CC: Causality-Aware Coverage Criterion for Deep Neural Networks

Zhenlan Ji, Pingchuan Ma*, Yuanyuan Yuan*, and Shuai Wang
The Hong Kong University of Science and Technology, Hong Kong, China
{zjiae, pmaab, yyuanaq, shuaiw}@cse.ust.hk

Abstract—**Deep neural network (DNN) testing approaches have grown fast in recent years to test the correctness and robustness of DNNs. In particular, DNN coverage criteria are frequently used to evaluate the quality of a test suite, and a number of coverage criteria based on neuron-wise, layer-wise, and path-/trace-wise coverage patterns have been published to date. However, we see that existing criteria are insufficient to represent how one neuron would influence subsequent neurons; hence, we lack a concept of how neurons, when functioning as causes and effects, might jointly make a DNN prediction.**

Given recent advances in interpreting DNN internals using causal inference, we present the first causality-aware DNN coverage criterion, which evaluates a test suite by quantifying the extent to which the suite provides new causal relations for testing DNNs. Performing standard causal inference on DNNs presents both theoretical and practical hurdles. We introduce CC (causal coverage), a practical and efficient coverage criterion that integrates a set of optimizations using DNN domain-specific knowledge. We illustrate the efficacy of CC using diverse, real-world inputs and adversarial inputs, such as adversarial examples (AEs) and backdoor inputs. We demonstrate that CC outperforms previous DNN criteria under various settings with moderate cost.

I. Introduction

Recent research has combined software testing approaches to successfully evaluate DNNs without human annotations, such as by mutating test inputs with always-hold relations (e.g. metamorphic relations) [41, 45, 56]. Such practice can effectively augment evaluation datasets and benchmark DNNs. Nevertheless, due to its inherent non-linearity, fundamental understanding of a DNN's mechanism has remained largely unexplored. Moreover, it has been discovered that the decisions made by a DNN are sensitive to input biases [4, 9, 52]. Consequently, DNN evaluation is heavily dependent on the representativeness of the test suite. Thus, testing criteria are essential to assess the quality of a test suite and to ensure that input mutations can exhaustively explore DNN behaviors.

A number of coverage criteria have been proposed for DNNs, which use neuron outputs (referred to as "neuron activations" in most literatures [31]) as the basic computational units. Upon neuron activations, current criteria model DNN internals mainly from three perspectives: (i) the coverage status of each individual neuron, (ii) the coverage status of each DNN layer, and (iii) a trace/path of covered DNN neurons. Despite the adoption of existing criteria in DNN testing, we find that prior criteria frequently lack modeling of how one neuron influences subsequent neurons, thus lacking an understanding of how neurons, when acting as causes and effects, can jointly determine DNN predictions. Real-world data (e.g., images) typically show distinct concepts to facilitate human cognition, e.g., a dog can be recognized by first viewing its fur, then noticing its era shape and size. It has also been observed [37] that adversarially constructed inputs [11, 12] might influence DNN predictions due to the unique and strong causality buried within such inputs.

An emerging, promising trend is to leverage *causal inference* [32] to facilitate interpreting DNN internals [5, 37]. A DNN model is converted to a causal graph, which is then used to interpret DNN predictions and internals from the perspective of causal relations. In tandem with recent efforts in DNN repairing [37], this work targets a vital demand to design a causality-aware coverage criterion, thus delivering a comprehensive, causality-aware assessment of DNN test suites. To this end, we establish a causal view on how DNN makes predictions, and formally prove that a DNN can be equivalently expressed by a structural causal model (SCM) [32]. Then, given a ground truth causal graph $\mathcal{G}$ derived from the DNN, we establish a criterion, CC (causal coverage), to evaluate a test suite by quantifying the extent to which the causal effects conveyed in the test suite can help retain edges of $\hat{\mathcal{G}}$. A good test suite should preserve as many edges as possible, indicating that it provides a thorough set of causal relations to test DNNs.

Despite the promising potential of CC, performing causal discovery in our research involves theoretical and practical challenges. First, causal discovery relies on several canonical assumptions about data distribution [32], which are often violated by causal graphs entailed by DNNs. Second, causal discovery is known to be slow with exponential complexity. Given that modern DNNs often contain tens of thousands of neurons, such scale exceeds the capability of standard causal discovery methods [36]. Furthermore, in DNN testing, coverage is typically computed *incrementally*, whereas existing causal discovery algorithms are *not* built for this purpose. CC incorporates various design principles and optimizations to overcome the aforementioned hurdles and fulfill the requirements. We also provide CC_m, a minimal version of CC, to selectively capture causal relations of a subset of DNN layers; we prove that CC_m is under theoretical correctness guarantee and shows that it offers significantly lower costs.

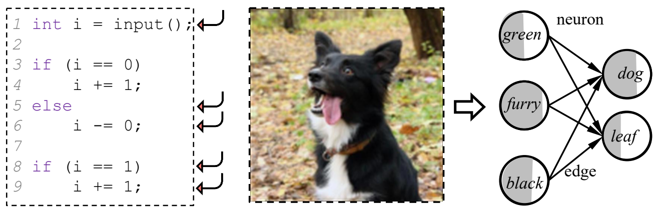
Our evaluation is performed on three DNNs, LeNet5 [21], VGG16 [35], and ResNet50 [14], for image classification

* Corresponding authors

task on three popular datasets, MNIST [7], CIFAR10 [20] and ImageNet [6]. We compare CC with 11 de facto DNN coverage criteria. We show that CC is substantially correlated with the varying degrees of causal relations conveyed in a test suite, outperforming all prior coverage criteria to a notable extent. This indicates that CC features a preeminent capacity to reflect test adequacy. Moreover, CC uncovers subtle causal relations concealed in adversarially generated inputs, including adversarial examples (AEs) and backdoor inputs. This illustrates CC's promising applicability in adversarial settings, for which existing coverage criteria are insufficient. In addition to quantitative results, we present a qualitative analysis illustrating how CC can be used to comprehend DNN decisions under various inputs. In sum, this research makes the following contributions:

- We give a causal view on how DNN make predictions and prove that common DNN models can be expressed equivalently using causality analysis notations, particularly SCM. CC is the first DNN coverage criterion based on causal relations entailed by DNN.
- CC incorporates various design principles and optimizations to enable causal-aware coverage computation over large DNNs and real-world test inputs. CC$_m$, a minimal version of CC, is proposed to complement CC with correctness guarantee and much lower cost.
- We show that CC can better assess the quality of test suites by comparing it with 11 de facto DNN coverage criteria. CC is correctly correlated with the causal relations conveyed by normal and adversarial inputs.

Artifact availability. We have released all the findings and CC to facilitate further research at [2].



```
int i = input();

if (i == 0)
    i += 1;
else
    i -= 0;

if (i == 1)
    i += 1;
```

(a) covered statements (b) DNN execution ("furry" + "dog" are more activated)

Figure 1. Coverage metrics of conventional programs vs. DNN.

II. Preliminary

A. Deep Neural Networks

Unlike conventional software whose internals can be characterized by observing the covered statements or branches, control/data flows in DNNs are typically constant given different inputs, as in Fig. 1. It is thus infeasible to formulate DNN coverage criteria by mimicking conventional software. The minimal computing unit of a DNN is generally deemed as one neuron, which is presumed to individually encode certain features (e.g., animal furry) [22, 31]. Due to the non-linear nature of DNNs, it remains a challenge to rigorously formulate the internals of DNNs. Nevertheless, neuron outputs (often referred to as "neuron activation" [31]) differ with inputs. It is acknowledged that DNN behaviors are correlated with neuron outputs to a reasonable extent.

Considering Fig. 1(b), when processing a dog image, certain neurons signify its color (e.g., neurons in the first layer), whereas some other neurons, often in deeper layers, signify higher-level concepts in this image. For example, the neuron in the top right corner signifies the "*dog*". This reflects how neuron outputs can characterize DNN behaviors, and for the input in Fig. 1(b), the (largely simplified) DNN relies on "*furry*" (and "*black*" color) to recognize this "*dog*". In contrast, the DNN may also focus on the leaf in background to incorrectly make decisions (an erroneous DNN state). Therefore, existing coverage criteria are often designed to measure DNN states by observing neuron outputs.

B. DNN Coverage

Existing DNN coverage criteria are built by taking neuron outputs as the basic computing units. We categorize them according to the graphical DNN representations they extract.

Neuron-Wise. The majority of existing criteria [22, 31] consider each neuron individually for calculating coverage. neuron coverage (NC; Fig. 2(a)) [31], regards a neuron as activated if its output is greater than a threshold T and the coverage is counted as the ratio of activated neurons. Later, several fine-grained criteria are proposed to monitor where the neuron outputs lie in. In particular, k-multisection neuron coverage (KMNC; Fig. 2(b)) [22] first scopes the output range of each neuron using DNN training data and splits the output range into k sections. It then calculates the coverage as the ratio of covered sections. Further, as a complement of KMNC, neuron boundary coverage (NBC; Fig. 2(c)) [22] is designed to observe neurons whose outputs fall outside the output range (decided as in KMNC). The coverage is calculated as the ratio of neurons having out-of-range outputs. Similarly, strong neuron activation coverage (SNAC; Fig. 2(d)) is designed to specifically focus on neurons whose outputs are greater than the upper bound of the pre-decided output range (i.e., hyperactive neurons [22]). It computes the coverage as the ratio of hyperactive neurons.

