Artificial Intelligence Applied to Software Testing: A Literature Review

Rui Lima, António Miguel Rosado da Cruz, Jorge Ribeiro
Instituto Politécnico de Viana do Castelo
Rua Escola Industrial e Comercial de Nun'Álvares, 34
4900-347, Viana do Castelo, Portugal
ruirochalima1@gmail.com, miguel.cruz@estg.ipvc.pt, jribeiro@estg.ipvc.pt

Abstract — **In the last few years Artificial Intelligence (AI) algorithms and Machine Learning (ML) approaches have been successfully applied in real-world scenarios like commerce, industry and digital services, but they are not a widespread reality in Software Testing. Due to the complexity of software testing, most of the work of AI/ML applied to it is still academic. This paper briefly presents the state of the art in the field of software testing, applying ML approaches and AI algorithms. The progress analysis of the AI and ML methods used for this purpose during the last three years is based on the Scopus Elsevier, web of Science and Google Scholar databases. Algorithms used in software testing have been grouped by test types. The paper also tries to create relations between the main AI approaches and which type of tests they are applied to, in particular white-box, grey-box and black-box software testing types. We conclude that black-box testing is, by far, the preferred method of software testing, when AI is applied, and all three methods of ML (supervised, unsupervised and reinforcement) are commonly used in black-box testing being the "clustering" technique, Artificial Neural Networks and Genetic Algorithms applied to "fuzzing" and regression testing.**

Keywords – Software Testing, Test Pattern, Artificial Intelligence, Machine Learning, Artificial Neural Network, Genetic Algorithm.

I. Introduction

Software testing is the main way of validating software against the defined requirements and accounts for about half of the cost and time of development [1] and it is argued that improvements in system infrastructure could save up to a third of its costs [2]. On the other hand, in the last decades the concepts of Artificial Intelligence (AI) and Machine Learning (ML) have been successfully used to explore the potentialities of the data in different fields [3]. ML [4] is used to teach machines how to handle the data more efficiently simulating the learning concept of the rational beings and can be implemented with AI algorithms (or techniques) reflecting the paradigms/approaches of the rational characteristics as connectionnist, genetic, statistical and probabilities, case based, etc. With the AI algorithms, and based on ML approach, it is possible to explore and extract information in order to classify, associate, optimize, clustering, forecasting, identify patterns, etc.

It is known that AI techniques provide predictive models to be used for multiple engineering purposes [5], but it still isn't a popular implementation for checking Systems Under Test (SUT) correctness. It seems logic to apply AI into software testing, however the lack of an oracle (the mechanism that distinguishes correct and incorrect behavior of the SUT) is currently a bottleneck [6]. AI is still rarely used for error detection in SUT because of the lack of automation of the oracle problem. The exception is regression testing where the correct performance of the SUT can be derived from its previous version behavior [7].

In this context, this paper briefly presents the state of the art and explores the available literature in software testing applying AI/ML approaches/algorithms. The progress of the AI and ML methods used for this purpose during the last three years is analyzed. The paper also tries to create relations between the main AI approaches and which type of tests they are applied to. For this study, research articles were downloaded from databases such as Scopus, Elsevier, web of Science and Google Scholar, between the year 2017 and December 2019. The search terms used have been "artificial-intelligence", "software-testing" and both these keywords. It was not used the term "Machine Learning" by the fact that most of the papers found implemented AI techniques (or hybrid [8]) and not specifically using a ML approach (as supervised, unsupervised or reinforced learning), where one of these learning types can be implemented with one or more AI algorithms.

The paper is organized as follows. In Section II we present an overview of the ML approaches, AI algorithms (and a brief relation between the ML and AI) and software test types referred in this paper. Section III presents an overview of AI applied to software testing. The results are analyzed and discussed in Section IV. In Section V, a conclusion is given.

II. Definitions

In this section, the most popular Artificial Intelligence (AI) algorithms and Machine Learning (ML) approaches are described with the main focus on the software testing field.

