面向边缘部署场景的轻量神经网络修复算法

方毓楚[1)] 李文中[1)] 曾 曜[1)] 郑 阳[2)] 胡 峭[2)] 陆桑璐[1)]

[1)](南京大学计算机科学与技术系 南京 210023)

[2)](华为技术有限公司可信理论技术与工程实验室 广东 深圳 518129)

摘 要 随着深度学习技术的不断进步,神经网络在各领域得到广泛应用,特别是在边缘计算环境中,例如智能交通和新型电网等典型场景.然而,神经网络的可靠性问题限制了其在真实世界的广泛应用.在复杂的边缘环境中,预训练模型往往因未涵盖所有可能的边缘情况而性能下降.因此,针对部署中的神经网络进行高效修复成为一个关键的研究课题.传统修复方法通常涉及整个模型的重新训练,这在边缘场景中具有诸多局限性.首先,不同地理区域的设备可能面临独特的自然噪声,使得统一模型难以适应所有环境.其次,深度神经网络的大规模参数使得其训练和部署时资源消耗巨大,且更新期间的服务中断将降低系统的可用性.为解决这些问题,本文提出了一种轻量级的补丁式神经网络修复算法.该算法通过引入个性化的补丁来增强神经网络对不同边缘环境中自然噪声和边角案例的鲁棒性.具体的,在故障定位阶段,类比于程序插桩中通过注入代码以检测、改进和分析软件行为,本文提出了神经网络插桩技术.通过将模型探针插桩进神经网络,观测其内部运行情况,实现了对错误样本的故障定位.在故障修复时,通过插入无监督搜索得到的神经网络补丁来纠正原始神经网络的输出.此外,本文提出了故障预测模块以提前预测潜在的错误输出,从而仅在必要时激活补丁.在基于 2 个数据集、15 种噪声以及 4 个神经网络模型的实验中,与现有修复算法相比,本文方法在修复性能上取得了 6.64%至 20.00%的提升.同时,本文方法所需的训练样本量减少了超过 90%,而所需更新的参数量最高减少了 91.94%.这种有效且轻量的特性为解决边缘计算环境中神经网络的可靠性问题提供了有效途径.

关键词 神经网络修复;深度边缘计算;故障定位;故障预测;神经网络补丁

中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2024.01413

Light-Weight Neural Network Repair for Edge Computing Scenarios

FANG Yu-Chu[1)] LI Wen-Zhong[1)] ZENG Yao[1)] ZHENG Yang[2)] HU Zheng[2)] LU Sang-Lu[1)]

[1)](*Department of Computer Science and Technology, Nanjing University, Nanjing* 210023)

[2)](*Trustworthiness Theory, Technology & Engineering Lab (TTE Lab), Huawei Technologies Co., Ltd., Shenzhen, Guangdong* 518129)

Abstract The continuous evolution and progress of deep learning technology have ushered in an era where neural networks play a pivotal role in various fields. This is particularly evident in edge computing environments such as intelligent transportation systems and next-generation power grids. However, despite the widespread applications, the reliability of neural networks remains a significant bottleneck, restricting their full potential in real-world scenarios. One of the primary challenges arises in complex edge environments, where pre-trained models often suffer from performance degradation due to the inherent difficulty of covering all possible edge scenarios comprehensively. Consequently, the need for an efficient repair mechanism for deployed neural networks has become a paramount focus of research. Traditional methods of repairing neural

收稿日期:2023-07-18;在线发布日期:2024-01-18.本课题得到国家自然科学基金面上项目(61972196)、国家自然科学基金重点项目(61832008,61832005)、江苏省前沿引领技术基础研究重大项目(BK20222003)资助.**方毓楚**,博士研究生,主要研究方向为神经网络压缩、神经网络边缘部署、神经网络鲁棒性.E-mail:fangyuchu@smail.nju.edu.cn.**李文中**(通信作者),博士,教授,博士生导师,主要研究领域为分布式计算、数据挖掘、移动云计算、无线网络、普适计算和社交网络.E-mail:lwz@nju.edu.cn.**曾 曜**,硕士,主要研究方向为神经网络压缩、边缘计算、神经网络鲁棒性.**郑 阳**,博士,主要研究方向为神经网络测试、神经网络修复、自动驾驶.**胡 峭**,博士,主要研究方向为软件可靠性、可靠性理论、ad-hoc 网络.**陆桑璐**,博士,教授,博士生导师,主要研究领域为分布式计算、普适计算、无线网络.

networks typically involve the cumbersome process of retraining the entire model. This approach presents several challenges, especially in edge scenarios. Firstly, devices located in different geographical regions may encounter unique natural noise, making it a challenging task for a unified model to seamlessly adapt to all diverse environments. Secondly, the large-scale parameters of deep neural networks contribute to substantial resource consumption during training and deployment, and the potential for service interruptions during updates poses a threat to system availability. To address these challenges head-on, this paper proposes a lightweight patch-based neural network repair algorithm. The core objective of this algorithm is to augment the robustness of neural networks against natural noise and corner cases prevalent in different edge environments by introducing personalized patches. The fault localization phase is a key aspect of this algorithm, drawing inspiration from probes in program instrumentation used to detect, improve, and analyze software behavior. In the context of neural networks, the paper introduces neural network instrumentation technology. This involves strategically inserting model probes into the neural network to observe its internal patterns, facilitating the localization of faults in error samples. During the fault repair phase, customized patches, obtained through an innovative unsupervised search, are seamlessly integrated into the original neural network to rectify its output. Additionally, a fault prediction module is introduced, capable of predicting potential errors in advance and activating patches only when deemed necessary. In a series of comprehensive experiments leveraging 2 datasets, 15 distinct types of noise, and 4 diverse neural network models, the proposed approach yielded remarkable results. Performance improvements ranging from 6.64% to an impressive 20.00% were observed when compared to existing repair algorithms. Furthermore, the innovative approach demonstrated its efficiency by drastically reducing the required quantity of training samples by over 90%. The highest reduction in required updated parameters reached an astounding 91.94%. This effective and lightweight patch-based approach not only addresses the reliability concerns of neural networks in edge computing environments but also sets the stage for more resilient and adaptable applications in the complex and dynamic real-world scenarios of the future.

