2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

A New Perspective of Deep Learning Testing Framework: Human-Computer Interaction Based Neural Network Testing

Wei Kong[1], Hu Li[1], Qianjin Du[2], Huayang Cao[1] and Xiaohui Kuang[1]

Abstract— **Deep learning models have revolutionized various domains but have also raised concerns regarding their security and reliability. Adversarial attacks and coverage-based testing have been extensively studied to assess and enhance the dependability of deep neural networks. However, current research in this area has reached a state of stagnation. Adversarial attacks focus on exploiting vulnerabilities in models, while coverage-based testing aims to achieve comprehensive testing but overlooks application scenarios. Moreover, evaluating test cases solely based on their fault-revealing capability is insufficient. To address these limitations, we propose an innovative interdisciplinary framework that incorporates human-computer interaction methods in deep learning security testing. By considering the attributes of model application scenarios, we can design more effective test suites that intend to reveal the model's behavior across various scenarios, aiding in the identification of potential defects. Consequently, the test suite plays a crucial role in the testing process of deep learning models, contributing to the assurance of model robustness and reliability. Additionally, we establish a comprehensive evaluation metric for test suite quality, considering factors such as diversity and naturalness. This framework promotes reliable and secure deployment of deep learning models, fostering interdisciplinary collaboration between artificial intelligence and human-computer interaction.**

I. INTRODUCTION

Deep neural networks have gained widespread traction across diverse domains [1], [2]. However, concomitant with their pervasive adoption, concerns pertaining to the security of deep learning models have garnered significant attention within both industry and academia. These concerns encompass the models' susceptibility to adversarial perturbations and their inherent opacity, thereby impeding their practical utility within safety-critical and high-stakes contexts. As a result, Deep learning security testing has emerged as a prominent research area, with the objective of guaranteeing the dependability and trustworthiness of models. However, existing research primarily centers around employing adversarial attacks and devising coverage criteria to assess deep neural networks. Adversarial attacks involve deliberately manipulating input data to exploit vulnerabilities within deep learning models. By carefully perturbing the input, attackers can mislead the model and induce inaccurate predictions. Ongoing research [3], [4] endeavors focus on developing robust models capable of withstanding such attacks and enhancing the model's resilience against adversarial examples. Conversely, coverage criteria [5], [6] are employed to quantify the extent to which different components of a deep learning model have been subjected to testing. Note that different components mainly refer to the neurons of the model that have been activated in the hidden layers. These criteria aim to ensure a comprehensive understanding of the model's behavior and facilitate the identification of potential vulnerabilities or errors. By formulating comprehensive coverage metrics, researchers aim to enhance the effectiveness of testing methodologies and augment the overall reliability of deep learning models.

However, the current research in deep learning security testing appears to have reached a state of stagnation. On one hand, theoretical analyses have demonstrated the severe vulnerabilities of existing models when confronted with powerful adversarial attacks. However, it would be incorrect to conclude that deep learning is entirely unsuitable for real-life applications. On the other hand, coverage-based testing has been striving to find more comprehensive coverage metrics. However, both adversarial-based [7], [8] and coverage-based [9], [10] test case generation methods have overlooked the model's application scenarios. Furthermore, these automated testing approaches can effectively identify test cases that can cause model misjudgments without human assistance. However, evaluating test cases solely based on their fault-revealing capability is unscientific. Therefore, there is a need to establish a comprehensive evaluation metric for test case quality.

To address the aforementioned limitations and foster interdisciplinary collaboration between artificial intelligence and human-computer interaction, innovative approaches to deep learning security testing need to be developed. One potential solution is to leverage human-computer interaction methods to identify the attributes of model application scenarios and establish a more comprehensive and context-aware evaluation metric for test case quality. To be specific, in the pursuit of comprehensive coverage metrics, researchers should consider the application scenarios of the model. Understanding the specific context in which the model will be deployed is crucial for designing effective and relevant test suites. Moreover, evaluating test case quality should go beyond just error-revealing capability. Factors such as diversity should be taken into account when assessing the quality of test cases. Establishing a holistic evaluation framework that incorporates these aspects would enable a more comprehensive and scientifically sound assessment of test case quality.

