基于 BERT 的日志异常检测自监督模型

一、日志基本概念的描述

日志是指系统所指定对象的某些操作和其操作结果按时间有序的集合。简而言之，日志记录包括记录有关系统（应用程序）在执行时执行的操作的信息比如系统事件、用户行为和系统错误等信息。我们将这些记录写入某些持久性介质，例如数据库表或磁盘中的文件。这样，我们可以在之后访问这些信息并对其进行审查，任何类型的日志记录的主要用途是故障排除。通读日志条目会可以复盘一些计算机事件，因为它提供了在指定时间跨度内系统和网络中发生的前后事件的有用信息[1]。例如，网络管理员可以通过分析网络日志来了解某个时间间隔内的网络带宽使用情况。同样，应用程序开发人员使用应用程序日志来识别和修复程序代码中的错误。图 1 是 Java 日志框架 SLF4J 生成日志的一个例子。

```
public void setTemperature(Integer temperature) {
  // ...
  logger.debug("Temperature set to {}. Old temperature was {}.", t, oldT);
  if (temperature.intValue() > 50) {
    logger.info("Temperature has risen above 50 degrees.");
  }
}
```

⇓

```
0 [setTemperature] DEBUG Wombat - Temperature set to 61. Old temperature was 42.
0 [setTemperature] INFO Wombat - Temperature has risen above 50 degrees.
```

图 1

日志一般有着如下特点。1.难以理解：目前为止，还没有关于计算机系统以及计算机网络设备日志的统一格式和标准，不同系统的日志文件记录的格式及记录的信息都存在一定的差别[2]。所以，需要相应的日志文件的说明文档，才能读懂日志中所包含的信息，这对于普通的非计算机专业人员来说，是非常困难的。2.容易遭到破坏：一般的系统日志、应用程序日志以及网络设备日志，都没有采取很强的保护措施，甚至有些日志文件，直接以文本的方式存储，没有加密以及校检处理的措施，因此，日志文件往往都很脆弱，容易遭到修改、删除等破坏。所以，日志文件非常脆弱。3.数据量大：一些服务器上的日志，如 Web 服务器日志、防火墙日志、FTP 日志等等，其数据量非常巨大，日志文件一天所产生的日志，少则几百兆，多则几个 G，几十个 G 甚至上 T 的数据量。海量的日志数据，给日志存储和日志分析，都带来了很大的困难，特别是依靠人工进行分析，则往往不可能完成。4.关联性：一条日志记录，只能反映某个时候发生了某些特定的操作，而多条日志，按照时间顺序，进行一定关联，就有可能反映一用户某一个活动的完整情况[3]。因此，不可孤立地看待日志，对日志的分析要结合其上下文。

日志可以用于多种目的，比如不同网络路径上流量的信息可能有助于用户优化网络性能或检测恶意入侵；通话详细记录可以监控谁打电话给谁，何时打电话，经过进一步分析，可以显示整个城市的通话量和下降率。当然，日志也可以作为计算机犯罪中的电子证据。

二、日志分析的现实意义-作为计算机犯罪的电子证据

1.计算机犯罪的背景

随着信息技术的迅猛发展，计算机和网络的应用日益广泛和深入，普及程度也越来越高，可以说已经融入人们的日常生活，成为其中必不可少的一部分。信息网络在给人类带来巨大便利和进步的同时，也因其特有的开放性、隐蔽性、高效性以及快捷性等特点，为犯罪分子提供了高超的作案技术手段和广阔的作案空间，滋生出了一种新型的犯罪方式，那就是在电子和数字领域中，特别是电脑空间中的犯罪。1991 年，我国公安部计算机安全监察司对计算机犯罪提出了我国自己的定义:计算机犯罪就是指“以计算机为工具或以计算机资产为对象实施的犯罪行为”[4]。计算机犯罪出现了值得我们研究的新动向[5]:一是利用计算机网络犯罪的案件呈多发趋势，犯罪种类多、领域广，涉案金额惊人。我国从 1986 年深圳市公安机关侦破第一起利用计算机网络盗窃储户存款案以来,计算机犯罪的案件一直呈大幅度上升趋势，而且涉案金额巨大，给国家和人民带来巨大的损失[6]。其中网络色情、赌博、窃取密码、截获数据、侵入网络、传播病毒、网络诈骗、非法盗版、信息炸弹等网络犯罪事件更是层出不穷，呈愈演愈烈之势。二是犯罪方法多样化，类型扩大化。目前我国常见计算机犯罪方法有十多种，如活动天窗、特洛伊木马术、意大利香肠术、数据欺骗、蠕虫、逻辑炸弹、冒名顶替、乘机而入、废品利用、社交方法、利用扫描器、计算机病毒等。

