Journal of Computer Applications
计算机应用,2021,41(6):1701 - 1708
ISSN 1001-9081
CODEN JYIIDU
2021-06-10
http://www.joca.cn

文章编号:1001-9081(2021)06-1701-08
DOI:10.11772/j.issn.1001-9081.2020091383

云工作流中基于多任务时序卷积网络的异常检测方法

姚　杰[1],程春玲[1],韩　静[2],刘　峥[1*]
(1.南京邮电大学 计算机学院,南京 210023;　2.中兴通讯股份有限公司 上海研发中心,上海 201203)
(*通信作者电子邮箱zliu@njupt.edu.cn)

摘　要:云计算数据中心在日常部署和运行过程中产生的大量日志可以帮助系统运维人员进行异常分析。路径异常和时延异常是云工作流中常见的异常。针对传统的异常检测方法分别对两种异常检测任务训练相应的学习模型,而忽略了两种异常检测任务之间的关联性,导致异常检测准确率下降的问题,提出了一种基于多任务时序卷积网络的日志异常检测方法。首先,基于日志流的事件模板,生成事件序列和时间序列;然后,训练基于多任务时序卷积网络的深度学习模型,该模型通过共享时序卷积网络中的浅层部分来从系统正常执行的流程中并行地学习事件和时间特征;最后,对云计算工作流中的异常进行分析,并设计了相关异常检测逻辑。在OpenStack数据集上的实验结果表明,与日志异常检测的领先算法DeepLog和基于主成分分析(PCA)的方法比较,所提方法的异常检测准确率至少提升了7.7个百分点。

关键词:异常检测;日志分析;时序卷积网络;多任务学习;云工作流
中图分类号:TP183　　**文献标志码**:A

Anomaly detection method based on multi-task temporal convolutional network in cloud workflow

YAO Jie[1], CHENG Chunling[1], HAN Jing[2], LIU Zheng[1*]
(1. *School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing Jiangsu* 210023, *China*;
2. *Shanghai Research and Development Center, Zhongxing Telecommunication Equipment Corporation, Shanghai* 201203, *China*)

Abstract: Numerous logs generated during the daily deployment and operation process in cloud computing platforms help system administrators perform anomaly detection. Common anomalies in cloud workflow include pathway anomalies and time delay anomalies. Traditional anomaly detection methods train the learning models corresponding to the two kinds of anomaly detection tasks respectively and ignore the correlation between these two tasks, which leads to the decline of the accuracy of anomaly detection. In order to solve the problems, an anomaly detection method based on multi-task temporal convolutional network was proposed. Firstly, the event sequence and time sequence were generated based on the event templates of log stream. Then, the deep learning model based on the multi-task temporal convolutional network was trained. In the model, the event and the time characteristics were learnt in parallel from the normal system execution processes by sharing the shallow layers of the temporal convolutional network. Finally, the anomalies in the cloud computing workflow were analyzed, and the related anomaly detection logic was designed. Experimental results on the OpenStack dataset demonstrate that, the proposed method improves the anomaly detection accuracy at least by 7.7 percentage points compared to the state-of-art log anomaly detection algorithm DeepLog and the method based on Principal Component Analysis (PCA).

Key words: anomaly detection; log analysis; temporal convolutional network; multi-task learning; cloud workflow

0　引言

云计算平台使用分布式计算、虚拟化等技术整合了大量的计算设备及基础设施,并通过网络为用户提供计算、数据存储等资源和相关服务,因其高效、廉价和易维护等特点被广泛应用。随着云服务市场的快速发展,云计算系统的安全、可靠运行受到了日益严峻的挑战[1]。云计算系统中存储着大量的数据,其系统规模庞大、结构复杂,系统内通常包含多个组件,相较于一般的计算机软件系统更容易存在系统漏洞或遭受到攻击。因此异常检测[2]是维护云计算系统安全可靠运行的必不可少的步骤之一。

为了了解系统的运行过程,系统中有各种检测程序监测系统在运行过程中的状态,采集监测数据。在基于监测数据的异常检测[3]中,监测数据主要为系统运行过程中采集的性能数据。Nguyen等[4]设计了一种在线异常定位系统,依靠采

收稿日期:2020-09-07;**修回日期**:2020-12-17;**录用日期**:2020-12-22。　**基金项目**:国家重点研发计划项目(2018YFB1003702);中兴通讯股份有限公司产学研合作项目;南京邮电大学国家自然科学基金孵化项目(NY219084)。

作者简介:姚杰(1996—),男,江苏盐城人,硕士研究生,主要研究方向:日志分析、深度学习;　程春玲(1972—),女,陕西西安人,教授,博士,主要研究方向:数据管理、资源管理和优化;　韩静(1978—),女,上海人,高级工程师,硕士,主要研究方向:机器学习、事件挖掘;刘峥(1980—),男,江苏南京人,讲师,博士,CCF会员,主要研究方向:网络数据挖掘。

集的性能数据建立系统正常执行下的波动模型，并根据找出的异常数据定位异常组件。Wang等[5]不同于使用阈值确定异常的方法，提出了用熵来计算性能数据的集中度和分散度，并使用数据分析方法区分正常熵和异常熵以检测系统异常。以上基于监测数据方法的不足之处在于监测数据的获取需要依靠相关数据采集程序，同时监测数据难以定位系统异常发生的位置。

基于日志的异常检测利用系统中普遍存在的日志数据，日志是进行系统异常检测的重要信息来源，相较于监测数据，日志数据更容易获取，而且也能够确定异常的位置。传统的基于日志的异常检测方法从系统日志中挖掘出系统正常运行时的特征以用于异常检测[6]。Xu等[7]通过日志解析从日志中提取出结构化信息，用定长的窗口划分日志序列，统计窗口内每个事件模板出现的次数作为一个特征向量，由此构造出特征矩阵，然后使用主成分分析(Principal Component Analysis，PCA)在数量上明显有异于其他集合的异常集合，最后通过统计检验确定异常的日志序列。随着数据的变化，该方法异常检测结果的准确性有很大的波动。基于监督学习的方法在数据处理上采用与PCA相类似的方式提取特征矩阵，然后使用决策树[8]、支持向量机[9]等方法进行建模。He等[10]详细比较了现有的各种使用传统机器学习的异常检测方法，这类方法在特定的云计算系统中能够取得较好的异常检测效果，但是缺乏对异常类型的判断。基于日志的事务流图方法构建系统正常执行流程下的事务流图，以用于检验系统的执行过程。Yu等[11]提出的ClouderSeer是一种专门为OpenStack平台而设计的异常检测算法，它通过多个正常运行下生成的日志流为每一个OpenStack虚拟机构建一个自动状态机，刻画了每个任务的工作流，在在线检测阶段自动识别交错的日志，检测可能存在的异常。基于日志的事务流图[12]能够为异常分析提供强有力的支持，但是由于事务流图难以描绘出整个系统的详细流程，所以其在异常检测方面的准确度较差。

