学习库, 由 Facebook 的人工智能研究组在 Torch 的基础上开发. Pytorch 在学术领域广泛应用, 作为常用深度学习研究平台之一[15], 其 API 接口统一, 采用了动态计算图机制和自动求导机制, 能够方便地搭建定制化的神经网络并进行训练. 相对于谷歌开发的 TensorFlow 框架, Pytorch 拥有以下几点优势:

(1) Pytorch 中的模型定义更为简单, 并且提供了容易调用的包, 而 TensorFlow 接口定义种类繁杂.

(2) Pytorch 所采用的动态图计算机制相比 TensorFlow 的静态图更加灵活, 对于研究者更加友好,

(3) 对 Caffe 具有良好的支持, 可以联合英伟达 GPU 进行高效的神经网络训练.

3 实验及结果分析

3.1 实验数据

云数据中心已经成为互联网的基础设施, 在网络流量日益增长的今天起到核心作用, 与此同时, 云数据中心的网络设备故障的发生率也在上升, 这使得服务器的性能下降, 对用户终端的使用体验造成影响. 本文采用循环神经网络训练公开的数据中心数据集, 并对未来故障进行预测, 这样运维人员可以在潜在故障发生前进行干预并解决.

实验使用数据集为阿里巴巴集团在 Github 平台公布的阿里云集群数据[16], 该数据集包括以下两部分数据:

(1) cluster-trace-v2017: 1300 台机器 12 小时的运行数据. 该批数据包含在线服务的数据集合以及批次化的工作负荷记录.

(2) cluster-trace-v2018: 4000 台机器长达 8 天的运行数据. 该批数据除了包含 v2017 的数据种类外, 额外包含 DAG 产品的运行负荷信息[17].

本文采用了 cluster-trace-v2018 数据集进行训练和测试, 预测所包含的故障分为设备响应时间过长, CPU 利用率过高, 内存利用率过高和传输信息速率低 5 种. 实验所采取的流程如图 3 所示.

如图 3 所示, 首先对数据进行预处理, 包括去冗余与清洗, 标准化两步. 冗余数据指与故障预测没有关联的数据或者数据不随时间变化的数据, 例如设备型号信息, 不同客户账号信息等. 冗余信息对预测效果没有影响并且会造成额外计算负担, 因此预先对冗余进行清洗是重要一步. 数据集标准化是提高预测效果的重要步骤, 当不同机器间指标水平相差较大时, 直接用原始指标进行分析会突出数值较高指标在预测中的作用, 导致高数值指标在预测中所占权重过大, 影响结果的可靠性. 这里我们对机器的不同指标采用最为常用的 Z-score 标准化, 以时间序列 $x_1, x_2, \cdots, x_n$ 为例, Z-score 计算方式为:

$$y_i = \frac{x_i - \overline{x}}{s} \tag{5}$$

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{6}$$

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{7}$$

则新序列 $y_1, y_2, \cdots, y_n$ 的均值为 0, 标准差为 1. Z-score 方法对数值区间较大, 离群值较多的情况比较合适, 在对不同指标进行 Z-score 处理后, 能够使数据落入一个较小的区间, 减轻不同指标间数值差异的影响.

