2023 IEEE International Conference On Artificial Intelligence Testing (AITest) | 979-8-3503-3629-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/AITest58265.2023.00014

Metamorphic Testing For Machine Learning: Applicability, Challenges, and Research Opportunities

Faqeer ur Rehman
Gianforte School of Computing
Montana State University
Bozeman, USA
faqeer.rehman@student.montana.edu

Madhusudan Srinivasan
University of Nebraska
Omaha, USA
msrinivasan@unomaha.edu

***Abstract*—The wide adoption and growth of Machine Learning (ML) have made tremendous advancements in revolutionizing a number of fields i.e., manufacturing, transportation, bio-informatics, and self-driving cars. Its ability to extract patterns from a large set of data and then use this knowledge to make future predictions is beyond the human imagination. However, the complex calculations internally performed in them make these systems suffer from the oracle problem; thus, hard to test them for identifying bugs in them and enhancing their quality. An application not properly tested can have disastrous consequences in the production environment. Metamorphic Testing (MT) has been widely accepted by researchers to address the oracle problem in testing both supervised and unsupervised ML-based systems. However, MT has several limitations (when used for testing ML) that the existing literature lacks in capturing them in a centralized place. Applying MT to test ML-based critical systems without prior knowledge/understanding of those limitations can cost organizations a waste of time and resources. In this study, we highlight those limitations to help both the researchers and practitioners to be aware of them for better testing of ML applications. Our efforts result in making the following contributions in this paper, i) providing insights into various challenges faced in testing ML-based solutions, ii) highlighting a number of key challenges faced when applying MT to test ML applications, and iii) presenting the potential future research opportunities/directions for the research community to address them.**

***Index Terms*—Metamorphic Testing, Machine Learning, Metamorphic Testing for Machine Learning, Testing Machine Learning, Challenges in Testing Machine Learning, Applicability of Metamorphic Testing in Machine Learning, Challenges of Metamorphic Testing**

I. Introduction

Machine Learning (ML) has progressed dramatically and is expanding its roots in almost every field of life i.e., manufacturing, marketing, healthcare, education, autonomous vehicles, and collaborative robots. Its high success and progress are driven by the availability of high computing resources, the abundance of data, and the development of efficient algorithms.

Since ML models are an integral part of various critical systems, they must be properly verified and validated before their deployment to the production environment. An ML-based system that lacks the existence of a systematic testing pipeline(s) may end up in resulting both financial and human loss. As evidence, history witnessed on its face when in Mar 2018, a woman in Arizona became a victim of Uber's self-driving car, resulting in her death [1]. This sad incident forced Uber to stop its services (i.e., testing self-driving vehicles); thus, resulting in huge financial and reputational losses for the company. Similarly, in Oct 2018, it was reported that the Amazon AI-based recruiting software became biased (favoring men over women) during the selection of candidates for the company-wide open positions, resultantly forcing Amazon to scrap it [2]; thus, causing wastage of time, cost, and the human resources. Last but not the least, Microsoft's AI-based chatbot (named Tay) became racist and generated offensive/hurtful tweets within its first 24 hours after its deployment [3]; thus, pushing the IT giant to immediately take it off, which seriously hurt its reputation around the globe.

The aforementioned failures strongly urge researchers to focus more on the quality assurance of ML models and propose effective testing strategies to better verify their correctness and robustness. One of the challenges faced in testing the correctness and robustness of such computationally complex AI-based models is their large input space for which they must be verified. This ultimately put these complex models under the category of non-testable programs, known as suffering from the *Oracle Problem* [4]. To better understand the problem domain, consider there is a Deep Neural Network (DNN) that takes executable files as input and predicts whether the files are malicious or not. Suppose, for the given executable file the model predicts (along with some probability score) that it belongs to the category of 'malicious' class label. However, the important question that arises here is *'how can we verify and test that the output produced by this model is correct and reliable?'* One of the potential solutions is to use human involvement i.e., a security expert first creates a virtual machine, configures the testing environment, and then executes the given input file to observe whether it is really malicious or not, and at the end using this knowledge to verify the output

979-8-3503-3629-0/23/$31.00 ©2023 IEEE
DOI 10.1109/AITest58265.2023.00014

produced by the ML model under test. This method is very expensive, laborious, infeasible, and error-prone when we have thousands of files to verify. This further raises an important question to answer *'how to test such complex non-testable programs that suffer from the oracle problem?'*

