文章编号:1671-4598(2022)07-0026-09 DOI:10.16526/j.cnki.11-4762/tp.2022.07.004 中图分类号:TP311 文献标识码:A

一种面向缺陷检测过程的警报自动确认方法

孔焦龙，金大海，宫云战
(北京邮电大学 网络与交换技术国家重点实验室，北京 100876)

摘要：静态分析工具能够一定程度上帮助开发者检测代码中的重要错误；然而，可扩展性和不可判定性的存在会影响这些工具的准确率，导致它们无法被用于更广泛的实践中；最近，研究人员开始利用人工智能的技术来提高这些工具的可使用性，通过将正确和错误的警报自动分类，以节省在软件开发过程中人工确认警报所需要的人力和时间的花费；传统的方法主要通过使用手工提取的特征来表示有缺陷的代码片段，难以抓住它们深层次的语义信息；为了克服传统方法的限制，设计并提出了一种创新的特征提取方法，通过收集并提取缺陷模式状态机实例状态转换过程中相关指令集所包含的细粒度的语法、语义信息，并将有效的深度学习框架与之相结合，从而实现跨工程的警报自动确认；在5个开源工程的警报数据集上实验，分别与基于传统度量元的自动确认方法比较，AUC指标提升幅度在1.83%～31.81%之间，表明该方法能够有效提升跨工程警报自动确认的表现。

关键词：静态分析；软件缺陷确认；缺陷模式状态机；词嵌入；深度神经网络

An Automatic Alarm Identification Method Oriented to Defect Detection Process

KONG Jiaolong, JIN Dahai, GONG Yunzhan
(State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract: Static analysis (SA) tools can aid the developers detect the critical errors in software to some degree. However, challenges such as scalability and undecidability are likely to have impact on their precision and performances, preventing these tools from being widely adopted in practice. Recently, researchers begin to utilize the artificial intelligence techniques to improve the usability of these tools by automatically classifying the alarms of false and positive, the manual identification of which is laborious and time-consuming in software development processes. Traditional approaches mainly focus on using hand-engineered features to represent the defective code snippets, it is hard to extract the deep semantic information of reported alarms. In order to overcome the limitations of traditional approaches, a novel feature extraction approach is designed and proposed. By collecting and capturing the fine-grained semantic and syntactic information, which is included in the instructions related to the state-transforming processes of instances of the fault pattern state machine, and combining them with an effective deep learning framework, the cross-project defect automatic identification can be achieved. The experiment is based on the alarm dataset of five open-source projects. Comparing with the traditional metrics-based method, the indicator AUC is increased by between 1.83%～31.81%. The experimental results show that the proposed method is effective and can yield significant improvement on the cross-project defect identification.

Keywords: static analysis; software defect identification; fault pattern state machine; word embedding; deep neural network

0 引言

软件出现在现代生活的方方面面，从社交媒体到基础设施。由于软件正在变得越来越复杂，其不可避免的存在重大的缺陷，这些缺陷可能会给人类社会造成重大的损失[1]。然而，由于有限的预算以及紧张的日程安排，仅仅通过软件测试来找出缺陷是不切实际的，因为这通常需要大量的时间来运行各种各样的测试用例。静态分析（SA，static analysis）能很好的弥补这一点，不需要运行程序也能检验程序源码，并且可以考虑到所有可能的程序执行。但是，通过静态分析在程序中找到的相关缺陷通常是不可判定的，这一点限制了静态分析工具的实用性[2]。因此静态分析工具报告出的有关代码缺陷的警报信息中会存在一部分错误，本文称之为误报，人工确认潜在缺陷是一项费时费力的工作[3]。

为了减轻人工确认的负担，大量能使这一过程自动化的方法已经被提出[4-5]。其中有通过设计并提取一系列的手工特征来训练有效的分类模型，这些被提取的特征主要和

收稿日期:2022-01-05； **修回日期**:2022-02-18。
基金项目:国家自然科学基金(U1736110)。
作者简介:孔焦龙(1996-)，男，湖北荆门人，硕士，主要从事软件测试、缺陷检测方向的研究。
金大海(1974-)，男，辽宁沈阳人，副教授，硕导，主要从事软件测试、缺陷检测方向的研究。
宫云战(1962-)，男，山东威海人，教授，博导，主要从事软件测试、软件可靠性方向的研究。
引用格式:孔焦龙，金大海，宫云战.一种面向缺陷检测过程的警报自动确认方法[J].计算机测量与控制,2022,30(7):26-34.

源代码的统计特征有关，包括 McCabe metrics[6]，Halstead metrics[7]，和 CK metrics[8]，然而当前文献表明这些特征在表示源代码深层语义信息的时候缺乏准确性。另外，之前大部分方法主要针对工程内的任务，即训练集和测试集来自同一个工程。跨工程确认的一个主要的问题在于不同工程之间数据分布的差异性。因此，直接把基于源工程特征构建起来的模型应用到目标工程，大部分情况下无法达到令人满意的效果。

最近，越来越多静态分析和人工智能结合的研究成为了解决上述问题的重要手段。明确地说，基于深度学习的方法已被广泛应用于特征的自动生成[9-13]。它们的主要思路是将从源代码对应的语法树（AST，abstract syntax tree）上获取的符号序列表示为符合深度神经网络的输入形式，再利用神经网络学到的特征去训练更准确的分类模型。相关实验结果表明基于深度学习的方法能够更好的获取软件缺陷相关的语义特征，因此它们的表现会胜过仅使用手工特征的传统方法。

本文提出一种面向缺陷检测过程的自动确认方法，即基于缺陷模式状态机[14]的检测过程。通过搜集在检测过程中与缺陷模式状态机实例状态转换相关的指令集，提取指令集中所包含的细粒度的语法、语义信息并转换为结构化的序列，将这些序列集合映射到高维向量空间后作为神经网络的输入，训练出能有效自动确认警报的模型。该方法一定程度上可以抵消不同工程之间的特征分布差异，本文将其命名为 State2Vec。

1　相关工作

1.1　缺陷模式状态机

缺陷模式描述了程序的某种属性，如果违反该属性则生成一条缺陷。例如：若某处申请的资源在使用完后未释放，则造成资源泄露缺陷（RL，resource leak）；若对空指针解引用操作，会导致空指针引用缺陷（NPD，null pointer dereference）。

