Improving the Accuracy of Estimating the Probability of Test Case Generation for Simulink Models Using Machine Learning

TIANCHENG JIN[†1] TAKUYA OGATA[†1]
YUGE LIU[†1] KENJI HISAZUMI[†2]

Abstract: Model-Based Development (MBD) is gaining popularity in a range of fields. In automatic test case generation, a Simulink model is utilized as an input, and test cases that meet decision criteria such as model coverage are output. As a result, test cases can't be generated for models that don't meet the decision criteria. Furthermore, the test case creation success or failure is frequently unknown until the test case generation is run. Suppose the test case generation success or failure is known before the test case generation. In that case, it is possible to encourage the rewriting of the model differently while keeping the functions being same. Some methods estimate whether a test case can be generated using machine learning, but do not achieve sufficient accuracy. This study proposes a new approach to improve accuracy by adopting a graph neural network. The accuracy for LightGBM and Random Forest is about 75%, and the accuracy of GCN and GAT is 81%. The result shows that there are still possibilities for optimization of the method.

Keywords: Model-based development, Machine learning, Simulink

1. Introduction

The development standards for control embedded systems were formerly specified in natural language. Model-based development (MBD)[1] is a software development method based on algorithmic modeling. As a method of growth, it is becoming increasingly popular.

A model generation system that can generate Simulink models at random is built into this work. The model generation mechanism has also been enhanced to enable the production of more complex models. In addition, a machine learning technique is proposed that uses a deep learning architecture such as a graphical neural network to estimate Simulink test case production time using machine learning more accurately.

2. Methodology

2.1 Overview

This section describes a method for estimating whether a test case can be generated that employs machine learning. Model generation is the first step, divided into two parts: graph generation and model transformation. The generated model will then be used to train a test case generated usability estimation method using the generated model. We'll go over the procedure in detail in this section.

2.2 Model Generation

As Figure 1 shows, Graph creation and model transformation are the two aspects of the procedure. Block selection, port matching, and loop elimination are all part of the graph generation process. Parameterization, model creation, and model export are all part of the model conversion process[2].

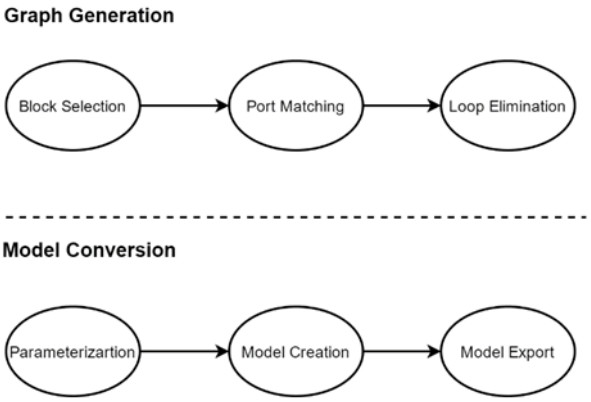

Figure 1: Model Generation Process

2.2.1 Block Selection

In Block Selection, all blocks should be supported by SLDV for the resulting model to be correctly authenticated.

2.2.2 Graph Generating

After block selection, the names of the blocks and the signal kinds for each input and output port must be kept in a list. It contains the block names, the signal types supported by each input port and output port, and the weight of each block.

2.2.3 Avoid Algebraic Loops

An algebraic loop arises when the input signal affects the output signal, and the output signal likewise determines the input signal, resulting in a compilation error. Add a unit delay block to the cycle to eliminate algebraic loops as Figure 2 shows.

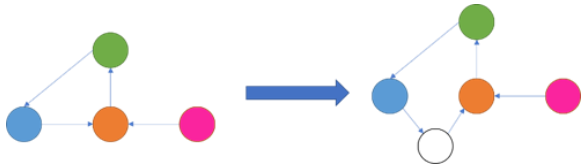

Figure 2: Loop Elimination

†1 Kyushu University
†2 Shibaura Institute of Technology

2.2.4 Graph to Model

Create a list containing the block name and output port number to keep track of line information during the graph production process. The MATLAB engine transfers this data to the MATLAB script, and the Simulink model is generated programmatically.

