Computer Science and Application 计算机科学与应用, 2024, 14(9), 33-46
Published Online September 2024 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2024.149185

Hans汉斯

基于注意力机制的虚拟机故障检测算法

党杜均

仁寿智仁智慧科技有限公司，四川 仁寿

收稿日期：2024年7月30日；录用日期：2024年8月27日；发布日期：2024年9月4日

摘　要

随着云计算和虚拟化技术的广泛应用，虚拟机故障检测成为保障数据中心和云计算平台稳定性和可靠性的重要环节。传统的虚拟机故障检测方法在处理复杂多变的虚拟化环境时，往往存在检测精度低、响应速度慢等问题。本文提出一种基于注意力机制的虚拟机故障检测算法，通过引入注意力机制，模型能够更精准地关注故障相关的关键特征，从而提高故障检测的准确性和效率。该方法结合卷积神经网络(CNN)和长短期记忆网络(LSTM)模型，并在其中嵌入注意力机制，对虚拟机的运行数据(如日志、性能指标等)进行分析和处理。实验结果表明，与传统故障检测算法相比，本文提出的算法在故障检测的准确率、召回率和F1-score等方面均有较大提升。

关键词

虚拟机，故障检测，注意力机制，卷积神经网络，长短期记忆网络

Virtual Machine Fault Detection Algorithm Based on Attention Mechanism

Dujun Dang

Renshou Zhiren Intelligent Technology Co., Ltd., Renshou Sichuan

Received: Jul. 30th, 2024; accepted: Aug. 27th, 2024; published: Sep. 4th, 2024

Abstract

With the widespread adoption of cloud computing and virtualization technologies, virtual machine (VM) fault detection has become crucial for ensuring the stability and reliability of data centers and cloud platforms. Traditional fault detection methods often suffer from low detection accuracy and slow response times when dealing with complex and dynamic virtualization environments. This paper proposes a VM fault detection algorithm based on attention mechanisms. By incorporating attention mechanisms, the model can more precisely focus on key features related to faults, thereby improving detection accuracy and efficiency. This method integrates Convolutional Neural Networks

文章引用: 党杜均. 基于注意力机制的虚拟机故障检测算法[J]. 计算机科学与应用, 2024, 14(9): 33-46.
DOI: 10.12677/csa.2024.149185

(CNN) and Long Short-Term Memory (LSTM) models with embedded attention mechanisms to analyze and process VM operational data, such as logs and performance metrics. Experimental results show that, compared to traditional fault detection algorithms, the proposed algorithm significantly improves fault detection accuracy, recall, and F1-score.

Keywords

Virtual Machine, Fault Detection, Attention Mechanism, Convolutional Neural Network, Long Short-Term Memory

1. 引言

随着云计算和虚拟化技术的快速发展，虚拟机(Virtual Machine, VM)已成为数据中心和云计算平台中广泛使用的基础技术之一。虚拟机的灵活性和资源利用效率使其在计算资源管理中具有重要地位。然而，随着虚拟化环境的复杂性和规模的不断增加，虚拟机故障也变得越来越频繁。常见的虚拟机故障包括 CPU 故障(如过高的 CPU 使用率、CPU 过热)、内存故障(如内存泄漏、内存不足)、存储故障(如磁盘空间不足、磁盘 IO 性能问题)、网络故障(如网络延迟、网络中断)、操作系统故障(如操作系统崩溃、内核恐慌)、应用程序故障(如应用程序崩溃、应用程序响应缓慢)、硬件故障(如物理主机硬件故障、虚拟化平台故障)和安全故障(如恶意软件感染、未授权访问)。

虚拟机故障可能导致服务中断、数据丢失和系统性能下降，严重影响云平台的稳定性和可靠性。因此，如何及时、准确地检测和处理虚拟机故障，成为保障数据中心和云平台高效运行的关键问题。

目前，针对虚拟机故障检测的研究已取得一定进展。传统的故障检测方法主要依赖于预设规则和阈值，通过监控虚拟机的运行状态和性能指标，发现异常情况[1]。然而，这些方法在处理复杂多变的虚拟化环境时，往往存在检测精度低、响应速度慢等问题。此外，基于机器学习的故障检测方法近年来逐渐受到关注，通过学习历史故障数据和特征，构建检测模型，取得了较好的效果。然而，这些方法在应对动态变化的虚拟机运行环境时，仍存在一定的局限性[2]。

为了克服传统方法的局限性，本文提出一种基于注意力机制的虚拟机故障检测算法。注意力机制作为一种在深度学习中广泛应用的技术，能够有效提升模型对关键特征的关注能力[3]。通过引入注意力机制，模型能够更精准地捕捉虚拟机故障相关的关键特征，从而提高故障检测的准确性和效率。本文方法结合卷积神经网络(Convolutional Neural Network, CNN)和长短期记忆网络(Long Short-Term Memory, LSTM)，并在其中嵌入注意力机制，对虚拟机的运行数据(如日志、性能指标等)进行分析和处理。

本文的主要工作如下：

1) 提出一种基于注意力机制的虚拟机故障检测算法，结合 CNN 和 LSTM 模型，提升故障检测的准确性和效率。

2) 通过实验验证本文算法在不同负载条件下的适应性和稳定性，并与传统故障检测算法进行对比，证明其在实际应用中的优越性。

本文结构如下：第二部分介绍相关工作，包括虚拟机故障检测方法和注意力机制的研究现状；第三部分详细描述基于注意力机制的虚拟机故障检测算法，包括模型架构和数据处理方法；第四部分进行实验与结果分析，评估算法在不同条件下的性能表现；第五部分总结研究工作并提出未来的研究方向。

2. 相关工作

虚拟机故障检测在云计算和虚拟化技术发展过程中起着至关重要的作用。传统的故障检测方法主要依赖于预设规则和阈值，通过监控虚拟机的运行状态和性能指标(如 CPU 使用率、内存占用率等)，检测异常情况。例如，Ganglia [4]和 Nagios [5]等监控系统常用于数据中心的故障检测。然而，这些方法在面对复杂和动态变化的虚拟化环境时，检测精度往往不足，容易出现误报或漏报。