Layer-Wise. Since a DNN is a composition of layers, some criteria, from another angle, calculates coverage from the layer perspective. Top-k neuron coverage (TKNC; Fig. 2(e)) [22] considers neurons in the same layer as one group and regards one neuron as activated if it has the top-k output within its group. The coverage is denoted as the ratios of activated neurons. Based on TKNC, Top-k neuron patterns (TKNP; Fig. 2(e)) counts the patterns of activated neurons as the coverage. In addition, TensorFuzz coverage (TFC; Fig. 2(f)) views neuron outputs from the same layer as one high-dimensional variable and gathers the high-dimensional variables (w.r.t. different inputs) as clusters. The derived coverage is counted as the number of clusters. NeuraL coverage (NLC) [50], from a distribution perspective, defines coverage over the outputs of all neurons in the same layer. It considers the neuron correlation in one layer (i.e., how one neuron affects other neurons in the same layer), which, to some extent, can reflect the causality in a layer-wise manner. In contrast, CC views the whole DNN as a causal model (see Sec. II-C) and the causality is established during the forward propagation in the DNN.

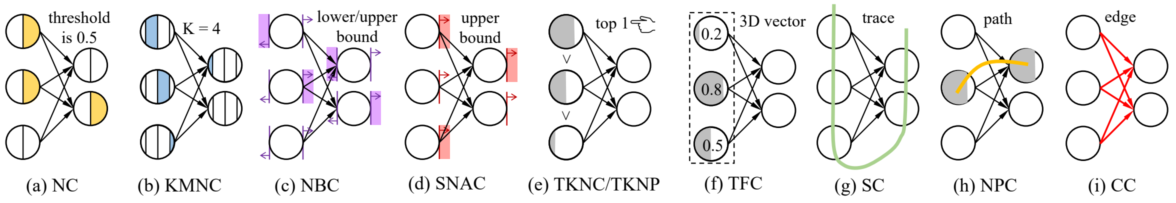


Figure 2. High-level illustration of previous coverage criteria and CC.

Trace-/Path-Wise. In addition to these layer-wise criteria, some recent works measure DNN coverage on neuron traces/-paths [17, 18, 38, 46]. For example, surprise coverage (SC; Fig. 2(g)) first constructs a trace using all neurons and records the trace outputs (one trace output is a list of all neuron outputs) w.r.t all training data. Then, given a test inputs, SC measures the distance between its trace output and all training trace outputs — the coverage is calculated as the ratio of covered distance score sections (which requires first splitting the range of distance scores as sections). Different distance metrics are considered to form three SC variants, namely, likelihood SC (LSC) [17], distance-ratio SC (DSC) [17], and Mahalanobis distance SC (MDSC) [18]. Similarly, neuron path coverage (NPC; Fig. 2(h)) uses white-box explainable AI (XAI) techniques to identify critical neurons when the DNN is making a prediction for an input. These critical neurons, from the first to the last layer, consist of a decision path. The coverage is therefore defined on the paths corresponding to different test inputs, either from the structure (i.e., the localization of critical neurons in the DNN) or the neuron output perspective. Unlike previous gray-box criteria, NPC requires a white-box access (requiring DNN gradients) to DNNs. CC, as introduced in the paper, captures how neurons influence subsequent neurons, denoting their causal relations. We also note that MCDC [38] focuses on the relations between adjacent neurons, but it appears mainly applicable to ReLU due to its dependence on the "sign change" of neuron outputs. This "sign change" can be seen as a special case of the edge falsification method (see Sec. IV-A). In short, CC are established over neuron edges, where we perform statistical independence tests to decide the causal relations derived from DNN edges.

C. Causal Inference

We now formalize basic concepts in causal inference. Structural causal model (SCM) denotes a simple yet highly expressive notion of hypothesized causal relations between variables [32]. SCM is sufficient to depict causal relations in many real-life scenarios like social science and biology [13]. In a general multivariate system, an SCM is defined as follows.

Definition 1 (SCM [32])**.** *An SCM $\mathcal{C} := (\boldsymbol{S}, P_{\mathrm{N}})$ consists a collection $\boldsymbol{S}$ of d structural assignments*

$$X_j := f_j(\mathrm{PA}_j, N_j), j = 1, \cdots, d \tag{1}$$

*where $\mathrm{PA}_j \subseteq \{X_1, \cdots, X_d\} \setminus \{X_j\}$ are called **parents** of X_j; and a joint probability distribution $P_{\mathrm{N}} = P_{N_1,\cdots,N_d}$ over the noise variables, which we require to be jointly independent; that is, P_{N} is a product distribution.*

A graph $\mathcal{G}$ can be obtained over the above notation of SCM, in the sense that we create a node for each variable X_j and adding a directed edge between X_j and each of its parents PA_j. It is evident that $\mathcal{G}$ is acyclic.

Causal Graph. The graph $\mathcal{G}$ comprises causal relations between variables in its SCM, and following the convention, $\mathcal{G}$ is referred to as the *causal graph*.

An SCM characterizes a joint probability distribution $P_{\boldsymbol{X}}$ over $\{X_1, \cdots, X_d\}$. Moreover, an SCM also entails a data-generating process. That is, every sample from $P_{\boldsymbol{X}}$ can be equivalently generated by the data-generating process, where the realization of each variable is concretized in topological order of the causal graph $\mathcal{G}$ derived from the SCM. We define the realization of an SCM as follows:

Definition 2 (Realization of SCM)**.** *Let $\mathcal{C} = (\boldsymbol{S}, P_{\mathrm{N}})$ be an SCM over d random variables $\{X_1, \cdots, X_d\}$. $N_1, \cdots, N_d \sim P_{\mathrm{N}}$ be a sample of P_{N}.*[1] *A realization of $\mathcal{C}$ is defined as $\boldsymbol{x} = (x_1, \cdots, x_d)$. The values of $\boldsymbol{x}$ are assigned iteratively from x_1 to x_d where $x_i = f_j(\mathrm{PA}_j, N_j)$.*

Causal Discovery from Observational Data. In real-world scenarios, the ground truth SCM of the underlying systems is usually unknown. Instead, we usually have a collection of observations to this system (i.e., SCM realizations). Typically, establishing the causal graph $\mathcal{G}$ from these observations is one fundamental step in comprehending the causal relations comprised by the underlying systems [10, 32, 36]. Such process is called *causal discovery*. Holistically, there are four popular paradigms for learning causal graphs from observational data, namely constraint-based [36], score-based [43], gradient-based [57] and supervision-based [23]. Below, we briefly introduce the constraint-based method.

Given $\mathcal{G}$ as a DAG, constraint-based causal discovery often has two steps in a mechanical manner, where the skeleton of $\mathcal{G}$ (i.e., an undirected graph) is first discovered, and edge directions are further assigned to undirected edges. Skeleton discovery often i) starts with a complete graph where each variable of the observed system has its node, and all nodes are fully connected, ii) enumerates all pairs of nodes and for each pair, conduct statistical tests with the given dataset to determine conditional independence, iii) removes their edge when independence is identified, and iv) repeats step 2–3 until no edge can be deleted. We clarify that the causal discovery process launched by CC is extended from the standard skeleton discovery process, and edge orientation is not needed; see details in Sec. IV.

[1]Without loss of generality, let $\{X_1, \cdots, X_d\}$ is in the topological order of the causal graph.

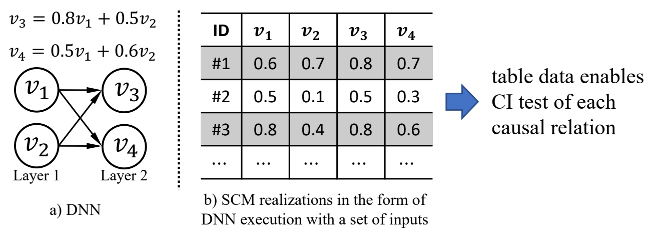
III. Research Motivation

A. A Causality View on DNN Execution

Sun et al. (ICSE'22) [37] have shown that FFNN (feed-forward neural networks) and CNN (convolutional neural networks) can be equivalently represented by an SCM.[2] This section extends their statement in a more general form. We show that any neural networks that can be characterized by a DAG [1] can also be represented by an SCM, including common DNNs like FFNN, CNN and RNN (recurrent neural networks). Then, we present an important statement that DNN execution is identical to the data generating process of an SCM.

Proposition 1. *Any DAG-compatible neural networks can be equivalently represented by an SCM.*

Proof Sketch. According to the definition in [1], let a DAG-compatible neural network $\mathcal{N}$ being represented by a DAG $G = (V, E)$, where V is the set of nodes (i.e., neurons in neural networks) on N and E is the set of edges. For each non-input[3] node $v \in N$, we can constitute a function $f_v : \mathbb{R}^{n_v} \to \mathbb{R}$ where n_v is the number of edges whose destination is v. Thus, we establish the corresponding SCM $\mathcal{M}$, where each non-input node $v \in \mathcal{N}$ is purely decided by its parents such that $v = f_v(\text{PA}_v)$, $N_i = 0$ in Eqn. 1 and PA_v is derived by taking the origins of edges targeting v. For each input node $x_i \in \mathcal{N}$ denoting the i-th dimension of the input $\boldsymbol{x}$, x_i is deterministically decided by the value of $\boldsymbol{x}$. And, the input itself $\boldsymbol{x} = f_{\boldsymbol{x}}(\emptyset, N_{\boldsymbol{X}}) = N_{\boldsymbol{X}}$ is sampled from a multivariate distribution $N_{\boldsymbol{X}} \sim P_{\boldsymbol{X}}$ where $\emptyset$ denotes that there are no parents for the input. $\square$


$v_3 = 0.8v_1 + 0.5v_2$
$v_4 = 0.5v_1 + 0.6v_2$
v_1
v_2
v_3
v_4
Layer 1
Layer 2
a) DNN
b) SCM realizations in the form of DNN execution with a set of inputs
table data enables CI test of each causal relation

ID	v_1	v_2	v_3	v_4
#1	0.6	0.7	0.8	0.7
#2	0.5	0.1	0.5	0.3
#3	0.8	0.4	0.8	0.6
...	...	...	...	...