Keywords neural network repair; deep edge computing; fault localization; fault prediction; neural network patch

1 引 言

深度学习算法,尤其是神经网络技术,正在快速进步并广泛应用于实际生产环境中.在边缘计算领域,其应用受到了学术界和工业界的广泛关注.由于神经网络在计算机视觉、强化学习等领域有着出色的性能,其在智能交通、新型电网等边缘计算典型场景中有着广阔的应用前景.然而,神经网络的可靠性问题是其不可忽视的缺陷[1-3].这在诸如智慧交通等安全相关领域的应用中限制了其发展前景.

在边缘计算环境中,神经网络模型通常在数据中心使用预先收集的训练数据进行训练,然后部署到实际场景中[4,5].但实际情况的复杂性意味着预收集的数据可能无法涵盖所有情况,导致模型在遇到所谓的边角案例(edge cases)输入时表现不佳[6].这些边角案例往往由真实世界中未知的噪声形成(如雨、雾、相机运动造成的模糊等).这种真实世界中存在的噪声被称为自然噪声,其可造成神经网络的准确率出现高达40%的下降[7].而在地理跨度大的边缘场景,如无人值守智能电网或城市安防监控中,由于设备分布在不同地理位置,它们可能面临不同的自然噪声影响(如戈壁中的沙尘、海滨的大雾、高寒地区的降雪等),这大大增加了模型修复的难度.因此,如何高效快捷地修复已经部署在边缘端的神经网络模型成为一个重要的研究课题.

传统的神经网络修复算法[5,8-10]利用错误样本来对神经网络的参数进行微调(fine-tune).这些方法要么直接将错误样本添加到预先收集的训练数据中进行重新训练,要么从错误样本中提取某些样本

特征，并将这些特征加入原始训练数据中以微调神经网络.然而，这种需要重新训练整个模型并进行部署的方法在边缘场景下有很大的局限性.

首先，正如前文所述，边缘端部署的神经网络可能会被部署在不同的环境中，面临不同环境噪声的挑战，这极大地增加了训练统一模型的难度.而为每个边缘环境训练定制的模型又会增加部署和维护的开销.其次，深度神经网络模型通常具有巨大的体积，对其重新训练会占用大量计算资源，同时也会消耗大量能源，从而产生巨大的碳足迹(carbon footprint)[11].此外，由于需要将修复后的神经网络模型再次传输回边缘端进行部署，这进一步增加了成本.同时，在更新期间，模型将无法继续提供服务，从而降低了系统的可用性.

在设计针对边缘场景的神经网络修复算法时，面临以下主要难点：(1)边缘场景的复杂性：边缘场景的复杂性意味着不同边缘端可能会遭遇不同风格的自然噪声和边角案例[12]，这增加了模型修复的难度；(2)边缘场景的低能耗要求：在边缘场景中，由于设备通常受限于能源供应，如电池寿命，修复算法必须高效利用计算资源，以减少训练和部署的能耗；(3)数据标记成本高：在边缘环境中，获取带有标签的数据通常较为困难且成本高昂.这导致边缘端的标签数据相对稀缺，使得传统的需要大量有标签数据的修复方法难以实施.

针对上述问题，本文提出了一种轻量级的补丁式神经网络修复算法.该算法通过为神经网络打上个性化的补丁来增强神经网络对不同边缘环境中自然噪声和边角案例的鲁棒性.具体而言，类比于程序插桩的思想[13]，在神经网络训练阶段，本文设计了一种模型探针来进行神经网络的插桩，以观测神经网络内部的运行情况.这样，在模型部署后，当神经网络性能下降时，利用模型探针观测模型在收集到的错误样本上的运行情况，来对每个错误样本进行故障定位.为了修复神经网络，本文结合错误样本，在神经网络的故障处打入一个轻量级的神经网络补丁.补丁是一个小型神经网络模块，用于纠正原始神经网络的输出.补丁的网络结构是通过在边缘环境中的实际数据上进行无监督结构搜索得到的，以适应其中的自然噪声.为了减少神经网络的遗忘性问题并提高神经网络的运行效率，本文利用收集到的错误样本和网络探针训练了一个故障预测模块，来预测运行时的输入样本是否会致使神经网络出现错误输出.这样，模型的补丁将只需要在模型被预测可能出错时才介入模型的运算，从而避免正常样本的运算被补丁所影响，并大大提高模型的计算效率.通过这种方式，每个边缘区域可以设计一个个性化的补丁结构来应对迥异的自然噪声.同时，由于补丁模型相较于原始神经网络具有较小的体积，其训练、部署和维护成本也大大降低.本文主要贡献有以下几点：

(1)提出了一种基于补丁的神经网络修复算法，该算法利用轻量化且个性化的补丁，为深度边缘计算提供了快速且定制化的修复服务.