随着机器学习技术的发展，基于机器学习的故障检测方法逐渐成为研究热点。通过对历史故障数据和特征进行学习，机器学习算法能够自动构建故障检测模型，提高检测的准确性和自动化水平。常见的方法包括支持向量机(SVM) [6]-[9]、决策树[10]-[14]、随机森林[15]-[18]和神经网络[19]等。这些方法在一定程度上提升了故障检测的性能，但在处理动态和多变的虚拟化环境时，仍存在适应性和泛化能力不足的问题。

注意力机制(Attention Mechanism) [3]是近年来在深度学习领域取得重大突破的一种技术，最初应用于自然语言处理(NLP)任务，如机器翻译和文本生成。注意力机制通过计算输入数据中各部分的重要性权重，使模型能够更关注关键特征，从而提高性能。

在故障检测领域，注意力机制也逐渐被引入[20]。通过结合注意力机制，故障检测模型能够更有效地捕捉和关注与故障相关的关键特征，提高故障检测的准确性和效率。注意力机制在图像处理中的应用，如结合卷积神经网络(CNN)的图像分类和目标检测，以及在时间序列数据中的应用，如结合长短期记忆网络(LSTM)的时间序列预测，为虚拟机故障检测提供了新的思路。

卷积神经网络(CNN)和长短期记忆网络(LSTM)是深度学习中两种重要的模型结构，广泛应用于各种任务中[21]。CNN 以其在图像处理中的卓越性能而闻名。它通过卷积层和池化层的组合，从输入数据中提取局部特征，并逐层组合成更高层次的抽象特征。CNN 在处理虚拟机运行数据(如性能指标和日志数据)时，可以有效提取空间和局部特征，识别出与故障相关的模式。LSTM 是一种特殊的递归神经网络(RNN)，专门用于处理和预测时间序列数据。LSTM 通过引入门控机制，有效解决了传统 RNN 的梯度消失和梯度爆炸问题，能够记住长时间的依赖信息。对于虚拟机的运行数据，LSTM 可以捕捉时间序列中的长期依赖关系，识别出故障前的预兆和特征。

为了充分利用 CNN 和 LSTM 的优势，研究者提出了结合两者的混合模型，用于虚拟机故障检测。混合模型通过 CNN 提取数据的局部和空间特征，再由 LSTM 捕捉时间序列的长期依赖关系。这种方法在故障检测中表现出色，但仍然存在对关键特征关注不足的问题[22]。

近年来，基于注意力机制的虚拟机故障检测研究逐渐增多。通过引入注意力机制，研究者能够提高模型对故障相关特征的关注度，进一步提升故障检测的准确性。例如，一些研究结合注意力机制和 LSTM 模型，对虚拟机日志数据进行分析，实现了更精准的故障检测[23] [24]；另一些研究则在 CNN 模型中嵌入注意力机制，优化虚拟机性能指标的特征提取过程[25] [26]。

综上所述，虚拟机故障检测方法经历了从传统规则和阈值检测到基于机器学习的方法，再到结合注意力机制的演进过程。尽管现有方法在一定程度上提升了故障检测的性能，但在复杂和动态的虚拟化环境中，仍存在诸多挑战。本文提出的基于注意力机制的虚拟机故障检测算法，可进一步提高检测的准确性和效率，填补现有研究的不足，为数据中心和云计算平台的稳定运行提供更可靠的技术支持。

3. 模型架构及算法设计

3.1. 模型架构设计

为了解决虚拟机故障检测中的关键特征提取问题，我们设计了一种结合卷积神经网络(CNN)、长短期

记忆网络(LSTM)和注意力机制的混合模型。该模型架构如图 1 所示。

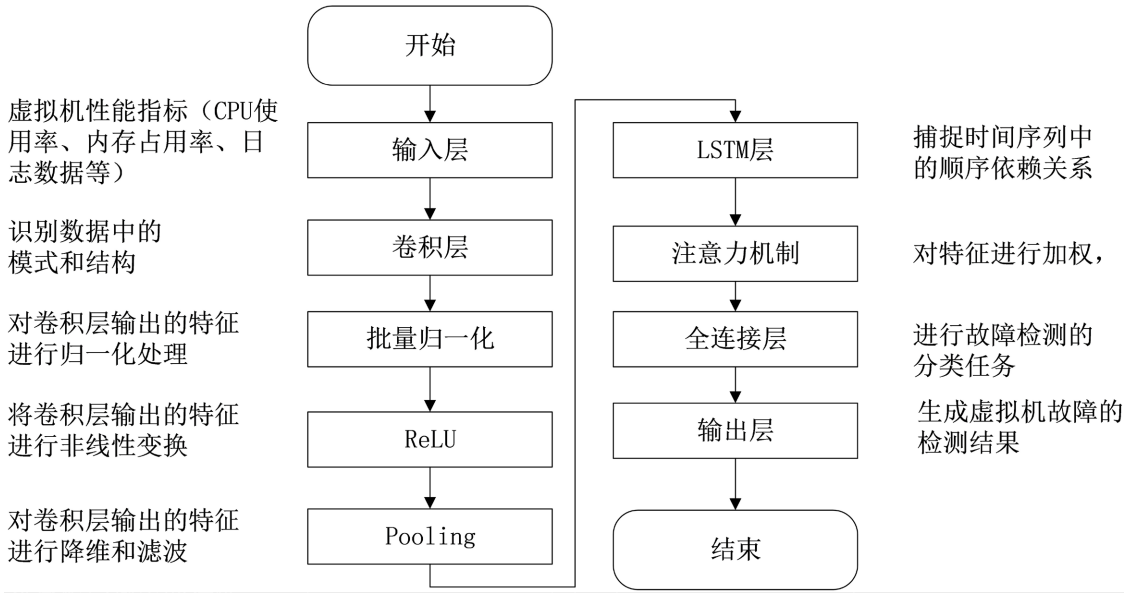

Figure 1. Architecture of CNN-LSTM virtual machine fault detection model based on attention mechanism
图 1. 基于注意力机制的 CNN-LSTM 虚拟机故障检测模型结构

