Mutation Testing of Convolutional Neural Network (CNN)

Abdulhayyu H Lawal

May 2019

1 Motivation

In the past few years, the use of Deep Learning (DL) has become increasingly an important concept in many applications and automation systems for better analytical decisions. Learning from large amount of unlabeled data, deep learning models complex relationships between the given training data yielding to some output results. The quality of these deep learning architectures remain a major concern. Convolutional Neural Network (CNN), one of the DL architectures has proven to be very effective in the area of visual recognition and classification and have achieved success in many field areas, such as facial identification, objects analysis and in autonomous driving field. However, the quality, correctness and performance of these DL systems present a big challenge especially in highly mission critical applications. Measuring the quality performance of CNN architecture using mutation testing depends mainly on the quality of the test dataset [6], [11]. Mutation Testing (MT) is chosen to be the type of testing technique to be applied to test and measure the quality of test data. Since mutation testing cannot be applied directly to DL systems, by measuring and evaluating of the test data, the performance of CNN can be examined through testing and evaluation of the test suites.

In this thesis, I propose a mutation testing for CNN to evaluate the quality performance of the model which depends mainly on the quality of the prepared test dataset. The testing technique to be applied will test and evaluate the quality of the prepared test data which the CNN will be trained and tested with to examine and evaluate its behavior performance. The thesis will include detail discussion and understanding of the contributions or effects of each of the test mutant operators and how the CNN performs when these mutant operators are introduced. The testing technique is planned to be applied by creating mutated version of the original version of the test data and the CNN model program. Different mutation test operators for input training source data (Mutant Data) and CNN model program (Mutant Program) are designed and injected respectively to create mutated versions and are then tested with given test suite to examine the extent to which the performance of each test mutant operator is detected and how the CNN behaves in regards to the change in the input data or program. The behavior change of the CNN triggered by the mutants if any will provide the insights to examining and evaluating of the model's quality performance.

Another important part of this thesis will be, the evaluation of the CNN performance in autonomous deep driving system. How the CNN output after applying of mutation testing effects or contributes to the driving decision in autonomous driving system with driving decision controller. The CNN model will be tested in another external regular system (The Driving Controller system) under Software Under Test (SUT). The quality behavior performance of the CNN is monitored in the driving decision, after applying the testing technique to the CNN, the driving controller decision is examined and evaluated to check if the change in the behavior of the CNN as a result of the mutant operators injection also triggers different driving decision from the driving controller. The evaluation of the SUT is carried out by examining the extent to which the driving decision behavior is affected because of the change in the CNN behavior. The initial or original driving decision(Unmutated SUT) is compared against the CNN behavior after applying MT (CNN with mutation) [11],[3].

2 Problem Statement

Majority of the research efforts conducted in the deep learning domain focused mainly on building more accurate models that can better achieve their intended goals [10]. However, very little work has been done on assuring the quality and correct performance of the deep learning applications [6],[10]. The need to measure, ensure and evaluate the quality of a convolutional neural network as one of the DL architectures arises and becomes more and more crucial [10]. This thesis looks to address this concern by evaluating the quality performance of a convolutional neural network, examining the model's performance is by testing the test dataset using mutation testing technique that measures quality of test data through faults injection and testing of test suite to detects the faults. The model's behavior difference towards the injected faults is examined and evaluation is made. Same performance of the model on the trained and tested dataset through the test cases is then passed to another external driving controller system in autonomous driving system to examine the behavior difference of the driving controller system with mutated performance result of the CNN. By passing the performance result of CNN on the mutated test dataset to the driving controller, the evaluation conclusion on the quality performance of CNN in an autonomous deep driving system with driving controller will be made.

3 State-of-the-Art and Related work

There has not been many research attempts on how mutation testing technique can be applied to specifically test Deep learning system's quality performance [6]. Although, there are many mutation testing research work designed to test software systems, but proving the quality performance and effectiveness of deep learning systems by applying mutation testing have not been given the attention it deserves [11]. The most recent adequate research work that applied mutation testing on deep learning system was conducted by [6]Lei Ma et al. August 2018.

In this section, I have gathered some of the literature that are relevant to mutation testing, convolutional neural network or related to both. Besides the papers mentioned, there are more other research papers that are read and are considered to be supportive in the implementation of the thesis but not mentioned here.

