How to solve the problems of test oracle, test adequacy criteria and test input generation in machine learning systems ETSN20

Axel Berglund, Artur Lidström, Robin Zack Fitz-Gibbon Jeppesen
Lund University, Sweden
{ine15abe, tfy14ali, tpi12rje}@student.lu.se

***Abstract*—This paper will offer an introduction to testing of machine learning systems, it will cover information about what machine learning and deep neural networks are and what some of the problems of testing them are. The problems this paper will focus on are: test oracle problem, test adequacy criteria and test input generation. For a possible solution to these problems the testing methods called metamorphic, neuron coverage and adversarial testing are introduced and discussed. In closing this projects conclusions was that training and testing of machine learning systems work together and how each method has own advantages and disadvantages so that by using them in combination a better result for training and testing a machine learning system could be achieved.**

I. Introduction

What are the differences in different methods of testing for machine learning(ML) and how do they compare to each other? What are the general problems with testing for ML? This project will focus on metamorphic testing, coverage based testing and adversarial testing, and how they try to solve for some of the general problems of testing.

A. Background

ML systems are getting more popular and are implemented in potentially dangerous situations, as in self-driving cars [1]. As a consequence of this there will be an added weight to test the systems before implementing them. Incorrect behaviours are often caused by corner cases and is not always found in a given dataset [2]. Different testing methods can be used to simulate these types of corner cases and we will try to research if that works well, one of these is called adversarial testing and that method will be researched more. Furthermore a discussion on how to avoid the problems of not having access to a testing oracle can affect the testing process as well as the advantages and limitations of neuron based coverage.

B. Purpose

The purpose of this paper is to give an overview of the most popular [3] methods currently available for testing ML systems. The paper will also provide a comparison of the different testing methods and will attempt to analyse the current issues that exist for each method. In order to grasp how the different testing methods are tied to ML and how they interact with each other the notion of how these methods work on a driverless car will be described by examples as a way to highlight differences.

C. Problems with testing for machine learning

One of the problems with testing in general is the test oracle problem. This refers to the problem that for a system to work as intended there has to be someone able to determine if a certain input gives the desired output. A test oracle in this sense would be something that knows what the desired output should be. When systems get more complex it might be hard to know what type of output is to be desired. When ML systems are based on very large sets of training data another problem is the cost of analysing and classifying all this data. This is where automated testing oracles is a proposed solution but it has not yet been solved. [4]

Another problem that testing for ML shares with testing of traditional software is the test adequacy criteria. In regular software statement, branch and path coverage is used as a measurement tool for testing but this is not as effective for ML programs. In an experiment of how a single randomly picked test input would affect code coverage for several ML systems it was found to achieve 100% code coverage. This introduces a problem, if previously used techniques for testing are not as well suited for ML, what would be an alternative? [5]

For a relatively non-complex program, like a sorting algorithm, generating a data set for testing is often a trivial task. When the programs get more complex the amounts of different types of inputs might increase. For example with driverless cars, gathering training data is often very time-consuming as presented by Sterling Anderson, the former Head of AutoPilot at Tesla, at MIT's EmTech Digital Conference in 2016 [6]. If one is to collect data by filming the road while driving their own vehicle there are a lot of variables that will affect the images captured, for instance driving can occur in a lot of different weather conditions and at a lot of different times during the day. A problem in this case is then how can we be sure that a ML program trained on images taken during the day will behave as intended if the system is used at night. Is there a solution for generating test inputs in a manner that a lot of different situations will be covered but in a more automated fashion? These questions give rise to the test input generation problem.

II. Description

In this section concepts about ML and the different types of testing methods will be explained more in depth.

A. Machine learning

ML is a type of artificial intelligence technique that makes decisions or predictions based on data. It can be classified into two different areas. Classic ML which for example is algorithms like decision tree and the other area is called deep learning. This paper will focus in on deep learning which applies something called Deep Neural Networks (DNN) which can generally be described as multiple layers of neurons. Each neuron in the DNN can be seen as some type of processor which takes in information and feeds it forward to the next layer of neurons. In the end the DNN makes a decision based on the output layer as shown in Fig. 1.

