Different ways of improving DevOps with Machine learning

Marcus Jonsson Ewerbring

April 20, 2020

Contents

1 Introduction

DevOps is getting more popular by companies, where the goal is to minimize the development time and provide well functioning systems [15]. Machine learning (ML) is a collection name for different learning methods that can predict output patterns from input patterns [10]. This ability makes it possible for the methods find patterns in data which can lead to efficient solutions.

This essay aims to be a collection of ideas of how different areas in DevOps could be improved by ML methods and how they can be applied. In order to locate these areas and to find earlier machine learning implementations used within DevOps, a litterature stude was conducted. The applicability of integrating each area in to a company were evaluated and suggestions on how to improve DevOps were constructed.

2 Background

This section aims to give the reader a understanding on what machine learning is and how the suggested methods works. Section 2.1 explains what machine learning is and how it works. Section 2.2 - 2.3 introduce two machine learning methods, section 2.4 explains what principal component analysis is and section 2.5 explains the concept of genetic algorithm.

2.1 What is Machine learning

Machine learning is a technique within the area of artificial intelligence [14]. A machine learning method requires input data, which is divided into *training data* and *test data*. If the data has a label attached to it which explains what the data represents, it is called *supervised learning*. The training data is used by the system to find correlations between the training data and its representations [3]. For example, if we have a machine learning method that should be able to determine to content of an image, we expect it to output "human" if we enter an image of a human and not outputting "car", see figure 1.

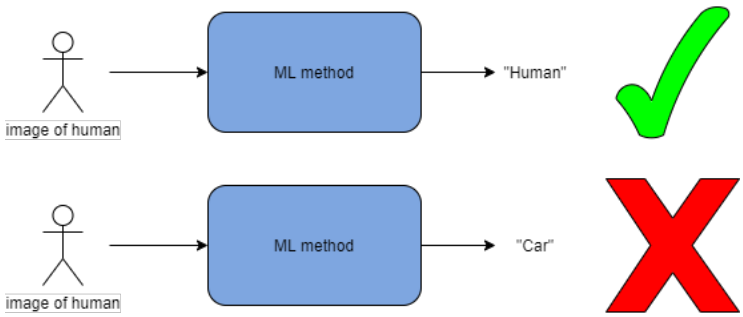

Figure 1: Illustration of how a machine learning method could interprets an image of human

The test data is used to test how well the network is performing on data it has not seen before. This determines how well the system preforms and is usually outputted as an percentage of correctly classified data samples [3].

2.2 Artificial Neural Network and MLP

An *artificial neural network* (ANN) is a machine learning method that tries to resemble connection between neurons in the human brain [3]. The network consists of perceptrons[18] which are a nodes that takes input from other nodes and output a value. The perceptrons are connected to each other via directed weights. The output of the perceptron is calculated by equation 1 where w is weights, x is output from previous node and b is the bias. The sum of the weights*input + bias is then taken through a sigmoid function that determines the output of the node.[3]

$$output = sigmoid(\sum_i w_i * x_i + b) \tag{1}$$

A network consists of three types of layers, input layer, hidden layer and output layer. Each layer consists of several perceptrons and are fully connected to the upcoming layer. The input layer is assigned the input values which is propagated to the hidden layer and finally to the output layer. The perecptrons outputs in the output layer will represent what the network thinks the input is representing, see figure 2.[3]