In comparison to ML, Software Engineering (SE) is a much more mature field having well-known, experimented, and trusted software testing techniques. In this context, SE4ML is a new trending research area in which one of the main focus areas is to leverage the existing SE testing techniques for better quality assurance of ML-based models [5] [6]. Among the traditional SE testing techniques, the most effective and common testing technique that we found to alleviate the oracle problem in testing ML-based models is the Metamorphic Testing (MT) approach [7]. The MT technique does not require the availability of ground truth to verify the correctness of individual input. Instead, it uses the concept of multiple executions (i.e., the source and follow-up executions) to check whether the necessary property captured in the Metamorphic Relation (MR) is satisfied or not. The buggy behavior of the program is said to be detected if the program fails to adhere to the relation specified in the MR(s).

One may argue that among the traditional software testing techniques, the MT technique is not the only testing technique that addresses the Oracle problem. Instead, other popular testing techniques like *Differential Testing* and *N-version programming* could be used to avoid the extra cost (of multiple executions) involved in using MT. This reasoning is valid and unquestionable but one of the major limitations that make these two testing techniques unrealistic/infeasible in a real environment is that they require multiple copies/versions of the same model under test which is not always available in real life. This makes the MT technique more realistic, and the feasible choice among researchers to alleviate the oracle problem in testing ML-based models. However, we have identified a number of limitations in MT (when used for testing ML) that the researchers must be aware of them when applying MT for testing ML applications.

It is important to highlight that a wide range of research studies are available in which MT has been applied in testing both supervised and unsupervised ML applications but to the best of our knowledge, we are unable to find any work that guides/informs the researchers about the open challenges faced in using MT for testing ML-based models and the potential future research opportunities/directions in a centralized place. Applying MT to test ML-based critical systems without prior knowledge/understanding of those limitations can cost organizations a waste of time and resources. The work of Chen et al. [14] does highlight some challenges faced in MT (at a broader level) but their work is solely focused on MT in general, neither discusses those challenges in the context of the emerging field of ML nor shows whether they are applicable in ML domain or not, which is equally the motivator for this research work. We aim to address this gap by making the following contributions.

- To better understand the problem space, we first provide insights into the general challenges encountered in testing ML-based solutions.
- We highlight the limitations/challenges faced in applying MT to test ML-based systems.
- We recognize those limitations/challenges as potential research opportunities and provide further directions to the research community to address them.

We organize the rest of the paper as follows. Section II discusses the related literature review in which we mainly focus on highlighting the applicability of MT in testing both supervised and unsupervised ML. Section III discusses the challenges faced in testing ML applications, whereas, Section IV highlights the limitations/challenges faced in applying MT for testing ML applications along with potential future research opportunities. In the end, we conclude our study in Section V.

II. Literature Review

Metamorphic Testing (MT) [7] is a property-based testing technique that identifies and targets the necessary properties of the program under test. It identifies *relations* known as Metamorphic Relations (MRs) that use a concept of multiple executions (known as *source* and *follow-up* executions) to check whether the necessary characteristic/property expected from the program under test is satisfied or not. The violation of necessary MR will depict that the program under test has some potential bug. Consider a program used for calculating variance among the data points. One of the MRs can be 'if we change the order of data points, it should not change the final calculated variance for the given data points'. Similarly, 'multiplying all the data points with -1 should keep the output consistent'.

The tremendous advancement of ML in almost every field of life is inevitable. ML has been largely embraced by researchers and is used in a variety of domains i.e., to test compilers [10], code obfuscators [14], protein function prediction tools [12], data analytics programs [9], machine translators [11], and autonomous vehicles [13]. However, their computational complexity and large input space make it a challenging task for software testers to devise systematic testing techniques for better testing them and improving their quality. In this context, MT has been shown to be a promising testing technique that has been widely applied in testing such computationally complex programs. To better understand the applications of MT in ML research space, we present a brief literature review targeting the testing of the most common types of ML i.e, supervised and unsupervised ML.

A. MT Applicability in Testing Supervised ML

Zhou et al. [13] used MT for the identification of faults in Apollo's perception module and reported the identified inconsistencies eight days before the incident when Uber's autonomous vehicle hit and killed the pedestrian. Shahri et al. [12] proposed several MRs that perform different transformations to the input (i.e., protein sequences) for performing the quality assurance of a protein function prediction software.