From the origins, the definitions of AI and ML are strongly connected. John McCarthy (1959), defined AI algorithms as "a branch of computer science dealing with the simulation of intelligent behavior in computers" [3], and Tom Mitchell (1997) defined Machine Learning as "the study of computer algorithms that allow computer programs to automatically improve through experience" [4]. Following the learning concept of rational beings, ML is structured in learning types (supervised, unsupervised and reinforcement learning) and AI is organized in paradigms/approaches of the rational

2020 15th Iberian Conference on Information Systems and Technologies (CISTI)
24 – 27 June 2020, Seville, Spain
ISBN: 978-989-54659-0-3

characteristics as connectionist, genetic, statistical and probabilities, case based or inductive rules materialized with Artificial Neural Networks, Genetic Algorithms, naive Bayes, Case Based Reasoning, inductive rules, or decision trees. With the AI algorithms and based on a ML approach it is possible to explore and extract information to classify, associate, optimize, clustering, forecasting, identify patterns, etc.

A. Machine Learning Approaches

Machine Learning approaches (or methods) are classified in three main categories and, for each ML approach, there's one or more AI algorithm that implements the learning approach:

1) Supervised Learning: In order to make predictions of a given set of samples, the training data is given all the right answers for the target variable, and the algorithm's job is to replicate the right answer based on the data set [13]. This type of learning is used for classification and regression purposes of AI objetives. Thus, the objective of this method is to classify the free parameters while the output errors are minimized [17]. So, the set might be separated into training data and testing data, to calculate accuracy and loss of a particular algorithm.

2) Unsupervised Learning: This method is used to discover hidden patterns in data that has no target variable (expected outcomes) and organizes the data into a group of clusters to describe its structure. In unsupervised learning no training data is provided to the system, and is usually more challenging than the other methods, so it is more often used for data exploration [3] [4] [13] [15]. Algorithms commonly used for this method might be k-means (where the user specifies the desired number of clusters) or hierarchical clustering (where the user doesn't need to commit to a particular number of clusters), both of which attempt to group objects together based on their similarity [14]. This type of learning is used for density extraction of the data and clustering purposes of AI objetives.

3) Reinforcement Learning: Here, the algorithm is constantly adapting to the environment, based on its feedback, which might be positive or negative [21]. The algorithm chooses an action based on each item of the data set and later learns the efectiveness of the decision, changing its strategy to learn better and achieve the best reward for the action taken [4].

B. Artificial Intelligence Algorithms

Besides the large number of AI algorithms (or techniques), when applied to software testing the studies mainly included Artificial Neural Networks (including Support Vector Machines - SVM), k-Nearest Neighbour (kNN) approach, naive Bayes classifier, decision trees and rules induction algorithm. In this section we focus on these AI algorithms:

1) *Artificial Neural Network:* Artificial Neural Networks (ANN) are function approximators commonly used to approximate continuous states and actions. Typically, they are better suited to classify large space states that couldn't be processed otherwise [9]. They are composed of an input layer (with the input data), an output layer (capable of classifying the data) and one or more hidden layers that process the information. A simple ANN only has one hidden layer while a deep ANN has two or more (Figure 1).

The unit of ANN is the neuron and the connections between neurons are called links or synapses. The neuron takes a number of inputs $x_1, x_2, \dots, x_n \in \mathbb{R}$ and give an output $o \in \mathbb{R}$. The most common design for a neuron is the perceptron and can be represented by 2 equations: the transfer function and the activation function [10]. The transfer function is as:

$$z(x_i)=b+\sum_{i=1}^{n} w_i x_i \tag{1}$$

Where b is the bias constant, which can be used to delay the triggering of the activation function (that might help to create a better model), and w_i represents the weight given to the input.

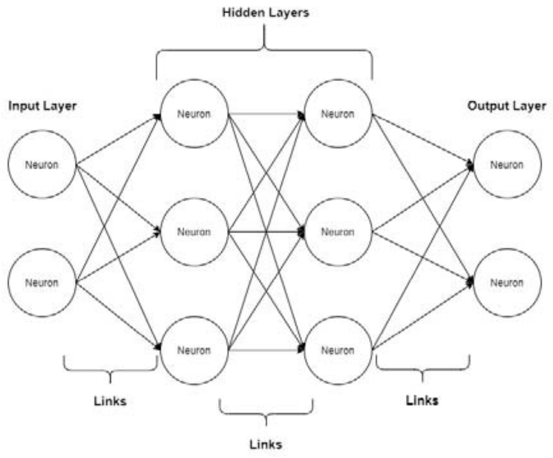

Figure 1. Diagram of a Deep ANN

There are multiple types of activation functions. One of the most commons is the binary step activation function:

$$o(z)=\begin{cases} 1, & z>0 \\ -1, & z\leq 0 \end{cases} \tag{2}$$

In this case the activation function outputs a 1 or -1, depending on a threshold (in this case 0) [10].