该架构中，输入层接收虚拟机的性能指标数据 $\mathbf{X} \in \mathbb{R}^{n \times d}$，其中 n 为样本数，d 为特征维度，如 CPU 使用率、内存占用率和日志数据，为后续处理提供基础输入。

卷积层通过卷积操作提取输入数据的局部特征，识别数据中的模式和结构。计算公式如下：

$$\mathbf{H}_{ij}^{(k)} = \sigma\left(\sum_{m=1}^{M}\sum_{n=1}^{N}\mathbf{W}_{mn}^{(k)}\mathbf{X}_{(i+m-1)(j+n-1)} + b^{(k)}\right) \tag{1}$$

$\mathbf{H}_{ij}^{(k)}$ 是第 k 个卷积核在位置 (i, j) 的输出，$\mathbf{W}_{mn}^{(k)}$ 是第 k 个卷积核的权重，$b^{(k)}$ 是偏置项，σ 是激活函数。

批量归一化层(Batch Normalization, BN)对卷积层输出的特征进行归一化处理，减小内部协变量偏移，加速模型训练，提高稳定性。计算公式如下：

$$\hat{\mathbf{H}} = \frac{\mathbf{H} - \mu}{\sqrt{\sigma^2 + \epsilon}}\gamma + \beta b^{(k)} \tag{2}$$

$\mathbf{H}$ 是卷积层输出，μ 和 σ 分别是均值和方差，ϵ 是一个很小的数以防止分母为零，γ 和 β 是可学习的参数。

ReLU 激活层(ReLU Activation)应用 ReLU 激活函数，将卷积层输出的特征进行非线性变换，增加模型的非线性表达能力。计算公式如下：

$$\mathbf{A} = \max\left(0, \hat{\mathbf{H}}\right) \tag{3}$$

$\mathbf{A}$ 是激活后的输出，$\hat{\mathbf{H}}$ 是批量归一化后的输入。

池化层对卷积层输出的特征进行降维和滤波，减小特征图的尺寸，保留重要特征，减少计算量。计算公式如下：

$$\mathbf{P}_{ij}^{(k)} = \max_{m,n}\left(\mathbf{A}_{(i+m-1)(j+n-1)}^{(k)}\right) \tag{4}$$

$\mathbf{P}_{ij}^{(k)}$ 是第 k 个特征图在位置 (i,j) 的池化输出，通常使用最大池化。

LSTM 层捕捉时间序列中的顺序依赖关系，通过长短期记忆网络处理数据的时间维度特征，识别出与故障相关的长期依赖信息。

输入门计算公式如下：

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i\right) \tag{5}$$

遗忘门计算公式如下：

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f\right) \tag{6}$$

输出门计算公式如下：

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o\right) \tag{7}$$

记忆单元计算公式如下：

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh\left(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c\right) \tag{8}$$

隐藏状态计算公式如下：

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right) \tag{9}$$

$\mathbf{x}_t$ 是当前输入，$\mathbf{h}_{t-1}$ 是前一时刻的隐藏状态，$\mathbf{W}$、$\mathbf{U}$、$\mathbf{b}$ 是可学习的参数，$\odot$ 表示元素乘。

注意力机制对 LSTM 层输出的特征进行加权，重点关注与故障相关的关键特征，增强模型的特征选择能力。注意力权重计算公式如下：

$$\alpha_t = \frac{\exp\left(e_t\right)}{\sum_{t'} \exp\left(e_{t'}\right)} \tag{10}$$

注意力评分计算如下：

$$e_t = \mathbf{v}^{\mathrm{T}} \tanh\left(\mathbf{W}_1 \mathbf{h}_{t-1} + \mathbf{W}_2 \mathbf{s}_t + \mathbf{b}\right) \tag{11}$$

上下文向量计算如下：

$$\mathbf{c} = \sum_t \alpha_t \mathbf{s}_t \tag{12}$$

$\mathbf{h}_{t-1}$ 是前一时刻的隐藏状态，$\mathbf{s}_t$ 是当前输入的隐状态，$\mathbf{W}_1$、$\mathbf{W}_2$、$\mathbf{v}$、$\mathbf{b}$ 是可学习的参数。

全连接层将前面层提取的特征进行组合和映射，实现故障检测的分类任务。

$$\mathbf{y} = \sigma\left(\mathbf{W}_c + \mathrm{b}\right) \tag{13}$$

$\mathbf{c}$ 是注意力机制的输出，$\mathbf{W}$ 和 b 是全连接层的权重和偏置，σ 是激活函数。

输出层生成虚拟机故障的检测结果，输出故障概率分布或分类标签，用于实际的故障识别和处理。

$$\hat{\mathbf{y}} = \mathrm{softmax}\left(\mathbf{y}\right) \tag{14}$$

$\mathbf{y}$ 是全连接层的输出，$\hat{\mathbf{y}}$ 是故障检测的概率分布，通过 softmax 函数生成。

3.2. 算法流程及分析

首先对虚拟机性能数据进行预处理，包括数据清洗和标准化。然后，使用卷积神经网络(CNN)提取数据的局部特征，通过批量归一化(BN)、ReLU 激活和池化层进一步处理特征。接着，使用长短期记忆网络(LSTM)捕捉时间序列中的顺序依赖关系，生成隐藏状态。应用注意力机制对隐藏状态进行加权，生成上下文向量。最后，将上下文向量输入全连接层进行分类，输出故障检测的预测结果。算法伪代码如下：

Algorithm: VM Fault Detection Algorithm Based on Attention Mechanism

Input: Data - Virtual machine performance metrics (CPU usage, Memory usage, Logs)