Figure 3. Relations of DNN executions and SCM realizations.

DNN $\mathcal{N}$ depicts a deterministic decision process that computes $\boldsymbol{y} = \mathcal{N}(\boldsymbol{x})$. Therefore, when $\mathcal{N}$ is represented by SCM $\mathcal{M}$ according to the above construction, $\boldsymbol{y}$ can be equivalently computed by initializing the input with $\boldsymbol{x}$ and then realizing the SCM. That is,

> The execution (prediction) of DNN on a given input $\boldsymbol{x}$ is equivalent to a realization of its corresponding SCM with the noise $N_{\boldsymbol{X}}$ being assigned as the input $\boldsymbol{x}$.

As in Fig. 3, given a DNN, its SCM realization is conducted by assigning the input $N_{\boldsymbol{X}}$ with particular values (i.e., assign values to v_1, v_2 in Fig. 3(a)) and collecting the neuron values (i.e., $v_{1\sim4}$). Thus, each row in the table of Fig. 3(b) encodes the outcome of a realization to the SCM entailed by the DNN. Later, this table will be used to facilitate conditional independence (CI) test to decide the existence of causality among nodes.

In the rest of the paper, we denote an execution of the DNN $\mathcal{N}$ as a realization of its corresponding SCM $\mathcal{M}$ that encodes the values of neurons in $\mathcal{N}$. Each execution/realization corresponds to one row in Fig. 3(b).

Illustrative Example.[4] Recall Fig. 1, where we illustrate how neurons are jointly involved in DNN decisions. Overall, when DNN is making prediction over an input, it is hypnotized that certain neurons influence each other, gradually establishing a causal chain to constitute the DNN predictions. In Fig. 1, the input image successfully triggers two causal relations "furry $\to$ dog" as well as "black $\to$ dog". Moreover, it is evident that other inputs may trigger distinct causal relations (e.g., "green $\to$ leaf") encoded in the DNN, in case the inputs are 1) *reasonably high quality* (i.e., not random noise) and 2) *sufficiently different* from the dog image. We view this observation illustrates a promising opportunity to formulate causal relations as DNN coverage criteria.

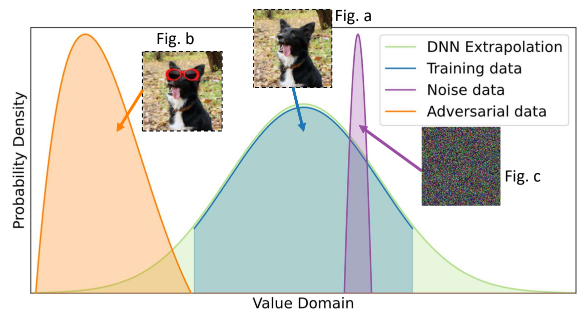


Figure 4. Distribution of different inputs.

Distinction of Different Input Types. Further to above discussions, we hypothesize to reveal subtle difference between normal DNN inputs, adversarial-generated data, and random noise through the lens of causality. Fig. 4 presents the schematic difference of neuron value distributions at deep DNN layers. First, normal inputs (i.e., training data) forms a distribution with finite samples (blue region in the middle). By training with normal inputs, DNN extrapolates the distribution of normal inputs into the entire input domain (green region). In normal inputs, DNN is expected to learn correct decision mechanisms in the form of multiple key causal relations in the SCM. In contrast, random noise does not contain any meaningful information, thus would only trigger few causal relations in the SCM. Hence, it implies a highly narrow distribution on the values of deep neurons (purple region). Nevertheless, adversarial-generated inputs often largely exploits some incorrect causal relations extrapolated by DNN to deceive model decisions. These incorrect causal relations are often rarely used in predicting normal inputs. Such inputs would yield a notable distribution shifting (orange region) with respect to normal inputs. Therefore, we anticipate that different input types are distinguishable by CC, as they trigger distinct causal relations. See consistent empirical findings in Sec. VI.

[2]To clarify, our proof is based on [1] instead of directly extending [37].

[3]"Non-input" node simply means that node v is not in the first DNN layer.

[4]Rigorously speaking, causal realtions between neurons are mathmatically sound (proved in Proposition 1), yet hardly interpretable by humans. We use this example to illustrate the concept in an understandable manner.

B. DNN Coverage Based on Causal Discovery

As introduced in Sec. II-B, in forming DNN input coverage criteria, existing efforts rely primarily on neuron outputs as the basic computing unit. Nevertheless, neuron-wise criteria treat neurons individually, without considering correlation/causality between neurons. Layer-wise and trace-/path-wise criteria view multiple neurons as a whole for calculating coverage. Nevertheless, it is not modeled how one neuron may influence another. Overall, these metrics may fail to sufficiently depict DNN's decision mechanisms, as will be shown in evaluation (Sec. VI). While NPC employs XAI techniques to identify critical neurons for DNN decisions, the constructed decision paths simply chain neurons from the first to the last layers, which hardly reflect the decision mechanism or executing states. Also, NPC requires white-box access (i.e., requiring gradients) to the DNN, which is not always practical for released models.

A Causal View of Test Input Assessment. By revisiting the problem through the lens of causality, we find that the causal graph to the SCM entailed by a DNN provides a plausible abstraction to the execution states of this DNN. By performing causal discovery over DNN executions to a collection of test inputs (i.e., multiple realizations of the SCM), we are able to deduce a causal graph $\mathcal{G}$ of high quality. Consider two connected neurons $v_1 \rightarrow v_2$ in a DNN. According to the SCM construction for Proposition 1, v_1 and v_2 is adjacent on $\mathcal{G}$. To decide if there exists an edge connecting v_1, v_2, we can launch CI tests using a DNN test suite. An edge in the causal graph indicates that the causal relation between two neurons is established with given inputs. Therefore, if a test suite allows us to identify the majority of edges in the ground truth causal graph, then this input collection should be deemed as comprehensive. Otherwise, if only few edges are identified, this test suite is deemed as homogeneous, triggering only few decision mechanisms in the DNN. Holistically, we presume that high coverage of causal relations in a test suite is indicative of high test adequacy, as discussed further in Sec. VII.

A good test suite should enable uncovering new edges on the causal graph extracted from a DNN, indicating that new causal relations in the test suite are offered to test DNNs.

CC: A Causal-Aware Criterion. With discussion above, let $\mathcal{G} = (V, E)$ be the ground truth causal graph entailed by the target DNN $\mathcal{N}$ and $\hat{\mathcal{G}} = (\hat{V}, \hat{E})$ be the causal graph learned from feeding a test suite to $\mathcal{N}$. The recall of edges in $\hat{\mathcal{G}}$ over the ground truth $\mathcal{G}$ (i.e., $\frac{|E \cap \hat{E}|}{|E|}$) constitutes a plausible coverage criterion that quantifies how many causal relations encoded in $\mathcal{N}$ are exhibited when processing the inputs. We name this criterion as CC $= \frac{|E \cap \hat{E}|}{|E|}$.

IV. Design of CC

Sec. III connects DNNs and SCM, where an DNN $\mathcal{N}$ derives a *ground truth* causal graph (Proposition 1). Sec. III also explains that DNN prediction can be simulated via SCM realization. Thus, to compute CC, the intuition is to first extract the ground truth causal graph $\mathcal{G} = (V, E)$. Then, when processing a test suite, we collect SCM realizations over all incurred DNN executions, thus learning the causal graph $\hat{\mathcal{G}} = (\hat{V}, \hat{E})$. CC can be trivially computed $\frac{|E \cap \hat{E}|}{|E|}$.

Technical Challenges of Discovering $\hat{\mathcal{G}}$. Despite the straightforward concept, it is highly difficult to apply standard causal discovery algorithms on SCM realizations collected from DNN executions. We discuss key technical challenges below.

First, exiting causal discovery relies on several canonical assumptions to ensure the identified causal graph is correct [32]. Markov condition and faithfulness assumption ensure that d-separation (a form of structural constraint) implies conditional independence (CI) in the joint probability distribution, and vice versa. However, these assumptions are not fully satisfied in causal graphs derived from DNNs. To address this hurdle, we recast the problem of causal discovery on DNN executions into edge falsifications, which checks whether edges on ground truth causal graph $\mathcal{G}$ can be evidently established in DNN executions; see Sec. IV-A.

