基于 GRU_LSTM 及 RL 算法的伪随机指令生成器

欧阳有恒[1*]，严大卫[2]
（1. 南京信息工程大学，江苏 南京 210044；
2. 无锡先进技术研究院，江苏 无锡 214125）

摘　要：在 CPU 验证过程中，传统伪随机指令生成器通过生成大量合法无序的指令序列，从而实现功能覆盖率或代码覆盖率的验证目标。然而，没有趋向针对性的指令生成，为达到指标需要耗费大量的人力及时间成本。该文以一款基于精简指令集（RISC-V）自研核心为例，在基于通用验证方法学（Universal Verification Methodology，UVM）的验证平台上设计出一种伪随机指令生成器，并针对普通伪随机指令生成器覆盖率低、收敛速度慢的问题，建立 GRU_LSTM 算法模型，并结合强化学习（Reinforcement Learning，RL）算法构建新算法模型 RLGRU_LSTM 应用于伪随机指令生成过程，并且针对 RL 方向决策，提出了基于霍夫曼编码的 CPU 指令包编码方式训练 opcode 分布，同时融合了 CPU 指令类型和指令间执行顺序因素，快速捕获人工定向验证预料不到的验证盲点，有效加快了代码覆盖率达到预期的进程。该文着重描述伪随机指令生成器及 RLGRU_LSTM 算法对模型训练过程的指导。实验结果表明，与直接使用伪随机指令生成技术相比，该方法在约定伪随机指令条目下，相比传统伪随机方法能提高约 19% 的覆盖率，收敛至目标覆盖率消耗时长减少 22%。

关键词：门控循环单元；长短记忆；强化学习；伪随机指令生成；通用验证方法学
中图分类号：TP301.6；TN492　　**文献标识码**：A　　**文章编号**：1673-629X(2024)02-0078-06
doi：10.3969/j.issn.1673-629X.2024.02.012

Pseudo Random Instruction Generator Based on GRU_LSTM and Reinforcement Learning Algorithms

OUYANG You-heng[1*]，YAN Da-wei[2]
（1. Nanjing University of Information Science and Technology，Nanjing 210044，China；
2. Wuxi Institute of Advanced Technology，Wuxi 214125，China）

Abstract：In the CPU verification process，the traditional pseudo random instruction generator generates a large number of legally unordered instruction sequences to achieve the verification goal of function coverage or code coverage. However，there is no trend towards targeted instruction generation，and it takes a lot of manpower and time to achieve the target. Taking a RISC-V based self-developed core as an example，we design a pesudo random instruction generator on the verification platform based on Universal Verification Methodology（UVM），and establish a GRU_LSTM algorithm model to solve the problems of low coverage and slow convergence of ordinary random instruction. A new algorithm model RLGRU_ LSTM combined with Reinforcement Learning（RL）is applied to the pesudo random instruction generation process. Aiming at the RL direction decision，LSTM proposes a CPU instruction package coding method based on Huffman coding proposed to train opcode distribution，which integrates the CPU instruction type and the execution order factors between instructions，quickly captures the unexpected verification blind spots of manual directional verification，and effectively speeds up the process of code coverage reaching the expected rate. We focus on the description of pesudo random instruction generator and RLGRU_ LSTM algorithm guides the model training process. The experimental results show that compared with the direct use of pesudo random instruction generation technology，the proposed method can improve the coverage rate by about 19% under the agreed pesudo random instruction entry compared with the traditional pseudo random method，and converge to the target coverage rate about 22% earlier.

Key words：gate recurrent unit（GRU）；long short-term memory（LSTM）；reinforcement learning；pesudo random instruction generation；universal verification methodology（UVM）

收稿日期：2023-02-15　　**修回日期**：2023-06-15
基金项目：国家自然科学基金（61732018）
作者简介：欧阳有恒（1996-），男，硕士，通讯作者，研究方向为集成电路；严大卫（1989-），男，硕士，研究方向为集成电路。