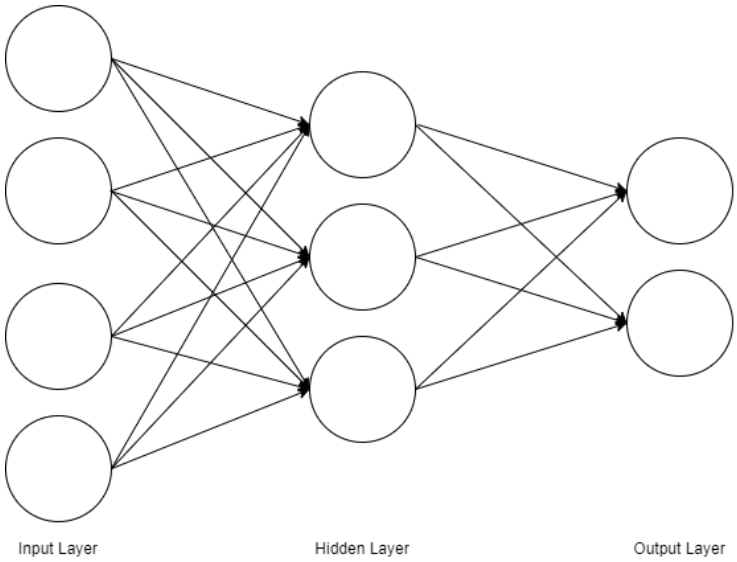

Figure 2: Illustration of how a artificial neural network looks like

A multi-layer perceptron network (MLP) is an ANN consisting of several hidden layers. Which makes it possible to model more complex problems in the network.[17]

2.3 Decision tree

Decision tree is a machine learning method that consists of different logical rules following a tree structure. Lets say that we have a decision tree that predict what the user will do today, see figure 3.

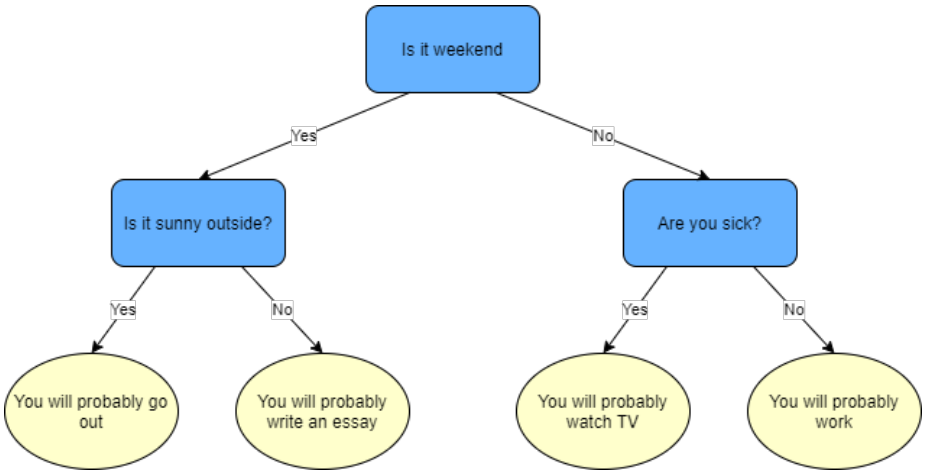

Figure 3: Fictional example of how a decision tree trying to predict what a user will do a day can look like.

The tree consists of yes and no questions, the output of the method will be one of the decision trees branches. For example, if it is a weekend and it is sunny outside the method will output "You will probably go out". The tree is constructed by determining each nodes *Information gain* which is a measurement on how much information the network gains by getting an answer to that question. The tree places the node with most information gain in the top of the tree.[7]

Decision trees prone to overfit which means that they are performing good on training data but bad on test data [7].

2.4 Principal Components Analysis (PCA)

Principal component analysis is a method to reduce the dimensionality of data in order to retrieve the key characteristics for a data, see figure 4. This method is used to pre-process the data before it is sent to a machine learning method. PCA is used to find the relevant feature in the input data [7].