The weight (for each perceptron) starts as a random guess and is updated according to the following rule:

$$\Delta w_i=\eta(t-o)x_i \tag{3}$$

Where η is the learning rate (a positive constant), t is the target output, o is the current output for the input x_i. If t and o are equal the desired output is reached, and the weight is not updated anymore [10].

2) Genetic Algorithm: A Genetic Algorithm (GA) is an optimization strategy that mimics the principles of Darwinian theory of evolution. GA is commonly used to solve non-linear, multimodal and discontinuous optimization problems [11]. The algorithm needs to have two things defined: a fitness function, capable of evaluating the best solutions for a problem, and a way to represent the "DNA" of the candidate solutions.

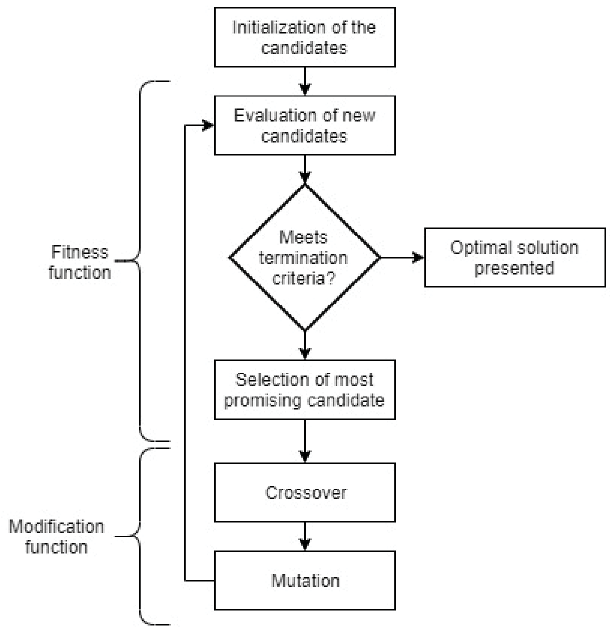
Initialization of the candidates
Evaluation of new candidates
Meets termination criteria?
Optimal solution presented
Selection of most promising candidate
Crossover
Mutation
Fitness function
Modification function

Figure 2. GA Flowchart.

The workflow of a GA (refer to Figure 2) starts with the initialization of the candidate solutions, and then the evaluation of those solutions by the fitness function. If the termination criteria are met, the optimal solution is presented. Otherwise the most promising candidates are selected to be modified. The candidates change involves crossover and then mutation. Crossover generates new candidates by combining existing ordered segments from existing solutions, while mutation randomly changes parts of an existing solution into a new one. This process is repeated until the optimal solution is meet [11].

When solving a GA problem, instead of coding a specific solution, one should provide rules that a solution must fulfill in order to be accepted (allowing to solve problems with a large quantity of potential solutions) [12].

For each ML approach there's one or more AI algorithms that implements the learning approach. As we mentioned before, Supervised Learning is used to solve classification problems (for target variables with discrete values) or regression problems (for target variables with continuous values) [18]. Examples of AI algorithms used with this approach are naive bayes, decision trees, support vector machines (SVM), ANN, recurrent neural networks (RNN) or convolutional neural networks (CNN). RNNs are usually applied in sequential data [19] and CNNs are common solutions for large scale inputs [20], such as sequential data or images.

For Unsupervised Learning, the algorithms commonly used are the k-means (where the user specifies the desired number of clusters) or hierarchical clustering (where the user doesn't need to commit to a particular number of clusters), both of which attempt to group objects together based on their similarity [14]. Autoencoders can also be used for feature extraction, generating new features from the original set of features with the help of ANN [14]. Autoencoders can be used in a variety of problems, such as: anomaly detection, denoising, pretraining, similarity detection or manual output generation. Autoencoders are usually an unsupervised means to a supervised end [14]. Another AI algorithm/technique is Bayesian networks, which is also an unsupervised algorithm [16], but unlike machine learning solutions (that can only deduce correlation in datasets), it can be used to try to infer causation between features.