0 引　言

随着处理器设计规模的日益增大,结构的复杂程度越来越高,处理器的验证也越来越困难,传统的定向测试虽然依旧无法替代但已经满足不了现在高效的验证需求。伪随机验证能在短时间内产生大量伪随机测试向量,有助于快速捕获人工定向测试预期之外的验证盲点,在现代微处理器验证领域有着非常重要的作用[1]。因此,如何合理有效地生成伪随机指令测试向量,提高覆盖率是伪随机指令生成器的研究热点。目前,人工智能和大数据算法飞速发展,机器学习算法(Machine Learning,ML)也大量用于测试向量生成中。将 ML 引入测试向量生成的早期工作可以追溯到 1997 年[2],其由遗传算法来实现。遗传算法(Genetic Algorithm,GA)被用作微处理器中高速缓存访问仲裁机制 CAAM 的测试生成器,基于遗传算法的生成器产生输入指令集刺激来触发 CAAM 状态,这些状态之前没有从早期运行指令的结果中学习到。结果表明,与传统的伪随机生成器相比,使用遗传算法时的状态数增加了约 13%。后罗春[3]、宋倩[4]、吴海涛[5]等人在此基础上,针对遗传算法的性能进行改进,不断实现机器学习在测试向量生成领域的优化。机器学习通过在验证过程中采用有监督和无监督模型作为测试向量生成器,提供了关于特定模型解决发现问题的有用见解。文献[6]提出使用 3 层神经网络作为激励源,用于一个简单 CPU DUT 的功能验证,以达到断言覆盖率指标;文献[7]提出使用支持向量机(Support Vector Machines,SVM)作为激励生成器来去除传统的伪随机生成器产生的冗余刺激,以 64 位 RISC 处理器作为待测设计(Design Under Test,DUT),结果显示,使用 SVM 方法的总时间,包括模型学习花费的时间,明显少于使用传统伪随机化方法花费的时间并在一定程度上提高了覆盖率;文献[8]对 SVM、深度神经网络(Deep Neural Networks,DNN)和伪随机森林模型(Random Forest,RF)作为四核缓存 DUT 的激励生成器进行了测试,工作结果表明,SVM,DNN 和 RF 的模拟时间分别加快了 68.5%,77% 和 78%。

此后,随着循环神经网络(Recurrent Neural Network,RNN)的广泛应用,逐渐将 RNN 用于指导指令生成。文献[9]提出使用 RNN 来引导生成器,反馈循环从功能和代码覆盖度量中获取输入,修改为处理器生成指令集的约束,以实现覆盖目标。结果表明,与传统的指令生成方法相比,覆盖率能达到总覆盖目标的 85%,但是在长期记忆和反向传播中存在梯度爆炸等问题。随后,瑞士人工智能研究所的 Jurgen Schmidhuber 提出长短期记忆结构(Long Short-Term Memory,LSTM),在此基础上,2014 年 Cho 等[10]提出一个广泛使用的门控循环单元(Gate Recurrent Unit,GRU)。GRU 神经网络引入门机制,结构简单且可解决长期记忆和反向传播中的梯度爆炸问题,但拟合精度却比不上 LSTM。目前在指令生成领域,GRU 与 LSTM 相结合的 GRU_LSTM 算法更受欢迎,不仅训练周期短,而且实现了高拟合精度的目标。文献[11-13]提出将 GRU 与 LSTM 构成组合模型 GRU_LSTM,这种模型比单一模型能更好地预测参数。同时,由于近年来深度学习掀起一股新的人工智能的浪潮,相继在围棋、视频学习领域取得显著成绩,强化学习[14-15]也被应用于指导指令生成,其采用一种奖励机制的学习模式,以便模型能持续地学习信息并更新参数。William Hughes 等[16]提出使用监督和强化学习以高度自动化的方式提供了比伪随机结果更好的结果,从而确保可以在加速的时间尺度和更少的资源下实现完全设计覆盖的设计验证目标。在文献[11-13]的指导下,该文利用伪随机指令生成器,基于 UVM 验证[17]方法学构建验证平台,并在此基础上将 GRU_LSTM 算法与强化学习结合,使用基于霍夫曼编码的伪随机指令包,在训练 GRU_LSTM 模型过程中得到推荐的伪随机指令类型,通过奖励函数对模型参数(影响伪随机指令类别分布)再次优化,自动优化测试用例的约束,得到更优测试用例,以得到更高的代码覆盖率。

