Journal of Big Data

RESEARCH Open Access

Testing coverage criteria for optimized deep belief network with search and rescue

Kiran Jammalamadaka[1*] and Nikhat Parveen[2]

*Correspondence:
jvskkiran@yahoo.com
[1] Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, A.P, India
Full list of author information is available at the end of the article

Abstract

A new data-driven programming model is defined by the deep learning (DL) that makes the internal structure of a created neuron system over a fixed of training data. DL testing structure only depends on the data labeling and manual group. Nowadays, a lot of coverage criteria have been developed, but these criteria basically count the neurons' quantity whose activation during the implementation of a DL structure fulfilled certain properties. Also, existing criteria are not adequately fine-grained to capture delicate behaviors. This paper develops an optimized deep belief network (DBN) with a search and rescue (SAR) algorithm for testing coverage criteria. For an optimal selection of DBN structure, the SAR algorithm is introduced. The main objective is to test the DL structure using different criteria to enhance the coverage accuracy. The different coverage criteria such as KMNC, NBC, SNAC, TKNC, and TKNP are used for the testing of DBN. Using the generated test inputs, the criteria is validated and the developed criteria are capable to capture undesired behaviors in the DBN structure. The developed approach is implemented by Python platform using three standard datasets like MNIST, CIFAR-10, and ImageNet. For analysis, the developed approach is compared with the three LeNet models like LeNet-1, LeNet-4 and LeNet-5 for the MNIST dataset, the VGG-16, and ResNet-20 models for the CIFAR-10 dataset, and the VGG-19 and ResNet-50 models for the ImageNet dataset. These models are tested on the four adversarial test input generation approaches like BIM, JSMA, FGSM, and CW, and one DL testing method like DeepGauge to validate the efficiency of the suggested approach. The simulation results proved that the proposed approach obtained high coverage accuracy for each criterion on four adversarial test inputs and one DL testing method as compared to other models.

Keywords: Deep learning, Coverage criteria, Optimization algorithm, Adversarial example, Software testing

Introduction

For a past few decades, a DL has obtained boundless victory in several safety–critical applications like speech recognition, image processing, and board games [1]. The DNN-based software could still have faults like traditional software, in spite of taking attained maximum testing accuracy as established by combative assaults. Specifically for those useful in mission and safety–critical situations, the significance of superiority and safety assurance of DL starts to draw attention [2]. In the DL system, the main leading concept

is to examine the reason for vulnerability by the way of creating adversarial test samples for the video and image-based DL schemes. Such sensibly erudite pixel-level disquiets invisible to the human eyes which can reason the DL-based taxonomy structure to outcome entirely erroneous choices along with high confidence [3]. Nowadays, the different study has been devoted to constructing the robust assailants as the beginning of adversarial assaults on the DL schemes. As a result, superior defense mechanisms are required in DL structures in contrast to adversarial assaults. In the current investigation, the numerous methods to train a large amount of robust DL structure and to invalidate the adversarial assaults [4]. The investigation in both empires creates a good circle and illuminations a trajectory for the construction of more common and vigorous DL structures.

The exhaustive coverage analysis has a tractable solution because the traditional programs are deterministic [5]. The normal methods for model testing are used to collect practical test information but the DNNs are data-driven. Such datasets are considered as costly and time-consuming process because they are labeled manually in a crowdsourced manner. Besides, the different DNNs classification boundaries are different according to their complication remark the statistics in a different way [6]. Hence, based on the model intricacy and the architecture details, the test data generation, and input data space are explored. Else, the structure may not be a true representation of practical application and the coverage of the model would be incomplete. In order to deliver shareholders with data about the quality of the services, software manufacturing be dependent on testing [7]. To measure the software at dissimilar levels, the investigation in the testing of software has stemmed in a wide-ranging of methods. The construction of a database is subjugated to produce test cases in white-box testing. In order to measure the extensiveness of a test collection, the encryption analysis criteria have been intended [8].

In software testing, the descriptive rules play a major role in choosing test input data. These rules have estimated the functionality of an AUT (application under test). For instance, the instruction for an insurance application is that if the clients have more preceding insurance fraud principles and deadbolt locks that are not installed on their locations, that particular client will posture a maximum insurance risk [9]. Along with a maximum insurance record, a high amount of resources are included in the assurance premium for a consumer. By the immaculate record, it equals this rule as opposed to a consumer. However, performing different computationally luxurious transactions is involved in the high-risk consumer record in contrast to a database [10]. For instance, the over-generalized rule demonstrates the concept. To make the efficient performance error illuminating test cases, the valuable descriptive rules frequently permit testers even though practical structures reveal much more complex behavior [11]. This intuition directed the investigators to consider the different ways which industrialize the testing role with intelligence in the optimization and test case assortment. By collecting the data about the software being investigated, the superiority assertion is maintained by the intellectual software testing actions. It points out the requirement for a search-based optimization procedure therefore the resources can be successfully employed [12].

The different test coverage criteria have been described with these test input generation procedures. In neuron coverage [13], conditions to detention exact neuron initiation

values to estimate the corner circumstances and a criterion which is stimulated by MC/DC [14]. Generally, the input of a DNN is high dimensional that creates an arbitrary testing complex. Using the symbolic implementation, the quantity of the execution path is very large in DNN to be entirely enclosed due to the extensive usage of the ReLU activation function for hidden neurons [15]. From the DL output, the software quality is estimated which is artificial in the logic in which basic sympathetic of the DL interior neuron actions and system behaviors are not moved. A set of testing criteria is required according to the DL decision output to solve the above-mentioned drawbacks. The developed criteria will have to observe and measure the actions of neurons and essential system connectivity at different granularity stages.

The major contribution of the paper as follows

In this paper, the optimized DBN with SAR approach is introduced for the testing of coverage criteria. For enhancing the coverage accuracy, the proposed DBN with SAR is used. The lack of robustness and complexity in the DL process reduces the accuracy of software testing. In order to reduce this kind of complexity and robustness, the SAR algorithm is introduced in DBN for optimal weight calculation. This optimized DBN with SAR reduces the error function presented in a DBN and improves the accuracy of testing. The proposed DBN is tested for different criteria such as KMNC, NBC, SNAC, TKNC, and TKNP. The coverage accuracy for each criterion is enhanced for the proposed method. For the coverage testing, the three standard datasets like MNIST, CIFAR-10, and ImageNet are used. For analysis, the developed approach is compared with the three LeNet models like LeNet-1, LeNet-4, and LeNet-5 for the MNIST dataset, the VGG-16, and ResNet-20 models for the CIFAR-10 dataset, and the VGG-19 and ResNet-50 models for the ImageNet dataset. These models are tested on four adversarial test input generation methods like JSMA,BIM,FGSM, and CW, and one DL testing method like DeepGauge to validate the efficiency of the suggested approach. The results proved that the suggested approach provides goodoutcomes than other models.

The remaining of the paper is systematized as trails: "Related works" section provide the recent related works, the overall proposed methodology is explained in the third section which include the DBN structure with optimization approach and the different coverage criteria for testing of DL systems. Simulation results are describe in the fourth section. Finally, the conclusion and future scopes are discussed in the fifth section.

Related works

Recent related papers are listed below

For DL structures, a tomographic combinatorial testing (CT) was developed by Lei Ma et al. [16]. In classic software testing, CT was the most dominant and suitable method. CT concentrates on the interactions of input testing instead of deeply analyzing all the groupings of input space. The key objective was to lessen the dimension of the test suite by gaining satisfactory defect discovery capabilities. A set of CT criteria was discussed on DL structures and also presented the CT coverage-guided test generation approach. Simulation outcomes illustrations that CT offers a hopeful opportunity for DL structures testing.