(2)提出了基于神经网络插桩和费雪信息的故障定位方法，用于确定错误样例对应的故障层次.此外，我们还训练了一个故障预测回归模型，以引导补丁仅在错误样例上介入模型推理，从而避免对正确样本造成遗忘性问题.

(3)设计了专为深度边缘计算而优化的修复部署策略，将修复流程嵌套在数据中心端和边缘端的训练中，有效平衡了边缘计算中的通信开销、计算开销和时间开销.

(4)通过在两个真实世界数据集上，使用四个不同的神经网络模型进行多组对比实验，验证了本文提出的方法在模型修复效果、部署与维护开销以及各模块的有效性方面的卓越表现.

本文第 2 节介绍相关工作；第 3 节介绍基于补丁的神经网络修复算法；第 4 节分析在边缘场景下应用该修复算法的完整流程及各类开销；第 5 节通过对比实验验证了所提方法的有效性；最后总结全文.

2　相关工作

本节将介绍在边缘端部署深度神经网络的相关工作，阐述在部署过程中所面临的神经网络鲁棒性挑战，同时还会涉及到已有的神经网络修复相关工作.

2.1　深度边缘计算

由于边缘设备多为轻量级设备，其计算资源、存储资源和能源资源往往受限.因此神经网络在边缘端的部署及运行必须具有高效的性质.为了支持这种轻量级的部署，许多工作尝试在尽量保持模型性能的情况下，通过剪枝[14-15]、量化[16-17]等方式压缩模型的大小.另一类工作尝试使用协同推理来解决这一困境[18-19].这个想法是将大型神经网络模型切分为若干部分并在多个设备之间进行部署.每个设备只承担原始模型的部分存储和计算，并通过与其

他设备的通信共同完成推理过程.然而,由于真实世界的复杂性以及边缘设备所处环境的多样性和不确定性,神经网络在部署后极可能会出现错误输出的情况.如何用收集到的错误样例对神经网络进行高效的修复将是神经网络在边缘端部署的重要课题.

2.2 神经网络鲁棒性

神经网络极易受到噪声的干扰,从而输出错误的结果.这些噪声可以是人为设计的对抗噪声[20-21],也可以是真实世界中潜在的自然噪声[7,22-23].在针对自然噪声的研究中,研究人员发现在自动驾驶场景中,雾、雨等驾驶环境的变化可以使得在 Udacity 自动驾驶挑战赛中胜出的模型产生数以千计的错误输出[24].同时,神经网络的模型结构也被证实和模型鲁棒性相关[25].一个合适的神经网络模型结构有利于增强神经网络对于某些噪声的抗干扰能力[26-27].

2.3 神经网络修复

神经网络修复是在神经网络产生错误输出后,利用错误样例对其进行修复的技术.CutMix[28]和 AugMix[29]是两种有助于增强模型鲁棒性的数据增强方法.它们通过随机遮掩、剪切等方法对输入图像进行一系列变化.这些方法可以作用于收集到的错误样例,并将其用于微调原始神经网络,从而增强其鲁棒性.FSGMix[8]采用类似 AugMix 的方法对输入图像进行增强.其利用错误样本训练一个高斯混合模型,从而指导对原训练图像进行数据增强时各操作的权重.Yu 等人[6]尝试使用风格转换模型以学习错误样本的风格,并将错误样本的风格迁移到原训练数据上,再以此微调原网络.一些工作尝试通过模型参数在错误样例的输出中起到的作用来对其进行针对性的调整.Zhang 等人[9]将原始数据集分割成一系列子集,并分别在子集上训练了一组相同结构的神经网络.然后,他们通过这些神经网络的参数的平均值来调整模型参数的大小和方向.Sohn 等人[10]利用敏感度分析定位错误神经元,并利用粒子群优化算法直接对其进行优化.文献[30]提出了一种基于补丁对神经网络进行修复的算法.本文提出了一种适用于边缘部署场景的轻量级、低开销的神经网络修复算法.与先前的工作不同,该算法不依赖于对原始数据集进行数据增强以重新训练和更新原始神经网络.该算法在增强模型鲁棒性的同时,避免了直接训练或更新已在边缘端部署的神经网络模型.