Second, the complexity of existing causal discovery algorithms is $\mathcal{O}(n^m)$ in general where n is the number of variables and m is the maximal in-degree of nodes (Max-InD) in the causal graph. Hence, existing algorithms are only scalable to process graphs with Max-InD $m \leq 5$. However, given that a neuron in a modern DNN would contain hundreds of preceding neurons (i.e., Max-InD $m \gg 5$), applying off-the-shelf algorithms on DNN is not amenable. Further, high Max-InD would also impose curse of dimensionality in statistical tests [55], which makes it almost impossible to establish CI with statistical significance in a reasonable sample size. Hence, based on domain specific knowledge and observation on DNNs, we propose a novel simplification and corresponding adjustment to tackle the issue imposed by high Max-InD; see Sec. IV-B.

Third, recall that, in common DNN testing scenarios, coverage is computed in an incremental manner. However, existing algorithms usually learn the causal graph in *one shot*. In other words, when a new DNN execution is observed, we need to start from scratch to perform causal discovery. Sec. IV-C presents an incremental learning paradigm.

Our tentative study also shows that when testing large DNNs, CC may be still slow due to the large number of DNN neurons/layers involved in edge falsification. We incorporate a selective causal learning strategy, where only a subset of layers in a DNN is considered, whose established CC is dubbed as CC$_m$. This strategy trades comprehensiveness for efficiency. More importantly, we formally prove that CC$_m$ enjoys the same properties of CC, and optimizations on analyzing full DNNs can be seamlessly applied. See details in Sec. IV-D.

Overview. In sum, CC is designed to deliver an *efficient*, *incremental* causal discovery scheme that is specifically tailored for DNN testing. For the technical pipeline of CC, we first convert the DNN $\mathcal{N}$ into the ground truth causal graph $\mathcal{G}$. Then, during a "warm-up" phase, we feed the DNN with a test suite to record neuron values for follow-up edge falsification analysis. After that, during the test suite assessment phase, we iteratively perform edge falsification (Sec. IV-A) to decide if any edge on the ground truth causal graph $\mathcal{G}$ can be removed, under the adjustment of confounding neurons (Sec. IV-B).

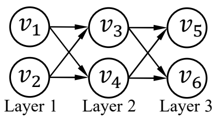
This step is conceptually similar to the standard "skeleton learning" procedure introduced in Sec. II-C. With well-designed optimizations, this online causal discovery phase is speedy, and can be performed incrementally with low cost (Sec. IV-C). Finally, we compute CC, which is the ratio of unfalsified edges over the total number of edges in $\mathcal{G}$.

A. From Costly Causal Discovery to Efficient Edge Falsification

The unique opportunity in designing CC, compared to standard causal discovery, is that the ground truth causal graph $\mathcal{G}$ is trivially derived from the target DNN. Recall that $\text{CC} = \frac{|E \cap \hat{E}|}{|E|}$ which only checks whether an edge in the ground truth also exists in $\hat{\mathcal{G}}$ recovered from DNN executions. An efficient way is to check whether an edge can be *falsified* by performing statistical tests using DNN executions of test inputs. That is, we decide if this edge can be removed (falsified) from the ground truth $\mathcal{G}$. Let a DNN represented in $\mathcal{G} = (V, E)$ and a pair of connected neurons $(v_1, v_2) \in E$. The sound and complete falsification condition [32] is

$$(v_1, v_2) \in E \iff \forall \boldsymbol{z} \subseteq V \setminus \{v_1, v_2\}, v_1 \not\perp v_2 \mid \boldsymbol{z} \quad (2)$$

Overall, CC explores an novel design angle by recasting the problem of causal discovery in DNN executions into edge falsifications (toward ground truth $\mathcal{G}$) from DNN executions. This way, CC omits non-adjacent nodes according to $\mathcal{G}$, because checking the RHS (right-hand side) condition in Eqn. 2 for each edge in $\mathcal{G}$ suffices to establish $\hat{\mathcal{G}}$ for computing CC. However, checking the RHS condition still suffers exponential complexity and is computationally expensive, which necessitates further simplification to the RHS condition while preserving the falsification power (i.e., low false positives).



$$v_3 = 0.8v_1 + 0.5v_2$$
$$v_4 = 0.5v_1 + 0.6v_2$$
$$v_5 = 0.1v_3 + 0.8v_4$$

Figure 5. Example of a three-layer DNN.

B. Adjustment to Confounding Neurons

A possible simplification to Eqn. 2 is to check whether $v_1 \not\perp v_2$ with $\boldsymbol{z} = \emptyset$, which forms a necessity condition w.r.t. the RHS condition in Eqn. 2. However, such simplification would be inaccurate due to the presence of confounding neurons. Given an edge $x \rightarrow y$, we define the confounding neurons as the other parents of y (i.e., after excluding x). We now show how confounding neurons make the above simplification procedure inaccurate. Consider the illustrative example of a 3-layer DNN in Fig. 5. While there is an edge $v_3 \rightarrow v_5$, the causal relation between them is subtle (with a relatively small coefficient of 0.1). In contrast, the confounding neuron v_4 has a stronger influence to v_5 (with a higher coefficient of 0.8). Furthermore, v_3 and v_4 are highly dependent, as they share same parents in the ground truth causal graph $\mathcal{G}$. Therefore, given that v_5 is largely decided by v_4 and v_3 is highly correlated with v_4, v_4 would create a spurious association between v_3 and v_5, making estimation of the edge between v_3 and v_5 very inaccurate. When the RHS condition in Eqn. 2 is enforced, we can perform CI tests to alleviate this issue (i.e., check $v_3 \not\perp v_5 \mid v_4$). However, as aforementioned, with hundreds of preceding neurons, curse of dimensionality makes launching RHS condition in Eqn. 2 infeasible.

Algorithm 1: Adjustment to Confounding Neurons

```
Input: Target Edge: $x \rightarrow y$, Confounding Neurons of $x \rightarrow y$:
       $\{v_1, \cdots, v_k\}$, Structural Assignment to $y$:
       $f_y : (x, v_1, \cdots, v_k) \mapsto y$
Output: Whether $x \rightarrow y$ can be falsified
1 foreach $v_i \in \{v_1, \cdots, v_k\}$ do
2     $\alpha_i \leftarrow \frac{|\text{cov}(v_i, x)|}{\sum_1^k |\text{cov}(v_j, x)|}$;
3     $v'_i \leftarrow (1 - \alpha_i) v_i$;
4 end
5 $y' \leftarrow f_y(x, v'_1, \cdots, v'_k)$;
6 if $x \not\perp y'$ then return yes;
7 else return no;
8 return $X$
```

We outline our adjustment strategy in Alg. 1. To falsify an edge $x \rightarrow y$, instead of directly checking independence between them (which may be biased due to confounding neurons), Alg. 1 checks whether x is independent of an adjusted y', where y' eliminates the influence from confounding neurons. For each confounding neuron v_i (line 1), we compute the absolute covariance between v_i and x which quantifies their correlation. This correlation is normalized with other confounding neurons and forms a coefficient α_i (line 2). The values of this confounding neuron are scaled by $(1 - \alpha_i)$ as a penalty (line 3). Finally, the adjusted y' is computed by x and the adjusted confounding neurons $v'_1, \cdots, v'_k$ with the same structural assignment function in the DNN (i.e., we need to replace the values in the confounding neurons and execute the DNN again) (line 5). If x is not independent of y', we can establish this edge (line 6), meaning that we confirm the causal relation. Otherwise, we falsify this edge (line 7); because the connection between x and y is not evidently observed by statistical tests. By checking x with the adjusted y', the influences from other confounding neurons are reduced and our falsification becomes more accurate and practically feasible. As shown in Sec. VII, Alg. 1 evidently improves the effectiveness.

C. Incremental Computing Scheme

In real-world DNN testing scenarios, coverage is usually computed in an online, batch-based manner. Therefore, it is important to implement an incremental computing scheme such that previous intermediate results can be effectively reused. The major computational overheads of CC originates from executing Alg. 1. With a collection of incoming DNN executions, if an edge is confirmed previously (i.e., cannot be falsified), Alg. 1 can simply skip this edge. However, if an edge is falsified, a follow-up check on the incoming DNN executions by Alg. 1 is required, which necessitates an incremental computing scheme for efficiency.

CC implements a GPU-based computing pipeline for edge falsification which maintains several updatable intermediate values, such that incremental DNN executions can be smoothly appended. In the warm-up phase, an initial test suite is used to compute α_i in Alg. 1. Recall that α_i is derived from covariance, which is generally invariant regarding incremental data. Hence, it is reuseable in subsequent incremental data without modifications (thus line 2 and 4 is merely executed in the warm-up phase). This simplification in turn allows us to reuse y' previous DNN executions as they are also invariant (line 3 and 5 is only computed on incoming data).

Hence, the major challenge stems from independence tests (line 6). However, in general, existing independence test methods on continuous data are *not* updatable. Inspired by prior coverage criteria that convert neuron values in discrete values, we group continuous neuron values into K splits and incorporate a discrete updatable independence test method called χ^2-test [3]. In essence, CC maintains the contingency table in GPU that encodes that joint distribution and marginal distribution of x, y. With the strong parallelism power of modern GPU, edge falsification is performed in parallel to all edges and computational overheads is further reduced.

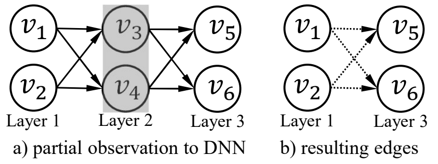


Figure 6. Example of performing layer selection toward a three-layer DNN.