2.日志对于打击计算机犯罪的作用

起初，日志文件通常被当作定位系统错误、调试系统的重要。而现在，人们发现日志文件记录着整个系统的历史操作，这非常有利于取证人员分析用户的行为，进而能够找出证明真相的证据,越来越具有取证价值。日志是由计算机系统内部运行的程序产生，因此能够将系统内部发生的各种各样的事件详细地记录下来。例如记录系统的日常运行状况，其中包含了某些组件运行错误的信息，此外，日志还能实时的记录一些系统状态等等。通过日志，取证人员能够较好较快地找到“入侵者”的操作行为，从而再现“入侵者”的入侵场景，为打击计算机犯罪提供了重要线索，并且是电子证据的一个重要来源。

日志的信息量非常庞大。普通应用程序一天产生的容量少则几十兆、几百兆，多则有几个 G，甚至几十个 G，如此海量的日志数据给日志的存储和分析带来极大的困难。这些海量日志数据若靠人工阅读和分析它们几乎是不可能的，更不可能实时通过计算机系统日志发现系统安全事件。现有的计算机取证工具中，没有特别实用的日志分析工具，也缺乏相关的规范性的指导。因此如何从海量的日志中发现作为关键证据的异常点是一个值得研究的课题。

三、日志的数据结构与存储位置

截至目前为止，计算机日志没有一个统一的标准。日志大概可以分为以下几种：分布式网络服务器日志，如 HDFS Log，Zookeeper Log；超级计算机系统日志，如 BGL Log，Thunderbird Log；一般操作系统日志，如 Windows Log，Linux Log；服务器日志，如 Apache Log，OpenSSH Log…… 它们的数据存储结构和存储位置都各不相同。

以 HDFS Log 为例（我下面的实验也是在 HDFS 数据集上完成），HDFS 是 Hadoop 分布式文件系统，可提供对应用程序数据的高吞吐量访问，解决海量数据存储问题。HDFS 它的设计目标就是把超大的数据集存储到多台普通计算机上（横向扩展），并且可以提供高

可靠性和高吞吐量的服务，支持通过添加节点的方式对集群进行扩容。不同服务器上的 HDFS 分为不同角色，比如在 master 服务器上安装的是 NameNode 角色，在 slave 服务器上安装的都是 DataNode 的角色。
HDFS 相关日志的默认存储路径为“/var/log/Bigdata/hdfs/角色名”。NameNode 的日志存储地址：“/var/log/Bigdata/hdfs/nn”，DataNode 的日志存储地址：“/var/log/Bigdata/hdfs/dn”。

HDFS 日志的具体格式如下，根据 Block ID 将日志分割为记录道，每个道中记录着流量包的往来信息，如图 2 所示：

```
081109 212703 2053 INFO dfs.DataNode$PacketResponder: Received block blk_-967145856473901804 of size 67108864 from /10.250.6.191
081109 212904 2174 INFO dfs.DataNode$PacketResponder: Received block blk_8408125361497769001 of size 67108864 from /10.251.70.211
081109 212931 2211 INFO dfs.DataNode$PacketResponder: Received block blk_-1614641487214609125 of size 67108864 from /10.251.193.175
081109 213009 2207 INFO dfs.DataNode$PacketResponder: Received block blk_7577595658377008671 of size 67108864 from /10.251.71.97
081109 213028 2206 INFO dfs.DataNode$PacketResponder: Received block blk_3777400576053320362 of size 67108864 from /10.251.31.5
```