An important test coverage criteria were developed by Youcheng et al. [17] for DNN. A group of four new test coverage criteria was suggested which motivated by the MC/DC coverage criterion. These are custom-made to operational features of DNN and their semantics. By means of indicating the test inputs, the criteria were authenticated which are created using the suggested criteria with the direction that are intelligent to confinement undesired behaviors in a DNN. With the help of gradient-based heuristic and symbolic methods, the test cases were built. The suggested criteria accomplish steadiness among their capability to discover bugs and the computational costs of the test input creation. The experiments were implemented using MNIST, ImageNet, and CIFAR-10 datasets on the current DNNs.

For a structure of DNN, a coverage-guided fuzz testing scheme was presented by Xiaofei et al. [18]. To identify the faults of DNNs, a coverage-guided fuzz testing structure was presented. For creating the new semantically well-maintained tests, a metamorphic mutation approach was discussed. Further, to direct the test creation, a numerous extensible coverage criterion was controlled as feedback. Moreover, a seed selection approach was discussed that merges the recency-based and diversity-based seed selection. In DeepHunter, five current testing criteria and four seed selection approaches were executed. The simulation outcomes show that the metamorphic mutation approach was suitable for creating the novel legal tests as the unique speed with similar semantics. To enhance the coverage and identify the faults, the diversity-based seed selection approaches were commonly utilized than the recency-based method. As related to other methods, a DeepHunter provides better outcomes in terms of quantity, coverage, and defects diversity discovered. The DeepHunter method was more helpful to capture imperfections in the course of DNN quantization for platform relocation.

For DL structures, a multi-granularity testing criterion was developed by Lei et al. [19]. In this study, DeepGauge was presented for DL structures to execute a multi-faceted depiction of the testbed. The detailed assessment of the suggested testing measures was validated using two datasets, five DL structures, and four existing adversarial attack approaches in contrast to DL. The prospective worth of proposedshelters on the additional common structures and effective DL structures. The suggested criteria enable the thoughtful of DNN and the quality of test information from various stages and angles. Usually, discovery imperfections could possibly allocate to the corner-case regions and major function regions of DNNs. The suggested criteria could cover the major region and the corner cases of the neurons for a specified set of inputs. Here, defects of DL could suffer.

For DL structures, a coverage guided differential adversarial testing was presented by Jianmin et al. [20]. To direct the DL structures showing unfitting behaviors, this study presented a DLFuzz structure. The suggested method saves closely altering the input to enhance the coverage of neuron and the forecast variance among the mutated and the new input, without considering physical labeling effort from other structures with similar functionality. To enhance the neuron coverage, numerous new approaches were presented for a selection of neurons. DLFuzz provides the unfitted behaviors and it was applied for retraining, thus enhance the models' reliability. The suggested approach was validated using the two standard datasets and it was related to DeepXplore, the existing DL white-box testing structure. For cross-referencing form, DLFuzz could not need any additional efforts

Table 1 Symbols and Descriptions

Symbols	Descriptions
g_h	The amount of hidden units
g_v	The amount of visible units
W	Simultaneous weights among hidden and visible units
L	Simultaneous weights among visible and visible units
J	Simultaneous weights among hidden and hidden units
v	Visible unit
h	Hidden unit
W_{ij}	Symmetric connection among hidden (j) and visible (i) unit
b_j and a_i	Bias terms
Z	Normalization constant
m	Amount of training data samples
X_{il}	$i\ th$ unit of the $l\ th$ data occasion
ε	Learning rate
$g(x) = 1/(1 + \exp(-x))$	Logistic sigmoid function
D	Entire amount of data samples
O_z^e	Expected output
Z_z^e	Predicted output
M_n	Location of the $n\ th$ stored clue
$N = \{n_1, n_2, ...\}$	Set of neurons
$T = \{x_1, x_2, ...\}$	Test inputs
L_i	Arrangement of neurons on the i^{th} layer
X_i	Preceding location
T	Chosen features
X	Weight
Low_n	Lower boundary output values for a neuron n
$High_n$	Upper boundary output values for a neuron n

to discover identical functional DL structures. At last, DLFuzz methodology improves the accuracy of DL structure while utilizing these adversarial inputs to rehabilitate.

From the literature, existing DNN testing methods need extremely costly human exertion to offer accurate actions for a goal task. For compound and high-dimensional actual inputs, human beings, frequently have trouble in effectively executing a job properly for a huge database. No one of the current testing methods even attempts to shelter various instructions. Hence, the test ideas frequently flop to expose various invalid DNN behaviors. In this paper, an optimized DBN with SAR is introduced for the testing coverage criteria. The main gain of the suggested technique is to enhance the coverage accuracy for all the coverage criteria. DBN with SAR is introduced as the proposed scheme to test the different coverage criteria. Also, the proposed scheme measures the testing ability of the DBN structure. The proposed system obtained good accuracy performance on the test data.

Deep belief network architecture

The following symbols are utilized in order to the convenience of reading. Table 1 displays the symbols and its explanations.

Numerous layers of RBMs (restricted Boltzmann machines) and MLPs (multilayer perceptron's) are included in the DBN and it is a portion of DNN. The RBM model is shown in

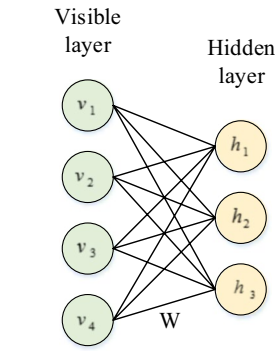

Fig. 1 RBM model

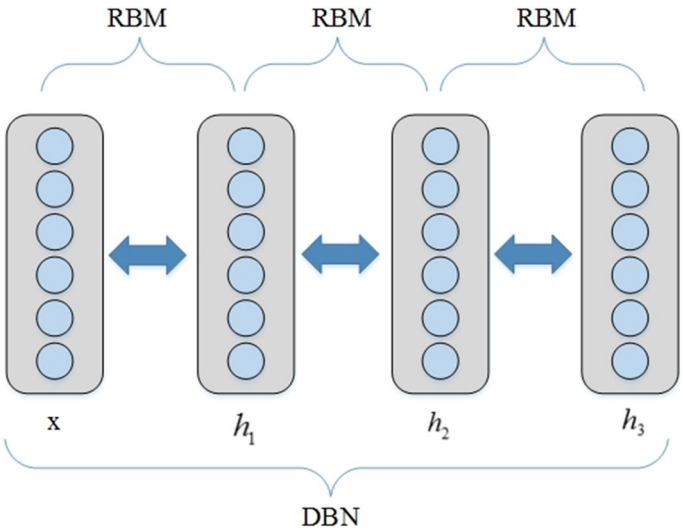

Fig. 2 DBN with RBM

Fig. 1. The visible and hidden units are included in the RBMs which are connected according to the weighted links. The input, hidden and output layers are included in the MLPs and these are deliberated as the feed-forward networks [21]. The undirected graphical design is also called as the BM. The DBN connected with RBM structure is shown in Fig. 2.