Lei et al. Aug 2018.[6] The paper proposes a mutation testing framework to measure the quality of deep learning systems. The paper proposed and designed two sets of mutation testing operators, a set of source-level mutation operators to inject faults to the source data and another set of model-level mutation operators to introduce faults to the model program. The paper evaluates the quality of the test case by analyzing the extent to which the injected and introduced faults can be detected. The quality of the DL system is then measured based on the behavior performance of the model to the introduced faults data and the original data.

Muhammed et al. 2017.[1] This paper conducted a review of different convolution neural network architectures, tested and evaluated their performance especially in autonomous driving systems. The paper also explained in details how convolutional neural network are used and applied in autonomous vehicles application, helping to make better critical decisions.

Xie et al. 2011.[10] This is a research paper that focuses on applying metamorphic testing to test and validate Machine learning classifiers. Machine learning classification applications have become more crucial due to their prevalence in society and the paper help to address the quality of such applications or programs by applying the test technique and validating the results. The paper conducted cross-validation and mutation analysis to prove the effectiveness and correctness of the classifiers.

Rene et al. 2014.[5] provides some suggestions on how to improve mutation testing analysis. They examined how mutants faults are differentiated from real faults in software testing.

Vincent et al. 2014.[2] focused on mutation analysis as a way to systematically qualify and improve a set of test data for detecting faults in a program under test. They created faulty versions of programs from original programs by systematically injecting one single fault per version of the program. They measured the ability to highlight the fault injected in each mutated version (killing these mutants).

Junhuo et al. 2017.[4] proposed an approach for validating the classification accuracy of a deep learning framework that includes a convolutional neural network, a deep learning executing environment, and a massive image data set. They proposed and explained approaches to validate deep learning classifiers using metamorphic validation approach.

Goran et al. 2017.[8] showed how mutation testing can be applied to test industrial applications. Their work will help in understanding the importance of mutation testing, quantifying the costs of unproductive mutants and suggesting ways to effectively handle the current challenges of efficiently and effectively applying mutation testing in an industrial-scale software development process.

Chen et al. 2015.[3] developed a model that is built upon the state-of-the-art deep CNN framework that automatically learn image features for estimating driving affordance and driving decision behavior. The framework and model implemented by them will be used in my thesis work.

Xiaowei Huang 2017.[9] talked about verification and testing of deep learning systems. They verify the DL systems by global optimization approach and by layer-by-layer approach with exhaustive search.

Murphy 2016.[7] discussed some of the major network parameters and learning techniques related to convolutional neural network. They provided an overview of some CNN architectures and their recent developments.

4 Research Goal

The main goal of this thesis is to apply mutation testing to convolutional neural network, to evaluate the quality performance of the model and to also examine and evaluate the performance quality of the model in an autonomous deep driving system. In the autonomous driving, the performance behavior of CNN to the mutated test data is examined to evaluate the behavior change of the driving decision from the driving controller in the autonomous deep learning system as a result of injecting faults mutant operators.

The test is planned to be carried out by first, training and testing of the original unmutated test data and CNN model program and pass the results into the SUT consisting of the driving controller, driving decision results of this test is then compared to the driving decision of the mutated CNN result. The CNN behavior differences between the mutated and unmutated trained and tested result is examined to determined the extent to which the CNN behavior affects the driving decision from the SUT. In the process of evaluating the quality of test cases, the model and that of the SUT the following research questions will be addressed.

- How can mutation testing be applied to convolutional neural network, to examine and evaluate the model's quality performance and its triggering contribution on driving decision in autonomous deep driving system?
- What makes more sense to test and which of the test mutant operator performs better or has the least impact as mutant? Detail discussion and explanation of each test mutant operators.
- Which convolutional neural network architecture type is more resilient to mutation? Atleast two or more CNN architecture will be tested.
- What is the overall quality performance and effectiveness of a convolutional neural network after applying mutation testing and what effect the MT (Mutant Operator) has on driving decision in SUT?

5 Proposed Methodology

5.1 Test Subject

The data for training and testing the model will be collected from TORCS game engine. TORCS game engine as an open source racing car simulator is widely used in artificial intelligence research field and will be used as the source of the training and testing data for this thesis, images will be recorded from a driving scenario. Alternatively, the dataset already recorded by [3] can be reused and mutated as well. Either the training or testing or both fractions of the data can be mutated and compared with the corresponding unmutated data results. From the perspective of the convolutional neural network, a mutation is a change in the CNN's neural architecture which will then be checked to see if after the change there is also change in the model's behavior and also if the output of the driving controller from [3] behaves differently given different CNN behavior input (result from mutated CNN). Mutating of the test data or the model program can be done either by manually introducing mutants or automatic way using some available mutation tools like MutPy. The two CNN architecture type to be considered for the test are; AlexNet and LeNet-5 and each will be analyzed and evaluated.