近年来随着深度学习的快速发展，各种深度学习模型被广泛运用于系统的异常检测[13]中，Du等[14]提出的DeepLog利用自然语言处理的方式处理日志流，将日志条目视为遵循某些特定模式和语法规则的序列元素，使用长短期记忆(Long Short-Term Memory，LSTM)网络[15]构建循环神经网络模型，从日志流中挖掘出事件序列和日志参数序列分别训练对应的LSTM模型，用于检测系统异常。Brown等[16]运用自然语言处理中的词袋模型对原始日志进行处理，将处理后的日志流序列作为模型的输入，并在LSTM中加入了注意力机制，最后根据日志的异常得分来确定是否发生异常。Lu等[17]构建了由embedding层、三个一维卷积层、dropout层和池化层组成的卷积神经网络用来检测输入的事件序列中是否发生了异常。王易东等[18]在N-gram语言模型的基础上提出了一种基于门控循环单元网络(Gated Recurrent Unit，GRU)的日志异常检测模型，该模型对权重参数的各维度采用不同的学习速率进行更新[19]。任明等[20]提出了一种基于LSTM的异常检测方法，利用TF-IDF提取日志特征，将异常检测建模为二分类任务，该方法在Web系统日志中取得了较好效果。杨瑞朋等[21]用PReLU(Parametric Rectified Linear Unit)激活函数替换时序卷积网络(Temporal Convolutional Network，TCN)中的ReLU函数，针对全连接层易引起过拟合的问题，用自适应平均池化层替换全连接层。现有的基于深度学习的方法中普遍采用事件和日志参数各自训练相应的模型，而忽略了这些任务的关联关系。

系统中的路径异常和时延异常是较为常见的异常类型，路径异常由系统偏离正常执行路径造成，而请求或响应超时是造成系统时延异常的常见原因。综合上文中提到的问题，本文提出了一种基于多任务深度学习模型的异常检测框架。该框架选用日志数据，生成日志对应的事件模板及模型的训练和检测数据，从事件和时间这两个维度针对系统中存在的路径异常和时延异常进行检测。

本文的主要工作如下：实际的云平台系统中事件发生频繁，相邻事件的时间差过小，因此通过间隔采样的方式生成时延异常检测任务的数据，有效解决了短时延难以预测及检测的问题。此外根据事件预测任务和时延预测任务之间的关联性，提出了一种基于TCN的多任务深度学习模型，并设计了异常检测逻辑。在实际日志数据中的检测结果表明，本文的异常检测框架取得了较高的异常检测准确度。

1 云计算工作流的异常

云计算工作流中的异常主要包括时延异常和路径异常。路径异常是系统中常见的异常类型之一，基于日志的路径异常检测主要依靠从日志流中挖掘的事件序列，路径异常在日志流中具体表现为某一位置处的事件模板偏离了正常的路径，可能出现的路径异常类型有跳过下一条关键日志和不遵循关键日志。例如，正常执行下得到路径为：事件1→事件2→事件3→事件4，如果某次执行产生的日志流对应的路径变为：事件1→事件2→事件5→事件4，则在事件5的位置处可能发生了类型为不遵循关键日志的路径异常；如果对应的路径变为：事件1→事件2→事件4，则在事件4的位置处可能发生了类型为跳过下一条关键日志的路径异常。

由于系统程序的执行是一种严格的流程，且系统的日志输出语句一般书写在程序的固定位置，所以两条日志间的输出时间间隔应该服从一定的分布，受计算机硬件以及网络环境等因素的影响，该时间间隔在一个固定的区间内波动。因此，系统的时延异常在日志流中具体表现为两条日志的输出时间间隔不在正常的范围内，例如“Deploying os_pkg_repo begins.”和“Deploy os_pkg_repo successfully.”这两个事件之间的时间间隔应该在一定的时间范围内，如果超出了这个范围，就可能存在时延异常。

本文提出了基于日志的异常检测框架，可以检测系统中存在的路径异常和时延异常，该异常检测框架包含以下三个步骤：

1)事件模板及序列生成：通过日志解析[22]将半结构化的日志解析为结构化的事件模板，提取出每一条日志对应的事件ID和时间戳，从而将日志流转化为事件序列和时间序列。

2)多任务学习模型：构建并训练基于时序卷积网络的多任务学习模型，多任务学习通过共享模型参数学习两个任务的共同特征。

3)日志异常检测：针对系统中存在的路径异常和时延异常设计相应的异常检测方法，利用多任务学习模型进行日志

的异常检测。

2　事件模板及序列生成

2.1　事件模板生成

日志解析是在进行异常检测之前必不可少的工作,因为系统产生的原始日志是半结构化的文本信息,由日志关键字和变量参数组成,其中变量参数记录着如时间戳、IP地址这类随着系统运行而变化的信息。日志解析从原始日志中提取日志关键字,将变量用占位符代替(比如可以用*代替的变量参数),例如如下的OpenStack原始日志:

2019-04-29 14:05:03.993 INFO sqlalchemy. orm. mapper. Mapper (Com_Server|com_servers) _configure_property (msb_info_v2, Column) req-115af119-b549-4b0b-9056-fbf97a5d6009.