Along with stochastic binary units, the Boltzmann machine is also termed as the concurrent system. The set of visible $v \in \{0, 1\}^{g_v}$ and hidden $h \in \{0, 1\}^{g_h}$ units are included in the system. Here, the amount of hidden and visible units are represented as g_h and g_v individually. In Boltzmann machine, the joint configuration $\{v, h\}$ energy can be given as:

$$E(v, h) = -\frac{1}{2}v^T Lv - \frac{1}{2}h^T Jh - v^T Wh \tag{1}$$

For the straightforwardness of performance, the bias is eliminated. The simultaneous weights among hidden and visible units are represented as W. Also, the simultaneous weights among visible and visible units are denoted as L. Then the simultaneous weights among hidden and hidden units are denoted as J. But the crosswise values of L and J are

zero. The complex concept and designs are involved in the Boltzmann machines. For this reason, the RBM model is utilized for straightforwardness. The popular RBM model is presented if $J = 0$ and $L = 0$. In RBM, by the way of including bias, the mutual pattern $\{v, h\}$ energy can be specified as:

$$E(v, h) = -v^T Wh - a^T v - b^T h = -\sum_{i=1}^{g_v}\sum_{j=1}^{g_h} W_{ij} v_i h_j - \sum_{i=1}^{g_v} a_i v_i - \sum_{j=1}^{g_h} b_j h_j \tag{2}$$

Here, the symmetric connection among hidden (j) and visible (i) unit is denoted as W_{ij}. The bias terms are denoted as b_j and a_i individually. In hidden and visible units, a likelihood value is allocated by the network to all state based on the energy function. The product of possibilities are described as the joint distribution. Therefore, for the possible functions the sum of energy is accomplished by including the energies. For the hidden and visible units, the joint likelihood distribution can be given as:

$$P(v, h) = \frac{1}{Z} \exp\left(-E(v, h)\right) \tag{3}$$

Here, the normalization constant is denoted as Z which is accomplished by adding all probable pairs of hidden and visible vectors.

$$Z = \sum_{v}\sum_{h} \exp\left(-E(v, h)\right) \tag{4}$$

By the way of disregarding out hidden vector, the likelihood allocated to a visible vector by the system is accomplished.

$$P(v) = \sum_{h} P(v, h) = \frac{1}{Z}\sum_{h} \exp\left(-E(v, h)\right) \tag{5}$$

By the way of altering the weights and biases, the likelihood of the system allocates to a training image which can be enhanced to the lower energy of that image [22]. Specifically, those images have less energy and the huge impact to the partition function is created. By the subsequent objective function, the better value for every component can be estimated.

$$\max imize_{\{w_{ij}, a_i, b_j\}} \frac{1}{m}\sum_{l=1}^{m} \log\left(\sum_{h} P\left(v^{(l)}, h^{(l)}\right)\right) \tag{6}$$

Here, the amount of training data samples are denoted as m. The major objective is to enhance the likelihood of training informations. Then the fractional derivate of the above-mentioned objective with regards to w_{ij} can be defined as:

$$\frac{\partial}{\partial w_{ij}}\left(\frac{1}{m}\sum_{l=1}^{m} \log\left(\sum_{h} P\left(v^{(l)}, h^{(l)}\right)\right)\right) = \frac{1}{m}\sum_{l=1}^{m}\sum_{h} X_{il} h_j P(h \mid v = x) - \sum_{v'}\sum_{h'} v_i' h_j' P\left(v', h'\right) \tag{7}$$

Here, the *ith* unit of the *lth* data occasion is denoted as X_{il}. The sum on the right hand side is inflexible but the quantity on the left hand side can be estimated accurately. With

respect to the weight, the derivative for log likelihood of a training vector can be given as:

$$-\frac{\partial \log P(v)}{\partial w_{ij}} = < v_i h_j >_{data} - < v_i h_j >_{\text{mod } el} \tag{8}$$

Here, to define the prospects, the angle brackets are employed under the distribution stated by the subscript. In the log likelihood of the training samples, this indications to a very simple learning instruction for executing stochastic sharpest gradient.

$$\Delta w_{ij} = \varepsilon\left(< v_i h_j >_{data} - < v_i h_j >_{\text{mod } el}\right) \tag{9}$$

Here, the learning rate is denoted as ε. The learning rules of the bias components can be given as:

$$\begin{aligned} \Delta a_i &= \varepsilon(< v_i >_{data} - < v_i >_{\text{mod } el}) \\ \Delta b_j &= \varepsilon\left(< h_j >_{data} - < h_j >_{\text{mod } el}\right) \end{aligned} \tag{10}$$

Subsequently there are no straight connection among hidden units in the RBM model. But these hidden units are autonomous for a specified visible units. The binary state h_j of all hidden unit j is fixed to 1 for a specified arbitrarily designated training image v. Its likelihood can be given as:

$$P(h_j = 1|v) = g\left(b_j + \sum_i v_i w_{ij}\right) \tag{11}$$

Here, the logistic sigmoid function is denoted as $g(x) = 1/(1 + \exp(-x))$. But $< v_i h_j >_{data}$ can be estimated simply. Meanwhile, there are no straight associates among visible units in an RBM. For a specified hidden vector, it is easy to compute the impartial model of the state for visible unit.

$$P(v_i = 1|h) = g\left(a_i + \sum_j h_j w_{ij}\right) \tag{12}$$

Even though, $< v_i h_j >_{\text{mod } el}$ estimation is very complex. By executing consecutive Gibbs sampling for anextensive period and originating from several arbitrary state of visible units, it can be accomplished. At last, CD (contrastive divergence) is employed because of the unfeasibility of this technique and huge run-times. Several advantages are included in the RBM and it has been extensively applied in the current decades particularly in DBNs. Figure 3 shows the training approach of DBN.

For high level RBM learning, the action values of its hidden units can be employed as the training data after the RBM has been learned. To obtain a numerous depiction of data, every RBM model is permitted in the sequence by the DBN. To fine tune the weights, the BP (back propagation) method can be utilized after pre-training in DBN. Pre-training supports simplification and to marginally alter the weights obtained through pre-training, the minimum amount of information in the data can be used.

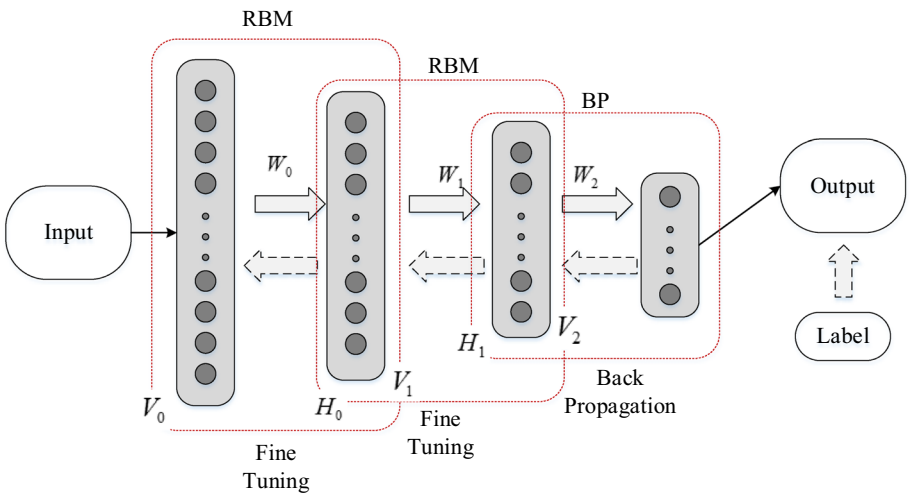

Fig. 3 Training procedures of DBN

Optimal weight updation of DBN using SAR

For the weight calculation, the SAR [23] algorithm is used in the DBN structure. The main objective of this work is to improve the coverage criterion for testing coverage of DBN structure. Using SAR, the optimal weights are accomplished in DBN. During the SAR operation, SAR is motivated by the explorations executed by humans. The locations of human is identical to the optimization issue solutions in SAR. For these solutions, the amount of clues are determined in these locations that denotes the impartial function. The DBN based SAR procedure of steps are presented below:

Initialization: In the arbitrary way, the DBN weights are initialized in the primary stage and it can be given as:

$$X = \{X_1, X_2, ..., X_g, ..., X_\alpha\};\ 1 < g \leq \alpha \tag{13}$$

Here, the entire weights are denoted as α.

