

DeepVec: State-Vector Aware Test Case Selection for Enhancing Recurrent Neural Network

Zhonghao Jiang, Meng Yan, Li Huang, Weifeng Sun, Chao Liu, Song Sun, David Lo, *Fellow, IEEE*

Abstract—Deep Neural Networks (DNN) have realized significant achievements across various application domains. There is no doubt that testing and enhancing a pre-trained DNN that has been deployed in an application scenario is crucial, because it can reduce the failures of the DNN. DNN-driven software testing and enhancement require large amounts of labeled data. The high cost and inefficiency caused by the large volume of data of manual labeling, and the time consumption of testing all cases in real scenarios are unacceptable. Therefore, test case selection technologies are proposed to reduce the time cost by selecting and only labeling representative test cases without compromising testing performance. Test case selection based on neuron coverage (NC) or uncertainty metrics has achieved significant success in Convolutional Neural Networks (CNN) testing. However, it is challenging to transfer these methods to Recurrent Neural Networks (RNN), which excel at text tasks, due to the mismatch in model output formats and the reliance on image-specific characteristics. What's more, balancing the execution cost and performance of the algorithm is also indispensable.

In this paper, we propose a state-vector aware test case selection method for RNN models, namely DeepVec, which reduces the cost of data labeling and saves computing resources and balances the execution cost and performance. DeepVec selects data using uncertainty metric based on the norm of the output vector at each time step (i.e., state-vector), and similarity metric based on the direction angle of the state-vector. Because test cases with smaller state-vector norms often possess greater information entropy and similar changes of state-vector direction angle indicate similar RNN internal states. These metrics can be calculated with just a single inference, which gives it strong bug detection and model improvement capabilities. We evaluate DeepVec on five popular datasets, containing images and texts as well as commonly used 3 RNN classification models, and compare it with NC-based, uncertainty-based, and other black-box methods. Experimental results demonstrate that DeepVec achieves an average relative improvement of 12.5%-118.22% over baseline methods in selecting fault-revealing test cases with time costs reduced to only 1% to 1%. At the same time, we find that the absolute accuracy improvement after retraining outperforms baseline methods by 0.29%-24.01% when selecting 15% data to retrain.

Index Terms—Software testing, State-vector, Test case selection, Recurrent Neural Network.

I. INTRODUCTION

Zhonghao Jiang, Meng Yan, Li Huang, Weifeng Sun, Chao Liu are with the School of Big Data and Software Engineering, Chongqing University, China. E-mail: {zhonghaojiang, mengy, lee.h, weifeng.sun, liu.chao}@cqu.edu.cn.

Song Sun is with the School of Computer and Information Science, Chongqing Normal University, China. E-mail: 20220033@cqnu.edu.cn.

David Lo is with the School of Information Systems, Singapore Management University, Singapore. E-mail: davidlo@smu.edu.sg.

Meng Yan and Song Sun are corresponding authors.

DEEP Neural Networks (DNN) have currently been widely employed in numerous fields such as image classification [1], speech recognition [2], natural language processing [3]–[5], and autonomous driving [6]. Despite DNN-driven systems performing well in clearly defined tasks, there are still concerns about their reliability and quality. Abnormal behaviors in these systems may lead to misunderstandings, compromise personal safety [7], and even cause political conflicts [8], especially in safety-critical environments [9]. For example, nearly 400 crashes involving cars with DNN-driven autonomous driver systems were reported in the United States from July 2021 through May 2022 [10]. Therefore, it is important to test and enhance the DNN-driven systems.

Unlike traditional software, which is developed through explicitly defined business logic by programmers, systems powered by DNNs adhere to a data-driven programming model [11], resulting in distinct approaches to testing. Consequently, developers cannot improve the system by observing and adjusting its internal parameters directly, rendering traditional software quality assurance methods ineffective for DNN-driven systems. In practice, developers use extensive data for testing and retraining the DNN to correct model errors and enhance the system's performance [12]. Manually annotating a dataset is often expensive and time-consuming. For instance, nearly 9 years and 49,000 workers from 167 countries were needed to annotate ImageNet [13], a dataset with millions of images across 20,000 categories. In fact, only a small fraction of a large dataset triggers system errors [14]–[16]. Therefore, developers often select a set of representative test cases for labeling and training without compromising testing performance.

Current test case selection methods have achieved great success and can be categorized into three types: neuron coverage-based (NC-based) methods [14], [15], [17], [18], uncertainty-based methods [12], [19]–[21], and other black-box methods [22], [23]. NC-based methods select test cases by independently evaluating layer and neuron-level activations using Coverage-Total or Coverage-Additional strategies. Uncertainty-based methods select cases by calculating the model's output confidence [12], [20], [24], [25] and the similarity of input images through various approaches [19]. Additionally, some other black-box methods select a set of prototypes or criticisms using Maximum Mean Discrepancy (MMD) metrics [22]. However, these methods still face the following challenges: ① NC metrics predominantly focus on feed-forward networks (FNN) [26], including convolutional neural networks (CNN) [27] and fully connected networks (FC) [28], which leads to the neglect of RNN's internal behaviors when adapting to feedback networks [20], [29]–[32]. ② Uncertainty-based

methods proposed for CNNs are difficult to adapt to RNNs due to differences in output formats [20], [21]. Some of them even use image characteristics to construct metrics [19], [24], which lead to the failure for text data. Moreover, using clustering or heuristic search [25] involves high time costs and resource expenditures. ❸ Uncertainty-based methods designed for RNNs focus only on the transition of predicted labels over time steps, neglecting the probability information of each category [12], [33], which results in a loss of useful information [34], [35].

Addressing the aforementioned challenges, this study introduces DeepVec, a novel test case selection technique tailored for pinpointing potential fault-inducing test scenarios in systems powered by RNNs, balancing the execution cost and performance. DeepVec includes three phases: constructing state-vector space, measuring the uncertainty and similarity of test cases, and selecting test cases. First, we capture the probability of each category output by the RNN at each time step, called state-vector. Then, we design two metrics, i.e., the Distance Uncertainty and Changing Similarity to measure the uncertainty and similarity of test cases. Distance Uncertainty is based on the norms of the state-vectors at each time step, as state-vectors with smaller norms often correspond to larger information entropy [36], indicating lower classification confidence in RNNs. We derive Changing Similarity from the direction angle changes of state-vectors, as it is a quantitative measure obtained through sequence weighting, containing the time step order. Finally, we sort the candidate test set in descending order of Distance Uncertainty to sequentially select test cases. We prioritize selecting all test cases with different similarities to ensure that the selected data is representative. In summary, DeepVec ❶ overcomes the limitations of NC-based methods by constructing a state-vector space, ❷ addresses the limitations of image features by developing a more generalizable uncertainty and similarity measure based on a single inference, leading a low time cost, and ❸ mitigates the information loss caused by using predicted labels by applying output probabilities within the state-vector.

We evaluate DeepVec on five popular datasets and three RNN models including LSTM, BiLSTM and GRU, covering tasks in image classification and text classification following prior study [12], [20], [33]. To assess the effectiveness of DeepVec, we compare it with random selection, existing NC-based selection methods, uncertainty-based selection methods and other black-box methods. Experimental results demonstrate that DeepVec boosts bug detection rates by 12.5%-118.22% through effective test case selection, aiding in testing RNN models and identifying potential faults. Furthermore, we evaluate the effectiveness of DeepVec by calculating the inclusiveness of the selected tests. The concept of inclusiveness, as defined by Rothermel et al. [37], evaluates the capacity of regression test selection strategies to accurately identify tests that expose faults from a pool of candidates. Results across multiple selection rates indicate that DeepVec's ability to select fault-revealing tests of RNN models has improved by over 20% on the augmented test set. Moreover, we further evaluate its capability to enhance RNN quality. We select 15% of the data from the candidate set and retrain the original RNN model using the selected data and record the improvement in

accuracy. Experimental results show that the absolute accuracy improvement after retraining outperforms baselines methods by 0.29%-24.01%. In addition, our experiments also demonstrate that DeepVec can identify the most of fault types compared with baseline methods, while requiring only one percent to one ten-thousandth of the time cost. Meanwhile, the sensitivity analysis results indicate that DeepVec's performance is minimally affected by the angle threshold τ , and the τ we recommend is effective. In summary, the main contributions of this paper are as follows.

- **Approach.** We propose a test case selection method based on the RNN's state-vector space to construct the internal states and capture the uncertainty of RNN, which effectively aids in analyzing the confidence of the RNN model's current behavior. We propose Distance Uncertainty and Changing Similarity to reflect the behavioral states of the RNN hidden layers, which can make better use of the information of the state-vector.
- **Study.** We conducted extensive and comprehensive experiments on five datasets (including text classification and image classification) and three RNN models. The evaluation results of DeepVec demonstrate that the test cases selected by DeepVec exhibit excellent bug detection and fault-revealing capabilities. While ensuring diversity in fault detection, the runtime of DeepVec is only one percent to one ten-thousandth of that of existing methods. Additionally, DeepVec showed good robustness in sensitivity analysis, making it a valuable method for guiding RNN retraining, thereby enhancing its accuracy and resilience.
- **Open Science.** Leveraging the state-vector space inherent to RNNs, we have developed a test case selection framework named DeepVec. Its core function is to adeptly filter out a concise subset of test scenarios from an extensive, unlabeled dataset, ensuring a heightened likelihood of uncovering bugs. The source code and data of DeepVec is publicly available on <https://github.com/ZhonghaoJiang/DeepVec>.

II. PRELIMINARIES

In this section, we will introduce the fundamental knowledge of RNN and existing test criteria for RNN models.

A. Recurrent Neural Networks

The recurrent neural network (RNN) is a typical kind of feedback neural network [26]. RNNs are particularly adept at processing sequential data, including tasks like generating image captions [1], text production, and categorizing text [3]–[5], among others. As the RNN progressively unfolds, every neuron in the hidden layers of the RNN will iteratively update its weights and states for predictions as time steps progress [38], which is shown in Figure 1.

Each time step of RNN produces probabilities for each category i.e., state-vector, which also contains valuable information. Vanishing gradients and exploding gradients [39] are two common problems encountered in training RNNs. Gated mechanisms are one way to address these problems, as in Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU)

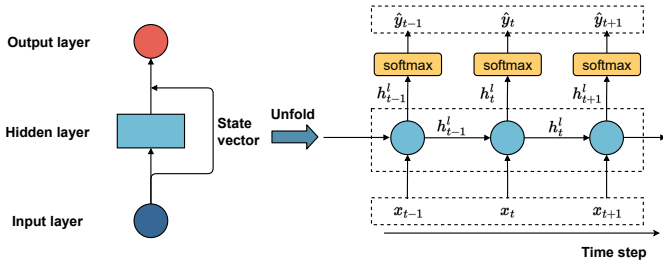


Figure 1. Unfold the state of RNN at each time step.

[39], where update gates and reset gates control the weight updates, achieving better capabilities in processing sequence data [40].

B. Neuron Coverage Criteria

In past research, researchers in the field of software engineering have proposed several neuron coverage criteria specifically for DNN testing, aiming to improve the quality and robustness of DNNs.

Neuron Coverage (NC). Neuron Coverage is initially introduced by Pei et al. [15]. It is defined as the ratio of the number of neurons exceeding the threshold k (activated) to the total number of neurons. The rate of $NC(k)$ for a test can be calculated as:

$$NC(k) = \frac{|activated\ neurons|}{|total\ neurons|}$$

Basic State Coverage (BSCov). BSCov is a quantitative metric proposed for RNN testing in DeepSteller [41]. It is proposed to quantify the extent to which the test inputs T encompass the principal functional region encountered during the training process. It can be calculated as:

$$BSCov(T, M) = \frac{|\hat{S}_T \cup \hat{S}_M|}{|\hat{S}_M|}$$

where \hat{S}_M , \hat{S}_T represent the abstract state of training inputs M , and the test inputs T .

Basic Transition Coverage (BTCov). BTCov is another metric introduced in DeepStellar [41]. BTCov is designed to assess the variations in abstract states within RNN models across the training and testing stages, denoted by $\hat{\delta}_T$ and $\hat{\delta}_M$, respectively. It is defined as:

$$BTCov(T, M) = \frac{|\hat{\delta}_T \cup \hat{\delta}_M|}{|\hat{\delta}_M|}$$

Step-wise Coverage (SC). Step-wise Coverage, as presented in testRNN [42], [43], is employed to analyze the temporal evolution of RNN hidden layer outputs. It characterizes the variation in the hidden vector between successive time steps in the following manner: It define the change of the hidden vector in adjacent time steps as follows:

$$\Delta \xi_t^h = |\Delta \xi_t^{h,+} - \Delta \xi_{t-1}^{h,+}| + |\Delta \xi_t^{h,-} - \Delta \xi_{t-1}^{h,-}|$$

where, $\Delta \xi_t^{h,+}$ represents the sum of all positive elements in the hidden state h at a given time step t , and $\Delta \xi_t^{h,-}$ captures the total of all negative elements within the same hidden state h at

time step t . Upon establishing a threshold v_{SC} , the calculation of SC proceeds as follows:

$$SC = \frac{|\{\Delta \xi_t^h \geq v_{SC} | t \in \{1, \dots, n\}\}|}{|\{\Delta \xi_t^h | t \in \{1, \dots, n\}\}|}$$

Hidden State Coverage (HSCov). HSCov is proposed in RNN-Test [44]. It measures the proportion of hidden states that reaches a maximum during the testing phase. Consider H as a four-dimensional matrix depicting the RNN's hidden state, with dimensions (T, L, B, E) . Here, T represents time steps, L layers, B batches, and E the number of hidden units. The HSCov is defined as:

$$HSCov = \frac{|\{e | \forall h \in H, e = \max(h)\}|}{|H|}$$

C. Uncertainty-Based Test Selection

In recent years, researchers have approached test case selection from a statistical perspective, focusing on the outputs of DNNs. Uncertainty-based methods prioritize test cases by measuring the confidence in the DNN's outputs and ranking the tests according to their uncertainty [12], [20], [21], [25], [45].

DeepState. Liu et al. [12] have proposed a uncertainty-based test case selection technique of RNN. DeepState designs changing rate (CR) and changing trend (CT) to measure RNN's uncertainty as follows:

$$CR(Seq(x, c)) = \frac{Count(y_{t+1} \neq y_t) \times t^2}{\sum t^2} \quad t = 1, 2, \dots, N$$

$$CT = \{(y_{t-1}, y_t) | y_t \text{ in } Seq(x, c), t = 1, 2, \dots, N\}$$

where x is the input test, c is the confidence threshold, $Seq(x, c)$ is a list of output labels at each time step t , represented as $\{y_1, y_2, \dots, y_N\}$. DeepState first sorts the test cases in descending order according to CR, and then calculates the Jaccard similarity between CTs to select test cases with different similarities in sequence.

DeepGini. DeepGini [20] utilizes the Gini coefficient to assess the purity of test inputs, which refers to the likelihood of correct classification. For a given test input x and its prediction probability $y = \langle y_1, y_2, \dots, y_M \rangle$ (where $\sum_{i=1}^M y_i = 1$), DeepGini identifies the most uncertain (i.e., least pure) data points by maximizing the following expression based on the output probabilities: $\arg\max_{x \in X} \left(1 - \sum_{i=1}^M y_i^2\right)$

ATS. ATS [21] introduces the concept of a fault pattern based on the uncertainty of unlabeled test inputs and fault direction. To describe the fault pattern of a test input, a set of intervals is used (for more details, refer to [21]). Additionally, ATS proposes a fitness metric to evaluate the pattern difference between a test input x and the selected set S . During the selection process, ATS first divides the candidate set into m subsets based on the prediction categories of the test inputs. Then, it selects the test inputs with the highest fitness value within each subset. If all test inputs in the candidate set have identical fitness scores, the test input with higher uncertainty is selected.

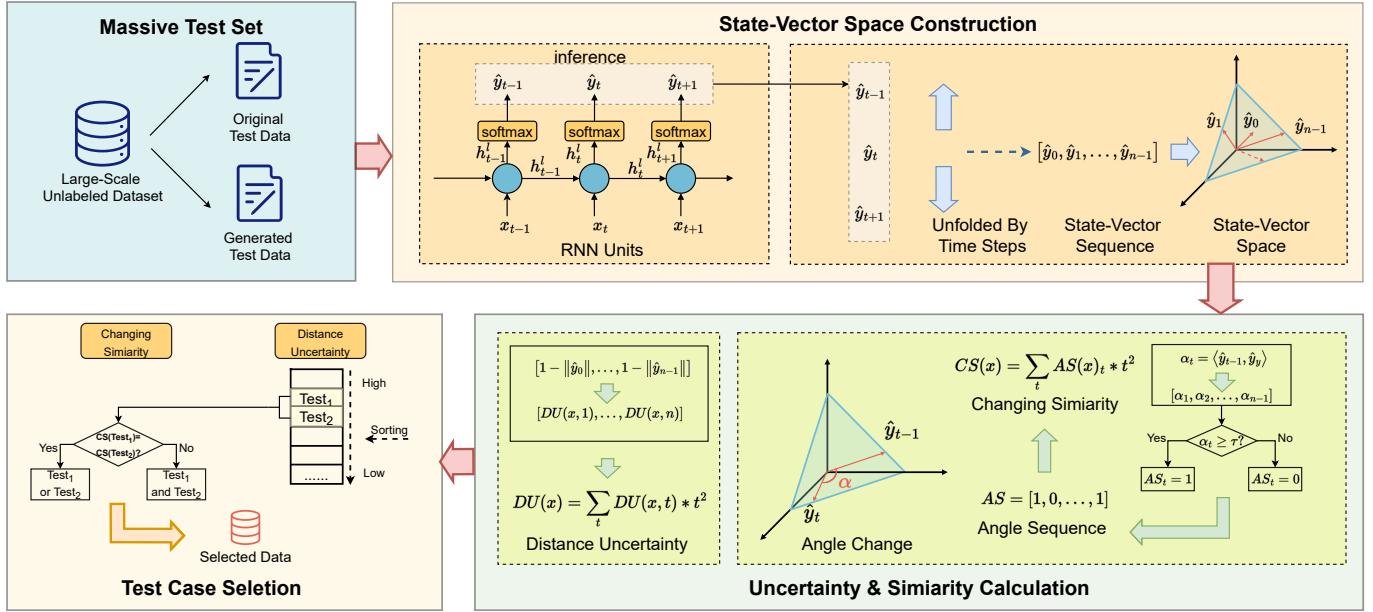


Figure 2. Overview of DeepVec.