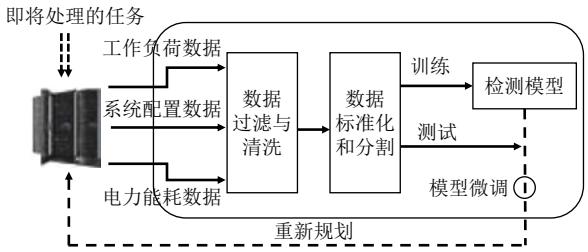

图 3 实验流程图

本文以每 1 分钟的数据为一个窗口, 将 1 分钟的阿里云硬件集群的各项指标进行平均, 作为单个数据进行统计. 本文使用统计检验将 CPU 使用率和内存分配率等指标与平均指标严重偏离的时间点作为故障时间点 (P-value<0.05). CPU 使用率在一天内使用率波动较大, 在部分时间点存在高峰, 经过统计后发现在高峰使用期故障发生率较高, 这与真实情况相符合. 针对内存使用故障, 使用请求内存和真实使用内存之差作为判断故障指标 (P-value<0.05). 统计结果表明 CPU 使用率高峰和内存使用故障高峰存在一定的关联, 这正常情况相一致. 在实验中仅考虑单个时间点只存在一种故障, 将故障检测视为单标签分类问题. 图 4 显示了不同故障数目的柱形图.

3.2 网络结构与实验结果

本文使用的检测模型设计架构如图 5 所示.

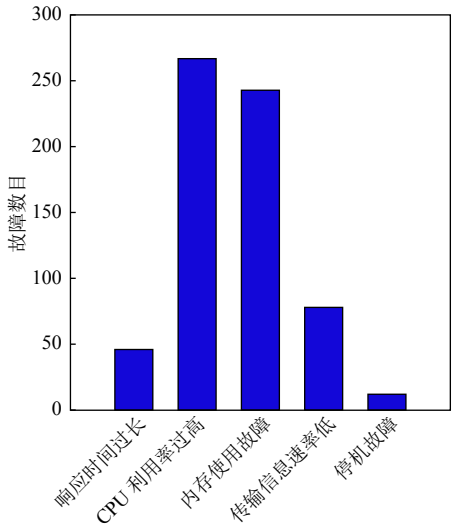

图 4 故障数目统计

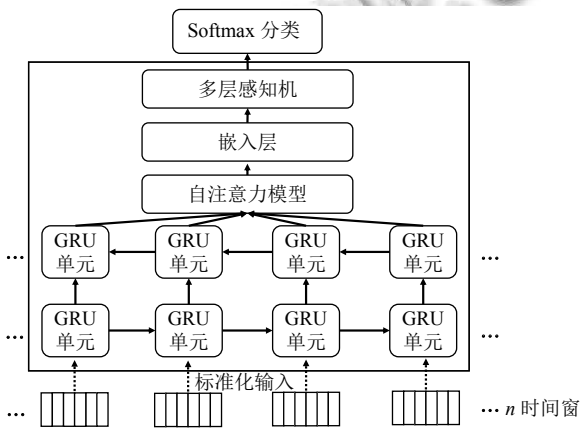

图 5 神经网络架构

图 5 中, 输入层包含 200 个神经元, 因此数据被分割为 200 个时间窗输入. 本文使用双向门控循环神经网络处理数据输入. 双向门控循环神经网络 (BiGRU) 可以看做的 LSTM 一种拓展, 将 LSTM 中的 LSTM 模块替换为 GRU 并使用双层 GRU 模块反向叠加. GRU 将 LSTM 中隐藏状态和细胞状态合并成一种状态, 因此显著缩短了训练时间. 更明确地说, GRU 读取词嵌入向量t_i以及隐藏层状态向量h_{i-1}后, 经过门控计算产生输出向量C_i和隐藏层状态向量h_i, 具体计算方法参考下列公式:

$$\begin{cases} z_i = \sigma(W_z t_i + V_z h_{i-1} + b_z) \\ r_i = \sigma(W_z t_i + V_z h_{i-1} + b_r) \\ c_i = \tanh(W t_i + v(r_i \odot h_{i-1}) + b) \\ h_i = z_i \odot h_{i-1} + (1 - z_i) \odot c_i \end{cases} \tag{8}$$

其中, $z \in \Re^d$, $r \in \Re^d$分别表示接受 d 维向量的输入门与重置门, $\{W_z, W_r, W, V_z, v_r, V\}$表示权重矩阵, $\{b_z, b_r, b\}$为偏置向量, $\odot$表示矩阵点乘. 在双向 GRU 后是 self-attention 层、embedding 层和多层感知机 (Multi-Layer Perception, MLP). 输出层为 5 个神经元组成的 Softmax 层, 计算输入时间段在 5 类故障上的概率.