图 2

四、基于 Bert 模型的日志数据的分析方法

该如何从海量日志中找出异常日志呢？正如前面所介绍的，日志具有难以理解，数据量大，上下文关联性等特点，因此不建议使用人工方法对提取出的日志进行分析。最早诞生过一批基于传统机器学习算法增强运维能力。然而由于数据不平衡问题（正常日志的数量远超于异常日志的数量），训练二分类器来检测异常日志序列是不可行的。因此，许多无监督学习模型，如主成分分析 PCA；或单类分类模型，如单类 SVM[7][8]，被广泛用于检测异常。近些年，随着算力发展和数据体量增大，深度学习技术开始被用于日志分析领域，研究者们认为半结构化的日志消息也包含部分系统语义，类似于自然语言语料。因此研究者们纷纷采用语言模型对日志数据进行建模分析，例如 LSTM[9]，Transformer[10]等。为解决标签难以获取问题，很多研究者采用自监督、无监督，半监督等不需要完整标签的方法。总而言之，研究者们正在深挖深度学习在该领域的研究和应用价值。截至目前，综合表现最好的 SOTA 是基于 BERT 的日志异常检测自监督框架 LogBERT[11]，它通过两个新的自监督训练任务学习正常日志序列的模式，并且能够检测出不规则模式偏离正常日志序列的异常。

我下面将主要介绍 LogBERT 模型的思想，假设我现在已经从第四章节介绍的动态取证系统中搜集到了一个全是正常样本的日志数据集。

1.日志的结构化解析

日志消息是半结构化的数据，包含自由形式的文本，因为开发人员通常使用灵活的文本格式记录系统事件，以方便和灵活地使用，这进一步增加了自动分析日志数据的难度。第一步也是最重要的一步是日志解析，这是一个将自由文本原始日志消息解析为结构化事件流的过程。日志解析的目的就是把半结构化的数据处理为成结构化的，排除因变量不同而带来的差异。这时即使两条日志的变量不同，但是当把它都用星号来代替的时候他们就是一样的。我们把这种称之为日志（事件）模板，最终人工智能就是对这个日志模板进行处理。几十万条日志数据在通过日志解析后，就被统一成了几十条或者是几百条的日志模板，再通过识别最终的日志模板的一个语义信息就可以明白日志的内容。下图 3 是一个将半结构化的原始日志结构化的例子。

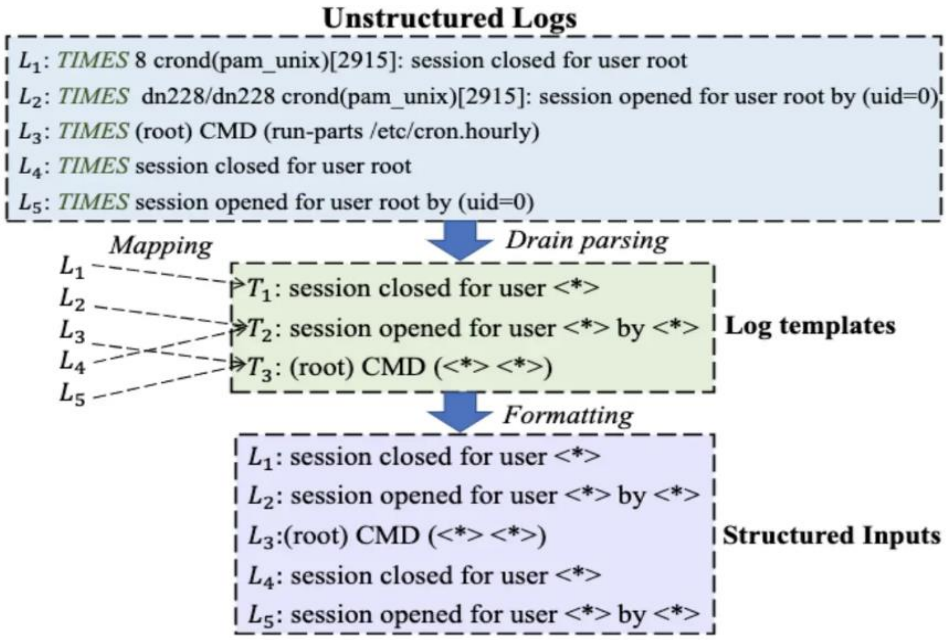


图 3

上图 3 展示了日志模版提取过程，从上到下依次是原始日志，解析后的日志模版，结构化好的日志，最后将结构化好的数据送入下游各种日志挖掘任务。详细来说，L1、L2、L3、L4、L5 表示五条原始日志，我们经过日志解析算法提取出三个模板：T1、T2、T3。经过 Mapping 后得到了五条结构化好的日志，即粉色框中的 L1～L5。日志解析会将我们认为日志中无关的信息去除，例如 Timestep、ID 等。常见的解析算法：Drain（基于树结构相似度） Spell（最长公共子序列） AEL （常数和变量的发生频率） IPLoM（迭代分区策略，根据消息长度、令牌位置和映射关系等）。