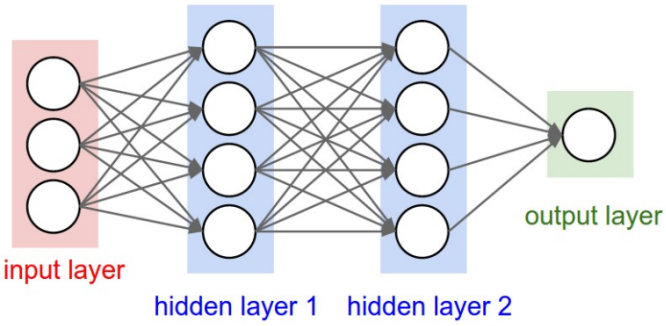


Fig. 1. A visualisation of the layers in a DNN, each circle is supposed to represent a neuron

The input in this paper will mostly focus on images and the classification of objects in images. In the case of driverless cars the input image is the environment in which the car is, and the DNN should be able to classify objects such as stop signs correctly, or else it could lead to a very dangerous situation. It is unfeasible to continuously keep track of the location of every object in the world and so the driverless car needs a way to know what is around it automatically. This is where ML is used, and the need for good testing methods. If you can not test the ML system, or if there is a way to trick the system into classifying something incorrectly, it could have fatal consequences.

B. Metamorphic testing

Consider a driverless car, being tested on the road for deployment. It would be nearly impossible to test it under every condition imaginable. Therefore there is a need to be able to automatically generate test cases from existing ones, to test if the ML system can handle these conditions. Examples include: A picture of the road is shown to the system which decides to turn right. The same picture, but distorted or transformed without changing the road, is then shown. Added rain or a camera lens distortion could have been introduced to the picture as seen in Fig. 2. However from the existing test case the correct answer is known, to turn right, an other output would point to a defect in the system. Therefore the answer is known, a test oracle for certain test cases has been produced and a lot more test cases can be generated to test the ML systems capabilities during these artificially altered conditions. [7]

Fig. 2. a) Original, output turn right. b) Altered picture with added rain and incorrect output.

Two general problems for ML testing was introduced in the example above. The lack of a testing oracle, in other words not knowing the correct output given the input. And also that the number of test cases that can be generated manually is limited. Metamorphic testing was introduced as a tool to combat these challenges. It is based on iterative testing of the system where there is a known answer to some of the test cases.

In metamorphic testing a set of metamorphic properties are identified for the system. Metamorphic here meaning that certain properties can be transformed without changing the answer derived from them, for example adding a constant to a list of numbers won't change their order. Transforming the input according to these properties creates new test cases where the output can be predicted according to the previous output using the same transformation. thereby automatically generating new test cases from the old test cases. When running these test cases if the output is unable to be predicted correctly it shows a fault in the system. The method is used in a way that given the correct answer to the first test case the system should use the metamorphic properties of the functions of the system to transform the input such that the new output should be predicted according to the transformation of the previous output. [8]

Certain benefits of this method of testing include: Generating additional test cases without being redundant, creating a test oracle of known outputs which are correct for a certain input and being able to show and kill mutants if they would appear.

According to [9] tests performed on a real and popular open source program discovered multiple violations of the metamorphic relations which indicated real faults. The tests also performed well when killing of mutants. Mutants are described such as variations in the source files, intentionally generated and introduced as a means to test if the test methods available are able to detect new faults, thus killing the mutant. The mutations performed by Xiaoyuan and her team randomly selected 30 new valid mutants for two different applications of the program using the mutant generator MuJava. After

excluding the mutants which produced the same faults detected in the original tests as well as mutations occurring outside of the test scopes , 21 and 22 mutants were run through the metamorphic tests. For both applications the metamorphic tests were able to kill of more than 90% of the mutants (90,5% and 90,9% respectively). Therefore showing quite some promise as a testing method for both producing a test oracle as well as identifying faults and defects. [9]

C. Coverage based testing

In regular software testing code coverage is a common tool for testing and this idea is carried on to work for ML as well. In the case of ML however another aspect is used instead of regular code coverage. This is called neuron coverage and that is a measurement for how many neurons that are activated given a certain test input. There exists different types of coverage based testing where another technique is to not only measure how many neurons that are activated but to look for how many different combinations of neuron activations there are and to what degree they activate.