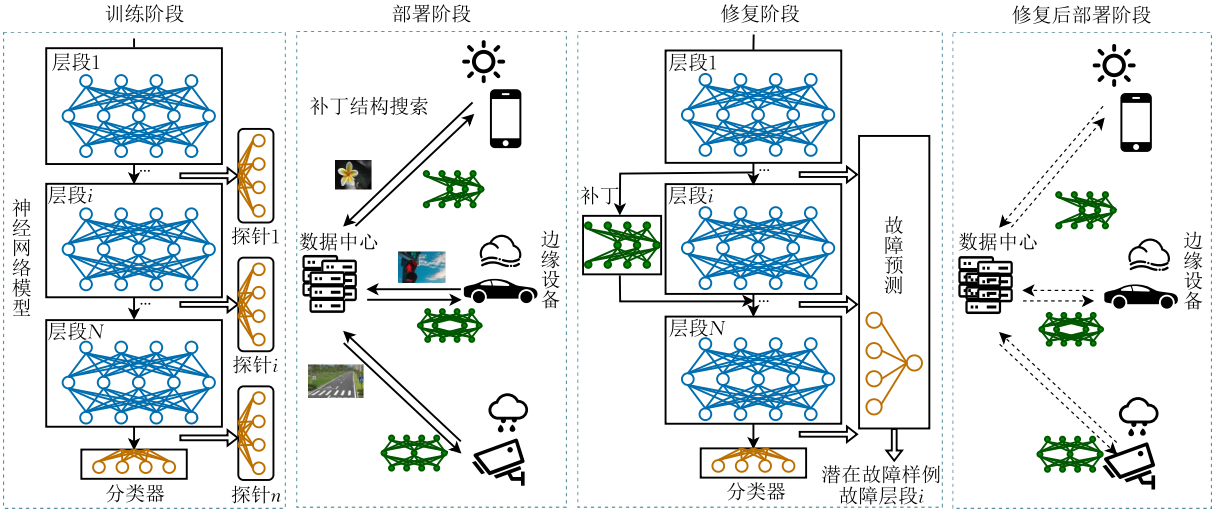


图 1 面向边缘部署的神经网络修复算法框架图

3 模型修复算法

在深度边缘计算的部署周期中,神经网络首先在预收集的有标签数据集 D_t 上进行训练,然后将其部署到实际的边缘环境中.在边缘设备上运行时,神经网络会接收来自实际生产环境中的无标签数据 D_u 作为输入.由于真实世界的多样性和复杂性,这些数据中可能包含一些致使神经网络输出错误结果的边角错误样例 D_e.这些错误样例可能是由多种未知的原因造成的,如自然天气噪声、受干扰的传感器输入、环境中的随机事件等等.假设运维人员在进行模型服务质量检测和维护时能够发现一部分异常样本.本文的目标是利用这些数据来修复已经在边缘端部署的神经网络.

本文以神经网络中最常见的图像分类任务作为行文阐述的基准任务.然而,需要注意的是,本文提出的修复算法并不局限于图像分类任务,它可以应

用于其他领域和任务.

3.1　故障定位

3.1.1　神经网络插桩

在软件工程中，程序插桩通过向计算机程序的源代码或二进制代码中插入附加的代码(称为“插桩代码”或“探针”(probe))，以收集额外的信息，来实现对程序运行时的监测、跟踪和改变[13]. 程序插桩可以在源代码级别或二进制级别进行. 在源代码级别，开发人员可以直接在程序代码中插入额外的函数调用、语句或注释. 在二进制级别，插桩工具通常会修改程序的可执行文件，以插入所需的代码. 通过插桩，开发人员可以测量程序执行时间、内存使用情况等性能相关指标、追踪程序的执行流程、确定测试用例是否覆盖代码的所有部分、检测和防止潜在安全漏洞和攻击等. 其被广泛应用于软件开发和测试领域. 类比于程序插桩，本文提出在神经网络中打入额外的探针，从而量化神经网络在正确运行时，其内部的运行情况，以此为基础来定位神经网络模型非正常运行时，异常点位所处的位置.

现代神经网络模型通常采用层段结构(stage)，每个模型包含若干个网络层段，而每个层段又包含若干运算层(layer，如卷积层、BN 层等). 同一层段内的运算层通常具有相同的输入输出维度，而层段间通常会有下采样、通道数增加等维度上的变换. 神经网络通过层段结构从浅到深对模型输入进行特征提取. 最终提取的特征会被喂入由全连接网络组成的分类器(classifier)中进行类别预测. 本文提出在神经网络的层段粒度上进行插桩. 具体而言，神经网络的每个层段都将训练一个探针. 探针的模型结构和神经网络的分类器的结构一致，其任务是利用模型层段的中间输出进行最终的分类任务. 由于现代神经网络模型多采用金字塔结构，其中间层输出维度通常小于最后一个层段的维度. 因此，可采用 1×1 卷积、池化等方式来将不同层段的输出维度与探针的输入维度对齐. 探针的训练损失函数为

$$\min_{W_i^P}\sum_{(x,y)\in D_t}\mathrm{CE}(y,\mathrm{softmax}(W_i^P\times O_i)) \tag{1}$$

其中 $(x,y)\in D_t$ 为预先收集的训练数据 D_t，O_i 表示模型的第 i 个层段在输入为 x 时的输出，W_i^P 为第 i 个网络层段的探针的参数，函数 CE(·)和 softmax(·)表示交叉熵函数和归一化指数函数. 公式中省略了对齐维度时的卷积、池化操作，若相应操作也有参数，会一并加入优化变量中.

通过探针，可以评估模型在正确运行时，各层段的输出中所包含的有助于最终分类的有效特征的含量. 当模型分类错误时，可以通过插桩的探针观察哪个层段的效果偏离了正常模式，从而定位问题所在.

3.1.2　基于费雪信息的故障定位

针对错误样本 $(x,y)\in D_e$，神经网络故障的层段应该是与产生这个错误输出最相关的层段. 因此本文采用费雪信息来评估模型的每个层段对于修复神经网络的重要性，其中最重要的层段被认为是故障层段. 费雪信息可用来描述随机变量分布的性质，它是度量可观测随机变量对目标分布中未知参数所包含信息量的指标[31-32]. 其可用来评估模型参数估计的精度. 费雪信息被定义为参数 θ 关于得分函数(score function，对数似然函数的梯度)的方差：

$$s(\theta)=\nabla_\theta\log p(y|\theta) \tag{2}$$

$$\boldsymbol{F}=\mathbb{E}_{p(y|\theta)}[\nabla\log p(y|\theta)\nabla\log p(y|\theta)^{\mathrm{T}}] \tag{3}$$

其中，$s(\cdot)$ 为得分函数，θ 为模型 $p(y|\theta)$ 的参数，$\boldsymbol{F}$ 是费雪信息矩阵.