在本例中，给定一个非结构化日志序列，遵循典型的预处理方法，我们首先通过日志解析器从每条日志中提取日志键 log keys（日志模板）。然后，我们可以将日志序列定义为有序日志键的序列 S={k1，…，kt，…，kT}，其中 kt∈ K 表示第 t 个(时序)位置的日志键，K 表示从全部数据集中提取的一个日志模板集合。

为了适应 Transformer encode 的结构，对于每一个样本 S={k1，…，kt，…，kT}，我们在 S 的开头添加一个特殊的标记 DIST。然后从每个 ki 推导出 xi：xi 是 ki 的日志键嵌入（e_ki）和位置嵌入（t_ki）的总和，分别 log key embedding matrix 和 sinusoid function 正弦函数推导。具体的模型输入格式可以见下图 4。

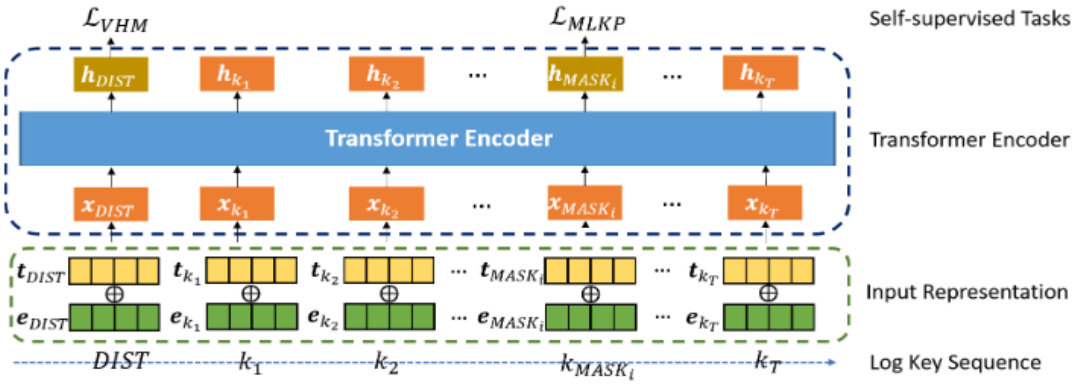


图 4

2.两个自监督训练任务

2.1 掩码日志键预测任务 (MLKP)

这基本上可以看作 Bert 模型的掩码预测任务。我们用特定 masked token 随机替换序列中的一些日志键（共替换 M 个）。训练目标是预测随机屏蔽的 log keys。目的是使 LogBERT 对正常序列的先验知识进行编码。

具体的来说，在 Bert 后接一个 softmax 层，我们采用交叉熵作为预测 masked token 差距的损失函数，定义为：

$$\mathcal{L}_{MLKP} = -\frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{M}\mathbf{y}^{j}_{[\mathrm{MASK}_i]}\log\hat{\mathbf{y}}^{j}_{[\mathrm{MASK}_i]},$$

其中，yj[MASKi]表示第 i 个 masked token 所对应的实际日志键 log keys，M 是第 j 个日志序列中屏蔽令牌 masked token 的总数，N 是数据集中样本的个数。由于正常和异常日志序列的模式不同，我们期望一旦 LogBERT 能够正确预测被屏蔽的日志键，它就能区分正常和异常日志序列。

2.2 超球体积最小化任务(Volume of Hypersphere Minimization)

我们提出了一个球形目标函数来约束正常序列的分布，该方法的目标是最小化所有样本构成的超球体的体积。其动机是正常的序列应该在嵌入空间中集中并彼此靠近，而异常的序列则远离球体的中心。