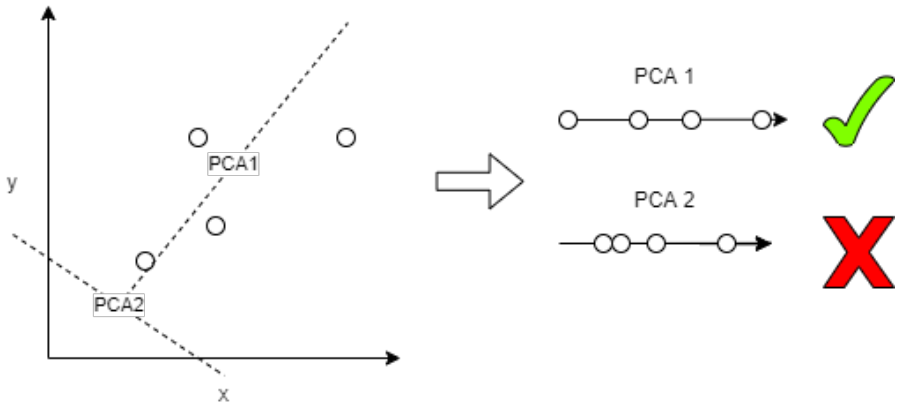

Figure 4: Illustration of two possible PCA from a 2-D dataset, the PCA with largest distance is picked.

The picked PCA is the PCA with largest distance is picked as the representation of the data [7].

2.5 Genetic algorithm

A genetic algorithm is a way to minimize or maximize a parameter. The algorithm construct a population were each is assigned random parameters. A limited amount of the population that have the best fitted solution to the problem are chosen to become parents. A new solution is constructed by using the parents as guide where a random mutation is applied to make the child different from the parents. This makes the solution converge against a maxima [11, 9].

3 Areas in DevOps that could be improved by Machine learning

In this section the findings of the literature study are presented. During the literature study it was discovered that not much work has been done where DevOps and machine learning were combined. The discovered areas where attempts had been made to improve DevOps where the following.

- Software failure detection.
- Anomaly detection.
- Software testing.

These areas are presented below together with reports that tries to improve the area.

3.1 Software failure detection

Machine learning could be used as way to detect when a software is about to fail, this could be done by letting a machine learning technique monitor a system and try to detect behaviors that result in a system failure.

In a report from J. Alonso, et al [1] a decision tree was trained with different parameters from a system in order to detect a system failure. Some of the parameters used were response time, system load, disk utilization and more. The authors simulated aging of the software by inserting memory leaks in the system. The author explains that the methods performed best when the number of threads and memory usage was the cause to the crash. The authors stated "predict with great accuracy the time to crash, when it was near"[1]. The time of crash differed 10% from the predicted one, which the authors regard as a success. They suggest that more research in this area is needed.

3.2 Anomaly detection

Machine learning methods could improve the are of anomaly detection, by increasing the amounts of correctly identified anomalies. Anomaly detection is a software that can detect unusual behavior in data [13]. This could be used to detect a intrusion on a system which the following report is trying to do.

In a report from Amira Sayed A. Aziz, et al [2] the authors attempted to build a intrusion detection system (IDS) with higher accuracy at successfully identify an attack. An IDS is software that monitors a network in order to detect attacks

or other malicious activity [16]. The system that Amira Sayed A. Aziz, et al built consists of 3 parts specified below.

- Layer 1, feature selection.
- Layer 2, detector generation.
- Layer 3, classification of anomalies by different ML methods.

In layer 1 the authors used PCA, see section 2.4, in order to determine which unique features the training data contains. Layer 2 generates detectors that discriminate between normal behavior and anomalous behavior, this was done by a genetic algorithm, see section 2.5. In layer 3 the behavior classified as anomaly is inputted to ML method to confirm if it is an attack or not.

The implementation could detect a denial of service (DOS), probe attack, R2L attack and U2R attack. The author noted that different implementations of decision tree gave in general the best classification accuracy, however MLP gave best result for R2L attacks, see section 2.2 . The system could detect a DOS attack with 81.97% accuracy, prob attack 65,42% accuracy, R2L attack 33.33% accuracy and U2R attack 20.35% accuracy.

3.3 Software testing

Software testing is a subject about testing a program in different ways to identify bug and errors in the code [12]. In a report from Lionel C. Briand [4] the authors summarize some ways that machine learning could be used to improve software testing in different ways.

