n_{k,l}$. Each neuron $n_{k,l}$ for $2 \leq k \leq K$ and $1 \leq l \leq s_k$ is associated with two variables $u_{k,l}$ and $v_{k,l}$, which record its values before and after an activation function, respectively. ReLU is by far the most popular activation function for DNNs, according to which the *activation value* of each neuron (within a hidden layer is defined as $v_{k,l} = ReLU(u_{k,l}) = \max\{v_{k,l}, 0\}$. Given one particular input x, the values of the variables $u_{k,l}$ and $v_{k,l}$ are fixed and denoted $u_{k,l}[x]$ and $v_{k,l}[x]$, respectively.

III. Adequacy Criteria for Testing DNNs

The core idea of MC/DC is that if a choice can be made, all the possible factors (conditions) that contribute to that choice (decision) must be tested. For traditional software, both conditions and the decision are usually Boolean variables or Boolean expressions.

Our instantiation of the concepts "decision" and "condition" for DNNs is inspired by the fact that the information represented by a neuron in the next layer can be seen as a *summary* (implemented by the layer function, the weights, and the bias) of the information in the current layer. For example, neurons in deeper layers represent more complex attributes of the input [3]. Unlike Boolean variables or expressions, where it is trivial to define change, i.e., true → false or false → true, in DNNs there are different ways of defining how a decision is *affected* by the changes in the conditions.

We let $\psi_{k,l}$ denote an arbitrary feaure at layer k, and we extend the notations $u_{k,l}[x]$ and $v_{k,l}[x]$ for a neuron $n_{k,l}$ to a feature $\psi_{k,l}$, and write $\psi_{k,l}[x]$ and $\phi_{k,l}[x]$ for the vectors before and after ReLU, respectively. First, the change observed on a feature can be either a sign change or a value change.

2574-1934/19/$31.00 ©2019 IEEE
DOI 10.1109/ICSE-Companion.2019.00134

	hidden layers	M_{SS}	AE_{SS}	M_{VS^g}	AE_{VS^g}	M_{SV^g}	AE_{SV^g}	$M_{VV^{g_1,g_2}}$	$AE_{VV^{g_1,g_2}}$
$\mathcal{N}_1$	67x22x63	99.7%	18.9%	100%	15.8%	100%	6.7%	100%	21.1%
$\mathcal{N}_2$	59x94x56x45	98.5%	9.5%	100%	6.8%	99.9%	3.7%	100%	11.2%
$\mathcal{N}_3$	72x61x70x77	99.4%	7.1%	100%	5.0%	99.9%	3.7%	98.6%	11.0%
$\mathcal{N}_4$	65x99x87x23x31	98.4%	7.1%	100%	7.2%	99.8%	3.7%	98.4%	11.2%
$\mathcal{N}_5$	49x61x90x21x48	89.1%	11.4%	99.1%	9.6%	99.4%	4.9%	98.7%	9.1%
$\mathcal{N}_6$	97x83x32	100.0%	9.4%	100%	5.6%	100%	3.7%	100%	8.0%
$\mathcal{N}_7$	33x95x67x43x76	86.9%	8.8%	100%	7.2%	99.2%	3.8%	96%	12.0%
$\mathcal{N}_8$	78x62x73x47	99.8%	8.4%	100%	9.4%	100%	4.0%	100%	7.3%
$\mathcal{N}_9$	87x33x62	100.0%	12.0%	100%	10.5%	100%	5.0%	100%	6.7%
$\mathcal{N}_{10}$	76x55x74x98x75	86.7%	5.8%	100%	6.1%	98.3%	2.4%	93.9%	4.5%

TABLE I: Results on 10 DNNs with different covering methods

Definition 1 (Sign Change): Given a feature $\psi_{k,l}$ and two test cases x_1 and x_2, the sign change of $\psi_{k,l}$ is exploited by x_1 and x_2, denoted as $sc(\psi_{k,l}, x_1, x_2)$, if

- $sign(n_{k,j}, x_1) \neq sign(n_{k,j}, x_2)$ for all $n_{k,j} \in \psi_{k,l}$

where $sign(n_{k,j}, x_1)$ and $sign(n_{k,j}, x_2)$ denote the sign of neuron $n_{k,j}$ on inputs x_1 and x_2, respectively. We define $sign(n_{k,j}, x) = 1$, if $u_{k,j}[x] = v_{k,j}[x]$, and $sign(n_{k,j}, x) = 0$, otherwise. Moreover, we write $nsc(\psi_{k,l}, x_1, x_2)$ if

- $sign(n_{k,j}, x_1) = sign(n_{k,j}, x_2)$ for all $n_{k,j} \in \psi_{k,l}$.