D. Optimization: Selective Causal Discovery

Modern DNNs usually have enormous neurons; we find that it is sometimes costly and unnecessary to launch causal discovery to the whole DNN. We thus offer CC$_m$, as a minimal version of CC, to abstract a complex DNN into several key layers. To date, it is acknowledged [27, 44] that neurons at DNN front layers process low-level features (e.g., colors and textures of an image), while neurons at back layers process high-level semantics of the input (e.g., overall image scenes).

CC$_m$ conducts *selective causal discovery* to minimize the cost of CC. Considering Fig. 6(a), where we expect that by capturing causal relations between the front and back layers, CC$_m$ can sufficiently model the execution states of DNNs.

Despite the promising potential, a few selected DNN layers however do *not* obviously entail a desirable causal graph on which we can perform edge falsifications, because this minimization scheme introduces extra latent variables [54], making it unclear if the kept neurons should be still connected on the minimal causal graph. Consider Fig. 6(a), when the second layer is dropped, v_3, v_4 become latent variables, and v_1 and v_5 are non-adjacent in the causal graph. We present the proposition below to neurons in the selected layers are deemed to be connected.

Proposition 2. *Let a DNN contain neurons $x, y, v_1, \cdots, v_n$ and the corresponding causal graph be $\mathcal{G}$. Without loss of generality, let $L = \{v_1, \cdots, v_k\}$ be the unobserved neurons (i.e., neurons in dropped layers) and $\mathcal{G}_m$ be the resulting graph when $v \in L$ is unobserved. If there exists an path from x to y and every non-endpoint neuron in the path is in L, there exists an edge in $\mathcal{G}_m$ from x to y.*

Proof Sketch. We first prove that if every non-endpoint neuron in the path from x to y is in L, the path is an inducing path [54]. Hence, by the definition of causal graph in the presence of latent variables [54], since there is an inducing path between x and y and x is an ancestor of y, $x \to y$ exists in G_m. □

CC$_m$ Correctness. Proposition 2 implies that, given two layers L_i and L_j, if L_i is a preceding layer of L_j and every layer between L_i and L_j is dropped, every neuron in L_i targets to every neuron in L_j, which leads to the edges in Fig. 6(b). Hence, all properties of an edge discussed in above subsections holds for CC$_m$. This justifies the correctness of CC$_m$, as a selective causal discovery scheme.

CC$_m$ Cost. We clarify that CC$_m$ can largely reduce the cost of applying CC. As an estimation, consider a FFNN with K layers and each layer contains n_i neurons ($1 \leq i \leq K$). CC needs to falsify a total of $\sum_1^{K-1} n_i \times n_{i+1}$ edges, whereas by hooking $m \ll K$ layers (let these layers be $l_1, \cdots, l_m$-th layer), CC$_m$ only falsifies $\sum_1^{m-1} n_{l_i} \times n_{l_{i+1}}$ edges. Hence, CC$_m$'s cost is largely reduced. In practice, we find that, when $m = 3$, the number of edges are reduced by 95% in ResNet50.

CC$_m$ Concretization. Despite the promising properties of CC$_m$, it is worth noting that real-world DNNs can have tens to hundreds of layers, where the plan of selecting which and how many layers may be obscure. As aforementioned, it has been widely acknowledged that different layers in a DNN are often responsible for focusing on either coarse-grained or fine-grained visual concepts [27, 44]. As a recommendation, we would suggest to select at least three layers from the front, middle, and back layers toward a commonly-seen DNN with tens to hundreds layers. We present empirical results that suggest that the selection of layers in **RQ3** (Sec. VI-C). Users can follow our recommendation or launching similar empirical exploration to decide a good concretization of CC$_m$.

V. Implementation and Evaluation Setup

CC is implemented using PyTorch with roughly 2,900 lines of code [2]. All experiments are launched on one Intel Xeon CPU E5-2683 with 256GB RAM and one Nvidia RTX 3090 GPU. We now introduce the evaluation setup.

Table I
Statistics of tested DNN models.

Model	#Parameters	#Neurons	#Layers
ResNet50 [14]	23.5M	26,570	54
VGG16 [35]	5.5M	7,162	56
LeNet5 [21]	7.0M	10,250	121

Table II
Statistics of evaluated datasets.

Dataset	#Train/Val Samples	#Classes	Image Size	Color
MNIST [7]	50,000/10,000	10	32 × 32	Black/White
CIFAR10 [20]	50,000/10,000	10	32 × 32	RGB
ImageNet [6]	1M/50,000	1,000	224 × 224	RGB

Models. Table I reports the statistics of three DNN models used for the evaluation. All three leveraged models are large-size DNNs with complex architectures and a diverse set of DNN operators. They are all widely-used for CV tasks. Each of them has 7k to 27k neurons with up to 23.5M parameters.

Datasets. Table II lists the three datasets used in our evaluation. MNIST consists of black and white images and CIFAR10 contains images of RGB color channel. Moreover, ImageNet has over 1M images of 1,000 classes which is very large-scale. Note that because of the property of Chi Square test (to guarantee its result be valid, the expected value of each category of the combination of two variables should be larger than 5), CC requires at least 250 samples in labels to retain the reliability. Considering in some experiment we only sample part of the whole test set, we sample 500 samples from each classes in ImageNet with the help of RandomResizedCrop, RandomHorizontalFlip, and RandomVerticalFlip provided by PyTorch. Without the loss of generality, we select the first 20 classes in our experiment to ensure the size of datasets is not too large. Overall, these datasets are representative and widely used in existing works [17, 22, 28].

Baseline Coverage Criteria. Below we briefly introduce prior criteria that are compared with CC and discuss their setups.

NC has a threshold T to decided whether a neuron is activated. Following previous works [30], we set T to 0.

NC$^+$ is a variant of NC where the threshold T is applied on rescaled neuron outputs (i.e., rescale the output into $[0, 1]$) [30]. We set T to 0.25 and 0.75.

KMNC splits the normal neuron output ranges into K section. We set K as 10, 100, and 1000 in this evaluation. These setting are also used in previous works [22, 46, 47].

NBC/SNAC does not have parameters. Note that KMNC/NBC/SNAC require deciding normal each neuron's output range using all training data, which is costly due to the large size of evaluation datasets. Therefore, we randomly select 1,000 training samples to scope neuron output ranges.

TKNC/TKNP regards a neuron as activated if it has the top-K output among neurons of the same layer. We set K to $1, 5, 10$ according to Ma et al. [22].

TFC has one parameter T depicting the distance threshold when forming a cluster. Following [28], we set T to $10, 20$.

LSA/DSA/MDSA has two key parameters for these criteria, the bucket count m and the maximal SA value U. Nevertheless, we find it is challenging to tune the two hyper-parameters, and the authors do not provide guide regarding the choice of hyper-prarameters. To ease the presentation, we only use the bucket size $T = \frac{U}{M}$ and report the covered buckets instead of the ratios of covered buckets in the original paper.

NPC requires white-box access to the tested DNN, whose assumption is stronger than existing gray-box criteria. We point out that white-box access may be infeasible in certain cases (e.g., the DNN is deployed on end devices). Thus, we omit the comparison with NPC.

Numerical Unit of Coverage Values. For results reported in tables in Sec. VI, the numerical unit of coverage values is percentage (%) for NC, NC$^+$, KMNC, NBC, SNAC, TKNC, and CC. In contrast, the coverage values in TKNP, TFC, LSC, DSC, MDSC are the exact numbers of patterns/clusters/patterns.

VI. Evaluation and Findings

We mainly explore the following research questions (RQs). **RQ1 (Quantify of Causality)**: Does CC correlate with real-world inputs of different causality degrees? **RQ2 (Adversarial/Backdoor Inputs)**: Can CC manifest high distinguishability over adversarial data? Can CC capture the hidden/strong causality in adversarial inputs? **RQ3 (Optimization)**: What are the performance of CC$_m$, as an optimized CC that only captures causality of a few selected DNN layers? In addition, we present a qualitative evaluation illustrating how causality of different DNN inputs can be reflected by CC in Sec. VI-D.

Table III
Coverage results. In the 50% column, we report increased ratio of 30% $\rightarrow$ 50%. In the 100% column, report the increased ratio of 50% $\rightarrow$ 100%. For those with high distinguishability, we highlight criteria whose increase ratios $\geq 10\%$, criteria whose increase ratios $\in [5\%, 10\%)$.