The author mentions that the machine learning algorithm RUBAR has been used for debugging/fault localization. RUBAR is a rule-based statement ranking of decision tree [5]. The machine learning algorithm learns conditions that results in application failure and assumes that test cases failing under similar conditions have the same error. However, the author points out that the method needs some guidance in form of categories and choices that can result in failure. With given rules the machine learning method could detect error with an accuracy of more than 90%. [4]

The writers described a way to automate black box [8] testing with MELBA which means machine learning based refinement of black box [4]. Black box testing is when we input parameters to a program and look if the output corresponds to the expected output. In black box testing the tester does not know the internal structure of the program. The MELBA method requires an input consisting of a test suite and catagory-partitioning. Where catagory-partitioning are different properties of parameters that can influence the tests. The MELBA method could then connect test problems like missclassification, unused categories, etc to different categories that could be cause to the problem.

4 Applicability

In section 3 some areas found during the literature study were explained. Several reports where cited where each had an idea or solution on how machine learning could improve DevOps. In this section it is discussed if these ideas are applicable in the industry or at a company to increase efficiency of different DevOps areas.

DevOps is commonly illustrated by infinity symbol, which is divided in to 7 different steps, plan/coding, create/building, verify/testing, package, release, configure and monitor [15], see figure 5.

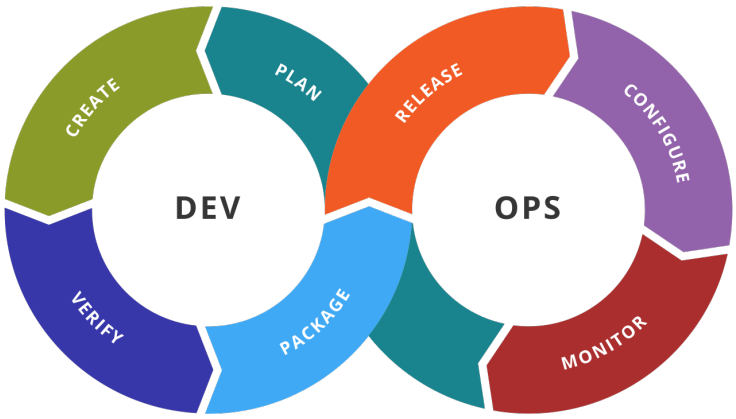

Figure 5: Illustration of the different parts in DevOps, the image is taken from Wikipedia Commons [6].

In the remaining part of this section the applicability in a company for each found area will be discussed.

Software failure detection

In section 3.1 a method for detecting software failure were proposed from J. Alonso, et al. By being able to detect when a system is about to fail, we could minimize the damage brought by it by speeding up the recovery process. This would benefit the monitoring part of DevOps by giving the producing company a warning when their system is about to fail. However, as the authors of the report mentioned, the method could detect a failure with a time difference of 10% when the program was close to fail. What the authors meant with close was not clearly defined in the report and this technology does not seem to be finished yet. This type of approach is not applicable in the industry yet, however when more research has been conducted in the area it could result in a improvement of DevOps.

Anomaly detection

The technology proposed by Amira Sayed A. Aziz, et al in section 3.2 could detect a DOS attack with 81.97% accuracy, this could improve the release, con-

figure and monitor part of the DevOps model by preventing system downtime. The technology was not equally successful in discovering other types of attacks. This technology could be used as an addition to existing IDS in order to increase detection rate of DOS attacks. It could result in a patch for existing IDS which would make it applicable for all the users of that IDS. If more research is conducted in the field, the detection rate for other types of intrusion may be increased and improve DevOps.

Software testing

Several suggestions about how software testing could be improved by machine learning was given in section 3.3 by Lionel C. Briand. He suggested a approach to identify cause of failed tests and a way for the debugging tool to find suspicious statements that could result in a failure of the program. However, the author mentions that both requires some kind of guidance in order to work properly. This affects the applicability of the program because the company needs to identify these categories and create the machine learning method. If they successfully manage to create these categories and machine learning method the company could save time by having the debugging and testing time reduced.

5 Suggestions and reflection

In this section my ideas on how DevOps could be improved by machine learning is discussed. I suggest that a system consisting of a combination of the machine learning methods proposed in section 3 should be investigated. In figure 6 my suggestion is illustrated on which parts of DevOps it could improve.

