Computer Science and Application 计算机科学与应用, 2025, 15(4), 469-477
Published Online April 2025 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2025.154118

Hans汉斯

基于多头注意力机制与特征融合的航天软件静态警报自动确认方法研究

王俊兴，刘紫阳*，周学花，李振乾

北华航天工业学院计算机学院，河北 廊坊

收稿日期：2025年3月25日；录用日期：2025年4月23日；发布日期：2025年4月30日

摘　要

本课题针对航天软件静态测试中误报率高、人工审查工作量大的问题，提出了一种基于深度学习的静态警报自动确认方法。首先，利用预训练模型Codebert对代码进行向量化，解决了传统词嵌入方法无法准确处理代码结构和语义的问题。其次，设计了一种CNN-BiGRU模型，结合卷积神经网络(CNN)的局部特征提取能力和双向门控循环单元(BiGRU)的全局序列特征提取能力，提升了模型的特征提取效果。进一步，为应对长代码中token距离较远、重要信息难以识别的问题，在CNN-BiGRU模型基础上引入多头注意力机制，构建了CNN-BiGRU-Multi-Head Attention模型，使模型能够识别并调整重要token的权重，忽略远距离token的干扰。实验结果表明，所提出的模型在静态警报自动确认任务中表现优异，有效降低了误报率并减少了人工审查的工作量，为软件质量保障提供了高效的技术支持。

关键词

静态警报自动确认，多头注意力，特征融合

Research on Automatic Confirmation Method of Static Alerts for Aerospace Software Based on Multi-Head Attention Mechanism and Feature Fusion

Junxing Wang, Ziyang Liu*, Xuehua Zhou, Zhenqian Li

School of Computer Science and Engineering, North China Institute of Aerospace Engineering, Langfang Hebei

*通讯作者。

文章引用: 王俊兴, 刘紫阳, 周学花, 李振乾. 基于多头注意力机制与特征融合的航天软件静态警报自动确认方法研究[J]. 计算机科学与应用, 2025, 15(4): 469-477. DOI: 10.12677/csa.2025.154118

Received: Mar. 25th, 2025; accepted: Apr. 23rd, 2025; published: Apr. 30th, 2025

Abstract

This topic addresses the problems of high false alarm rate and heavy manual review workload in static testing of aerospace software, and proposes a deep-learning-based automatic confirmation method for static alerts. First, the pre-trained model Codebert is utilized to vectorize the code, which solves the problem that traditional word embedding methods cannot accurately deal with the structure and semantics of the code. Second, a CNN-BiGRU model is designed, which combines the local feature extraction capability of convolutional neural network (CNN) and the global sequential feature extraction capability of bi-directional gated recurrent unit (BiGRU) to improve the feature extraction effect of the model. Further, in order to cope with the problem of long distance of tokens and difficulty in recognizing important information in long codes, the CNN-BiGRU-Multi-Head Attention model is constructed by introducing the Multi-Head Attention mechanism on the basis of the CNN-BiGRU model, so that the model can recognize and adjust the weight of the important tokens, and ignore the interference of the long-distance tokens. The experimental results show that the proposed model performs well in the static alert auto-confirmation task, effectively reduces the false alarm rate and reduces the workload of manual review, and provides efficient technical support for software quality assurance.

Keywords

Static Alert Auto-Confirmation, Multi-Attention, Feature Fusion

1. 引言

在当今数字化世界中，软件已成为各行各业的核心，尤其在航天领域，软件系统的可靠性和安全性直接关系到任务成败和宇航员安全。航天软件测试[1] [2]是确保软件质量的关键环节，其中静态测试通过在开发早期阶段审查代码和文档，能够提前发现问题并显著提高软件可靠性。然而，静态分析工具[3]-[5]存在误报率高(可达 30%~100%)的问题[6]，导致大量误报需要人工排查，增加了工作量和成本。

针对这一问题，本文从航天软件测试工程的角度出发，提出了一种基于深度学习的静态警报自动确认模型，旨在降低误报率并减少人工审查工作量。研究内容主要包括以下三个方面：

1) 静态警报数据集构建与向量化：针对缺乏相关数据集的问题，本文自行制作了静态警报数据集，并通过词法分析和预训练模型 Codebert 将静态警报代码转化为向量表示，解决了传统词嵌入模型(如 Word2Vec)在代码语义理解上的局限性。