Error estimation: To determine the output, the weight X and the chosen features T are applied to the DBN. The error of output is computed by the quantity of squares of present output of the system and the output of training label for network training and it can be given as:

$$Err^{e+1} = \frac{1}{D}\sum_{z=1}^{D}\left[O_z^e - Z_z^e\right] \tag{14}$$

Here, the entire amount of data samples are denoted as D, the expected output is denoted as O_z^e at present location and the predicted output is denoted as Z_z^e.

Weight updation: Using the SAR algorithm, the updated weight is computed. Based on the individual and social stages, the group members will search in every iteration. In an arbitrary location of the memory matrix (M), the preceding location (X_i) can be stored using Eq. (15) if the impartial function in location $X_i^{'}(f(X_i^{'}))$ is superior to the preceding

one ($f(X_i)$) after each stage. Then using Eq. (16) this position can be acknowledged as a new location. Or else, this location is left and the remembrance is not upgraded.

$$M_n = \begin{cases} X_i, & if\, f(X_i^{'}) > f(X_i) \\ M_n, & otherwise \end{cases} \tag{15}$$

$$X_i = \begin{cases} X_i^{'}, & if\, f(X_i^{'}) > f(X_i) \\ X_i, & otherwise \end{cases} \tag{16}$$

Here, the location of the *nth* kept clue is denoted as M_n in the memory matrix, the arbitrary number is denoted as n in the range of [1, N]. The algorithm diversity can be improved by this kind of memory updating procedure. The algorithm capability to determine the global optimum.

The misplacedindividuals may be damaged and the interruption of SAR groups may yield in their deaths. For this reason, time is the most significant factor in SAR operations. In the minimum probable time, the biggest space is examined and by this way SAR operations must be accomplished. He/she leaves the present location and goes to an original location if a human cannot discover superior clues around his/her present location after a particular amount of hunts. For every human being, USN (unsuccessful search number) is fixed to zero. If a humanoiddiscoveries superior clues in the initial and another stage of the hunt then the USN is 0 for that human or else, it can be improved by 1 point by the subsequent equation.

$$USN_i = \begin{cases} USN_i + 1, & if\, f(X_i^{'}) < f(X_i) \\ 0, & otherwise \end{cases} \tag{17}$$

Here, the human i has not been capable to determine the superior clues in the number of times is denoted as USN_i. In the hunt space, He/she drives to an arbitrary location by the subsequent equation if the USN is superior to the maximum unsuccessful search number (MU) for a human and USN_i is fixed to 0 for that human:

$$X_{i,j} = X_j^{\min} + r_4 \times \left(X_j^{\max} - X_j^{\min}\right), \quad j = 1, ..., D \tag{18}$$

Here, the arbitrary number is denoted as r_4 in the range of [0, 1]. For each dimension, this number is changed. Using Eq. (18), the weight value is assigned to the network if the estimated error is lower than that estimated error for the earlier instance. Or else, the USN count is improved. The optimal weight is obtained with minimum error value. Hence, the solution with the least value of the fault is selected as the optimal weight.

Determination of feasible weights: At last, based on the SAR algorithm, the weight value is updated using Eq. (18) in DBN structure.

Stopping criterion: In the iterative way, the optimal weights are computed until the maximum iteration is found.

Coverage criteria for testing of DL structures

To estimate the software internal states, coverage criteria is used fortesting. By this, the input space is separated and the connection of an input and an estimated software interior state are constructed. As of a distinct input subspace, the test information's is compared. Here, the identical amount of test data from numerous input subspaces provide the higher decision to cover additional assorted software states [19]. It provides the maximum likelihood to perceive the additional assorted software defects. In this work, the test coverage criteria is used from different levels for DBN structure. The major objective is to measure the testing capability of DBNs and enable the discovery of those incorrect behaviours from numerous depictions. The test criteria should be modest and accessible in the industry level applications.Also it is enough to utilize for a high range of DBNs without keeping on particular DBN scheme. Theoretically, the DBNs behaviours can be distributed into groups such as corner-case behaviours and major function behaviours. These two behaviours includes incorrect behaviours. For the strategy of coverage criteria, these factors are considered. Consider the set of neurons $N = \{n_1, n_2, ...\}$ in DBN structure. Assume $T = \{x_1, x_2, ...\}$ is the test inputs. In a specified test input $x \in T$, $\varphi(x, n)$ is employed to define a task that takings the neuron output $n \in N$. l layers are included in the DBN structure. The arrangement of neurons on the i^{th} layer ($i \leq i \leq l$) is denoted as L_i.

Neuron-level coverage criteria

The output values of neuron is employed at the neuron level which accomplished from the training to classify its behaviours. Using the training sample, the DBN internal structure is typically programmed. For every neuron of a DBN, more or less statistical distribution is trailed by the output of neuron. Using the training informations, these scattering is mostly accomplished.From the training informationsinspection, the output scattering of a neuron is accomplished and it permit to properly classify the major function areas. Along with an identical numerical distribution, the input data is activate the output values to the training data. The output values for corner cases rarely happen. For all neuron, the exact output distribution would be computationally demanding for a practical-sized DBN. To estimate the corner case region and major function region, the neuron output values are controlled which obtained from the training data. The lower and upper boundary output values are denoted as low_n and $high_n$ for a neuron n. From the training dataset analysis, these boundary values are obtained. The major function region is denoted as [low_n,$high_n$] for a neuron n.

Definition 1: A DBN is mounted in its major function region for a test input $x \in T$ that given x *iff* $\forall n \in N : \varphi(x, n) \in [low_n, high_n]$.

The lower and upper boundary values are divided into k sections to completely cover the major function regions. Also using the test inputs each of them to be covered. This coverage is called as KMNC.

1. k-multisection Neuron Coverage (KMNC)

The KMNC measures the specified arrangement of test inputs T which shelters the range [low_n,$high_n$] for a specified n. For $k > 0$, the range of [low_n,$high_n$] is distributed into k sections. For $i \leq i \leq k$, the set of values are denoted as S_i^n in the *ith* section. The test input x is cover the *ith* section when $\varphi(x,\ n) \in S_i^n$. The fraction of amount of segments sheltered by T and the entire amount of segments is known as KMNC for T and n. KMNC for a neuroncan be given as:

$$\frac{\left|\left\{S_i^n \middle| \exists x \in T : \varphi(x,\ n) \in S_i^n\right\}\right|}{k} \tag{19}$$

Also, KMNC of a DBN can be given as:

$$KMNCov(T,\ k) = \frac{\sum_{n \in N}\left|\left\{S_i^n \middle| \exists x \in T : \varphi(x,\ n) \in S_i^n\right\}\right|}{k \times |N|} \tag{20}$$

Here, two cases exists. One is $\varphi(x,\ n)$ may localize out of the boundary output values that means $\varphi(x,\ n) \in (-\infty,\ low_n)$ or $\varphi(x,\ n) \in \left(high_n,\ +\infty\right)$. Another one is corner case region for a neuron n that given $(-\infty,\ low_n) \cup \left(high_n,\ +\infty\right)$.