III. APPROACH

In this section, we begin with the output vectors of the RNN at each time step, constructing the state-vector space of the RNN. Based on the space, we designed and implemented the DeepVec test case selection method. Initially, DeepVec expands the RNN model along its temporal dimensions, leveraging the state-vector space framework of the RNN outlined in Section III-A as shown in Figure 2. It then calculates the Distance Uncertainty and Changing Similarity for each test case as depicted in Section III-B. These metrics can be utilized to measure the uncertainty and similarity in RNN behavior when processing input data. We provide a detailed account of how DeepVec selects test cases in Section III-C. Lastly, Section III-D discusses how to enhance RNN models using DeepVec.

A. State-Vector Space Construction



RNNs excel at classifying sequential data by dynamically mapping high-dimensional features to a few numerical outputs over time. For instance, at a given time step t , the RNN could transform a complex hidden state vector into a probability distribution vector $\vec{p} = \langle 0.95, 0.03, 0.02 \rangle$ through the utilization of the softmax activation function. This means that the RNN model believes there is a 95% probability that the case belongs to category 0. During the inference, RNN produces an output vector at each time step, called state-vector. As time steps advance, state-vectors constitute a state-vector space, with the dimension of the vector space equal to the number of target categories in the RNN classification task.

Viewing through the lens of RNN state-vector space, it becomes feasible to quantify the model's uncertainty and delineate the dynamic patterns of hidden states across different time steps. In the vector space, the state-vector contains two important elements, namely the size and direction of the vector. Assuming that the RNN model classifies the data into N categories, then the state-vector can be expressed as

$\vec{p} = \langle p_0, p_1, \dots, p_N \rangle$. Due to the constraint of the output vector itself $\sum_{i=0}^N p_i = 1$, the endpoints of all state-vectors are confined to a hyperplane. For adjacent time steps t and $t+1$, the change in hidden state is seen as a scaling and deflection of the state-vector \hat{y}_t to vector \hat{y}_{t+1} . In the early time steps of RNN, because of the lack of training, the state-vector will deflect significantly within the confined space, and the confidence is low. With the accumulation of information, the RNN model approaches convergence, leading to heightened confidence in its predictions and a tendency for the state-vector's deviation to stabilize in subsequent time steps. Therefore, during the state transition process, the deflection angle and length variation of the vector become indicators carrying valid information.

To select test cases that might cause errors in the RNN model, we weigh situations from a vector space perspective that would lead to high uncertainty in RNN model predictions across all time steps. For instance, Table I shows an LSTM [39] model's intermediate prediction process during time steps 14-18 when classifying two similar images from the MNIST [46] dataset. The correct label for these two images is 5. However, at the 16th time step, the RNN mistakenly predicts the result of the second image (ID=2) as 7, while the first image (ID=1) is correctly predicted. By the 18th time step, both images are predicted correctly. Due to the MNIST is a dataset made up of 28×28 gray-scale images, each piece of data is divided into 28 time steps (i.e., a row of pixels from the image is input into the model at each time step). We capture the state-vectors for all time steps and choose the 14th-18th time steps to explore the uncertainty information contained in the vectors. We calculate the angle and norm of the vectors as they changed in the output space and analyzed the changes in the predicted labels. The state-vector's norm of the first image (ID=1) increases monotonically, with the angle of change decreasing gradually. This indicates that the model's confidence in the result is increasing and the model's confidence level is stable. Similarly, the state-vector's

Table I
EXPLANATION OF STATE-VECTOR SEQUENCE, TAKING TWO PICTURES IN MNIST AS AN EXAMPLE.

ID	Figure	State-Vector Sequence	Angle Changes	Norm Changes	Label Changes	Output
1		$[\langle 0.01, 0, 0.51, 0, 0.1, 0.16, 0.04, 0.02, 0.15, 0 \rangle,$ $\langle 0, 0, 0.28, 0, 0.05, 0.38, 0.03, 0.01, 0.26, 0 \rangle,$ $\langle 0, 0, 0.13, 0, 0.02, 0.55, 0.01, 0, 0.29, 0 \rangle,$ $\langle 0, 0, 0.03, 0, 0, 0.77, 0, 0, 0.19, 0 \rangle,$ $\langle 0, 0, 0, 0, 0, 0.97, 0, 0, 0.02, 0 \rangle]$	35.751° 20.810° 16.968° 12.861°	$0.566 \xrightarrow{-0.024} 0.542$ $0.542 \xrightarrow{0.093} 0.636$ $0.636 \xrightarrow{0.158} 0.794$ $0.794 \xrightarrow{0.177} 0.970$	$2 \rightarrow 5$ $5 \rightarrow 5$ $5 \rightarrow 5$ $5 \rightarrow 5$	5
2		$[\langle 0.23, 0.03, 0.4, 0.02, 0.01, 0.01, 0.11, 0.2, 0, 0 \rangle,$ $\langle 0.06, 0.01, 0.42, 0.01, 0.01, 0.01, 0.13, 0.34, 0.01, 0 \rangle,$ $\langle 0.01, 0, 0.35, 0.02, 0.01, 0.03, 0.06, 0.47, 0.05, 0 \rangle,$ $\langle 0, 0, 0.2, 0.04, 0.01, 0.12, 0.03, 0.35, 0.25, 0 \rangle,$ $\langle 0, 0, 0.05, 0.16, 0, 0.46, 0, 0.1, 0.22, 0 \rangle]$	23.530° 17.393° 29.645° 53.011°	$0.516 \xrightarrow{0.043} 0.559$ $0.559 \xrightarrow{0.033} 0.592$ $0.592 \xrightarrow{-0.100} 0.492$ $0.492 \xrightarrow{0.054} 0.546$	$2 \rightarrow 2$ $2 \rightarrow 7$ $7 \rightarrow 7$ $7 \rightarrow 5$	5

norm and angle of change of the second image (ID=2) always oscillate around a small value, indicating that the model is more uncertain about the classification result of the second image. Through the two mathematical metrics of the state-vector's norm and direction angle, we can clearly perceive the RNN's uncertainty.

B. Uncertainty and Similarity Calculation

After the construction of state-vector space inherent to RNNs through inference, we perform a detailed quantitative analysis of the RNN state-vectors across all temporal steps. From the norm and direction angle of the vector, we define Angle Sequence and Distance Uncertainty. Based on the Angle Sequence, we further derive the Changing Similarity analysis indicator. Distance Uncertainty quantifies the RNN model's indeterminacy for a specified input test. Angle Sequence is a boolean sequence that describes whether the RNN model can determine the predicted label has changed. Changing Similarity measures the consistency in the hidden state responses of the RNN across varied test scenarios.

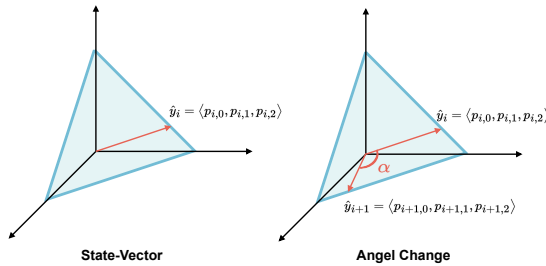


Figure 3. Angle Change in the state-vector for the three-category problem.

Definition 1 (Angle Sequence). *For a given input x , assuming that the RNN model will run for t time steps and will classify tests into n classes, the unfolded RNN model will yield a list containing t state-vectors $[\hat{y}_0, \hat{y}_1, \dots, \hat{y}_{t-1}]$, where \hat{y}_i is an n -dimensional vector. We calculate the Angle Change α between adjacent time steps of the state-vectors, which is shown in Figure 3, and set an adjustable hyperparameter τ as a threshold. When $\alpha > \tau$, we consider that the RNN model has produced a significant prediction change. For test case x , we define the Angle Sequence at each time step t as the following boolean sequence:*

$$\alpha_t = \arccos\left(\frac{\hat{y}_{t-1} \cdot \hat{y}_t}{\|\hat{y}_{t-1}\| \cdot \|\hat{y}_t\|}\right) \quad (1)$$

$$AS(x)_t = \begin{cases} 0, \alpha_t < \tau \\ 1, \alpha_t \geq \tau \end{cases} \quad t = 1, 2, \dots, n-1 \quad (2)$$

$$AS(x) = [AS(x)_1, AS(x)_2, \dots, AS(x)_{n-1}] \quad (3)$$

The Angle Sequence reflects the stability of the RNN model's output during its operation. The more positive elements in the Angle Sequence, the more times the state-vectors have undergone a drastic deviation, indicating that the output confidence of the RNN is unstable.

The determination of the threshold τ needs to be based on the geometric properties of the n -dimensional vector space, where the projection length of the prediction vector onto each coordinate axis corresponds to the prediction probability. Consider a binary classification task ($n = 2$), with prediction vectors $y_1 = \langle 0.3, 0.7 \rangle$ and $y_2 = \langle 0.6, 0.4 \rangle$, where the final predicted label will change from 1 to 0. It is easy to observe that the prediction label changes when the deviation of the prediction vector exceeds $\langle 0.5, 0.5 \rangle$. Therefore, the maximum deviation angle of the prediction vector that causes a label change is from $\langle 1, 0 \rangle$ to $\langle 0.5, 0.5 \rangle$ (i.e., 45°). Extending this conclusion to an n -dimensional vector space, it is evident that when the deviation angle of the prediction vector exceeds $\arccos\left(\frac{\mathbf{1}_{1 \times n} \cdot \mathbf{e}_{n \times 1}}{\|\mathbf{1}_{1 \times n}\| \cdot \|\mathbf{e}_{n \times 1}\|}\right)$, the predicted label is bound to change. Therefore, we recommend determining τ as $\arccos\left(\frac{\mathbf{1}_{1 \times n} \cdot \mathbf{e}_{n \times 1}}{\|\mathbf{1}_{1 \times n}\| \cdot \|\mathbf{e}_{n \times 1}\|}\right)$.

Definition 2 (Changing Similarity). *For the Angle Sequence AS of test x , the Changing Similarity is the weighted sum of each component of AS , reflecting the behavioral similarity between any two test cases during the progression of RNN time steps. We define Changing Similarity as follows:*

$$\begin{cases} CS(AS(x)) = \frac{AS(x)_t \cdot Weight(t)}{\sum Weight(t)} \\ Weight(t) = t^2 \end{cases} \quad t = 1, 2, \dots, n-1 \quad (4)$$

Considering that as time steps increase, the RNN will receive more valid information and have a sufficient basis for prediction, the components of AS in the later time steps should be more important. So we use a weighted sum instead of a simple sum. For example, for two input test cases x_1, x_2 , if $CS(AS(x_1)) = CS(AS(x_2))$, it implies that x_1 and x_2 have highly similar behavior within the RNN.

Definition 3 (Distance Uncertainty). *Assuming the RNN model will classify tests into n classes, when the RNN is unfolded, the*

state-vector \hat{y}_t at any time step should have the following form: $\hat{y}_t = \langle p_{t,0}, \dots, p_{t,n-1} \rangle$, where $p_{t,n-1}$ represents the probability that the RNN predicts x belongs to the $(n-1)$ th class at the t th time step. For a given test x , we define the Distance Uncertainty of the RNN at the t th time step as:

$$DU(x, t) = 1 - \sqrt{\sum_{i=0}^{n-1} p_{t,i}^2} \quad (5)$$

In the definition, we actually set the value of Distance Uncertainty as the difference between 1 and the magnitude of the state-vector. This is because any state-vector will satisfy the following two constraints:

$$\begin{cases} \sum_{i=1}^n p_{t,i} = 1 \\ \forall p_{t,i} \geq 0 \end{cases} \quad i = 1, 2, \dots, n \quad (6)$$

Under the above two constraints, we explore the extreme values of the norm of the state-vector. In vector space, we can abstract the problem into finding the minimum distance from the origin to the hyperplane $\sum_{i=0}^{n-1} p_i = 1$. Through rigorous mathematical proof [47], we find that the norm of the state-vector is minimized, and thus the Distance Uncertainty is maximized, if and only if the state-vector is collinear with $\langle 1, 1, \dots, 1 \rangle_{1 \times n}$. This aligns with our understanding that when the RNN model believes the input x has an equal probability of belonging to each class, the model is more prone to potential errors, indicating a higher uncertainty.

Consistent with Definition 2, we calculate the Distance Uncertainty value for a given test x using the same weighted approach. The revised DU is defined as follows:

$$\begin{cases} DU(x) = \frac{DU(x,t) * Weight(t)}{\sum Weight(t)} \\ Weight(t) = t^2 \end{cases} \quad t = 1, 2, \dots, n \quad (7)$$

C. Test Case Selection

DeepVec prioritizes test cases with high Distance Uncertainty values, particularly focusing on the RNN model's performance in the later time steps. In detail, DeepVec prioritizes each test case within an extensive, unlabeled test collection by its Distance Uncertainty metric. This measure reflects the RNN's ambiguity regarding a specific test, thereby indicating the potential for the model to exhibit faults. However, it is not adaptable to test sets with a large amount of duplicate data or with numerous test cases having the same DU value, as the selected test cases might not be representative. Therefore, DeepVec further selects test cases with different Changing Similarity values as representatives. Additionally, we consider two test cases with Changing Similarity values both equal to 0 to be unobservable rather than similar.

Algorithm 1 presents DeepVec's process for selecting test cases to test and enhance RNN models during the testing phase. Given an RNN model M and a large-scale unlabeled test set T : First, DeepVec lets the RNN iterate through all test cases during the inference phase, while calculating the Distance Uncertainty and Changing Similarity values corresponding to each data point (lines 1-9). Second, DeepVec sorts all tests in descending order based on their DU values (Line 10). Third,

Algorithm 1 DeepVec Selection Algorithm

Input: $T = [t_0, t_1, \dots, t_n]$ ▷ The original unlabeled dataset
Input: M ▷ The RNN model to be tested
Input: k ▷ The number of test cases to select
Output: *Selected* ▷ The selected test cases

```

1:  $i = 0$ ;
2:  $\tau = setThreshold()$ ;
3: while  $i < n + 1$  do ▷ Perform RNN inference
4:    $t_i.VecSeq = getOutputVectorSequence(t_i, M)$ ;
5:    $t_i.AS = getAngleSequence(t_i.VecSeq, \tau)$ ;
6:    $t_i.CS = getChangingSimilarity(t_i.AS)$ ;
7:    $t_i.DU = getDistanceUncertainty(t_i.VecSeq)$ ;
8:    $i = i + 1$ ;
9: end while
10:  $T = reverseSorted(T, T.DU)$ ;
11:  $CS\_Set = \{\}$ ;
12:  $j = 0$ ;
13: while  $j < n$  and  $|Selected| < k$  do
14:   if  $t_j.CS \notin CS\_Set$  or  $t_j.CS == 0$  then
15:      $Selected.add(t_j)$ ;
16:      $CS\_Set.add(t_j.CS)$ ;
17:   end if
18:    $j = j + 1$ ;
19: end while
20:  $j = 0$ ;
21: while  $|Selected| < k$  do
22:   if  $t_j \notin Selected$  then
23:      $Selected.add(t_j)$ ;
24:   end if
25:    $j = j + 1$ ;
26: end while
27: return Selected;

```

for the reordered test set T , DeepVec traverses the test set again and sequentially selects all test cases where CS is different or CS is zero (Line 11-19). If the final number of selected test cases is insufficient, DeepVec selects additional test cases based on DU, from highest to lowest, until the required number is reached (Line 20-27).

Table II
AN EXAMPLE TO SHOW METRIC CALCULATIONS IN DEEPVEC

Test	x	Threshold	$\tau = \arccos\left(\frac{(1,0,0,0) \cdot (1,1,1,1)}{\ (1,0,0,0)\ \cdot \ (1,1,1,1)\ }\right) = 60^\circ$
Time Step (t)	State-Vector at Time Step t	Distance Uncertainty $DU(x, t)$	Angle Change (α)
1	(0.1, 0.2, 0.3, 0.4)	0.4523	44.85° < τ
2	(0.1, 0.1, 0.7, 0.1)	0.2789	68.30° > τ
3	(0.1, 0.5, 0.1, 0.3)	0.4	58.98° < τ
4	(0, 0, 0.1, 0.9)	0.0945	82.19° > τ
5	(0, 0.8, 0.1, 0.1)	0.1876	
$DU(x) = \frac{0.4523 \cdot 1^2 + 0.2789 \cdot 2^2 + 0.4 \cdot 3^2 + 0.0945 \cdot 4^2 + 0.1876 \cdot 5^2}{1^2 + 2^2 + 3^2 + 4^2 + 5^2} = 0.0267$			
$AS(x) = [0, 1, 0, 1]$			
$CS(AS(x)) = \frac{0 \cdot 1^2 + 1 \cdot 2^2 + 0 \cdot 3^2 + 1 \cdot 4^2}{1^2 + 2^2 + 3^2 + 4^2} = 0.6667$			

A running example. Assuming an RNN with 5 time steps is designed to classify tests into 4 classes, Table II illustrates how to compute the Distance Uncertainty and Changing Similarity for test case x . We assume that the threshold is 60° ($\tau = 60^\circ$). First, we acquire the state-vectors from the model for all time steps. Next, we compute the Distance Uncertainty components

for the state-vectors at each time step. We then calculate the angle changes between adjacent time step state-vectors to obtain the angle sequence. Following this, for each change in angle, we assess its relation to the threshold τ to further derive the Angle Sequence. The angle changes of x in the second and fourth instances are greater than τ , we set $AS(x) = [0, 1, 0, 1]$. Finally, we calculate x 's Distance Uncertainty and Changing Similarity using a weighted method. Suppose we have seven test

Table III
THE PROCESS OF SELECTING TEST CASES BY USING DEEPVEC.