1 基于 GRU_LSTM 算法及强化学习算法的伪随机指令生成器

该文的主要任务是使用 GRU 和 LSTM 算法,应用循环递归网络序列记忆特性,挖掘测试指令包序列对覆盖率的影响,将指令包指纹编码序列作为样本,代码覆盖率作为指导,开展模型训练;并在训练完成的 GRU_LSTM 网络输入伪随机指令包的编码序列,得到该序列的最优覆盖率,作为现实奖励,反馈给强化学习,用于指示下一步结合强化学习的新模型训练。本节对 GRU_LSTM 算法的理论基础和强化学习的理论基础进行描述,并在此基础上搭建 RLGRU_LSTM 模型。

1.1 GRU_LSTM 算法

门控循环单元的主要功能是在指令生成过程中捕捉指令序列的时序特性(即指令执行的逻辑顺序),提升数据重构的准确性。作为 RNN 的一种变形单元,GRU 解决了长期记忆和反向传播中的梯度消失和爆炸问题,在达到相同模型性能的前提下其计算量小,且效率更高[18-19],但其在拟合精度上却逊色于 LSTM。因此,该文考虑使用一种第一层为 GRU、第二层为 LSTM 的双层神经网络模型,既有 LSTM 的高精度、高稳定性,又有 GRU 的训练周期短特点。对待输入指令序列进行样本处理,根据霍夫曼编码[20]得到测试伪随机指令包的编码序列,输出是基于验证平台得到的覆盖率,作为损失函数的参数,反馈给模型。

GRU_LSTM 模型第一层由多个 GRU 门控单元组成,融合更新门和重置门,具体结构如图 1 所示。

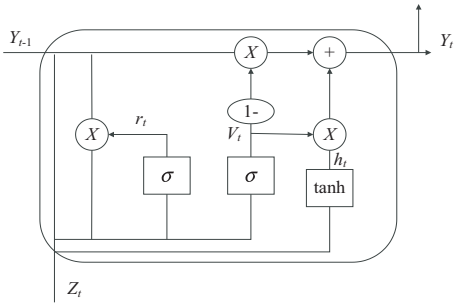


图 1 门控循环单元结构

其中, Z_t 表示当前时刻的输入, Y_t 表示当前时刻的输出, Y_{t-1} 表示上一时刻的输出。

v_t 为更新门,更新门决定了在每一层网络中有多少信息可以传递给未来,计算公式如式 1 所示[21]:

$$v_t = \sigma(\boldsymbol{W}_v * [Y_{t-1}, Z_t] + \boldsymbol{b}_v) \tag{1}$$

其中, σ 表示激活函数 sigmod, $\boldsymbol{W}_v$ 表示重置门的权重矩阵, $\boldsymbol{b}_v$ 表示偏差向量。r_t 为重置门,重置门与更新门作用相反,它决定了当前网络层中有多少信息不能传递到未来,计算公式如式 2 所示[21]:

$$r_t = \sigma(\boldsymbol{W}_r * [Y_{t-1}, Z_t] + \boldsymbol{b}_r) \tag{2}$$

其中, $\boldsymbol{W}_r$ 表示更新门的权重矩阵, $\boldsymbol{b}_r$ 表示偏差向量。根据更新门和重置门结果,GRU 算法计算出候选隐藏状态 h_t ,计算公式如式 3 所示[21]:

$$h_t = \tanh(\boldsymbol{W}_h * [r_t * Y_{t-1}, Z_t] + \boldsymbol{b}_h) \tag{3}$$

其中, tanh 表示激活函数 tanh, $\boldsymbol{W}_h$ 表示权重矩阵, $\boldsymbol{b}_h$ 表示偏差向量。最后,GRU 算法的输出 Y_t 计算公式如式 4 所示[21]:

$$Y_t = (1 - v_t) * Y_{t-1} + v_t * h_t \tag{4}$$

在 GRU 网络层输出后,结果接入 LSTM 网络层,GRU 算法的输出 Y_t 就是 LSTM 的输入。LSTM 的单元结构如图 2 所示。其中, N_t 为当前时刻的输入, U_t 为当前时刻的输出, C_t 为当前时刻的状态, N_{t-1} , U_{t-1} , C_{t-1} 为上一时刻参数。LSTM 有遗忘门、输入门、输出门。每个 LSTM 单元结构的前向传播公式组如式 5 ~ 式 9 所示[21]:

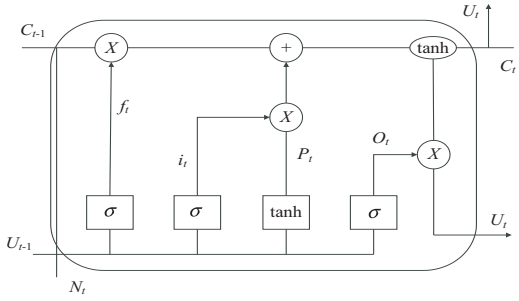


图 2 LSTM 单元结构

$$f_t = \sigma(\boldsymbol{W}_f * [U_{t-1}, N_t] + \boldsymbol{b}_f) \tag{5}$$

$$p_t = \tanh(\boldsymbol{W}_c * [U_{t-1}, N_t] + \boldsymbol{b}_c) \tag{6}$$

$$C_t = f_t C_{t-1} + i_t * p_t \tag{7}$$

$$o_t = \sigma(\boldsymbol{W}_o * [U_{t-1}, N_t] + \boldsymbol{b}_o) \tag{8}$$

$$U_t = o_t \tanh(C_t) \tag{9}$$

其中, $\boldsymbol{W}_f$ 表示遗忘门的权重矩阵, $\boldsymbol{W}_i$, $\boldsymbol{W}_c$ 表示输入门的权重矩阵, $\boldsymbol{W}_o$ 表示输出门的权重矩阵; $\boldsymbol{b}_f$, $\boldsymbol{b}_i$, $\boldsymbol{b}_c$, $\boldsymbol{b}_o$ 为偏差向量; f_t 表示最后一层神经元被遗忘的概率; i_t 和 p_t 表示当前需要保留的负载信息的比例; o_t 为输出门。

在模型训练过程中,利用 MSE 梯度随误差减小的特性以及为使损失函数具有可导性,使用 Huber 损失函数[22]来计算模型期望覆盖率 Y 与真实覆盖率 $f(x)$ 之间的差值,以获得更精确的最小值,使模型训练更准确。计算公式如式 10 所示:

$$L = \begin{cases} \dfrac{1}{2}(Y - f(x))^2, & |Y - f(x)| \leqslant \delta \\ \dfrac{1}{2}(Y - f(x))^2, & |Y - f(x)| > \delta \end{cases} \tag{10}$$

其中, δ 作为选择参数,由真实实验确认。

1.2 强化学习

强化学习与监督学习和无监督学习构成了机器学习领域。强化学习的过程通常可以由马尔可夫决策过程(Markov Decision Process, MDP)[23]进行描述,原理如图 3 所示。

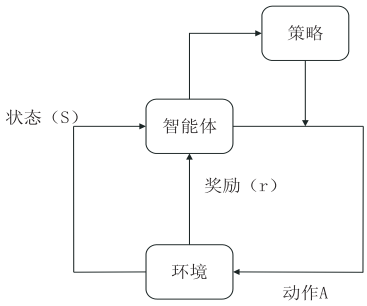


图 3 强化学习原理

智能体(agent)在环境中,通过不断的“试错”来学习,通过与环境的交互获得环境感知的状态(S),在价值策略(Π)的指导下进行决策并指导环境进行一系列的动作(A),而环境会将由这一系列动作触发的奖励反馈给智能体,智能体根据获得的奖励更新动作,向着最大化累计奖励的方向进行迭代,从而不断提高自身的决策能力。因此,强化学习的目标相当于学习一个最优策略来最大化奖励函数。如何设置合适的状态、策略和奖励函数至关重要。

1.3 RLGRU_LSTM 算法

1.3.1 基于 RLGRU_LSTM 算法的验证平台架构

通过研究大量国内外文献发现,目前在伪随机指令生成器的生成策略大体分为两种:动态的生成策略和静态的生成策略[24]。这两种生成策略各有优缺点。通过分析两者的优缺点,该文在研究伪随机指令生成技术时采用静态生成策略,通过随机组合的方式,覆盖

一些设计验证工程师无法预料的缺陷。整个验证平台主要由伪随机指令生成器、待测设计 DUT 和指令集模拟器(Instruction Set Processor,ISP)构成,每个模块由脚本控制启动。在伪随机指令生成模块,针对强化学习在样本训练阶段,训练样本的收敛速度满足不了现有技术的需求,样本学习效率低下。该文在现有 GRU_LSTM 算法[11-13]和强化学习[15-16]基础上,将 GRU_LSTM 和强化学习算法进行结合,构建新算法 RLGRU_LSTM,提高代码覆盖率,验证平台架构如图 4 所示。