Output: Prediction - Fault detection prediction

function preprocess_data(data)
    cleaned_data ← clean(data) // 数据清洗
    standardized_data ← standardize(cleaned_data) // 数据标准化处理
    return standardized_data
end function

function feature_extraction(data)
    conv_out ← σ(W * data + b) // 卷积层
    bn_out ← ((conv_out - μ) / √(σ^2 + ε)) * γ + β // 批量归一化
    relu_out ← max(0, bn_out) // ReLU 激活
    pooled_out ← max_pool(relu_out) // 池化层
    return pooled_out
end function

function lstm_processing(data)
    $\mathbf{i}_t \leftarrow \sigma\left(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i\right)$ // 输入门
    $\mathbf{f}_t \leftarrow \sigma\left(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f\right)$ // 遗忘门
    $\mathbf{o}_t \leftarrow \sigma\left(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o\right)$ // 输出门
    $\mathbf{c}_t \leftarrow \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh\left(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c\right)$ // 记忆单元
    $\mathbf{h}_t \leftarrow \mathbf{o}_t * \tanh\left(\mathbf{c}_t\right)$ // 隐藏状态
    return $\mathbf{h}_t$
end function

function attention_mechanism(hidden_state)
    attention_weights ← softmax(score_function(hidden_state)) // 注意力权重
    context_vector ← sum(attention_weights * hidden_state) // 上下文向量
    return context_vector
end function

function classification(context_vector)
    output ← W * context_vector + b // 全连接层
    prediction ← softmax(output) // 输出层
    return prediction
end function

function vm_fault_detection_algorithm(data)
    preprocessed_data ← preprocess_data(data) // 数据预处理
    features ← feature_extraction(preprocessed_data) // 特征提取
    hidden_state ← lstm_processing(features) // 时间序列处理

```
35:     context_vector ← attention_mechanism(hidden_state) // 注意力机制
36:     prediction ← classification(context_vector)        // 分类任务
37:     return prediction
38: end function

39: data ← load_data('vm_performance_metrics.csv')          // 加载数据
40: prediction ← vm_fault_detection_algorithm(data)         // 运行故障检测算法
41: print("Fault Detection Prediction:", prediction)       // 输出故障检测预测结果
```

该算法的时间复杂度主要包括卷积层的 $O\left(n\times d\times k^2\right)$，LSTM 层的 $O\left(n\times T\times h^2\right)$，以及全连接层的 O(h × c)，总时间复杂度约为 $O\left(n\times d\times k^2+T\times n\times h^2+h\times c\right)$。空间复杂度主要由卷积层的 $O\left(n\times d\times k^2\right)$、LSTM 层的 $O\left(T\times h\right)$ 和全连接层的 $O\left(h\times c\right)$ 组成，总空间复杂度约为 $O\left(n\times d\times k^2+T\times h+h\times c\right)$，其中，$n$ 表示输入数据的样本数，即虚拟机的数量。D 表示输入数据的特征维度，即每个虚拟机的性能指标数量。K 为卷积核的大小，卷积层用于特征提取的参数。T 是时间步长，即 LSTM 层处理的时间序列长度。H 为 LSTM 层的隐藏状态维度，表示 LSTM 单元的数量。c 是全连接层的输出维度，即分类任务的类别数。

4. 实验分析

4.1. 实验环境

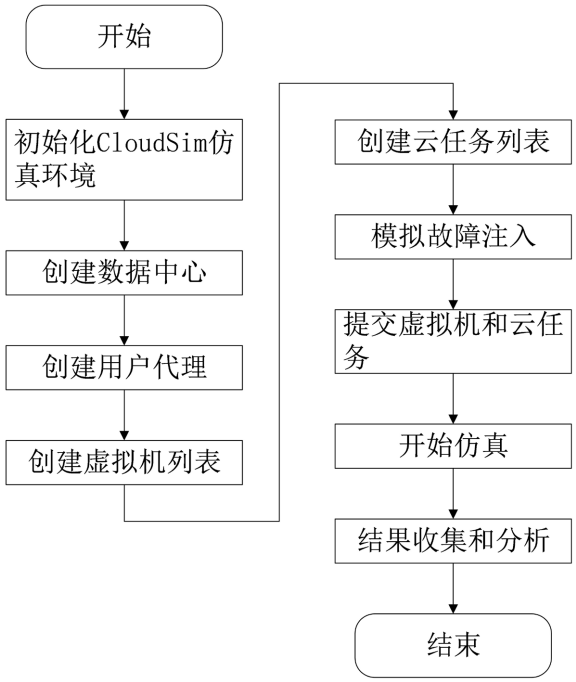

Figure 2. Virtual machine fault detection simulation process flowchart
图 2. 虚拟机故障检测仿真过程流程图

在本实验中，我们使用 CloudSim [27]作为仿真平台来模拟虚拟机故障检测数据。CloudSim 是一个广泛使用的云计算仿真工具，支持模拟复杂的云计算环境和虚拟机管理。虚拟机故障检测仿真过程流程图如图 2 所示。首先，初始化 CloudSim 仿真环境，并创建配置良好的数据中心和用户代理(Datacenter Broker)以管理虚拟机和云任务。配置并创建虚拟机列表和云任务列表，设置它们的资源需求和调度策略。在仿

真过程中，随机选择虚拟机并注入故障，例如增加 CPU 使用率或内存使用率，以模拟真实的故障情况。随后，将配置好的虚拟机和云任务提交给用户代理进行管理和调度，启动 CloudSim 仿真并监控其执行状态。仿真结束后，收集虚拟机的性能指标数据和云任务的执行状态，分析故障对虚拟机性能的影响，以评估和优化故障检测模型。

本实验在高性能工作站上进行。工作站配备了多核处理器(Intel Core i7)、32 GB 内存、500 GB 的 SSD，NVIDIA GPU (GTX 1080)用于加速深度学习模型的训练。

实验软件环境包括 Linux 操作系统(Ubuntu 20.04 LTS)，Java Development Kit (JDK) 8，Python 3.7，Anaconda 等。主要开发和仿真工具包括 CloudSim、Eclipse、Jupyter Notebook，深度学习框架采用 TensorFlow，NumPy、Pandas 用于数据处理，scikit-learn 用于模型评估和性能指标计算。

为了全面评估虚拟机故障检测算法在不同参数环境下的性能表现，我们设计了一系列实验，针对虚拟机性能指标数据的主要参数进行测试和分析，主要实验参数包括虚拟机数量、时间步长、特征维度、和故障注入比例。