2.3 Test Case Generation Availability Estimation Method

We present a system that employs supervised machine learning to estimate whether or not to construct test cases with two classes of output using features collected from graphing Simulink models as input in this method, as shown in Figure 3.

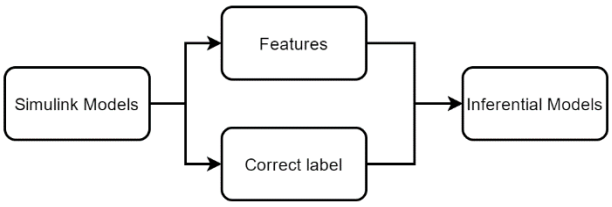

Figure 3: Creation of Inferential Models

2.3.1 Feature Extraction

To do feature extraction, a Simulink model is turned into a directed graph in the feature extraction block. Internally stored variables are used to conduct this modification at the nodes. The following operations are then carried out.

2.3.2 Bag of Nodes

Bag of nodes is a method that counts the number of nodes in a graph and characterizes their frequency. It turns the node information of the longest path into frequency information by taking all the shortest paths from input to output.

Get the Simulink model's list of import blocks, Constant blocks, and Outport blocks. Find the shortest path between all Import and Constant blocks and Outport blocks. On the route, replace the block names with the properties of each block. Replace the Constant block with "Constant + M" for the rewritten blocks on the route, using the number of digits M of the constant value N that the block possesses. If the block L has a Constant block as its input, the blocks are replaced with "L+constant+M" for the replacement blocks on the pathway.

Features that can be derived from an effective graph are extracted. Include the following four characteristics: the average cluster coefficient c in the graph G, the number of nodes in the graph G, n, the number of edges, m, the graph density $d = m/(n * (n-1))$.

2.3.3 Supervised Learning

We utilize a supervised machine learning approach to tackle the two-class classification problem because the test generation possibility is represented by two classes in this method: test case generation possibility and test case generation impossibility. There are a variety of these approaches, however we employed Random Forest [3] and LightGBM [4] in our tests. We use Graph Convolutional Network [5] (GCN) and Graph Attention Network [6] (GAT) as learning models in experiments

3. EVALUATION

This section discusses the tests that were carried out to evaluate and debate the proposed method's performance.

3.1 Model Generation

The model generator size is between 10 and 60 and we compared with existing models from GitHub. The size of generated models is mainly larger than existing models.

3.2 Evaluation Results of Estimation Method

Using the feature extraction method, we compared the recognition accuracy of the trained models. A suitable evaluation metric is employed to assess the suggested approach. The proposed technique uses supervised machine learning to evaluate the possibility of test case creation for two classes of classification problems. The Table 1 shows the outcomes of the evaluation.

Table 1: System evaluation results

	Accuracy	Precision	Recall	F1 score
LightGBM	0.763	0.792	0.694	0.739
Random Forest	0.748	0.766	0.694	0.728
GCN	0.814	0.834	0.771	0.787
GAT	0.817	0.844	0.764	0.803

4. CONCLUSIONS

The purpose of this study is to determine how to complete the development of an automatic Simulink model generator in order to address the issue of a shortage of training data. A method for determining test case generation time and MBD of Simulink model test case generating availability is also described. Internal information from Simulink is used as a feature in a machine learning approach to see if a test case can be generated.

Reference

[1] B. Schätz, A. Pretschner, F. Huber and J. Philipps, "Model-Based Development of Embedded Systems, " in Advances in Object-Oriented Information Systems, pp. 298-311, 2002.

[2] D. K. Chaturvedi, Modeling and simulation of systems using MATLAB and Simulink. CRC press, 2017.

[3] M. Pal, Random forest classifier for remote sensing classification, "International journal of remote sensing," vol.26, no.1, pp.217–222, 2005.

[4] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu,"Lightgbm: A highly efficient gradient boosting decision tree, " Advances in neural information processing systems, pp.3146–3154, 2017.

[5] T.N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in ICLR 2017.

[6] P. Veli˘ckovi´c, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in ICLR 2018.