Definition 2: A DBN is mounted in its corner-case region for a specified test input $x \in T$ that assumed x *iff* $\exists n \in N : \varphi(x,\ n) \in (-\infty,\ low_n) \cup \left(high_n,\ +\infty\right)$.

Two coverage criteria such as NBC and SNAC are utilized to cover the corner-case regions for a DBN structure. The resultant corner-case region is covered for a specified trial input if $\varphi(x,\ n)$ be appropriate to $(-\infty,\ low_n)$ or $\left(high_n,\ +\infty\right)$. The amount of sheltered corner-case areas are separated as trails:

$$\begin{aligned} UpperCornerNeuron &= \left\{n \in N \middle| \exists x \in T : \varphi(x,\ n) \in (high_n,\ +\infty)\right\} \\ LowerCornerNeuron &= \left\{n \in N \middle| \exists x \in T : \varphi(x,\ n) \in (-\infty,\ low_n)\right\} \end{aligned} \tag{21}$$

2. Neuron boundary coverage (NBC)

   For a particular arrangement of test input T, several corner-case areas are protectedbased on the upper and lower border values which is defined by the NBC. The fraction of quantity of sheltered corner-cases and the entire amount of corner-cases $(2 \times |N|)$ are called as NBC.

$$NBCov(T) = \frac{|UpperCorne\ \ rNeuron| + |LowerCorne\ rNeuron|}{2 \times |N|} \tag{22}$$

3. Strong neuron activation coverage (SNAC)
   For a particular arrangement of test input T, the quantity of corner-cases are covered based on the upper boundary values which is defined by the SNAC. The proportion of the amount of protected upper-corner circumstancesand the entire amount of corner-cases $(|N|)$ are called as SNAC.

$$SNACov(T) = \frac{|UpperCornerNeuron|}{|N|} \tag{23}$$

Layer-level coverage criteria

To describe DBN behaviours, the top hyperactive neurons and their combinations are employed at layer level. The neurons n_1 and n_2 on the equivalent layer for a particular set of test input x. If $\varphi(x, n_1) > \varphi(x, n_2)$, n_1 is more active for a specified x than n_2. To describe the neurons, $top_k(x, i)$ is employed for the i^{th} layer. The neurons ensure the highest k outputs on that layer for a particular x.

1. Top-k neuron coverage (TKNC)

   It gauges neurons which act as the maximum active k neurons in all layer. The fraction of the entire amount of TKN on every layer and the entire amount of neurons in a DBN are called as TKNC.

   $$TKNCov(T, k) = \frac{\left|\cup_{x \in T}(\cup_{1 \le i \le l} top(x, i))\right|}{|N|} \tag{24}$$

   The neurons play a major task which accomplished from the identical layer of DBN. To describe the foremost functionality of a DBN, top active neurons are significant factors which obtained from numerous layers. An additional top active neurons are uncovered by the test dataset to more systematically test a DBN.
2. Top-k neuron patterns (TKNP)
   The arrangement of TKN makes a pattern on every layer for a specified arrangement of test input x. The amount of TKNP for the test input T can be given as:

   $$TKNPat(T, k) = \left|\left\{(top_k(x, l), ..., top_k(x, l))|x \in T\right\}\right| \tag{25}$$

   From the highesthyper neurons of all layer, the numerous types of activated situations are defined by the TKNP.

In this work, the coverage accuracy is obtained using the above-mentioned coverage criteria for the proposed approach. The proposed approach achieved more coverage's on each criteria using the three famous standard datasets like CIFAR-10,MNIST,and ImageNet. Accuracy only depends onhow many amounts of neurons covered for the proposed approach. The proposed approach maximum amount of neurons are covered for each criteria. The proposed approach tested on 4 adversarial test inputs and one DL system like DeepGauge. The coverage accuracy of the suggestedtechnique is equated with the different DNN models.

Simulation results and discussions

In this work, coverage criteria testing for the DBN with SAR algorithm is implemented on the Python 3.6 platform with Anaconda IDE. The implementation is processed on a windows machine using 3.20 GHz i3 CPU and 6G memory. For the testing of DBN structure, three well-standard datasets like MNIST, CIFAR-10and ImageNet are used.

MNIST [24] a huge digital handwritten dataset obtained from https://www.kaggle.com/oddrationale/mnist-in-csv?select=mnist_train.csvwhich contains images (pixels $28*28*1$) whose labels may range from 0 to 9. It contains totally 70,000 input data, among that 60,000 are utilized for training and 10,000 are utilized for testing. Another dataset is ImageNet [25] obtained from http://image-net.org/downloadwhich also contains three-channel image with pixel size $224*224*3$. It is a large dataset with 1000 diverse types of images. For training purpose, more than 1 million images are used and for testing 50000 images are utilized.CIFAR-10 [26] is the most well-known dataset obtained from https://www.cs.toronto.edu/~kriz/cifar.html. It is a collection of common classification images. This dataset contains $32*32*3$ pixel three-channel images and ten various types of pictures. In this dataset, 50,000 training samples and 10,000 test samples are involved. As compared to the MNIST, classification of CIFAR-10 dataset is more difficult due to the largequantity of information and high complexity. Hence, two obtain the better performances, two pre-trained models like VGG-16 and ResNet-20 are used. To measure the ability of the proposed structure, 4 existing adversarial test examples[27] such as FGSM, BIM, JSMA, and CW and 1corresponding DL testing method like DeepGauge [19] are considered for the comparative study.Five types of coverage criteria such as KMNC, NBC, SNAC, TKNC, and TKNP are used for the testing of DBN structure. For each datasets, coverage accuracy is accomplished using these coverage criteria's.

In the MNIST dataset, all images are distinct channel of dimension $28 \times 28 \times 1$. By the way of 60,000 training data, the DBN's neuron output is accomplished using the runtime profiling before starting the assessment process. Then 10,000 test data is execute to accomplish the equivalent analysis for DBN if assessment of testing originates. In this work, four existing adversarial test input schemes such as FGSM, BIM, JSMA, and CW and one DL testing scheme like Deep Gauge are employed for the testing of proposed system. Each test input is combined with the MNIST dataset that permit us to activate the qualified study. Defect detection capability is enhanced by the adversarial test inputs. The adversarial test input approaches are model-dependent.For each neuron during profiling, the lower and upper bounds are indicated as l and u, respectively; and the standard deviation is represented as σ. Due to the large image size ($224 \times 224 \times 3$) and data size, the ImageNet is more challenging for evaluation. Furthermore, the DBNs that obtain maximum accuracy is frequently complex. The 5,000 images are utilized as the test data for assessment. In the ImageNet, the computational complexity is reduced by the arbitrary sample images in the adversarial test inputs.