令 hj DIST 为第 j 个输入序列{DIST，x1，x2… ，xi}经过 Transformer encode 编码的结果。我们通过平均运算计算出一个集合中正常的序列生成的 h DIST 的中心 c，即 c=平均值（hj DIST)。目标函数是使正常的序列的 hj-DIST 接近中心 c，其中 N 是数据集中样本的个数。

$$\mathcal{L}_{VHM} = \frac{1}{N}\sum_{j=1}^{N}||\mathbf{h}^{j}_{\mathrm{DIST}} - \mathbf{c}||^2.$$

2.3 综合损失函数

我们期望训练集中的所有正常序列都靠近中心，而异常序列离中心的距离更大。最后，训练 LogBERT 的目标函数定义如下，其中 α 是平衡两个训练任务的超参数。

$$\mathcal{L} = \mathcal{L}_{MLKP} + \alpha\mathcal{L}_{VHM},$$

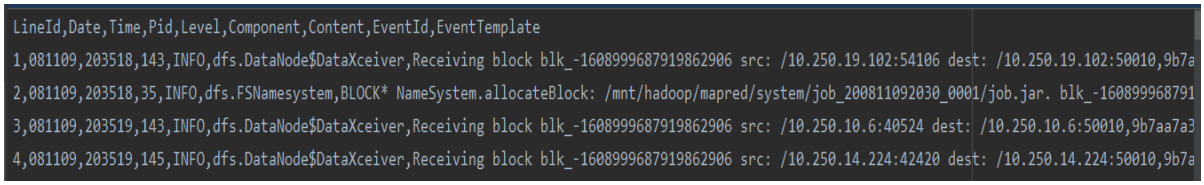
3.自监督思想异常检测的原理

由于 LogBERT 是在正常的日志序列上训练的，因此，如果测试的日志序列是正常的，则它可以在预测屏蔽的日志键时获得较高的预测精度。给定一个测试日志序列，我们首先用掩码令牌随机替换一些日志键，作为 LogBERT 的输入，计算各日志键出现在掩码令牌位置的概率分布。我们构建了一个候选集，该候选集由 g 个正常的日志键组成。如果测试样本中 masked 键位置真实的日志键在候选集中，我们将其视为正常键。当一个日志序列由 r 个以上的异常日志键组成时，我们将此日志序列标记为异常的。g 和 r 都是超参数，将根据验证集进行调整。

五、中间过程和结果展示

我采用 HDFS 数据集[12]在 LogBERT 上进行实验。

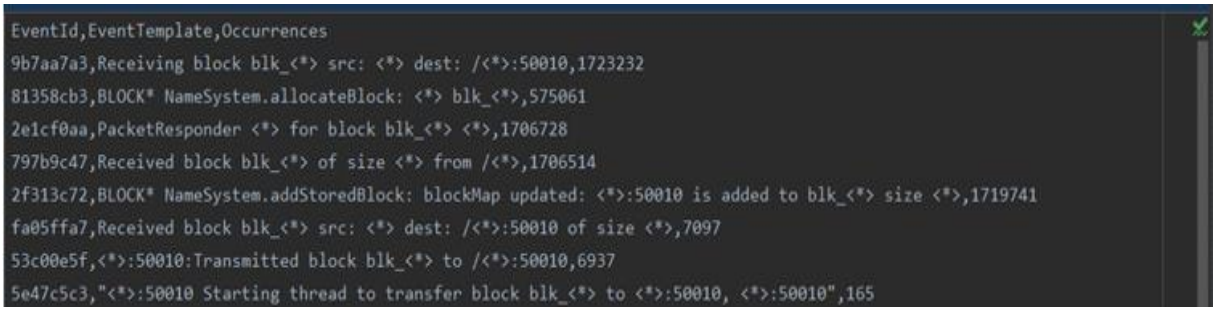
1. 原始的日志文件，我们挑选出都是正常的日志



```
LineId,Date,Time,Pid,Level,Component,Content,EventId,EventTemplate
1,081109,203518,143,INFO,dfs.DataNode$DataXceiver,Receiving block blk_-1608999687919862906 src: /10.250.19.102:54106 dest: /10.250.19.102:50010,9b7a
2,081109,203518,35,INFO,dfs.FSNamesystem,BLOCK* NameSystem.allocateBlock: /mnt/hadoop/mapred/system/job_200811092030_0001/job.jar. blk_-160899968791
3,081109,203519,143,INFO,dfs.DataNode$DataXceiver,Receiving block blk_-1608999687919862906 src: /10.250.10.6:40524 dest: /10.250.10.6:50010,9b7aa7a3
4,081109,203519,145,INFO,dfs.DataNode$DataXceiver,Receiving block blk_-1608999687919862906 src: /10.250.14.224:42420 dest: /10.250.14.224:50010,9b7a
```