Moreira et al. [17] leveraged MT in testing six acoustic scene classifiers by performing sample and attributes permutations. The results obtained show that the proposed approach is able to identity both verification and validation issues in the applications under test. Ma et al. [18] proposed MT-based approach for addressing the oracle problem in testing a fake news detection system. However, there is no clear evidence showing the identification of implementation bugs in the program under test. Jiang et al. [16] proposed a set of 20 MRs and checked their effectiveness by validating four sentiment analysis systems. Further, the authors tried to explore the effect of *false satisfactions* on MT effectiveness and their potential causes when testing sentiment analysis systems. Ma et al. [15] took an advantage of using MT to test a classification model trained on COVID-19 CT images which predict whether the patient has been diagnosed with COVID or not. The authors proposed 8 MRs to test and detect inconsistencies in the COVID-19 diagnosis classification program under investigation. Rehman et al. [19] [20] work uses a combination of MT and statistical concepts for verification of MRs to identify the induced buggy behaviors in the network intrusion detection systems.

B. MT Applicability in Testing Unsupervised ML

Research shows that the researchers are focusing more on using MT for testing supervised ML applications but much-limited research work is available in the space of leveraging MT for testing unsupervised ML algorithms. To the best of our knowledge, we are able to find just four research papers in which MT has been used in testing unsupervised ML algorithms [22] [23] [24] [25]. Yang et al. [22] proposed a list of MRs and used them to test (from a validation perspective) a clustering algorithm (i.e., k-means) implemented in WEKA tool. Similarly, Xie et al. [23] also focused on testing WEKA tool in which they leveraged the MT technique for validating a set of clustering algorithms (i.e., belonging to the category of Prototype-based systems, Hierarchy-based systems, and Density-based systems) to check whether the algorithms under investigation adhere to the user expectations or not. Rehman et al. [24] [25] took MT one step further and proposed a large set of diverse MRs for testing python based library (sci-kit learn) in which they tested k-means, Agglomerative clustering, and DBSCAN algorithms from both the verification (targeting the necessary characteristics of the algorithms under test) and validation perspective (targeting the user expectations expected from the algorithms under test). The limited research available urges researchers to make further significant contributions in this research space.

III. Challenges in Testing Machine Learning Programs

Before discussing the open challenges faced in using MT for testing ML-based applications, we first put some light on highlighting notable challenges posed by testing ML-based systems, which are as follows.

1) Requirements Specification Document: In comparison to traditional software, preparing a requirements specification document for the ML model is a challenging task. The complex concepts involved in the implementation of ML algorithms are the driving force that prevents customers/business analysts to capture the clear requirements expected from the final product. The lack of a specification document ultimately makes it a harder task for the software tester to verify the acceptance criteria for the ML model under test.

2) Strong Cohesiveness Among ML Components: The ML model can broadly be classified into three components, i) the data, ii) the application code written by the ML developer, and iii) frameworks/libraries (i.e., Pytorch, Sci-kit learn, etc.) but the program available for testing/deployment is the final trained ML model. The trained ML model is treated as a single unit, making it a challenging task for the software testers to break down the system into smaller components and then test each component individually for better fault localization and quality improvement.

3) Large Input Space: When testing traditional software systems, a software tester may have some information beforehand to test the program i.e., the availability of a list of valid and invalid users to test the authentication mechanism of the program under test. However, in testing ML-based solutions, a software tester grapples with understanding the large input space e.g., features like price, location, and weather conditions may contain any possible valid value. Hence, validating the ML model for such a large input space is a challenging task. This raises an important question to answer *'how can a well-representative and limited set of data be generated/used to better test the ML solutions within limited resource constraints'*, which is equally an open research problem.

4) Decision Logic: When developing traditional software, a software engineer hard-code the rules and the business logic is fixed to perform the desired functionality. However, in ML-based models the software engineers do not hard-code the decision logic, instead, these rules are learned from the training data. This brings an extra challenge for the software tester to test ML-based applications in which decision logic changes based on the data provided to them.

5) Testing Multiple Components: When testing traditional software, the main component for testing is the application code. However, ML-based models have multiple components to test that include, i) the data, ii) the code written by the software engineer, and iii) the underlying framework/library i.e., Pytorch, and Sci-kit learn. Testing all these components requires extra resources and introduces its own challenges for the software testers to perform their testing within the limited available time and resources.