5.2 Implementation

The implementation of this thesis will involve designing of mutation testing technique that will execute convolutional neural network under different mutations. Mutation is achieved by designing mutation operators that introduce potential faults into the software under test to create mutated versions, the CNN and the driving decision controller together is referred to as the software under test (SUT). Two different sets of mutation operators are to be designed, source-level mutation operators that introduce faults into the source input training data of the deep learning system and model-level mutation operators that introduced faults into the deep learning model program. A set of faulty mutated programs P' are injected into the original program P to slightly modify P original. For example, replacing a '+' sign operator in the program to a '-' sign operator. After the modification or injecting the potential faults, a complete test set T is executed against P and only T', the subset of T that passes the test are used for the mutated testing. Each of the mutant P' is executed on T' and if the test result for a mutant p' of P' is different from that of P, mutant p' is killed otherwise mutant p' survived. After all P' mutants are tested against T', a mutation score is then calculated to find the ratio of the killed mutants ($p'killed$) to all generated mutants p'. The quality of test set can then be determined and the higher the mutation score the more likely to capture real defects in the program and to have different behavior from the model [6].

When testing the SUT, a test without faults/mutants in the CNN program will be carried out and the final driving decision will be noticed and recorded which will later be compared to the mutated results. Mutations or faults will

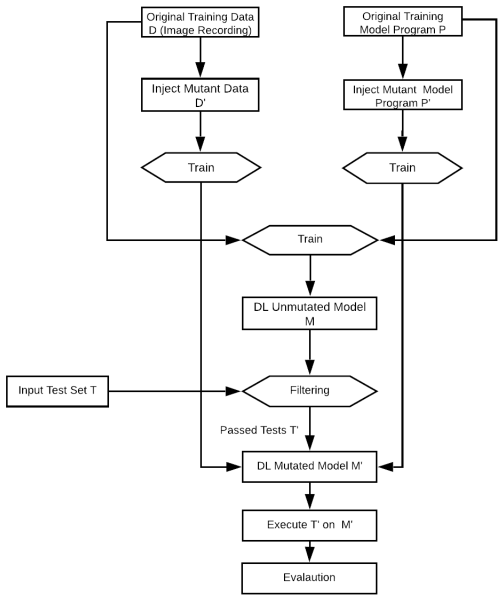

Figure 1: Mutation Testing Architecture [6].

then be introduced and the driving decision from the SUT will be examined to evaluate how the faults introduction changes the behavior of the CNN model which may lead to change in the driving decision. The process of applying the test technique to the convolutional neural network (see Figure 1) and into the software under test (see Figure 2) involves re-training of the models any time faults are injected to obtained mutated versions. Each mutant model version (M_1', M_2'...M_n') [6] generated is executed and analyzed against the test set T in correspondence to original system model M. With a particular given input test set of t ϵ T, M' is analyzed against M and if the results of M and M' are inconsistent then faults/mutants are detected [6]. The software under testing in figure 2 showed the overview process of the system under test. Automated training data mutant generator for mutating the input data and python training program mutant generator will be used to generate the mutants. The implementation will run on Linux and will be built based on Keras consisting converted models from Caffe, and Tensorflow.

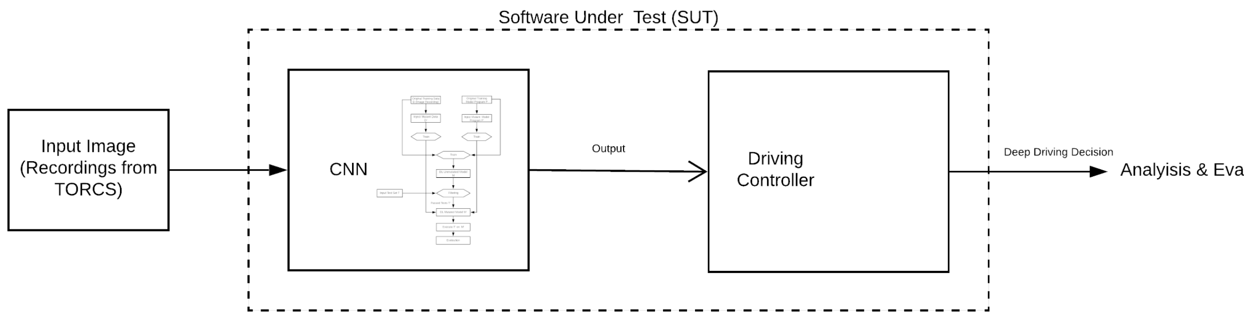

Figure 2: Overview of the Software Under Test.