定义 1：每一种缺陷模式都有其对应的缺陷模式状态机，其本质上为有限状态机（FSM，finite state machine），包括状态集合 D、状态转移集合 T，及转移条件集合 $Conditions$，其中 $D_{other}=\bigcup\{\$error, \$start\}$，$D\times Conditions\xrightarrow{T}D$。$\$error$ 和 $\$start$ 分别表示错误状态和起始状态，D_{other} 表示其他状态的集合。缺陷模式状态机也可以用有向图直观地表示。例如：RL 和 NPD 模式可用状态机描述如图 1 所示。

缺陷模式状态机包含多种缺陷模式，是对所有缺陷的一种统称。若要对函数内部某些具体缺陷模式检测，需要根据其各自的状态机实例创建条件，创建相应实例。例如根据函数中每一处资源分配创建一个 RL 状态机实例；对每一处可能被解引用的变量创建一个 NPD 状态机实例。

1.2　基于缺陷模式状态机的测试方法

定义 2：检查点（IP，inspection point）。设 P 是待测程序，将缺陷模式 S 分成 N 类，类，$S=\{S_1, S_2, \cdots, S_n\}$，每类分成 K 种，$Si=\{S_{i1}, S_{i2}, \cdots, S_{iK}\}$，从 P 中计算出和 S 相匹配的检查点的集合 $IP=\{IP_1, IP_2, \cdots, IP_m\}$。

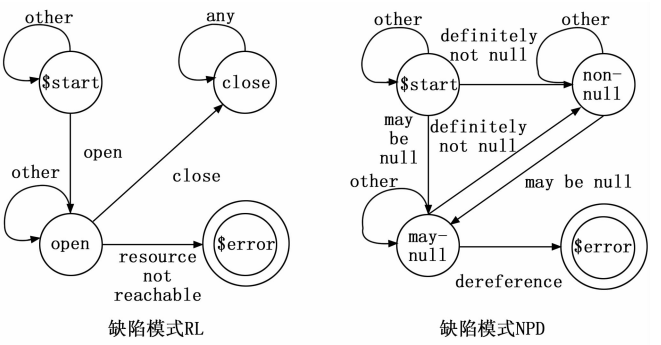

图 1　RL 和 NPD 缺陷模式状态机

定义 3：警报点（AP，alarm point）。在 IP 集合中，每一条记录由一个三元组 $\{S, P, J\}$ 表示：S 为静态分析报告中一条记录对应的缺陷模式。P 为一个六元组，用于描述 IP 前 5 个分别表示工程名，变量名、变量出现的位置、文件名、行号、IP 所在方法名，若两个及以上相同变量出现在同一行代码中，$Index$ 表示变量出现的索引。J 表示对此 IP 自动确认后的结果，未进行缺陷确认前，该值为 NULL，缺陷确认后，真实缺陷（True），误报（False Alarm）。

现对选定程序 P 缺陷检测，首先选定要检测的缺陷模式种类，根据各自条件创建相应的状态机实例，新创建的实例初始状态为 $\$start$，且实例之间互相独立。接着结合数据流分析，采用对控制流图进行迭代的方法，对当前节点各个实例可能的状态集合进行迭代。数据流分析中，数据流信息通过建立和解方程来收集方程联系程序不同点的信息。

基于缺陷模式状态机的检测过程如算法 1 所示，算法描述如下。

算法 1：

输入：程序控制流图和缺陷模式描述状态机

输出：每个控制流节点的 inStateSet[n]和 outStateSet[n]

```
1:begin
2:for 每个控制流节点 n do
3:inStateSet[n] ← Ø/*初始化*/
4:outStateSet[n] ← Ø
5:outStateSet[Entry] ← D{$start(Var[])}/*入口节点的初始状态集合*/
6:statechange ← true
7:while statechange do begin
8:statechange ← false
9:for 除 Entry 的每个节点 n do begin
10:nStateSet[n] ← U_{p∈pred[n]} outStateSet[p]
11:oldout ← outStateSet[n]
12:outStateSet[n] ← genState[n] U(inStateSet[n]−kill[n])
13:if outStateSet[n] ≠ oldout
14:hen statechange ← true
```

15:end
16:end
end

算法 1 中，输入被测程序 P 的控制流图以及状态机描述文件，得到控制流各节点输入及输出的实例状态集合。其中 inStateSet [n] 表示进入节点 n 之前的所有实例状态集合，outStateSet [n] 表示通过节点 n 之后所有实例状态集合。genState [n] 表示在节点 n 处新产生得到的所有实例状态集合，kill [n] 表示在节点 n 处被注销的所有实例的状态集合。pred [n] 表示节点 n 的入度节点集合，即 n 的前驱节点。

2 面向缺陷检测过程的警报自动确认

本文基于缺陷模式状态机的检测机理提取相关指令集，并设计了一种新的细粒度的特征提取方法，结合程序语言处理的相关技术，提升在跨工程场景下警报确认的效果。本文将该自动确认方法命名为 State2Vec，共计三部分。

State2Vec 可分为以下三部分：1）基于缺陷模式状态机的检测机理提取相关指令集，通过提取状态机实例沿着控制流计算过程中状态发生改变的节点集合，得到相应的代码段集合，称之为相关指令集。2）设计并利用一种新的细粒度的特征提取方法，提取相关指令集所包含的细粒度的语法、语义信息。3）利用词嵌入技术，将结构化的字符串序列映射到高维向量空间，作为神经网络的输入，并将合适的深度学习模型与之相结合，训练得到可以学习缺陷语义特征的深度神经网络。该方法整体流程如图 2 所示。

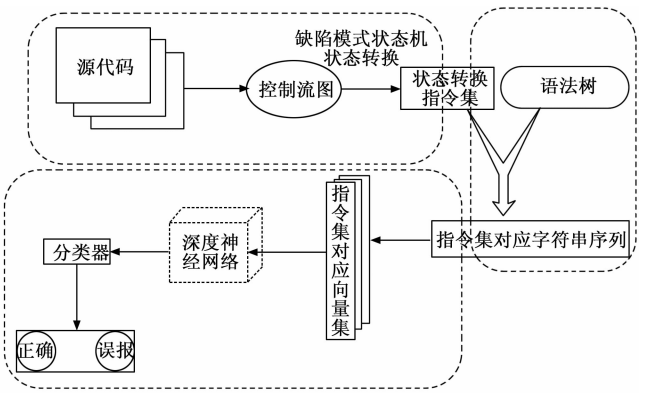

图 2 State2Vec 工作流程

2.1 基于缺陷模式状态机的指令集提取方法

目前大部分缺陷语义相关的指令集提取技术都存在以下问题——关于程序特征的指令集序列通过人工建模的方式提取，这类方法的缺陷在于人工提取的指令集较为粗糙，无法准确获取代码深层次的语义信息。