经过日志解析后得到的事件模板如下:sqlalchemy orm mapper Mapper (Com_Server|com_servers) (*Column) <req-param>。日志解析将半结构化的日志信息转化为结构化的事件模板,每个事件模板尽可能代表着一条日志打印语句。

云计算数据中心包含多个组件,首先生成每个组件的事件模板,本文采用的日志解析算法为基础签名生成(Basic Signature Generation, BSG)算法[23],BSG通过分析日志消息的长度和关键词,从而实时解析出日志消息所属事件和对应的消息签名,能够在保证解析速度的情况下解决现有日志解析算法需要真实消息签名辅助调参工作的问题。日志解析后得到事件模板集合,每一个事件模板用一个事件ID进行编号。由于BSG是一种流式解析算法,会造成解析质量的下降,解析算法会将本属于同一事件模板的日志,解析为不同类型的事件模板,影响了异常检测中对异常的判定,所以进一步用编辑距离计算事件模板 E_i 与其他的事件模板的相似度,然后对解析生成的事件模板采用层次聚类合并相似度较高的事件模板,生成新的事件模板,得到新的事件模板集合 $E=\{E_1,E_2,\cdots,E_m\}$。

2.2　序列生成

序列是由系统某次运行产生的原始日志条目中的参数或其对应的事件ID组成的,通过滑动窗口划分序列数据以生成模型的训练和检测数据,用事件模板匹配原始日志条目得到原始日志对应的事件ID,系统一次运行对应的事件序列 $S_E=\{e_1,e_2,\cdots,e_i,\cdots,e_n\}, e_i\in E$。

系统一次运行对应的时间序列由日志条目中的时间戳组成。对于时间序列 $S_T=\{t_1,t_2,\cdots,t_n\}$,普遍采用的处理方式为取两个相邻事件的时间差,但是系统事件发生频繁,相邻事件间的时间差过小。如图1所示为OpenStack某次正常部署下时延的分布情况,当 $k=1$,即取两个相邻事件的时间差时,多数时延小于0.001 s,不利于模型训练和预测以及时延异常的检测。因此本文采用间隔采样的方式生成新的时延序列,采样的间隔为 k,取事件序列中事件 e_i 发生时间与事件 e_{i-k} 发生时间的时间间隔,当 $i-k<0$ 时,采用zero padding生成时延序列 $S_{T'}=\{0,0,\cdots,t_i-t_{i-k},\cdots,t_n-t_{n-k}\}$。

如图1所示,当采用间隔采样后,相较于相邻采样,当 $k=5$ 时,处于0~0.001 s内的时延大幅减少,多数时延处于0.001~0.01 s的区间内。进一步扩大间隔采样值,即 $k=10$ 时,时延继续增大,位于0.01~0.04 s区间的数量明显增多。可通过间隔采样的方式减小短时延对异常检测带来的影响。

k 值的选取通过事件发生处时延集合的均值来确定,时间是与事件相关的,不同的事件流程可能导致不同的时延,假设每个事件 $E_i(i=1,2,\cdots,m)$ 在其发生位置处对应的时延集合 $T_{Ei}=\{t'_0,t'_1,\cdots\}$ 符合正态分布,其均值和标准差分别为 M_{Ei} 和 S_{Ei}。如图2所示的序列生成过程中,展现了事件 E_5 的时延集合生成。为了保证所有事件对应的时延处于一个较为合理的值,设定阈值 $t_{\text{threshold}}$,当 k 的取值为某个固定值时,应满足如下条件:

$$\min(M_{E1},M_{E2},\cdots,M_{Em})>t_{\text{threshold}} \tag{1}$$

本文取 $t_{\text{threshold}}=0.01$,其 $k=30$,最后使用长度为 h 的滑动窗口划分事件序列 S_E 和时延序列 $S_{T'}$。

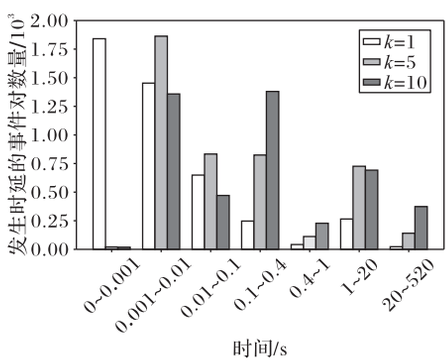


图1　不同 k 值下时延分布

Fig. 1　Time delay distribution with different k values

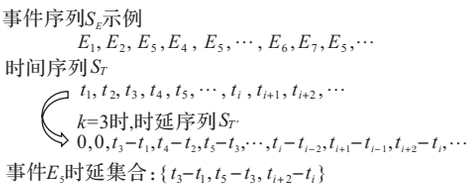


图2　序列生成示例

Fig. 2　Sequence generation example

3　基于时序卷积网络的多任务学习模型

3.1　基于时序卷积网络的单任务学习模型

循环神经网络因其能够有效保留历史信息,一直是序列建模的首选,但是因为循环神经网络需依次计算每一步状态,所以不适合并行。时序卷积网络是Bai等[24]在近年来使用卷积结构解决时序任务研究的基础上,提出的对时序任务有较好适用性的卷积网络结构。作为一种卷积结构的序列模型,TCN结构更加简单清晰并且能够并行化,相较于LSTM等循环神经网络结构,TCN可以通过堆叠卷积结构,捕捉到更加长远的依赖关系。

时序卷积网络的设计主要遵循以下两个原则:1)网络的输出与输入等长,卷积结构使用一维全卷积,使得隐藏层的输出与输入等长,采用zero padding来保持层与层之间的长度相同;2)网络不会遗漏任何历史信息,卷积层采用因果卷积[24]。

TCN使用了扩张卷积,扩张卷积可以有效地减小网络的深度并使模型具有长期记忆,此外,增大扩张系数或卷积核大小可以扩大感受野,感受野 $R_f=d*z$。对于输入的一维序列:

$\boldsymbol{x} \in \mathbf{R}^n$ 和卷积核 $f = \{0, 1, \cdots, z-1\}$，序列中某个元素 a 扩张卷积运算 F 定义如下：

$$F(a) = (\boldsymbol{x} * df)(a) = \sum_{i=0}^{z-1} f(i) \cdot x_{a-d \cdot i} \tag{2}$$