*This work was supported by National Key Laboratory of Science and Technology on Information System Security

[1]Wei Kong, Hu Li, Huayang Cao and Xiaohui Kuang are with National Key Laboratory of Science and Technology on Information System Security, Beijing, China [1] Kong_Wei@ieee.org,lihu_lh@126.com, chystudy@foxmail.com, xiaohui_kuang@126.

[2] Qianjin Du is withDepartment of Computer Science and Technology, Tsinghua University, Beijing, China dqj20@mails.tsinghua.edu.cn

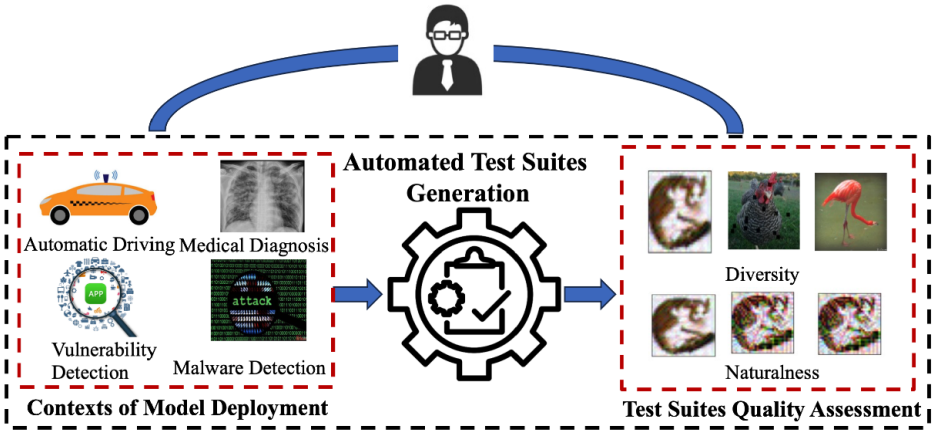

Fig. 1. Two aspects are overlooked in deep learning model testing

In conclusion, by leveraging human-computer interaction methods to identify model application scenario attributes and establishing a comprehensive and context-aware evaluation metric for test suite quality, we can overcome limitations in deep learning security testing. The interdisciplinary collaboration between various fields is crucial in developing innovative approaches that ensure the reliable and secure deployment of deep learning models in real-world applications. Overall, our paper makes the following contributions:

- We propose an innovative, interdisciplinary framework for deep learning model testing based on human-computer interaction. This framework aims to ensure the secure deployment of deep learning models in real-world applications.
- We utilize human-computer interaction methods to identify the attributes of model application scenarios. By doing so, we can design coverage operations that are more likely to induce model misjudgments. This enhances the effectiveness of our testing approach.
- We establish a comprehensive evaluation metric for test suite quality, incorporating the insights gained from human-computer interaction. This metric goes beyond the fault-revealing capability and takes into account factors such as diversity and naturalness. This provides a more holistic assessment of the quality of test suites

II. Motivation

To address the problem of test case generation and evaluation matching with application contexts, a simple but powerful observation is to find the more plausible and potentially efficient direction of designing coverage criterion and assessment metrics from the perspective of requirements. Therefore, we propose a novel structural testing approach for DL models deployed in different contexts. As illustrated in Fig.2, our proposed research sketch aims to establish structural coverage criteria and comprehensive assessment metrics by incorporating human expertise. In general, we divide the DL testing process into three key stages as outlined below:

- **Identify the properties of DL pipeline context by human-computer interaction.** At a high level, the DL pipeline can be divided into data sourcing/labeling, data training, and model prediction. Based on the specific context, the DL pipeline will possess particular properties. We then formulate the pipeline structural coverage criteria on the basis of understanding the challenging properties of the DL pipeline.
- **Automated test suites generation algorithm.** To mitigate the issue of enormous test space, our algorithm automatically selects pipeline structural coverage criteria that can trigger a diverse range of defects.
- **Comprehensive metrics to assess test suites by human-computer interaction.** More comprehensive test metrics needed to be formulated to assess the quality of test suites by human expertise, which should fit well with the characteristic of the DL pipeline.