Test	DU(Test)	AS(Test)	CS(AS(Test))
A	0.8	[1, 1, 1, 0]	0.4667
B	0.6	[1, 1, 1, 0]	0.4667
C	0.6	[0, 0, 0, 0]	0
D	0.5	[0, 0, 0, 0]	0
E	0.4	[1, 0, 1, 0]	0.3333
F	0.3	[1, 0, 0, 0]	0.0333
G	0.2	[1, 0, 0, 0]	0.0333

cases $\{A, B, C, D, E, F, G\}$. Table III provides their Distance Uncertainty and Changing Similarity values. DeepVec will sort all test cases in descending order based on DU and then iterate through the test set sequentially. DeepVec starts by selecting test case A . It then compares the CS of A and B . As $CS(AS(A)) = CS(AS(B))$ and is not equal to 0, DeepVec considers the behaviors of A and B within the RNN to be similar and thus does not select B . As for C and D , since both of their CS values are 0, DeepVec determines that it cannot judge whether their behaviors are similar or not. Consequently, DeepVec selects both C and D .

To encapsulate, considering the instance illustrated in Table III, DeepVec identifies A, C, D, E, F as the chosen test cases. Should the need arise for more than five test scenarios, DeepVec will sequentially incorporate B and G into the selection.

D. Enhancing RNN with DeepVec

In real-world application scenarios, the accuracy and performance of RNN-driven software can be improved by periodically collecting genuine data from application contexts and retraining the RNN. DeepVec measures the uncertainty of RNN outputs from a statistical view in a state-vector space with limited constraints, where a smaller norm of the state-vector corresponds to higher entropy. DeepVec abstracts change of state-vector into significance levels, allowing for more accurate identification of test cases with similar behaviors. This enhances the ability to select test cases, ensuring the accuracy of the RNN is improved in the shortest possible time. Our observations indicate that the test cases curated by DeepVec outperform those selected through coverage-based methods and the uncertainty-driven technique, Deepstate, in improving RNN performance.

IV. EXPERIMENT SETUP

We evaluate DeepVec through extensive experiments. In this section, we will clarify the experimental setup and research questions. Section IV-A presents the models and datasets for evaluating DeepVec. Section IV-B describes the generation process of the test cases that to be selected. We compare the performance of DeepVec with the methods introduced in Section IV-C, and evaluate the results using the metrics mentioned in Section IV-D.

A. Datasets and RNN Models

As shown in Table IV, we evaluate DeepVec on two major tasks: text classification and image classification. We independently evaluate three RNN models on three image datasets and two text datasets by following prior study [12], [20], [33].

Table IV
DATASETS AND RNN MODELS.

Datasets	Models	Category	Type	Description	ACC _{ori}	ACC _{aug}	ACC' _{aug}
Mnist	LSTM	10	Image	Handwritten numbers 0-9	97.92	31.44	92.99
	GRU				98.4	28.36	92.53
	BiLSTM				98.28	30.44	92.79
Fashion	LSTM	10	Image	Zalando's article images	86.99	48.07	80.98
	GRU				86.81	49.89	81.89
	BiLSTM				82.17	47.79	80.93
Snips	LSTM	7	Text	Intent recognition	84.39	73.98	88.27
	GRU				72.54	64.33	86.07
	BiLSTM				58.57	52.08	78.66
Agnews	LSTM	4	Text	News article texts	79.46	74.95	82.31
	GRU				81.4	77.08	83.18
	BiLSTM				78.67	73.63	82.64
SVHN	LSTM	10	Image	Street View door number	78.72	18.84	61.24
	GRU				78.13	23.68	57.33
	BiLSTM				75.65	17.52	53.42

1. ACC_{ori}: Accuracy of the model trained on original set and test on original set.
2. ACC_{aug}: Accuracy of the model trained on original set and test on augmented set.
3. ACC'_{aug}: Accuracy of the model trained on original and augmented set and test on augmented set.

MNIST [46] is widely used in handwritten digit recognition. It is divided into a training set of 60,000 images and a test set of 10,000 images. Fashion [48] is a set of article images from Zalando. Its division is consistent with MNIST. Snips [49] is a natural language dataset containing over 16,000 crowd-sourced queries. AgNews [50], [51] is a set of news that can be divided into four categories. It is divided into a training set of 120,000 news and a test set of 7,600 news. SVHN [52] is a real-world image dataset, similar to MNIST, but with an order of magnitude more labeled data (over 600,000 digit images). It is collected from house numbers in Google Street View images.

The LSTM [39] is the most commonly used variant of RNNs. It employs three gates and a cell state aimed at addressing the vanishing and exploding gradient problems, thus being capable of capturing dependencies over longer time spans. The GRU [53] is a simplification of the LSTM. With its streamlined reset and update gate [54] structure, it addresses the long-term dependency issues inherent in RNNs. The Bidirectional LSTM (BiLSTM) [55] extends the LSTM [56], allowing it to process sequences from both directions simultaneously, capturing relationships in both the preceding and following contexts more effectively.

To simulate DNN mispredictions triggered in real-world scenarios, we do not use augmented data in initial training of the DNN. Instead, we use the augmented data to generate the candidate set for selection and retraining because selecting test that can trigger DNN faults is the purpose of DeepVec. Initially training the DNN without using perturbed data ensures the acquisition of more trigger tests. The evaluation of the model's accuracy under initial training on the original test set and the augmented test set is shown in Table IV.

B. Augmented Test Data Generation

In order to simulate the test cases that may be collected in actual application scenarios. We randomly use several

methods to add perturbations to the original data to generate candidate test sets [12], [20]. For image classification, we chose Keras ImageDataGenerator [57] for perturbation, and for text classification we used an open source tool nlpaug [58]. For each data in the original test set, we randomly perform contrast adjustments, brightness changes, translation, rotation, scaling, and cropping on images [59], [60], and synonymous replacement, back-translation, word insertion, and summarization on text [61]–[63] to obtain augmented data. It is worth noting that the amplitude of the perturbation is controlled at 5% to prevent changes in the labels of the perturbed data set.

C. Baseline Methods

We compare DeepVec with the neuron coverage-based methods, uncertainty-based methods and 2 additional black box testing methods.

1) *Coverage-Based Methods*: Among the coverage-based strategies for test case selection, the Coverage-Total Method (CTM) and the Coverage-Additional Method (CAM) stand out as prevalent options [64].

Coverage-Total Method (CTM). CTM, acknowledged for its 'next best' approach, prioritizes choosing test cases that offer maximum coverage from the pool of candidates, without regard to the tests already selected.

Coverage-Additional Method (CAM). CAM employs a selection strategy guided by an additional greedy algorithm. It dynamically adjusts the next chosen test according to the already selected tests, consistently targeting the test that can encompass the majority of uncovered code structures within the candidate set.

For coverage-based methods, we arrange and combine the CAM and CTM ranking strategies with the coverage metrics introduced in Section II-B, and use them all as baseline methods. Nevertheless, certain metrics are incompatible with simultaneous application of both CAM and CTM approaches. Specifically, BSCov and BTCov (DeepSteller) [41] are unsuitable for CAM, since they abstract the hidden layer output states into a model and calculate its coverage. Conversely, HSCov (RNNTest) [44] does not align with CTM, as it merely identifies the activation of specific hidden neurons in the RNN for each input.

2) *Uncertainty-Based Methods*: For uncertainty-based methods, we select the three the state-of-the-art methods mentioned in Section II-C. We apply the metric proposed by DeepState [12] to sort the test cases, and then select the test cases in order according to their similarity. Since DeepGini [20] is originally proposed to address issues based on CNNs, we adapt DeepGini for use on RNNs by applying the weighted method from DeepVec (i.e. weight the output vector at each time step by t^2). Similar to DeepGini, we apply the same weighting method to process ATS [21].

In addition, we also consider comparisons with RTS [19], DeepGD [25], and PRIMA [24]. However, since these methods utilize certain image-specific characteristics, such as P-hash and Geometric Diversity, they cannot be adapted to text datasets. Additionally, these methods have already been compared with

ATS [21] or DeepGini [20], so we did not include them as baselines in our study.

3) *Other Methods*: **K-Medoids**. K-Medoids [22] is utilized as a method for selecting a representative set of prototypes from the dataset. In this approach, we use the K-Medoids algorithm to choose a number of prototypes that matches the number of samples selected by the proposed method. The K-Medoids clustering technique is applied to partition the data into clusters, and from each cluster, the most centrally located data point (medoid) is selected as the prototype. This ensures that the selected prototypes are representative of the diverse patterns present in the dataset.

MMD-Critic. MMD-Critic [23] is employed as a method for selecting both prototypes and criticisms from the dataset. In this approach, we use the MMD-Critic algorithm to select a number of prototypes and criticisms that correspond to the number of samples chosen by the proposed method. The MMD-Critic framework first identifies prototypes that represent the central tendencies within the data, and then it selects criticisms, which are the data points that deviate from these prototypes, highlighting the diversity and the outlying aspects of the data. This combination of prototypes and criticisms offers a balanced representation of the dataset, making it an effective baseline for comparison with the proposed approach.

Random Select. To provide a benchmark for comparison, we incorporate a baseline method that involves the random selection of a specified percentage of test cases from the pool of candidates.

D. Research Questions

To evaluate DeepVec's efficiency, we formulate the following research questions (RQs) for a comprehensive performance evaluation. Figure 4 clarifies the composition of the candidate datasets and test sets used for each research question. The candidate set is used during the tests selection phase to simulate the data obtained from real-world applications. The selected data is used for retraining and evaluation metric calculation. The original test set and the augmented test set are used to assess the accuracy changes of the DNN after retraining.

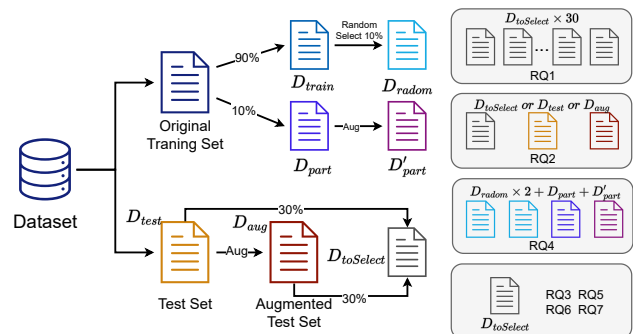


Figure 4. Overview of the candidate dataset construction used for each research question.

RQ1. Effectiveness: Can test cases selected by DeepVec detect more bugs than other baseline methods?

DeepVec is a test case selection technique for enhancing RNN models. It aims to select a part of the data from a

large unlabeled test cases. Analogous to conventional software testing, the response to RQ1 can be determined by assessing the bug detection efficacy of test cases chosen by DeepVec, alongside those selected by other foundational approaches.

To provide a candidate test set, we extract 30% of the data randomly, from both the original test set and the augmented test set described in section IV-B and combine them into a new mixed set. Therefore, the size of the candidate set is 60% of the original test set size. To mitigate potential biases caused by random sampling, we repeat the above process 30 times, constructing thirty mixed sets D_1, D_2, \dots, D_{30} . We apply DeepVec and all baseline methods to all test sets. At the selection ratios $r=\{10\%, 20\%\}$, We separately calculate and show the bug detection rate. Using D_i to represent the candidate set and $D_{i,bug}$ to represent the test cases caused the model to produce incorrect output (i.e., bug test cases) in D_i , the bug detection rate calculation method is as follows:

$$Bug\ Detection\ Rate(D_i) = \frac{|D_{i,bug}|}{|D_i|} \times 100\% \quad (8)$$

RQ2. Inclusiveness: Can DeepVec effectively select as many fault-revealing test cases as possible from the candidate test data set?

In RQ2, we focus on whether DeepVec can comprehensively select fault-revealing test cases in the candidate set, that is, the ability of DeepVec to reveal the potential defects. Similar to recall, a widely recognized evaluation metric, we draw on inclusiveness proposed by Rothermel et al. [37]. Although inclusiveness is first proposed to measure the ability of regression testing methods to select wrong test cases, we can still reasonably transfer it to evaluate RNN test case selection techniques. We independently used the original test set, the augmented test set, and a dataset from RQ1 as candidate sets, calculate inclusiveness separately under multiple selection rates. Specifically, the calculation formula of inclusiveness is as follows:

$$Inclusiveness = \frac{|D_{select,bug}|}{|D_{test,bug}|} \times 100\% \quad (9)$$

where, $D_{test,bug}$ and $D_{select,bug}$ respectively represent the number of test cases in the selected test cases that can cause RNN failures and the number of test cases in the candidate test set that cause RNN to generate bugs.

RQ3. Fault Diversity: Do the test cases selected by DeepVec cover a variety of fault types?

In traditional software testing, Chan et al. [65] point out that the distribution of fault-revealing tests is usually dense and close. Therefore, a comprehensive evaluation of DeepVec not only aims to assess its ability to detect as many bugs as possible but also focuses on its capability to select diverse test cases. Gao et al. [21] propose a method for measuring the diversity of test cases.

For a given test case x being misclassified, its fault type is defined as:

$$Fault\ Type(x) = (Label(x)^* \rightarrow Label(x)) \quad (10)$$

where $Label(x)^*$ denotes the ground-truth label, and $Label(x)$ denotes the prediction label calculated by DNN model. It is

worth noting that the *Fault Type* in the dataset is related to the target categories of the classification task. For example, in a dataset D_{test} with n classes, the theoretical number of fault type it contains is $D_{test,type} = n \times (n - 1)$. Assuming the number of fault types covered by the selected test cases is $D_{select,type}$, the *Fault Diversity* can be calculated using the following equation:

$$Fault\ Diversity = \frac{|D_{select,type}|}{|D_{test,type}|} * 100\% \quad (11)$$

RQ4. Guidance: Can DeepVec enhance the accuracy of RNN models by guiding the retraining?

In principle, test cases exhibiting a heightened rate of bug detection are theoretically poised to enhance the optimization of RNN models. In order to verify this, we conduct RQ3 to assess the feasibility of utilizing the selected data for retraining.

We construct a candidate dataset D_{select} through combining augmented and original training data to answer RQ3 [12], [33]. To maintain variety in D_{select} , we reserve a random subset of 10% from D_{train} , denoted as D_{part} , and exclude it from the initial training phase. Subsequently, we produce the enhanced dataset D'_{part} derived from D_{part} . Thus, D_{select} is a combination of D'_{part} , D_{part} , and a portion of D_{train} , with the augmented D'_{part} making up 25% of D_{select} .

Using each selection method, 15% of data from D_{select} are used to improve the model following prior study [12], [33]. To make sure accuracy improvements are not due to dataset irregularities, the accuracy increases of the model are confirmed with both the combined test set D_{mix} (which includes D'_{test} augmented data and original D_{test}) and the augmented test set D'_{test} . To ensure the generalizability of the experimental results, we repeat the experiments 10 times and report the average improvement in accuracy on both original test set and augmented test set.

RQ5. Time Cost: How much time does DeepVec take to select test cases?

In real-world business scenarios, it is crucial to reduce the cost incurred during the test case selection phase while also minimizing the resources consumed by DNN retraining. Therefore, we design this research question to evaluate the time cost associated with DeepVec. We repeat the selection process 30 times using one of the datasets from RQ2 with all selection methods and record the time consumed in each selection to ensure the generalizability of the experimental results.

RQ6. Ablation Study: How do Distance Uncertainty and Changing Similarity contribute to DeepVec?

To comprehensively understand the contribution of Distance Uncertainty (DU) and Changing Similarity (CS) to DeepVec, we conduct two sets of ablation experiments. These experiments evaluate the performance of DeepVec on various datasets and models without using DU or CS. When DU is not used, we first randomly sort the test cases in the candidate set and then sequentially filter them using CS. When CS is not used, we sort the candidate set based on DU and then decide whether to remove the next test case with a 50% probability. Finally, we evaluate the Bug Detection Rate and Inclusiveness of DeepVec, DeepVec w/o DU and DeepVec w/o CS separately.



Figure 5. Comparative analysis of the bug detection rates achieved by various selection techniques, based on 10% of test cases chosen for 30 separate trials.

RQ7. Sensitive Analysis: How does the choice of angle threshold τ affect DeepVec performance?

In the similarity measurement phase of DeepVec, we set an angle threshold τ to determine whether the state-vector has undergone a significant change. However, since the number of target categories varies across different classification tasks, the setting of the angle threshold τ may significantly impact the performance of DeepVec.

To explore the performance differences of DeepVec across different datasets and models under various angle thresholds, as well as its sensitivity to the threshold τ , we used the same data as in RQ2 as the candidate set. We apply DeepVec to select 10% and 20% of the test cases with $\tau = \{15, 30, 45, 60, 75, 90\}$ respectively, and evaluate the differences in their *Bug Detection Rate*, *Inclusiveness* and *Fault Diversity*. Furthermore, to verify the relationship between the number of target categories in a task and the reasonableness of our recommended angle threshold $\tau_{\text{given}} =$

$\arccos(\frac{\mathbf{1}_{1 \times n} \cdot \mathbf{e}_{n \times 1}}{\|\mathbf{1}_{1 \times n}\| \cdot \|\mathbf{e}_{n \times 1}\|})$, we observe whether the experimental results exhibit significant differences when $\tau > \tau_{\text{given}}$.