本文采用面向缺陷检测过程的指令集提取技术——将空指针引用、资源泄露等具体的程序缺陷抽象为一种缺陷模型，通过缺陷模式状态机的创建条件及程序语义确定是否创建该缺陷的状态机实例，并通过在为源代码构建的分析底层框架（控制流图等）上迭代状态引发状态间的转换检测出程序中的潜在缺陷，在此缺陷检测过程中，收集并提取导致状态发生改变的控制流节点集合，这些节点对应的源代码片段即为相关指令集。基于该原理的指令集提取过程如算法 2 所示。

算法 2：
输入　程序控制流图和缺陷模式描述状态机
输出　缺陷模式状态机实例状态转换对应的控制流节点集合
1:for n in G(N, E, Entry, Exit) do
2:inStateSet[n] ← Ø
3:outStateSet[n] ← Ø /* 初始化 */
4:end
5:changeList ← List() /* 声明 changeList 用来存放状态改变的控制流节点 */
6:outStateSet[Entry]←$D\{\$start(\mathrm{Var}[\,])\}$/* 入口节点的初始状态集合 */
7:statechange ← true
8:while statechange do
9:statechange ← false
10:for 除 Entry 的每个节点 n do
11:inStateSet[n] ←
12:$\mathrm{U}_{p\in pred[n]}$ outStateSet[p]13:oldout ← outStateSet[n]outStateSet[n] ← gen[n]
14:U (inStateSet[n]−kill[n])if outStateSet[n] ≠oldout then /* 满足该条件时状态发生改变 */
15:statechange ← true
16:changeList.add(n) /* 将状态改变的控制流节点放入 list */
17:end
18: end
19:end
20: return changeList

以 bad 函数的源码为例，相关指令集提取的具体细节如图 3 所示。bad 函数共有 5 个控制流节点，在节点 Entry 处，状态机实例处于 start 状态；在节点 n1 处，结构体指针 twoInts 初始化为 NULL，此时发生状态转换 START->MAYNULL；在节点 n2 处，twoInts 在位操作符 & 后发生解引用，此时状态转换 MAYNULL->ERROR。因此，bad 函数中与 NPD 警报相关的指令集为控制流节点 n1、n2 对应的代码段。

2.2 基于指令集的细粒度特征提取方法

源代码语句中细粒度结构信息在之前的方法中往往会被忽视。为了做一个简单的解释，本文引入两个 C 语言函数为例，如图 4 所示。两个函数都有类型声明 type declaration 语句，if 语句，以及函数调用 function invocation 语句。两者唯一的区别在于 if 语句中操作符不同，使用单一的 & 操作符会使表达式的两侧均被计算，从而导致 NPD 缺陷，正如 bad 函数展示的那样。但是在 good 函数中，使用 && 操作符可以避免 NPD 缺陷，当表达式左侧不满足条件时，右侧将不会被计算。现有的基于深度学习的方法大多只是根据上下文信息将代码片段映射为字符串序列，这样无法

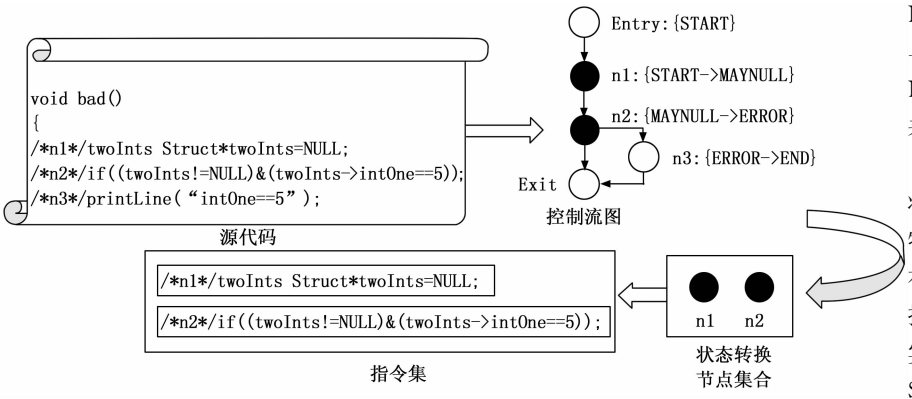

图 3　bad 函数相关指令集提取

```
1 void bad(
2 {
3   twoIntsStruct *twoInts = NULL;
4   if((twoInts != NULL) & (twoInts->intone == 5))  //警报1
5     printLine(“intone == 5”);
6 }

7 void good()
8 {
9   twoIntsStruct *twoInts = NULL;
10  if((twoInts != NULL) && (twoInts->intone == 5))  //警报2
11    printLine(“intone == 5”);
```

图 4　两个 C 语言函数

区别条件谓词表达式中细微的区别。因此两个函数的字符串序列表示也是一样的，可以用序列［TYPE _ DECLARATION，IF _ STATEMENT，FUNCTION _ INVOCATION］表示，但是它们的确认结果是完全相反的。

本文提出一种新的细粒度的特征提取方法，可以准确地反映条件谓词表达式中各操作符与操作数之间的细微区别。该方法从以下 5 种类型的语法树节点中提取特征：1）Assignment Expression Node；2）Declaration Node；3）Function Invocation Node；4）SelectionStatement Node（例如 if、switch 语句）；5）IterationStatement Node（例如 for、while 语句）；6）JumpStatement Node（例如 return、break 语句）。通过从这些节点及其子节点中提取操作符及操作数，再按照表达式从左到右的顺序将他们排序，最后以该顺序映射为定义好的字符串序列。

图 5 列出了本文中使用的所有选定节点类型和原子指令，使用每个节点的名称类型和原子指令作为标识符。考虑到函数和变量的名称通常是特定于工程的，不同工程中具有相同名称的函数可能实现不同的功能，为了保证跨工程确认的可行性，本文使用两种类型名称，表示库函数和用户定义函数分别标记函数调用节点而不是使用特定名称，字符串 LIBRARY _ FUNCTION _ INVOCATION 和 USER _ DEFINE _ FUNCTION _ INVOCATION 分别表示库函数和用户自定义函数调用。对于 IP（Inspection Point）变量——在检查点与缺陷直接相关的变量，本文用 VAR _ IP 表示，其他变量用 VAR _ OTHER 表示。