图 5

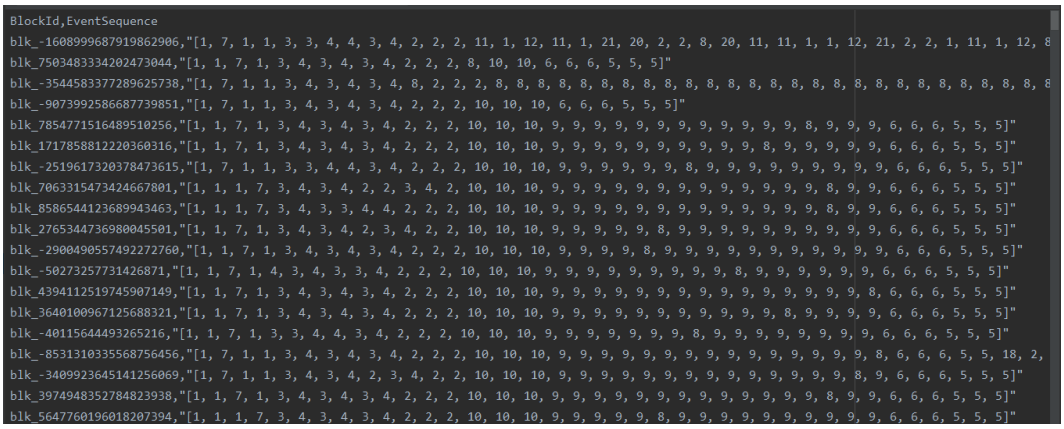
2. 日志的结构化解析，整个日志数据集共归类为为 47 个日志模板，编号为 1-47



```
EventId,EventTemplate,Occurrences
9b7aa7a3,Receiving block blk_<*> src: <*> dest: /<*>:50010,1723232
81358cb3,BLOCK* NameSystem.allocateBlock: <*> blk_<*>,575061
2e1cf0aa,PacketResponder <*> for block blk_<*> <*>,1706728
797b9c47,Received block blk_<*> of size <*> from /<*>,1706514
2f313c72,BLOCK* NameSystem.addStoredBlock: blockMap updated: <*>:50010 is added to blk_<*> size <*>,1719741
fa05ffa7,Received block blk_<*> src: <*> dest: /<*>:50010 of size <*>,7097
53c00e5f,<*>:50010:Transmitted block blk_<*> to /<*>:50010,6937
5e47c5c3,"<*>:50010 Starting thread to transfer block blk_<*> to <*>:50010, <*>:50010",165
```