6 Experiment and Evaluation

6.1 Mutant Operators Experiment

The mutation operators to be used in mutating of the CNN program will be discussed and evaluated in this section. Each mutation operator that performs a modification to the original program by introducing potential faults will be discussed and evaluated to examine the extent to which the modification triggers different behavior of the CNN output and that of the driving decision. The two sets of mutation operators introduced in the implementation section are to be explained and discussed in details in this section. Experimental procedure on each operator will be carried out to demonstrate how the outcome of each operator as a result of the modification changes the CNN behavior and hence affects the driving decision. The following mutation testing operators for the input training data include;

Data Duplication: This operator duplicates a portion of the same data in the training data. Since the training data is planned to be collected from the TORCS simulation environment, some small portion of the collected or recorded image data will be duplicated to serve a as a modification to it.

Data Missing: This mutation operator deletes some specific or selected data part from the training data. With some data deleted, the model will be examined again to evaluate its behavior difference.

Data Property Manipulation: This operator manipulates the properties of some data. This can be done by editing the properties of an image, e.g editing image's pixel dimension or the number of layers in the image.

Data Outliers Introduction: This operator will involve introducing data outliers to the dataset group that do not fit well with rest of the dataset. This is just to indicate that the dataset now contains some additional unfit and unusual data to evaluate the test case.

Data Label Modification: This operator modifies or changes the label of a data, some data points of some of the collected input data are changed to mutate the data.

Noise Perturbation: Noise is added to the training data randomly, some images of the training data will be added or similar disturbance that can cause noise to the data.

Data Shuffle: Training data is re-shuffle in different orders before the training process, this is for the training data to appear to the model in different order. re-shuffle the order of the training data can allow the model to be observed if the re-shuffling impact its behavior or not.

The second set of operators are the CNN program model mutation operators that inject faults to the model's program. As mentioned earlier, mutation tools MutPy will be used to generate the mutants that do not cause the training program to fail. Manually generated operators are also proposed and implemented. The proposed operators for the program include;

Layer Addition: This operator adds a layer to the neuron network. Adding a layer to the neuron CNN program modifies the original neuron network hence testing the quality of the test case and that of the CNN behavior.

Neuron Switch: With this operator, neurons of the same layer are switched to exchange their roles and influence for the next layers.

Weight Shuffling: This operator shuffles the weight of a neuron's connection, previous neuron determines the weight of a layer below it and with this operator the connection weight between the two layers is shuffled.

Layer Deactivation: This operator removes only the transformation effects of a layers, because removing the whole layer would bring inconsistency to the DL model, only the effects are removed and it will be as if the whole layer is deleted.

Neuron Activation Inverse: This operator tries to invert the activation status of a neuron to create non-linear behaviors of the DL model.

Activation Function Removal: The effects of activation function of a whole layer is removed using this operator. The operator removes the effects on the layer level, serving as a modification to the neuron network.

Neuron Effect Block: This operator blocks the neuron effect of all of the connected neurons on the layer following it.

6.2 Evaluation

The evaluation will be based on the mutation score that will be obtained from mutation process execution, the number of faults/mutants killed in the mutated version of the dataset or model program detected by the test case and in comparison against the original version of the system. Mutation score is calculated by comparing the ratio at which the input test set T is able to detect and killed mutants M' against all generated and introduced mutants (number of mutant killed over the number of all mutants generated) [6]. The SUT is evaluated only through the driving decision result of the mutated CNN output version will be compared against the original driving decision output from the CNN training.

Upon completion of this thesis, the answer to the research question on how mutation testing can be applied to evaluate convolutional neural network's quality and its performance in an autonomous driving will be answered and also be able to tell which mutant operator has more impact on the performance quality of the CNN. Depending on the number of convolutional neural network architecture type to be trained and tested, each architecture will be executed and evaluated separately based on its performance quality, highlighting evaluating which architecture is more resilient to mutation testing.

7 Requirement List and Schedule

7.1 Requirement List

The thesis requirement is organized into three categories as shown in Table 1:

- MUST HAVE requirements are the most important part of the thesis that I must implement and accomplish.
- MIGHT HAVE requirements are to support and improve the quality of this thesis.
- MUST NOT requirements define the scope of this thesis that are not planned to be implemented.