When taking the same driverless car from the case in metamorphic testing, neuron coverage based testing analyses the DNN after its been introduced to a dataset of, in this case, a lot of pictures of roads and the direction the car is supposed to turn. After the DNN has trained on these pictures, measurements of coverage is collected. There are a couple of different types of neuron coverage types and three of them will be introduced in this section [10].

k-multisection Neuron Coverage: After analysing the training data set each neuron in the DNN is shown to be contained to a lowest low_n and a highest $high_n$ possible value, each value is given by the activation function of each neuron. k-multisection Neuron coverage sets then sets out to divide the range into k number of sections. When the DNN later is tested using a test data set the coverage is measured by how many of the sections that are covered by the test inputs. [10]

Neuron Boundary Coverage: In contrast to k-multisection Neuron Coverage, Neuron Boundary Coverage is used to look for how many corner cases that are found by the test data set. Since the training data set sets the limit for the boundaries this type of coverage looks for neurons that gets the value of either lower than low_n or higher than $high_n$. [10]

Top k-neuron Coverage: Instead of checking for which values the different neurons take the top k-neuron coverage focuses on measuring how layers of neurons behave. This method measures how many times the top k neurons in each layer has been activated given a certain test data input, here k stands for the number of neurons one wants to check for. For example: Top 2-neuron coverage looks at how many times the two most active neurons in each layer has been activated. That a neuron has a higher activation than another means that the value associated with the neuron is higher than another as in the case of k-multisection Neuron Coverage. [10]

D. Adversarial testing

Adversarial testing is the testing of ML systems where the data contains noise/perturbations. This can either occur naturally or through adversarial attacks. By using adversarial testing the system will be less affected by adversarial examples and it will help defend against adversarial attacks where an attempt is made to fool the ML system through malicious input. [3]

Kurakin et al. summarizes the problem as follows:

> Let's say there is a ML system M and input sample C which we call a clean example. Let's assume that sample C is correctly classified by the ML system, i.e. $M(C) = y_{true}$. It's possible to construct an adversarial example A which is perceptually indistinguishable from C but is classified incorrectly, i.e. $M(A) \neq y_{true}$. [11]

To give an example of this Papernot et al. added perturbations to an image of a stop sign, see Fig. 3, which makes a particular DNN classify it as a yield sign while still looking visually the same (i.e. as a stop sign) to the human eye. As ML is becoming more and more integrated in products such as self-driving cars this poses a great danger if there are no failsafes. It is conceivable that the perturbation could be added by modifying the sign itself by using, for example, a sticker or paint, which makes this a great security concern. [12]

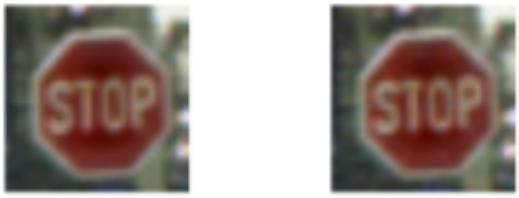


Fig. 3. Two images looking the same to the human eye, but being classified differently

In order to make applications for ML such as driverless cars a reality issues like these must be solved, which lead to the research of adversarial ML and the testing against adversarial attacks.

At the moment there is no standardized way to perform adversarial testing, but several methodologies and algorithms have been proposed. One such algorithm is DeepFool [13]. DeepFool tests the robustness of classifiers by computing perturbations that has a high probability of fooling the system. By testing the system against the perturbations computed by DeepFool the tester can detect weaknesses of the system which will help lead to more robust classifiers.

Biggio et al. [14] suggests three golden rules when it comes to protecting a ML system from adversarial attacks. The three rules are (i) know your adversary, (ii) be proactive and (iii) protect yourself. They suggest that to defend yourself against adversary attacks you must be able to model the threats against the learning-based system under design. You can then test how the system reacts to the threats during training and testing and can thus build a more robust system that will be able to defend itself against such attacks.

