June 13, 2019

Rapid software testing using transfer learning

N/A

Follow this and additional works at: https://www.tdcommons.org/dpubs_series

Recommended Citation
N/A, "Rapid software testing using transfer learning", Technical Disclosure Commons, (June 13, 2019)
https://www.tdcommons.org/dpubs_series/2275

This work is licensed under a Creative Commons Attribution 4.0 License.
This Article is brought to you for free and open access by Technical Disclosure Commons. It has been accepted for inclusion in Defensive Publications Series by an authorized administrator of Technical Disclosure Commons.
Rapid software testing using transfer learning

ABSTRACT

A given software version has a set of features and known bugs. A subsequent version of the software is written to add features, modify existing features, and correct known bugs. Testing the subsequent version from scratch is a time-consuming and tedious process. This disclosure describes techniques that leverage knowledge of a certain version of software to accelerate testing of a subsequent version of the software via transfer learning.

KEYWORDS

- Transfer learning
- Software testing
- Automated testing
- Software pattern
- Software logs

BACKGROUND

A given software version has a set of features and known bugs. A subsequent version of the software is written to add features, modify existing features, and correct known bugs. Testing the next version from scratch is a time-consuming and tedious process.

DESCRIPTION

Per the techniques of this disclosure, knowledge of a certain software version is used to accelerate testing of a subsequent version of the software via transfer learning. Transfer learning is a deep learning technique that enables the reuse of a pre-trained model on a new, but similar, problem. Transfer learning can shorten or eliminate the training time needed for the new problem.
Fig. 1: Rapid software testing using transfer learning

Fig. 1 illustrates rapid software testing using transfer learning, per techniques of this disclosure. A software version $x$ (102) that has certain features and known bugs, is characterized by an actual pattern $x$ (108). An example of an actual pattern is the set of logs generated by the software over $N$ periods, each period being, e.g., twenty-four hours. The log over a period can be labeled as *good* or *bad*, depending on whether the operation of the software over the period was error-free or not. Another example of an actual pattern is a set of tuples comprising an input to the software, the corresponding output, and the generated log: \{input, output, log\}.

Changes (104) are made to software version $x$ to obtain software version $y$ (106). Software version $y$ is characterized by an actual pattern $y$ (110).

A transfer learning model (112) accepts as input the actual pattern $x$ produced by software version $x$ and the software change between versions $x$ and $y$ to produce a predicted pattern (114) for software version $y$. 
A comparator (116) accepts as input the actual and predicted patterns for software version \( y \), and produces as output the difference between actual and predicted patterns for software version \( y \). A large difference between the predicted and actual patterns for software version \( y \) is indicative of errors in software version \( y \).

The transfer learning model can be a deep learning model of type support vector machine, logistic regression, decision tree learner, deep neural network, etc.

CONCLUSION

This disclosure describes techniques that leverage knowledge of a certain version of software to accelerate testing of a subsequent version of the software via transfer learning.