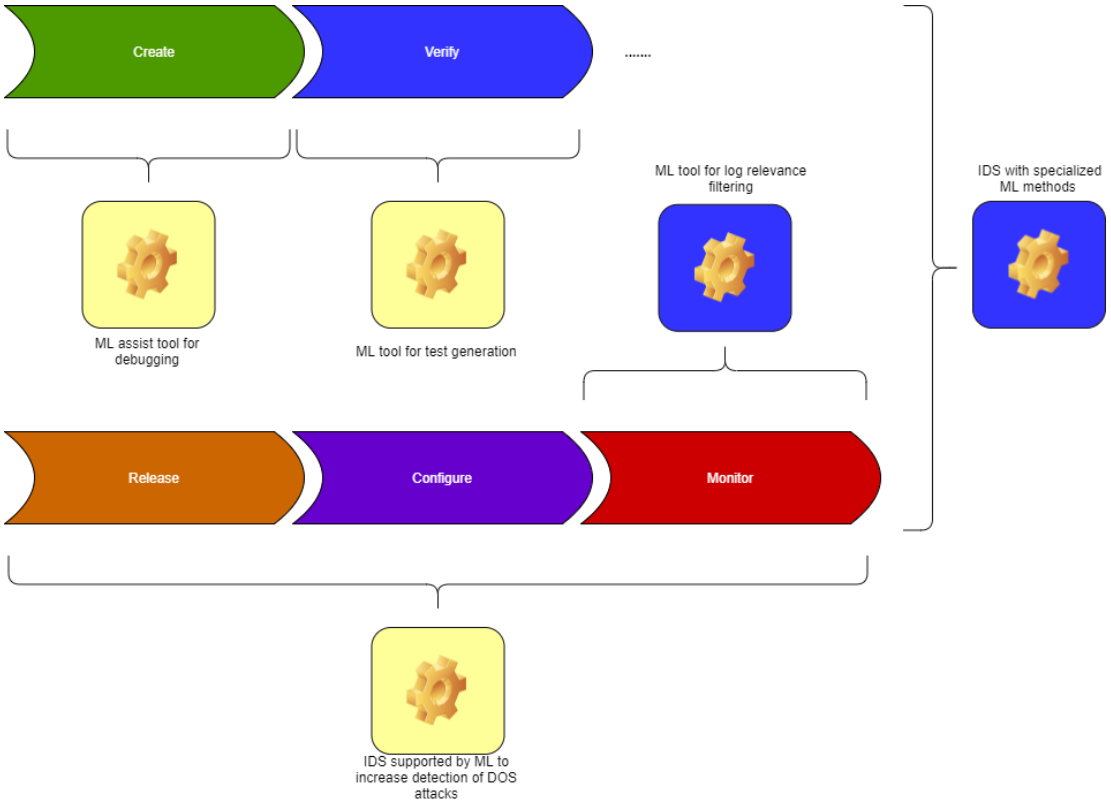

Figure 6: Illustration of suggested improvements

The yellow images in figure 6 represent one of the methods that were found during the literature study and the blue images represent an area that I think could be improved by machine learning. The proposed combination of methods is:

- ML assist tool for debugging.
- ML tool for test generation.
- IDS supported by ML to increase detection of DOS attacks
- ML tool for log relevance filtering
- IDS with specialized ML methods

In the bullet list above the two last bullets are two additional areas that I think could be improved by machine learning. Machine learning algorithms can find patterns in data, this could be used to find the most relevant logs from a system. The system could filter out log report that is not relevant for the problem that a user is trying to investigate.

The IDS system explained in section 3.2 they managed to construct a method that could detect DOS attacks with higher accuracy. Since the machine learning method were able to detect one type of attack well, I suggest that several different versions of that machine learning method should be used. Each of the methods should be specified on one type of intrusion. If the IDS is capable to detect several kinds of intrusion it could give more protection to the entire company which benefits the entire DevOps model.

Below some pros and cons for the suggestion is presented.