RQ8. Generalizability: How does DeepVec perform on DNNs in practical scenarios?

Before DNNs are deployed in practice, the training set is often perturbed and mixed to use. To investigate whether DeepVec can further improve DNNs in practical scenarios, we apply the perturbation method mentioned in Section IV-B to augment the training set, and then used the expanded training set to train the original DNN, evaluating by following assessment method in RQ4.

V. RESULT ANALYSIS

In this section, we provide all of the experimental results to prove the performance of DeepVec. DeepVec is developed in Python 3.7.16 [66], utilizing Keras [67] with TensorFlow 1.14.0 [68]. The experimental setup is hosted on an Ubuntu 20.04.1 LTS server, equipped with 1 Nvidia A800 GPU, a 96-core CPU, and 512GB RAM.

Table V

THE AVERAGE BUG DETECTION RATE OF 10% AND 20% SELECTED TEST CASES. VALUES HIGHLIGHTED IN RED AND BLUE INDICATE THE BEST AND SECOND BEST.

	Model	Sel	Ran.	HSCov (CAM)	BSCov (CTM)	BTCov (CTM)	SC (CTM)	SC (CAM)	NC (CTM)	NC (CAM)	K- Medoids	MMD Critic	Deep Gini	ATS	Deep State	Deep Vec
Minst	LSTM	10%	0.36	0.52	0.13	0.26	0.55	0.36	0.27	0.37	0.57	0.37	0.80	0.75	0.70	0.82
		20%	0.36	0.44	0.14	0.22	0.46	0.36	0.35	0.37	0.48	0.37	0.76	0.62	0.62	0.79
	GRU	10%	0.37	0.56	0.15	0.48	0.32	0.36	0.51	0.38	0.54	0.38	0.62	0.69	0.66	0.45
		20%	0.36	0.49	0.15	0.31	0.32	0.36	0.41	0.37	0.47	0.37	0.58	0.50	0.59	0.61
	BiLSTM	10%	0.37	0.50	0.16	0.27	0.39	0.37	0.43	0.45	0.56	0.38	0.84	0.69	0.68	0.81
		20%	0.37	0.43	0.20	0.26	0.39	0.37	0.43	0.40	0.48	0.37	0.77	0.48	0.58	0.75
Fashion	LSTM	10%	0.38	0.45	0.26	0.25	0.27	0.38	0.34	0.38	0.48	0.39	0.69	0.70	0.62	0.75
		20%	0.37	0.41	0.30	0.30	0.28	0.38	0.35	0.38	0.49	0.38	0.61	0.56	0.55	0.68
	GRU	10%	0.38	0.40	0.30	0.29	0.36	0.39	0.23	0.37	0.48	0.39	0.57	0.69	0.61	0.70
		20%	0.38	0.39	0.30	0.30	0.34	0.38	0.29	0.38	0.49	0.39	0.50	0.53	0.55	0.63
	BiLSTM	10%	0.40	0.55	0.27	0.31	0.41	0.40	0.41	0.41	0.51	0.41	0.68	0.69	0.66	0.74
		20%	0.40	0.47	0.30	0.32	0.41	0.40	0.35	0.40	0.51	0.41	0.60	0.56	0.62	0.66
Snips	LSTM	10%	0.25	0.29	0.24	0.23	0.25	0.26	0.25	0.26	0.23	0.21	0.48	0.48	0.45	0.45
		20%	0.26	0.27	0.24	0.23	0.25	0.25	0.25	0.26	0.22	0.23	0.43	0.38	0.40	0.46
	GRU	10%	0.35	0.36	0.31	0.27	0.32	0.35	0.31	0.35	0.34	0.32	0.60	0.55	0.53	0.59
		20%	0.34	0.35	0.32	0.30	0.32	0.35	0.32	0.35	0.32	0.33	0.51	0.46	0.49	0.56
	BiLSTM	10%	0.19	0.21	0.17	0.16	0.17	0.19	0.17	0.21	0.17	0.16	0.40	0.42	0.38	0.43
		20%	0.19	0.20	0.17	0.16	0.18	0.19	0.17	0.20	0.16	0.18	0.35	0.30	0.32	0.41
Agnews	LSTM	10%	0.23	0.29	0.14	0.17	0.24	0.23	0.31	0.22	0.21	0.23	0.51	0.58	0.47	0.53
		20%	0.23	0.26	0.15	0.17	0.24	0.23	0.28	0.22	0.20	0.22	0.46	0.40	0.40	0.49
	GRU	10%	0.24	0.24	0.14	0.16	0.25	0.24	0.18	0.24	0.22	0.24	0.51	0.59	0.44	0.54
		20%	0.24	0.24	0.15	0.18	0.25	0.23	0.21	0.24	0.21	0.24	0.47	0.40	0.37	0.48
	BiLSTM	10%	0.19	0.27	0.13	0.15	0.19	0.20	0.17	0.20	0.18	0.20	0.48	0.54	0.46	0.49
		20%	0.20	0.24	0.12	0.14	0.19	0.20	0.18	0.20	0.18	0.20	0.43	0.34	0.41	0.45
SVHN	LSTM	10%	0.55	0.70	0.48	0.49	0.38	0.54	0.60	0.55	0.31	0.57	0.75	0.77	0.71	0.79
		20%	0.54	0.60	0.51	0.51	0.40	0.54	0.60	0.54	0.31	0.57	0.72	0.66	0.65	0.77
	GRU	10%	0.52	0.52	0.37	0.39	0.56	0.52	0.54	0.54	0.29	0.55	0.74	0.68	0.78	0.78
		20%	0.51	0.52	0.41	0.39	0.55	0.52	0.52	0.53	0.29	0.55	0.72	0.59	0.73	0.76
	BiLSTM	10%	0.89	0.87	0.87	0.86	0.88	0.88	0.85	0.87	0.88	0.88	0.86	0.88	0.88	0.86
		20%	0.88	0.88	0.87	0.86	0.88	0.88	0.88	0.88	0.89	0.88	0.87	0.87	0.88	0.87

A. RQ1: Effectiveness

Figure 5 shows the box plot of the bug detection rate under thirty independent repeated experiments when the selection rate is 10%. Table V shows the evaluation results of all selection methods when the selection rates are 10% and 20% respectively. Compared to most baselines, DeepVec ranks in the top 2 for bug detection rate, with an average relative improvement of 12.25% to 118.22%. This demonstrates that DeepVec can more fully utilize the information contained in the state-vector and help RNN detect the most bugs under constraints.

On SVHN-BiLSTM, the performance of all methods was similar to random selection, which may be due to the incompatibility of BiLSTM with SVHN. This is likely because RNN is originally designed for text tasks, not image-based tasks. RNN models have a lower bug detection rate on text classification tasks compared to image classification tasks. However, DeepVec achieves an average relative improvement of 95.88% over baseline methods on text datasets, which is higher than the 73.78% improvement observed on image datasets. This may be due to the fact that the selected data are not long text sequences, and the text augmentation method we used has a limited degree of perturbation to the data, which leads to a higher accuracy. Meanwhile, DeepVec selects the test set with the highest bug detection rate over DeepState. This is because DeepVec uses a quantitative measure obtained through

sequence weighting, containing the time step order, resulting in more accurate similarity.

Answer to RQ 1: Under the same selection rate, DeepVec achieves an average relative improvement of 12.25% to 118.22% over other baselines in selecting bug-revealing tests. This indicates that DeepVec has a stronger capability for discovering fault-revealing tests.

B. RQ2: Inclusiveness

The inclusiveness of test cases selected by each baseline method across all datasets is methodically assessed at selection rates of 1%, 2%, ..., up to 40%. This approach enables an examination of how inclusiveness evolves in relation to the selection rate. As shown in Figure 6, as the selection rate increases, the inclusiveness of almost all methods increases, and the increase of DeepVec is particularly significant.

DeepVec has seen a significant improvement compared to traditional coverage-based selection methods. For some models and datasets, DeepState, SC(CAM), and HSCOV(CAM) are effective for test case selection, but not as effective as DeepVec. The experimental results show that methods based on uncertainty significantly outperform other baselines. At lower selection rates, ATS achieves higher inclusiveness than

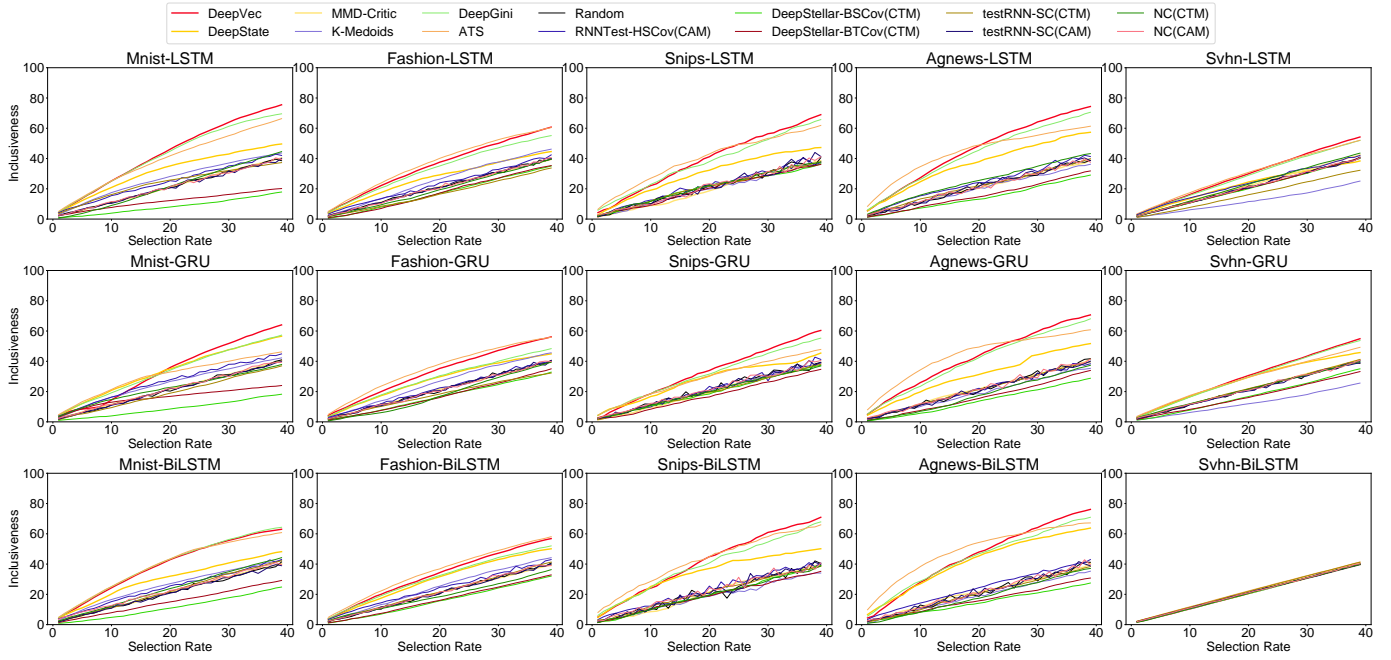


Figure 6. The changing trend of inclusiveness of different methods with selection rate.

Table VI
AVERAGE INCLUSIVENESS OF 20% AND 40% SELECTED TEST CASES ON THE ORIGINAL AND AUGMENTED TEST SETS.

Model	Set	Method									
		Deep State	Deep Gini	ATS	Deep Vec	Deep State	Deep Gini	ATS	Deep Vec		
		20%					40%				
Mnist	LSTM	ori	0.21	0.23	0.25	0.24	0.42	0.44	0.45	0.46	
		aug	0.58	0.58	0.52	0.66	0.72	0.70	0.62	0.78	
	GRU	ori	0.22	0.21	0.24	0.21	0.44	0.43	0.44	0.44	
		aug	0.36	0.34	0.60	0.48	0.57	0.66	0.71	0.86	
	BiLSTM	ori	0.23	0.24	0.25	0.23	0.42	0.45	0.44	0.45	
		aug	0.45	0.82	0.62	0.88	0.55	0.92	0.64	0.92	
Fashion	LSTM	ori	0.24	0.23	0.26	0.25	0.42	0.44	0.46	0.47	
		aug	0.46	0.39	0.56	0.53	0.66	0.60	0.79	0.81	
	GRU	ori	0.24	0.22	0.26	0.24	0.41	0.40	0.46	0.45	
		aug	0.44	0.34	0.53	0.51	0.63	0.60	0.77	0.80	
	BiLSTM	ori	0.23	0.23	0.26	0.24	0.44	0.43	0.46	0.46	
		aug	0.46	0.33	0.44	0.47	0.64	0.56	0.62	0.73	
Snips	LSTM	ori	0.36	0.37	0.38	0.38	0.49	0.66	0.54	0.71	
		aug	0.28	0.34	0.34	0.35	0.45	0.59	0.52	0.63	
	GRU	ori	0.30	0.31	0.29	0.33	0.48	0.57	0.49	0.62	
		aug	0.25	0.28	0.28	0.30	0.40	0.52	0.47	0.57	
	BiLSTM	ori	0.40	0.41	0.43	0.46	0.53	0.71	0.55	0.76	
		aug	0.29	0.34	0.37	0.40	0.45	0.62	0.54	0.68	
Agnews	LSTM	ori	0.38	0.44	0.43	0.48	0.56	0.72	0.58	0.76	
		aug	0.33	0.37	0.38	0.40	0.52	0.66	0.54	0.69	
	GRU	ori	0.31	0.41	0.40	0.43	0.52	0.69	0.56	0.71	
		aug	0.29	0.38	0.36	0.39	0.50	0.65	0.51	0.67	
	BiLSTM	ori	0.43	0.46	0.46	0.47	0.63	0.73	0.60	0.77	
		aug	0.39	0.41	0.41	0.43	0.60	0.69	0.56	0.72	
SVHN	LSTM	ori	0.21	0.20	0.22	0.21	0.41	0.40	0.42	0.41	
		aug	0.30	0.41	0.34	0.43	0.41	0.66	0.51	0.70	
	GRU	ori	0.22	0.21	0.22	0.22	0.42	0.42	0.42	0.43	
		aug	0.38	0.38	0.32	0.44	0.55	0.65	0.51	0.71	
	BiLSTM	ori	0.20	0.20	0.20	0.20	0.40	0.40	0.40	0.40	
		aug	0.20	0.20	0.21	0.20	0.40	0.40	0.42	0.40	

DeepVec. However, as the selection rate increases, DeepVec attains higher inclusiveness, indicating its suitability when the total number of candidates is small and a higher selection ratio is required. On the Agnews and Snips text datasets, both ATS

and DeepState initially improve with increasing selection rates and then stabilize, indicating the limitations of these methods on text datasets and their adaptation to RNNs. In the case of SVHN-BiLSTM, all methods almost converge into a single line, which may be due to the unsuitability of BiLSTM for the SVHN dataset.

To further explore the performance differences of various test case selection methods on the original and augmented test sets, Table VI calculates the inclusiveness values for four uncertainty-based methods at selection rates of 20% and 40%. On the original test set, DeepVec outperforms most baseline methods in most cases, achieving an average relative improvement of 12.25% and 33.01% at selection rates of 20% and 40%, respectively. On the augmented test set, DeepVec significantly outperforms the baseline methods, with an average relative improvement of 52.16% and 70.57% at selection rates of 20% and 40%, respectively. This demonstrates that DeepVec has a stronger ability to select fault-revealing tests, thereby enhancing the generalization and robustness of the DNN.

Answer to RQ 2: DeepVec outperforms most baselines in inclusiveness and achieves an average relative improvement of 22.63% over baselines on the original test set and an average relative improvement of 61.36% on the augmented test set. This indicates that DeepVec can select more test cases capable of revealing faults within the same candidate set.

C. RQ3: Fault Diversity

At selection rates of 1%, 2%, ..., up to 20%, we calculate the Fault Diversity of the test cases selected by various methods. As shown in Figure 7, the Fault Diversity of all methods increases as the selection rate rises. It is evident that DeepVec achieves

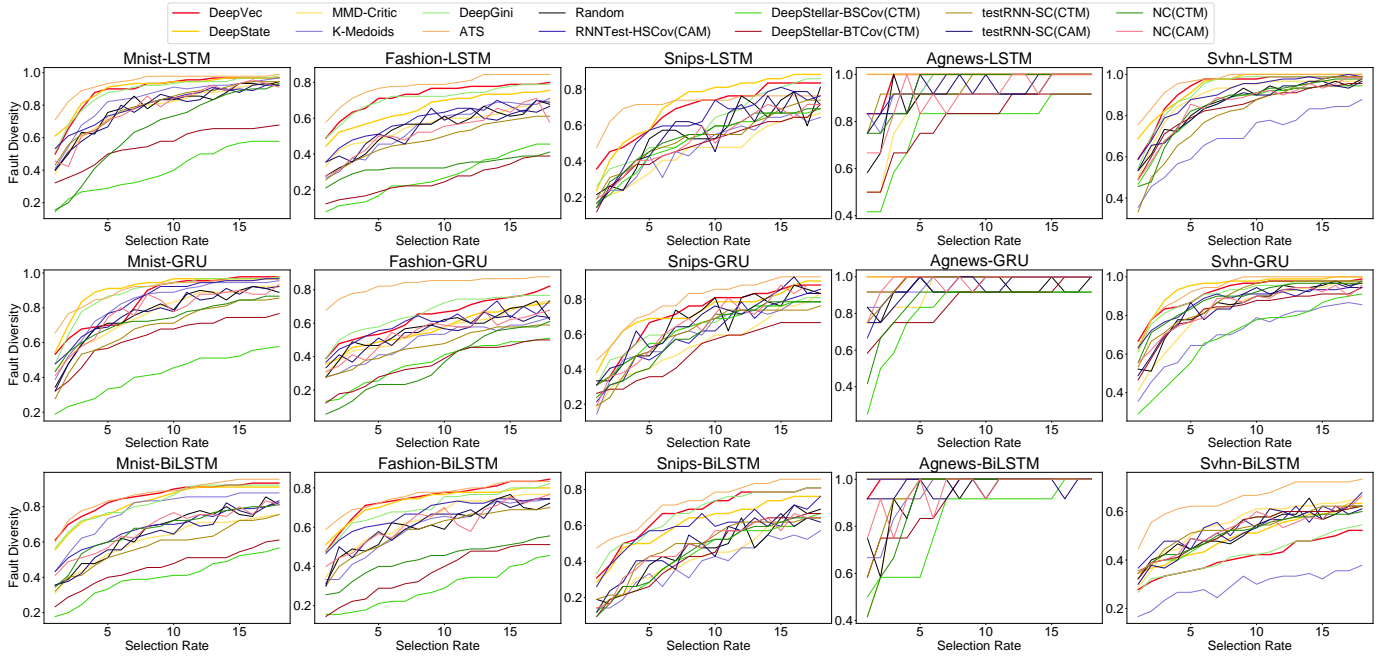


Figure 7. The changing trend of Fault Diversity of different methods with selection rate.