With Reinforcement Learning, unlike normal machine learning, the objective is to reach a goal and not to look at training datasets. The environment itself might be known or unknown depending on the used algorithm (model-based or model-free algorithms, respectively). Examples of commonly used algorithms are Q-Learning and Deep Q-Learning Networks (DQN) [21].

C. *Software Testing Methods*

A brief definition of the software testing methods, from the point of view of code visibility and analysis (white-box, grey-box and black-box), is given in this section:

1) White-Box Testing: The detailed analysis of a program's code, such as internal logic or structure. In white-box testing, the tester, needs to have full access to the source code to find which part of it is having an unwanted behaviour. This method has a high granularity, so it is potentially the most time-consuming method and also requires skilled testers. On the other hand, it allows for maximum coverage during test scenario writing. It is applicable at unit, integration and system levels of the software testing process, and common techniques are basis path and control flow testing [22].

2) Black-Box Testing: This approach tests the application based on its results, at several levels, without having any knowledge of its internal working. Although this method has limited coverage it is the least time consuming and the test perception is very simple. Commonly used techniques are fuzzing (injecting malformed data for finding implementation bugs) or state transitioning testing (testing state machines and graphical user interface navigation) [22].

3) Grey-Box Testing: This method is used to test an application with limited knowledge of its internal working and full knowledge of the fundamental aspects of the system. In grey-testing the modules are studied for the design of test cases (white-box), but the actual tests are performed in the exposed interfaces (black-box). Since source code is not available the coverage is limited. Some forms of grey-box testing are regression testing (checking if new changes are made to the software) and pattern testing (verifying the architecture and design of the application) [22].

III. AI Applied to Software Testing – Survey Results

This section is divided into three parts. Subsection A presents an overview of the research published about AI and software testing. Subsection B shows a more detailed analysis of research works referencing AI algorithms and software testing methods. Subsection C tries to scrutinize the most popular test cases involving AI and describe some examples observed in some of the papers.

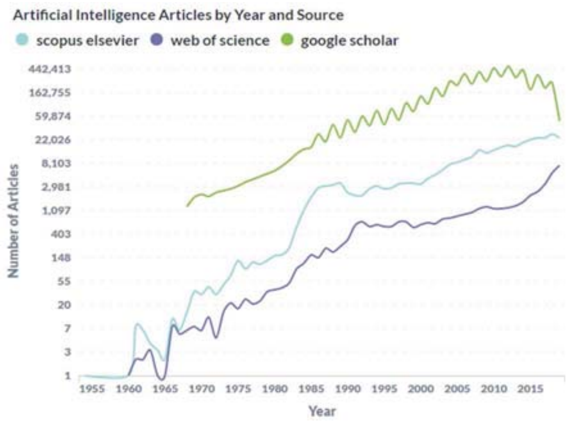

Figure 3. AI Research Published by Year and Source

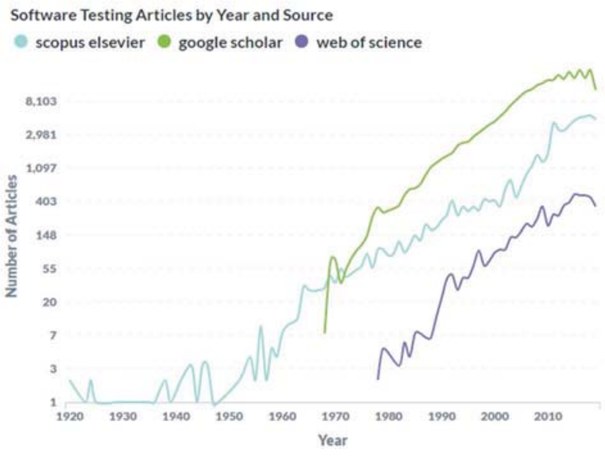

Figure 4. Software Testing Research Published by Year and Source

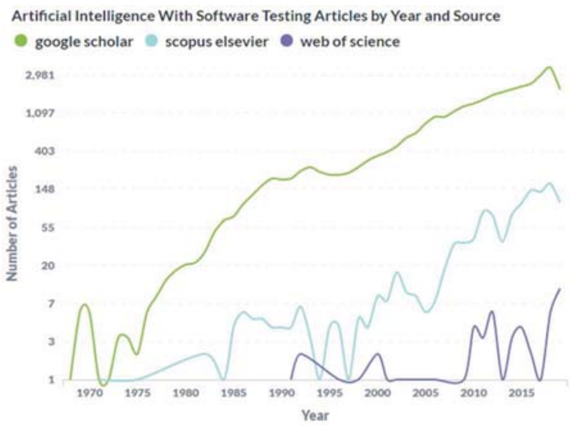

Figure 5. Research with the Keywords Software-Testing and Artificial-Intelligence Published by Year and Source