E. Method backgrounds

1) Metamorphic testing: Metamorphic testing was invented by T.Y. Chen in a technical report in 1998 and was first introduced to ML by Murphy and his colleagues in 2008. [15] One of the older methods to test ML systems and still one of the most widely used.

2) Coverage-based testing: Coverage based testing is widely used in traditional and more recently developed for ML systems to measure and indicate the quality of the software. It is adjusted for ML by going from code based coverage to neuron based coverage, this is still in early stages of development. One of the first papers presented on this subject was DeepXplore that was written in 2017. [5]

3) Adversarial testing: Adversarial testing has been proposed since many ML systems have shown good performance in classification tasks but are often highly unstable to adversarial examples. If the system is not tested on adversarial examples there is a high risk of an attacker being able to fool the system and performing black box attack without knowing the model's parameters.

One of the first instances of this problem being highlighted was by Dalvi et al. [16] who showed that linear classifiers used in spam filtering could be fooled by small changes in the content. However, it wasn't until 2014 that adversarial testing really gained traction when Szegedy et al. [17], and subsequent work [13] [18] [19], showed that a DNN can be fooled by small perturbations that look the same to the human eye.

III. Analysis

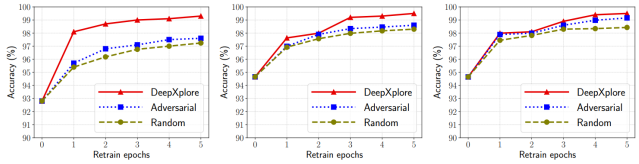
Based on our literature research we will here present an analysis of the testing methods, including what issues there might be with using the particular testing method. When analyzing each method the question to consider is if the issues of each method hinders the ability to solve the problems of testing introduced in the beginning. This section also includes a validation of the project, where we discuss the trustworthiness of the project and the presented results.

A. Metamorphic testing

Even when proven useful Metamorphic testing has the issue of being hard to implement without a deep understanding of the ML system. Identifying the properties of the data sample which you can transform and still retain predictability of the answer would demand a close inspection of the algorithms of the system.

B. Coverage based testing

Since this method is still in development the proof for the advantage of coverage based testing is still lacking, however the findings in DeepXplore show that with coverage based testing the accuracy of their tool for training DNN:s was able to achieve a 1-3% better result compared to other DNN:s that used either adversarial or random augmentation, see Fig. 4. [5]



(a) LeNet-1 (b) LeNet-4 (c) LeNet-5

Fig. 4. Results from testing three DNN:s with DeepXplore, random and adversarial augmentation [5]

Another sign that coverage based testing is a valid way of testing and training DNN is that it has been shown that each neuron in a network works in a more specialized way instead of generalized which has lead to the conclusion that if every neuron has a certain function then the network would be improved by having been subjected to training data that utilises every neuron in one way or another [20].

In neuron coverage there is still a problem that exists in code coverage, there might be a perfect coverage of the code or neurons that exists but that in itself will not be a complete verification of the system. In the same way that code coverage lacks the possibility to warn the tester of bugs caused by missing implementation, neuron based coverage has the same problem.

C. Adversarial testing

As ML is getting more and more applications, adversarial testing is getting more and more important. As mentioned previously failing to test against adversarial attacks could have fatal consequences. The algorithms and methods currently proposed for adversarial testing mostly works by producing adversarial examples that has a high probability of fooling the system. This tests the robustness of the classifier as you can see how high the misclassification rate is when introducing the adversarial examples.

This however does not exclude the possibility to perform an adversarial attack as it is impossible to test every kind of perturbation to the data. By having a robust classifier which is able to correctly classify most data, regardless of perturbations, means that it is harder to fool the model, but it does not rule out that there still exists a way to fool it. The current research tries to solve this by focusing on algorithms and models that introduce perturbations designed specifically to be able to fool state-of-the-art classifiers. By figuring out the weaknesses the current classifiers has the information could be used to build a more robust model.