III. Methodology

A. Human-Computer Interaction based Pipeline Properties Identification

Indeed, aligning the context of DL pipeline deployment with its specific characteristics is crucial for achieving effective coverage of incorrect outputs. In the study by Schlogl et al. [11], a method is proposed to determine the deployment context of a DL pipeline based on the training set and observable outputs. The method involves inferring the properties of the deep learning pipelines by tracing feature numerical deviations in observable outputs. For example, the intrusion detection model is used to predict intrusion in your network traffic. We can deploy the same intrusion detection model in multiple executable environments that change over time. We can observe that the system needs to periodically re-train otherwise it will flag all network traffic as anomalous. Based on the model's behavior in different execution environments, the analyst can manually infer and identify the particular characteristics of the DL pipeline within that context, as shown in Table I. More specifically, in the context of a DL pipeline used for biomedical image diagnosis, data labeling can be a challenging task that requires specialized knowledge and significant time investment. On the other hand, in the context of vulnerability detection, the

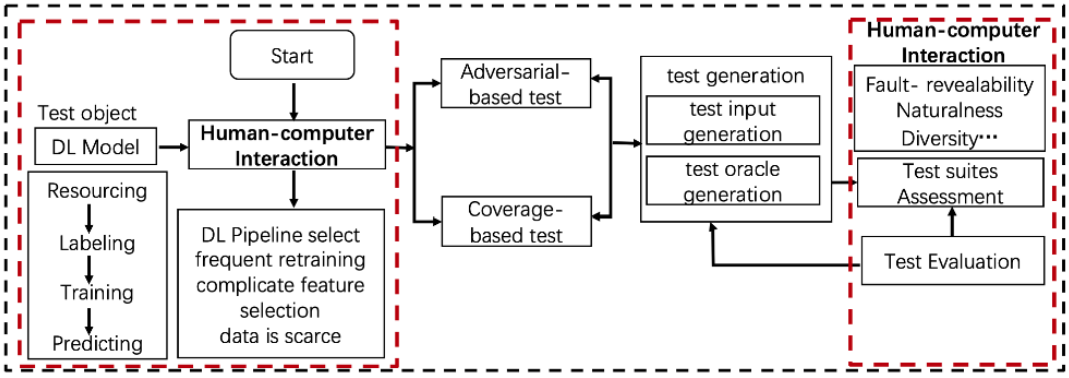

Fig. 2. Architecture of our proposed scheme

source code is transformed into structures such as Abstract Syntax Tree and Call Graph, which are then mapped into embedding vectors using techniques like Code2Vec [12]. This process involves complex feature selection, although some useful features may be flattened in the end. Therefore, it can be classified as a pipeline with complicated feature selection. In the case of malware detection, the performance of the classifier can significantly degrade or age over time due to the evolution of malware. Consequently, this type of pipeline requires frequent re-training to maintain its effectiveness. Therefore, understanding the specific properties of the pipeline becomes crucial, and relying solely on model structural coverage criteria (e.g., Neuron Coverage and its variants) may not be sufficient for testing DL systems.

This observation inspired us to design the pipeline structural coverage according to the properties of DL pipeline. We believe it is a plausible direction to trigger more defects by covering these coverage operations. The table.I describes coverage operations in different contexts. Firstly, in the biomedical image diagnosis example where a pipeline with labeling-hard, we select seven representative metamorphic-based transformation operations [13] in medical images. Secondly, for a pipeline with complex feature extraction in the context of vulnerability detection, the code structure can be changed by refactoring operators [14]–[16] while maintaining semantics-invariant. For AST(Abstract Syntax Tree) structure, we extract ten semantic-preserving patterns to produce new the data structure. Thirdly, in the context of malware detection in which the pipeline with frequent re-training, we extract and cover the evaluation rules of Android malware to generate the test suites in feature space.