为了让神经网络能够学习到对预测故障有效的特征, 本文采用了自然语言处理中的 embedding 技术. 类似于常用的预训练词向量, 在输入层后加入 embedding 层, 其中包含 100 个神经元, 对输入特征起到降维作用[17]. 此外, 在 embedding 层前加入了注意力层, 并使用了经典的自注意力模型. 自注意力机制引入了查询向量 q (query vector), 通过打分函数计查询向量和输入向量直接的相关性, 同时引入了一个注意力变量 $t \in [1, N]$代表选择的索引位置. 具体计算方式如下:

$$\begin{aligned} a_i &= p(t = i|X, q) \\ &= Softmax(s(x_x, q)) \\ &= \frac{\exp(s(x_i, q))}{\sum_{j=1}^{N} \exp(s(x_i, q))} \end{aligned} \tag{9}$$

其中, a_i 是注意力分布, $s(x_i,q)$ 是注意力打分函数. 自注意力打分函数采用基于缩放点积函数, 缩放点积的定义如下:

$$s(x_i, q) = \frac{x_i^{\mathrm{T}} q}{\sqrt{d}} \tag{10}$$

其中, d 表示输入向量的维度. 缩放点积模型是基于点积模型的一种改进, 区别在于缩放点积模型除以向量维度 d 的平方根. 当 d 很大时, 点积模型的值会出现较大的方差, 因此导致 Softmax 的梯度变小, 缩放点积模型的提出解决了这一问题. 可以看出, 自注意力层通过查询向量实现对输入数据的权重分配, 这一查询向量可以通过反向传播进行学习和优化, 从而能够对重要的特征分配更大的权重. 在阿里云数据中心数据集中, 不同类型的数据对故障分类的重要性存在差别. 例如, 线程分配数目和消息队列排队数目对 CPU 响应时间过长和 CPU 利用率过高这两类故障类型起到直接的影响, 缓存区数据量和堆栈区使用率则对内存使用故障和传输速率低起到决定性作用, 而在输入数据中无法体现此类差别. 因此, 在双向 GRU 层充分学习输入数据的时序信息后, 引入自注意力层能够对映射后特征进行权重分配, 使得不同位置的特征对每类故障学

习不同的权重, 从而提升最终的检测效果. 在模型实现中, 我们采用了以自注意力模型为基础的 Multi-head Attention. Multi-head Attention 在自注意力基础上进行了进一步拓展, 能够同时学习到 GRU 输出序列的位置编码信息和特征的权重信息. 该注意力机制在著名的自然语言处理模型 Transformer 和 Bert 中被广泛采用.

实验平台的具体配置如下:

操作系统: Windows 10.

GPU: NVIDIA RTX2080Ti, 11 GB 显存.

RAM: 64 GB.

深度学习框架: Pytorch 1.3 稳定版.

开发工具: Visual Studio Code.

编程语言: Python 3.6.

Adam 是一种基于随机梯度下降 (Stochastic Gradient Descent, SGD) 的一阶优化算法, 与 SGD 不同在于 SGD 在训练过程中学习率不会改变, 而 Adam 通过计算梯度的一阶和二阶矩估计动态改变学习率, 是一种自适应学习率优化算法, 同时结合了 AdaGrad 和 RMSProp 两种算法的优点[18].

本文将数据集分割为训练集与测试集, 其中验证集数目占 20%, 训练集数目占 80%. 使用 Adam 优化算法训练神经网络, 实验结果显示 Adam 算法效果卓越, 如图 6 所示, 在使用 Adam 算法后, 训练集和测试集上的 loss 均能够降低到 0.05 左右.

