BLG 475E, SOFTWARE QUALITY AND TESTING SPRING 2019-2020 HOMEWORK 1

O. Kürşat Karayılan
150140011

1) These days, machine learning has started to be used in more fields. This brings to mind how reliable certain machine learning algorithm is. To find the answer, we need to test machine learning procedures. Self-driving systems and medical treatments that use machine learning play significant role in our lives. People can get harm if not machine learning properly implemented. Thus, it is important to know if a machine learning algorithm is correct, robust and efficient enough.

2)
   a. Test oracle refers to what should be the output of an application in the case of random input. Generating reliable test oracles one of the issues. Passing or failing the test is determined by comparing test oracle with output we get from the machine learning application. Thus, generating reliable test oracles important in order to get reliable test environment. One suggested solution to relieving the test oracle problem is pseudo oracle testing. Another method is metamorphic testing. In pseudo oracle testing we generate multiple independent implementations of same algorithm. Then we compare output of these implementations under the same input.

   b. Another issue is generating effective corner cases. A corner case includes variables or situations at extreme levels. So, it is high chance to get error in the result. If the test conditions are complex, manual collection of corner case is almost impossible. We need to generate corner cases automatically. But, generating them automatically is a challenge. There are three different approaches on generating corner cases. The first approach is changing the value of source case along the direction of objective function's gradient ascent. The second is transformation due to application scenarios, and the last is generating adversarial examples.

   c. Test coverage is a measure used to define the extent to which a program's source code is evaluated with the provided test entries. The higher the scope of a test, the less likely the application under test will have undetected errors. There are several metrics to measure test coverage such as statement coverage, branch coverage, condition coverage, modified condition/decision coverage, multiple condition coverage, etc. But, because of the different rule and logic expressions between ML, new coverage metrics are needed. Pei et al. suggested neuron coverage for the deep neuron networks as the new metric.

d. These days the scale of the ML applications becomes larger and larger. Large scale ML applications cause different types of difficulties to testing systems. First, a lot of test scenarios are needed to execute such a large number of parameters. Test cases, especially corner cases, cannot be easily produced. Second, although much more attention is paid to covering all parameters, it is impossible to evaluate all parameters. Third, there is a desperate need for methods to automatically identify the system's malpractice. Finally, a large amount of time and memory space is consumed during the test run.

3) In the ML test, metamorphic relationships have been extensively studied to solve the problem of test oracle. Many metamorphic relationships are based on transformations of education or test data that are unchanged in the forecast output or expected to yield some expected changes. Another type of test oracle is Cross-Referencing for ML test, including differential Testing and N-version Programming. Differential testing is a traditional software testing technique that detects errors by observing whether similar applications provide different outputs for the same inputs. It is a test run to detect compiler errors.

4) Importance: In order to determine whether the importance-based algorithm that supports DeepImportance to identify important neurons is comfortably going through a strategy that randomly selects such neurons. To find out, they make us DeepImportance to find the most important neurons for the MNIST and Cifar-10, and Udacity datasets, respectively. They conclude that DeepImportance can detect the most important neurons of a deep learning system and those neurons are more sensitive to changes in relevant pixels of a given input.
Diversity: They measured the importance driven metric given the original test set of each dataset and corresponding deep learning systems for multiple values of important neurons. By that way they conclude that DeepImportance with its importance driven coverage criterion can support software engineers to create a diverse test set that comprises semantically different test inputs.
Effectiveness: They compared the importance driven values between an unmodified test set and sets enhanced with perturbed inputs using white noise and adversarial inputs carefully crafted using state-of-the-art adversarial generation techniques. So, they conclude that importance driven is sensitive to adversarial inputs and is effective in detecting misbehaviors in test sets with inputs semantically different than those encountered before.
Correlation: Importance-driven is consistent with criteria based on input surprise and aggregation of neuron property values. Similar to IDC, most of the criteria, experience a similar increase to their coverage results when evaluated using test sets augmented with adversarial inputs.
Layer Sensitivity: They observed that IDC value increases when the analysis is performed on deeper instead of shallow layers. They consider this observation as a confirmation of the ability of DL systems to extract more meaningful features in deeper layers.

5) Neuron coverage is the first white-box testing metric for deep learning systems that can estimate the amount of deep learning logic explored by a set of test inputs. Neuron coverage is defined as a set of test inputs as the ratio of the number of unique activated neurons for all test inputs and the total number of neurons in the Deep Neural Networks. It is considered that a neuron to be activated if its output is higher than a threshold value. On the other hand code coverage is a measurement of how many lines or blocks of the code are executed while the automated tests are running.