Besides defining pipeline structural coverage, we combine them with the model internal coverage criteria(e.g. NC, KMNC, NBC). The work selects NC as a backend and introduces it in detail. NC was significant research in deep learning testing, which measures the proportion of activated neurons to the total number of neurons for given test input. Following Reference [17], we adopted their automated test suites generation algorithm and have the following crucial observation: NC has shown a direct correlation with input-output diversity and can guide to systematic test generation.

B. Human-Computer Interaction based Test suites Quality Assessment

While automated testing approaches can identify test cases without human assistance, the involvement of human expertise is crucial for evaluating test case quality. Domain experts and practitioners should be involved in the testing process to provide insights, verify the practicality of test suites, and ensure alignment with real-world scenarios. We recognize that there is no one-size-fits-all approach to assess test suites for all kinds of DL pipelines, and therefore, different pipeline contexts may prioritize different metrics based on their specific challenges and requirements. In scenarios where labeling is difficult, diversity can be a valuable metric. These tasks involve complex real-world challenges and limited training samples. In such cases, the trade-off between task complexity and data scarcity necessitates diverse test suites. On the other hand, in DL pipelines with frequent retraining where test suites remain similar over multiple time points, diversity may not be as critical.

In our work, we propose fault-revealing ability and coverage as common assessment metrics for test suites. Specifically, in the context of labeling-hard pipelines, we employ naturalness measured by Frechet Inception Distance (FID) [18] and diversity assessed using Standard Deviation (STD) [19] to evaluate the quality of test suites. These metrics help determine if the test suites effectively reveal faults and exhibit sufficient diversity. For pipelines involving complex feature selection, we utilize diversity metrics such as p-value and Standard Deviation (STD) to assess if the distribution of test suites significantly differs from the original data. This aids in evaluating the effectiveness of the test suites in capturing the range of features relevant to the pipeline. In the case of pipelines with frequent retraining, we quantify the deviation of test suites from the original ones using p-value [20] due to the continuously evolving Android malware. Additionally, we measure the Feature Space Stability Score (FSPs) [21] to assess the stability of the feature space in the test suites, as the evolved malware tends to retain similar API features. By tailoring the assessment metrics to the specific context of each DL pipeline, we can effectively evaluate the quality and suitability of test suites for addressing the challenges and

TABLE I
PIPELINE STRUCTURAL COVERAGE

DL Pipeline Properties	Context	Coverage Scheme	Coverage Operation
a pipeline with labeling-hard	medical diagnosis	metamorphic-based transformations coverage	horizontal & vertical flip, transpose, random rotate, elastic transform, grid distortion, optical distortion, brightness, contrast
a pipeline with complex feature extraction	Method Name Prediction	Abstract patterns coverage	Local variable renaming/adding, Parameter renaming/adding, Function name renaming, Semantic-equivalent API substitution, Replacement between for and while, Logic-equivalent substitution to IF, Change a return variable
a pipeline with frequent retraining	Malware Detection	Evolution rules coverage	Semantic-equivalent API substitution, API drop out, API Rename, API Reorder, Permission Restatement, Malware Repackage

requirements unique to each scenario.

IV. EMPIRICAL STUDY

The overall goal of our work is to establish DL system test paradigm from the perspective of DL pipeline properties, and prove that our work is plausible and promising research to test the vulnerabilities of DL model.

A. Biomedical Image Diagnosis

We train the LesaNet following the configuration [22] on deeplesion dataset, which is developed for the detection and classification of lesions in medical images. The DeepLesion dataset is a large-scale dataset of medical images that was created specifically for training and evaluating deep learning models. It contains more than 32,000 CT and MRI scans from over 10,000 patients and includes annotations for a wide range of lesion types, including pulmonary nodules, liver tumors, and bone fractures. LesaNet utilizes VGG-16 as the backbone of the network to automatically extract features from medical images, which are then used to classify the images as either normal or abnormal. More specifically, the input of LesaNet is the lesion image patch, then it predicts the image lesion area and obtains the semantic labels. Considering the highly imbalanced and long-tailed labels, we only test the pulmonary nodules classification task of LesaNet in the evaluation process. Note that the LesaNet achieves 98.6% accuracy in classifying the pulmonary nodules. We can refer [22] for a more detailed implementation exposition.