其中：d 是扩张系数；z 是卷积核大小。

随着输入历史序列长度的增加，TCN模型也在加深，为了避免TCN模型出现梯度弥散，引入了残差连接。如图3所示为残差块的结构，其中扩张系数 $d = 1$，卷积核大小 $z = 2$。输入长度为3的序列 $\boldsymbol{x} = (x_0, x_1, x_2)$，残差块的输出 O 如下：

$$O = Activation(\boldsymbol{x} + F(\boldsymbol{x})) \tag{3}$$

其中 $F(\boldsymbol{x})$ 为 $\boldsymbol{x}$ 经过的一系列计算。

本文将事件序列的路径预测任务建模为多分类问题，将时间序列上的时延预测任务建模为回归问题。对于输入的历史事件序列和时间序列，模型预测的是正常情况下的下一位置处应该发生的事件和时间，模型的训练数据为系统多次正常运行产生的日志数据。单任务学习模型基于TCN分别训练路径异常检测和时延异常检测对应的模型。

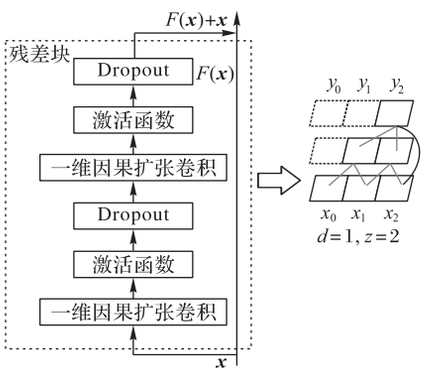


图3 残差块结构

Fig. 3 Structure of residual block

单任务的路径预测任务中，给定窗口长度 h，对于长度 h 的一维事件子序列 $w_e = \{e_{r-h}, \cdots, e_{r-2}, e_{r-1}\}$，采用常用的Word2Vec方法，经过词嵌入后将每个事件映射为一个固定长度 H 的一维向量，词嵌入层输出维度为 $h \times H$ 的特征 $\boldsymbol{h}_e$。$\boldsymbol{h}_e$ 为TCN的输入，TCN的输出通过一个激活函数为softmax的全连接层后，输出每个事件 $E_i \in E(i = 1, 2, \cdots, m)$ 在下个位置 r 处发生的概率分布 $P(e_r|w_e) = \{E_1: p_1, E_2: p_2, \cdots, E_n: p_n\}$。损失函数为交叉熵函数 L_1：

$$L_1 = \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{m} (y_i \log(p_i)) \tag{4}$$

其中：y_i 为事件 E_i 的标签；p_i 为事件 E_i 的概率。

单任务的时延预测任务中，TCN的输入为长度 h 的时间子序列 $w_t = \{t'_{r-h}, \cdots, t'_{r-2}, t'_{r-1}\}$，TCN的输出通过一个激活函数为linear的全连接层后，输出下个位置 r 处的时延大小 T'_r，损失函数为均方误差函数 L_2：

$$L_2 = \frac{1}{n} \sum_{j=1}^{n} (t'_r - T'_r) \tag{5}$$

其中 t'_r 为下个位置 r 处的时延实际值。

3.2 多任务学习模型

本文从事件和时间两个维度检测系统异常，即路径异常和时延异常。利用3.1节中提出的时序卷积网络，可以分别为路径异常和时延异常两个任务构建两个单独的学习模型，从而完成相应的异常检测。为路径异常或时延异常单独构建学习模型忽略了两种异常直接的关联关系，例如不同事件对于事件直接时延的影响，或是不同时延对于事件预测的判断，从而不能取得较好的异常检测效果。

多任务学习[25]通过学习任务之间的共有特征，提高了学习模型的表现。Long等[26]先通过共享卷积层，然后在特定任务层中设定张量的正态先验，学习多个任务之间的关系。Misra等[27]在并行的两个任务之间设计了十字绣单元，通过正则化模型参数距离的相似度，来约束各个任务模型参数的相似度，从而控制任务之间的共享程度。

本文考虑到日志类型的事件ID和日志发生时间的时间戳之间的相关性，提出利用多任务学习同时解决路径异常和时延异常的检测，设计了融合的多任务时序卷积网络，通过共享深度神经网络隐藏层的参数，不仅捕捉不同种类异常之间的关联关系，提升了异常的检测效果，还降低了模型的复杂度。更优的是，共享参数的多任务融合的时序卷积网络为不同种类的异常检测引入了噪声，提高了模型的泛化能力，进一步提高了异常的检测效果。

本文提出的多任务融合的时序卷积网络中包括两个异常检测的学习任务：路径预测任务和时延预测任务，模型在时序卷积网络的浅层部分共享路径预测任务和时延预测任务的隐藏层参数，再分叉成两个各自的特定任务层。基于TCN的多任务学习示例模型如图4(a)所示，该多任务学习模型主要由输入层、时序卷积网络(包括共享层和特定任务层)和输出层组成。时序卷积网络的详细示例如图4(b)所示，由共享层和任务层组成，共享层中扩张系数 d=1，2时残差块共享参数，任务层中扩张系数 $d = 4, 8, \cdots$ 时为两个任务各自的残差块。给定窗口长度 h，多任务模型路径预测任务和时延预测任务的输入同单任务学习模型。在时序卷积网络层中，将事件序列经过词嵌入层后输出的特征 $\boldsymbol{h}_e$ 和时间序列 w_t 进行特征联合(concatenate)后的特征 $\boldsymbol{X} = (x_0, x_1, \cdots, x_n)$ 输入共享层中，特征 $\boldsymbol{X}$ 的维度为 $h \times (H + 1)$，时序卷积网络的激活函数为ReLU函数，定义共享层中有 k_1 个残差块，$\boldsymbol{h}_{si}(i \leqslant k_i)$ 为特征 $\boldsymbol{X}$ 经过共享层中第 i 个残差块的输出，式(6)表示共享层第一个残差块的输出 $\boldsymbol{h}_{s1}$：