以 bad 函数警报相关的指令集为例，将其映射为上下文相关的字符串序列，特征提取的详细过程及结果如图 6 所示。在该函数中，有一条 NPD 警报生成，在控制流节点 n1 和 n2 处缺陷模式状态机实例发生状态转换，n1 处状态转换为 START->NULL，n2 处状态转换为 NULL->ERROR。

节点 n1 对应代码行是声明语句，对应语法树节点 Declaration Node 类型，twoIntsStruct 是结构体类型，twoInts 是一个结构体指针，twoInts 是 IP 变量，且初始化为 NULL，因此节点 n1 相关指令可以映射为字符串序列｛DECLARATION，STRUCT _ TYPE，POINTER，VAR _ IP，NULL｝。

节点 n2 对应代码行是选择语句，对应语法树节点 SelectionStatement Node 类型，twoInts->intOne 是其他变量，“!=”和“≠”是比较操作符，“&”是位操作符，“5”是常数，因此节点 n2 相关指令可以映射为字符串序列｛IF _ SELECTION，VAR _ IP，NOT _ EQUAL，NULL，INCLUSIVE _ AND，VAR _ OTHER，EQUAL，CONSTANT｝。

2.3　字符串编码以及深度神经网络的构建和训练

2.3.1　字符串向量编码

因为字符串序列无法直接作为神经网络的输入，需要通

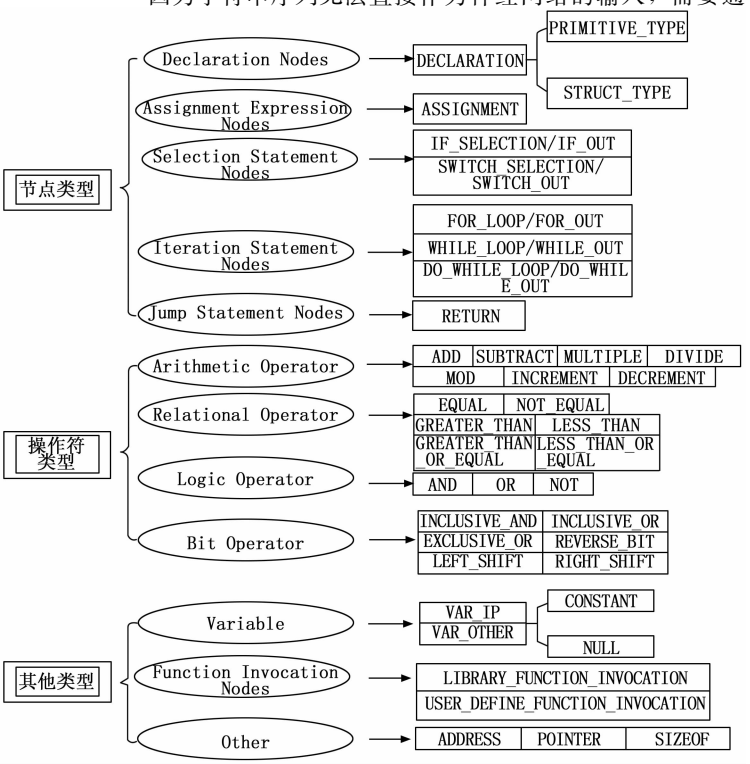

图 5　各节点类型和原子指令对应字符

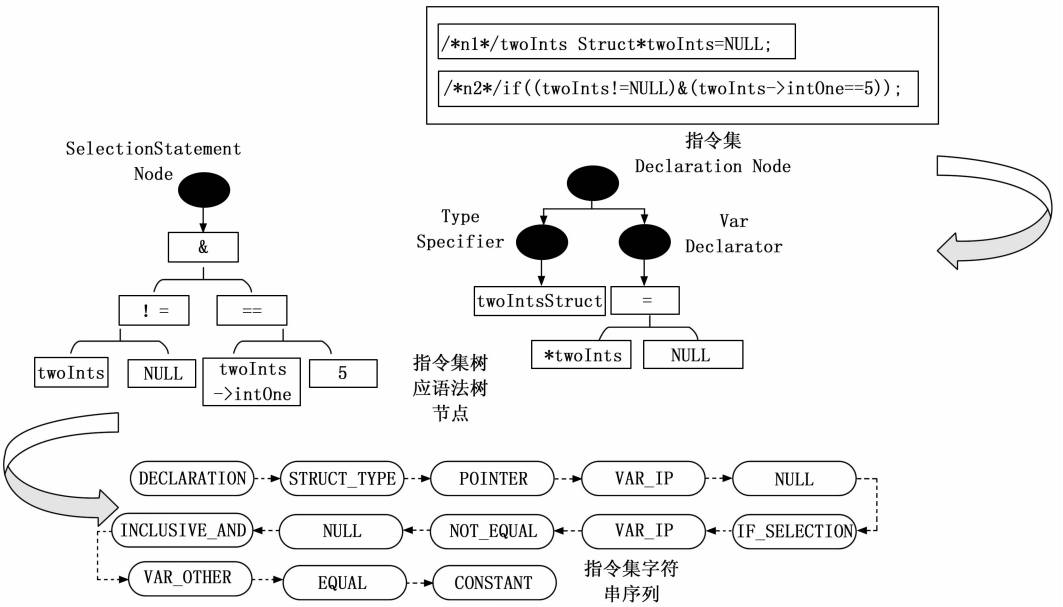

图 6 相关指令集特征提取

过词嵌入技术将其转换为数值向量。本文首先统计字符串的总类，将每一种字符串都对应一个正整数索引，从 1 开始。由于低维的整数无法有效描述字符串序列中的结构及上下文信息，本文将其映射到高维的向量空间，并且有着相似上下文的字符串在高维向量空间中的分布会更加接近。

Word2vec 中的 Skip-gram[15] 模型可以通过预测中心词周围的背景词来训练词向量，因为本文提取的字符串序列与指令集中操作符和操作数出现顺序一致，所以字符串之间存在上下文关系。本文采用 skip-gram 模型并结合负采样（negative sampling）的统计方式来训练词向量，目标函数如式（1）所示。其中 v_{wI} 表示当前中心词对应的向量表示，v'_{wo} 表示真实背景词对应的向量表示，$v'_{wi}{}^{T}$ 表示通过负采样得到的背景词的向量表示，k 表示负采样样本的大小。

$$\log\sigma(v'_{wo}{}^{T}v_{wI})+\sum_{i=1}^{k}E_{w_i\sim P_n(w)}[\log\sigma(-v'_{wi}{}^{T}v_{wI})] \tag{1}$$