Table VII

AVERAGE IMPROVEMENT IN RNN ACCURACY ON THE TEST DATASET AFTER 10 RETRAININGS USING 15% OF THE SELECTED TEST DATASET. VALUES HIGHLIGHTED IN RED AND BLUE INDICATE THE BEST AND SECOND BEST.

Model	Set	Ran.	HSCov (CAM)	BSCov (CTM)	BTCov (CTM)	SC (CTM)	SC (CAM)	NC (CTM)	NC (CAM)	K-Medoids	MMD Critic	Deep Gini	ATS	Deep State	Deep Vec	100% Tests
Mnist	ori	0.58%±0.05	-0.16%±0.52	0.35%±0.2	0.01%±0.25	-0.01%±0.07	0.36%±1.01	-0.35%±0.25	0.38%±0.39	0.24%±1.27	0.4%±0.22	0.31%±2.34	0.25%±1.39	0.37%±0.12	0.18%±1.44	0.6%
	aug	14.17%±0.23	17.27%±0.73	2.95%±0.62	5.82%±1.2	12.19%±0.35	14.21%±1.16	5.63%±0.7	14.15%±0.98	14.71%±1.11	13.18%±0.71	23.42%±1.52	26.14%±1.32	17.73%±0.55	24.82%±0.83	37.13%
	mix	7.38%±0.12	8.56%±0.38	1.65%±0.37	2.92%±0.64	6.09%±0.19	7.28%±0.54	2.64%±0.43	7.27%±0.67	7.48%±1.16	6.79%±0.43	11.87%±1.83	13.2%±0.96	9.05%±0.28	12.5%±0.96	18.87%
	ori	0.0%±0.03	0.06%±0.06	-0.07%±0.01	-0.29%±0.04	-0.03%±0.06	-0.18%±0.02	-0.04%±0.02	0.01%±0.07	-0.02%±0.04	0.18%±0.03	0.07%±0.01	-0.25%±0.02	-0.02%±0.02	0.2%±0.02	0.16%
	aug	10.74%±0.08	16.65%±0.63	4.07%±1.2	9.47%±0.58	10.23%±0.36	11.01%±0.85	13.64%±0.47	12.18%±0.08	14.59%±0.27	15.73%±1.26	21.1%±0.11	23.32%±0.05	19.29%±0.06	23.82%±0.17	35.6%
	mix	5.37%±0.04	8.35%±0.29	2.0%±0.6	4.59%±0.31	5.1%±0.16	5.41%±0.43	6.8%±0.24	6.09%±0.08	7.28%±0.15	7.95%±0.64	10.58%±0.06	11.53%±0.03	9.63%±0.02	12.01%±0.09	17.88%
Fashion	ori	0.23%±0.19	-0.09%±0.51	0.05%±1.12	-0.39%±0.19	0.16%±2.23	-0.03%±0.77	-0.8%±0.42	-0.03%±0.35	-1.0%±0.5	-0.18%±0.44	0.12%±0.47	-0.12%±0.18	-0.94%±0.08	-0.01%±1.64	0.12%
	aug	15.52%±0.53	15.24%±0.77	2.18%±0.78	1.81%±0.51	14.38%±1.39	17.35%±1.09	16.43%±0.59	15.67%±0.88	12.78%±0.68	14.92%±0.45	16.86%±0.91	19.79%±0.72	16.58%±0.5	19.75%±0.99	38.84%
	mix	7.88%±0.32	7.57%±0.57	1.12%±0.87	0.71%±0.29	7.27%±1.79	8.66%±0.87	7.82%±0.49	7.82%±0.48	5.89%±0.55	7.37%±0.43	8.49%±0.58	9.83%±0.33	7.82%±0.28	9.87%±1.1	19.48%
	ori	-0.04%±0.09	0.01%±0.28	0.14%±0.3	-0.31%±0.28	0.41%±0.29	-0.41%±0.12	0.35%±0.15	-0.27%±0.14	0.24%±0.07	0.33%±0.06	0.31%±0.05	0.13%±0.07	0.2%±0.11	0.43%±0.09	0.76%
	aug	5.62%±0.13	6.82%±0.27	2.82%±0.8	3.07%±0.88	5.57%±0.24	6.47%±1.22	3.51%±0.16	6.81%±0.3	6.93%±0.14	7.13%±0.13	10.67%±0.11	11.23%±0.11	9.79%±0.15	12.49%±0.16	16.42%
	mix	2.79%±0.09	3.42%±0.27	1.48%±0.53	1.38%±0.57	2.99%±0.24	3.03%±0.59	1.93%±0.11	3.27%±0.1	3.58%±0.08	3.73%±0.08	5.49%±0.06	5.68%±0.08	5.0%±0.1	6.46%±0.1	8.59%
Snips	ori	0.06%±0.09	0.1%±0.27	-0.47%±0.08	0.14%±0.38	0.13%±0.07	-0.27%±0.09	0.03%±0.11	-0.6%±0.12	-0.11%±0.06	-0.16%±0.26	-0.19%±0.06	0.27%±0.05	-0.16%±0.04	0.3%±0.05	1.03%
	aug	3.99%±0.39	5.17%±0.64	1.98%±0.17	1.57%±0.83	3.16%±0.21	4.42%±0.24	4.36%±0.65	3.33%±0.65	5.67%±0.14	6.42%±1.03	8.24%±0.09	9.96%±0.06	7.76%±0.12	7.61%±0.08	15.79%
	mix	2.03%±0.22	2.64%±0.45	0.76%±0.06	0.85%±0.6	1.64%±0.13	2.07%±0.09	2.2%±0.33	1.37%±0.27	2.78%±0.08	3.13%±0.64	4.03%±0.6	5.12%±0.04	3.8%±0.06	3.96%±0.04	8.41%
	ori	2.44%±0.76	1.8%±0.72	1.79%±0.39	1.23%±1.32	1.73%±0.48	2.25%±0.51	0.69%±0.93	0.97%±0.44	1.77%±0.23	1.53%±0.43	1.44%±0.6	1.92%±0.43	1.35%±0.63	0.09%±0.37	2.83%
	aug	5.35%±0.93	5.4%±0.79	4.16%±0.63	1.67%±0.79	4.78%±1.1	4.84%±0.57	5.2%±0.48	4.68%±0.53	5.57%±0.94	4.44%±0.46	7.77%±0.74	9.34%±0.43	6.93%±0.54	9.34%±0.69	15.27%
	mix	3.89%±0.73	3.6%±0.69	2.97%±0.49	1.45%±0.95	3.25%±0.65	3.54%±0.51	2.94%±0.63	2.82%±0.35	3.67%±0.54	2.99%±0.35	4.61%±0.65	5.63%±0.31	4.14%±0.47	4.72%±0.46	9.05%
Agnews	ori	-5.84%±0	-6.16%±0.05	-5.5%±0	-4.8%±0.04	-4.96%±0.05	-5.56%±0	-6.45%±0.04	-6.65%±0.06	0.52%±0	-1.16%±0.06	-0.32%±0	-2.12%±0	-5.4%±0.01	0.0%±0.02	4.24%
	aug	-4.36%±0	-4.32%±0.02	-4.72%±0	-3.56%±0.04	-4.32%±0	-4.52%±0	-4.88%±0.03	-3.84%±0.08	-0.04%±0.04	-0.68%±0.08	-0.32%±0	-0.56%±0.01	-4.52%±0.03	0.08%±0	5.88%
	mix	-5.1%±0	-5.24%±0.03	-5.14%±0	-4.18%±0.04	-4.64%±0.03	-5.04%±0	-5.66%±0.04	-5.24%±0.06	0.24%±0.02	-0.92%±0.07	-0.82%±0	-1.34%±0.01	-4.96%±0.02	0.04%±0.01	5.06%
	ori	-3.6%±2.48	-3.28%±0.95	-3.04%±1.34	-1.36%±1.18	-5.68%±2.81	-1.92%±1.14	-1.08%±1.9	-3.64%±2.47	4.8%±1.53	2.92%±0.09	4.12%±0.71	2.8%±0.73	4.32%±0.7	4.84%±1.62	9.77%
	aug	-3.84%±2.34	-2.36%±0.9	-2.52%±1.28	-1.0%±1.27	-6.24%±2.82	-2.48%±1.16	-0.52%±1.71	-3.32%±2.3	4.6%±1.45	3.24%±0.06	3.44%±0.8	2.84%±1.15	4.08%±0.9	4.6%±1.64	9.77%
	mix	-3.72%±2.41	-2.82%±0.92	-2.78%±1.31	-1.18%±1.23	-5.96%±2.81	-2.2%±1.15	-0.8%±1.8	-3.48%±2.38	4.7%±1.49	3.08%±0.08	3.78%±0.75	2.82%±0.94	4.2%±0.8	4.72%±1.63	9.69%
SVHN	ori	26.66%±1.91	27.54%±2.77	27.86%±2.66	29.62%±2.85	27.42%±1.99	27.7%±2.16	27.7%±2.76	27.98%±2.34	18.65%±0.08	21.54%±0.73	20.9%±0.48	21.58%±0.01	27.38%±2.3	27.66%±2.35	26.54%
	aug	24.1%±1.99	24.5%±2.48	24.26%±2.46	25.74%±2.5	23.7%±1.47	25.1%±2.28	25.18%±2.61	24.58%±2.08	16.53%±0.1	18.98%±0.29	18.69%±0.29	19.22%±0.13	23.98%±1.93	25.22%±2.36	23.94%
	mix	25.38%±1.95	26.02%±2.63	26.06%±2.56	27.68%±2.67	25.56%±1.73	26.4%±2.22	26.44%±2.68	26.28%±2.21	17.59%±0.09	20.26%±0.52	19.8%±0.39	20.4%±0.07	25.68%±2.11	26.44%±2.35	25.24%
	ori	-0.64%±0.03	0.08%±0.04	-0.09%±0.05	0.41%±0.06	-0.07%±0.03	0.5%±0.04	0.28%±0.04	0.18%±0.04	0.14%±0.08	0.04%±0.02	0.57%±0.03	-0.79%±0.03	0.96%±0.05	0.97%±0.05	0.24%
	aug	-0.22%±0.05	-0.16%±0.07	0.21%±0.04	0.5%±0.07	0.12%±0.04	0.28%±0.04	0.66%±0.05	-0.29%±0.04	-0.07%±0.03	-0.72%±0.03	0.41%±0.06	-0.11%±0.04	0.49%±0.02	1.34%±0.05	0.83%
	mix	-0.43%±0.03	-0.04%±0.04	0.06%±0.03	0.45%±0.06	0.03%±0.03	0.39%±0.04	0.47%±0.04	-0.05%±0.04	0.04%±0.04	-0.34%±0.02	0.49%±0.04	-0.45%±0.02	0.72%±0.03	1.16%±0.04	0.53%
SVHN	ori	0.63%±0.04	0.47%±0.05	0.45%±0.08	-0.67%±0.06	0.36%±0.03	0.57%±0.04	0.32%±0.04	0.67%±0.08	0.58%±0.03	0.89%±0.06	0.32%±0.06	0.45%±0.03	0.88%±0.06	1.03%±0.04	1.88%
	aug	0.62%±0.08	0.49%±0.07	0.49%±0.07	-0.95%±0.04	0.42%±0.03	0.79%±0.06	0.21%±0.06	0.61%±0.06	0.39%±0.03	0.76%±0.04	0.25%±0.04	0.51%±0.05	0.34%±0.04	0.96%±0.07	2.51%
	mix	0.63%±0.03	0.24%±0.06	0.47%±0.07	-0.81%±0.03	0.39%±0.03	0.68%±0.03	0.26%±0.04	0.54%±0.03	0.49%±0.02	0.83%±0.03	0.28%±0.03	0.48%±0.03	0.61%±0.03	0.99%±0.04	2.2%
	ori	0.78%±0.09	0.88%±0.06	0.21%±0.04	0.72%±0.39	1.07%±0.06	0.14%±0.32	0.64%±0.07	0.92%±0.12	0.39%±0.26	0.67%±0.37	1.13%±0.13	1.14%±0.65	1.34%±0.3	1.58%±0.04	2.22%
	aug	0.82%±0.21	1.16%±0.13	0.78%±0.16	1.38%±0.34	0.88%±0.16	0.04%±0.07	1.17%±0.25	1.25%±0.22	0.03%±0.5	0.47%±0.27	1.76%±0.34	1.36%±0.75	1.67%±0.29	2.32%±0.04	3.07%
	mix	0.8%±0.14	1.04%±0.08	0.49%±0.09	1.05%±0.37	0.97%±0.11	0.09%±0.17	0.91%±0.16	1.09%±0.16	0.28%±0.38	1.25%±0.32	1.45%±0.23	1.25%±0.7	1.51%±0.29	1.95%±0.04	2.64%
SVHN	ori	1.79%±1.12	1.82%±0.2	3.83%±0.97	0.85%±0.56	1.75%±0.43	2.34%±0.53	3.45%±0.5	0.43%±0.58	2.42%±0.81	1.54%±0.76	2.14%±0.98	2.51%±0.94	-0.8%±0.97	2.96%±0.49	3.33%
	aug	1.63%±0.25	2.14%±0.34	2.8%±0.49	2.25%±1.06	0.7%±0.48	2.9%±0.29	2.37%±0.35	2.09%±0.45	0.8%±0.48	2.02%±0.34	2.77%±0.5	1.84%±0.33	1.3%±0.37	2.74%±0.45	7.85%
	mix	1.71%±0.56	1.98%±0.21	3.32%±0.71	1.55%±0.71	1.22%±0.38	2.62%±0.25	2.91%±0.21	1.26%±0.39	1.66%±0.62	1.78%±0.71	2.45%±0.71	2.18%±0.47	0.25%±0.53	2.85%±0.38	5.59%
	ori	1.31%±0.49	-0.07%±0.22	1.44%±0.16	1.14%±0.33	1.51%±0.24	0.26%±0.44	1.8%±0.28	0.52%±0.62	0.91%±0.06	1.12%±0.6	0.69%±0.48	1.04%±0.18	1.87%±0.22	2.01%±0.5	3.4%
	aug	2.24%±0.62	1.33%±0.21	2.07%±0.51	2.04%±0.74	1.59%±0.23	2.2%±0.5	1.01%±0.74	0.76%±0.39	-0.02%±0.21	1.66%±0.33	2.01%±0.83	2.14%±0.48	2.65%±0.33	1.92%±0.32	6.47%
	mix	1.78%±0.08	0.63%±0.08	1.76%±0.25	1.59%±0.37	1.8%±0.15	0.92%±0.31	2.0%±0.23	0.76%±0.39	0.45%±0.09	1.39%±0.46	1.35%±0.63	1.59%±0.16	2.26%±0.25	1.96%±0.21	4.93%
SVHN	ori	-4.44%±0.9	-2.76%±1.59	-0.02%±1.51	-4.18%±1.7	-1.38%±1.2	-3.51%±1.52	0.38%±1.77	-2.66%±1.21	-2.65%±0.63	-2.52%±1.54	-3.12%±2.09	0.05%±1.55	0.41%±1.08	0.36%±1.59	1.01%
	aug	-0.34%±0.4	0.77%±0.6	0.57%±0.45	-0.52%±0.51	0.88%±0.41	0.0%±0.47	0.0%±0.42	0.34%±0.73	-0.11%±0.61	-0.78%±0.36	0.01%±0.58	0.61%±0.26	0.18%±0.55	-0.03%±0.54	3.75%
	mix	-2.39%±0.6	-0.99%±1	0.28%±0.86	-2.35%±0.84	-0.25%±0.67	-1.76%±0.79	0.19%±0.97	-1.16%±0.8	-1.38%±0.44	-1.65%±0.72	-1.55%±1.25	0.33%±0.82	0.3%±0.66	0.17%±1	2.38%
Statistical Results		p-value	***	***	***	***	***	***	***	***	***	***	***	***	***	---
Effect size		0.26	0.27	0.33	0.30	0.26	0.26	0.23	0.29	0.21	0.18	0.10	0.10	0.10	---	---

1. *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$, $p > 0.05$.
2. Effect size is computed according to Cliff's delta.

higher Fault Diversity in the selected test cases compared to all methods except ATS. This may be because ATS divides test inputs into subsets based on their predicted categories and then selects test cases that are least likely to belong to these subsets. This diversity-oriented clustering approach gives ATS a stronger ability to select more diverse tests. At lower selection rates (e.g., below 20%), ATS demonstrates a strong ability to identify diverse test cases. However, as the selection rate increases, the Fault Diversity of ATS gradually stabilizes. At a 20% selection rate, DeepVec achieves performance on par with ATS.