According to the proprieties of medical image diagnosis, we can complete the structural coverage operations in Table .I. Since our target is to exemplify the usefulness of our work, we will not enumerate all metamorphic-based transformation operations. Our proposed operations not only include common augmentations (i.e. Affine, Elastic, Brightness, Contrast), but are also distinctively tailored to the specificities of the medical field (i.e. grid distortion, optical distortion). For generating the test suites, our work selected the metamorphic-based transformations coverage operations to the seed set of pulmonary nodules. As the algorithm iterates, synthetic images that effectively improve the selected coverage (i.e. NC, KMNC) will be given priority as test cases for further evaluation.

B. Method Name Prediction

Method name prediction is a challenging task for the requirement of semantic understanding to code snippets and implicit pattern capture between embedding vector and semantic label, while this model can efficiently improve the performance of downstream tasks such as code search and code comprehension. Normally, the task converts the input AST of code snippets to the real-valued vector representation by Code2vec [12] and finally maps the code vector embedding to the prediction name.

During the model training, we follow the same configuration [12] to train a path-attention network, in which the function of full-connected layer and attention weight are mapping the path contexts to corresponding embedding, and aggregating multiple contexts into a single vector representation. We train the model on java-large dataset [23] which consists of 14 million Java methods from the GitHub codebase. The dataset is split into training, validation, and test sets, with around 12 million methods in the training set, 1 million in the validation set, and 1 million in the test set. Readers can refer to [12] for a detailed exposition of the model architecture and hyper-parameters. For testing the trained model, we leverage the property in the context of method name prediction that inevitable information loss (i.g. semantic information and code snippets structure) in the process of code embedding by code2Vec. Since the above-mentioned features may be flattened, the trained model cannot establish a guaranteed connection between learned embeddings and the corresponding semantic label. Hence, we formalize nine kinds of abstract patterns to transform the code snippets while keeping semantically-equivalent. More specifically, local variable renaming or adding involves changing the names of local variables within the code or introducing new local variables, which is similar to parameter renaming or adding in the function. Function name renaming can alter the function names while maintaining the functionality. Semantic-equivalent API Substitution involves replacing one API call with another that provides similar functionality. The next three abstract patterns are very intuitive including replacement between For and While, return variable changing, and replacing one logical structure with an equivalent 'if' statement. Note that these abstract patterns will cause the change to AST structure or nodes but the output of method name prediction will not substantially differ from the original code snippets.

C. Android Malware Detection

Learning-based malware detection mechanisms aim to categorize input data into either a malicious or benign category. DREBIN is an Android dataset containing 275K