The KMNC coverage accuracy for MNIST dataset is shown in Fig. 4a. The proposed method obtained the high coverage accuracy for KMNC coverage criteria. The 4 adversarial test examples and one DL testing system were considered for the testing of the proposed and other DNN models. For the MNIST dataset, the three different DNN models such as LeNet1, LeNet4 and LeNet5 are considered. Three different models has lowest accuracy than the proposed method. In the proposed scheme, SAR approach is used to select the optimal value of DBN. KMNC coverage obtained more coverage accuracy in the DBN structure. KMNC coverage accuracy for the proposed scheme is 81% which is maximum than other approaches. For each test method like DeepGauge, FGSM, BIM, JSMA and CW, the KMNC coverage accuracy for LeNet1 is 64%, 75%, 68%, 78% and 73% individually. Also for LeNet4, the KMNC accuracy of existing methods such as

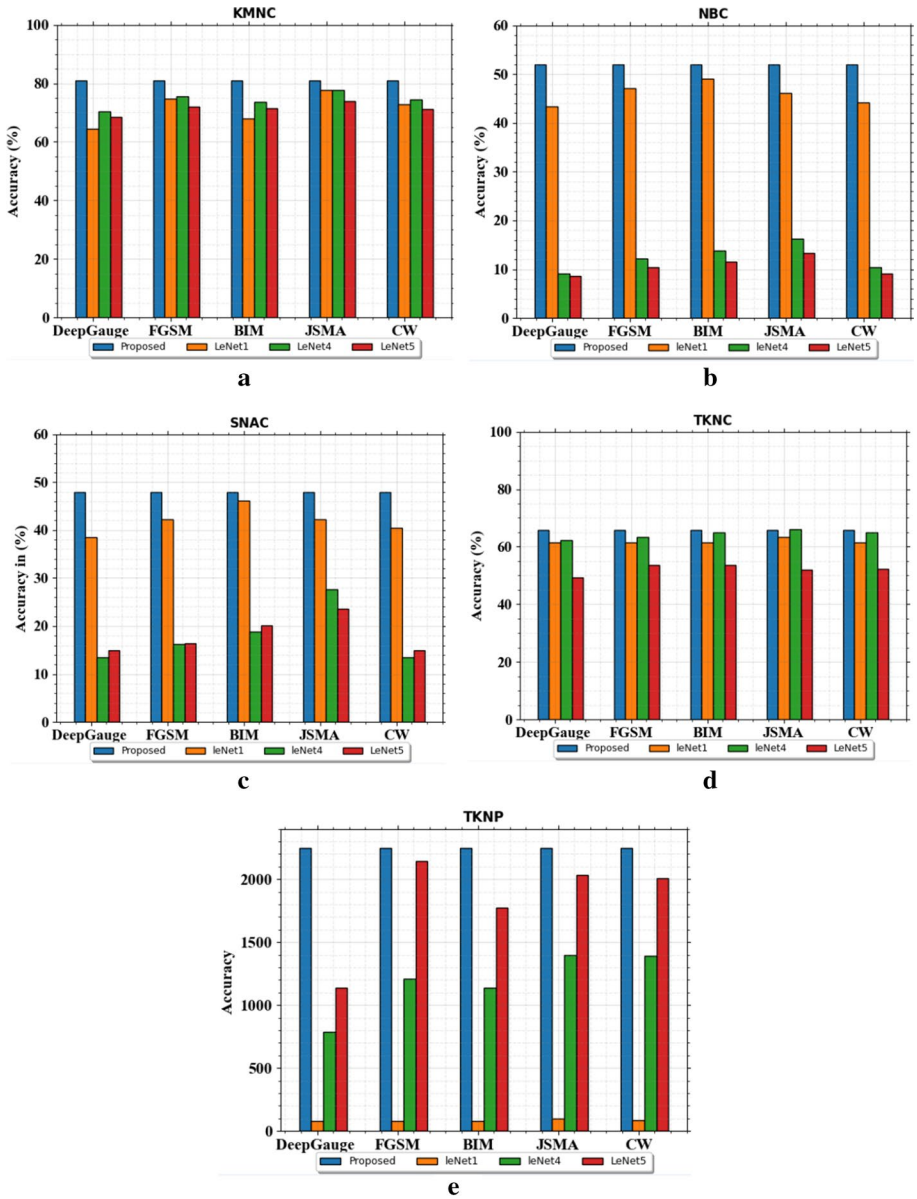

Fig. 4 Coverage accuracy on MNIST dataset (**a**) KMNC, (**b**) NBC, (**c**) SNAC, (**d**) TKNC and (**e**) TKNP

DeepGauge, FGSM, BIM, JSMA and CW is 70%, 75.5%, 74%, 78%, and 75% respectively. The KMNC accuracy for LeNet5 is 68%, 72%, 72%, 74% and 72% respectively. For the comparison of each model, the KMNC coverage improvement is minimum as compared to the proposed approach.For MNIST dataset, NBC coverage accuracy for the proposed and existing models is plotted as shown in Fig. 4b. The existing models are obtained low coverage accuracy for the NBC coverage than the proposed DBN approach. The proposed method obtained the highest (52%) coverage accuracy as compared to the all DNN models. In the existing methods, LeNet1 models obtained higher coverage than LiNet4 and LiNet5 models. SNAC coverage is improved in the proposed scheme using the MNIST dataset as shown in Fig. 4c. SNAC coverage improvement is 48% for

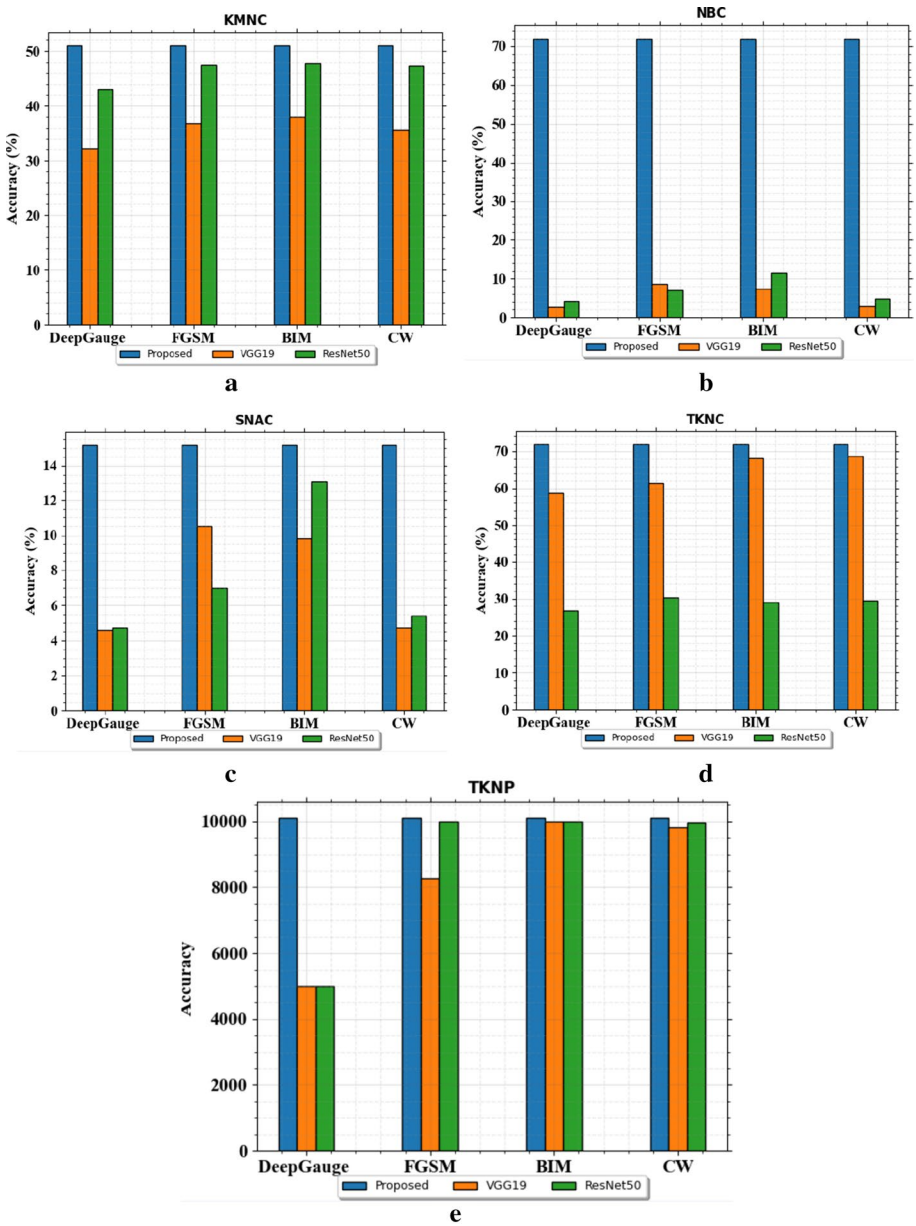

Fig. 5 Coverage accuracy on ImageNet dataset (**a**) KMNC, (**b**) NBC, (**c**) SNAC, (**d**) TKNC and (**e**) TKNP