A. Overview

In order to have a broad view of the AI and software testing research done over time, some scientific search engines and repositories were analysed. Used repositories and papers' search engines have been Google Scholar (https://scholar.google.com), Scopus (https://www.scopus.com) and Web of Science (https://apps.webofknowledge.com). The searches have been done for documents with the keywords "artificial-intelligence" (Figure 3), "software-testing" (Figure 4) and with both the keywords (Figure 5), up to the year 2019.

B. Detailed View

For a more detailed analysis, the total number of papers is determined for combinations between AI learning methods and software testing methods (omitted combinations mean that no papers were found). The used AI learning methods are supervised, unsupervised and reinforcement learning. The used software testing methods are white, black and grey-box. The number of papers is the total sum in the last 3 years, from 2017 up until the time of this writing (January 2020). The measure is made for each of the three search engines: Google Scholar (Figure 6), Scopus (Figure 7) and Web of Science (Figure 8).

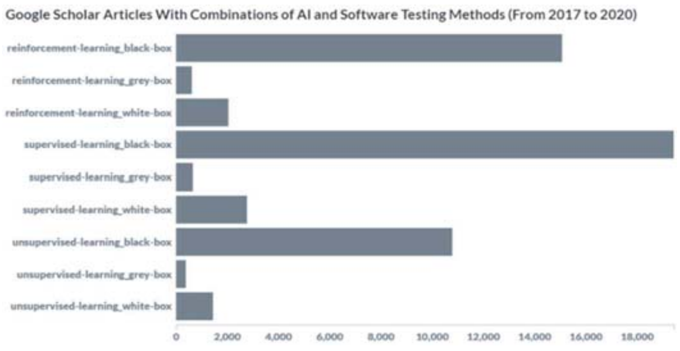

Figure 6. Number of Articles with the Keyword Combination of AI and Software Testing Methods from 2017 to January 2020 for Google Scholar

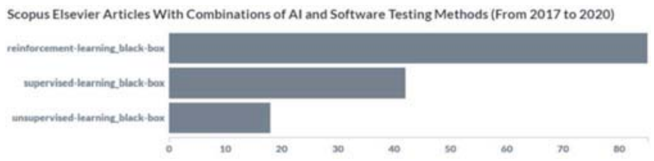

Figure 7. Number of Articles with the Keyword Combination of AI and Software Testing Methods from 2017 to January 2020 for Scopus Elsevier

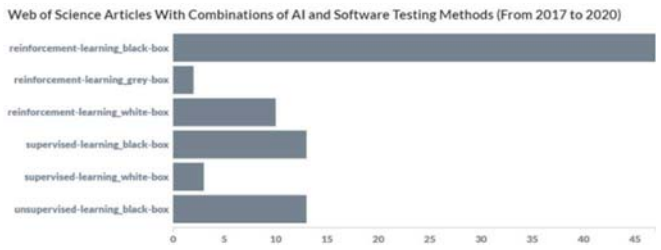

Figure 8. Number of Articles with the Keyword Combination of AI and Software Testing Methods from 2017 to January 2020 for Web of Science

C. Test Patterns

Specific AI algorithms used in software testing are explored to check their popularity. The AI algorithms in question are: clustering, encoders, decision trees, ANN, support vector machines, naïve bayes, bayesian networks, DQN and GA. Keywords for those algorithms are used in conjunction with the "software testing" keyword to have an idea of the most favored (omitted combinations mean that no papers were found). The number of articles is the sum from 2017 to January 2020 for

Google Scholar (Figure 9), Scopus (Figure 10) and Web of Science (Figure 11).