6) Oracle Problem: The ML models have unique characteristics i.e., they have large input space for which they need to be tested. Specifying/identifying the oracle (in the form of a class label) for such a large input space makes these systems suffer from the oracle problem [4].

7) Non-Deterministic Results: Generally, the business rules written in traditional software output deterministic results

across multiple runs. However, in ML-based models i.e., Neural Networks, each time the model is trained on a fixed set of training data, it is not necessary that the model will generate deterministic predictions for the same set of test data. This stochastic nature makes the traditional software testing techniques infeasible to address the oracle problem in testing such types of computationally complex models.

8) Black-box Nature: In general, complex ML-based models i.e., Deep Neural Networks, are treated as a black-box model because of the computational complexity involved in their training and making predictions for unseen data. This makes them less transparent, difficult to interpret, and harder to test. ML/AI Explainability is another hot research area where researchers focus on addressing such problems at scale.

9) Misleading Performance Metrics: In supervised ML, we strongly rely on using performance metrics to check the prediction power of the model under test. However, it is important to highlight that sometimes these performance metrics i.e., accuracy, may be misleading, especially, when the model under test is producing high accuracy but does contain a hidden implementation bug [26]. Thus, it is important to mention that in addition to using performance metrics, the researchers should also focus on proposing effective testing strategies for better testing of ML programs that are now getting an integral part of mission-critical systems.

IV. Challenges/Limitations of MT in Testing ML and Future Research Opportunities/Directions

The existing literature shows that researchers are widely using the Metamorphic Testing (MT) technique for better testing of a large range of ML applications and is undoubtedly considered one of the most effective testing techniques to address the oracle problem in testing ML-based systems. However, like every testing technique, the MT technique has its own limitations/challenges faced in testing ML but we are unable to find any work that highlights those limitations/challenges to better understand them and provide directions for making future research contributions. Applying MT to test ML-based critical systems without prior knowledge/understanding of those limitations/challenges can cost organizations a waste of time and resources. This motivated us to focus on addressing this gap that result in this study which captures the following list of limitations/challenges along with future research directions. We hope that this research work will enable researchers and practitioners to make well-informed decisions when they aim to apply MT for testing ML.

1) Testing Neural Networks: One of the challenges faced by researchers in finding implementation bugs in Neural Networks (NNs) based models is their non-deterministic nature in making predictions for the same test inputs (due to random assignment of network weights during the training phase) [26] [27]. It is important to highlight that the traditional MT works well for testing traditional ML-based models i.e., SVM, Decision trees, and K-NN but it becomes infeasible when the program under test (i.e., NN-based model) has stochastic nature in the final outputs and the software tester has used equality condition to verify the results (obtained for the source and follow-up inputs). Therefore, in such scenarios, the classical MT technique can't be used directly to verify the MRs for such ML models. The most recent work [19] [20] integrates MT with statistical methods to verify the proposed MRs for testing Deep Neural Networks (DNNs). However, one of the limitations of their approach is the high computational cost because in order to statically verify the MRs, the NN model under test needs to be trained multiple times. A possible research direction can be the optimization of their statistical MT approach to lower the computational cost. As an example, one of the possible approaches to extend their work can be using a less number of well-representative features/samples (that will lower the training time for large ML models) and showing their fault detection effectiveness using empirical studies.

2) Consistent Outputs Can Lead To False Acceptance of MR(s): The MT technique identifies the violation of MR if the output obtained for the source and follow-up inputs are not consistent. The important question to answer is *what if the buggy program version outputs the wrong but consistent outputs for both the source and follow-up inputs?* Such cases will result in high false positives, wrongly assuming that the MR(s) have been satisfied, which ultimately can lead to disastrous consequences when used in testing mission-critical systems. Therefore, we recommend organizations use MT as a supplementary testing technique on top of using other testing techniques, especially, when the cost of a false positive is very high. Apart from that, knowing about such limitations of MT will help organizations in making much informed decisions. Some potential future directions in this area can be to identify which other criteria (other than relying on the final outputs/predictions) can be used to verify the MRs for the program under test. As an example, Rehman et al. [24], uses multiple criteria (i.e., final outputs, cluster centroids, and nearest neighbors) to verify the MRs for testing k-means clustering algorithm. Similarly, instead of relying on the final output, more granular details i.e., the features used by the ML model (for making predictions) can also be used to verify the MRs.