2) CNN-BiGRU 模型构建：本文结合卷积神经网络(CNN)的局部特征提取能力和双向门控循环单元(BiGRU)的全局序列建模能力，提出了 CNN-BiGRU 模型。实验表明，该模型能够同时捕获局部特征和上下文信息，显著提升了静态警报自动确认的性能。

3) CNN-BiGRU-Multi-Head Attention 模型构建：为进一步解决长代码中 token 距离较远、重要信息难以识别的问题，本文在 CNN-BiGRU 模型基础上引入多头注意力机制，提出了 CNN-BiGRU-Multi-Head

Attention 模型。该模型通过并行计算多个注意力头，动态调整重要 token 的权重，有效处理长距离依赖问题，进一步提升了模型性能。

2. 相关工作

在静态警报自动确认领域，研究者们通过借鉴自然语言处理(NLP)和深度学习技术，提出了多种创新方法，显著提升了缺陷预测和警报确认的准确性。Xing 等人[7]将 NLP 中的预处理和特征提取技术应用于跨项目缺陷预测(CPDP)，通过连续袋模型将源代码和目标代码文件的抽象语法树转换为数值向量，并提出了生成对抗长短期记忆神经网络(GAN-LSTM)模型，结合生成对抗网络(GAN)和长短期记忆网络(LSTM)的优势，有效提升了跨项目缺陷预测的性能。Pin 等人[8]提出了一种新型的一维卷积神经网络(1D-CNN)结构，通过卷积操作捕捉局部特征，减少过拟合并提升模型的泛化能力。Kharkar 等人[9]则利用基于 Transformer 的自注意力机制，解决了长距离依赖问题，进一步提高了警报确认的准确性。G. S J 等人[10]结合卷积神经网络(CNN)和长短期记忆单元(LSTM)，通过 CNN 提取局部特征，LSTM 捕捉时间序列依赖关系，综合提升了缺陷预测的性能。Sun 等人[11]提出了一种混合深度学习网络模型，结合双向门控循环单元(BiGRU)和深度神经网络(DNN)，通过 BiGRU 捕捉上下文信息，DNN 优化特征表示，显著提升了模型的预测能力。Ansari 等人[12]则提出了一种 CNN-LSTM 混合集合模型，通过集成表现最好的单个模型，综合了不同模型的优势，进一步提高了静态警报自动确认的准确性和鲁棒性。这些研究通过多样化的技术路径，推动了静态警报自动确认领域的发展，为缺陷预测和警报确认提供了更高效、更可靠的解决方案。

3. 方法

CNN-BiGRU-Multi-Head Attention 网络包括 CNN-BiGRU 网络和多头注意力机制层两部分。输入数据首先经过 CNN-BiGRU 模型的输入层，采用预训练模型 Codebert 将输入的静态警报代码转换成 768 维数字向量。随后，数字向量输入到卷积层，本文卷积层采用两个一维卷积层，第一个一维卷积层，输入通道数为 768，输出通道数为 256，卷积核大小 3，激活函数为 ReLU 函数，用于引入非线性，填充方式为“same”，来保持输出的宽度与输入相同。第一层将输入通道数从 768(每个时间步的特征数量)转换为 256。第二个一维卷积层，输入通道数为 256，输出通道数为 128，卷积核大小为 3，它作用于第一个一维卷积层的输出，因此它的输入通道数是 256，输出通道数是 128。通过卷积核提取代码的局部特征。接着，BiGRU 层对卷积层提取的特征进行进一步处理，捕捉代码中的全局序列信息。最后，融合后的特征经过多头注意力机制层，通过并行计算多个注意力头，动态调整重要 token 的权重，增强模型对长距离依赖关系的建模能力，从而提升对复杂代码结构的理解能力。这种层次化的特征提取与融合机制，显著提高了静态警报自动确认的准确性和效率。

3.1. 基于特征融合的 CNN-BiGRU 模块

卷积神经网络(CNN)和双向门控循环单元(BiGRU)在代码分析任务中各有优势：CNN 通过卷积核捕获代码中的局部特征模式，如语法结构和操作符使用模式；而 BiGRU 通过双向递归结构捕捉代码中的前向和后向依赖关系，尤其适合处理跨越多行的控制流和变量作用域等长距离依赖特征。基于两者的互补特性，本研究创新性地构建了 CNN-BiGRU 联合模型，通过融合 CNN 提取的局部特征和 BiGRU 捕获的全局上下文信息，显著提升了模型的特征表达能力。模型结构如图 1 所示。