It is undeniable that ATS excels in selecting a small number of test cases, but this advantage becomes a drawback for datasets with a smaller overall size and a higher number of categories. A low selection rate may not cover a sufficient number of samples, which can actually hinder the improvement of model performance despite having more diverse sample categories.

Answer to RQ 3: Under the same selection rate, the test cases selected by DeepVec outperform those chosen by most baseline methods, but is not as diverse as ATS.

D. RQ4: Guidance

Table VII shows the average improvement in RNN accuracy on original, augmented and mixed test set after retraining the RNN 10 times using tests selected from the candidate set with various selection methods. Additionally, we report the standard deviation of the results over 10 runs. To represent the extent of improvement, we also retrained the RNN using all data in the candidate set and displayed its results. Experiment results show that DeepVec achieves an absolute improvement in average accuracy ranging from 0.29% to 24.01% compared to all baselines.

For image classification tasks, the improvement in RNN accuracy on mixed test set surpasses that of text classification tasks. This might be because the enhancement methods for text features are not as evident as those for images. In addition, ATS also demonstrate excellent performance on the image classification dataset, almost on par with DeepVec. This may be due to ATS's use of image test case clustering, which enhances its performance on image datasets. DeepState and K-Medoids are effective on the text dataset, which not only validates the rationale behind the RNN state view but also highlights the necessity of test case clustering. This provides guidance for future improvements to DeepVec.

DeepVec focuses more on improving accuracy on the original test set for image datasets, while it tends to enhance accuracy on the augmented test set for text datasets. This could be attributed to the different working natures of RNNs, as they are better suited for handling serialized, context-dependent data rather than images. For SVHN-BiLSTM, the performance of all methods remains comparable to random selection. Therefore, we will no longer analyze this dataset in subsequent RQs.

We further performs a statistical analysis of the experiment results by conducting a Wilcoxon signed-rank test at a significance level of 95% to investigate whether DeepVec

significantly outperforms baselines. Additionally, we use the non-parametric Cliff's delta effect size to assess the magnitude of the difference between the two methods. Specifically, we find that all p-values were below 0.01, indicating that DeepVec outperforms the compared methods in improving DNNs. The effect size results show that, compared to other coverage-based selection methods, DeepVec achieves a larger performance difference. However, when compared to DeepGini and ATS, the average performance difference of DeepVec is smaller.

Table VIII
THE AVERAGE TIME COST (MS) OF DIFFERENT SELECTION ALGORITHMS RUNNING 30 TIMES ON THE SAME DATASET. VALUES HIGHLIGHTED IN RED AND BLUE INDICATE THE BEST AND SECOND BEST.

	Model	Sel	Cov Based	K-Medoids	MMD Critic	Deep State	ATS	Deep Gini	Deep Vec
Mnist	LSTM	5%	<1	5928.99	64835.22	61.17	308282.88	6.70	2.62
		10%	<1	13156.14	265690.58	306.64	326394.17	1.07	7.82
		15%	<1	11404.88	343915.16	443.59	345240.20	2.32	16.39
		20%	<3	17031.33	494450.59	292.73	338558.94	1.21	3.62
	GRU	5%	<1	7391.95	123225.12	292.30	344638.82	1.29	22.85
		10%	<1	10741.67	236921.98	445.92	321000.52	6.16	2.75
		15%	<1	10843.51	326093.46	1038.03	329958.96	1.12	3.99
		20%	<3	15343.65	443259.19	1025.98	309731.31	37.27	3.19
	BiLSTM	5%	<1	11548.56	108859.13	53.97	174721.23	0.98	2.06
		10%	<2	13750.78	265121.03	443.82	182076.71	1.31	12.53
		15%	<2	16616.19	334134.24	403.06	178295.35	1.12	2.57
		20%	<4	15710.82	409426.31	732.67	182120.80	1.29	3.94
Fashion	LSTM	5%	<1	2652.40	25487.84	69.63	220877.97	0.80	1.57
		10%	<2	3026.27	2064.26	105.34	218768.14	17.12	2.53
		15%	<2	3701.49	3186.37	216.18	220676.99	0.90	2.17
		20%	<2	3175.20	4566.13	163.07	222008.99	0.94	2.51
	GRU	5%	<2	2897.84	1068.86	53.95	220361.33	0.81	1.63
		10%	<2	3261.73	2238.45	143.45	215165.64	0.87	1.96
		15%	<2	2206.76	4301.36	162.37	214760.46	0.92	2.19
		20%	<2	3504.27	2815.56	183.19	217940.81	0.97	2.63
	BiLSTM	5%	<4	2069.55	471.24	126.77	230603.80	1.24	2.66
		10%	<5	2516.38	2953.62	116.99	232683.71	0.84	1.95
		15%	<6	2630.87	2513.29	206.50	225454.35	0.96	2.44
		20%	<9	3763.65	3306.79	487.75	223729.44	1.52	4.43
Snips	LSTM	5%	<0.1	209.00	772.29	7.83	35649.41	0.37	1.02
		10%	<0.3	225.52	237.00	13.72	35825.82	0.35	1.04
		15%	<0.7	281.32	460.50	14.06	34033.58	0.31	0.88
		20%	<1	185.21	221.34	35.57	36046.70	0.49	1.65
	GRU	5%	<0.3	120.87	203.36	12.81	37049.84	0.26	0.66
		10%	<0.6	162.95	279.97	17.03	36041.09	0.28	0.82
		15%	<0.7	203.22	462.30	23.13	36433.58	0.29	0.93
		20%	<1.5	203.83	486.98	27.48	36485.25	0.51	1.96
	BiLSTM	5%	<0.2	141.00	230.82	10.48	36338.46	0.39	1.29
		10%	<0.4	226.79	1655.77	20.37	34868.31	0.43	1.20
		15%	<0.7	250.29	420.13	15.86	34603.28	0.29	0.84
		20%	<0.7	221.68	1895.93	19.98	34259.67	0.31	0.90
Agnews	LSTM	5%	<0.3	515.58	391.76	65.38	105244.20	0.91	2.21
		10%	<0.4	544.12	3043.75	143.04	138431.21	1.02	2.47
		15%	<0.4	766.80	1000.03	206.94	139217.47	0.93	2.70
		20%	<0.6	659.79	1764.08	245.95	138976.44	0.80	2.77
	GRU	5%	<0.3	496.44	570.70	36.11	129484.36	0.76	1.88
		10%	<0.3	543.00	507.14	57.46	139180.79	0.65	1.58
		15%	<0.5	659.51	1710.42	187.85	138172.29	1.08	3.04
		20%	<0.6	709.07	1674.40	223.42	136941.38	1.16	3.75
	BiLSTM	5%	<0.3	443.70	577.07	86.11	139559.13	0.60	1.17
		10%	<0.4	569.47	1365.95	86.45	140221.45	0.64	1.56
		15%	<0.6	629.95	2401.32	256.17	135901.07	1.04	2.86
		20%	<0.5	701.21	1301.97	221.04	133645.76	0.72	2.01
SVHN	LSTM	5%	<4	4011.65	1824.41	65.68	197226.32	1.42	2.79
		10%	<5	4001.06	2004.98	131.42	198004.87	1.59	3.81
		15%	<4	7840.46	169468.70	260.38	207326.35	1.48	4.06
		20%	<3	3954.11	7054.44	151.94	198173.25	1.11	3.08
	GRU	5%	<2	2970.11	1123.34	30.33	199102.01	0.94	1.81
		10%	<2	3735.06	3015.47	77.91	221435.04	1.04	2.41
		15%	<4	10186.96	134730.17	264.20	203819.19	1.32	3.64
		20%	<4	5281.05	4758.53	333.70	187817.18	1.71	5.28
	BiLSTM	5%	<4	3521.18	923.93	3615.13	156326.38	1.44	3.84
		10%	<4	4428.03	2916.42	6748.24	157287.97	1.57	4.58
		15%	<3	4904.79	151988.08	9541.67	161736.94	1.06	3.41
		20%	<3	4471.82	8669.92	14657.79	160303.24	1.21	4.02

Answer to RQ 4: The average absolute accuracy improvement outperforms existing methods by 0.29%-24.01%, indicating that the data selected by DeepVec can guide RNN retraining to enhance accuracy and robustness.

E. RQ5: Time Cost

In 30 independent repeated experiments, we record and report the runtime required by various algorithms, as shown in Table VIII. Coverage-based methods, such as CAM and CTM, have an average runtime of under 10 ms because they pre-collect neuron coverage in the DNN.

DeepGini has the lowest time complexity since it only requires simple Gini calculations combined with a quick sort algorithm. DeepVec exhibits the second-lowest time complexity, with its time overhead primarily coming from the global selection traversal for different CS values. DeepState also shows acceptable time overhead due to the involvement of multiple set operations and different set constructions within the method.

Although DeepGini has the smallest time cost, the experimental results from RQ1 to RQ4 clearly show that DeepGini exhibits limited performance in discovering fault-revealing tests, identifying diverse test cases, and improving DNN accuracy. DeepVec, while having a time cost on the same order of magnitude as DeepGini, strikes a better balance between performance and time efficiency.

For K-Medoids, MMD-Critic, and ATS, their time overhead is 10^5 to 10^6 times higher than DeepVec's. This is because, although clustering enhances the algorithm's ability to identify diverse test cases, it also introduces significant overhead due to the unsupervised training process.

Overall, DeepVec achieves performance that matches or exceeds ATS while maintaining time overhead on the same order of magnitude as DeepGini, offering the highest cost-effectiveness.

Answer to RQ 5: DeepVec typically has the smallest order of time cost as DeepGini ($< 10ms$) while offering a stronger capability to enhance the robustness of DNNs. The runtime of DeepVec is only 1% to 1%₀₀₀ of that of other methods, which makes DeepVec more practical and cost-effective.

F. RQ6: Ablation Study

Table IX shows the performance changes in DeepVec after removing DU or CS. Excluding the anomalous performance on SVHN-BiLSTM, when DU is removed, DeepVec's Bug Detection Rate decreases by 30.2% to 56.25%, and Inclusiveness decreases by 28.31% to 56.8%. When CS is removed, DeepVec's Bug Detection Rate decreases by 1.61% to 21.05%, and Inclusiveness decreases by 0% to 56.8%. This not only proves the effectiveness of each component in DeepVec but also validates the rationale behind DU. The purpose of CS is to help DeepVec select more diverse test cases, which is why CS has a smaller impact on the algorithm compared to DU.

Table IX
ABLATION EXPERIMENTS RESULTS MEASURING THE CONTRIBUTION OF EACH COMPONENT OF DEEPVEC.

Model		DeepVec	DeepVec w/o DU	DeepVec w/o CS	DeepVec	DeepVec w/o DU	DeepVec w/o CS
		Bug Detection Rate			Inclusiveness		
Mnist	LSTM	0.80	0.35	0.69	0.44	0.19	0.38
	GRU	0.62	0.35	0.61	0.34	0.19	0.34
	BiLSTM	0.76	0.37	0.60	0.41	0.20	0.32
Fashion	LSTM	0.67	0.37	0.57	0.36	0.20	0.30
	GRU	0.64	0.38	0.53	0.34	0.20	0.28
	BiLSTM	0.67	0.40	0.57	0.33	0.20	0.28
Snips	LSTM	0.48	0.27	0.41	0.39	0.22	0.33
	GRU	0.55	0.33	0.52	0.33	0.20	0.31
	BiLSTM	0.39	0.15	0.33	0.43	0.17	0.37
Agnews	LSTM	0.52	0.23	0.42	0.47	0.21	0.37
	GRU	0.50	0.23	0.42	0.42	0.19	0.35
	BiLSTM	0.45	0.19	0.38	0.46	0.20	0.39
SVHN	LSTM	0.76	0.53	0.73	0.29	0.20	0.27
	GRU	0.76	0.53	0.71	0.29	0.20	0.28
	BiLSTM	0.89	0.87	0.88	0.20	0.20	0.20

Answer to RQ 6: Both DU and CS contribute to the performance of the DeepVec algorithm, with DU having a greater impact than CS.

G. RQ7: Sensitive Analysis

Table X reports the Bug Detection Rate and Inclusiveness achieved by DeepVec under different angle thresholds τ . Table XI presents the corresponding Fault Diversity. In the vast majority of cases, as τ increases, DeepVec's ability to identify fault-revealing tests initially improves and then stabilizes, demonstrating the robustness of DeepVec across various angle thresholds. On certain models and datasets, there is a slight decrease in DeepVec's ability to select diverse test cases. However, this is acceptable since the decline does not exceed 3% and is accompanied by higher Bug Detection Rate and Inclusiveness.

We observe that on the Agnews dataset, the Fault Diversity remains at 100% across all τ values. This is likely due to the smaller number of classes in the Agnews dataset (only 4 classes), leading to fewer types of faults and easier detection. Additionally, we closely examine DeepVec's performance when τ exceeds τ_{given} . For Mnist, Fashion, and SVHN, τ_{given} is 71.57; for Agnews, it is 60; and for Snips, it is 67.79. When $\tau > \tau_{\text{given}}$, DeepVec's performance no longer changes, which further validates the reasonableness of the recommended angular threshold τ . Combined with the ablation experiment results from RQ6, it is evident that the calculation and selection of similarity are indispensable. Moreover, the angle threshold in the similarity calculation does not affect the robustness of DeepVec.

Answer to RQ 7: The performance of DeepVec increases initially and then stabilizes as τ increases, with DeepVec achieving optimal performance when $\tau > \tau_{\text{given}}$. Although the value of τ has minimal impact on DeepVec, the similarity calculation remains indispensable.

Table X
SENSITIVITY OF DEEPVEC'S PERFORMANCE (BUG DETECTION RATE & INCLUSIVENESS) TO THE ANGLE THRESHOLD.

Model	τ	0	15	30	45	60	75	90	0	15	30	45	60	75	90
	Sel	Bug Detection Rate							Inclusiveness						
Mnist	LSTM	10%	78.67%	85.17%	85.17%	85.17%	85.17%	85.17%	21.52%	23.30%	23.30%	23.30%	23.30%	23.30%	23.30%
		20%	69.00%	80.83%	80.83%	80.83%	80.83%	80.83%	37.76%	44.23%	44.23%	44.23%	44.23%	44.23%	44.23%
	GRU	10%	73.67%	79.67%	79.67%	79.67%	79.67%	79.67%	20.16%	21.81%	21.81%	21.81%	21.81%	21.81%	21.81%
		20%	61.75%	73.58%	73.58%	73.58%	73.58%	73.58%	33.80%	40.28%	40.28%	40.28%	40.28%	40.28%	40.28%
	BiLSTM	10%	84.33%	84.67%	84.50%	83.50%	84.33%	84.33%	22.75%	22.84%	22.80%	22.53%	22.75%	22.75%	22.71%
		20%	76.83%	77.33%	76.67%	75.83%	76.25%	77.17%	41.46%	41.73%	41.37%	40.92%	41.14%	41.64%	41.64%
Fashion	LSTM	10%	67.67%	76.17%	76.17%	76.17%	76.17%	76.17%	18.08%	20.36%	20.36%	20.36%	20.36%	20.36%	20.36%
		20%	57.17%	67.42%	67.33%	67.42%	67.42%	67.42%	30.56%	36.04%	35.99%	36.04%	36.04%	36.04%	36.04%
	GRU	10%	63.67%	70.00%	70.17%	70.00%	70.00%	70.00%	16.70%	18.36%	18.41%	18.36%	18.36%	18.36%	18.36%
		20%	52.75%	64.25%	64.42%	64.25%	64.25%	64.25%	27.68%	33.71%	33.80%	33.71%	33.71%	33.71%	33.71%
	BiLSTM	10%	67.50%	76.50%	76.50%	76.50%	76.50%	76.50%	16.55%	18.76%	18.76%	18.76%	18.76%	18.76%	18.76%
		20%	57.92%	67.67%	67.75%	67.75%	67.75%	67.75%	28.40%	33.18%	33.22%	33.22%	33.22%	33.22%	33.22%
Snips	LSTM	10%	55.50%	56.00%	56.00%	56.50%	54.00%	54.50%	22.52%	22.72%	22.72%	22.92%	21.91%	22.11%	22.11%
		20%	48.75%	49.75%	49.75%	50.25%	50.25%	49.50%	39.55%	40.37%	40.37%	40.77%	40.77%	40.16%	40.16%
	GRU	10%	59.00%	58.00%	59.50%	59.00%	60.50%	60.00%	17.64%	17.34%	17.79%	17.64%	18.09%	17.94%	17.94%
		20%	53.00%	54.00%	55.00%	54.75%	55.50%	55.25%	31.69%	32.29%	32.88%	32.74%	33.18%	33.03%	33.03%
	BiLSTM	10%	42.50%	42.00%	42.00%	41.50%	42.00%	41.50%	23.55%	23.27%	23.27%	22.99%	23.27%	22.99%	22.99%
		20%	40.25%	39.75%	39.75%	39.50%	39.50%	39.50%	44.60%	44.04%	44.04%	43.77%	43.77%	43.77%	43.77%
Agnews	LSTM	10%	56.58%	57.24%	57.24%	57.46%	57.46%	57.46%	25.24%	25.54%	25.54%	25.64%	25.64%	25.64%	25.64%
		20%	50.99%	52.52%	52.52%	52.63%	52.63%	52.63%	45.50%	46.87%	46.87%	46.97%	46.97%	46.97%	46.97%
	GRU	10%	55.48%	56.36%	56.58%	56.14%	56.14%	56.14%	23.28%	23.64%	23.74%	23.55%	23.55%	23.55%	23.55%
		20%	50.00%	49.45%	49.67%	49.45%	49.56%	49.56%	41.95%	41.49%	41.67%	41.49%	41.58%	41.58%	41.58%
	BiLSTM	10%	53.73%	53.51%	53.29%	53.51%	53.51%	53.51%	27.71%	27.60%	27.49%	27.60%	27.60%	27.60%	27.60%
		20%	44.74%	46.38%	46.27%	46.38%	46.38%	46.38%	46.15%	47.85%	47.74%	47.85%	47.85%	47.85%	47.85%
SVHN	LSTM	10%	77.43%	78.43%	78.43%	78.43%	78.43%	78.43%	14.56%	14.75%	14.75%	14.75%	14.75%	14.75%	14.75%
		20%	72.64%	75.93%	76.00%	76.00%	76.00%	76.00%	27.32%	28.55%	28.58%	28.58%	28.58%	28.58%	28.58%
	GRU	10%	75.14%	79.43%	79.14%	79.43%	79.43%	79.43%	14.53%	15.36%	15.30%	15.36%	15.36%	15.36%	15.36%
		20%	71.21%	75.64%	75.50%	75.71%	75.64%	75.64%	27.54%	29.25%	29.20%	29.28%	29.25%	29.25%	29.25%
	BiLSTM	10%	80.14%	81.43%	81.43%	81.43%	81.43%	81.43%	9.31%	9.46%	9.46%	9.46%	9.46%	9.46%	9.46%
		20%	82.50%	81.21%	81.21%	81.21%	81.21%	81.21%	19.17%	18.87%	18.87%	18.87%	18.87%	18.87%	18.87%

Table XI
SENSITIVITY OF DEEPVEC'S PERFORMANCE (FAULT DIVERSITY) TO THE ANGLE THRESHOLD.