4.2. 实验性能指标

在实验中，我们采用多种性能指标来评估虚拟机故障检测模型的有效性和准确性。主要性能指标包括准确率(Accuracy)、召回率(Recall)、精确率(Precision)和 F1 分数(F1-Score)。

1) 准确率(Accuracy)

准确率是指模型预测正确的样本数占总样本数的比例，计算公式为：

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{FP} + \text{FN} + \text{TP} + \text{TN}) \tag{15}$$

其中，TP 为真正例，TN 为真负例，FP 为假正例，FN 为假负例。准确率反映了模型整体的分类能力，但在样本类别不平衡的情况下，单独使用准确率可能会导致误导。

2) 召回率(Recall)

召回率是指在所有实际为正例的样本中，模型正确识别出的正例样本数的比例，计算公式为：

$$\text{Recall} = \text{TP} / (\text{FN} + \text{TP}) \tag{16}$$

召回率反映了模型对正类样本的敏感性，是衡量模型漏检率的重要指标。在故障检测任务中，高召回率意味着模型能够识别出大多数故障样本，降低漏报风险。

3) 精确率(Precision)

精确率是指在所有模型预测为正例的样本中，实际为正例的样本数的比例，计算公式为：

$$\text{Precision} = \text{TP} / (\text{FP} + \text{TP}) \tag{17}$$

精确率反映了模型预测结果的可靠性，是衡量模型误报率的重要指标。在故障检测任务中，高精确率意味着模型的故障报警较为准确，减少了误报。

4) F1 分数(F1-Score)

F1 分数是精确率和召回率的调和平均数，综合考虑了模型的精确率和召回率，计算公式为：

$$\text{F1-Score} = 2 \times \text{Precision} \times \text{Recall} / (\text{Precision} + \text{Recall}) \tag{18}$$

F1 分数在样本类别不平衡的情况下尤为重要，能够提供对模型性能的更全面评价。

4.3. 实验对比算法

在本实验中，我们对比了多种故障检测算法，以评估基于注意力机制的 CNN-LSTM 模型在虚拟机故障检测任务中的优越性。对比算法包括随机森林[28]、梯度提升树[29]和长短期记忆网络算法[30]。

随机森林是一种集成学习算法，通过构建多个决策树，并结合其预测结果进行分类或回归。在虚拟机故障检测任务中，随机森林可以通过学习虚拟机性能指标的特征，建立故障预测模型，实现对故障的高效检测。随机森林具有较高的分类精度和稳定性，能够处理大规模数据，但训练和预测时间较长，模型解释性较差。

梯度提升树(Gradient Boosting Decision Trees, GBDT)是一种集成学习算法，通过逐步构建一系列决策树，并优化每一步的误差，实现高效的分类或回归。GBDT 在处理复杂的非线性关系和特征交互方面表现优越。在虚拟机故障检测任务中，GBDT 可以通过学习虚拟机性能指标的特征，逐步优化故障预测模型，实现对故障的高效检测。GBDT 具有较高的分类精度和稳定性，适用于大规模数据，但训练时间较长，参数调优复杂。

长短期记忆网络(LSTM)是一种特殊的递归神经网络(RNN)，通过引入记忆单元和门机制，能够有效捕捉和学习长时间序列数据的依赖关系。在虚拟机故障检测任务中，LSTM 可以通过学习虚拟机性能指标的时间序列特征，建立故障预测模型，实现对故障的高效检测。LSTM 能够有效处理时间序列数据，具有较高的预测精度，但模型训练时间较长，计算资源需求较高。

4.4. 不同虚拟机数量算法性能对比

该实验评估基于注意力机制的 CNN-LSTM 模型与其他对比算法(随机森林、梯度提升树、LSTM)在不同虚拟机数量下的故障检测性能，以验证 CNN-LSTM 模型在虚拟机数量变化下的优越性和鲁棒性。

我们使用数据生成器函数生成不同虚拟机数量(100, 500, 1000, 2000)的虚拟机性能指标数据(时间步长固定为 60，特征维度固定为 5，噪声水平固定为 0.2，故障注入比例固定为 20%)。在数据进行了预处理后，使用基于注意力机制的 CNN-LSTM 模型、随机森林、梯度提升树和 LSTM 模型分别进行训练和预测，并记录每次实验的训练时间和测试时间。实验结果如表 1 所示。

Table 1. Performance comparisons of algorithms under different VM numbers
表 1. 算法在不同虚拟机数量时性能对比

Algorithm	Number of VMs	Accuracy	Recall	Precision	F1-Score
Random Forest	100	0.82	0.8	0.81	0.8
GBDT	100	0.83	0.81	0.82	0.81
LSTM	100	0.86	0.84	0.85	0.84
CNN-LSTM	100	0.88	0.86	0.87	0.86
Random Forest	500	0.85	0.83	0.84	0.83
GBDT	500	0.86	0.84	0.85	0.84
LSTM	500	0.9	0.88	0.89	0.88
CNN-LSTM	500	0.92	0.9	0.91	0.9
Random Forest	1000	0.87	0.85	0.86	0.85
GBDT	1000	0.88	0.86	0.87	0.86
LSTM	1000	0.92	0.91	0.92	0.91
CNN-LSTM	1000	0.94	0.93	0.94	0.93
Random Forest	2000	0.89	0.87	0.88	0.87
GBDT	2000	0.9	0.88	0.89	0.88
LSTM	2000	0.94	0.93	0.94	0.93
CNN-LSTM	2000	0.96	0.95	0.96	0.95

从表 1 可以看出，随着虚拟机数量的增加，随机森林的性能指标(准确率、召回率、精确率、F1 分数、AUC-ROC)逐步提高，但其性能提升幅度相对有限。在处理大规模数据时，训练和预测时间较长，模型解释性较差。梯度提升树在不同虚拟机数量下表现稳定，性能指标较为优秀，特别是在处理复杂的非线性关系和特征交互方面。然而，GBDT 的训练时间较长，参数调优复杂。长短期记忆网络(LSTM)在捕捉时间序列数据的长时间依赖关系方面表现出色，随着虚拟机数量的增加，其性能指标显著提升。LSTM 模型训练时间较长，计算资源需求较高。基于注意力机制的 CNN-LSTM 模型在所有虚拟机数量下均表现出色，其性能指标明显优于其他对比算法，特别是在处理复杂和大规模数据时，表现尤为突出。通过引入注意力机制，CNN-LSTM 能够更好地关注故障相关的关键特征，提高故障检测的准确性和效率。