传统词嵌入模型 Word2Vec 通过 CBOW 或 Skip-Gram 架构将单词映射到低维向量空间，捕捉语义关系，但其存在上下文无关、无法处理一词多义和长距离依赖等局限性。针对这些问题，本文采用基于 Transformer 架构的 CodeBERT 预训练模型，通过双向编码和自注意力机制动态生成上下文相关的词向量，能够区分同一单词在不同上下文中的语义差异，并捕捉代码中的长距离依赖关系。

在静态警报自动确认任务中，输入层采用 Codebert 相比于 Word2Vec，能够解决一词多义问题，为模型提供更准确的输入表示。卷积层采用一维卷积操作，通过滑动窗口提取代码序列中的局部模式，并使用两个卷积层分别将输入通道数从 768 降至 256，再降至 128，同时通过批标准化稳定训练过程。池化层则通过最大池化操作减少特征图尺寸，保留重要信息并降低计算量。

BiGRU 层在卷积层提取局部特征的基础上，进一步捕捉全局特征和长距离依赖关系。通过更新门和重置门机制，解决了传统 RNN 的梯度消失问题，并利用双向结构全面理解上下文信息，增强模型的特征提取能力。最后，输出层将 BiGRU 的输出拼接并传递给分类器。由于任务需要区分真实缺陷及其类型，分类器采用 Sigmoid 函数分别计算每种类别的概率，从而完成静态警报的自动确认。

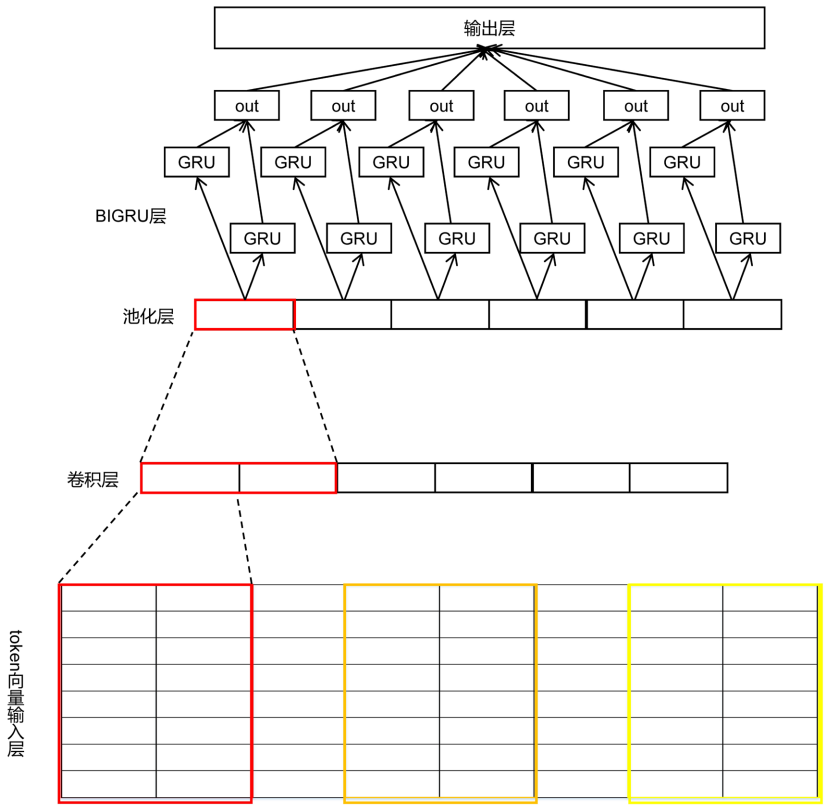

Figure 1. CNN-BiGRU model structure diagram
图 1. CNN-BiGRU 模型结构图

3.2. 多头注意力机制模块

多头注意力(Multi-Head Attention)是注意力机制的扩展形式，通过将输入映射到多个低维空间(称为“头”)，在每个空间中独立应用注意力机制，最后合并结果。这种方式使模型能够在不同特征维度上同时关注多样化信息，显著提升了捕捉复杂模式的能力，是 Transformer 架构的核心组件之一。