其中伪随机指令生成器根据指令集格式、指令类型添加约束,生成大量伪随机测试向量,传递给 ISP 和 DUT。该文构建了能生成合法指令的伪随机约束模板。在构建指令模板过程中,根据每条指令所属基本类型的基本特点,将各个字段约束在合理范围内,并拼接成合法指令。其中约束包含如操作码、功能码范围、存储器可偏移地址大小、寄存器编址、立即数和符号数大小、指令中各个域的排列组合以及核心模式等。指令集模拟器用于判断 DUT 运行命令后的结果正确与否。ISP 和 DUT 在接收伪随机指令生成器输出的指令或者定向测试指令后,使用正则表达式等方法,通过脚本对两者输出结果进行对比并输出正确或错误的判断。

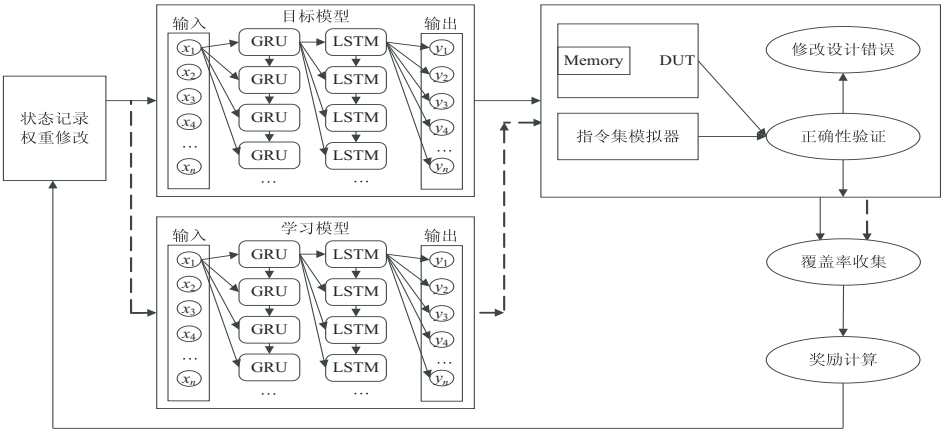


图 4　验证平台架构

由图 4 可知,在模型训练开始之前,根据强化学习的目标策略和历史覆盖率结合进行模型训练。在 RLGRU_LSTM 模型中,GRU_LSTM 作为 RL 智能体,将按照其遵循的策略发送下一步的操作,以验证平台作为整体环境,根据环境反馈的奖励,按照策略进行状态调整,GRU_LSTM 模型中神经网络如何学习修改功能码和操作码类别权重系数是策略的具体表现,智能体的奖励与当前训练模型覆盖率和上一时刻覆盖率相关。以指令包指纹编码序列作为样本,覆盖率指导损失函数,进行模型训练,向训练完成的 GRU_LSTM 网络输入伪随机指令包的指纹序列,得到该序列的最优覆盖率模型并作为目标模型和学习模型,在此基础上与强化学习相结合,将当前状态指令包输入目标模型,下一状态指令包输入学习模型,将学习模型得到的覆盖率与目标模型学习得到的覆盖率差值作为奖励,根据策略调节学习模型的权重参数,当调节次数到达某一固定值如 10 次后,将学习模型的权重参数赋值给目标模型,完成对 RLGRU_LSTM 模型的训练。

1.3.2　*动作与奖励*

RLGRU_LSTM 在模型训练过程中,可以看作是快速在一堆指令中寻找出可以达到最大覆盖率的指令集合,通过奖励函数,学习 GRU_LSTM 算法模型对 opcode 和 func 等类别的权重分配。因此在强化学习过程中对奖励函数的设计十分重要,奖励函数的好坏决定了智能体是否有效的学习环境,能否高效地完成模型训练。在指令生成模型中,芯片验证代码覆盖率能否快速达到最高值,取决于生成指令是否能够在每次模型训练过程中有目标地按照奖励函数生成。因此,若想高效找到一组指令集达到最高覆盖率,需要在当前训练覆盖率 f 大于上一时刻覆盖率 best(f) 时给予正奖励;当前覆盖率小于上一时刻覆盖率时给予负奖励。t 时刻,最佳覆盖率奖励如式 11 所示:

$$R(t)=\begin{cases}+1.5, & f>\text{best}(f)\\ -0.5, & f=\text{best}(f)\\ -1.0, & f<\text{best}(f)\end{cases} \tag{11}$$