4.5. 不同时间步长算法性能对比

该实验评估基于注意力机制的 CNN-LSTM 模型与其他对比算法(随机森林、梯度提升树、LSTM)在不同时间步长下的故障检测性能，以验证 CNN-LSTM 模型在时间步长变化下的优越性和鲁棒性。

我们使用数据生成器函数生成不同时间步长(参数范围：30、60、90、120)的虚拟机性能指标数据(虚拟机数量固定为 1000，特征维度固定为 5，噪声水平固定为 0.2，故障注入比例固定为 20%)。对生成的数据进行标准化处理，并划分训练集和测试集。实验使用基于注意力机制的 CNN-LSTM 模型、随机森林、梯度提升树和 LSTM 模型分别进行训练和预测，并记录每次实验的训练时间和测试时间。实验结果如表 2 所示。

Table 2. Performance comparisons of algorithms under different time length parameters
表 2. 算法在不同时长下性能对比

Algorithm	Time Steps	Accuracy	Recall	Precision	F1-Score
Random Forest	30	0.82	0.8	0.81	0.8
GBDT	30	0.83	0.81	0.82	0.81
LSTM	30	0.86	0.84	0.85	0.84
CNN-LSTM	30	0.88	0.86	0.87	0.86
Random Forest	60	0.85	0.83	0.84	0.83
GBDT	60	0.86	0.84	0.85	0.84
LSTM	60	0.9	0.88	0.89	0.88
CNN-LSTM	60	0.92	0.9	0.91	0.9
Random Forest	90	0.87	0.85	0.86	0.85
GBDT	90	0.88	0.86	0.87	0.86
LSTM	90	0.92	0.91	0.92	0.91
CNN-LSTM	90	0.94	0.93	0.94	0.93
Random Forest	120	0.89	0.87	0.88	0.87
GBDT	120	0.9	0.88	0.89	0.88
LSTM	120	0.94	0.93	0.94	0.93
CNN-LSTM	120	0.96	0.95	0.96	0.95

从表 2 可以看出，随着时间步长的增加，随机森林的性能指标(准确率、召回率、精确率、F1 分数、AUC-ROC)逐步提高，但其性能提升幅度相对有限。在处理长时间序列数据时，训练和预测时间较长，模

型解释性较差。GBDT 在不同时间步长下表现稳定，性能指标较为优秀，特别是在处理复杂的非线性关系和特征交互方面。然而，GBDT 的训练时间较长，参数调优复杂。LSTM 在捕捉时间序列数据的长时间依赖关系方面表现出色，随着时间步长的增加，其性能指标显著提升。LSTM 模型训练时间较长，计算资源需求较高。CNN-LSTM 模型在所有时间步长下均表现出色，其性能指标明显优于其他对比算法，特别是在处理长时间序列数据时，表现尤为突出。通过引入注意力机制，CNN-LSTM 能够更好地关注故障相关的关键特征，提高故障检测的准确性和效率。

4.6. 不同特征维度算法性能对比

该实验主要评估基于注意力机制的 CNN-LSTM 模型与其他对比算法(随机森林、梯度提升树、LSTM)在不同故障特征维度下的故障检测性能。实验使用数据生成器函数生成不同特征维度(参数范围：3、5、10、20)的虚拟机性能指标数据(虚拟机数量固定为 1000，时间步长固定为 60，噪声水平固定为 0.2，故障注入比例固定为 20%)。对生成的数据进行标准化处理，并划分训练集和测试集后，训练上述四种算法，实验结果如表 3 所示。

Table 3. Performance comparisons of algorithms under different length feature parameters
表 3. 算法在不同特征维度下性能对比

Algorithm	Feature Dimensions	Accuracy	Recall	Precision	F1-Score
Random Forest	3	0.8	0.78	0.79	0.78
GBDT	3	0.82	0.8	0.81	0.8
LSTM	3	0.85	0.83	0.84	0.83
CNN-LSTM	3	0.87	0.85	0.86	0.85
Random Forest	5	0.85	0.83	0.84	0.83
GBDT	5	0.86	0.84	0.85	0.84
LSTM	5	0.9	0.88	0.89	0.88
CNN-LSTM	5	0.92	0.9	0.91	0.9
Random Forest	10	0.87	0.85	0.86	0.85
GBDT	10	0.88	0.86	0.87	0.86
LSTM	10	0.92	0.91	0.92	0.91
CNN-LSTM	10	0.94	0.93	0.94	0.93
Random Forest	20	0.89	0.87	0.88	0.87
GBDT	20	0.9	0.88	0.89	0.88
LSTM	20	0.94	0.93	0.94	0.93
CNN-LSTM	20	0.96	0.95	0.96	0.95

从表 3 可以看出，随着特征维度的增加，随机森林的性能指标(准确率、召回率、精确率、F1 分数、AUC-ROC)逐步提高，但其性能提升幅度相对有限。GBDT 在不同特征维度下表现稳定，性能指标较为优秀，特别是在处理复杂的非线性关系和特征交互方面。LSTM 在捕捉时间序列数据的长时间依赖关系方面表现出色，随着特征维度的增加，其性能指标显著提升。CNN-LSTM 模型在所有特征维度下均表现出色，其性能指标明显优于其他对比算法，特别是在处理高维数据时，表现尤为突出。通过引入注意力机制，CNN-LSTM 能够更好地关注故障相关的关键特征，提高故障检测的准确性和效率。