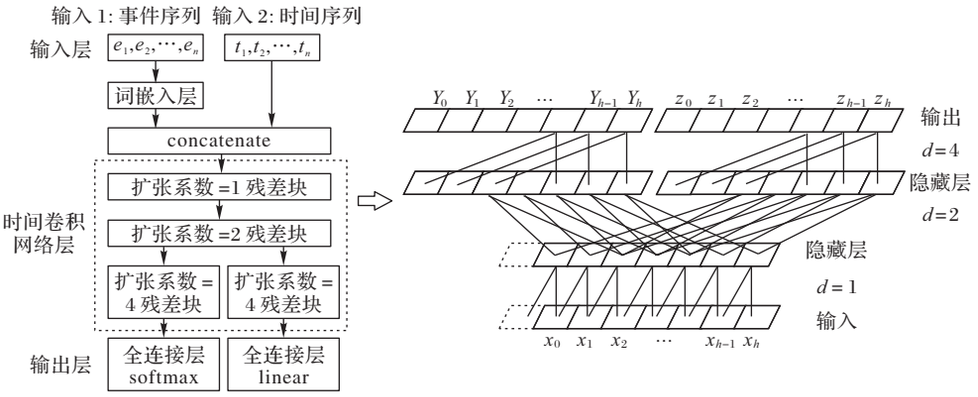
$$\boldsymbol{h}_{s1} = relu(\boldsymbol{X} + F(\boldsymbol{X})) = \max(0, \boldsymbol{X} + F(\boldsymbol{X})) \tag{6}$$

将共享层的输出 $\boldsymbol{h}_{sk_1}$ 输入两个任务各自的任务层中，定义任务层中事件预测任务和时延预测任务分别有 k_2 和 k_3 个残差块，任务层中事件预测任务和时间预测任务输出分别为 $\boldsymbol{h}_{tk_2} = (Y_1, Y_2, \cdots, Y_h)$ 和 $\boldsymbol{h}_{tk_3} = (z_1, z_2, \cdots, z_h)$。

$\boldsymbol{h}_{tk_2}$ 和 $\boldsymbol{h}_{tk_3}$ 通过一个全连接层后输出，多任务模型的路径预测任务和时延预测任务的输出同单任务模型，基于TCN的多任务学习模型的损失函数 L 如下：

$$L = \lambda L_1 + (1 - \lambda) L_2 \tag{7}$$

损失函数中的 λ 和 $1 - \lambda$ 分别为交叉熵损失函数和均方误差损失的权重。由于不同的任务之间的损失尺度不同，所以通过设置不同的权重来平衡各个任务的训练，以保证模型在所有的任务上都能取得较好的效果。



(a) 多任务学习示例模型　　(b) 时序卷积网络的详细示例

图4　基于TCN的多任务学习模型

Fig. 4　Multi-task learning model based on TCN

4　异常检测

4.1　路径异常检测

本文将路径预测任务建模为多分类任务,对于历史序列 $w_e=\{e_{r-h},\cdots,e_{r-2},e_{r-1}\}$ 和实际发生的事件 e_r,模型预测的是下一个位置 r 处每一个事件发生的概率,如果实际发生的事件 e_r 在模型预测的高概率事件中,则该位置处正常。本文的路径异常判断方法如下:设立概率阈值 $P_{\text{threshold}}$,如果 e_r 预测的概率大于该阈值,则正常;否则异常。

4.2　时延异常检测

将时延预测任务建模为回归任务,对于历史序列 $w_t=\{t'_{r-h},\cdots,t'_{r-2},t'_{r-1}\}$ 和实际的时延 t'_r,模型预测的是下一个位置 r 处的时延大小。本文认为事件和时间是相关的,不同的事件可能会导致不同的时间间隔。

具体示例如图5所示,在利用模型进行事件预测时,输入固定长度的事件序列{A,B,C,D}和其对应的时延序列,假设为{0.1 s,0.2 s,0.3 s,0.4 s},模型预测的高概率事件可能有多个,比如事件E和事件F都有可能发生,事件E发生时对应的时延大小为 t_1,事件F发生时对应的时延大小为 t_2,而模型预测的值为单一的固定值 t_3,因此单一地依靠模型的预测值来判断时延异常是不准确的。

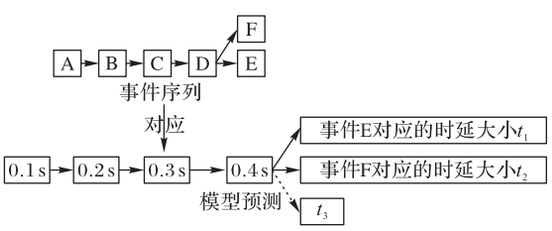


图5　模型的时延预测

Fig. 5　Time delay prediction of model

本文对于时延异常的检测主要分以下两步:1)将训练集中每个时间步的时延的实际值与模型的预测值的绝对误差拟合为高斯分布,可以得到正常情况下误差的均值 M 和标准差 σ,输入长度为 h 的时间子序列 $w_t=\{t'_{r-h},\cdots,t'_{r-2},t'_{r-1}\}$,如果实际的时延大小 t'_r 与模型的预测值 t_{pre} 在上述高斯分布的置信区间内,则是正常的;否则进行下一步。即:

$$M-\gamma_1\sigma<|t_r-t_{\text{pre}}|<M+\gamma_1\sigma;\ \gamma_1=1,2,\cdots,n \tag{8}$$

2)判断实际的时延大小是否处于该时间步的事件 e_r 时延的高斯分布的置信区间内,即:

$$M_{er}-\gamma_2 S_{er}<t_r<M_{er}+\gamma_2 S_{er};\ \gamma_2=1,2,\cdots,n \tag{9}$$

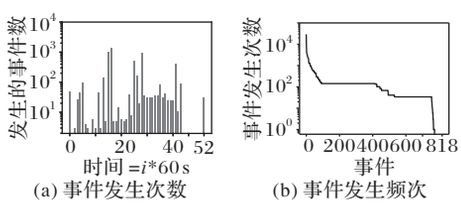
在本文的异常检测的流程中,时延异常检测依赖于路径异常检测,在路径检测正常后才进行时延异常检测,路径和时延只要有一个为异常,则认为当前位置发生了异常。