3) Large Set of MRs can Make MT Infeasible: Due to the critical nature of ML-based applications, it is always desirable to have a large set of MRs for their effective testing. However, a large set of MRs may not be applicable in a regression testing environment when the organizations have limited testing resources. A future research direction can be to identify which characteristics of ML-system under test can be utilized for effective MRs prioritization and minimization; thus, saving organizational testing cost and resources. A potential extension to it can be showing the generalization of the prioritized MRs for testing a large number of ML models.

4) Human Involvement in Identification of MRs: One of the challenges faced in using MT for testing ML applications is the human involvement in the identification of MRs for the ML model under test. This is a very challenging task because of the complex nature of ML algorithms which requires their deep

understanding; thus, making it a harder task for a software tester to test them. A possible research direction can be to leverage ML-based techniques for the automatic identification and prediction of suitable MRs for the ML model under test. However, addressing this problem using supervised ML may require generating a sufficient amount of data and then accurately labeling them.

5) Unrealistic Test Cases: MT is not only a testing technique but can also be used for generating new test cases, i.e., the follow-up execution phase of MR helps in the generation of new data. However, in some cases, the generated test cases may not be realistic enough to capture the real-life scenarios encountered by the program under test in a production environment. An example of such MR can be i.e., the multiplication of features (i.e., date of birth, IP address, etc.) with some constant may result in the generation of new follow data that may never be encountered/exist in a real life. Therefore, such MRs will serve no/less fruitful purpose at the cost of wasting organizational testing resources. A potential future research direction can be to identify and propose new approaches that can help in generating more realistic and representative test cases (for both structured and unstructured data) for the MRs that will be encountered by the model(s) in a real environment.

6) MR's Fault Detection Effectiveness: To show the effectiveness of proposed MRs, researchers normally use the *Mutation Score* i.e. the percentage of faults detected, and then use this measure to identify the MRs that are most effective in uncovering the injected bugs [26] [27]. This process requires the execution of both the source and follow-up test cases (to calculate the mutation score) which is a costly and resource-intensive task, especially, when the model under test is complex and takes a few hours to train itself for single program execution. A potential future direction can be using ML-based techniques for the prediction of fault detection effectiveness of each MR (before executing them) and then using this knowledge to predict and execute the MRs that are 'more effective'; thus, saving the organizational's testing cost in preventing the execution of MRs that are least effective.

7) High Execution Cost: In order to verify a single MR, the ML model under test needs to be trained multiple times i.e., one for the source inputs and one for the follow-up inputs. Consider, a deep neural network that takes at least a day to train itself on a given dataset. This execution time will get double when a software tester intends to verify a single MR for the model under test. Similarly, to verify a set of 15 MRs, a software tester will need at least a month to execute all of them. Such a high execution cost makes MT inapplicable for testing large-scale complex ML models. A possible solution can be using a subset of well-representative features and instances that can help in reducing the overall training time for verification of MR(s). However, identifying a small set of well-representative features and samples for such a problem itself is another interesting open research problem.

8) Inability of MRs to Identify The Location of Bug(s): When applying MT to test ML-based models, a violation of MR just reveals the existence of a potential bug in the model under test, however, it tells nothing about its location i.e., where the bug is (either in the data or in the program code)?. A possible approach to address this problem can be, i) using MT for testing each component (i.e., data and code) separately, and ii) integrating MT with other testing techniques i.e., symbolic execution of programs and Unit testing to better help software engineers identify the location of bug in the model under test.

9) Documenting the type of MR(s): In MT, one of the common misconceptions is that the violation of MR will depict the existence of some bug in the program under test. However, it is important to highlight that this may not hold true for every scenario because the violation of MR does not necessarily mean that there is some implementation bug in the program under test [21]. As an example, when testing the k-NN algorithm, the addition of a new data point (during follow-up execution) may change the proportion of data points belonging to some specific class label that can create a tie, and can change the final results. Such a violation of MR will not depict that there is some implementation bug in the algorithm under test. Instead, the algorithm is correctly implemented but the result changed because the algorithm randomly selected one of the class labels to break the tie. Thus, it is very important to explicitly mention/document whether the proposed MR(s) target the verification aspect or the validation aspect of testing the algorithm under investigation. The violation of MR(s) targeting the verification aspect can be characterized as the existence of an implementation bug but the MR(s) targeting the validation aspect does not necessarily mean that there is some bug identified in the program under test. Instead, such a violation captures the deviation of the model from the general user's expectations. If such important information about the MRs is not properly highlighted and documented, this will result in high waste of time and resources for an organization. As an example, imagine a case when a developer spent a significant amount of time and effort to find a bug(s) that in reality was not existing at all because it was not explicitly documented whether the violated MR targets the verification or the validation aspect of testing the model under test.