在神经网络中，费雪信息度量了神经网络中的参数对于给定任务的重要性. 给定神经网络参数 θ，费雪信息矩阵定义为

$$\boldsymbol{F}=\mathbb{E}\left[\left(\frac{\partial\log p(y|x;\theta)}{\partial\theta}\right)\left(\frac{\partial\log p(y|x;\theta)}{\partial\theta}\right)^{\mathrm{T}}\right] \tag{4}$$

在实际应用中，神经网络权重的费雪信息由经验费雪信息矩阵的对角元素计算得来. 对于第 i 个模型层段 S_i 的参数 θ，本文利用先前插桩的探针来计算其在给定的数据批次中的费雪信息值的公式如下：

$$f(\theta,x)=\frac{1}{|B|}\sum_{(\hat{x},\hat{y})\in B}\left(\frac{\partial\log p(\hat{y}|\hat{x};\theta;W_i^P)}{\partial\theta}\right)^2 \tag{5}$$

其中 W_i^P 为探针的参数，$(\hat{x},\hat{y})$ 为数据批次 B 中的一个样本.

本文定义一个模型层段的费雪信息值为其所含参数的费雪信息值的平均值，因此可通过如下公式计算：

$$f(S_i,x)=\frac{1}{|S_i|}\sum_{\theta\in S_i}f(\theta,x) \tag{6}$$

模型层段的费雪信息值越高，该层段在纠正模型输出中的重要性就越大. 利用该工具，本文定义费雪信息值最大的模型层段 $\arg\max_i(f(S_i,x))$ 为导致错误用例 (x,y) 的故障层段. 算法 1 完整地阐述了神经网络故障定位的训练及定位的过程.

算法 1.　神经网络故障定位.

输入：待修复的神经网络模型 θ 及其各层段 $S_1\sim S_N\in\theta$，有标签训练集 D_t，错误样例 D_e，探针训练轮次 E

输出：模型探针 W_i^P，各层段费雪信息值 $f(S_i,x)$，故障层段 $\arg\max_i(f(S_i,x))$

//神经网络插桩

1. 初始化模型探针参数 $W_1^P\sim W_N^P$

2. FOR $e=1,2,\cdots,E$ DO
3. FOR $x\in D_t$ DO
4. 将 x 输入神经网络,得到各层段输出 $O_1\sim O_N$
5. 根据式(1)计算目标函数
6. 反向传播更新模型探针参数 $W_1^P\sim W_N^P$
7. END FOR
8. END FOR
//故障定位
9. FOR $B\in D_e$ DO
10. 根据式(6)计算 $x\in B$ 的各层段费雪信息值 $f(S_i,x)$
11. 定位层段 $\arg\max_i(f(S_i,x))$ 为故障层段
12. END FOR

3.2 故障修复

在定位到故障的模型层段后,如图 2 所示,可通过在该故障层段旁添加一个神经网络模块——补丁,来修正该层段的输出结果,从而保证最终模型分类结果的准确性.然而,由于不同的错误样例可能来源于不同的故障层段,为每个层段添加一个补丁会大大增加修复开销.为了保证模型在边缘端的高效部署,可采用共享的方式,让所有模型层段共用同一个补丁.这样做可以显著减少模型修复时所需的训练量以及模型部署过程中传输的参数量.

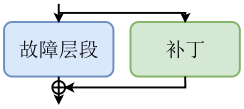


图 2 补丁修复示意图

3.2.1 补丁结构搜索

补丁是一个小的神经网络模块,其最直接的设计方法就是复制原始神经网络的层段.然而,神经网络的鲁棒性和其模型结构有一定的相关性[25].一个合适的模型结构可以显著地提高神经网络对某些自然噪声的抗性[26].因此,针对边缘部署环境中的真实输入样例来设计补丁的结构[33-35],有利于应对边缘环境中潜在的自然噪声的影响.然而,由于数据标注的成本往往很高,在边缘环境中获得足够的标记数据是几乎不可能实现的.鉴于无监督的神经网络结构搜索往往能够取得和有监督搜索相近的结果[36-37],本文提出了一种无监督的补丁结构搜索方案.如算法 2 所示,该方案旨在为给定边缘环境中的神经网络模型设计一个对应的补丁结构.

算法 2. 补丁结构搜索.

输入:待修复的神经网络模型 θ,无标签环境数据集 D_u,无监督任务 $t_1\sim t_M$,补丁备选运算池 $\mathcal{G}$,补丁训练轮次 E

输出:搜索得到的补丁网络结构 $\mathcal{G}'$

//训练阶段
1. 初始化补丁结构及各无监督任务的分类器 $W_1^u\sim W_M^u$
2. 将 D_u 划分为训练集 D_u^1 和测试集 D_u^2
3. FOR $e=1,2,\cdots,E$ DO
4. FOR $x\in D_u^1$ DO
5. 以概率 $1-e\cdot\left(1-\frac{1}{|\mathcal{G}|}\right)\Big/E$ 对运算池进行抽样得到本轮的补丁结构 $\pi(\mathcal{G})$
6. FOR $j=1,2,\cdots,M$ DO
7. 根据任务 j 对 x 预处理,得到样本 (x',y)
8. 将 x' 输入神经网络,计算各层段输入 $O_0\sim O_{N-1}$
9. 根据式(10)计算任务 j 上各层段输入经补丁计算后的输出 $U_i^j(O_{i-1})$
10. 根据式(11)计算任务 j 上的损失:CE_j
11. END FOR
12. 聚合所有任务上的损失 $\sum_j CE_j$
13. 反向传播更新结构 $\pi(\mathcal{G})$ 的参数和分类器参数 W_j^u
14. END FOR
15. END FOR
//搜索阶段
16. 以概率 $\frac{1}{|\mathcal{G}|}$ 从运算池抽样 200 个补丁结构
17. FOR $j=1,2,\cdots,M$ DO
18. 在 D_u^2 上对 200 个补丁结构进行测试并排序
19. END FOR
20. 选取平均排名最高的补丁结构作为 $\mathcal{G}'$