Definition 2 (Value Change): Given a feature $\psi_{k,l}$, two test cases x_1 and x_2, and a value function g, the value change of $\psi_{k,l}$ is exploited by x_1 and x_2 with respect to g, denoted as $vc(g, \psi_{k,l}, x_1, x_2)$, if

- $g(\psi_{k,l}, x_1, x_2)$=true

such that the function $g(\psi_{k,l}, x_1, x_2)$ can, e.g., express the distance between two vectors $\psi_{k,l}[x_1]$ and $\psi_{k,l}[x_2]$ by norm-based distances $||\psi_{k,l}[x_1] - \psi_{k,l}[x_2]||_p \leq d$, for a real number d and a distance measure L^p.

Subsequently, we develop a family of four methods to cover these causal changes in a DNN.

Definition 3 (Sign-Sign Coverage, or SS Coverage): A feature pair $\alpha = (\psi_{k,i}, \psi_{k+1,j})$ is SS-covered by two test cases x_1, x_2, denoted as $SS(\alpha, x_1, x_2)$, if the following conditions are satisfied:

- $sc(\psi_{k,i}, x_1, x_2)$ and $nsc(P_k \setminus \psi_{k,i}, x_1, x_2)$;
- $sc(\psi_{k+1,j}, x_1, x_2)$.

where P_k is the set of neurons in layer k.

The SS coverage provides evidence that the change of a condition feature $\psi_{k,i}$'s sign independently affects the sign of the decision feature $\psi_{k+1,j}$ in the next layer. Intuitively, the first condition says that the sign change of feature $\psi_{k,i}$ is exploited using x_1 and x_2, without changing the signs of other non-overlapping features. The second says that, the sign change of feature $\psi_{k+1,j}$ is exploited using x_1 and x_2.

To complement the SS coverage, we also define three other methods to cover the causal relationship between condition and decision features: namely, Sign-Value (SV) Coverage, Value-Value (VV) Coverage and Value-Sign (VS) Coverage.

IV. Experiments

These MC/DC-inspired coverage metrics are used to guide the test case generation for DNNs trained using the MNIST dataset, and the state-of-the-art VGG16 model for the ImageNet dataset. Two approaches are utilised to generate test cases: the concolic testing in [4], which combines symbolic execution and concrete testing, and is suitable for small scale DNNs (e.g., trained on MNIST); a gradient descent (GD) based search heuristic is also designed and applied to VGG16. Selected representative results are shown in Table I and Figure 1.

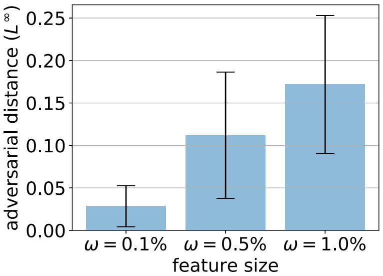

Fig. 1: Adversarial distance with different feature sizes

In Table I, we apply the covering method defined in Section III to 10 randomly generated DNNs for MNIST. We measure the level of coverage, M_f, for each covering method $f \in \{SS, SV, VV, VS\}$, and the portion of adversarial examples AE_f in the generated test suite. The test results are promising, with the covering methods identifying a significant portion of adversarial examples.

Applying SS coverage on 2,000 randomly sampled feature pairs with size parameter $\omega \in \{0.1\%, 0.5\%, 1.0\%\}$ such that a feature $\psi_{k,i}$ is required to have its size $\leq \omega \cdot s_k$, we report the adversarial examples' average distance and standard derivation in Figure 1. The results confirm that there is a relation between the feature pairs and the input perturbation. Note, among the generated adversarial examples, a more fine-grained feature is able to capture smaller perturbations than a coarse one.

Acknowledgements: This document is an overview of UK MOD (part) sponsored research and is released for informational purposes only. The contents of this document should not be interpreted as representing the views of the UK MOD, nor should it be assumed that they reflect any current or future UK MOD policy. The information contained in this document cannot supersede any statutory or contractual requirements or liabilities and is offered without prejudice or commitment.

References

[1] R. Ashmore and E. Lennon, "Progress towards the assurance of non-traditional software," in *Safety-critical Systems Symposium*, 2017.

[2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *In ICLR*. Citeseer, 2014.

[3] J. Yosinski, J. Clune, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," in *In Deep Learning Workshop, International Conference on Machine Learning (ICML)*, 2015.

[4] Y. Sun, M. Wu, W. Ruan, X. Huang, M. Kwiatkowska, and D. Kroening, "Concolic testing for deep neural networks," in *Automated Software Engineering (ASE), 33rd IEEE/ACM International Conference on*, 2018.