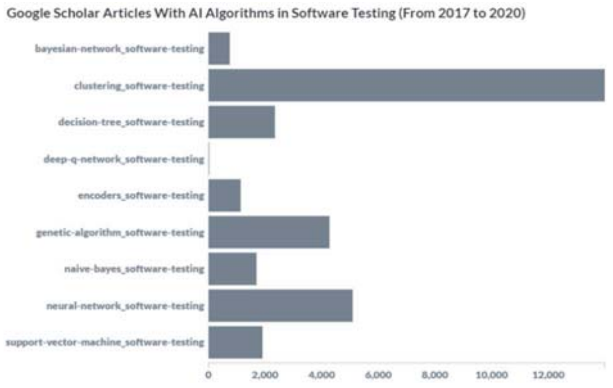

Figure 9. Number of Articles with AI Algorithm Keywords used in Software Testing from 2017 to January 2020 for Google Scholar

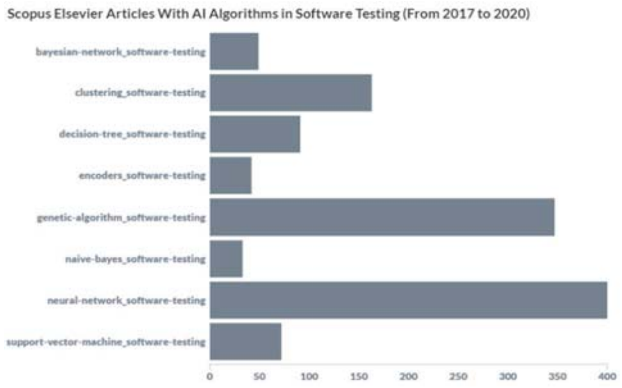

Figure 10. Number of Articles with AI Algorithm Keywords used in Software Testing from 2017 to January 2020 for Scopus Elsevier

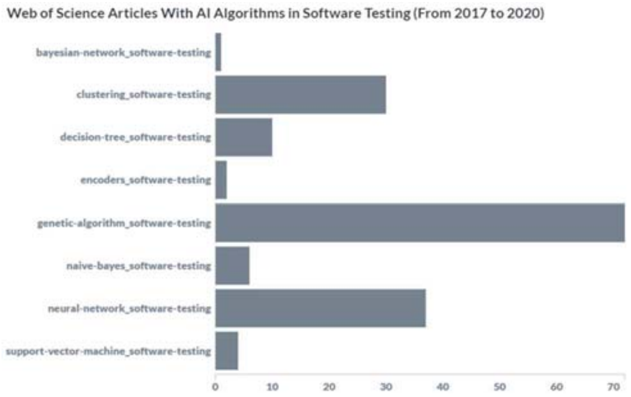

Figure 11. Number of Articles with AI Algorithm Keywords used in Software Testing from 2017 to January 2020 for Web of Science

Software testing techniques used in AI are also explored in the same manner referenced in the previous paragraph. The examined techniques are: basis path testing, data flow testing, fuzzing, regression testing and pattern testing. The time interval is also from 2017 to January 2020 and the researched sources are the same: Google Scholar (Figure 12), Scopus (Figure 13) and Web of Science (Figure 14). If no papers were found for a particular combination, the bar is omitted.

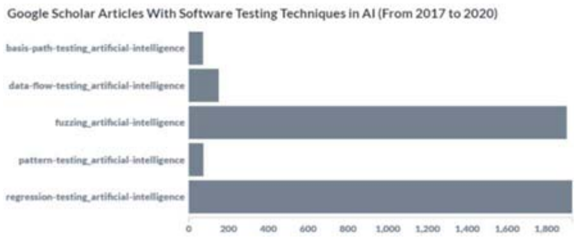

Figure 12. Number of Articles with Software Testing Algorithm Keywords used in AI from 2017 to January 2020 for Google Scholar

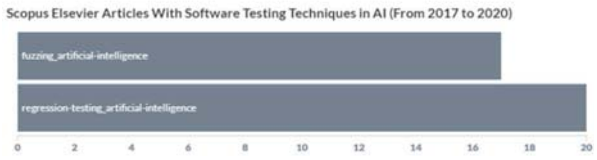

Figure 13. Number of Articles with Software Testing Algorithm Keywords used in AI from 2017 to January 2020 for Scopus Elsevier