本文采用神经网络结构搜索中常用的基于单元模块(cell)的方法[33,38]来搜索补丁结构.如图 3,补丁结构由两个单元模块堆叠形成.初始时,单元模块可构成一个全连接的有向无环图.在图中,节点代表 3 维张量,边代表张量间的运算关系(如卷积、池化、恒等映射等),边从输入张量指向输出张量.节点 v_i 的值等于它的所有入度节点的计算之和,即:

$$v_i=\sum_{j<i}norm(e_{(j,i)}(v_j)) \tag{7}$$

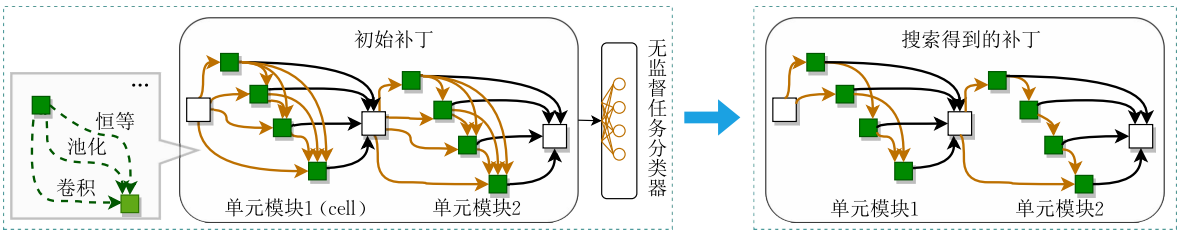


图 3 补丁结构搜索示意图

$$norm(x)=\frac{x-\mu_x}{\sqrt{\sigma_x^2+\epsilon}} \tag{8}$$

函数 $norm(\cdot)$对输入进行了标准化，$e_{(j,i)}(\cdot)$代表以节点 v_j为输入进行复合运算，其计算结果包含了从运算池$\mathcal{G}$(1×1、3×3 卷积，1×1、3×3 深度可分离卷积，3×3 最大池化，3×3 平均池化，恒等映射)中抽样的多个运算的结果的和. 其具体计算方法如下：

$$e_{(j,i)}(x)=\sum_k norm(\mathcal{G}_k(x)) \tag{9}$$

其中$\mathcal{G}_k(\cdot)$为抽样选中的运算之一.

每个单元模块的输出为其所有节点(除输入节点)的拼接(concatenation). 补丁结构搜索的目标就是为单元模块中的每一条边确定其所采用的运算. 若搜索后该边不含任何运算符，则等同于该边消失.

由于在边缘环境中难以收集足够的有标签数据，本文采用无监督的方式对补丁进行结构搜索. 我们选取常见的无监督任务，如预测图像旋转的角度，求解拼图游戏(jigsaw puzzle)等，来对补丁进行训练. 具体而言，从边缘环境中收集的图像将进行对应无监督任务预处理(旋转图像、切分图像等)，然后被喂入神经网络模型中. 在神经网络模型进行相应计算后，每个层段的输入(即上一个网络层段的输出)将被抽取出来，并喂入初始的补丁中. 补丁从这些输入中抽取相应特征，并将这些特征进一步输入到无监督任务的分类器中. 整个计算过程可表示为

$$U_i(O_{i-1})=W^u\times K^{\pi(\mathcal{G})}(O_{i-1}) \tag{10}$$

其中，O_{i-1}为神经网络第 $i-1$ 个层段的输出，即第 i 个层段的输入，$U_i(\cdot)$为第 i 个层段的输入经过补丁后输出的无监督任务的预测结果，W^u为无监督任务的分类器的参数，$\pi(\mathcal{G})$代表从运算池中被抽样后的补丁的复合运算，而 $K^{\pi(\mathcal{G})}(\cdot)$代表补丁利用抽样后的复合运算从输入中抽取特征的过程.

在搜索阶段，训练补丁的目标函数为

$$\min_{W\in\pi(\mathcal{G}),W^u} \mathrm{CE}(y,\mathrm{softmax}(\sum_{i=1}^{n}U_i(O_{i-1}))) \tag{11}$$

其中，y 代表无监督任务中自动生成的标签，$W\in\pi(\mathcal{G})$是被抽样选中的运算所包含的参数.

在搜索补丁阶段的训练中，补丁单元模块中的运算符将不断被抽样，并通过梯度下降对其中所含的参数进行训练. 本文采取了与文献[38]相同的抽样策略，在训练过程中线性降低了每个运算符被抽中的概率. 在训练后的搜索过程中，我们以$\frac{1}{|\mathcal{G}|}$的概率，随机对补丁的边进行抽样，以采集 200 个子网络结构. 我们评估这 200 个子网络结构在无监督任务上的性能，并选出其中综合性能最好的子结构$\mathcal{G}'$，作为搜索得到的补丁结构. 由于这个结构很好地完成了以边缘设备所处环境为输入的无监督任务，这意味着该结构非常适合从相应的输入中提取有效特征，从而有助于无监督分类任务. 这种抽取有效特征的能力也将有助于提高模型在有监督的图像分类任务中的表现.