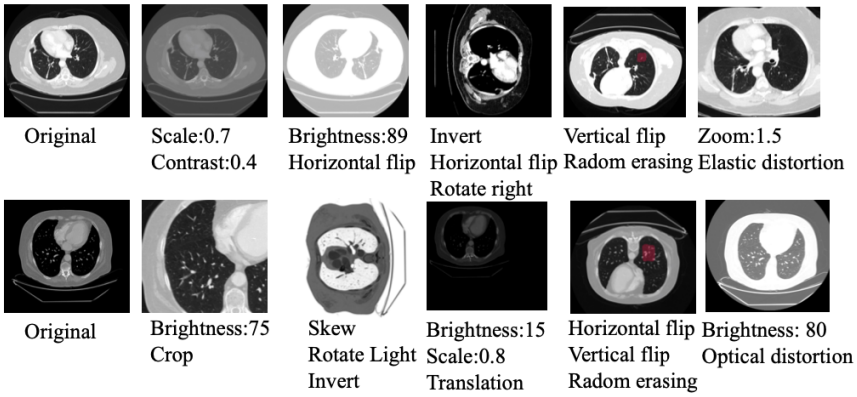

Fig. 3. Transformations-based Medical Images

Android samples we collected from Androzoo [24]. They are relatively recent samples between 2017 to 2020 and labeled by VirusTotal [25]. The dataset is split into a 220K training dataset (120K benign samples and 100K malicious samples) and a 55K testing dataset (30K benign samples and 25K malware samples). To extract DREBIN features, applications are turned into vectors with discrete input values $X \in \{0,1\}^m$ which indicate the presence of certain features (i.e., whether an application uses specific permission or not). DREBIN features use eight feature classes to represent Android applications, such as 1) Permissions and hardware component access requested by each application (e.g. for CAMERA or INTERNET access). 2) Restricted and suspicious (i.e. accessing sensitive data, e.g. getDeviceID()) API-calls made by the applications. 3) Application components such as activities, service, content provider and broadcast receivers used by each application, and 4) Intents used by applications to communicate with other applications. These syntactic features are extracted as many as possible from Manifest file and Dexcode file. Since the immense feature set harms model performance and increases computational complexity, we applied a feature selection based on L1-regularization to reduce the feature set into 1507 features.

In our implementation of Drebin, we utilized Deep Neural Networks (DNN) as the underlying algorithm. The resulting Drebin-NN architecture features four hidden layers, with the first three layers utilizing ReLU activation, Batch Normalization, and a dropout rate of 50%. The final layer leverages a Softmax layer to compute probabilities. Drebin-NN takes as input a vector with 1507 dimension feature space. Intuitively, Android malware keeps evolving over time to avoid being detected by existing classifiers. That is, 1) Android malware keeps evolving with similar functionalities but varied implementations, 2) request different or additional permissions in order to perform new or more advanced malicious actions, 3) modify the information sent to the operating system which makes it appear that the malware is performing a benign activity, and 4) repackage a bonafide benign Android APP to contain malware while retaining most of the benign features. Hence, we extract six evaluation rules of Android malware to generate the test suites in feature space.

For generating test suites, our DREBIN feature set is all editable. We apply the our work algorithm to feature vectors rather than the underlying information sources represented by the features. For example, our algorithm sets the feature value of similar API usages to 0 in the case of semantically equivalent features or removes selected (top) benign features in the case of repackaged malware. When we feed the test sites to DNN model, its output indicates whether the inputs are malicious (output is zero), or benign (output is one).

D. Results and Discussion

To answer whether our scheme can detect diverse erroneous behaviors in DNN model, figure 3 shows that the metamorphic-based transformations can effectively trigger the erroneous behaviors in DL model by combining different transformations. Our finding in Figure 4 proves the conclusion [17] that cumulative transformations can easily detect more erroneous behaviors, in which the error rate significantly ascends as the search times rise from 10.6 % to 36 % on average. Additionally, in the context of Malware detection, the error rate produced by original samples and test suites is from 2.7 % to 32 % on average. The error rate increased from 27.7 % to 48.5 % on average in the context of method name prediction. Furthermore, to answer our scheme can efficiently improve neuron coverage, our experimental result in Figure 4 demonstrates that our work efficiently boosts neuron coverage by increasing the number of search times. In general, we conclude that our work can significantly increase the neuron coverage in the configurations of models and datasets that we used. To answer whether our scheme can improve the quality of test suites under our assessment metrics, we design context-adaptive assessment metrics to measure the quality of test suites in a comprehensive manner. From the experiment result in Figure 5, we clearly get the conclusion that our work can efficiently improve the diversity of the test suite in three contexts, such as STD to LesNet and Path-Attention, and P-value to Drebin-NN. However, the increasing search times are likely to make the generated inputs more unnatural (i.e. FID to LesNet) in the context of medical diagnosis in Figure 5. Note that in the context of Android malware detection,