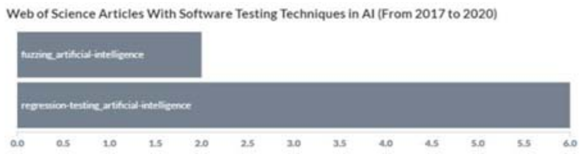

Figure 14. Number of Articles with Software Testing Algorithm Keywords used in AI from 2017 to January 2020 for Web of Science

Some popular papers [23][24][25] present methods that make the code exploration more efficient. The tools extract automatically the code interface and generate path tests following a determined algorithm. One of the papers [24] uses a tool (scalable automated guided execution or SAGE) that implements a search algorithm, called “generational search”, that tries to cover the maximum output using the minimum input combinations, to save the number of necessary tests. In another case [23] the tool (directed automated random testing or DART) is capable of extracting automatically the program interface, generate path tests and make a dynamic analysis to verify the code behavior and generate new tests and input data. A Microsoft program (Program Exploration tool or Pex) for exploratory white-box unit testing is also referred in a paper [25] and is capable of automatically generating new paths.

An example of AI in regression testing is the use of ANN in reinforcement learning, presented in [9], allowing the algorithm to adapt when new test cases are added or removed. This pattern is capable of selecting and giving priority to test cases according to their duration, last execution and error history. In [26], an algorithm is proposed (MultiWalk) for being used in the reduction of the set of tests to reduce their execution time as a whole. Another paper [27] investigates a group of optimization algorithms for test suit reduction. Those algorithms are compared between themselves and a random search algorithm (used for reference).

IV. Result Analysis

Figures 3, 4 and 5 show that the keywords “artificial intelligence” and “software testing” gained an exponential popularity in the last few decades. But, the last year (2019) presents a drop of research made about those subjects. Papers with both keywords present follow the same pattern. Figure 6, 7 and 8 show that, in the last three years, the AI solutions where mostly applied to “black-box” testing methods. For Google Scholar, the most popular ML method (in the past 3 years) used in software testing is supervised learning, followed by reinforcement learning, however, for Scopus Elsevier and Web of Science reinforcement learning was the most popular. Although not as popular as the two methods referenced above, unsupervised learning shows to be a very common occurrence in black-box testing. White-box and grey-box testing do not seem to use AI quite as often.

The most popular AI algorithms appear to be “clustering” (a common unsupervised learning method), ANN (used in supervised and reinforcement learning) and GA (commonly used in reinforcement learning), as shown in Figures 9, 10 and 11. The most common software testing types performed by AI algorithms seem to be “fuzzing” and regression testing (Figures 12, 13 and 14). The most cited papers read describe test patterns that correlate with the analyzed data. Some examples are ANN algorithms, used with reinforcement learning, to perform regression testing [9], optimization algorithms used for test suit reduction [26][27] and “fuzzing” testing used for path exploration in code [23][24][25].

V. Conclusions

The apparent decrease of research (regarding AI and software testing) published in 2019, shown in Figures 3, 4 and 5, could be due to the fact that some papers from the last year are still unindexed, or it could indicate the beginning of a third AI winter, where new AI development stalls [28].

Black-box testing is, by far, the software testing method that most benefit takes from AI. All three methods ML (supervised, unsupervised and reinforcement) are commonly used in black-box testing. Arguably, the most common AI algorithms used for software testing are “clustering”, ANN and GA, applied to “fuzzing” and regression testing. The results show that the majority of the observed AI solutions are used as a form of optimization. The algorithms are used, in most cases, to help cover most of the code with the least amount of test cases. For that to occur AI is used: (i) to find efficient ways to cover the code; (ii) prioritize more pertinent test cases; (iii) reduce the quantity of test cases, filtering tests with more coverage, less time-consuming or with a bigger fail track record.

Since the test phase is usually a very time intensive phase of any software project, not all tests can be fulfilled before the publishing deadline. Because of that, it is important to apply tests in a clever manner. One of the solutions for an efficient testing phase can be the routine application of AI in software testing.

References

[1] Myers, G. J., Sandler, C., & Badgett, T. (2011). The art of software testing. John Wiley & Sons.

[2] Planning, S. (2002). The economic impacts of inadequate infrastructure for software testing. National Institute of Standards and Technology.

[3] Haenlein, Michael & Kaplan, Andreas. (2019). A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence. California Management Review.