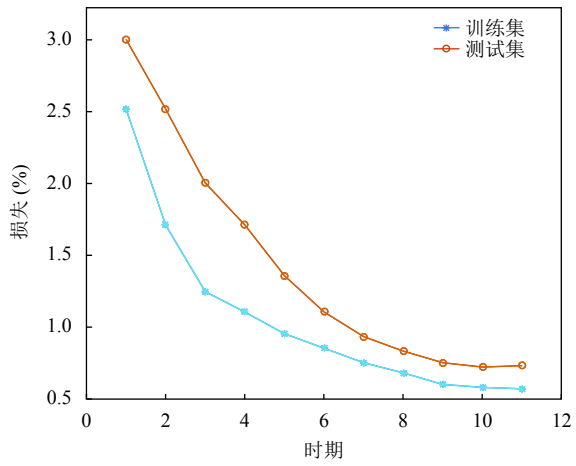

图 6 模型损失函数变化

图 7 展示了神经网络对于故障检测的准确率, 在实验中, 设置 batch 大小为 200, 通过 11 个 epoch 后算法已经接近收敛并在测试集上获得了超过 98% 的准确率.

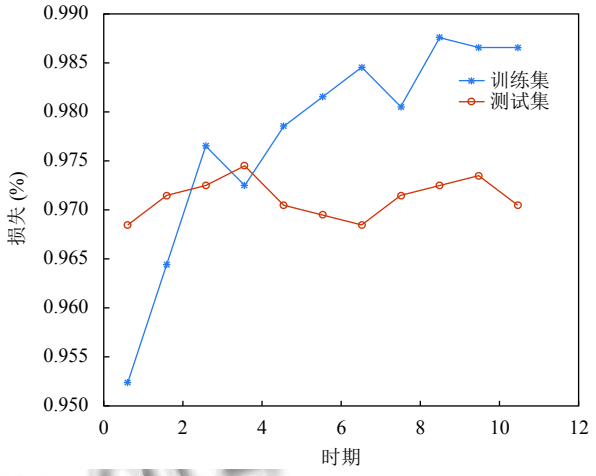

图 7 模型准确率

作为对比, 本文使用 SVM, KNN 和普通 LSTM 模型对故障进行检测, 检测结果如表 1 所示.

表 1 不同模型准确率对比

模型	训练集准确率	测试集准确率
BiGRU+embedding+attention	0.987	0.971
LSTM	0.964	0.933
SVM	0.977	0.903
KNN	0.918	0.882

综合来说, 深度学习模型相比传统机器学习模型拥有更高的诊断准确率. SVM 作为经典的分类模型, 其在训练集上表现出良好的效果, 但是测试集上准确率显著下降, 存在明显的过拟合现象. KNN 分类器因为没有显式的训练过程, 在训练集和测试集上的诊断效果均较为一般, LSTM 在训练集和测试集上表现较为稳定, 表明深度学习模型能够学习到数据中的时序变化信息. 相比普通 LSTM, 加入了 embedding 层和自注意力机制后 BiGRU 模型准确度相比基线模型 LSTM 有 2% 的准确度的提升. 实验结果显示深度学习模型能够在云数据中心进行部署, 相比传统机器学习模型拥有更高的故障诊断准确率. 除此之外, 随着数据的积累, 深度学习模型的准确率能够进一步提高.

4 结论与展望

为了解决当前云数据中心缺乏故障检测方法, 且当前方法均基于仿真数据实验的问题, 本文提出了一种基于 Pytorch 和双向 GRU 网络的云数据中心故障检测方法. GRU 模型的使用相比传统的 LSTM 提高了训练速度, 并且双向机制的结合进一步提高了模型的检

测准确度. 在对数据进行预处理后, 利用 embedding 技术使神经网络能够提取关于故障检测的相关特征, 并使用特征进行进一步加工和处理, 最后利用 Adam 优化算法训练神经网络. 基于阿里云集群数据的实验结果显示, 相比于其他模型, 本文提出的模型准确率有着明显改善, 有助于智能电网云数据中心故障检测的准确率, 可靠性的全面提升.