因为 Word2vec 本质上也是训练神经网络，本文采用随机梯度下降（SGD，stochastic gradient descent）进行优化。最终每一个字符串都被表示成了一个 n 维向量 R^n，n 的大小可以自己设置，本文将 n 设置为 150，负采样样本大小 k 设置为 100。

由于字符串序列的长度存在差异，而神经网络的输入要求每一个样本的长度相同，本文需要对长度较短的序列进行填充（padding），将多个 padding 字符串填充在较短的序列末尾，使其长度和样本中最长的字符串序列相同，padding 字符串用 n 维零向量表示。

2.3.2 训练模型及警报自动确认

循环神经网络（RNN，recurrent neural networks）是一个在时间上传递的神经网络，网络的深度就是时间的长度。该神经网络是专门用来处理时间序列问题的，能够提取时间序列的信息。如果是前向神经网络，每一层的神经元信号只能够向下一层传播，样本的处理在时刻上是独立的。对于循环神经网络而言，神经元在这个时刻的输出可以直接影响下一个时间点的输入，该神经网络能够处理时间序列方面的问题。因此，RNN 对具有序列特性的数据非常有效，它能挖掘时序信息以及语义信息，利用这种能力，使深度学习模型在解决语音识别、语言模型、机器翻译以及时序分析等 NLP 领域的问题有所突破。同样，RNN 网络也可以应用到程序语言处理（PLP，program language process）问题中。

因为模型的输入是存在上下文依赖关系的向量序列，本文选择 RNN 作为基本模型，通过调整 RNN 的种类，具体表现为长短期记忆网络（LSTM，long-short term memory）与门控循环单元网络（GRU，gated recurrent unit）。RNN 的层数为两层，RNN 的方向为双向，并与逻辑回归（LR，logistic regression）分类器结合。首先把 5 个开源工程中所有的警报人工确认好，按照 State2Vec 的前两步映射为对应的字符串序列，接着按照第三步将每种出现的字符串训练为词向量，将每一条字符串序列转换为向量集合，并在它们末尾以零向量填充，数据集构建完成。开始进行工程内的自动确认实验，将 5 个工程的数据集依次分为训练集和测试集，再分别用 5 个工程的训练集训练，调整 RNN 的类型搭配，分别训练出 LSTM 和 GRU 类型的 RNN 模型，再分析不同的模型在不同工程测试集内的自动确认情况。挑选出效果最好的模型，进行跨工程的实验。因为在同一工程内，训练集和测试集样本的分布可以认为在同一向量空间，本文提出的细粒度的特征提取方法也是尽可能缩小不同工程间样本在向量空间中分布的差异性。因此，在使用此特征提取方法的前提下，工程内确认效果更好的模型理论上在跨工程确认场景中也会取得更好的效果。跨

工程实验的训练集是某 4 个工程全部样本的集合，测试集是另一个工程的全部样本，实验过程中依次将每个工程的全部样本作为测试集。

3　实验

3.1　实验构建

针对模型的构建和训练，本文使用 Pytorch 框架进行实验，版本为 1.7.1，python 版本为 3.8.5，实验在服务器上运行，服务器环境如下：Ubuntu 18.04.5 LTS，Intel（R）Xeon（R）CPU E5-2620 v4 @ 2.10 GHz，160 GB RAM，GPU TITAN Xp（12 GB memory）。

3.1.1　数据准备

本文用 5 个开源 C 工程（如表 1 所示）中人工标签好的警报作为数据集，从近期的研究方法[16-18]中挑选了 4 个工程，它们的警报由 SA 工具测出，已通过人工确认打上标签。最后，本文制作了基于 android-4.0 C 语言源码的漏洞数据库，通过 SA 工具 DTS[19]对其 C 源码分析得到警报，再人工确认并打上标签。这些工程代码行数从 1 991 到 325 164 1 不等，警报数量从 74 到 5 235 不等，警报中阳性比例（确认结果为 True）最小占 50.0%，最大占 87.3%。

表 1　5 个开源 C 工程基本信息

工程名	代码行数/行	警报数量/个	警报中阳性比例(%)
uucp-1.07	52 595	545	71.0
spell-1.0	1 991	74	54.4
barcode-0.98	138 102	129	87.3
sphinxbase-0.3	22 517	240	50.0
android-4.0	325 164 1	5 235	61.8

3.1.2　实验表现度量元

过去的研究[20-21]表明，当数据集中正负样本比例不平衡时，用精准率（precision）和召回率（recall）来衡量实验的表现是存在问题的，因为它们对正负样本的分布比较敏感。同时，LR 阈值（threshold）的设置也会对它们产生影响。因此，本文采用曲线下面积（AUC，area under curve）来度量模型自动确认的表现。

在二分类实验中，本文一般关注 4 个指标：1）真阳性（TP，true positive）：数据集中本身既是缺陷也被模型预测为缺陷的警报的数量；2）假阳性（FP，false positive）：数据集中本身不是缺陷却被模型预测为缺陷的警报的数量；3）真阴性（TN，true negative）：数据集中本身既不是缺陷也被模型预测为非缺陷的警报的数量；4）假阴性（FN，false negative）：数据集中本身是缺陷却被模型预测为非缺陷的警报的数量。AUC 表示受试者工作特征（ROC，receiver operating characteristic）曲线下的面积，ROC 曲线是基于样本的真实类别和预测概率来画的。具体来说，ROC 曲线的 x 轴是假阳性率（FPR，false positive rate），y 轴是真阳性率（TPR，true positive rate），FPR 和 TPR 的计算见示（2）和式（3）。

$$FPR = FP/(FP + TN) \tag{2}$$

$$TPR = TP/(TP + FN) \tag{3}$$

AUC 的值在区间［0，1］内，值越接近 1，说明模型的表现越好。若接近 0.5，说明分类器近似于随机分类。若小于 0.5，这种情况说明分类器趋向把正样本分类为负样本，负样本分类为正样本。AUC 的计算方法同时考虑了分类器对于正例和负例的分类能力，在样本不平衡的情况下，依然能够对分类器做出合理的评价。

3.1.3　实验参数设置

1）词向量训练相关参数：词典大小是 63，即一共有 63 种字符串。通过词嵌入映射成的高维向量维度设置为 150，学习率（LR，Learning Rate）设置为 0.1，训练时每一批数据数量为 128，优化器选择 SGD，迭代次数 Epoch 为 5 次。