D. Validation

ML is a relatively new subject and because of that there arises a question of the validity of the research texts referenced in this article. The majority of the articles were written from 2015 and later and based on our search for other articles regarding the same area of research they seem to be up to date. However since a lot of the tools described in our references might be subject to newer versions it might be reasonable to

see if this comparison holds when this is read in the future. ML is a fast-growing field and testing of ML systems is an even newer field of study. Many papers are being published and ideas discussed and there is not yet a consensus on how it should be done.

When identifying the problems we have discussed in the paper we settled for three distinct problems. There are of course other issues but these were the ones we researched and the ones our research claimed to be of importance. From this we analysed and conducted further research into the three test methods that each solved one of the problems. Once again there are many more test methods and if more time spent researching every single one of them other conclusions could arise but due to time limits and the scale of the paper this was not possible. However these methods were the three most commonly observed methods which is why we picked them and how we validate our findings. [3]

The research papers we have read and collected our data from were published and peer-reviewed, and while this doesn't mean they were correct, given the recent date of publication they seem to be discussing the newest and most relevant ideas of the field.

IV. Conclusion

This section will include a summation and conclusions derived from the analysis of the literary works we have studied. The conclusion presented will try to answer how each of the testing techniques solve part of the problems that were introduced in the beginning, the test oracle problem, test adequacy criteria and generating test inputs and how the techniques complement each other in the process.

A. Issues from analysis and its affect

Based on the analysis, there were some issues with each method and if they would have an impact for solving the problems of the different methods. For metamorphic testing the issue is primarily that a deep understanding of the system is needed in order to construct the test oracle, this might lead to it being harder to use the method but it does not invalidate the method as a technique to use for the test oracle problem.

Neuron based coverage has been shown to demonstrate a higher result in accuracy in test results. Even if the method is new the evidence for it being able to validate seems sufficient, we determine that neuron based coverage has a use in validating the test results and that it serves as part of the solution to the test adequacy criteria.

The problems with adversarial testing was shown in the analysis to be that is hard to predict all kinds of perturbation to the data. Despite this we conclude that the method itself delivers a method for constructing new test input data.

B. Testing vs Training

For ML the testing and training of the systems are intertwined in a way which is described in Fig. 5, it sets out to repeat the training and testing until satisfactory results are achieved. For a ML system, the testing is done to cover two parts. The first one is to make sure that the DNN produces the right output for a certain input from a test dataset and if not more training is required for it to work better. Secondly testing is done as a way to validate if enough types of input has been tested, this is to make sure that new corner cases will not jeopardize the behaviour of the DNN. In this case more training is also required but there is also a matter of finding the right type of training data.

The methods presented in this paper discusses how they each have a solution to one or both of these types of testing and here in the discussion we argue that they complement each other for that purpose.

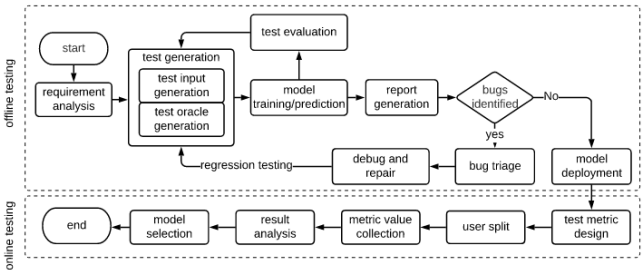


Fig. 5. Workflow for machine learning systems [21]

After evaluating the three different techniques in this paper, there seems to be a connection between:

- Metamorphic testing - test oracle generation/automatic test input generation
- Neuron coverage - test evaluation
- Adversarial testing - test input generation

For the explanation between metamorphic testing and the test oracle, it serves as a tool for validating that the ML system is sure of its answer to a given test input and as explained in the introduction by the same method it serves to generate new test inputs.

Test evaluation is often based on how high the percentages of right answers are when there is a clear right answer. Another part of test evaluation would be how to determine if the results from the test data set is good enough to be more general or if the ML system would be specialized in that types of data but not others. By this definition the neuron coverage covers the second part of that definition by evaluating if the test result should be considered to be good enough, since by itself it will not affect the testing results but will give the testers a measurement of if the ML systems itself can be considered to being tested and trained to an acceptable degree.