10) Limited Research in Testing Unsupervised ML: In comparison to supervised ML, we have seen much less research focus on using MT for testing unsupervised ML. To the best of our knowledge, we are able to find just four research papers in which MT has been used in testing unsupervised ML algorithms [22] [23] [24] [25]. This creates a tremendous opportunity for researchers to explore and propose effective testing strategies for using MT in a such less focused area.

11) Lack of MRs Generalization: Another challenge faced when using MT for testing ML programs is the difficulty encountered in the generalization of MRs. An MR applicable in one domain data (i.e., healthcare image data) may not be applicable/scalable to some other similar domain data (i.e., healthcare tabular data). A potential future direction can be proposing a framework that can serve as a guideline for researchers to create/propose MRs that could be generalized to multiple domains effectively.

12) Identification of Sufficient Number of MRs: The literature suggests that different researchers proposed a different number of MRs for the ML models under investigation. Sun et al. [11] proposed a single MR for testing machine translators i.e., Google, and Microsoft, whereas, Ma et al. [15] proposed 8 MRs to test and detect inconsistencies in the COVID-19 diagnosis classification model under investigation. This raises an important question to address *'how many MRs can be considered sufficient to check the correctness and robustness of the ML model under test'*. On one hand, using a large number of MRs may be impractical in a regression testing environment, whereas, on the other hand, a small number of MRs may not be sufficient to effectively test the model under test; thus, finding the optimal number/balance is critical. One potential avenue for future research can be to mine historical data and use that knowledge to suggest/predict the sufficient number of MRs for testing the ML model(s) under test. Furthermore, organizations can tweak their data mining algorithm (for finding a sufficient number of MRs) based on the available resources they have.

V. Conclusion

Metamorphic Testing is undoubtedly a very powerful and simple approach for testing the correctness and robustness of ML-based solutions. However, applying MT to test ML-based critical systems without having prior knowledge about its limitations/challenges may cost organizations a waste of time and resources. In this study, we briefly highlight the applicability of MT in testing ML applications (both supervised and unsupervised ML), its limitations/challenges faced in testing ML solutions, and potential future research opportunities for researchers to address them. We believe that this study will help the researchers and practitioners to be aware of the MT limitations/challenges beforehand (when testing ML systems) and possible consequences during the software testing phase. In the future, we aim to address some of the highlighted limitations/challenges that will further expand the body of knowledge in the emerging field of 'Metamorphic Testing for Machine Learning'.

References

[1] Ohnsman, A. (2018). Lidar maker velodyne 'baffled'by self-driving uber's failure to avoid pedestrian. Forbes, March.

[2] https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G

[3] https://www.theguardian.com/technology/2016/mar/24/tay-microsofts-ai-chatbot-gets-a-crash-course-in-racism-from-twitter

[4] Weyuker, E. J. (1982). On testing non-testable programs. The Computer Journal, 25(4), 465-470.

[5] Kumeno, F. (2019). Sofware engneering challenges for machine learning applications: A literature review. Intelligent Decision Technologies, 13(4), 463-476.

[6] Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., ... & Zimmermann, T. (2019, May). Software engineering for machine learning: A case study. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (pp. 291-300). IEEE.

[7] Tsong Y Chen, Shing C Cheung, and Shiu Ming Yiu. Metamorphic testing: a new approach for generating next test cases. Technical Report HKUST-CS98-01, Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, 1998.

[8] Chen, T. Y., Kuo, F. C., Liu, H., Poon, P. L., Towey, D., Tse, T. H., & Zhou, Z. Q. (2018). Metamorphic testing: A review of challenges and opportunities. ACM Computing Surveys (CSUR), 51(1), 1-27.

[9] Jarman, D. C., Zhou, Z. Q., & Chen, T. Y. (2017, May). Metamorphic testing for Adobe data analytics software. In 2017 IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET) (pp. 21-27). IEEE.