多头注意力机制通过并行计算多个注意力头，能够从不同子空间捕获输入序列中多样化的依赖关系。每个注意力头独立工作，可以专注于输入的不同语义特征，这种并行化设计不仅提高了计算效率，还增强了模型的表征能力。具体而言，该机制的核心是对输入序列执行多次缩放点积注意力计算(Scaled Dot-Product Attention)，然后将各注意力头的结果进行拼接和线性变换，最终生成综合的特征表示。多头注意力机制的结构如图 2 所示。

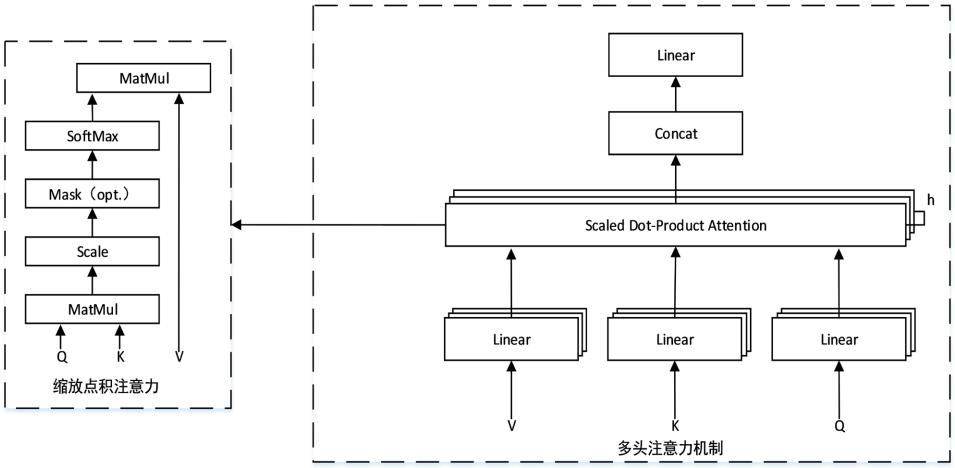

Figure 2. Multi-head attention mechanism structure diagram
图 2. 多头注意力机制结构图

对于查询 Q，键 K 和值 V，先分别进行线性变换得到多个头的表示，计算公式如下：

$$Q_i = W_i^Q Q \tag{1}$$

$$K_i = W_i^K K \tag{2}$$

$$V_i = W_i^V V \tag{3}$$

在每个头中独立通过打分函数计算每个头的注意力分数，打分函数计算公式为：

$$attention\_scores_i = \frac{Q_i K_i^T}{\sqrt{d_k}} \tag{4}$$

应用 *soft*max 函数，将得分转换成权重分布，通过加权求和每个头的上下文向量，计算公式如下：

$$attention\_weights_i = soft\max\left(attention\_scores_i\right) \tag{5}$$

$$context_i = attention\_weights_i V_i \tag{6}$$

将所有头产生的上下文向量拼接起来，再进行一次线性变换获得最终的输出，计算公式为：

$$output = W^o\left[context_1; context_2; L; context_h\right] \tag{7}$$

式中：

d_k ——键的维度；

h ——头的数量；

W_i^Q ——可训练的权重矩阵，用于第 i 个头；

W_i^K ——可训练的权重矩阵，用于第 i 个头；

W_i^V ——可训练的权重矩阵，用于第 i 个头；

; ——拼接操作；

W^o ——可训练的矩阵。

为进一步提升模型对长距离依赖关系的建模能力，本文引入多头注意力机制，增强模型对重要特征的关注能力。输出层通过 sigmoid 函数将聚合的特征映射为概率值，生成预测结果，并通过反向传播优化模型参数。这种端到端的学习机制使模型能够自适应地学习任务相关特征，显著提升了静态警报自动确认的准确性和效率。

3.3. 损失函数

静态警报自动确认任务被视为一种特殊的多标签二分类问题，本文使用二元交叉熵损失函数。损失函数为：

$$BCE\ Loss = -\frac{1}{N}\left[y_i \cdot \log\left(p_i\right) + \left(1 - y_i\right) \cdot \log\left(1 - p_i\right)\right] \tag{8}$$