Pros	**Cons**
• It affect many different areas in DevOps. • It could save money by reducing downtime and damage done by different attacks. • The area is new, improvements in the area i likely to appear.	• It requires many different machine learning methods to be implemented. • The detection rate for DOS attacks were 80%, this results in that 20% of the attacks is undetected. • The area is new which could result in that this model becomes outdated.

Table 1: Pros and cons of the suggestion

6 Conclusion

In this essay different ways to improve DevOps with machine learning were proposed. A background of different machine learning techniques was given to increase the readers understanding of machine learning. A literature study was conducted to find existing attempts to improve different areas with machine learning. The areas found were software failure detection, anomaly detection and software testing. The applicability of each method on a company was investigated. The solution for software failure detection could not be used since more research were needed for the method to give good results. The anomaly detection system could be used, however only for detection of DOS attacks. It was suggested to combine this with existing IDS to construct a patch to in-

crease applicability. Machine learning could improve the area software testing, however the techniques requires that categories that could cause errors are given.

A suggestion was proposed where the previous techniques were combined to decrease downtime and increase security of a company. There was also suggested that system logs could be filtered by machine learning methods to find the most relevant logs to an error. The model could be hard to implement, however it could increase different areas of DevOps in a company.

References

[1] J. Alonso et al. "Adaptive on-line software aging prediction based on machine learning". In: *2010 IEEE/IFIP International Conference on Dependable Systems Networks (DSN)*. 2010, pp. 507–516.

[2] A. S. A. Aziz et al. "Multi-layer hybrid machine learning techniques for anomalies detection and classification approach". In: *13th International Conference on Hybrid Intelligent Systems (HIS 2013)*. 2013, pp. 215–220.

[3] Giuseppe Bonaccorso. *Machine Learning Algorithms*. Packt Publishing Ltd, July 2017. ISBN: 978-1-78588-962-2.

[4] L. C. Briand. "Novel Applications of Machine Learning in Software Testing". In: *2008 The Eighth International Conference on Quality Software*. 2008, pp. 3–10.

[5] L. C. Briand, Y. Labiche, and X. Liu. "Using Machine Learning to Support Debugging with Tarantula". In: *The 18th IEEE International Symposium on Software Reliability (ISSRE '07)*. 2007, pp. 137–146.

[6] Wikipedia commons. *File:Devops-toolchain.svg*. June 2019. URL: `https://commons.wikimedia.org/wiki/File:Devops-toolchain.svg`.

[7] Pratap Dangeti. *Statistics for Machine Learning*. Packt Publishing Ltd, July 2017. ISBN: 978-1-78829-575-8.

[8] Software Testing Fundamentals. *Black Box Testing*. URL: `https://commons.wikimedia.org/wiki/File:Devops-toolchain.svg`.

[9] MathWorks. *What Is the Genetic Algorithm?* URL: `https://se.mathworks.com/help/gads/what-is-the-genetic-algorithm.html`.

[10] Gareth James Daniela Witten Trevor Hastie Robert Tibshirani. *An Introduction to Statistical Learning*. 2013. ISBN: 9781461471370.

[11] tutorialspoint. *Genetic Algorithms - Introduction*. URL: `https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_introduction.htm`.

[12] tutorialspoint. *Software Testing Tutorial*. 2020. URL: `https://www.tutorialspoint.com/software_testing/index.htm`.

[13] Wikipedia. *Anomaly detection*. Feb. 2020. URL: `https://en.wikipedia.org/wiki/Anomaly_detection`.

[14] Wikipedia. *Artificial intelligence*. Apr. 2020. URL: `https://en.wikipedia.org/wiki/Artificial_intelligence`.

[15] Wikipedia. *DevOps*. Apr. 2020. URL: https://en.wikipedia.org/wiki/DevOps.

[16] Wikipedia. *Intrusion detection system*. Apr. 2020. URL: https://en.wikipedia.org/wiki/Intrusion_detection_system.

[17] Wikipedia. *Multilayer perceptron*. Oct. 2019. URL: https://en.wikipedia.org/wiki/Multilayer_perceptron.

[18] Wikipedia. *Perceptron*. Apr. 2020. URL: https://en.wikipedia.org/wiki/Perceptron.