Model	τ	0	15	30	45	60	75	90
	Sel	Fault Diversity						
Mnist	LSTM	10%	94.44%	93.33%	92.22%	93.33%	93.33%	93.33%
		20%	98.89%	96.67%	96.67%	96.67%	96.67%	96.67%
	GRU	10%	96.67%	96.67%	96.67%	96.67%	96.67%	96.67%
		20%	97.78%	98.89%	98.89%	98.89%	98.89%	98.89%
	BiLSTM	10%	88.89%	90.00%	90.00%	88.89%	90.00%	88.89%
		20%	94.44%	92.22%	94.44%	94.44%	93.33%	94.44%
Fashion	LSTM	10%	71.11%	76.67%	76.67%	76.67%	76.67%	76.67%
		20%	77.78%	81.11%	81.11%	81.11%	81.11%	81.11%
	GRU	10%	66.67%	65.56%	65.56%	65.56%	65.56%	65.56%
		20%	78.89%	82.22%	82.22%	82.22%	82.22%	82.22%
	BiLSTM	10%	77.78%	76.67%	76.67%	76.67%	76.67%	76.67%
		20%	81.11%	84.44%	84.44%	84.44%	84.44%	84.44%
Snips	LSTM	10%	66.67%	61.90%	64.29%	64.29%	59.52%	59.52%
		20%	85.71%	83.33%	85.71%	85.71%	83.33%	83.33%
	GRU	10%	66.67%	73.81%	73.81%	66.67%	66.67%	66.67%
		20%	80.95%	85.71%	85.71%	85.71%	85.71%	85.71%
	BiLSTM	10%	76.19%	73.81%	73.81%	73.81%	71.43%	71.43%
		20%	83.33%	80.95%	80.95%	80.95%	80.95%	80.95%
Agnews	LSTM	10%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
		20%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
	GRU	10%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
		20%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
	BiLSTM	10%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
		20%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
SVHN	LSTM	10%	97.78%	97.78%	97.78%	97.78%	97.78%	97.78%
		20%	97.78%	98.89%	98.89%	98.89%	98.89%	98.89%
	GRU	10%	93.33%	96.67%	95.56%	96.67%	96.67%	96.67%
		20%	100.00%	98.89%	97.78%	98.89%	98.89%	98.89%
	BiLSTM	10%	46.67%	46.67%	46.67%	46.67%	46.67%	46.67%
		20%	61.11%	61.11%	61.11%	61.11%	61.11%	61.11%

H. RQ8: Generalizability

Table XII shows the improvement in model accuracy

achieved by the data selected by DeepVec for models trained on the augmented dataset. On almost all datasets, some methods lead to a decrease in model accuracy, which may be due to the fact that using the same method to augment the training set resulted in fewer fault-revealing tests in the selected test set. Even under these circumstances, DeepVec still performs excellently, indicating its generalization capability in real-world applications. Additionally, we observe that compared to ATS and DeepGini, which are designed for CNNs, method designed for RNNs demonstrate more stable performance in test environments with lower-quality test cases. This may be related to the fact that both methods utilize state transition information from the RNN.

Answer to RQ 8: DeepVec improved the performance of models trained on augmented data by 0.26%-2.91%, demonstrating its generalization capability.

VI. DISCUSSION

In this section, we will discuss the reason why DeepVec works and threats to validity. Moreover, we will discuss the differences between test case selection techniques and adversarial training, as well as the novelty of DeepVec.

A. Why does DeepVec Work?

Through the comparison of DeepVec with existing NC-guided methods and DeepState, it has been demonstrated that

Table XII
ACCURACY IMPROVEMENT OF RNNs ON THE AUGMENTED TEST DATASET AFTER RETRAINING USING 15% OF THE SELECTED TEST DATASET. VALUES HIGHLIGHTED IN RED AND BLUE INDICATE THE BEST AND SECOND BEST.

	Model	Ran.	HSCov (CAM)	BSCov (CTM)	BTCov (CTM)	SC (CTM)	SC (CAM)	NC (CTM)	NC (CAM)	K- Medoids	MMD Critic	Deep Gini	ATS	Deep State	Deep Vec
Mnist	LSTM	0.14	-0.33	0.26	0.04	-0.67	-0.41	-0.6	-0.23	-0.12	-0.13	-0.59	-1.27	-0.56	0.26
	GRU	0.15	0.33	0.34	0.51	0.26	0.37	1.22	0.55	0.80	0.83	0.86	0.05	0.33	1.06
	BiLSTM	-0.01	0.19	0.04	0.40	0.27	0.54	-0.60	0.37	0.35	-0.62	0.44	-0.59	0.18	0.53
Fashion	LSTM	0.53	1.19	0.62	0.64	0.65	0.92	0.69	0.89	0.74	1.19	0.56	0.07	0.64	0.77
	GRU	-0.29	0.28	-0.06	0.13	-0.86	-0.28	0.24	-0.19	-0.26	-0.21	0.48	-1.46	-0.23	0.28
	BiLSTM	0.06	0.9	0.21	-0.77	0.74	1.41	0.57	1.49	-0.24	1.51	1.3	0.45	1.84	1.85
Snips	LSTM	0.02	-0.1	-0.6	-0.08	-1.04	-0.60	0.14	0.12	-0.08	0.4	-0.42	-0.10	0.44	0.60
	GRU	-0.76	-0.34	-1.04	-0.14	-0.46	-0.34	-1.34	-0.96	0.44	-0.86	-1.72	0.20	0.42	0.92
	BiLSTM	-0.12	-1.28	0.18	-2.04	0.04	0.56	0.78	0.44	1.30	-1.60	-1.22	0.52	-0.16	1.34
Agnews	LSTM	0.38	0.43	0.30	0.43	0.05	0.23	-0.09	0.36	0.40	0.09	0.15	0.32	0.70	0.63
	GRU	-0.08	-0.21	-0.18	-0.38	0.06	0.08	-0.14	-0.11	-0.37	-0.26	-0.31	-0.02	0.18	0.32
	BiLSTM	0.24	0.28	0.07	-0.06	0.18	-0.20	0.06	-0.13	-0.03	-0.16	-0.45	0.32	0.41	0.16
SVHN	LSTM	2.06	1.63	1.99	2.40	1.81	0.86	0.32	1.43	0.09	-0.48	1.02	1.04	1.24	2.69
	GRU	0.53	0.23	-0.69	0.22	-1.43	0.97	-0.18	0.91	-2.02	0.22	-0.95	-1.07	-0.07	0.02
	BiLSTM	2.47	2.89	2.00	2.69	1.15	1.2	-1.99	1.52	0.76	2.9	1.79	2.15	2.48	2.91
p-value		**	*	***	***	***	**	***	**	***	**	***	***	**	—
effect size		0.58	0.40	0.55	0.52	0.56	0.36	0.62	0.32	0.53	0.52	0.47	0.59	0.32	—

1. *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$, - $p > 0.05$.
2. Effect size is computed according to Cliff's delta.

the test cases selected by DeepVec possess a stronger fault detection capability and can guide the retraining of RNNs to enhance robustness. Compared to DeepState's practice of using the output label of RNN, DeepVec measures the uncertainty of a test case from the state-vector at each time step, fully utilizing the intermediate state data of the RNN model. Compared to neuron coverage, DeepVec better captures the state transitions of RNNs. Moreover, by combining the advantages of both CTM and CAM, DeepVec makes a more detailed selection for test cases with similar behaviors. By utilizing the Changing Similarity metric, it minimizes the potential faults brought by the set construction characteristic of DeepState, aligning with CAM's intent to trigger more behavioral states in RNNs. Meanwhile, the sorting process based on Distance Uncertainty takes reference from the advantages of CTM. Therefore, the test set selected by DeepVec is diverse and representative, fundamentally enhancing its performance.

B. Threats to Validity

1) *Internal Validity: Parameters settings.* Although we validate the sensitivity of the angle threshold in our sensitivity analysis, it is undeniable that the performance of DeepVec varies with the setting of the angle threshold, especially when the classification task involves a large number of target classes. We recommend that readers use the parameter settings suggested in Section III-B, as they align with the general understanding of probability distribution vectors and have been thoroughly validated through extensive experiments.

2) *External Validity: Test Objects.* The choice of datasets and RNN models are among the primary threats to their effectiveness. RNNs have various structures, and some differences in model structure cannot be completely ignored. Additionally, the quality of the datasets will directly affect the training results of the model, then impacting the model's accuracy and robustness.

To mitigate this threat, we employed a combination of eight datasets and RNNs to evaluate DeepVec.

Generated Datasets. Datasets originated in the real world are often complex and varied. During data augmentation, we only employed specific methods, and we cannot ensure that the noise contained in real unknown input data is distributed the same way as the noise we simulated.

C. Comparison with Adversarial Training Methods

Adversarial training aims to achieve good accuracy with fewer adversarial training samples generated by adversarial algorithm [69], which is an indispensable part of the DNN testing field. It is undeniable that adversarial attack algorithms represented by GAN [70] have shown excellent performance in test case generation tasks, including FGSM [71] and PGD [72]. Thus, both adversarial training and test case selection attempt to solve the problem of retraining the model through test cases to improve robustness.

Although they have the similar core purpose and motivation, it is worth emphasizing that there are inherent differences between test selection and adversarial training. The key idea of adversarial training is to improve the robustness of DNNs by considering adversarial samples during the training phase. Unlike adversarial training, test case selection acts on DNNs that have been working in actual application scenarios for some time (i.e. a pre-trained model). Test case selection aims to select a small portion of data that can trigger potential faults from the large-scale unlabeled datasets collected during the DNN work process. Using this subset for DNN retraining can effectively reduce the consumption and cost of training resources and improve the robustness of DNN.

To fairly evaluate the ability of both methods in fault detection for DNN models, we generate tests using adversarial training with the same experimental setup as in RQ4. The number of generated tests is equal to 15% of the candidate set.

We consider comparison with DRFuzz [73] and SN-GAN [74]. DRFuzz is the state-of-the-art GAN-based fidelity assurance mechanism that filters mutated inputs. SC-GAN is a seminal generation method used to generate adversarial test samples. We use the default configure of these methods to get tests. The reason of choosing DRFuzz is this technique is originally proposed for regression fuzzing tests and is able to transfer to the test selection task. Table XIII presents the accuracy and improvement after retraining the RNN with the same number of tests. The numbers in the table represent the accuracy of the retrained model, and the numbers in parentheses indicate the improvement in accuracy. Result shows that all these methods can help DNN fault detection. Tests generated by SN-GAN demonstrate stronger fault detection capability on Mnist compared to DRFuzz, but perform weaker on Fashion and SVHN. This may be because SN-GAN cannot guarantee that the generated tests are of high quality and capable of triggering incorrect behaviors in DNNs. In contrast, DRFuzz mitigates this issue to some extent through its fidelity assurance mechanism. The tests selected by DeepVec achieve the strongest fault detection capability, likely because DeepVec filters out more invalid test cases from the test pool and retains more fault-revealing tests. The superior performance of DeepVec suggests that while using GANs to generate tests can help DNN fault detection, it should be combined with test selection to achieve higher fault detection performance.

Table XIII

THE IMPROVEMENT IN RNN ACCURACY ON THE MIXED TEST DATASET USING 15% OF THE TEST SET SIZE AUGMENTED DATA FOR TRAINING.

Dataset	Model	DRFuzz	SN-GAN	DeepVec
Mnist	LSTM	70.67 (5.99%)	77.11 (12.44%)	77.18 (12.5%)
	GRU	69.79 (6.41%)	72.89 (9.5%)	75.39 (12.01%)
	BiLSTM	70.48 (6.12%)	70.03 (5.67%)	74.23 (9.87%)
Fashion	LSTM	71.32 (3.79%)	70.56 (3.03%)	73.99 (6.46%)
	GRU	70.61 (2.26%)	70.99 (2.64%)	72.3 (3.96%)
	BiLSTM	70.17 (3.67%)	68.96 (2.46%)	71.21 (4.72%)
SVHN	LSTM	50.85 (2.07%)	51.12 (2.34%)	51.63 (2.85%)
	GRU	52.53 (1.62%)	52.38 (1.47%)	52.87 (1.96%)
	BiLSTM	47.0 (0.92%)	46.92 (0.84%)	46.25 (0.17%)

Furthermore, it is important to distinguish DeepVec from other methods that rely on uncertainty and similarity metrics, as well as from the regularizers [30] used in adversarial training. In adversarial training, researchers often add penalty items to measure the distribution of neurons (such as KL divergence) to ensure that the trained DNN can present a richer activation state. However, the uncertainty mentioned in DeepVec is only the confidence of DNN in a certain test case, while the similarity is the similarity between each test case, which is used to find the diverse failures generated by DNN.

D. Novelty of DeepVec

Although uncertainty-based test case selection techniques have achieved significant success [19], [21], most of the work has focused primarily on improving the performance of CNNs in image classification tasks [20], [25], [45]. Some of these approaches utilize image-specific features, such as the P-hash algorithm for preprocessing images [19], which makes them unsuitable for RNN-based text classification tasks. The primary innovation of DeepVec lies in its ability to draw on advanced uncertainty metrics from the image

classification domain while introducing a more precise and general similarity metric that allows it to be adapted for both text and image tasks. Table XIV presents the comparative experimental results between DeepVec and the current state-of-the-art method, RTS [19], on image classification tasks under the experimental setup described in Section IV-D. The results indicate that DeepVec performs nearly on par with RTS in improving DNN performance. This demonstrates that DeepVec does not expense image classification performance to enhance text classification tasks but rather possesses broader general applicability. Additionally, DeepVec's innovation is also reflected in its high cost-effectiveness. By sacrificing a small amount of fault diversity, it significantly reduces time costs, making DeepVec more suitable for practical application scenarios.

Table XIV

PERFORMANCE COMPARISON BETWEEN DEEPVEC AND SOTA METHOD ON IMAGE CLASSIFICATION TASKS (BDR INDICATES *Bug Detection Rate* AND INC INDICATES *Inclusiveness*).

Method		RTS	DeepVec	RTS	DeepVec	RTS	DeepVec	RTS	DeepVec	
Model	Sel	BDR		Inc-ori		Inc-aug		Acc Imp		
Mnist	LSTM	10%	0.72	0.82	0.22	0.24	0.21	0.66	8.94%	13.97%
		20%	0.56	0.79	0.39	0.46	0.41	0.78	9.63%	17.31%
	GRU	10%	0.74	0.45	0.21	0.21	0.22	0.48	11.16%	10.38%
		20%	0.61	0.61	0.42	0.43	0.43	0.86	12.39%	12.01%
	BiLSTM	10%	0.69	0.81	0.21	0.23	0.21	0.88	6.60%	7.43%
		20%	0.55	0.75	0.39	0.45	0.42	0.92	10.55%	14.10%
Fashion	LSTM	10%	0.74	0.75	0.25	0.25	0.26	0.53	5.44%	4.63%
		20%	0.70	0.68	0.49	0.47	0.46	0.81	7.71%	6.13%
	GRU	10%	0.70	0.70	0.24	0.24	0.25	0.51	5.15%	4.48%
		20%	0.69	0.63	0.48	0.45	0.44	0.80	6.33%	6.33%
	BiLSTM	10%	0.78	0.74	0.25	0.24	0.25	0.47	5.96%	6.76%
		20%	0.74	0.66	0.49	0.46	0.47	0.73	8.11%	7.15%
SVHN	LSTM	10%	0.75	0.79	0.20	0.21	0.21	0.43	2.62%	0.92%
		20%	0.74	0.77	0.41	0.41	0.42	0.70	1.94%	2.85%
	GRU	10%	0.75	0.78	0.21	0.22	0.22	0.44	1.89%	2.13%
		20%	0.76	0.76	0.42	0.43	0.43	0.71	2.85%	1.96%
	BiLSTM	10%	0.87	0.86	0.20	0.20	0.19	0.20	1.03%	1.67%
		20%	0.86	0.87	0.40	0.40	0.39	0.40	1.94%	1.31%

VII. RELATED WORK

In this section, we focus on some work related to this paper, including testing technology for RNN and the development of test case selection techniques.