[10] Donaldson, A. F., & Lascu, A. (2016, May). Metamorphic testing for (graphics) compilers. In Proceedings of the 1st international workshop on metamorphic testing (pp. 44-47).

[11] Sun, L., & Zhou, Z. Q. (2018, November). Metamorphic testing for machine translations: MT4MT. In 2018 25th Australasian Software Engineering Conference (ASWEC) (pp. 96-100). IEEE.

[12] Shahri, M. P., Srinivasan, M., Reynolds, G., Bimczok, D., Kahanda, I., & Kanewala, U. (2019, April). Metamorphic testing for quality assurance of protein function prediction tools. In 2019 IEEE International Conference On Artificial Intelligence Testing (AITest) (pp. 140-148). IEEE.

[13] Zhou, Z. Q., & Sun, L. (2019). Metamorphic testing of driverless cars. Communications of the ACM, 62(3), 61-67.

[14] Chen, T. Y., Kuo, F. C., Ma, W., Susilo, W., Towey, D., Voas, J., & Zhou, Z. Q. (2016). Metamorphic testing for cybersecurity. Computer, 49(6), 48-55.

[15] Ma, Y., Pan, Y., & Fan, Y. (2022, August). Metamorphic Testing of Classification Program for the COVID-19 Intelligent Diagnosis. In 2022 9th International Conference on Dependable Systems and Their Applications (DSA) (pp. 178-183). IEEE.

[16] Jiang, M., Chen, T. Y., & Wang, S. (2022). On the effectiveness of testing sentiment analysis systems with metamorphic testing. Information and Software Technology, 106966.

[17] Moreira, D., Furtado, A. P., & Nogueira, S. (2020, August). Testing acoustic scene classifiers using Metamorphic Relations. In 2020 IEEE International Conference On Artificial Intelligence Testing (AITest) (pp. 47-54). IEEE.

[18] Ma, Y., Towey, D., Chen, T. Y., & Zhou, Z. Q. (2021, July). Metamorphic Testing of Fake News Detection Software. In 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC) (pp. 1508-1513). IEEE.

[19] ur Rehman, F., & Izurieta, C. (2021, July). Statistical Metamorphic Testing of Neural Network Based Intrusion Detection Systems. In 2021 IEEE International Conference on Cyber Security and Resilience (CSR) (pp. 20-26). IEEE.

[20] ur Rehman, F., & Izurieta, C. (2021, September). A Hybridized Approach for Testing Neural Network Based Intrusion Detection Systems. In 2021 International Conference on Smart Applications, Communications and Networking (SmartNets) (pp. 1-8). IEEE.

[21] Xie, X., Ho, J. W., Murphy, C., Kaiser, G., Xu, B., & Chen, T. Y. (2011). Testing and validating machine learning classifiers by metamorphic testing. Journal of Systems and Software, 84(4), 544-558.

[22] Yang, S., Towey, D., & Zhou, Z. Q. (2019, May). Metamorphic exploration of an unsupervised clustering program. In 2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET) (pp. 48-54). IEEE.

[23] Xie, X., Zhang, Z., Chen, T. Y., Liu, Y., Poon, P. L., & Xu, B. (2020). METTLE: A metamorphic testing approach to assessing and validating unsupervised machine learning systems. IEEE Transactions on Reliability, 69(4), 1293-1322.

[24] Rehman, F. U., & Izurieta, C. (2022, June). MT4UML: Metamorphic Testing for Unsupervised Machine Learning. In 2022 9th Swiss Conference on Data Science (SDS) (pp. 26-32). IEEE.

[25] Rehman, F. U., & Izurieta, C. (2022, August). An Approach For Verifying And Validating Clustering Based Anomaly Detection Systems Using Metamorphic Testing. In 2022 IEEE International Conference On Artificial Intelligence Testing (AITest) (pp. 12-18). IEEE.

[26] Li, Z., Cui, Z., Liu, J., Zheng, L., & Liu, X. (2020, January). Testing neural network classifiers based on metamorphic relations. In 2019 6th International Conference on Dependable Systems and Their Applications (DSA) (pp. 389-394). IEEE.

[27] Dwarakanath, A., Ahuja, M., Sikand, S., Rao, R. M., Bose, R. J. C., Dubash, N., & Podder, S. (2018, July). Identifying implementation bugs in machine learning based image classifiers using metamorphic testing. In Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis (pp. 118-128).