Summary of "DeepXplore: Automated Whitebox Testing of Deep Learning Systems"

Xiangfeng Zhu(zxfeng), Jiachen Liu(amberljc), Chris Chen(zhezheng)

Problem and Motivation

Safety- and security-critical DL systems must be tested systematically to detect and fix ideally any potential flaws or undesired behaviors. Existing DNN testing techniques are not ideal because they require expensive human effort to provide correct labels and, more importantly, they achieve very low test coverage, which will leave different behaviors of DNNs unexplored.

Solution Overview

The key ideas of DeepXplore are the concept of neuron coverage for measuring the parts of a DL system's logic exercised by a set of test inputs based on the number of neurons activated and differential testing, in which multiple models are used to identify erroneous corner cases without manual checks.

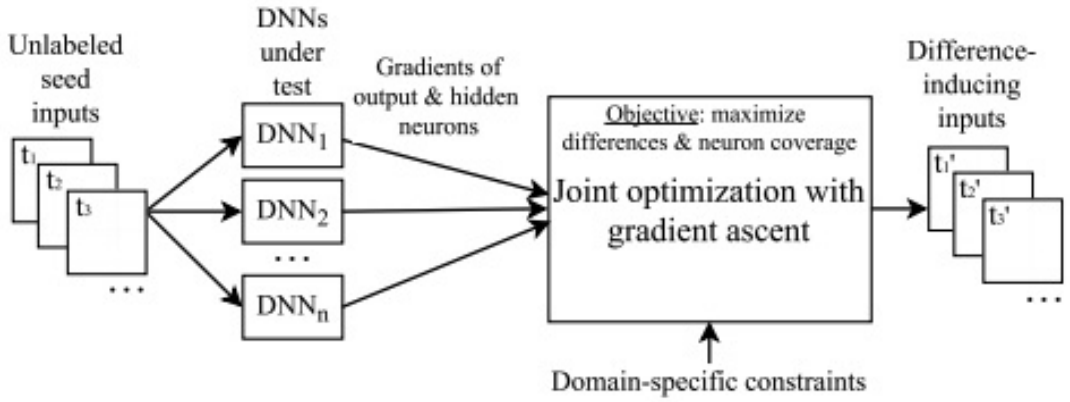

Figure 5: DeepXplore workflow.

DeepXplore takes unlabeled test input as seeds and generates new tests. While generating tests, DeepXplore tries to maximize both neuron coverage and the chances of tests that cause the DNN models to behave differently(i.e., output different labels). Both goals are necessary for thorough testing that exposes erroneous corner cases.

DeepXplore solves the above joint optimization problem for neuron coverage and differential behavior maximization using gradient ascent.

Limitations

One major limitation of DeepXplore is its completeness. Although neuron coverage is clearly a better metric compared to code coverage or random testing, it is still far from complete verification of the model.

In particular, for traditional software testing, if we have an if statement(e.g., if(x < 0) … else …), it's easy to generate tests that cover both branches. However, it's hard to use neuron coverage to generate all possible inputs. In addition, DeepXplore requires users to provide input seeds and make minor changes to the input seeds to get the difference- inducing inputs, but how to pick these input seeds? For an image classification model, if the input seeds do not contain any cat images, how to find bugs that can only be caused by cat images?

Summary of Class Discussion

Is neuron coverage the best metric? What are some alternative coverage measures?

Neuron Coverage has some problems. For example, DeepXplore uses the same threshold as activation evaluation for all the neurons, but it might be suboptimal if the statistical distribution of the output of different neurons is diverse.

One possible alternative approach is feature-guided safety testing, but it is computationally challenging to achieve.

Are the techniques applicable to other applications? (e.g., nlp)

The paper only discussed DNN on image recognition tasks. The idea could be used in NLP models but it's hard to generate meaningful sentences.

Summary of "DeepBase: Deep Inspection of Neural Networks"

Problem and Motivation

There is increasing need about when and why deep learning models work, i.e. the high-level logic. It used to take a researcher days to simply extract activations from a newly trained model and plot them as a heat map. Performing more complex analyses such as identifying patterns in those activations to understand the model can take even longer. DeepBase is such a system able to inspect neural network behaviors at scale. The primary contribution is that DeepBase formalizes Deep Neural Inspection(NDI) and develops a declarative interface to specify DNI analyses.

Hypothesis

Deep neural inspection(DNI) analyses primarily use statistical measures to quantify the affinity between hidden unit behaviors and hypotheses, and simply differ in the specific NN models, hypotheses, or types of hidden unit behaviors that are studied.

DeepBase is able to quickly compute the affinity score between each (hidden units group, hypothesis) pair, given groups of hidden units, hypotheses, and statistical affinity measures.

Solution Overview

To understand how DeepBase works, we need to clarify what are hypotheses and common approaches for interpretation. Hypotheses encode the logic that we search for, Intuitively, is how the model is learning from the dataset.

Approaches for interpretation includes

1. Manual Visual Inspection: Visualize each unit's activations and let users manually check that the units behave as expected.
2. Saliency Analysis: Seek to identify the input symbols that have the largest "effect" on a single or group of units.
3. Statistical Analysis: Using annotated datasets to analyze groups of units.

Generally speaking, DeepBase is a system that provides a declarative abstraction to efficiently express and execute these analyses. DeepBase takes as input a test set, a trained model, a set of Python functions that encode hypotheses of what the model may be learning, and a scoring function, e.g., a measure of statistical dependency. From those inputs, DeepBase generates behaviors for each unit/hypothesis and use GPU to extract unit activations. Then, DeepBase outputs a set of scores that quantify the affinity between the hypotheses and the model's hidden units.

On a system perspective, DeepBase proposes lots of optimization to speed up the DNI.

1. Shared Computation: Put multiple measures/hypotheses into a single Keras computation graph and utilize Keras's graph optimization
2. Early stopping: Stop when the score has converged (error less than preset bound), Stop materializing more data when early stopped, streaming behavior extraction, extract & materialize behaviors in an online fashion

Limitations and Possible Improvements

1. DeepBase can use its inspection results to prune/improve learning models.
2. DeepBase can run DNI in a distributed fashion (store records in the key-value pair, compute pairwise affinity on different nodes, etc)
3. DeepBase may learn from DeepXplore to test what the model fails to learn.

Summary of Class Discussion

Why is DeepBase a DB-oriented Design?

In the paper, it's said using a database can help manage the massive unit and hypothesis behavior matrices that can easily exceed the main memory.

Can DeepBase be used to help ML researchers improve or prune their model?

DeepBase can use hypotheses to replace part to unit to speed up, which could be useful for RNN.

Given such large search space, how does deep base make inspection scalable?

*Algorithm perspective: Consider choices of units, scores, etc… as hyperparameters and perform local search.

*System perspective: Affinity scores are computed pairwise, can be run in the distributed fashion.