A. Recurrent Neural Networks Testing

Inspired by coverage in traditional software testing, researchers have proposed various neuron coverages to test RNNs to ensure their effectiveness. DeepStellar [41], drawing from DeepGauge's [14] five coverage metrics, refines them for RNN evaluation, employing a Discrete-Time Markov Chain (DTMC) [75] for abstract representation. In parallel, TestRNN [42], [43] formulates neuron coverage metrics for LSTM architectures and devises a fuzzing technique enhanced with randomized mutations. RNN-Test [44] measures hidden state coverage based on the peak value attainment ratio during testing. Additionally, DeepState [12] assesses hidden state dynamics in RNNs, introducing a test case selection strategy rooted in inter-case relationships.

Inspired by DeepState, DeepVec starts from the vector space formed by the state-vectors of RNN, and analyzes the uncertainty at each time step, effectively selecting test cases from a large-scale unlabeled test set. In addition, DeepVec measures the similarity produced by different test cases within the RNN behavior quantitatively, making the selected tests more representative.

B. Test Case Selection Techniques

Test case selection techniques are initially proposed in traditional regression testing [37], [76]–[78], aiming to find an appropriate test order in situations with limited resources and time, software testers could achieve the maximum benefit to enhance test efficiency. Subsequently, Rothermel et al. [37] enrich this technique in a more general context. They construct control flow graphs for the original program and subsequently modified programs, and then propose a safe test case selection method based on the control flow graphs. This graph-based method can execute code modified by test cases, ensuring traditional software quality. HyRTS pioneers a hybrid regression test selection method [78]. It parses programs separately at multiple granularities and combines the advantages of multiple traditional test case selection methods.

In the realm of DNN testing, active learning [79] has surfaced as a method that leverages interactive queries to information sources for annotating new data points during the training of DNN models [80]. The predominant query paradigm, uncertainty sampling [81], endeavors to select unlabeled instances that pose the greatest classification challenge for the DNN. The degree of this uncertainty can be ascertained through methodologies like conditional random fields [76], margin-based approaches [82], and entropy [83] calculations.

In recent years, test case selection techniques have achieved significant success in FNNs. Feng et al. proposed DeepGini [20], which prioritizes tests based on the Gini index of the probability distribution vector output by DNNs from a statistical perspective. Gao et al. introduced ATS [21], which incorporates the concept of fault modes based on the uncertainty of unlabeled test inputs and fault directions, and proposed a fitness measure to evaluate the pattern difference between a test input x and the selected set S . Wang et al. introduced PRIMA [24], based on the idea that test inputs capable of killing many mutant models and producing different prediction results through many mutant inputs are more likely to reveal DNN errors. These methods are primarily designed to improve the accuracy of CNNs and have shown limited performance on RNNs. Moreover, Aghababaeian et al. proposed DeepGD [25], which employs a genetic algorithm to perform multi-objective search to find input images with high uncertainty scores. Sun et al. proposed RTS [19], which measures the similarity between test inputs using the P-hash algorithm to achieve better selection results. These methods utilize certain image-specific characteristics, such as P-hash and Geometric Diversity, which cannot be adapted to text datasets.

In the field of RNN testing, DeepState [12] has been proposed based on the stateful view of RNN to select test cases and achieve the goal by capturing the changes in the internal state of RNN. RNNtcs [33], based on the structural characteristics of RNNs, proposed a test case selection method that combines clustering and uncertainty, achieving more diversified test case selection.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we propose DeepVec to select test cases from a large unlabeled dataset to enhance the robustness of RNNs.

We first unfold the predicted probabilities of each category of the RNN model at each time step. Relying on the state-vector space of the RNN, we analyze the uncertainty level of RNN for each test case and the similarity between different test cases. By computing the Distance Uncertainty and Changing Similarity within the RNN's state-vector space, we have designed and implemented a test case selection method. The experimental findings demonstrate DeepVec's proficiency in discovering a test case with a notably high bug detection rate. DeepVec succeeds in helping testers improve the testing and enhancing efficiency of RNN by reducing retraining costs and reducing data annotation costs. In addition, using test cases with higher fault detection capabilities selected by DeepVec for RNN retraining can improve the quality and robustness of the RNN model. In the future, we plan to extend DeepVec to more deep learning models such as transformer-based models and large language models. We also intend to explore the behavioral characteristics exhibited by models on test cases from other views, such as statistics, to more precisely select representative test cases.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China (No. 62372071 and No. 62202074), the Chongqing Technology Innovation and Application Development Project (No. CSTB2022TIAD-STX0007 and No. CSTB2023TIAD-STX0025), the State Key Laboratory of Intelligent Vehicle Safety Technology (No. IVSTSKL-202412), the Scientific and Technological Research Program of Chongqing Municipal Education Commission (No. KJQN202300547) and the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant (Project ID: 23-SOL-SMU-004). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.

REFERENCES

- [1] D. Lu and Q. Weng, "A survey of image classification methods and techniques for improving classification performance," *Int. J. Remote Sens.*, vol. 28, no. 5, pp. 823–870, Jan 2007.
- [2] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "Achieving human parity in conversational speech recognition," *arXiv preprint arXiv:1610.05256*, 2016.
- [3] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.
- [4] Q. Chen, X. Zhu, Z. Ling, S. Wei, H. Jiang, and D. Inkpen, "Enhanced lstm for natural language inference," *arXiv preprint arXiv:1609.06038*, 2016.
- [5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
- [6] A. M. Seras, J. D. Ser, and P. Garcia-Bringas, "Efficient object detection in autonomous driving using spiking neural networks: Performance, energy consumption analysis, and insights into open-set object discovery," 2023.
- [7] BBC News, "Amazon promises fix for creepy alexa laugh," 2018, accessed on 09/06/2023. [Online]. Available: <https://www.bbc.com/news/technology-43325230>
- [8] J. Deriu, A. Rodrigo, A. Otegi, G. Echegoyen, S. Rosset, E. Agirre, and M. Cieliebak, "Survey on evaluation methods for dialogue systems," *Artificial Intelligence Review*, pp. 1–56, 2020.

- [9] Đorđe Petrović, R. Mijailović, and D. Pešić, "Traffic accidents with autonomous vehicles: Type of collisions, manoeuvres and errors of conventional vehicles' drivers," *Transportation Research Procedia*, vol. 45, pp. 161–168, 2020, transport Infrastructure and systems in a changing world. Towards a more sustainable, reliable and smarter mobility.TIS Roma 2019 Conference Proceedings. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2352146520301654>
- [10] National Highway Traffic Safety Administration, "Automated vehicle safety," 2023. [Online]. Available: <https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety>
- [11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [12] Z. Liu, Y. Feng, Y. Yin, and Z. Chen, "Deepstate: Selecting test suites to enhance the robustness of recurrent neural networks," in *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, 2022, pp. 598–609.
- [13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 248–255.
- [14] L. Ma, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, T. Su, L. Li, Y. Liu, J. Zhao, and Y. Wang, "Deepgauge: Multi-granularity testing criteria for deep learning systems," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ser. ASE 2018. Montpellier, France: Association for Computing Machinery, 2018, pp. 120–131.
- [15] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," *Commun. ACM*, vol. 62, no. 11, pp. 137–145, Oct 2019.
- [16] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th International Conference on Software Engineering*, ser. ICSE '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 303–314.
- [17] Q. Hu, L. Ma, X. Xie, B. Yu, Y. Liu, and J. Zhao, "Deepmutation++: A mutation testing framework for deep learning systems," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE/ACM, 2019.
- [18] L. Ma, F. Zhang, J. Sun, M. Xue, B. Li, F. Juefei-Xu, C. Xie, L. Li, Y. Liu, J. Zhao *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.
- [19] W. Sun, M. Yan, Z. Liu, and D. Lo, "Robust test selection for deep neural networks," *IEEE Transactions on Software Engineering*, vol. 49, no. 12, pp. 5250–5278, 2023.
- [20] Y. Feng, Q. Shi, X. Gao, J. Wan, C. Fang, and Z. Chen, "Deepgini: Prioritizing massive tests to enhance the robustness of deep neural networks," in *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2020. New York, NY, USA: Association for Computing Machinery, 2020, pp. 177–188.
- [21] X. Gao, Y. Feng, Y. Yin, Z. Liu, Z. Chen, and B. Xu, "Adaptive test selection for deep neural networks," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 73–85.
- [22] N. K. Kaur, U. Kaur, and D. Singh, "K-medoid clustering algorithm-a review," *Int. J. Comput. Appl. Technol.*, vol. 1, no. 1, pp. 42–45, 2014.
- [23] B. Kim, R. Khanna, and O. O. Koyejo, "Examples are not enough, learn to criticize! criticism for interpretability," *Advances in neural information processing systems*, vol. 29, 2016.
- [24] Z. Wang, H. You, J. Chen, Y. Zhang, X. Dong, and W. Zhang, "Prioritizing test inputs for deep neural networks via mutation analysis," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 397–409.
- [25] Z. Aghababaeian, M. Abdellatif, M. Dadkhah, and L. Briand, "Deepgd: A multi-objective black-box test selection approach for deep neural networks," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 6, pp. 1–29, 2024.
- [26] D. Svozil, V. Kvasnicka, and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemometrics and intelligent laboratory systems*, vol. 39, no. 1, pp. 43–62, 1997.
- [27] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology*, ser. ICET, 2017, pp. 1–6.
- [28] P. Petersen and F. Voigtlaender, "Equivalence of approximation by convolutional neural networks and fully-connected networks," *Proc. Amer. Math. Soc.*, vol. 148, no. 4, pp. 1567–1581, 2020.
- [29] F. Harel-Canada, L. Wang, M. A. Gulzar, Q. Gu, and M. Kim, "Is neuron coverage a meaningful measure for testing deep neural networks?" in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2020. New York, NY, USA: Association for Computing Machinery, 2020, pp. 851–862.
- [30] Z. Li, X. Ma, C. Xu, and C. Cao, "Structural coverage criteria for neural networks could be misleading," in *Proceedings of the 41st International Conference on Software Engineering: New Ideas and Emerging Results*, ser. ICSE-NIER '19. IEEE Press, 2019, pp. 89–92.
- [31] S. Yan, G. Tao, X. Liu, J. Zhai, S. Ma, L. Xu, and X. Zhang, "Correlations between deep neural network model coverage criteria and model quality," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2020. New York, NY, USA: Association for Computing Machinery, 2020, pp. 775–787.
- [32] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2805–2824, 2019.
- [33] X. Wu, J. Shen, W. Zheng, L. Lin, Y. Sui, and A. O. A. Semasaba, "Rnntcs: A test case selection method for recurrent neural networks," *Knowledge-Based Systems*, vol. 279, p. 110955, 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0950705123007050>
- [34] A. N. Uma, T. Fornaciari, D. Hovy, S. Paun, B. Plank, and M. Poesio, "Learning from disagreement: A survey," *Journal of Artificial Intelligence Research*, vol. 72, pp. 1385–1470, 2021.
- [35] B. Wu, Y. Li, Y. Mu, C. Scarton, K. Bontcheva, and X. Song, "Don't waste a single annotation: Improving single-label classifiers through soft labels," *arXiv preprint arXiv:2311.05265*, 2023.
- [36] Z. Li, X. Ma, C. Xu, C. Cao, J. Xu, and J. Lü, "Boosting operational dnn testing efficiency through conditioning," ser. ESEC/FSE 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 499–509. [Online]. Available: <https://doi.org/10.1145/3338906.3338930>
- [37] G. Rothermel and M. J. Harrold, "Analyzing regression test selection techniques," *IEEE Transactions on Software Engineering*, vol. 22, no. 8, pp. 529–551, 1996.
- [38] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020.
- [39] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [40] T. Mikolov, S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur, "Extensions of recurrent neural network language model," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 5528–5531.
- [41] X. Du, X. Xie, Y. Li, L. Ma, Y. Liu, and J. Zhao, "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 (ESEC/FSE 2019)*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 477–487.
- [42] W. Huang, Y. Sun, X. Huang, and J. Sharp, "testrnn: Coverage-guided testing on recurrent neural networks," *arXiv preprint arXiv:1906.08557*, 2019.
- [43] W. Huang, Y. Sun, X. Zhao, J. Sharp, W. Ruan, J. Meng, and X. Huang, "Coverage-guided testing for recurrent neural networks," *IEEE Transactions on Reliability*, pp. 1–16, 2021.
- [44] M. Guo, Y. Zhao, X. Han, Y. Jiang, and J. Sun, "Rnntest: Adversarial testing framework for recurrent neural network systems," *arXiv preprint arXiv:1911.06155*, 2019.
- [45] Q. Hu, Y. Guo, X. Xie, M. Cordy, L. Ma, M. Papadakis, and Y. Le Traon, "Test optimization in dnn testing: a survey," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 4, pp. 1–42, 2024.
- [46] Y. LeCun, C. Cortes, and C. J. Burges, "Mnist," <http://yann.lecun.com/exdb/mnist/>, 1998.
- [47] K. Binmore and J. Davies, *Calculus: Concepts and Methods*. Cambridge University Press, 2002.
- [48] K. Rasul and H. Xiao, "Fashion," <https://research.zalando.com/welcome/mission/research-projects/fashion-mnist/>, 2017.
- [49] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril *et al.*, "Snips voice platform: An embedded spoken language understanding system for private-by-design voice interfaces," *arXiv preprint arXiv:1805.10190*, 2018.
- [50] "Ag's corpus of news articles," http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html, accessed on 09/22/2023.
- [51] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proceedings of the 28th International*

- Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 649–657.
- [52] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng *et al.*, “Reading digits in natural images with unsupervised feature learning,” in *NIPS workshop on deep learning and unsupervised feature learning*, vol. 2011, no. 2. Granada, 2011, p. 4.
- [53] R. Dey and F. M. Salem, “Gate-variants of gated recurrent unit (gru) neural networks,” in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2017, pp. 1597–1600.
- [54] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.
- [55] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional lstm networks,” in *2005 IEEE International Joint Conference on Neural Networks*, vol. 4, 2005, pp. 2047–2052.
- [56] C. Olah, “Understanding lstm networks,” 2015.
- [57] T. O’Malley, E. Bursztin, J. Long, F. Chollet, H. Jin, L. Invernizzi *et al.*, “Keras tuner,” <https://github.com/keras-team/keras-tuner>, 2019.
- [58] E. Ma, “Nlp augmentation,” <https://github.com/makcedward/nlpaug>, 2019.
- [59] A. Mikołajczyk and M. Grochowski, “Data augmentation for improving deep learning in image classification problem,” in *2018 International Interdisciplinary PhD Workshop (IIPHDW)*. IEEE, 2018, pp. 117–122. [Online]. Available: <https://doi.org/10.1109/IIPHDW.2018.8388338>
- [60] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.
- [61] R. Sennrich, B. Haddow, and A. Birch, “Improving neural machine translation models with monolingual data,” *arXiv preprint arXiv:1511.06709*, 2015.
- [62] S. Kobayashi, “Contextual augmentation: Data augmentation by words with paradigmatic relations,” *arXiv preprint arXiv:1805.06201*, 2018.
- [63] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *arXiv preprint arXiv:1910.10683*, 2019.
- [64] S. Yoo and M. Harman, “Regression testing minimization, selection and prioritization: A survey,” *Software Testing, Verification and Reliability*, vol. 22, no. 2, pp. 67–120, Mar 2012.
- [65] T. Y. Chen, R. Merkel, P. Wong, and G. Eddy, “Adaptive random testing through dynamic partitioning,” in *Fourth International Conference on Quality Software, 2004. QSIC 2004. Proceedings*. IEEE, 2004, pp. 79–86.
- [66] “Python release python 3.7.16,” <https://www.python.org/downloads/release/python-3716/>, accessed on 09/22/2023.
- [67] F. Chollet *et al.*, “Keras,” <https://keras.io>, 2015.
- [68] “Tensorflow,” <https://www.tensorflow.org/>, accessed on 09/22/2023.
- [69] T. Miyato, A. M. Dai, and I. Goodfellow, “Adversarial training methods for semi-supervised text classification,” *arXiv preprint arXiv:1605.07725*, 2016.
- [70] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.
- [71] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *arXiv preprint arXiv:1412.6572*, 2014.
- [72] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” *arXiv preprint arXiv:1706.06083*, 2017.
- [73] H. You, Z. Wang, J. Chen, S. Liu, and S. Li, “Regression fuzzing for deep learning systems,” in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2023, pp. 82–94.
- [74] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral normalization for generative adversarial networks,” *arXiv preprint arXiv:1802.05957*, 2018.
- [75] J. R. Norris and J. R. Norris, *Markov Chains*. Cambridge University Press, 1998.
- [76] T. L. Graves, M. J. Harrold, J.-M. Kim, A. Porter, and G. Rothermel, “An empirical study of regression test selection techniques,” *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 10, no. 2, pp. 184–208, apr 2001.
- [77] G. Rothermel and M. J. Harrold, “A safe, efficient regression test selection technique,” *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 6, no. 2, pp. 173–210, apr 1997.
- [78] L. Zhang, “Hybrid regression test selection,” in *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, 2018, pp. 199–209.
- [79] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang, “A survey of deep active learning,” *ACM Computing Surveys (CSUR)*, vol. 54, no. 9, pp. 1–40, 2021.
- [80] B. Settles, “Active learning literature survey,” 2009, technical report, University of Wisconsin-Madison.
- [81] D. D. Lewis and W. A. Gale, “A sequential algorithm for training text classifiers,” in *SIGIR’94*. Springer, 1994, pp. 3–12.
- [82] A. J. Joshi, F. Porikli, and N. Papanikolopoulos, “Multi-class active learning for image classification,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 2372–2379.
- [83] A. Holub, P. Perona, and M. C. Burl, “Entropy-based active learning for object recognition,” in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2008, pp. 1–8.