4.7. 不同故障注入比例算法性能对比

该实验评估基于注意力机制的 CNN-LSTM 模型与其他对比算法(随机森林、梯度提升树、LSTM)在不同故障注入比例下的故障检测性能，以验证 CNN-LSTM 模型在故障注入比例变化下的优越性和鲁棒性。实验使用数据生成器函数生成不同故障注入比例(参数范围：10%、20%、30%、40%)的虚拟机性能指标数据(虚拟机数量固定为 1000，时间步长固定为 60，特征维度固定为 5，噪声水平固定为 0.2)。实验结果如表 4 所示。

Table 4. Performance comparisons of algorithms under different percentage of failure parameters
表 4. 算法在不同故障注入比例下性能对比

Algorithm	Percentage of failure	Accuracy	Recall	Precision	F1-Score
Random Forest	10%	0.81	0.79	0.8	0.79
GBDT	10%	0.83	0.81	0.82	0.81
LSTM	10%	0.87	0.85	0.86	0.85
CNN-LSTM	10%	0.89	0.87	0.88	0.87
Random Forest	20%	0.85	0.83	0.84	0.83
GBDT	20%	0.86	0.84	0.85	0.84
LSTM	20%	0.9	0.88	0.89	0.88
CNN-LSTM	20%	0.92	0.9	0.91	0.9
Random Forest	30%	0.87	0.85	0.86	0.85
GBDT	30%	0.88	0.86	0.87	0.86
LSTM	30%	0.92	0.91	0.92	0.91
CNN-LSTM	30%	0.94	0.93	0.94	0.93
Random Forest	40%	0.89	0.87	0.88	0.87
GBDT	40%	0.9	0.88	0.89	0.88
LSTM	40%	0.94	0.93	0.94	0.93
CNN-LSTM	40%	0.96	0.95	0.96	0.95

从表 4 可以看出，随着故障比例的增加，随机森林和梯度提升树的性能有所提升，但提升幅度较有限，主要在于它们在处理高比例故障数据时的解释性和效率较低。LSTM 在捕捉时间序列特征方面表现出色，性能显著提高，体现了其在处理复杂依赖关系方面的优势。CNN-LSTM 模型的性能提升最为显著，表现出在处理高比例故障数据时的强大能力，尤其是在故障特征明显且数据均衡的情况下，其鲁棒性和整体故障检测性能得到了充分发挥。

此外，故障比例的增加使得数据分布更加均衡，特征更加明显，从而减少了类别不平衡问题，提高了模型的召回率、精确率和整体检测性能。同时，高比例的故障样本增强了模型的鲁棒性，使其在处理不同场景和数据集时表现更加优异。因此，故障比例越大，算法的性能通常会越高。

5. 总结

本文提出了一种基于注意力机制的 CNN-LSTM 虚拟机故障检测算法，通过结合卷积神经网络(CNN)和长短期记忆网络(LSTM)，并引入注意力机制，设计了一种高效的虚拟机故障检测算法。CNN 负责提取局部特征，LSTM 捕捉时间序列依赖关系，而注意力机制进一步增强了模型对故障相关关键特征的关注。

在不同实验条件下，全面评估了 CNN-LSTM 模型的性能。结果表明，CNN-LSTM 模型在处理大规模数据和复杂环境时，检测准确率、召回率、精确率和 F1 分数均优于其他对比算法，表现出较高的鲁棒性和适用性。

未来工作将致力于进一步优化 CNN-LSTM 模型的结构和参数，以提高其效率和准确性，并探索多源数据融合(如网络日志、性能指标、用户行为数据等)以提升综合性能。此外，还将致力于在真实云计算环境中进行部署和测试，验证模型的实用性和稳定性，从而为云计算平台的稳定运行提供更强有力的保障。

参考文献

[1] 刘畅. 云环境下虚拟机异常的多属性分析[D]: [硕士学位论文]. 大连: 大连理工大学, 2014.

[2] 翟嘉琪, 杨希祥, 程玉强, 等. 机器学习在故障检测与诊断领域应用综述[J]. 计算机测量与控制, 2021, 29(3): 1-9.

[3] Vaswani, A., Shazeer, N., Parmar, N., *et al*. (2017) Attention Is All You Need. *Proceedings of the* 31*st International Conference on Neural Information Processing Systems*, Long Beach, 4-9 December 2017, 6000-6006.

[4] Massie, M.L., Chun, B.N. and Culler, D.E. (2004) The Ganglia Distributed Monitoring System: Design, Implementation, and Experience. *Parallel Computing*, **30**, 817-840. https://doi.org/10.1016/j.parco.2004.04.001

[5] Katsaros, G., Kübert, R. and Gallizo, G. (2011) Building a Service-Oriented Monitoring Framework with REST and Nagios. 2011 *IEEE International Conference on Services Computing*, Washington DC, 4-9 July 2011, 426-431. https://doi.org/10.1109/scc.2011.53

[6] 彭红星, 陈祥光, 徐巍, 等. 多变量过程传感器故障检测的 SVM 方法[J]. 北京理工大学学报, 2008, 28(8): 727-731.

[7] 高运广, 王仕成, 刘志国, 等. 一种基于 LS-SVM 的联邦滤波故障检测方法[J]. 控制与决策, 2011, 26(9): 1433-1435.

[8] Yin, Z. and Hou, J. (2016) Recent Advances on SVM Based Fault Diagnosis and Process Monitoring in Complicated Industrial Processes. *Neurocomputing*, **174**, 643-650. https://doi.org/10.1016/j.neucom.2015.09.081

[9] Widodo, A. and Yang, B. (2007) Support Vector Machine in Machine Condition Monitoring and Fault Diagnosis. *Mechanical Systems and Signal Processing*, **21**, 2560-2574. https://doi.org/10.1016/j.ymssp.2006.12.007

[10] Sun, W., Chen, J. and Li, J. (2007) Decision Tree and PCA-Based Fault Diagnosis of Rotating Machinery. *Mechanical Systems and Signal Processing*, **21**, 1300-1317. https://doi.org/10.1016/j.ymssp.2006.06.010

[11] Abdallah, I., Dertimanis, V., Mylonas, H., Tatsis, K., Chatzi, E., Dervili, N., *et al*. (2018) Fault Diagnosis of Wind Turbine Structures Using Decision Tree Learning Algorithms with Big Data. In: Haugen, S., *et al*., Eds., *Safety and Reliability—Safe Societies in a Changing World*, CRC Press, Boca Raton, 3053-3061. https://doi.org/10.1201/9781351174664-382

[12] Sridharan, N.V. and Sugumaran, V. (2021) Visual Fault Detection in Photovoltaic Modules Using Decision Tree Algorithms with Deep Learning Features. *Energy Sources, Part A*: *Recovery, Utilization, and Environmental Effects*. https://doi.org/10.1080/15567036.2021.2020379

[13] 孙志鹏, 孙志龙, 魏建. 基于决策树支持向量机算法的电力变压器故障诊断研究[J]. 电气工程学报, 2020, 14(4): 42-45.