		LeNet5/MNIST			VGG16/CIFAR10			Resnet50/ImageNet		
	Config.	**30%**	**50%**	**100%**	**30%**	**50%**	**100%**	**30%**	**50%**	**100%**
CC	k=8	0.327	34.6%	40.8%	0.341	33.5%	33.7%	0.043	13.4%	12.2%
NC	T=0	0.695	0.6%	0.5%	0.837	0.3%	0.9%	0.991	0.0%	0.0%
NC$^+$	T=0.25	1.000	0.0%	0.0%	1.000	0.0%	0.0%	0.968	0.1%	0.0%
	T=0.75	0.849	0.7%	-0.5%	0.587	0.8%	2.5%	0.350	6.4%	1.4%
KMNC	T=10	0.995	0.2%	0.0%	0.990	0.3%	0.1%	0.980	-2.9%	1.2%
	T=100	0.934	0.9%	1.7%	0.925	-0.3%	2.0%	0.853	-3.3%	0.9%
	T=1000	0.719	5.4%	1.1%	0.705	3.9%	1.5%	0.622	-6.1%	0.3%
NBC	N/A	0.638	10.0%	14.2%	0.716	7.5%	4.1%	0.502	6.3%	11.7%
SNAC	N/A	0.653	2.9%	11.7%	0.747	3.4%	5.9%	0.518	14.8%	13.2%
TKNC	K=1	0.578	1.0%	0.1%	0.082	1.6%	1.3%	0.129	12.5%	-1.4%
	K=5	0.717	0.1%	0.2%	0.205	1.5%	0.0%	0.321	5.4%	5.0%
	K=10	0.752	0.1%	0.6%	0.308	0.7%	2.3%	0.460	3.0%	4.7%
TKNP	K=1	549	10.1%	14.0%	3936	2.9%	-0.9%	3862	-24.5%	-0.8%
	K=5	4067	3.1%	0.0%	4100	2.4%	0.0%	4000	-25.0%	0.0%
	K=10	4012	4.7%	0.0%	4100	2.4%	0.0%	4000	-25.0%	0.0%
TFC	T=10	9992	3.3%	5.0%	172	21.8%	-0.9%	13482	-26.2%	6.9%
	T=20	9048	6.1%	2.0%	93	-0.4%	0.2%	11452	-27.3%	8.7%
LSC	T=1	2255	4.3%	-1.2%	287	10.5%	-5.8%	N/A	N/A	N/A
	T=10	1870	3.8%	1.9%	115	-6.4%	-5.3%	N/A	N/A	N/A
DSC	T=0.1	16	-2.1%	0.0%	48	8.3%	3.8%	N/A	N/A	N/A
	T=0.01	109	0.3%	4.9%	210	-1.4%	15.9%	N/A	N/A	N/A
MDSC	T=10	11	6.2%	-8.8%	164	-14.0%	-10.6%	N/A	N/A	N/A
	T=100	2	0.0%	0.0%	9	44.4%	7.7%	N/A	N/A	N/A

A. *RQ1: Causality*

Motivation. Ideally, a criterion should yield higher coverage values if a test suite contains greater diversity, which can be reflected by the number of conveyed causal relations. To benchmark whether prior criteria and CC can faithfully reflect the diversity, we construct test suites that convey causal relations of different degrees. A proper criterion is expected to distinguish these test suites, yielding coverage results that are aligned with the conveyed causal relations of varying degrees.

Setup. Intuitively, a test suite with more classes should bring in a higher degree of causal relations, because a DNN behaves distinctly for inputs of different classes. Therefore, for each dataset, we form three test suites by using 30%, 50%, and 100% of the classes. We keep the number of selected test cases in different test suit be the same. The derived coverage for each criterion should rank as: $100\% > 50\% > 30\%$. In addition, since the number of classes in these test suites vary notably, their coverage values are expected to be distinguishable enough.

Results. Table III reports the results over different criteria across different settings, and we highlight testing criteria with plausible results. Since all SC criteria (LSC, DSC, and MDSC) require the whole training set to initialize and for each test input, LSC and DSC iterate over all neuron output traces generated using training data, we find that it is impractical to complete them in a reasonable time (ImageNet has 1M training samples). MDSC stores a class conditional covariance matrix for representing the training data. Since ImageNet has 1,000 classes, MDSC requires huge amount of GPU memory, resulting in an infeasible evaluation. Therefore, we do not report the results for these criteria on Resnet50/ImageNet.

We do not highlight coverage criteria whose increase ratios are not larger than 0% in the 50% and 100% columns even if it has a very high ration in the 30% column. Except NBC, SNAC, and CC, other criteria fail to exhibit growing coverage values when the number of classes increase in all three models/datasets. We use purple of different degrees to reflect the increase ratio, where CC manifests the *highest* increase ratio.

As introduced in Sec. II-B, coverage criteria based on individual neurons, including NC/NC$^+$, KMNC, and NBC/SNAC, treat each neuron separately. The causal relations among neurons are thus neglected, and as a result, they fail to comprehensively assess the execution DNN input quality. TKNP/TKNC have similar issues, the relative magnitude of neuron outputs can hardly reflect the causal relations among neurons during the DNN execution. Moreover, since TFC/SCs prefer inputs that are distinct with training data, they are distracted by focusing on input *dissimilarity*. It is reasonable that they do not respond when the number of classes increases. Nevertheless, CC, which models the causal relations among neuron by design, can correctly reflect the internal activities and execution states of a DNN.

NBC and SNAC also manifest growing and distinguishable coverage values for the three test suites (in ascending order of class labels). Recall that NBC/SNAC regarded a neuron as activated if its output falls outside the normal range (decided using training data). Given that test inputs are different from the training data and the #classes in three test suites vary notably, it is likely that they trigger out-of-bound (OOB) neuron outputs with distinct frequencies. Nevertheless, we argue that these two coverage criteria do not necessarily reflect true causal relations of test suites, as evaluated below.

Comparison with NBC/SNAC. Though NBC/SNAC have expected performance on these test suites, we hypothesize that they have limits by design: the OOB behaviors of neurons may be *falsely* distracted toward meaningless inputs, as they can likely trigger suspicious neuron outputs. Following, we evaluate NBC, SNAC, and CC, using new test suites that are generated by adding white noise on the test dataset. The white noise is generated using the random utility provided by Pytorch. We control the range of noise (i.e., $[-1, 1]$, $[-100, 100]$, and $[-10000, 10000]$) and record how coverage values change.

Table IV
COVERAGE ACROSS DIFFERENT CRITERIA ON MUTATED TEST DATASET.

	LeNet5/MNIST				VGG16/CIFAR10				Resnet50/ImageNet			
	x0	**x1**	**x100**	**x10000**	**x0**	**x1**	**x100**	**x10000**	**x0**	**x1**	**x100**	**x10000**
CC	0.878	0.532	0.156	0.040	0.899	0.096	0.081	0.036	0.425	0.233	0.012	0.003
NBC	0.914	0.802	0.955	0.957	0.909	0.230	0.389	0.596	0.677	0.458	0.418	0.521
SNAC	0.910	0.766	0.917	0.918	0.919	0.280	0.180	0.306	0.794	0.493	0.585	0.807

As shown in Table IV, NBC and SNAC keep increasing when enlarging the range of noise, whereas CC keeps *decreasing*. We interpret the results as highly intuitive. Overall, noisy inputs of higher magnitudes are more likely to trigger OOB neuron outputs, such that NBC/SNAC speciously treat overly large neuron outputs as a new and legit "coverage", and therefore, their yielded coverages incorrectly grow. Contrarily, with noise magnitude increasing, inputs lose the ability to trigger causal relations between neurons. Additionally, since noise data have poor diversity and contain non-causal relation, CC can retain fewer edges on the ground truth causal graph $\mathcal{G}$ when the noise magnitude increases, such that the coverages keep decreasing.

Answer to RQ1: CC manifests high distinguishability and correct correlation with real-world test inputs of varying causality degrees, thereby comprehensively modeling the execution states of DNNs. In contrast, existing coverage criteria either reflect a negligible correlation with causality (i.e., diversity), or get easily misled by unreasonable cases.

B. RQ2: Adversarial Inputs

Adversarial Examples (AE). From a causality perspective, we consider AEs contain a large volume and unique causality information, in the sense that they can trigger many erroneous outputs over a DNN, which, to some extent, denote DNN states that haven't been explored. To prepare AEs, we use all 10,000 images from the testing split of the CIFAR-10 dataset to form a collection of *normal inputs*. Then, we generate 10,000 AEs using two stateo-of-the-art (SOTA) AE generation algorithm, Projected Gradient Descent (PGD) and Fast Gradient Sign Method (FGSM) [16, 26]. We follow the default configuration which has a perturbation budget of $\epsilon = 0.3$. Note that, as both algorithms launch a white-box attack (requires DNN gradients to guide AE generation), it has proved very difficult to defend against, either theoretically or empirically [33, 49]. For each setting, we generate 10,000 AEs, and we confirm all of them can trigger DNN mis-predictions.

Backdoor Inputs. Recent works have explained DNN predictions over backdoor inputs from the view of causality [37]. In short, arbitrary inputs, when being added with a pre-defined backdoor trigger (e.g., a pair of glass), will enforce the DNN predictions toward a fixed label. It is believed that backdoor inputs impose unique and strong causal relations to manipulate DNN predictions [37]. To prepare backdoor inputs, we use the classical backdoor attack method, badnet [12].

Setup. We clarify that at this step, we first run the CIFAR10 test split on VGG16 equipped with each of the coverage criteria. Then, we use either AEs or backdoor inputs to test those already "warmed-up" criteria. As a baseline, we also run on the same amount of random noise, which is generated based on the randn API of PyTorch, in comparison with AEs and backdoor inputs. AEs are generated using the CIFAR10 test inputs. Compared with the normal data, AEs introduce new causal relations. Therefore, a correct criterion is expected to yield much higher coverage values for two AE sets compared with the random noise. Similarly, backdoor inputs contain backdoor triggers that are never "tested" using training data, and their corresponding coverage values should also be much larger than the random noise.

Table V
Adversarial inputs evaluation across different criteria.