[4] Mitchell, T. M. (1997). Machine Learning. New York: McGraw-Hill. ISBN: 978-0-07-042807-2.

[5] Groz, R., Simao, A., Bremond, N., & Oriat, C. (2018). Revisiting AI and testing methods to infer FSM models of black-box systems. In IEEE/ACM 13th Int. Work. on Automation of Software Test,(pp.16-19).

[6] Barr, E. T., Harman, M., McMinn, P., Shahbaz, M., & Yoo, S. (2014). The oracle problem in software testing: A survey. IEEE transactions on software engineering, 41(5), (pp:507-525).

[7] Shahamiri, S. R., Kadir, W. M. N. W., & Mohd-Hashim, S. Z. (2009). A comparative study on automated software test oracle methods. In 2009 4th Int. Conf. on Software Engineering Advances (pp. 140-145).

[8] Shafiabady, N., Lee, L. H., Rajkumar, et al. (2016). Using unsupervised clustering approach to train the support vector machine for text classification. Neurocomputing, 211, (pp:4–10).

[9] Spieker, H., Gotlieb, A., Marijan, D., & Mossige, M. (2018). Reinforcement learning for automatic test case prioritization and selection in continuous integration. arXiv preprint arXiv:1811.04122.

[10] Kraus, D. (2018). Machine Learning and Evolutionary Computing for GUI-based Regression Testing. arXiv preprint arXiv:1802.03768.

[11] Windisch, A., Wappler, S., & Wegener, J. (2007). Applying particle swarm optimization to software testing. In Procs of the 9th annual conference on Genetic and evolutionary computation (pp. 1121-1128).

[12] Sheppard, C. (2017). Genetic algorithms with python. Smashwords Ed.

[13] Dangeti, P. (2017). Statistics for machine learning. Packt Publishing Ltd.

[14] Patel, A. A. (2018). Hands-On Unsupervised Learning Using Python: How to Build Applied Machine Learning Solutions from Unlabeled Data. O'Reilly Media, Incorporated.

[15] Li, X., & Wong, K. C. (Eds.). (2019). Natural Computing for Unsupervised Learning. Springer International Publishing.

[16] Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. Machine learning, 29(2-3), pp:131-163.

[17] Suresh, S., Sundararajan, N., & Savitha, R. (2013). Supervised learning with complex-valued neural networks (pp. 125-132). Berlin: Springer.

[18] Huang, T. M., Kecman, V., & Kopriva, I. (2006). Kernel based algorithms for mining huge data sets (Vol. 1). Heidelberg: Springer.

[19] Mikolov, T., Karafiát, M., Burget, L., Černocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In 11th annual conf. of the international speech communication association.

[20] Gulli, A., & Pal, S. (2017). Deep learning with Keras. Packt Publ. Ltd.

[21] Nandy, A., & Biswas, M. (2017). Reinforcement Learning: With Open AI, TensorFlow and Keras Using Python. Apress.

[22] Khan, M. E., & Khan, F. (2012). A comparative study of white box, black box and grey box testing techniques. Int. J. Adv. Comput. Sci. Appl, 3(6).

[23] Godefroid, P., Klarlund, N., & Sen, K. (2005). DART: directed automated random testing. In Proc. of 2005 ACM SIGPLAN conference on Programming language design and implementation (pp. 213-223).

[24] Godefroid, P., Levin, M. Y., & Molnar, D. (2012). SAGE: whitebox fuzzing for security testing. Queue, 10(1), 20-27.

[25] Tillmann, N., De Halleux, J., & Xie, T. (2014). Transferring an automated test generation tool to practice: From Pex to Fakes and Code Digger. In Procs. of the 29th ACM/IEEE Int. Conf on Automated software engineering (pp. 385-396).

[26] Chi, Z., Xuan, J., et al. (2017). Multi-level random walk for software test suite reduction. IEEE Comp. Intelligence Magazine, 12(2), (pp.24-33).

[27] Sahin, O., & Akay, B. (2016). Comparisons of metaheuristic algorithms and fitness functions on software test data generation. Applied Soft Computing, 49, 1202-1214.

[28] Yasnitsky L.N. (2020). Whether Be New “Winter” of Artificial Intelligence?. In: Antipova T. (eds) Integrated Science in Digital Age. ICIS 2019. Lecture Notes in Networks and Systems, vol 78. Springer, Cham.