[14] 庞梦洋, 索中英, 郑万泽. 基于 RS-CART 决策树的航空发动机小样本故障诊断[J]. 航空动力学报, 2020, 35(7): 1559-1568.

[15] 陈剑, 蔡坤奇, 陶善勇, 等. 基于 IITD 模糊熵与随机森林的滚动轴承故障诊断方法[J]. 计量学报, 2021, 42(6): 774-779.

[16] 常梦容, 王海瑞, 肖杨. mRMR 特征筛选和随机森林的故障诊断方法研究[J]. 电子测量与仪器学报, 2022, 36(3): 175-183.

[17] Cao, Y., Ji, Y., Sun, Y. and Su, S. (2023) The Fault Diagnosis of a Switch Machine Based on Deep Random Forest Fusion. *IEEE Intelligent Transportation Systems Magazine*, **15**, 437-452. https://doi.org/10.1109/mits.2022.3174238

[18] Sun, Y., Zhang, H., Zhao, T., Zou, Z., Shen, B. and Yang, L. (2020) A New Convolutional Neural Network with Random Forest Method for Hydrogen Sensor Fault Diagnosis. *IEEE Access*, **8**, 85421-85430. https://doi.org/10.1109/access.2020.2992231

[19] Jiao, J., Zhao, M., Lin, J. and Liang, K. (2020) A Comprehensive Review on Convolutional Neural Network in Machine Fault Diagnosis. *Neurocomputing*, **417**, 36-63. https://doi.org/10.1016/j.neucom.2020.07.088

[20] Lv, H., Chen, J., Pan, T., Zhang, T., Feng, Y. and Liu, S. (2022) Attention Mechanism in Intelligent Fault Diagnosis of Machinery: A Review of Technique and Application. *Measurement*, **199**, Article ID: 111594. https://doi.org/10.1016/j.measurement.2022.111594

[21] Mutegeki, R. and Han, D.S. (2020) A CNN-LSTM Approach to Human Activity Recognition. 2020 *International Conference on Artificial Intelligence in Information and Communication* (*ICAIIC*), Fukuoka, 19-21 February 2020, 362-366. https://doi.org/10.1109/icaiic48513.2020.9065078

[22] Leka, H.L., Fengli, Z., Kenea, A.T., Tegene, A.T., Atandoh, P. and Hundera, N.W. (2021) A Hybrid CNN-LSTM Model for Virtual Machine Workload Forecasting in Cloud Data Center. 2021 18*th International Computer Conference on Wavelet Active Media Technology and Information Processing* (*ICCWAMTIP*), Chengdu, 17-19 December 2021, 474-478. https://doi.org/10.1109/iccwamtip53232.2021.9674067

[23] He, N., Yang, S., Li, F., Trajanovski, S., Zhu, L., Wang, Y., *et al*. (2023) Leveraging Deep Reinforcement Learning with Attention Mechanism for Virtual Network Function Placement and Routing. *IEEE Transactions on Parallel and Distributed Systems*, **34**, 1186-1201. https://doi.org/10.1109/tpds.2023.3240404

[24] Li, S., Zhang, S., Chen, L., Chen, H., Liu, X. and Lin, S. (2020) An Attention Based Deep Reinforcement Learning Method for Virtual Network Function Placement. 2020 *IEEE* 6*th International Conference on Computer and Communications* (*ICCC*), Chengdu, 11-14 December 2020, 1005-1009. https://doi.org/10.1109/iccc51575.2020.9345041

[25] Dogani, J., Khunjush, F., Mahmoudi, M.R. and Seydali, M. (2022) Multivariate Workload and Resource Prediction in Cloud Computing Using CNN and GRU by Attention Mechanism. *The Journal of Supercomputing*, **79**, 3437-3470. https://doi.org/10.1007/s11227-022-04782-z

[26] Chen, L., Zhang, W. and Ye, H. (2022) Accurate Workload Prediction for Edge Data Centers: Savitzky-Golay Filter, CNN and BiLSTM with Attention Mechanism. *Applied Intelligence*, **52**, 13027-13042. https://doi.org/10.1007/s10489-021-03110-x

[27] Sundas, A. and Panda, S.N. (2020) An Introduction of Cloudsim Simulation Tool for Modelling and Scheduling. 2020 *International Conference on Emerging Smart Computing and Informatics* (*ESCI*), Pune, 12-14 March 2020, 263-268. https://doi.org/10.1109/esci48226.2020.9167549

[28] Schneider, S., Satheeschandran, N.P., Peuster, M. and Karl, H. (2020) Machine Learning for Dynamic Resource Allocation in Network Function Virtualization. 2020 6*th IEEE Conference on Network Softwarization* (*NetSoft*), Ghent, 29 June-3 July 2020, 122-130. https://doi.org/10.1109/netsoft48620.2020.9165348

[29] Alfarizi, M.G., Vatn, J. and Yin, S. (2023) An Extreme Gradient Boosting Aided Fault Diagnosis Approach: A Case Study of Fuse Test Bench. *IEEE Transactions on Artificial Intelligence*, **4**, 661-668. https://doi.org/10.1109/tai.2022.3165137

[30] Wang, Y., Dong, X., Wang, L., Chen, W. and Zhang, X. (2022) Optimizing Small-Sample Disk Fault Detection Based on LSTM-GAN Model. *ACM Transactions on Architecture and Code Optimization*, **19**, Article No. 13. https://doi.org/10.1145/3500917