1.4　算法训练过程

基于 GRU_LSTM 及强化学习算法的伪随机指令生成器的流程和算法流程如图 5 所示。

(1)加载验证环境,使用脚本调用所需编译、仿真软件,配置仿真、编译相关参数,初始 DUT 所需条件、寄存器、只读存储器 rom 等,创建初始约束条件;

(2)对指令以及权重进行霍夫曼编码;

(3)确定 GRU_LSTM 网络的神经元个数,Huber 的选择参数 δ,梯度优化函数,对 GRU_LSTM 模型进行训练,得到最优 GRU_LSTM 模型;

(4)根据图 4,将当前状态值和下一时刻状态输入到目标 GRU_LSTM 模型和学习模型,并返回奖励值,计算 Huber 损失函数,根据 Adam 优化器自动更新下

一次状态,以及迭代次数 T,得到权重,完成 RLGRU_LSTM 的训练。

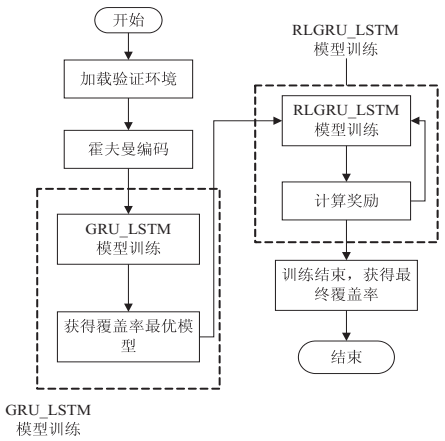


图 5 基于 GRU_LSTM 及强化学习算法的伪随机指令生成器的流程

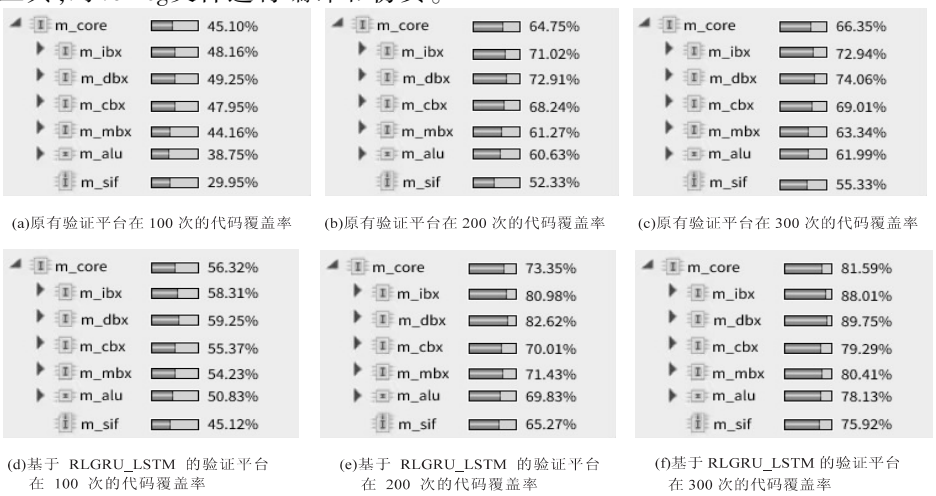
2 实际应用及性能分析

2.1 实验环境

验证平台在 Python3.6 环境下运行,DUT 为基于 RISC-V 指令集架构开发的一款面向低功耗嵌入式领域的处理器核心,在微结构上实现单译码、单发射、顺序执行、静态预测的五级流水线。采用 cadence 的 xcelium 的仿真工具,对 verilog 文件进行编译和仿真。采用 IMC(Integrate Metrics Cener)收集、合并覆盖率并输出覆盖率日志。整个验证平台通过 Python 脚本整合,分析终端操作命令,调用对应工具、软件、脚本、算法和实验对象代码等,根据各软件运行日志,输出仿真、覆盖率、对比等结果。实验中构建的 GRU_LSTM 的两层神经网络均采用 128 个神经元,Huber 损失函数的超参数 σ 根据实际实验结果设置为 0.98,使用 Adam 优化器优化权重 w_i,完成对所提 RLGRU_LSTM 算法的训练。该文以伪随机指令的覆盖率与应用新算法训练模型得到的覆盖率,以及其他文献所提方法得到的覆盖率做对比,验证算法的有效性。