proposed DBN as compared to other DNN models such as LeNet1, LeNet4 and LeNet5 on the FGSM, BIM, JSMA, CW and DeepGauge.For the MNIST dataset, TKNC and TKNP coverage criteria's are obtained for the proposed DBN and the existing models as shown in Fig. 4d, e. In such cases, adversarial test data still trigger more neurons which detect the hidden defects. TKNC accuracy is maximum in the proposed DBN method which is 63% and also TKNP coverage is maximum as related to other DNN structure of LeNet1, LeNet4 and LeNet5. In the TKNP coverage, LeNet5 obtained the maximum coverage which is less as compared to the proposed and maximum than other LeNet4 and LeNet1 models.

While showing some differences, some similarity is shared by the testing coverage on the ImageNet along with MNIST data. ResNet-50 and VGG-19 models have high complexity and large size, potentially provide the low coverage.For the ImageNet dataset, the coverage criteria of KMNC is accomplished for the proposed and existing methods as shown in Fig. 5a. The existing DNN models such as VGG19 and ResNet50 are tested on the FGSM, BIM, CW and DeepGauge. KMNC coverage is enhanced in the proposed DBN model than other structures. The proposed obtained highest (51.5%) coverage accuracy for ImageNet dataset as related to other models of VGG19 and ResNet50. As compared to ResNet50, the model of VGG19 provide the maximum coverage which is less than the proposed method.NBC and SNAC coverage for the ImageNet dataset is shown in Fig. 5b, c. For each coverage criteria, the proposed method obtained the highest coverage accuracy than other approaches. NBC and SNAC coverage accuracy is 12.1% and 14.7% for the suggestedprocess with the ImageNet dataset. The existing models of VGG19 and ResNet50 are achieved lowest coverage improvement for the ImageNet dataset. Fig. 5d, e shows the TKNC and TKNP coverage for the ImageNet dataset. The proposed DBN structure has maximum coverage accuracy for both TKNC and TKNP criteria's. The existing models of VGG19 and ResNet50 are obtained the low coverage accuracy for the test input generation approaches such as DeepGauge, FGSM, CW, and BIM. TKNP would frequently be capable to distinguish the input data if k is correctly designated for a specified target DBN. Creating test case to cover more TKNPand it have a maximum choice to discover faults of a DBN.

Figure 6 shows the different coverage accuracy on CIFAR-10 dataset using the proposed and the existing models. The several coverage criteria such as KMNC, NBC, SNAC, TKNC and TKNP are used to test the quality of the DBN structure. Here, the existing pre-trained DNN models of VGG-16 and ResNet-20 are used for the comparison process. The proposed scheme and the existing models are tested on the adversarial test input generation method like FGSM, BIM and CW, and also one more DL structure like DeepGauge. The proposed system covered the more amount of neurons on each criteria, hence, the proposed method provides better outcomes for each criterion than other models. Because, the proposed method utilized the SAR algorithm for the optimal weight selection. This optimized DBN with SAR approach reduce the learning complexity. Therefore, the proposed method obtained more coverage accuracy on CIFAR-10 dataset as compared to other DNN models like VGG-16 and ResNet-20.

Conclusion

In this work, an optimized DBN with SAR algorithm has been proposed for testing the numerouscoverage criteria. To effectively measure the testing accuracy, the different coverage criteria are discussed for the DBN structure. For an optimal weight selection of DBN, the SAR algorithm is used. Then five test coverage criteria are used for the testing of optimized DBN with SAR structure. The proposed coverage criteria for the DBN system enhances the coverage accuracy and the amount of potentially erroneous behaviours. The proposed test coverage criteria is executed by python tool with the help of threewell-standard datasets like CIFAR-10,MNIST, and ImageNet. The simulation

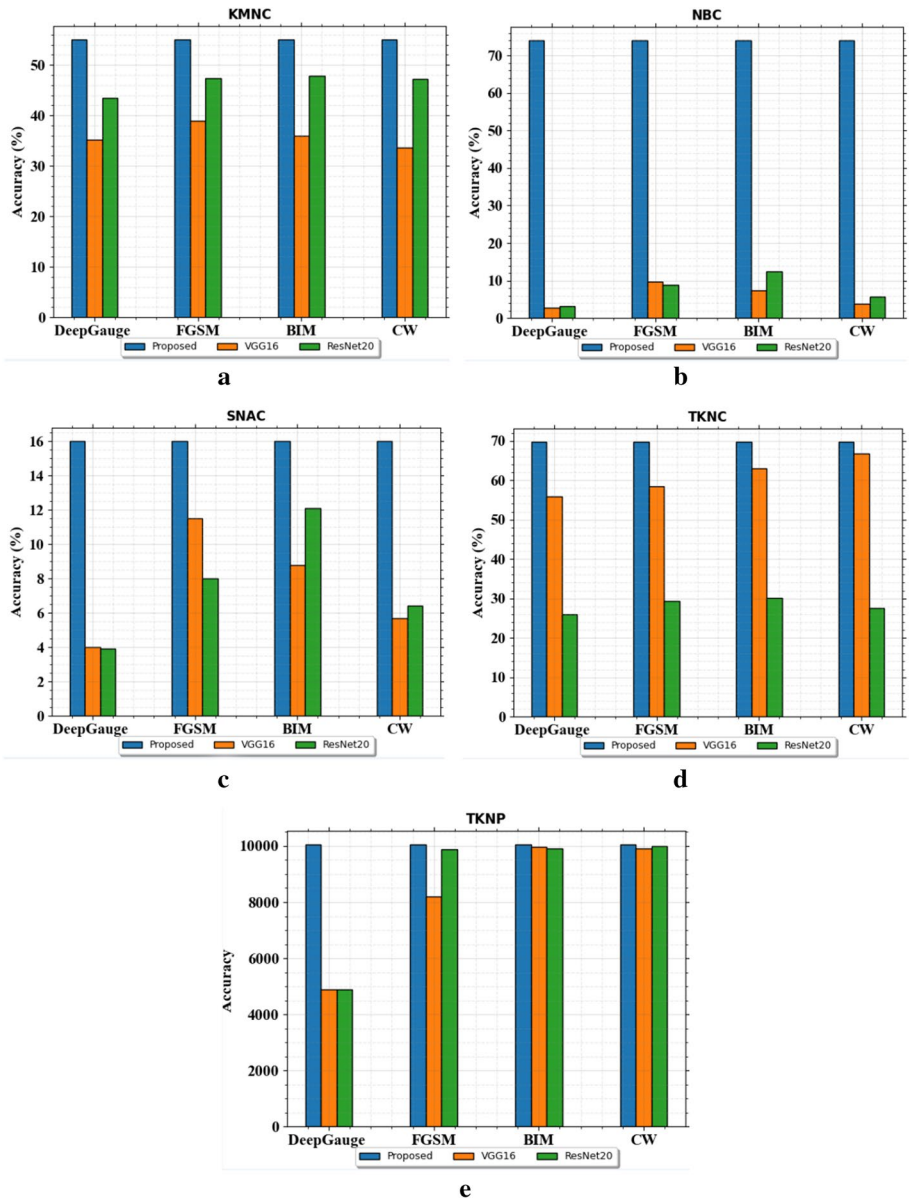

Fig. 6 Coverage accuracy on CIFAR-10 dataset (**a**) KMNC, (**b**) NBC, (**c**) SNAC, (**d**) TKNC and (**e**) TKNP