7.2 Schedule

The schedule in Table 2 shows how the thesis is planned to be carried out, highlighting time allocation for phases, duration and tasks to be involved in the thesis.

Requirement List	
Requirements Options	Remark
Must Have	• Applying Mutation Testing to deep learning system (convolutional neural network). Examine and evaluate the mutation operators effect on the CNN behavior output result in detail. • Evaluate CNN architecture after examining the quality of the test dataset through MT and observe the effect of the CNN behavior change in the SUT.
Might	• Mutate atleast two different convolutional neural network architecture type. Find out which architecture is more resilient to mutation testing. • Perform the test also on some widely used datasets apart from the training dataset (Recordings from the TORCS environment).
May Not	• No own driving decision controller program will be developed, the driving controller developed by [3] paper will be modified and used. • Sending of output results back to TORCS environment after execution may not be implemented(No visual decision). The output or the driving decision produced after comparing will be analyzed and evaluated.

Table 1: Thesis requirement list.

Schedule		
Phase	Duration in Months	Tasks
Initiation, Planning, Proposal and presentation	1-4 (Before official registration)	topic research, -related material gathering, -state-of-the-art research, proposal documentation, - submission and presentation of proposal
Implementation and Documentation	1-5	reading, -coding, -design, -documentation -discussion/feedback
Testing and Submission	1	final testing, -presentation, -submission, -closing

Table 2: Time allocation

8 Conclusion

By the end of successful implementation of this proposed thesis. There will be two evaluation analysis reports, evaluation analysis on the quality performance of the convolutional neural network and another evaluation analysis on the software under test system that examine how change in CNN's output behavior affects the driving decision. It will become clear to whether mutation testing can be applied and how, to evaluate convolutional neural network's quality and if the same testing technique can also be carried out when convolutional neural network is executed in deep driving application of an autonomous driving system. I proposed to design the different sets of mutation operators that can be used in introducing faults to the model to evaluate its quality performance not only as a single model system but also when put to practice together with other deep driving systems like the driving decision controller. A mutation score framework is to be used to find out the number of mutants detected and killed from the mutated system version, evaluating the quality of the test dataset which will then be useful to evaluate the quality of the CNN and its behavior effect in autonomous driving system.

References

[1] M. Al-Qizwini, I. Barjasteh, H. Al-Qassab, and H. Radha. Deep learning algorithm for autonomous driving using googlenet. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 89–96, June 2017.

[2] Vincent Aranega, Jean-Marie Mottu, Anne Etien, Thomas Degueule, Benoit Baudry, and Jean-Luc Dekeyser. Towards an automation of the mutation analysis dedicated to model transformation. *Software Testing, Verification and Reliability*, 25(5-7):653–683, 2015.

[3] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[4] J. Ding, X. Kang, and X. Hu. Validating a deep learning framework by metamorphic testing. In *2017 IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET)*, pages 28–34, May 2017.

[5] Rene Just, Darioush Jalali, Laura Inozemtseva, Michael D Ernst, Reid Holmes, and Gordon Fraser. Are mutants a valid substitute for real faults in software testing? 11 2014.

[6] Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu, Chao Xie, Li Li, Yang Liu, Jianjun Zhao, et al. Deepmutation: Mutation testing of deep learning systems. In *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*, pages 100–111. IEEE, 2018.

[7] John Murphy. An overview of convolutional neural network architectures for deep learning. 2016.

[8] G. Petrovic, M. Ivankovic, B. Kurtz, P. Ammann, and R. Just. An industrial application of mutation testing: Lessons, challenges, and research directions. In *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 47–53, April 2018.

[9] Youcheng Sun, Xiaowei Huang, and Daniel Kroening. Testing deep neural networks. *CoRR*, abs/1803.04792, 2018.

[10] Xiaoyuan Xie, Joshua WK Ho, Christian Murphy, Gail Kaiser, Baowen Xu, and Tsong Yueh Chen. Testing and validating machine learning classifiers by metamorphic testing. *Journal of Systems and Software*, 84(4):544–558, 2011.

[11] Yuhao Zhang, Yifan Chen, Shing-Chi Cheung, Yingfei Xiong, and Lu Zhang. An empirical study on tensorflow program bugs. In *Proceedings*

of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2018, pages 129–140, New York, NY, USA, 2018. ACM.