2.2 实验结果对比

以代码覆盖率作为 RLGRU_LSTM 模型的损失函数参数,指导生成伪随机指令的一个最直接目的就是提高代码覆盖率,减少验证收敛时间,快速覆盖验证盲点。图 6 给出了基于原伪随机指令生成器的验证平台和基于 RLGRU_LSTM 的伪随机指令生成器的验证平台分别在迭代 100,200,300 次收集到的代码覆盖率。图 7 给出随着次数的增加,基于原伪随机指令生成器的验证平台和基于 RLGRU_LSTM 的伪随机指令生成器的验证平台代码覆盖率的收敛情况。表 1 给出所构建模型与其他文献所提模型在指令生成应用上,与不采用任何提高覆盖率方式的原覆盖结果的提高对比。



(a)原有验证平台在 100 次的代码覆盖率 (b)原有验证平台在 200 次的代码覆盖率 (c)原有验证平台在 300 次的代码覆盖率

(d)基于 RLGRU_LSTM 的验证平台在 100 次的代码覆盖率 (e)基于 RLGRU_LSTM 的验证平台在 200 次的代码覆盖率 (f)基于 RLGRU_LSTM 的验证平台在 300 次的代码覆盖率

图 6 原有验证平台和基于 RLGRU_LSTM 的验证平台代码覆盖率对比

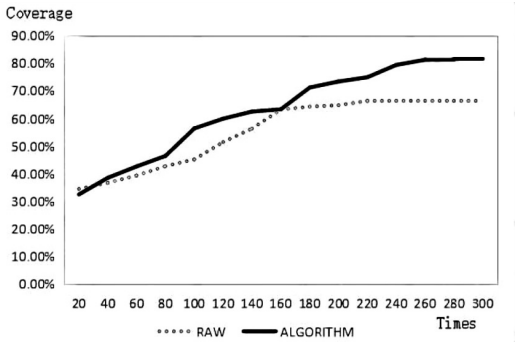


图 7 原有验证平台和基于 RLGRU_LSTM 的验证平台代码覆盖率的收敛情况对比

表 1 RLGRU_LSTM 与其他模型覆盖率提高对比 %

模型	原覆盖率	现覆盖率	提高百分比
GA[2]	66.35	75.54	12.17
SVM[7]	66.35	70.95	6.49
RNN[9]	66.35	72.55	8.55
GRU+2LSTM[11-13]	66.35	76.79	13.61
文中	66.35	81.59	18.69

实验结果清楚表明,对比其他文献所提方法,文中模型能明显提高覆盖率,并且随着次数的增加,基于原伪随机指令生成器的验证平台的代码覆盖率已趋于不

变,覆盖率值约 62%,而基于 RLGRU_LSTM 的伪随机指令生成器的验证平台的代码覆盖率依旧在缓慢增加至约 82%,且基于 RLGRU_LSTM 的伪随机指令生成器的验证平台会比基于原伪随机指令生成器的验证平台较早地到达目标覆盖率值。但是由于有些覆盖条件是在约束之外、部分分支条件无法进入等原因,覆盖率最后无法达到 100%。

3　结束语

为解决实际情况中如何提高伪随机指令生成器代码覆盖率的问题,结合 RLGRU_LSTM,以及霍夫曼编码指令包方式,针对该文搭建的验证平台,在约束指令条目情况下,将代码覆盖率提高至 82% 左右,高于原伪随机指令生成器约 19%,且对比文献[2,7,9,11-13],RLGRU_LSTM 模型在指令生成过程中代码覆盖率增幅明显优于文献所提模型。同时相比原伪随机指令生成器,覆盖率提前约 20 s 达到某一目标值。但在实际应用过程中仍存在一个问题,覆盖率包含功能覆盖率和代码覆盖率,该文只考虑了代码覆盖,然而 100% 的代码覆盖率并不一定意味着该设计已经得到了完全的验证。代码的覆盖范围并没有完全定义验证的完整性和质量,功能覆盖率需要状态机在不同状态间相互切换完成某一流程或严苛到特定节拍的触发条件,如何将功能点覆盖率考虑进来是有待考虑的一点。

参考文献:

[1] 聂　鹏,耿　技,秦志光. 软件测试用例自动生成算法综述[J]. 计算机应用研究,2012,29(2):401-405.