3.2.2　基于补丁的神经网络修复

基于上节搜索得到的补丁模型，我们可以利用它对原神经网络进行相应的修复. 对于在边缘设备运行维护中收集到的错误样本(x,y)，本文定位其对应的故障层段，并将该故障层段的输入喂入到补丁中. 补丁会从其中抽取相应的特征，并将其和原模型层段的输出进行合并，当补丁输出维度与原模型层段输出维度不同时，可利用 1×1 卷积进行维度上的匹配. 该计算过程可表示为

$$O_i'=O_i+K^{\mathcal{G}'}(O_{i-1}) \tag{12}$$

其中 $i=\arg\max_i(f(S_i,x))$为第 3.1 节中定位发现的模型的故障层段，O_i为模型第 i 层段原有输出，$K^{\mathcal{G}'}(O_{i-1})$为补丁的输出，$O_i'$为模型的第 i 层段经修复后的最终输出，此处我们省略了对维度匹配的描述.

修复过程中训练的损失函数为

$$\min_{W\in\mathcal{G}'}\sum_{(x,y)\in D_e}\mathrm{CE}(y,O') \tag{13}$$

其中 O'为补丁介入后，神经网络的最终输出.

由于在相同环境中，补丁在无监督任务中表现出良好的性能，这意味着它能够从含有相同自然噪声的环境输入中有效地抽取高质量特征. 因此，将这些特征加回原网络，将有助于原网络的后续层段更有效地进行特征提取. 这将进一步对最终的图像分类任务产生积极的影响. 需要注意的是，由于只训练了补丁中的参数，原神经网络模型的参数保持不变. 这意味着模型的更新也只需要更新一个小的补丁网络结构，而无需改变原神经网络. 这大大降低了后续模型部署和维护的成本.

3.3　故障预测

理论上，在经过故障定位和补丁修复后，神经网络模型应该能在收集到的错误样例上输出正确的结果. 然而，当神经网络模型部署回实际的边缘环境后，如果遇到和预先收集到的错误样例不同的输入，模型可能无法判断当前输入是否会导致故障，也无法进行故障定位和修复. 为此，本文提出一个在神经网络模型部署时生效的故障预测方法，使得当模型输出结果不可靠时，补丁可以及时介入其前向传播过程.

通过使用式(6)分别计算普通样例和错误样例

上的神经网络模型各层段的费雪信息值,我们发现在错误样例上,模型的费雪信息值普遍更高.同时,由于错误样例的出现本质上代表神经网络模型存在某种缺陷,这些错误样例很可能包含某些特殊的信息(如某种自然噪声),导致神经网络模型无法从中有效提取特征.因此,本文提出利用已有的错误样例来训练一个分类器.这个分类器的任务是确定未来的输入是否与历史上的错误样例相似,从而判断神经网络当前的输出是否可靠,并决定是否需要补丁的介入.

由于故障定位中利用神经网络插桩获得了模型运行时的内部状态,并计算出了神经网络模型在正确样例和错误样例上的费雪信息值.这些数据可以构成一个数据集来训练一个回归模型.该回归模型同神经网络插桩的探针类似,以神经网络层段的输出为输入,并预测该层段在输入样本上的费雪信息值.具体而言,本文选用了一层全连接层作为回归模型.给定输入样本 x,回归模型需要预测其在每个模型层段中的费雪信息值:

$$p^{FI}(S_i,x)=\mathrm{ReLU}(W_i^E\times O_i) \tag{14}$$

其中 $p^{FI}(S_i,x)$为回归模型预测的模型层段 i 在输入 x 上的费雪信息值,$\mathrm{ReLU}(\cdot)$为线性整流函数,$W_i^E\in\mathrm{R}^{c\times 1}$ 为回归模型的参数,其中 c 为输入的纬度,O_i为神经网络模型第 i 层段的输出.

回归模块的训练方法如算法 3 所示.在训练时,其损失函数为

$$\min_{W_i^E}\frac{1}{N}\sum_{i=1}^{N}(f(S_i,x)-p^{FI}(S_i,x))^2 \tag{15}$$

其中 $f(S_i,x)$为利用式(6)计算出的神经网络模型层段在输入样本为 x 时的费雪信息值,$p^{FI}(S_i,x)$为回归模型预测的对应的费雪信息值,N 为模型的总层段数量,本文在训练时采用错误样例和与错误样例数量相同的普通样例训练该回归模型.

算法 3. 故障预测回归模型训练

输入:待修复的神经网络模型 θ 及其各层段 $S_1\sim S_N\in\theta$,有标签训练集 D_t,错误样例 D_e,回归模型训练轮次 E

输出:回归模型 W_i^E

1. 初始化模型探针参数 $W_1^E\sim W_N^E$
2. 从 D_t中抽样$|D_e|$样本,构成数据集 D_t'
3. FOR $e=1,2,\cdots,E$ DO
4. FOR $x\in D_e+D_t'$ DO
5. 根据式(6)和式(14)计算 $f(S_i,x)$和 $p^{FI}(S_i,x)$
6. 根据式(15)计算目标函数
7. 反向传播更新模型探针参数 $W_1^E\sim W_N^E$
8. END FOR
9. END FOR

若回归模型预测的神经网络模型各层段费雪信息值总和超过一个预先设定的阈值 σ(即$\sum_i p^{FI}(S_i,x)>\sigma$),则代表神经网络模型在当前输入样本 x 上的输出是不可靠的,可能出现错误,需要补丁在第 $\arg\max_i(p^{FI}(S_i,x))$ 个层段上对其运算进行干预,计算过程如同式(12)所示.

在引入了故障预测模块后,补丁仅在预测为错误样例的输入中激活,以干预神经网络的前向传播.这一设计旨在最大限度地减少对正确样本推理的影响,从而避免可能引发灾难性遗忘等问题.整个神经网络的推理过程遵循如算法 4 所述的流程.