outcomes shows that the effectiveness of the proposed system in supporting DBN coverage, enhancing accuracy model and finding potential errors in DBN. In future, the coverage metrics may be introduced to offer extra intuitions for domain specialists while they are deliberating the capability of a specific datasets for an application usage.

Abbreviations

DL: Deep learning; DBN: Deep belief network; SAR: Search and resuce; MNIST: National Institute of Standards and Technology; DNN: Deep neural network; AUT: Application under test; CT: Combinatorial testing; RBMs: Restricted Boltzmann Machines; MLPs: Multilayer perceptrons; KMNC: K-multisection Neuron Coverage; NBC: Neuron boundary coverage; SNAC: Strong neuron activation coverage; TKNC: Top-k Neuron Coverage; TKNP: Top-k Neuron Patterns.

Acknowledgements
Not applicable.

Authors' contributions
KJ has found the proposed algorithms and obtained the datasets for the research and explored different methods discussed. PN contributed to the modification of study objectives and framework. Their rich experience wasinstrumental in improving our work. All authors contributed to the editing and proofreading. All authors read and approved the final manuscript.

Funding
Authors did not receive any funding for this study.

Availability of data and materials
The Datasets are not available in publicly so we collected the from MNIST Database.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no Competing interests.

Author details
[1] Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, A.P, India. [2] Departmentof Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, A.P, India.

Received: 15 December 2020 Accepted: 13 April 2021
Published online: 21 April 2021

References

1. Guo J, Jiang Y, Zhao Y, Chen Q, Sun J. Dlfuzz: Differential fuzzing testing of deep learning systems, In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM, 2018. pp. 739–743
2. Odena A, Olsson C, Andersen D, Goodfellow I. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. In: International Conference on Machine Learning. 2019, pp. 4901–4911.
3. Sun Y, Min W, Wenjie R, Xiaowei H, Marta K, Daniel K. Concolic testing for deep neural networks. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, 2018, pp. 109–119
4. Du X, Xiaofei X, Yi L, Lei M, Jianjun Z, Yang L. Deepcruiser: Automated guided testing for stateful deep learning systems. arXiv preprint arXiv:1812.05339. 2018.
5. Wieland B, Jonas R, Matthias B. Decision-Based adversarial attacks: reliable attacks against black-box machine learning models. In ICLR. 2018
6. Cihang X, Jianyu W, Zhishuai Z, Zhou R, Alan Y. Mitigating Adversarial Effects through Randomization. In ICLR. 2018.
7. Ma L, Fuyuan Z, Jiyuan S, Minhui X, Bo L, Juefei-Xu F, Xie C et al. Deepmutation: Mutation testing of deep learning systems. In: 2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE), IEEE, 2018. pp. 100–111
8. Luo Q, Nair A, Grechanik M, Poshyvanyk D. Forepost: Finding performance problems automatically with feedback-directed learning software testing. Empir Softw Eng. 2017;22(1):6–56.
9. Tian Y, Pei K, Jana S, Ray B. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In: Proceedings of the 40th international conference on software engineering, 2018. pp. 303–314.
10. Zhang P, Qiyin D, Patrizio P. CAGFuzz: coverage-guided adversarial generative fuzzing testing of deep learning systems." arXiv preprint arXiv:1911.07931. 2019.
11. Dong Y, Peixin Z, Jingyi W, Shuang L, Jun S, Jianye H, Xinyu W, Li W, Jin SD, Dai T. There is Limited Correlation between Coverage and Robustness for Deep Neural Networks. arXiv preprint arXiv:1911.05904. 2019.
12. Du X, Xie X, Li Y, Ma L, Liu Y, Zhao J. Deepstellar: Model-based quantitative analysis of stateful deep learning systems. In: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 477–487. 2019.
13. Kexin P, Yinzhi C, Junfeng Y, Suman J. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In: Proceedings of the 26th Symposium on Operating Systems Principles. ACM, 2017. p. 1–18.
14. Youcheng S, Xiaowei H, Daniel K. Testing Deep Neural Networks. arXiv preprint arXiv:1803.04792. 2018.
15. Lei M, Felix JX, Jiyuan S, Chunyang C, Ting S, Fuyuan Z, Minhui X, Bo L, Li L, Yang L et al. DeepGauge: Comprehensive and Multi-Granularity Testing Criteria for Gauging the Robustness of Deep Learning Systems. arXiv preprint arXiv:1803.07519. 2018.

16. Ma L, Juefei-Xu F, Xue M, Li B, Li L, Liu Y, Zhao J. Deepct: Tomographic combinatorial testing for deep learning systems. In: 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE. 2019. pp. 614–618
17. Sun Y, Huang X, Kroening D, Sharp J, Hill M, Ashmore R. Structural test coverage criteria for deep neural networks. ACM Transact Embedd Comput Syst. 2019;18(5s):1–23.
18. Xie X, Ma L, Juefei-Xu F, Xue M, Chen H, Liu Y, Zhao J, Li B, Yin J, See S. 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 2019. pp. 146–157.
19. Ma L, Juefei-Xu F, Zhang F, Sun J, Xue M, Li B, Chen C, Su T, Li L, Liu Y, Zhao J. Deepgauge: Multi-granularity testing criteria for deep learning systems. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering 2018. pp. 120–131.
20. Guo J, Zhao Y, Jiang Y, Song H. Coverage guided differential adversarial testing of deep learning systems. IEEE Transactions on Network Science and Engineering. 2020.
21. Hua Y, Junhai G, Hua Z. Deep belief networks and deep learning. In: Proceedings of 2015 International Conference on Intelligent Computing and Internet of Things, IEEE, 2015. pp. 1–4
22. Keyvanrad MA, Mohammad MH. A brief survey on deep belief networks and introducing a new object oriented toolbox (DeeBNet). arXiv preprint arXiv:1408.3264. 2014.
23. Shabani A, Asgarian B, Gharebaghi SA, Salido MA, Giret A. A new optimization algorithm based on search and rescue operations. Mathe Probl Eng. 2019. https://doi.org/10.1155/2019/2482543.
24. Deng L. The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Process Mag. 2012;29(6):141–2.
25. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, et al. Imagenet large scale visual recognition challenge. Int J Comput Vision. 2015;115(3):211–52.
26. Li H, Liu H, Ji X, Li G, Shi L. Cifar10-dvs: an event-stream dataset for object classification. Front Neurosci. 2017;11:309.
27. Yuan X, He P, Zhu Q, Li X. Adversarial examples: attacks and defenses for deep learning. IEEE Transact Neural Netw Learn Syst. 2019;30(9):2805–24.

Publisher's Note