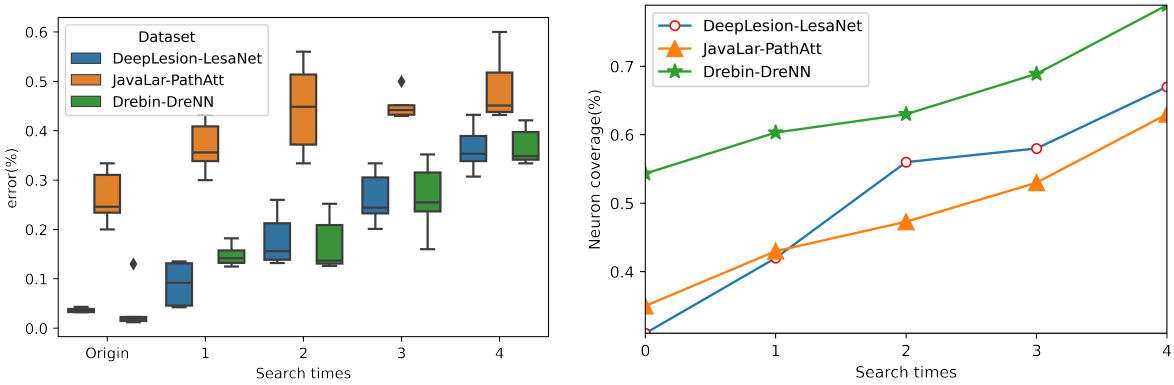

Fig. 4. NC&Error rate change with increasing search times

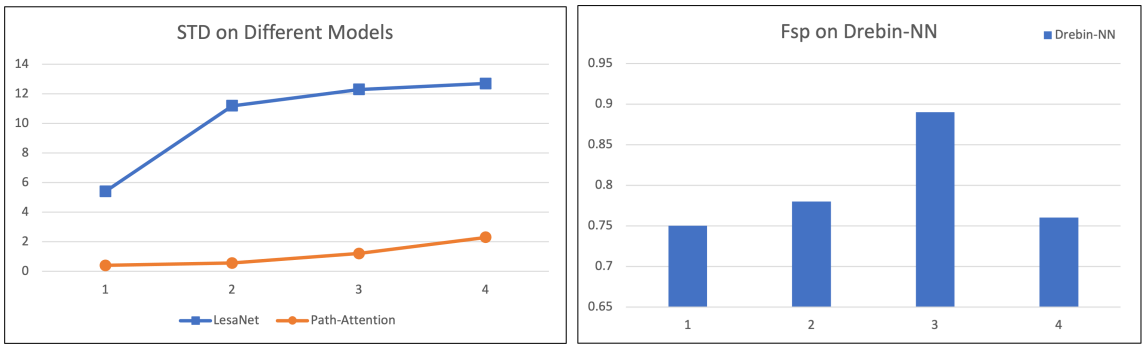

Fig. 5. Test Suites Quality Assessment with Increasing Search Times

We can see that the feature space stability score (FSPs) is very close to 1 in Figure 5 which represents the generated test suites that can help models capture malware evolution.

The responses above aim to demonstrate the effectiveness of deep learning model testing based on human-computer interaction. By utilizing the characteristics of deep learning pipelines to generate test cases, the experimental results demonstrate the enhanced ability to detect model errors and improve coverage metric. The comprehensive test case quality evaluation metrics provide a holistic measure for testing, rather than relying solely on the ability to detect errors. In an unconstrained scenario, generating test cases that solely aim to find model errors may not provide meaningful guidance for testing professionals.

V. Conclusion

Our work contributes to the advancement of deep learning model testing by leveraging human-computer interaction and cross-disciplinary expertise to design a novel framework for deep learning model testing. By combining the insights from the expertise of domain specialists, we have devised a testing framework that considers the specific context in which the model will be deployed. This context-aware approach ensures that the test cases generated align with the real-world application scenarios. Furthermore, our framework incorporates a holistic evaluation scheme for test case quality with limited testing resources. We can assess the quality of test cases more comprehensively and ensure that the test cases reveal the model's undesirable errors efficiently. By integrating these dimensions, our framework ultimately enhances the reliability and trustworthiness of deep learning models in real-world applications.