式中：

N——样本数量；

y_i——第 i 个样本的真实标签，取值为 0 或 1；

p_i——第 i 个样本的预测概率(模型输出的 Sigmoid 值)，范围为[0, 1]。

4. 实验

4.1. 数据集介绍

进行静态警报自动确认研究，获取静态警报代码是必不可少的。然而，目前缺乏相关开源数据集，因此本文手动构建了一个综合性的静态警报数据集，数据来源包括 SARO 缺陷数据集和某型号航天软件操作系统的开源代码。SARO 缺陷数据集可在链接 https://samate.nist.gov/SARD/test-suites 下载，某型号航天软件操作系统的开源代码为在北京航天自动控制研究所实习期间，研究所提供的非涉密但未公开的操作系统源码。通过静态分析工具对代码进行测试，生成静态分析报告，并利用 Python 工具提取警报代码信息。根据《GJB 8114 C/C++ 语言编程安全子集》和航天软件工程需求，对警报代码进行人工标注，生成 one-hot 编码标签。最终构建的数据集包含 18,585 条静态警报代码，其中 6441 条为真实缺陷，12,144 条为误报。

4.2. 实验设置

本文实验选用 PyTorch 框架实现模型，并在 NVIDIA GeForce GTX 1050 Ti 上进行训练和测试。对于模型的具体参数配置，本文采用 Adam 优化器，学习率为 1e-4。由于受到 GPU 资源的限制，通过大量的实验验证，本文将批量大小(batch size)设置为 64，训练周期(epoch)设为 16，正则化参数(dropout)设置为 0.5，以保证模型能够达到收敛。

4.3. 评价指标

在静态警报自动确认任务中，每条静态警报代码可以在是真实缺陷的前提下违反多个编码规则，所以需要对警报代码的每个类别都采用 sigmoid 函数，单独去计算警报代码属于每个类别的概率。

为全面评估各模型在静态警报自动确认任务中的性能，本研究采用多维度评价指标体系，包括 Accuracy(准确率)、Precision(精确率)、Recall(召回率)和 F1 分数(F1-score)。对每一个类别计算混淆矩阵的值，继而可以计算出所有类的总体值。有了上面这四个所有类总体值之后，就可以通过计算得到上面这四个常用的评估指标。

其中，Accuracy 反映模型整体预测的准确程度，Precision 衡量预测为正例的样本中真实正例的比例，Recall 评估模型识别正例的完整度，而 F1 分数则综合了 Precision 和 Recall 的平衡表现。通过这四个指标的协同分析，能够从不同角度全面评估模型的分类性能。

$$TP = \sum_i TP_i \tag{9}$$

$$FP = \sum_i FP_i \tag{10}$$

$$FN = \sum_i FN_i \tag{11}$$

$$TN = \sum_i TN_i \tag{12}$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{13}$$

$$Percision = \frac{TP}{TP + FP} \tag{14}$$

$$\mathrm{Re}call = \frac{TP}{TP + FN} \tag{15}$$

$$F1Score = 2 \cdot \frac{Percision \cdot \mathrm{Re}call}{Percision + \mathrm{Re}call} \tag{16}$$

式中：

TP_i——第 i 类被正确预测为正类的样本数；

FP_i——第 i 类被错误预测为正类的样本数；

FN_i——第 i 类被错误预测为负类的样本数；

TN_i——第 i 类被正确预测为负类的样本数；

TP 为真正类，用于表示被正确预测为正类的样本数；

FP 为假正类，用于表示被错误预测为正类的样本数；

TN 为真负类，用于表示被正确预测为负类的样本数；

FN 为假负类，用于表示被错误预测为负类的样本数。

4.4. 对比方法

为了证明本文方法的改进方法的有效性，设置多组对比实验。对于模型的选择包括卷积神经网络，双向门控循环单元以及本文提出的 CNN-BiGRU 模型，对于代码序列向量化方法的选择包括 Work2Vec 和 Codebert，对于注意力机制的选择包括传统的注意力机制和多头注意力机制。

4.5. 实验结果与分析

本节在自建数据集上对选择不同模块的不同方法进行对比实验。在自建数据集上选择 Work2Vec 和 Codebert 代码序列向量化方法时，三个模型的定量结果如表 1 所示。

Table 1. Quantitative results of vectorization of Work2Vec and Codebert code sequences
表 1. Work2Vec 和 Codebert 代码序列向量化的定量结果