	Config	Randn	PGD	FGSM	Backdoor
CC	k=8	1.08%	7.66%	8.69%	8.55%
NC	T=0	0.49%	1.55%	1.15%	0.17%
NC^+	T=0.25	0.00%	0.04%	0.02%	0.00%
	T=0.75	0.36%	0.80%	1.36%	1.69%
KMNC	T=10	0.01%	0.01%	0.01%	0.04%
	T=100	0.16%	0.90%	0.41%	0.99%
	T=1000	1.22%	3.43%	2.41%	5.96%
NBC	N/A	0.39%	2.46%	0.92%	1.54%
SNAC	N/A	0.43%	3.36%	1.13%	1.01%
TKNC	K=1	1.97%	5.83%	8.72%	4.46%
	K=5	0.71%	3.75%	5.63%	1.84%
	K=10	1.37%	4.17%	4.79%	0.91%
TKNP	K=1	17.35%	67.36%	86.13%	68.07%
	K=5	90.91%	90.91%	90.89%	81.82%
	K=10	90.91%	90.91%	90.89%	81.82%
TFC	T=10	1.97%	9.23%	197.16%	0.81%
	T=20	1.09%	8.49%	101.88%	0.70%
LSC	T=1	362.35%	419.86%	278.05%	4.77%
	T=10	117.57%	169.18%	88.59%	0.00%
DSC	T=0.1	0.00%	80.26%	47.89%	0.00%
	T=0.01	19.60%	163.77%	113.78%	0.00%
MDSC	T=10	159.66%	246.79%	89.13%	220.75%
	T=100	52.63%	299.70%	170.83%	50.00%

Results. Table V reports the results. For CC, coverage values increase much more on AE and Backdoor, in comparison with that of feeding noise data (i.e., $\times 7 \sim \times 8$). Most previous coverage criteria do not exhibit further increase over AEs (e.g., TKNP with $K = 5$) or backdoor inputs (e.g., TKNC with $K = 10$) when compared with the noise data. NC^+ does not nearly gain any increase because it saturates at very early stage. For TFC, although it increases much more on FGSM compared with noise data, it fails to perform as expected on backdoor inputs. DSC ($T = 0.1$) has the similar problem.

Answer to RQ2: CC further gain much more increase on adversarial inputs than noise data, after running a whole training set. This meets the expectation that adversarial inputs can reveal more subtle edges of the ground truth causal graph. Nevertheless, previous criteria either fail to assess AEs, or have no response to backdoor inputs.

Table VI
Evaluating different minimization schemes of CC on diversity.

	VGG16/CIFAR10			Resnet50/ImageNet		
	30%	50%	100%	30%	50%	100%
CC	0.341	33.5%	33.7%	0.043	13.4%	12.2%
CC_m	0.286	34.6%	10.8%	0.126	15.3%	15.5%
CC_m-F	0.152	-5.2%	7.8%	0.170	1.7%	13.3%
CC_m-M	0.265	6.2%	38.1%	0.118	-12.6%	1.4%
CC_m-L	0.766	9.8%	15.8%	0.032	9.8%	-21.0%

Table VII
Evaluating different minimization schemes of CC on adversarial inputs.

	Randn	PGD	FGSM	Backdoor
CC	1.08%	7.66%	8.69%	8.55%
CC_m	0.14%	1.73%	0.63%	5.85%
CC_m-F	8.98%	0.23%	2.48%	21.83%
CC_m-M	2.42%	3.30%	1.32%	16.25%
CC_m-L	0.34%	1.94%	0.22%	0.04%

C. RQ3: Optimization

This section explores CC_m, which selectively captures the causal relations of a few DNN layers to reduce cost. To this end, we launch an ablation study to configure CC_m to capture only three layers of the tested DNNs. Given VGG16, a complex DNN, we select one front layer, which is deemed to capture basic, low-level features, one middle layer which should abstract inputs' key feature, and one back layer which directly relates to the prediction.

We re-run evaluations conducted in Sec. VI-A and Sec. VI-B on CC_m. To verify whether the layer selection strategy is correct, we also form CC_m-F, a variant of CC_m that captures the first three layers, CC_m-M, a variant of CC_m that captures the middle three layers, and CC_m-L, a variant of CC_m that captures the last three layers. The results are shown in Table VI and Table VII. Overall, we find that CC_m still manifests good distinguishability in terms of full vs. biased data (Sec. VI-A), and noise vs. adversarial inputs (Sec. VI-B). In contrast, all three variants of CC_m fail to perform well on these experiments. For instance, we find that CC_m-F gain much more increase on noise data compared with AE data. Therefore, though CC_m is a good candidate for SOTA coverage criteria, a well-performing CC_m requires users to have a good knowledge of models under test. We suggest that user should use CC first, getting familiar with the DNN under test (and coverage results under CC), and then consider using CC_m to improve efficiency.

Table VIII
The time overhead (minutes) when processing 10,000 inputs.

	LeNet5/MNIST	VGG16/CIFAR10	Resnet50/ImageNet
CC	0.15	2.27	80.88
CC_m	0.15	1.28	26.35
NC	0.02	0.02	0.13
NC^+	0.02	0.02	0.13
KMNC	0.02	0.03	0.15
NBC	0.02	0.02	0.13
SNAC	0.02	0.02	0.13
TKNC	0.02	0.03	0.13
TKNP	0.20	0.58	0.87
TFC	10.52	0.38	18.43
LSC	0.18	3.30	N/A
DSC	1.80	16.93	N/A
MDSC	0.02	3.57	N/A

Table IX
The GPU memory consumption when batch size is 50.

	LeNet5/MNIST	VGG16/CIFAR10	Resnet50/ImageNet
model only	2031 MB	2207 MB	2793 MB
with CC	2107 MB	3479 MB	14695 MB
with CC_m	2053 MB	2277 MB	3863 MB

Computing Overhead. We also report the inference time regarding 1) the vanilla DNNs and 2) DNNs + coverage criteria enabled. We report the results of different DNNs and datasets in Table VIII. It is seen that CC (not CC_m) manifests a comparable overhead with other criteria, and is much faster than all SCs. As expected, CC_m is much faster. In addition, the GPU memory consumption is presented in Table IX. We interpret the overall memory usage (the 3rd row) as reasonable and in line with our expectations, and a notable improvement is observed on CC_m compared CC (3,863 MB vs. 14,695 MB).

CC_m also provides patterns that are easier to understand than CC, because CC_m has fewer layers that are easy for visualization. In Sec. VI-D, we further explore the interpretability of CC_m qualitatively. Thus, we make the following conclusion:

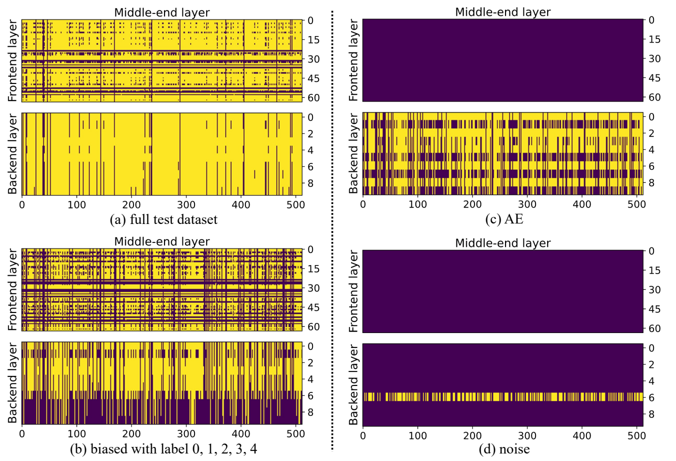


Figure 7. Qualitative illustration of how different inputs are reflected by CC.

Answer to RQ3: Capturing the causality of only a few DNN layers manifests adequate support as a test criterion. This illustrates the robustness of our causal-aware scheme, and the generalization of interpreting DNN predictions via causality. Nonetheless, CC$_m$ heavily relies on users' priori knowledge of the models being tested, and is susceptible to failure due to erroneous layer selection.

D. Qualitative Study

We now present a qualitative study about how inputs with different causality are be reflected by CC. We use CC$_m$, which facilitates better visual interpretability, at this step. We prepare four CIFAR10 input collections used in **RQ1/RQ2**, i.e., full test inputs (including label 0∼9), test inputs with biased labels (only include label 0, 1, 2, 3, 4), adversarial inputs, and noise inputs. We then feed these inputs to VGG16 with CC$_m$ enabled.

We present the evaluation results in Fig. 7. Due to the limited space, we present more visualization cases at [2]. Overall, we interpret the empirical findings here are highly correlated with our theoretical analysis in Fig. 4. CC$_m$ hooks three layers at the frontend, middle-end, and back-end of VGG16. It is observed that inputs with different causality manifests distinct coverage patterns upon CC$_m$. For instance, when comparing Fig. 7(a) and Fig. 7(b), we find that inputs with biased (less amount of) labels can trigger notably fewer edges in CC$_m$. Furthermore, most of those triggered edge are related to classes contained by the biased test inputs. This result justifies that noise data do not contain any meaningful causality. Besides, we can also observe that adversarial data (Fig. 7(c)) are different from normal inputs in terms of causality, as they trigger new (abnormal) causal patterns. Noise data only have a narrow distribution in Fig. 7(d), as they only trigger few causal relations.

VII. Discussion

Causal Relation Coverage vs. Test Adequacy. We wish to reiterate the position of this research. We generalize CARE [37] to formally establish the one-to-one mapping and equivalence of any DGA-compatible DNN and SCM. We (and CARE) presume that causal relations adequately reflect DNN's internal decision mechanisms. This is a fairly obvious observation that has also been recognized by the machine learning community [19, 48]. Then, we establish our coverage criterion over SCM (and optimizations) based on the core premise that *fully covering all causal relations in the SCM signals that the associated DNN has been sufficiently and thoroughly tested*. Consequently, evaluating how DNN responds to various test inputs is recast as evaluating how many causal relations can be detected in the SCM. In other words, a test suite is thought to completely stress the varied decision mechanisms of DNN (adequately test the model), if it unveils more causal relations.