2）RNN 相关参数：工程内确认实验时，对比双层双向 LSTM 和双层双向 GRU，隐层神经元均设置为 100 个。训练时每一批数据数量为 64，LR = 0.01，优化器选择 SGD，迭代次数 Epoch 为 200 次。跨工程确认实验时，选择工程内确认实验效果较好的模型，其他参数设置不变。

3.1.4　实验过程

本文先分别用 5 个工程进行工程内的缺陷确认实验。采用十折交叉验证的方法，先将每个工程的数据集均分为十份，轮流将其中九份作为训练集，剩下一份作为测试集，共实验十次，并取十次结果的平均值。在跨工程实验中，训练集是某 4 个工程全部样本的集合，测试集是另一个工程的全部样本，并依次将每个工程的全部样本作为测试集。再训练之前工程内实验效果更好的一组 RNN 模型，计算在不同工程下的 AUC 值。同时，基于这些工程引入对比试验。

为了评价在跨工程任务上的表现，本文挑选了两组基准，与它们的实验结果对比。

1）LR：一种传统的方法，基于 28 种通用的度量元特征，这些特征的完整定义在[22]中有详细介绍。

2）FRM-TL：一种基于路径变量特征（PVC，path-variable characteristic）等级匹配的跨工程缺陷确认模型[16]。

3.2　实验结果及分析

3.2.1　工程内缺陷确认实验结果及分析

工程内缺陷确认（WPDI，within project defect identification）反映了当数据集和验证集在向量空间同一区域分布时，State2Vec 方法的检测结果，基于各个工程得到的 AUC 值如表 2 所示。

表 2　工程内缺陷确认实验结果

工程	模型	
	LSTM	GRU
uucp-1.07	0.818	0.843
spell-1.0	0.796	0.750
barcode-0.98	0.805	0.762
sphinxbase-0.3	0.807	0.816
android-4.0	0.865	0.887
总计	0.860	0.879

可以发现，在 uucp，sphinxbase 和 android 这 3 个工程

中，GRU 组的确认效果会更好一些，而在 spell 和 barcode 这两个工程中，LSTM 组的效果更好。具体来看，spell 和 barcode 数据集中警报数量分别为 74 和 129，是 5 个数据集中最少的。样本数量的不充足，导致模型一定程度上欠拟合，在 GRU 组中，spell 和 barcode 的 AUC 值分别为 0.750 和 0.762，低于 LSTM 组的 0.796 和 0.805。在 uucp，sphinxbase 和 android 数据集中，警报的数量分别为 545，240 和 5235，此时 LSTM 组的 AUC 值分别为 0.818，0.807 和 0.865，均低于 GRU 组的 0.843，0.816 和 0.887。根据该发现可以得出，当数据集中警报数量足够多时，选择 GRU 作为神经网络的基本单元，可以更有效地学习特征，达到更好的分类效果。在后续跨工程缺陷确认的实验中，训练集和验证集的警报数量都会大幅度提升，选择 GRU 作为基本单元更加符合实验场景。

3.2.2 跨工程缺陷确认实验结果及分析

跨工程缺陷确认（CPDI，cross project defect identification）反映了当训练集和验证集在向量空间不同区域分布时，State2Vec 方法的检测结果。将该方法与 LR 和 FRM-TL 两组基准对比，各方法在各个工程上所得 ROC 曲线如图 7～11 所示，并将各方法实验结果的 AUC 值记录在表 3 中。在表 3 中，若 State2Vec 的实验结果好于基准 FRM-TL，则将 State2Vec 列的数据加粗，若好于基准 LR，则加入下划线。

表 3 跨工程缺陷确认实验结果

工程	方法		
	State2Vec	FRM-TL	LR
uucp-1.07	0.643	0.608	0.561
spell-1.0	0.607	0.637	0.538
barcode-0.98	0.556	0.539	0.546
sphinxbase-0.3	0.692	0.598	0.525
android-4.0	0.624	0.598	0.559

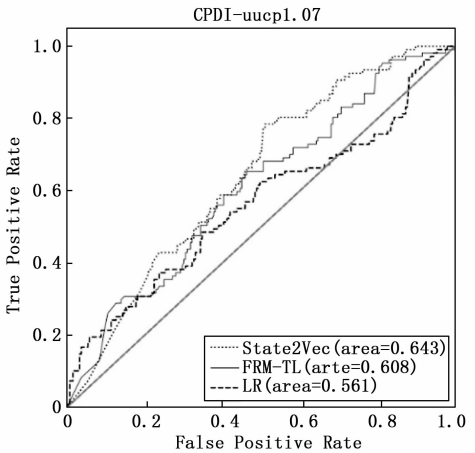

图 7 uucp 数据集上各方法跨工程缺陷确认 ROC 曲线

可以发现，在 uucp，barcode，sphinxbase 和 android 这 4 个工程中，使用 State2Vec 方法确认结果，AUC 值分别为 0.643，0.556，0.692，0.618，确认效果均好于基准 FRM-TL 的 0.608，0.539，0.598，0.598，以及基准 LR 的 0.561，0.546，0.525，0.559。但在工程 spell 中，State2Vec 的 AUC 值为 0.607，低于 FRM-TL 的 0.637，高于 LR 的 0.546。