模型	Accuracy	Precision	Recall	F1 分数
Work2Vec-CNN	95.80%	93.12%	87.82%	89.04%
Work2Vec-BiGRU	96.56%	96.49%	92.37%	93.22%
Work2Vec-CNN-BiGRU	97.36%	98.37%	96.37%	96.74%
Codebert-CNN	96.80%	96.15%	91.72%	92.99%
Codebert-BiGRU	97.36%	98.28%	95.50%	95.96%
Ours (Codebert-CNN-BiGRU)	97.54%	98.40%	97.56%	97.28%

Work2Vec 和 Codebert 代码序列向量化的定量结果说明，选用 Codebert 代码序列向量化方法的 CNN、

BiGRU 和 CNN-BiGRU 模型均优于选用传统 Word2Vec 代码序列向量化方法的模型，主要得益于 Codebert 的上下文感知特性，能够动态捕捉 token 间的长距离依赖关系。进一步对比发现，CNN-BiGRU 模型在各项指标上均优于单一 CNN 和 BiGRU 模型，尤其是在使用 Codebert 时，准确率、精确率、召回率和 F1 分数分别提升了 0.74%、2.25%、5.84%和 4.29%(相较于 CNN)以及 0.18%、0.12%、2.06%和 1.32%(相较于 BiGRU)，验证了其通过融合 CNN 的局部特征提取能力和 BiGRU 的序列建模优势显著提升了性能。

上文对比实验已经表明 Codebert 代码序列向量化方法相比于传统的 Work2Vec 方法的优越性，CNN-BiGRU 模型相比于单模型(CNN 或 BiGRU)能进行更全面的特征提取，所以下面的对比实验均选用 Codebert 代码序列向量化方法和 CNN-BiGRU 模型。对比无注意力机制、传统注意力机制和多头注意力机制对模型的影响的定量结果如表 2 所示。

Table 2. Quantitative results of attention mechanisms
表 2. 注意力机制的定量结果

模型	Accuracy	Precision	Recall	F1 分数
CNN-BiGRU	97.31%	98.36%	95.78%	96.42%
CNN-BiGRU-Attention	97.44%	99.11%	96.24%	97.07%
CNN-BiGRU-Multi-Head Attention	97.66%	99.20%	97.39%	97.78%

注意力机制的定量结果表明引入传统注意力机制的 CNN-BiGRU-Attention 模型优于基础 CNN-BiGRU 模型，而 CNN-BiGRU-Multi-Head Attention 模型在准确率、精确率、召回率和 F1 分数上分别比 CNN-BiGRU 模型提升了 0.31%、0.84%、1.61%和 1.36%，相较于 CNN-BiGRU-Attention 模型分别提升了 0.22%、0.1%、1.15%和 0.71%。这一性能提升得益于多头注意力机制的设计，能够从不同子空间捕获多样化的依赖关系，显著增强了模型的表达能力。

4.6. 消融实验

消融实验主要是对代码序列向量化方法、CNN-BiGRU 模型和多头注意力机制的效果进行验证，消融实验的定量结果如表 3 所示。

Table 3. Quantitative results of ablation experiments
表 3. 消融实验的定量结果

模型	Accuracy	Precision	Recall	F1 分数
Work2Vec-CNN	95.80%	93.12%	87.82%	89.04%
Codebert-CNN	96.80%	96.15%	91.72%	92.99%
Codebert-CNN-BiGRU	97.54%	98.40%	97.56%	97.28%
Codebert-CNN-BiGRU-Multi-Head Attention	97.66%	99.20%	97.39%	97.78%

从表 3 的消融实验定量结果可以看出，Codebert-CNN-BiGRU-Multi-Head Attention 在选用 Codebert 代码序列向量化方法、采用 CNN-BiGRU 融合模型并且引入多头注意力机制的共同作用下，在数据集上的分类预测能力显著优于移除任意模块后的模型版本。这充分说明了 Codebert 代码序列向量化方法、CNN-BiGRU 融合模型并且多头注意力机制对模型提升整体性能的关键贡献。