Table X
Coverage results of CC and CC without Alg. 1.

	LeNet5/MNIST			VGG16/CIFAR10			Resnet50/ImageNet		
	30%	50%	100%	30%	50%	100%	30%	50%	100%
w/ Alg. 1	0.327	34.6%	40.8%	0.341	33.5%	33.7%	0.043	13.4%	12.2%
w/o Alg. 1	0.458	29.1%	25.7%	0.319	21.6%	28.6%	0.084	10.2%	-4.3%

Adjustment Algorithm. We propose a heuristically-designed adjustment algorithm to alleviate the impact of confounding neurons in Sec. IV-B. We study its effectiveness via an ablation study (see results in Table X). We observe that CC with Alg. 1 substantially outperforms the ablated version in all cases. In particular, on ImageNet, the counterpart fails to correctly reflect test suite's diversity. These empirical results show the necessity and effectiveness of our adjustment algorithm.

Threat To Validity. The primary threat to internal validity denotes the correctness of our computed causality scores and the studied DNN models. We use publicly available DNN models and datasets. Our causality computation is based on the well-established techniques like χ^2-test. We generate AEs and backdoor inputs also using publicly available implementations. These can mostly avoid the chances of incorrect implementation. There exists the potential threat that CC may not adapt to other types of inputs or DNN models. To mitigate this external threat, we design an approach that is inputs and DNN architectures agnostic. Thus, we believe that CC is applicable to other settings with comparably promising performance.

VIII. Related Work

We have reviewed existing DNN coverage criteria and causal discovery in Sec. II-B. We now give an overview of the related work in DNN testing. To benchmark the quality of DNNs, it has been seen that standard dataset can only reflect the hold-out accuracy and many real-world corner cases are rarely included in standard test inputs [31, 34, 39, 53]. To date, the SE community has primarily used well-formed testing oracles in conventional software, like metamorphic testing and differential testing, to determine if a DNN or a group of DNNs behave consistently to mutated inputs [8, 24, 31, 41, 45, 51, 56]. In addition to prediction accuracy, some domain-specific properties like model robustness [42], fairness [25, 58], safety [29] and interpretability [15, 40] are also incorporated to test DNNs.

IX. Conclusion

We propose CC, a causality-aware DNN coverage criterion. We formulate a causality perspective to depict DNN executions, and CC incorporates a set of optimizations for an efficient implementation. Evaluation over real-world normal and adversarial inputs justify the effectiveness of CC.

References

[1] Aws Albarghouthi et al. Introduction to neural network verification. *Foundations and Trends® in Programming Languages*, 7(1–2):1–157, 2021.

[2] anonymous. CC artifact. https://anonymous.4open.science/r/DL_CC-4000/, 2022.

[3] Yvonne M Bishop, Stephen E Fienberg, Stephen E Fienberg, and Paul W Holland. Discrete multivariate analysis. 1976.

[4] Wieland Brendel and Matthias Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. *arXiv preprint arXiv:1904.00760*, 2019.

[5] Aditya Chattopadhyay, Piyushi Manupriya, Anirban Sarkar, and Vineeth N Balasubramanian. Neural network attributions: A causal perspective. In *International Conference on Machine Learning*, pages 981–990. PMLR, 2019.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[8] Anurag Dwarakanath, Manish Ahuja, Samarth Sikand, Raghotham M. Rao, R. P. Jagadeesh Chandra Bose, Neville Dubash, and Sanjay Podder. Identifying implementation bugs in machine learning based image classifiers using metamorphic testing. In *ISSTA*, 2018.

[9] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019.

[10] Clark Glymour, Kun Zhang, and Peter Spirtes. Review of causal discovery methods based on graphical models. *Frontiers in genetics*, 10:524, 2019.

[11] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[12] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[13] Trygve Haavelmo. The probability approach in econometrics. *Econometrica: Journal of the Econometric Society*, pages iii–115, 1944.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[15] Bernease Herman. The promise and peril of human evaluation for model interpretability. *arXiv preprint arXiv:1711.07414*, page 8, 2017.

[16] Hoki Kim. Torchattacks: A pytorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950*, 2020.

[17] Jinhan Kim, Robert Feldt, and Shin Yoo. Guiding deep learning system testing using surprise adequacy. In *ICSE*, 2019.

[18] Jinhan Kim, Jeongil Ju, Robert Feldt, and Shin Yoo. Reducing dnn labelling cost using surprise adequacy: An industrial case study for autonomous driving. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1466–1476, 2020.

[19] Murat Kocaoglu, Christopher Snyder, Alexandros G Dimakis, and Sriram Vishwanath. Causalgan: Learning causal implicit generative models with adversarial training. *arXiv preprint arXiv:1709.02023*, 2017.

[20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[21] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.

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

[23] Pingchuan Ma, Rui Ding, Haoyue Dai, Yuanyuan Jiang, Shuai Wang, Shi Han, and Dongmei Zhang. Ml4s: Learning causal skeleton from vicinal graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1213–1223, 2022.

[24] Pingchuan Ma and Shuai Wang. Mt-teql: evaluating and augmenting neural nlidb on real-world linguistic and schema variations. *Proceedings of the VLDB Endowment*, 15(3):569–582, 2021.

[25] Pingchuan Ma, Shuai Wang, and Jin Liu. Metamorphic testing and certified mitigation of fairness violations in nlp models. In *IJCAI*, pages 458–465, 2020.

[26] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[27] Junhua Mao, Xu Wei, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan L Yuille. Learning like a child: Fast novel visual concept learning from sentence descriptions of images. In *Proceedings of the IEEE international conference on computer vision*, pages 2533–2541, 2015.

[28] Augustus Odena and Ian Goodfellow. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. *arXiv preprint arXiv:1807.10875*, 2018.

[29] Qi Pang, Yuanyuan Yuan, and Shuai Wang. Mdpfuzz: testing models solving markov decision processes. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 378–390, 2022.

[30] Kexin Pei. Deepxplore code release. https://github.com/peikexin9/deepxplore, 2017.

[31] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. DeepXplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 1–18, New York, NY, USA, 2017. ACM.

[32] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

[33] Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. Adversarial attacks and defenses in deep learning. *Engineering*, 6(3):346–360, 2020.

[34] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online, July 2020. Association for Computational Linguistics.

[35] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, May 2015.

[36] Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. *Causation, prediction, and search*. 2000.

[37] Bing Sun, Jun Sun, Long H Pham, and Jie Shi. Causality-based neural network repair. In *Proceedings of the 44th International Conference on Software Engineering*, pages 338–349, 2022.

[38] Youcheng Sun, Xiaowei Huang, Daniel Kroening, James Sharp, Matthew Hill, and Rob Ashmore. Structural test coverage criteria for deep neural networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5s):1–23, 2019.

[39] Zeyu Sun, Jie M Zhang, Mark Harman, Mike Papadakis, and Lu Zhang. Automatic testing and improvement of machine translation. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pages 974–985, 2020.

[40] Yongqiang Tian, Shiqing Ma, Ming Wen, Yepang Liu, Shing-Chi Cheung, and Xiangyu Zhang. To what extent do dnn-based image classification models make unreliable inferences? *Empirical Software Engineering*, 26(5):1–40, 2021.

[41] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. DeepTest: Automated testing of deep-neural-network-driven autonomous cars. ICSE '18, 2018.

[42] Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

[43] Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.

[44] Koen EA Van de Sande, Theo Gevers, and Cees GM Snoek. A comparison of color features for visual concept classification. In *Proceedings of the 2008 international conference on Content-based image and video retrieval*, pages 141–150, 2008.

[45] Shuai Wang and Zhendong Su. Metamorphic object insertion for testing object detection systems. In *ASE*, 2020.

[46] Xiaofei Xie, Tianlin Li, Jian Wang, Lei Ma, Qing Guo, Felix Juefei-Xu, and Yang Liu. NPC: N euron p ath c overage via characterizing

decision logic of deep neural networks. *ACM Transactions on Software Engineering and Methodology*, 2022.

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

[48] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. Dag-gnn: Dag structure learning with graph neural networks. In *International Conference on Machine Learning*, pages 7154–7163. PMLR, 2019.

[49] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824, 2019.

[50] Yuanyuan Yuan, Qi Pang, and Shuai Wang. Revisiting neuron coverage for dnn testing: A layer-wise and distribution-aware criterion. ICSE, 2023.

[51] Yuanyuan Yuan, Shuai Wang, Mingyue Jiang, and Tsong Yueh Chen. Perception matters: Detecting perception failures of vqa models using metamorphic testing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16908–16917, 2021.

[52] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[53] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 2020.

[54] Jiji Zhang. On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence*, 172(16-17):1873–1896, 2008.

[55] Kun Zhang, Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Kernel-based conditional independence test and application in causal discovery. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 804–813, 2011.

[56] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. DeepRoad: GAN-based Metamorphic Testing and Input Validation Framework for Autonomous Driving Systems. In *ASE*, 2018.

[57] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[58] Indre Zliobaite. Fairness-aware machine learning: a perspective. *arXiv preprint arXiv:1708.00754*, 2017.