图 6

3. 构造的日志序列



```
BlockId,EventSequence
blk_-1608999687919862906,"[1, 7, 1, 1, 3, 3, 4, 4, 3, 4, 2, 2, 2, 11, 1, 12, 11, 1, 21, 20, 2, 2, 8, 20, 11, 11, 1, 1, 12, 21, 2, 2, 1, 11, 1, 12, 8
blk_7503483334202473044,"[1, 1, 7, 1, 3, 4, 3, 4, 3, 4, 2, 2, 2, 8, 10, 10, 6, 6, 6, 5, 5, 5]"
blk_-3544583377289625738,"[1, 7, 1, 1, 3, 4, 3, 4, 3, 4, 8, 2, 2, 2, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8
blk_-9073992586687739851,"[1, 7, 1, 1, 3, 4, 3, 4, 3, 4, 2, 2, 2, 10, 10, 10, 6, 6, 6, 5, 5, 5]"
blk_7854771516489510256,"[1, 1, 7, 1, 3, 4, 3, 4, 3, 4, 2, 2, 2, 10, 10, 10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 8, 9, 9, 9, 6, 6, 6, 5, 5, 5]"
blk_1717858812220360316,"[1, 1, 7, 1, 3, 4, 3, 4, 3, 4, 2, 2, 2, 10, 10, 10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 8, 9, 9, 9, 9, 9, 6, 6, 6, 5, 5, 5]"
blk_-2519617320378473615,"[1, 7, 1, 1, 3, 3, 4, 4, 3, 4, 2, 2, 2, 10, 10, 10, 9, 9, 9, 9, 9, 9, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 6, 6, 6, 5, 5, 5]"
blk_7063315473424667801,"[1, 1, 1, 7, 3, 4, 3, 4, 2, 2, 3, 4, 2, 10, 10, 10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 8, 9, 9, 6, 6, 6, 5, 5, 5]"
blk_8586544123689943463,"[1, 1, 1, 7, 3, 4, 3, 3, 4, 4, 2, 2, 2, 10, 10, 10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 8, 9, 9, 6, 6, 6, 5, 5, 5]"
blk_2765344736980045501,"[1, 1, 7, 1, 3, 4, 3, 4, 2, 3, 4, 2, 2, 10, 10, 10, 9, 9, 9, 9, 9, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 6, 6, 6, 5, 5, 5]"
blk_-2900490557492272760,"[1, 1, 7, 1, 3, 4, 3, 4, 3, 4, 2, 2, 2, 10, 10, 10, 9, 9, 9, 9, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 6, 6, 6, 5, 5, 5]"
blk_-50273257731426871,"[1, 1, 7, 1, 4, 3, 4, 3, 3, 4, 2, 2, 2, 10, 10, 10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 8, 9, 9, 9, 9, 9, 9, 6, 6, 6, 5, 5, 5]"
blk_4394112519745907149,"[1, 1, 7, 1, 3, 4, 3, 4, 3, 4, 2, 2, 2, 10, 10, 10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 8, 6, 6, 6, 5, 5, 5]"
blk_3640100967125688321,"[1, 1, 7, 1, 3, 4, 3, 4, 3, 4, 2, 2, 2, 10, 10, 10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 8, 9, 9, 9, 9, 6, 6, 6, 5, 5, 5]"
blk_-40115644493265216,"[1, 1, 7, 1, 3, 3, 4, 4, 3, 4, 2, 2, 2, 10, 10, 10, 9, 9, 9, 9, 9, 9, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 6, 6, 6, 5, 5, 5]"
blk_-8531310335568756456,"[1, 7, 1, 1, 3, 4, 3, 4, 3, 4, 2, 2, 2, 10, 10, 10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 8, 6, 6, 6, 5, 5, 18, 2,
blk_-3409923645141256069,"[1, 7, 1, 1, 3, 4, 3, 4, 2, 3, 4, 2, 2, 10, 10, 10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 8, 9, 6, 6, 6, 5, 5, 5]"
blk_3974948352784823938,"[1, 1, 7, 1, 3, 4, 3, 4, 3, 4, 2, 2, 2, 10, 10, 10, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 8, 9, 9, 6, 6, 6, 5, 5, 5]"
blk_5647760196018207394,"[1, 1, 1, 7, 3, 4, 3, 4, 3, 4, 2, 2, 2, 10, 10, 10, 9, 9, 9, 9, 9, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 6, 6, 6, 5, 5, 5]"
```

图 7

4. 自监督训练任务的训练过程

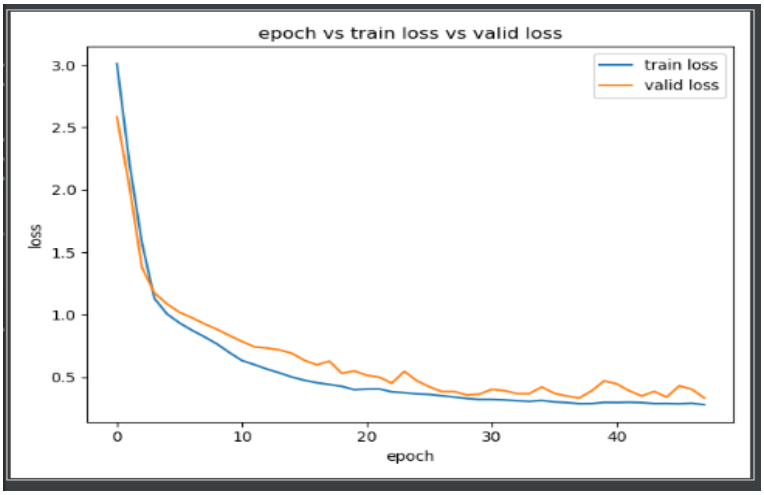


图 8

5. 在测试集上验证该模型的异常识别效果

```
Saving test normal results
Saving test abnormal results
Saving test normal errors
Saving test abnormal results
best threshold: 0, best threshold ratio: 0.0
TP: 8213, TN: 551111, FP: 2257, FN: 2434
Precision: 78.44%, Recall: 77.14%, F1-measure: 77.79%
elapsed_time: 2241.1670072078705

Process finished with exit code 0
```