5. 总结

在当今数字化时代，软件已成为推动社会发展的核心动力，尤其在航天领域，软件系统的可靠性和安全性至关重要。然而，传统静态分析工具存在误报率高、人工验证成本大等问题，且缺乏公开的基准数据集，制约了相关研究的深入开展。针对这些问题，本研究从航天软件工程的实际需求出发，构建了专用静态警报数据集，并提出了一种基于深度学习的静态警报自动确认方法。首先，通过静态分析工具收集静态警报代码，并采用 Codebert 预训练模型将 token 序列映射为 768 维数字向量，克服了传统词嵌入模型的局限性。其次，创新性地将卷积神经网络(CNN)与双向门控循环单元(BiGRU)结合，提出了 CNN-BiGRU 模型，实现了局部特征与全局序列信息的有效融合。进一步，针对长距离依赖问题，在 CNN-BiGRU 模型基础上引入多头注意力机制，增强了模型对复杂代码结构的理解能力。本研究的创新点包括专用数据集的构建、CNN-BiGRU 混合模型的设计以及多头注意力机制的引入。然而，研究仍存在数据集来源单一、参数调优不全面以及实时性与效率不足等局限性，未来需进一步扩展数据来源、优化模型设计并提升实际应用价值。

参考文献

[1] Bessey, A., Block, K., Chelf, B., Chou, A., Fulton, B., Hallem, S., *et al.* (2010) A Few Billion Lines of Code Later: Using Static Analysis to Find Bugs in the Real World. *Communications of the ACM*, **53**, 66-75. https://doi.org/10.1145/1646353.1646374

[2] Yang, Z.H., Gong, Y.Z., Xiao, Q. and Wang, Y.W. (2008) DTS—A Software Defects Testing System. 2008 *Eighth IEEE International Working Conference on Source Code Analysis and Manipulation*, Beijing, 28-29 September 2008, 269-270. https://doi.org/10.1109/scam.2008.12

[3] Kremenek, T. and Engler, D. (2003) Z-Ranking: Using Statistical Analysis to Counter the Impact of Static Analysis Approximations. In: Cousot, R., Ed., *Static Analysis*, Springer, 295-315. https://doi.org/10.1007/3-540-44898-5_16

[4] Johnson, B., Song, Y., Murphy-Hill, E. and Bowdidge, R. (2013) Why Don't Software Developers Use Static Analysis Tools to Find Bugs? 2013 35*th International Conference on Software Engineering* (*ICSE*), San Francisco, 18-26 May 2013, 672-681. https://doi.org/10.1109/icse.2013.6606613

[5] Kang, H.J., Aw, K.L. and Lo, D. (2022) Detecting False Alarms from Automatic Static Analysis Tools: How Far Are We? *Proceedings of the* 44*th International Conference on Software Engineering*, Pittsburgh, 21-29 May 2022, 698-709. https://doi.org/10.1145/3510003.3510214

[6] Kim, S. and Ernst, M.D. (2007) Which Warnings Should I Fix First? *Proceedings of the* 6*th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Dubrovnik, 3-7 September 2007, 45-54. https://doi.org/10.1145/1287624.1287633

[7] Xing, Y., Qian, X., Guan, Y., Yang, B. and Zhang, Y. (2022) Cross-Project Defect Prediction Based on G-LSTM Model. *Pattern Recognition Letters*, **160**, 50-57. https://doi.org/10.1016/j.patrec.2022.04.039

[8] Pin, K., Ho Chang, J. and Nam, Y. (2022) Software Defect Prediction Harnessing on Multi 1-Dimensional Convolutional Neural Network Structure. *Computers*, *Materials & Continua*, **71**, 1521-1546. https://doi.org/10.32604/cmc.2022.022085

[9] Kharkar, A., Moghaddam, R.Z., Jin, M., Liu, X., Shi, X., Clement, C., *et al.* (2022) Learning to Reduce False Positives in Analytic Bug Detectors. *Proceedings of the* 44*th International Conference on Software Engineering*, Pittsburgh, 21-29 May 2022, 1307-1316. https://doi.org/10.1145/3510003.3510153

[10] G, S.J. and Charles, J. (2024) Revolutionizing Software Project Development: A CNN-LSTM Hybrid Model for Effective Defect Prediction. *International Journal of Advanced Computer Science and Applications*, **15**, 595-603. https://doi.org/10.14569/ijacsa.2024.0150158

[11] Sun, H. (2024) Network Intrusion Detection Using Transformer and BiGRU-DNN in Edge Computing. *Journal of Information Processing Systems*, **20**, 458-476.

[12] Ansari, N.U. and Richhariya, P. (2024) Deep Hybrid Intelligence: CNN-LSTM for Accurate Software Bug Prediction. *International Journal of Innovations in Science Engineering and Management*, **3**, 26-33. https://doi.org/10.69968//ijisem.2024v3i426-33