5　实验与结果分析

5.1　数据及配置

实验日志数据为OpenStack云平台日志数据集:OpenStack是一种基于PASS的云计算平台,从OpenStack收集的数据集包括虚拟机部署过程中cf-pdman组件和pdm-cli组件的生成日志,考虑到系统内的故障也有可能是跨组件或者跨服务的,单独对每个组件进行异常检测会遗漏这些异常。因此合并各个组件的日志文件,并按时间重新排序。数据集一共包括125次部署,其中包括16个虚拟机部署过程中的异常部署(包括缺失关键日志和跳过下一条日志等异常)。日志是由三台计算机用分布式的方式收集,这些部署的持续时间大约为3 d。每次部署的日志条目数量为747~109 493不等,所有部署共包括771 125条日志。日志解析后的事件模板经过层次聚类后共包含819个事件。将正常部署的日志数据经过数据处理后提取出事件和时间序列,经过定长的滑动窗口划分打乱后,按6:2:2的比例划分为训练集、验证集和检测集。本文将异常部署的日志数据也加入检测集以测试对于异常部署检测的效果。

如图6(a)所示为正常部署的日志数据中事件发生次数的分布,可以发现多数事件发生次数较低,少数事件以较高次数发生,这表明模型预测的发生概率较小的事件也有可能发生。如图6(b)所示为OpenStack某次正常部署过程中事件发生的频次,每隔60 s统计发生的事件数,可以发现事件发生的频繁程度和程序运行的过程相关。



(a) 事件发生次数　　(b) 事件发生频次

图6　正常部署中有关事件统计

Fig. 6　Statistics of related events in normal deployment

5.2 参数选择及评价指标

这里介绍本文的多任务TCN模型和异常检测过程中一些较为重要的参数设置。在模型参数方面，模型的输入序列长度为40，设置损失函数中的参数$\lambda = 0.7$，用以调节两个损失函数的尺度。受日志解析和较长日志参数的影响，尽管进行了层次聚类，还是有部分的事件模板是相似的。模型预测的高概率事件和实际发生的事件e_r不是同一个事件，但是两个事件对应的事件模板很相似，因此如果事件e_r不在预测的高概率事件中，则计算事件e_r和高概率事件的事件模板的相似度，并设立阈值$S_{\text{threshold}}$，如果大于$S_{\text{threshold}}$，认为这两个事件是一个事件，判断其为正常。模板相似度$S_{\text{threshold}}$要保证模型在检测出异常的同时，避免将正常检测为异常，由于事件模板经过了层次聚类，所以模板相似度$S_{\text{threshold}}$需要设置为较小的值，本文的实验中设置为0.6。

用于对比的方法如下：DeepLog是基于LSTM的日志异常检测模型，该模型利用从日志流中挖掘出事件序列和日志参数序列分别训练对应的LSTM模型，预测发生在系统中的路径异常和参数异常；PCA统计定长的窗口内每个事件模板出现的次数作为一个特征向量，由此构造出特征矩阵，使用PCA分析在数量上有异于其他集合的异常集合，通过统计检验确定异常的日志序列。本文以DeepLog为基线方法进行比较。

本文所采用的评价指标主要有如下几种：

1）准确率（Accuracy）表示被正确检测出的样本占总样本的比例，计算式为：

$$Accuracy = \frac{TP + TN}{P_s + N_s}$$

2）精度（Precision，P）表示实际为异常的样本占检测出异常的样本的比例，计算式为：

$$P = \frac{TP}{TP + FP}$$

3）召回率（Recall，R）表示正确检测出异常的样本占异常样本的比例，计算式为：

$$R = \frac{TP}{TP + FN}$$

4）F1值（F1_score）为精度和召回率的调和平均数，用于评价异常检测的整体性能：

$$F1\_score = \frac{2*P*R}{P + R}$$

其中：P_s为异常样本数；N_s为正常样本数；TP为被正确检测为异常的样本数；TN表示被正确检测为正常的样本数；FP表示被错误检测为异常的样本数；FN表示被错误检测为正常的样本数。

5.3 结果分析

如表1所示为OpenStack数据集中检测出的路径异常和时延异常示例，通过异常位置处实际输出的原始日志和模型的预测信息，可以大致分析出异常发生的原因。以pdm-cli组件中检测出的异常为例：表1中的第一条路径异常示例可以结合模型预测的高概率事件模板，进一步分析得出其异常原因为相关部署初始化问题；在表1中的第三条时延异常示例中，模型预测其时间间隔在正常情况下应为200 s左右，而实际的时间间隔为260 s，结合实际输出的日志可以进一步发现导致时延异常的原因为http请求失败。通过对比正常执行和异常位置处的相关信息可以进一步深入分析导致异常的原因。

表1 OpenStack数据集中检测出的异常示例

Tab. 1 Examples of anomalies detected in OpenStack dataset

组件	异常类型	实际输出日志	模型预测事件模板	实际事件模板
pdm-cli	路径异常	2019-04-29 15:40:03,844-INFO-__main__-1430-pdm-cli task list	-playground migration-1430-Deploy initial component:*successful!	-__main__-1430-pdm-cli*list
cf-pdman	路径异常	2019-04-29 13:25:15.330 CRITICAL cloudframe NoSuchOptError: no such option drNZZclZcsE in group [DEFAULT]	“sqlalchemy orm mapper Mapper (Server|servers)_configure_property(Node, RelationshipProperty) <req-param>”	cloudframe NoSuchOptError: no such option drNZZclZcsE in group [DEFAULT]

组件	异常类型	实际输出日志	模型预测时间间隔/s	实际时间间隔/s
pdm-cli	时延异常	2019-04-29 15:20:17,119-DEBUG-pdmcli. common. httpclient-1422-Http request failed. (timed out)	198.925 1	260.424 0
pdm-cli	时延异常	2019-04-29 15:31:10,063-DEBUG-pdmcli. common. httpclient-1422-Request POST http://192.168.1.72:1103/nodeworker/v1/tenants/admin/regions	300.266 2	354.349 0

如表2所示为PCA、DeepLog、单任务的TCN模型和基于TCN的多任务学习模型在OpenStack数据集上的异常检测效果。从Recall中可以发现，单任务的TCN模型和基于TCN的多任务学习模型都可以检测出所有的异常部署。从Precision可以看出，为了确保异常部署能够被检测出，基于深度学习的异常检测方法都不同程度地将正常部署划分为异常的，DeepLog尤其容易将正常的部署误检为异常。PCA虽然避免了这一情况的发生，但是其异常检测的效果较差。本文的基于TCN的多任务学习模型取得了较好的异常检测效果，其异常检测准确率（Accuracy）相较其他对比模型至少提升了7.7个百分点。