参考文献

1 Uchechukwu A, Li KQ, Shen YM. Retraction note: Sustainable cost and energy consumption analysis for cloud data centers. Frontiers of Computer Science, 2020, 14(1): 239. [doi: 10.1007/s11704-015-4020-6]

2 Mokadem R, Hameurlain A. A data replication strategy with tenant performance and provider economic profit guarantees in cloud data centers. Journal of Systems and Software, 2020, 159: 110447. [doi: 10.1016/j.jss.2019.110447]

3 宋杰, 孙宗哲, 刘慧, 等. 混合供电数据中心能耗优化研究进展. 计算机学报, 2018, 41(12): 2670–2688. [doi: 10.11897/SP.J.1016.2018.02670]

4 赵伟, 白晓民, 丁剑, 等. 基于协同式专家系统及多智能体技术的电网故障诊断方法. 中国电机工程学报, 2006, 26(20): 1–8. [doi: 10.3321/j.issn:0258-8013.2006.20.001]

5 罗孝辉, 童晓阳. 计及可信度的变结构贝叶斯网络电网故障诊断. 电网技术, 2015, 39(9): 2658–2664.

6 徐彪, 尹项根, 张哲, 等. 基于拓扑图元信息融合的电网故障诊断模型. 电工技术学报, 2018, 33(3): 512–522.

7 Novelo AF, Cucarella EQ, Moreno EG, *et al*. Fault diagnosis of electric transmission lines using modular neural networks. IEEE Latin America Transactions, 2016, 14(8): 3663–3668. [doi: 10.1109/TLA.2016.7786348]

8 宣恒农, 张润驰, 左苗, 等. 面向数据中心网络的分层式故障诊断算法. 电子学报, 2014, 42(12): 2536–2542. [doi: 10.3969/j.issn.0372-2112.2014.12.029]

9 Qi XG, Wang BC, Liu LF. Fault diagnosis based on dial-test data in datacenter networks. Journal of Systems Engineering and Electronics, 2019, 30(5): 1035–1043. [doi: 10.21629/JSEE.2019.05.19]

10 Cybenko G. Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems, 1989, 2(4): 303–314. [doi: 10.1007/BF02551274]

11 Greff K, Srivastava RK, Koutník J, *et al*. LSTM: A search space odyssey. IEEE Transactions on Neural Networks and Learning Systems, 2017, 28(10): 2222–2232. [doi: 10.1109/TNNLS.2016.2582924]

12 Wang TY, Qiu RG, Yu M. Predictive modeling of the progression of Alzheimer's disease with recurrent neural networks. Scientific Reports, 2018, 8(1): 9161. [doi: 10.1038/s41598-018-27337-w]

13 Filonov P, Lavrentyev A, Vorontsov A. Multivariate industrial time series with cyber-attack simulation: Fault detection using an LSTM-based predictive data model. arXiv preprint arXiv: 1612.06676, 2016.

14 Park D, Kim S, An YL, *et al*. LiReD: A light-weight real-time fault detection system for edge computing using LSTM recurrent neural networks. Sensors, 2018, 18(7): 2110. [doi: 10.3390/s18072110]

15 Ketkar N. Introduction to Pytorch. In: Ketkar N, ed. Deep Learning with Python. Berkeley: Apress, 2017. 195–208.

16 Lu CZ, Ye KJ, Xu GY, *et al*. Imbalance in the cloud: An analysis on Alibaba cluster trace. Proceedings of 2017 IEEE International Conference on Big Data (Big Data). Boston, MA, USA. 2017. 2884–2892.

17 Pennington J, Socher R, Manning CD. Glove: Global vectors for word representation. Proceedings of 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar. 2014. 1532–1543.

18 Kingma DP, Ba LJ. Adam: A method for stochastic optimization. Proceedings of the 3rd International Conference on Learning Representations. San Diego, CA, USA. 2015.