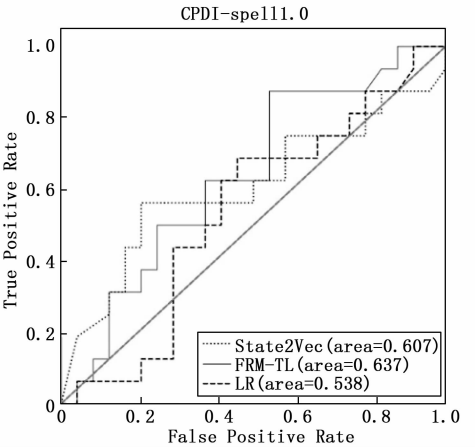

图 8 spell 数据集上各方法跨工程缺陷确认 ROC 曲线

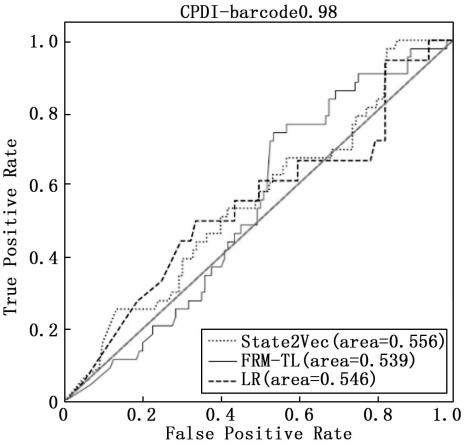

图 9 barcode 数据集上各方法跨工程缺陷确认 ROC 曲线

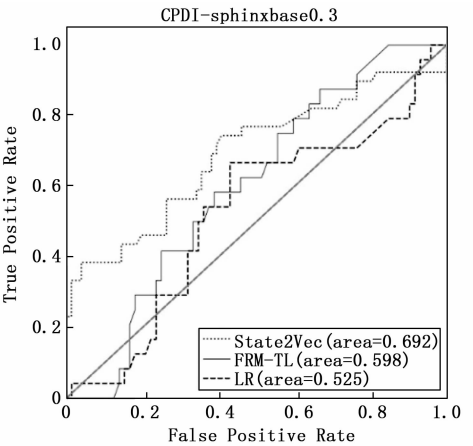

图 10 sphinxbase 数据集上各方法跨工程缺陷确认 ROC 曲线

由此得出，当警报训练集和验证集分布在向量空间的

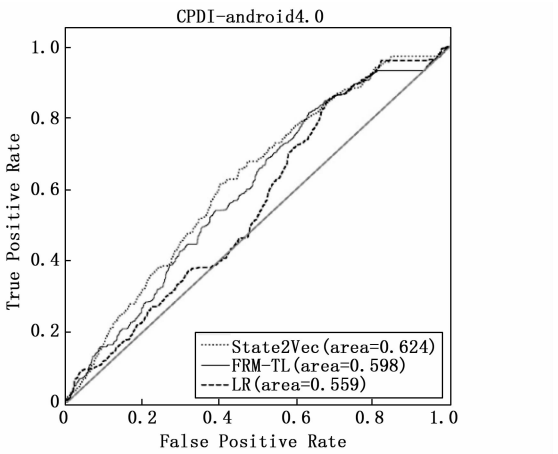

图 11　android 数据集上各方法
跨工程缺陷确认 ROC 曲线

不同区域时，State2Vec 的确认效果会受到一定程度的限制。与 WPDI 的结果相比，CPDI 的 AUC 值在 5 个工程中分别下降了 23.72%，5.93%，27.03%，15.20% 和 30.32%。但是与 FRM-TL 的 CPDI 表现相比，State2Vec 方法在工程 uucp，barcode 以及 android 上确认效果均有小幅度提升，提升幅度依次为 5.76%，3.15%和 3.34%；在工程 sphinxbase 上有较大的提升，提升幅度为 15.72%；但是在工程 spell 中，AUC 值下降了 4.70%，这是因为工程 spell 的警报数据集中仅含 NPD 和 MLF 两种缺陷模式，而由其他 4 个工程构成的警报训练集中包含多种缺陷模式，因此通过学习含有多种缺陷模式的警报数据集中的语义信息，无法准确地将所学知识迁移到目标域，导致指标 AUC 在工程 spell 的警报验证集上下降。与 LR 相比，该方法在工程 uucp，spell，sphinxbase，android 上确认效果均有较大幅度的提升，AUC 指标提升幅度依次为 14.62%，12.83%，31.81%，10.55%；在工程 barcode 上有轻微的提升，提升幅度为 1.83%。

4　结束语

本文提出 State2Vec 方法，该方法面向缺陷模式状态机的检测过程，先收集导致缺陷模式状态机状态发生改变的指令集，再通过特征提取将指令集转换成设计好的字符串序列，最后通过词嵌入技术将字符串映射到高维向量空间。实验证明，State2Vec 方法可以较为有效的提取不同工程中的状态指令集，通过定义并构建结构化的可以表示指令集语法及语义信息的字符串序列，加以词嵌入技术将字符串集合映射到高维向量空间，并以此训练神经网络模型，使得模型能有效学习指令集中的特征，一定程度上提升了在 CPDI 任务场景中的泛化能力。

在日常的开发过程中构建某新的软件项目的警报数据库时，通常缺少足够的历史数据，或者能够用于训练模型的警报数据较为稀疏，导致无法基于同项目警报自动确认技术构建警报数据库，此时可以采用 State2Vec 方法，利用历史项目的警报数据库构建该新项目的警报数据库。

在接下来的工作中，会在更多实际的工程上构建缺陷数据库，用新的数据集检验本文方法的可行性，以及该方法在新的数据集上是否依旧存在较好的泛化能力。因为大多数的程序语言都有语法树和控制流图，也会将该方法拓展到其他程序语言上。另外，未来也考虑构建结构更复杂的神经网络模型，比如将循环神经网络与卷积神经网络（CNN，convolutional neural network）结合，使模型能够更好的学习代码中的语义特征，从而达到更好的分类效果。

参考文献：

[1] BIRD C，BACHMANN A，AUNE E，et al. Fair and balanced? bias in bug-fix datasets [C] //Proceedings of the 7th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering. 2009：121－130.

[2] MUSKE T. Improving review of clustered-code analysis warnings [C] //2014 IEEE International Conference on Software Maintenance and Evolution. IEEE，2014：569－572.

[3] DILLIG I，DILLIG T，AIKEN A. Automated error diagnosis using abductive inference [J]. ACM SIGPLAN Notices，2012，47 (6)：181－192.

[4] HECKMAN S，WILLIAMS L. A systematic literature review of actionable alert identification techniques for automated static code analysis [J]. Information and Software Technology，2011，53 (4)：363－387.

[5] MUSKE T，SEREBRENIK A. Survey of approaches for handling static analysis alarms [C] //2016 IEEE 16th International Working Conference on Source Code Analysis and Manipulation (SCAM). IEEE，2016：157－166.

[6] MCCABE T J. A complexity measure [J]. IEEE Transactions on software Engineering，1976 (4)：308－320.

[7] HALSTEAD M H. Management Prediction-Can Software Science Help? [C] //The IEEE Computer Society's Second International Computer Software and Applications Conference，1978. COMPSAC'78. IEEE，1978：126－128.

[8] CHIDAMBER S R，KEMERER C F. A metrics suite for object oriented design [J]. IEEE Transactions on software engineering，1994，20 (6)：476－493.

[9] WANG S，LIU T，TAN L. Automatically learning semantic features for defect prediction [C] //2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE). IEEE，2016：297－308.

[10] LI J，HE P，ZHU J，et al. Software defect prediction via convolutional neural network [C] //2017 IEEE International Conference on Software Quality，Reliability and Security (QRS). IEEE，2017：318－328.

[11] LI H，LI X，CHEN X，et al. Cross-project Defect Prediction via ASTToken2Vec and BLSTM-based Neural Network [C] //2019 International Joint Conference on Neural Networks (IJCNN). IEEE，2019：1－8.