[2] SMITH J E,BARTLEY M,FOGARTY T C. Microprocessor design verification by two-phase evolution of variable length tests[C]//Proceedings of the 1997 IEEE international conference on evolutionary computation (ICEC '97). Indianapolis: IEEE,1997:453-458.

[3] 罗　春,杨　军,凌　明. 基于遗传算法和覆盖率驱动的功能验证向量自动生成算法[J]. 应用科学学报,2005,23(4):375-379.

[4] 宋　倩. 基于遗传算法的测试用例生成技术[J]. 计算机系统应用,2014,23(11):264-267.

[5] 黄陈辉,吴海涛,阮江涛,等. 基于混沌遗传算法的测试用例自动生成研究[J]. 计算机与数字工程,2021,49(1):31-35.

[6] WANG Fanchao,ZHU Hanbin. Accelerating coverage directed test generation for functional verification: a neural network-based framework[C]//GLSVLSI'18: Proceedings of the 2018 on great lakes symposium on VLSI. California: University of Southern California,2018:207-212.

[7] GUO Qi,CHEN Tianshi,SHEN Haihua,et al. On-the-fly reduction of stimuli for functional verification[C]//Proceedings of the 2010 19th IEEE Asian test symposium. Shanghai: IEEE,2012:448-454.

[8] GOGRI S, HU Jiang, TYAGI A, et al. Machine learning-guided stimulus generation for functional verification[J]. Computer Science,2019(4):1-10.

[9] FAJCIK M,SMRZ P,ZACHARIASOVA M. Automation of processor verification using recurrent neural networks[J]. Computer Science,2017(6):15-20.

[10] CHO K,MERRIENBOER B,GULCEHRE C,et al. Learning pharse representations using RNN Encoder_Decoder for statistical machine translation[J]. Computer Science, 2014, 45(6):1-7.

[11] 王　磊,王永华,何一汕,等. 基于 GRU 和 LSTM 组合模型的车联网信道分配方法[J]. 电讯技术,2023,9(2):25-36.

[12] 杨耀红,刘德福,韩兴忠,等. 基于 LSTM-GRU 模型的 TBM 掘进参数时序预测研究[J]. 水力发电,2023,49(2):78-84.

[13] UWAISU A,YAHAYA A S,KAMAL S M,et al. A hybird deep stacked LSTM and GRU for water price prediction[C]//2020 2nd international conference on computer and information sciences(ICCIS). Sakaka: IEEE,2020:1-6.

[14] 李茹杨,彭慧民,李仁刚,等. 强化学习算法与应用综述[J]. 计算机系统应用,2020,29(12):13-15.

[15] 贾政轩,林廷宇,肖莹莹,等. 基于强化学习的最优控制指令模仿生成方法[J]. 系统仿真学报,2023,35(11):2410-2418.

[16] HUGHES W,SRINIVASAN S. Optimizing design verification using machine learning: doing better than random[J]. arXiv: 1909.13168,2019.

[17] 刘　斌. 芯片验证漫游指南:从系统理论到 UVM 的验证全视界[M]. 北京:电子工业出版社,2018.

[18] YANG S,YU X,ZHOU Y. LSTM and GRU neural network performance comparison study: taking yelp review dataset as an example[C]//2020 international workshop on electronic communication and artificial intelligence (IWECAI). Shanghai: IEEE,2020:98-101.

[19] CHUNG J,GULCEHRE C,CHO K H,et al. Empirical evaluation of gated recurrent neural networks on sequence modeling[J]. arXiv:1412.3555,2014.

[20] 杨　可. 一种具有伪随机存取能力的压缩编码技术[D]. 合肥:中国科学技术大学,2019.

[21] 贺小伟,徐靖杰,王　宾,等. 基于 GRU-LSTM 组合模型的云计算资源负载预测研究[J]. 计算机工程,2022,48(5):11-17.

[22] CAI Xiong,XUE Liugen,WANG Zhaoliang. Robust estimation with modified Huber's function for functional linear models[J]. Statistics,2020,54(6):1276-1286.

[23] SUTTON R S,BARTO A G. Reinforcement learning: an introduction[M]. 2nd ed. Cambridge: MIT Press,2018:1-526.

[24] 刘　婧,王天成,王　健,等. 基于指令模板的通用处理器约束伪随机指令生成方法[J]. 计算机工程,2015,41(10):309-313.