表2 OpenStack数据集上的异常检测效果

Tab. 2 Anomaly detection effects on OpenStack dataset

模型	Accuracy	Precision	Recall	F1_score
PCA	0.825	0.980	0.670	0.790
DeepLog	0.885	0.882	0.938	0.909
单任务TCN	0.923	0.889	1.000	0.941
多任务TCN	0.962	0.941	1.000	0.970

为了评价多任务TCN模型相较于单任务TCN模型的优势，下面将比较基于TCN的多任务学习模型和单任务学习模

型在路径异常检测和时间预测任务上的效果。如表3所示为路径异常检测中不同概率阈值$P_{\text{threshold}}$对于两种模型异常检测准确率的影响。在系统正常运行过程中,有部分事件发生次数很少,这在模型预测中表现为相关事件的概率较小,因此在路径异常的检测中概率阈值的设置尤为重要,较大的概率阈值会导致正常的部署被误判成异常,当$P_{\text{threshold}} = 10^{-6}$时,异常检测的准确率最高,多任务TCN模型的准确率高于单任务TCN模型,因此本文的路径异常检测中$P_{\text{threshold}}$设置为10^{-6}。

表3　概率阈值对异常检测准确率的影响

Tab. 3　Influence of probability threshold on anomaly detection accuracy

$P_{\text{threshold}}$	单任务TCN	多任务TCN
1E－3	0.615	0.654
1E－4	0.730	0.769
1E－5	0.808	0.846
1E－6	0.923	0.962

为了对比单任务TCN模型和多任务TCN模型在路径预测任务上的准确性,对于某种类型的事件E_i,统计并计算下个位置处实际发生事件为E_i时两种模型预测事件E_i发生的概率的均值。如图7所示为多任务TCN模型和单任务TCN模型在正常的测试样本中对于每个事件预测概率均值的对比。从图7中可以发现,较多事件的多任务TCN模型的预测概率均值大于单任务TCN模型的预测概率均值,检测的正常样本中共包含752个事件,其中533个事件的多任务TCN模型的预测概率均值大于单任务TCN模型的预测概率均值。多任务模型在预测下个位置处发生的事件时,正常的事件有更高的概率,多任务学习模型预测更为准确。

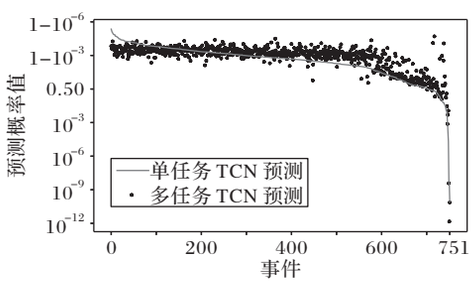


图7　事件预测概率值对比

Fig. 7　Comparison of event prediction probability

图8为多任务TCN模型和单任务TCN模型在时延预测任务上的对比,数据为检测集中的正常样本,计算了每个事件模板发生位置处对应时延预测值和实际值的平均误差。

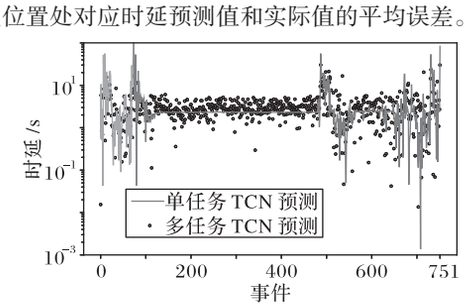


图8　时延预测误差对比

Fig. 8　Comparison of time delay prediction error

从图8中可以看出,单任务TCN模型在较小的时延数值预测中要优于多任务TCN模型,但是整体上多任务TCN模型的时延平均误差区间比单任务模型更小,总体上多任务模型的时延预测效果要优于单任务模型。

6　结语

针对云计算工作流中存在的路径异常和时延异常,本文提出了一种基于日志的异常检测方法。该方法将路径异常检测建模为多分类任务,将时延异常检测建模为回归任务,构建并训练了一种基于时序卷积网络(TCN)的多任务学习模型用于日志的异常检测,通过事件模板生成从日志流中提取事件序列,通过间隔采样的方法从日志流中提取时间序列。基于TCN的多任务学习模型通过共享TCN的浅层部分,促进各自任务的学习。实验结果表明该方法在异常检测方面取得了较好的效果。

接下来进一步的工作如下:首先,所提模型中损失函数参数权重的优化较困难,下一步工作需要设计相关方法以代替设置损失函数权重。其次,本文时延异常检测方法对数值较小的时延并不敏感,如何避免将正常划分为异常以提高异常检测的精度,仍是未来需要进一步解决的问题。

参考文献(References)

[1] 陆杰,李丰,李炼. 分布式系统中的日志分析及应用[J]. 高技术通讯,2019,29(4):303-320. (LU J, LI F, LI L. Log analysis for distributed systems and its application [J]. High Technology Letters, 2019, 29(4): 303-320.)

[2] 胡珉,白雪,徐伟,等. 多维时间序列异常检测算法综述[J]. 计算机应用,2020,40(6):1553-1564. (HU M, BAI X, XU W, et al. Review of anomaly detection algorithms for multidimensional time series [J]. Journal of Computer Applications, 2020, 40(6): 1553-1564.)

[3] 仇媛,常相茂,仇倩,等. 基于长短期记忆网络和滑动窗口的流数据异常检测方法[J]. 计算机应用,2020,40(5):1335-1339. (QIU Y, CHANG X M, QIU Q, et al. Stream data anomaly detection method based on long short-term memory and sliding window [J]. Journal of Computer Applications, 2020, 40(5): 1335-1339.)

[4] NGUYEN H, SHEN Z, TAN Y, et al. FChain: toward black-box online fault localization for cloud systems [C]// Proceedings of the 2013 IEEE 33rd International Conference on Distributed Computing Systems. Piscataway: IEEE, 2013: 21-30.