图 9

六、总结

通过这次大作业，我通过阅读文献查阅资料，深入理解了计算机日志，以及如何处理日志并分析日志。相比于自然语言，日志有着难以理解，数据量大的特点，难以从海量日志中挖掘出黑客破坏的关键性活动。所幸的是，随着算力的发展，诞生了很多用于自然语言处理的深度学习模型，在日志挖掘方面它们同样有着不错的效果。

虽然日志取证仅仅是信息安全专业中的一小部分。但它的发展历程却能以小博大地反映信息安全的其他领域如恶意代码审计，日志检测的发展过程并指出信息安全发展的新方向：与深度学习结合。信息安全领域如此，其他学科领域亦如此。机器学习的智能算法，尤其是深度学习，在各行各业都开辟了新天地。我们要善于把握时代趋势，主动学习一些新技术，而不能固步自封，抱残守缺。未来的时代学科与学科之间的壁垒会越来越小，我们学生需要培养的是知识交叉的能力，即用别的专业的知识来改进自己专业的东西。

计算机是一个实践性很强的专业，仅仅是通过考试这种理论性考核是不够的。国内计算机教育存在的普遍问题是学生缺乏动手实践的能力，无论将来是工作还是读研，这

都是不利的。而现在这种采用调研大作业的形式，一方面可以培养学生对某一个取证方向的兴趣，另一方面也可以提前让他们适应读文献，做研究的过程，为将来打下更好的基础。

七、参考文献

[1] Suleman Khan, Abdullah Gani, Ainuddin Wahid Abdul Wahab, Mustapha Aminu Bagiwa, Muhammad Shiraz, Samee U. Khan, Rajkumar Buyya, and Albert Y. Zomaya. 2016. Cloud Log Forensics: Foundations, State of the Art, and Future Directions. ACM Comput. Surv. 49, 1, Article 7 (March 2017), 42 pages. https://doi.org/10.1145/2906149

[2] Oliner A, Ganapathi A, Xu W. Advances and challenges in log analysis[J]. Communications of the ACM, 2012, 55(2): 55-61.

[3]李秋香. 基于 Windows 日志的计算机取证研究[D]. 吉林大学, 2008.

[4]朱建辉，计算机动态取证系统，吉林大学硕士论文，长春，2009.

[5]匿名，浅议我国计算机犯罪现状及综合控制机制的建立，宁波市鄞州区人民检察院，宁波，2007.

[6]于志刚, 计算机犯罪研究, 中国检察出版社, 北京， 1990.

[7] K.L, Huang, H.K., Tian, S.F., Xu, W.: Improving one-class SVM for anomalydetection. In: Proceedings of the 2003 International Conference on Machine Learning and Cybernetics. (2003).

[8] Wang, Y., Wong, J., Miner, A.: Anomaly intrusion detection using one class SVM.In: Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004. pp. 358–364 (2004).

[9] Gers F A， Schmidhuber J， Cummins F . Learning to Forget: Continual Prediction with LSTM[J]. Neural computation, 2000, 12(10):p.2451-2471.

[10] Vaswani A， Shazeer N， Parmar N , et al. Attention Is All You Need[C]// arXiv. arXiv, 2017.

[11] H. Guo, S. Yuan and X. Wu, "LogBERT: Log Anomaly Detection via BERT," 2021 International Joint Conference on Neural Networks (IJCNN), 2021, pp. 1-8, doi: 10.1109/IJCNN52387.2021.9534113.

[12] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I. Jordan. 2009. Detecting large-scale system problems by mining console logs. In Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles (SOSP '09). Association for Computing Machinery, New York, NY, USA, 117–132. https://doi.org/10.1145/1629575.1629587