When generating new test inputs, adversarial testing is made to manipulate the test data in order to fool the system but at the same time it generates more training data for the DNN.

We can see that all these methods cover different parts of the workflow in ML systems and work in combination with each other, for instance adversarial testing would if it is done correctly lead to DNN that utilise a higher range of its neurons which would lead to a higher neuron coverage measurement and the metamorphic testing would make give

us a measurement if more adversarial test input generation would be needed.

Our conclusion is that by combining these different types of testing a more well performing type of DNN can be constructed than if a DNN was trained and tested by any of the testing techniques individually. This ties back to testing of regular software in when different types of white and black-box techniques often are used in combination to achieve a greater degree of testing the software since each has its advantages and disadvantages.

C. Future work

This project has given an insight into three of the most common techniques when testing ML systems. As ML is rapidly increasing in popularity there is a high amount of research output in this field. This means that the results presented in this paper may become outdated very quickly as advances are being made in the field. One way to build on this project would be to, at a future date, compare what has changed when it comes to ML systems and which of the issues has been solved.

Another way to build on this project would be to compare even more testing methods, and so get an even better overview of the methods currently available and in turn also what the future may hold. Some alternative testing methods for ML methods which we have not discussed in this paper are:

- Mutation testing
- Concolic testing
- Evolutionary computing
- Multi-implementation testing
  [3]

V. Contribution Statement

Each author has concentrated on one part of ML testing methods. Axel researched and wrote about adversarial testing, Zack researched and wrote about metamorphic testing and Artur wrote and researched about neuron coverage. The texts were then reviewed by the other authors. The other sections of the paper, e.g. introduction, were co-written in a joint effort by all three authors. This contributed to an even workload for the paper.

References

[1] C. Ziegler, "Na google self-driving car caused a crash for the first time," 2016.

[2] J. Bolte, A. Bär, D. Lipinski, and T. Fingscheidt, "Towards corner case detection for autonomous driving," *CoRR*, vol. abs/1902.09184, 2019.

[3] S. Sherin, M. Z. Iqbal, *et al.*, "A systematic mapping study on testing of machine learning programs," *arXiv preprint arXiv:1907.09427*, 2019.

[4] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE transactions on software engineering*, vol. 41, no. 5, pp. 507–525, 2014.

[5] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," *CoRR*, vol. abs/1705.06640, 2017.

[6] S. Anderson, "The road to autonomous vehicles presented at mit's em tech digital conference," 2016.

[7] Z. Q. Zhou and L. Sun, "Metamorphic testing of driverless cars," *Commun. ACM*, vol. 62, no. 3, pp. 61–67, 2019.

[8] C. Murphy, G. Kaiser, L. Hu, and L. Wu, "Properties of machine learning applications for use in metamorphic testing.," in *Proceedings of the Twentieth International Conference on Software Engineering & Knowledge Engineering*, pp. 867–872, 2008.

[9] X. Xie, J. W. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Testing and validating machine learning classifiers by metamorphic testing," *Journal of Systems and Software*, vol. 84, no. 4, pp. 544–558, 2011.

[10] L. Ma, F. Juefei-Xu, J. Sun, C. Chen, T. Su, F. Zhang, M. Xue, B. Li, L. Li, Y. Liu, J. Zhao, and Y. Wang, "Deepgauge: Comprehensive and multi-granularity testing criteria for gauging the robustness of deep learning systems," *CoRR*, vol. abs/1803.07519, 2018.

[11] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[12] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pp. 506–519, ACM, 2017.

[13] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

[14] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, 2018.

[15] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. H. Tse, and Z. Q. Zhou, "Metamorphic testing: A review of challenges and opportunities.," *ACM Computing Surveys*, vol. 51, no. 1, pp. 4:1 – 27, 2018.

[16] N. Dalvi, P. Domingos, S. Sanghai, D. Verma, *et al.*, "Adversarial classification," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 99–108, ACM, 2004.

[17] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[18] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 427–436, 2015.

[19] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[20] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," *arXiv preprint arXiv:1506.06579*, 2015.

[21] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," *CoRR*, vol. abs/1906.10742, 2019.