[5] WANG C. EbAT: online methods for detecting utility cloud anomalies [C]// Proceedings of the 2009 Middleware Doctoral Symposium. New York: ACM, 2009: Article No. 4.

[6] 贾统,李影,吴中海. 基于日志数据的分布式软件系统故障诊断综述[J]. 软件学报,2020,31(7):1997-2018. (JIA T, LI Y, WU Z H. Survey of state-of-the-art log-based failure diagnosis [J] Journal of Software, 2020, 31(7): 1997-2018.)

[7] XU W, HUANG L, FOX A, et al. Detecting large-scale system problems by mining console logs [C]// Proceedings of the 2009 ACM SIGOPS 22nd Symposium on Operating Systems Principles. New York: ACM, 2009: 117-132.

[8] CHEN M, ZHENG A X, LLOYD J, et al. Failure diagnosis using decision trees [C]// Proceedings of the 2004 1st International Conference on Autonomic Computing. Piscataway: IEEE, 2004: 36-43.

[9] LIANG Y, ZHANG Y, XIONG H, et al. Failure prediction in IBM BlueGene/L event logs [C]// Proceedings of the 2007 7th IEEE International Conference on Data Mining. Piscataway: IEEE, 2007: 583-588.

[10] HE S, ZHU J, HE P, et al. Experience report: system log analysis for anomaly detection [C]// Proceedings of the 2016 IEEE 27th International Symposium on Software Reliability Engineering. Piscataway: IEEE, 2016: 207-218.

[11] YU X, JOSHI P, XU J, et al. CloudSeer: workflow monitoring of cloud infrastructures via interleaved logs [J]. ACM SIGARCH Computer Architecture News, 2016, 44(2): 489-502.

[12] TAK B C, TAO S, YANG L, et al. LOGAN: problem diagnosis in the cloud using log-based reference models [C]// Proceedings of the 2016 IEEE International Conference on Cloud Engineering. Piscataway: IEEE, 2016: 62-67.

[13] NOVAES M P, CARVALHO L F, LLORET J, et al. Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment [J]. IEEE Access, 2020, 8: 83765-83781.

[14] DU M, LI F, ZHENG G, et al. DeepLog: anomaly detection and diagnosis from system logs through deep learning [C]// Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, 2017: 1285-1298.

[15] HOCHREITER S, SCHMIDHUBER J. Long short-term memory [J]. Neural Computation, 1997, 9(8): 1735-1780.

[16] BROWN A, TUOR A, HUTCHINSON B, et al. Recurrent neural network attention mechanisms for interpretable system log anomaly detection [C]// Proceedings of the 2018 1st Workshop on Machine Learning for Computing Systems. New York: ACM, 2018: 1-8.

[17] LU S, WEI X, LI Y, et al. Detecting anomaly in big data system logs using convolutional neural network [C]// Proceedings of the 2018 IEEE 16th International Conference on Dependable, Autonomic and Secure Computing/ the 16th International Conference on Pervasive Intelligence and Computing/ the 4th International Conference on Big Data Intelligence and Computing and Cyber Science and Technology/ the 3rd Cyber Science and Technology Congress. Piscataway: IEEE, 2018: 151-158.

[18] 王易东,刘培顺,王彬. 基于深度学习的系统日志异常检测研究[J]. 网络与信息安全学报,2019,5(5):105-118. (WANG Y D, LIU P S, WANG B. Research on system log anomaly detection based on deep learning [J]. Chinese Journal of Network and Information Security, 2019, 5(5): 105-118.)

[19] MACMAHAN H B, HOLT G, SCULLEY D, et al. Ad click prediction: a view from the trenches [C]// Proceedings of the 2013 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2013: 1222-1230.

[20] 任明,宋云奎. 基于深度学习的云计算系统异常检测方法[J]. 计算机技术与发展,2019,29(5):54-57. (REN M, SONG Y K. Anomaly detection for cloud computing systems based on deep learning [J]. Computer Technology and Development, 2019, 29(5): 54-57.)

[21] 杨瑞朋,屈丹,朱少卫,等. 基于改进时间卷积网络的日志序列异常检测[J]. 计算机工程,2020,46(8):50-57. (YANG R P, QU D, ZHU S W, et al. Anomaly detection for log sequence based on improved temporal convolutional network [J]. Computer Engineering, 2020, 46(8): 50-57.)

[22] 钟雅,郭渊博. 基于机器学习的日志解析系统设计与实现[J]. 计算机应用,2018,38(2):352-356. (ZHONG Y, GUO Y B. Design and implementation of log parsing system based on machine learning [J]. Journal of Computer Applications, 2018, 38(2): 352-356.)

[23] GUO S, LIU Z, CHEN W, et al. Event extraction from streaming system logs [C]// Proceedings of the 2018 International Conference on Information Science and Applications, LNEE 514. Singapore: Springer, 2018: 465-474.

[24] BAI S, KOLTER J Z, KOLTUN V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling [EB/OL]. [2020-08-22]. https://arxiv. org/pdf/1803. 01271. pdf.

[25] CARUANA R. Multitask learning [J]. Machine Learning, 1997, 28(1): 41-75.

[26] LONG M, CAO Z, WANG J, et al. Learning multiple tasks with multilinear relationship networks [C]// Proceedings of the 2017 31st International Conference on Neural Information Processing Systems. Red Hook: Curran Associates Inc., 2017: 1594-1602.

[27] MISRA I, SHRIVASTAVA A, GUPTA A, et al. Cross-stitch networks for multi-task learning [C]// Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2016: 3994-4003.

This work is partially supported by the National Key Research and Development Program of China (2018YFB1003702), the Industry-University-Research Cooperation Project of Zhongxing Telecommunication Equipment Corporation, the Incubation Project of National Natural Science Foundation of China from Nanjing University of Posts and Telecommunications (NY219084).

YAO Jie, born in 1996, M. S. candidate. His research interests include log analysis, deep learning.

CHENG Chunling, born in 1972, Ph. D., professor. Her research interests include data management, resource management and optimization.

HAN Jing, born in 1978, M. S., senior engineer. Her research interests include machine learning, event mining.

LIU Zheng, born in 1980, Ph. D., lecturer. His research interests include network data mining.