References

[1] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, pp. 1–74, 2021.

[2] L. Ma, Y. Liu, X. Zhang, Y. Ye, G. Yin, and B. A. Johnson, "Deep learning in remote sensing applications: A meta-analysis and review," *ISPRS journal of photogrammetry and remote sensing*, vol. 152, pp. 166–177, 2019.

[3] T. Bai, J. Luo, J. Zhao, B. Wen, and Q. Wang, "Recent advances in adversarial training for adversarial robustness," *arXiv preprint arXiv:2102.01356*, 2021.

[4] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[5] J. Sekhon and C. Fleming, "Towards improved testing for deep learning," in *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 2019, pp. 85–88.

[6] W. Ma, M. Papadakis, A. Tsakmalis, M. Cordy, and Y. L. Traon, "Test selection for deep learning systems," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 30, no. 2, pp. 1–22, 2021.

[7] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," *arXiv preprint arXiv:1712.04248*, 2017.

[8] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Computer Science Review*, vol. 37, p. 100270, 2020.

[9] L. Ma, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, T. Su, L. Li, Y. Liu *et al.*, "Deepgauge: Multi-granularity testing criteria for deep learning systems," in *Proceedings of the 33rd ACM/IEEE international conference on automated software engineering*, 2018, pp. 120–131.

[10] Y. Sun, M. Wu, W. Ruan, X. Huang, M. Kwiatkowska, and D. Kroening, "Concolic testing for deep neural networks," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 109–119.

[11] A. Schlögl, T. Kupek, and R. Böhme, "Forensicability of deep neural network inference pipelines," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 2515–2519.

[12] U. Alon, M. Zilberstein, O. Levy, and E. Yahav, "code2vec: Learning distributed representations of code," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–29, 2019.

[13] Y. Hou, J. Liu, D. Wang, J. He, C. Fang, and Z. Chen, "Taumed: test augmentation of deep learning in medical diagnosis," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2021, pp. 674–677.

[14] M. V. Pour, Z. Li, L. Ma, and H. Hemmati, "A search-based testing framework for deep neural networks of source code embedding," in *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2021, pp. 36–46.

[15] M. R. I. Rabin and M. A. Alipour, "Evaluation of generalizability of neural program analyzers under semantic-preserving transformations," *arXiv preprint arXiv:2004.07313*, 2020.

[16] J. Henke, G. Ramakrishnan, Z. Wang, A. Albarghouth, S. Jha, and T. Reps, "Semantic robustness of models of source code," in *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2022, pp. 526–537.

[17] 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*, 2018, pp. 303–314.

[18] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[19] Z. Aghababaeyan, M. Abdellatif, L. Briand, S. Ramesh, and M. Bagherzadeh, "Black-box testing of deep neural networks through test case diversity," *IEEE Transactions on Software Engineering*, 2023.

[20] R. Jordaney, K. Sharad, S. K. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro, "Transcend: Detecting concept drift in malware classification models," in *26th USENIX security symposium (USENIX security 17)*, 2017, pp. 625–642.

[21] X. Zhang, M. Zhang, Y. Zhang, M. Zhong, X. Zhang, Y. Cao, and M. Yang, "Slowing down the aging of learning-based malware detectors with api knowledge," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 2, pp. 902–916, 2022.

[22] K. Yan, Y. Peng, V. Sandfort, M. Bagheri, Z. Lu, and R. M. Summers, "Holistic and comprehensive annotation of clinically significant findings on diverse ct images: learning from radiology reports and label ontology," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8523–8532.

[23] S. A. Chowdhury, G. Uddin, and R. Holmes, "An empirical study on maintainable method size in java," in *Proceedings of the 19th International Conference on Mining Software Repositories*, 2022, pp. 252–264.

[24] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "Androzoo: Collecting millions of android apps for the research community," in *Proceedings of the 13th international conference on mining software repositories*, 2016, pp. 468–471.

[25] S. Sample, "Virustotal."