[12] LEE S, HONG S, YI J, et al. Classifying false positive static checker alarms in continuous integration using convolutional neural networks [C] //2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST). IEEE, 2019: 391 - 401.

[13] QIU S, LU L, CAI Z, et al. Cross-Project Defect Prediction via Transferable Deep Learning-Generated and Handcrafted Features [C] //SEKE, 2019: 431 - 552.

[14] 肖 庆, 杨朝红, 毕学军. 一种基于故障模式状态机的测试方法 [J]. 北京化工大学学报 (自然科学版), 2007, 34 (S1): 73.

[15] MIKOLOV T, SUTSKEVER I, CHEN K, et al. Distributed representations of words and phrases and their compositionality [J]. Advances in neural information processing systems, 2013, 26: 3111 - 3119.

[16] ZHANG Y, JIN D, XING Y, et al. Automated defect identification via path analysis-based features with transfer learning [J]. Journal of Systems and Software, 2020, 166: 1 - 110585. 16.

[17] MUSKE T, TALLURI R, SEREBRENIK A. Repositioning of static analysis alarms [C] //Proceedings of the 27th ACM SIGSOFT international symposium on software testing and analysis. 2018: 187 - 197.

[18] ZHANG D, JIN D, GONG Y, et al. Diagnosis-oriented alarm correlations [C] //2013 20th Asia-Pacific Software Engineering Conference (APSEC). IEEE, 2013, 1: 172 - 179.

[19] YANG Z H, GONG Y Z, XIAO Q, et al. Dts-a software defects testing system [C] //2008 Eighth IEEE International Working Conference on Source Code Analysis and Manipulation. IEEE, 2008: 269 - 270.

[20] TAN M, TAN L, DARA S, et al. Online defect prediction for imbalanced data [C] //2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. IEEE, 2015, 2: 99 - 108.

[21] HE H, GARCIA E A. Learning from imbalanced data [J]. IEEE Transactions on knowledge and data engineering, 2009, 21 (9): 1263 - 1284.

[22] MENZIES T, GREENWALD J, FRANK A. Data mining static code attributes to learn defect predictors [J]. IEEE transactions on software engineering, 2006, 33 (1): 2 - 13.

(上接第 25 页)

4 结束语

通用化、标准化的测试硬件和构件化、模块化的软件平台组成天线测试通用平台。硬件设计充分考虑控制、定时、频率、数据记录的各项指标，满足 40 GHz 以下频率、64 路以下控制信号的模拟相控阵和 15 GHz 以下频率、72 路以下阵面数据、8 GHz 以下光纤速率的子阵数字化或单元数字化相控阵的排故、校准、波瓣测试等常规天线测试需求。利用该平台对天线阵面做功能和性能试验测试，不再依赖雷达除阵面之外其他的分系统，形成通用测试环境和流程，提高天线测试能力和标准化程度，显著提升测试系统搭建和开发效率。经过应用验证，天线测试通用平台满足不同架构的天线测试需求，可大力推广，对于有源相控阵雷达的研制具有重要意义。

参考文献：

[1] TALISA S H, HAVER K W O, COMBERIATE T M, et al. Benefits of Digital Phased Array Radars [J]. Proceedings of the IEEE, 2016, 104 (3): 530 - 543.

[2] 张光义. 相控阵雷达原理 [M]. 北京: 国防工业出版社, 2009.

[3] 吴曼青. 数字阵列雷达及其进展 [J]. 中国电子科学研究院学报, 2006, 1 (1): 11 - 16.

[4] 舒 汀, 陈新竹, 余啟波, 等. 子阵级数字波束形成抗多主副瓣干扰及测角技术 [J]. 现代雷达, 2016, 12 (38): 22 - 26.

[5] 曹书华, 汪凌艳, 王 勋. 子阵级数字阵列雷达波束形成性能分析 [J]. 电子测量技术, 2020, 43 (5): 33 - 38.

[6] 王 峰, 傅有光. 数字相控阵与模拟相控阵雷达的性能对比分析 [J]. 中国电子科学研究院学报, 2012, 7 (2): 148 - 151.

[7] 石国超, 王 璇, 周子巍. 子阵级数字化接收阵列性能分析 [J]. 数字技术与应用, 2020, 38 (8): 73 - 75.

[8] 吴鸿超, 徐欣欢, 薛 羽. 数字相控阵天线阵面的暗室测试方法研究 [J]. 现代雷达, 2015, 37 (8): 49 - 52.

[9] 侯 飞, 柏 利, 乔淑君. 数字化相控阵天线测试方法及测试系统设计 [J]. 计算机测量与控制, 2017, 25 (1): 47 - 49, 53.

[10] 曹俊锋, 倪向东. 有源相控阵雷达天线测试实现与优化 [J]. 火控雷达技术, 2015, 44 (1): 99 - 102.

[11] 方 鑫. 有源相控阵天线近场测试方法研究 [J]. 舰船电子对抗, 2018, 41 (5): 108 - 110.

[12] 王 侃, 王 真. 大型电扫相控阵雷达的天线方向图测试 [J]. 现代雷达, 2015, 37 (8): 55 - 57, 64.

[13] 王亚斌. 具有高精度控制的天线测试系统 [D]. 南京: 南京理工大学, 2019.

[14] 李晓峰, 侯 飞, 白 雪. 天线近场测试软件通用化和多任务设计 [J]. 计算机测量与控制, 2016, 24 (11): 48 - 51.

[15] 陈靖宇, 刘 收. 新一代综合自动测试系统在船用雷达测试保障中的应用研究 [J]. 计算机测量与控制, 2020, 28 (4): 146 - 151.

[16] 吴阳勇, 郭 凯, 王树友, 等. Qt 与 MATLAB 混合编程设计雷达信号验证软件 [J]. 电子测量技术, 2020, 43 (22): 13 - 18.

[17] 王建刚, 吕永乐, 于 鑫. 雷达远程智能诊断与健康评估系统设计与实现 [J]. 测控技术, 2020, 39 (12): 135 - 140.

[18] 张 云. 相控阵天线近场幅相校准 [J]. 中国电子科学研究院学报, 2007, 2 (6): 611 - 614.

[19] 戴 晴, 黄纪军, 莫锦军. 现代微波与天线测量技术 [M]. 北京: 电子工业出版社, 2008.

[20] 谢继东. 微波暗室测试系统软件研究与设计 [D]. 南京: 南京邮电大学, 2015.