Assignment 2 Report

Amirhossein Sohrabbeig - STD: 1744420

2022-03-02

In this report, I use the terms "DNN models" and "AI models" interchangeably.

1 The difference between traditional software and AI model

1.1 Characteristics of traditional software

Traditional software is any program written in high-level programming languages (e.g., C/C++, Java, Python). In traditional programs, each statement performs a specific operation that either modifies the outputs from the preceding statement or alters the program states (e.g., assign new values to variables).

Traditional software represents its logic as control flows designed by human knowledge, whereas a DNN's behavior is defined by the weights of edges and the nonlinear activation functions (determined by the training data). For example, for classifying five from nine, one may notice that the sum of the pixels' values for five in the MNIST dataset is greater than 20,000, while this is not true for nine. Knowing that a practitioner can easily classify them just using if statement. But using AI models, an individual just set-up the architecture, dataset, and training process, and the model has to learn the concept by tuning its parameters.

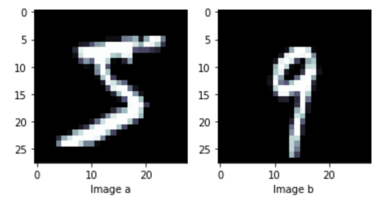


Figure 1: Classification of 5 and 9

The inner workings of DNN models are black-box to humans and as a result, finding erroneous behaviours in DNNs differs from detecting them in traditional software, necessitating new test generation approaches.

1.2 Characteristics of AI model

DNNs, unlike traditional software, are programmed by training data, selected features, and network architectures rather than deterministic algorithms (e.g., number of layers). A DNN, in particular, is made up of several interconnected neurons that are grouped into layers: an input layer, an output layer, and one or more hidden layers. Each neuron is a computer unit that uses an activation function to compute its output. Each neuron in a classical DNN is fully connected to all neurons on the following layer, and each edge has a weight that represents the strength of the connections between neurons. Generally, a DNN can be thought of as a function that transforms a given input into an output, with the aggregated effects from its computation units (i.e., neurons) determining the function.

The absence of interpretability in a DL system, in contrast to traditional software systems with explicit and controllable logic and functionality, makes system analysis and problem discovery challenging, which could stymie its real-world deployment.

2 Quality assurance of DL models

2.1 Testing metric

Code implementation. Please see HW2.ipynb file. The result of running the implemented code shows coverage of **0.14933958244567533** on the test dataset.

2.2 Mutation

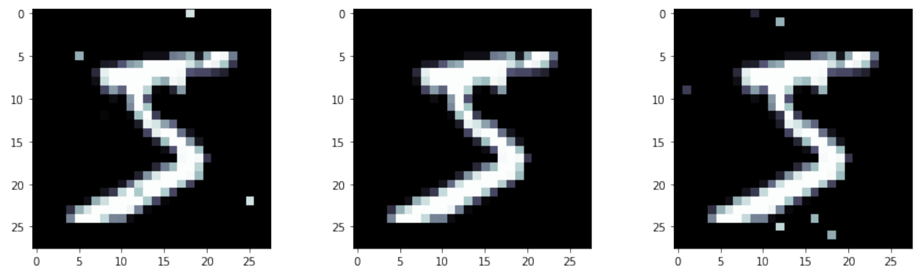
- Answer the question: For all images returned in Algorithm 2, how many mutations are applied for each image if the success flag is True ? How does sequential mutations done in DeepHunter?

  The Metamorphic mutation Algorithm makes at most **TRY_NUM** mutations for each image that it returns.

  An image s' is sequentially mutated from s if s' is generated after a sequence of one-time mutations ($s \xrightarrow{t_0} s_1, s_1 \xrightarrow{t_1} s_2, ..., s_n \xrightarrow{t_n} s'$)(denoted as $s \xrightarrow{t_0,t_1,...,t_n} s'$). It is expected that after a one-time mutation, the new image will retain the same semantics as the old image by using conservative parameters for distinct transformations. However, one image can be sequentially changed from the initial seed during fuzzing, making it impossible to maintain its validity after multiple mutations. We use a conservative strategy that selects the Affine Transformation to be used only once (we assume that mutation with only one affine transformation will not affect the semantics with the carefully selected parameters) in order to keep the semantics of the mutants close to the original seed. An image is more likely to be unrecognisable if it is changed by multiple affine transformations.

To make a new test image, DeepHunter tries to mutate the seeds with a maximum number of trials **TRY_NUM**. If the reference image and seed are identical, the Affine Transformation was not used. Both Affine Transformation and Pixel Value Transformation might be selected in this scenario. Otherwise, it can only use a pixel value transformation. It selects a parameter at random and performs the transformation for the selected transformation t. To see if the new mutant s' is meaningful, DeepHunter computes L_0 and L_∞ between the reference image s'_0 and the new mutant $s_{\prime}$. When the Affine Transformation option is selected, the reference image is updated. If no effective mutation is found after **TRY_NUM** tries, the original image s is added to T.

- Complete Algorithm 2 and attach three images after mutating.



(a) first mutated image (b) second mutated image (c) third mutated image

Figure 2: Three mutated images resulted from Metamorphic algorithm

3 Experiments

Code implementation. Please see HW2.ipynb file.

I followed the first suggestion and created one new test image of each image in the test dataset. After adding mutated images to the test data set and run the coverage metric NBC on the dataset, I reached 100% of coverage.

References

[1] Ma, Lei, et al. "Deepgauge: Multi-granularity testing criteria for deep learning systems." Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. 2018.
[2] Xie, Xiaofei, et al. "Deephunter: a coverage-guided fuzz testing framework for deep neural networks." Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis. 2019