算法 4. 基于故障预测模块和补丁模块的神经网络推理

输入:神经网络模型 θ,神经网络补丁 $K^{\varphi'}(\cdot)$,故障预测模块 W_i^E,故障预测阈值 σ,输入样本 x

输出:推理结果

1. 输入x,得到神经网络层段输出$O_1\sim O_N$及最终输出p
2. 将O_i喂入故障预测模块,根据式(14)计算 $p^{FI}(S_i,x)$
3. IF $\sum_i p^{FI}(S_i,x)\leq\sigma$ //预测为正常样本
4. 输出 p
5. ELSE //预测为错误样本
6. 定位故障层段 $j=\arg\max_i(p^{FI}(S_i,x))$
7. 计算 $O_j'=O_j+K^{\varphi'}(O_{j-1})$ //打补丁
8. 以 O_j'为新输入,重复网络层段 j 后的所有计算并得到修复后的输出 p'
9. 输出 p'
10. END IF

4 深度边缘计算修复部署策略

在本节中,我们将讨论将第 3 节中的神经网络修复算法应用于深度边缘计算的全过程,包括从训练到修复的各个环节.如图 1 所示,深度边缘计算通常包含一个算力、存储、能源资源丰富的数据中心端和实际部署的轻量级边缘端.在修复深度边缘计算应用中的神经网络时,需要着重考虑三个指标:传输数据和更新模型时的通信开销;训练神经网络模块时的计算开销;以及整个修复流程花费的时间开销.如何高效地进行神经网络的训练、修复和部署,是深度边缘计算需要解决的核心问题之一.

4.1 训练和部署阶段

在边缘计算中,边缘节点通常是轻量级设备,其算力、存储和能耗资源有限.因此,神经网络模型通常在数据中心端利用预先收集到的数据集进行初始训练.由于数据中心资源丰富,这部分的计算开销往

往可以忽略不计. 此阶段的通信开销主要是将训练好的模型传输到边缘端时所涉及的传输开销. 同时，针对本文提出的修复算法，可以在神经网络模型训练后立即按照算法 1 的 1～8 行，进行神经网络插桩. 利用预先收集的数据集训练模型探针，并将探针一并发送至边缘端. 需要注意的是，探针 t 模型为简单的全连接模型，其体量远小于原神经网络模型. 因此，这一步骤的额外的通信开销即为传送探针所需的传输开销 $\sum_i |W_i^P|$.

在模型部署阶段，本文提出的方法需要利用边缘端所处环境的输入样本来搜索一个适应环境中潜在自然噪声的合适补丁结构. 然而，在初始状态下，补丁的运算池 $\mathcal{G}$ 较大，训练运算池中的运算所含有的参数对训练设备的计算资源要求较高，这在边缘环境中通常是不可行的. 因此，考虑在部署后，将边缘环境中的输入样本打包传输至数据中心端，按照算法 2 进行补丁结构搜索. 获得合适的补丁后，再将其传输回边缘端. 由于在数据中心进行补丁结构搜索，该部分的计算开销可忽略不计. 该阶段的通信开销主要为将边缘端的输入数据传输至数据中心时的开销 $|D_u| \times img\_size$，以及将搜索得到的补丁传输回边缘端的开销. 由于本文采用所有神经网络层段共用一个补丁的方式，搜索后得到的补丁的大小（$|W|, W \in \mathcal{G}'$）将远小于神经网络模型，传输补丁部分的开销相对较小.

需要注意的是，无论是神经网络插桩还是补丁结构搜索都不需要错误样例. 因此，一旦神经网络模型部署至边缘后，就可以立刻开始利用环境数据来搜索补丁的结构，这不涉及模型修复，因此并没有相关的时间开销. 而一旦搜集到错误样例，可直接利用搜索到的补丁结构来对神经网络进行相应的修复.

4.2　模型修复及修复后的部署阶段

由于补丁是一个轻量级的模型，其前向传播和训练所需的存储和计算资源很少，这使得其训练既可以在数据中心进行，也可直接在边缘端执行.

4.2.1　边缘端修复

当在边缘端利用补丁对神经网络模型进行修复时，可直接利用搜集到的错误样例依据式(13)和式(15)来训练补丁以及故障预测模块中的回归模型. 然而，需要注意的是，在模型修复后，神经网络的运行会依据故障预测的结果来判断补丁是否需要介入模型前向传播. 在实际运行中，可能会有普通样本被故障预测模块错误判断为错误样例，这种情况下，补丁的输入将是一个普通样本. 若是补丁训练阶段只使用搜集到的错误样例的话，可能导致其只学习到如何针对错误样本提取有效信息，而不能正确地从普通样本上抽取信息. 这将导致在实际部署后，神经网络在某些原本可以正确分类的样本上，因为补丁的错误介入而转为输出错误的结果. 针对这个问题，一个解决方案是在补丁的训练阶段引入普通样本，在实际操作时，可利用训练神经网络时预先收集的普通样本 D_t 对补丁进行训练. 在边缘端进行修复时，可在数据中心端利用故障预测模块对原训练数据进行过滤，将被故障预测模块判断为错误样例的普通数据传输至边缘端，加入在边缘端收集到的错误样例 D_e 中，用这些数据一起训练补丁. 在这种训练方式中，补丁可以接触到容易被故障预测模块误判的正确样本，从而学习到在这类样本上如何进行特征提取，以避免造成神经网络模型在普通样本上的精度的下降. 在边缘端修复神经网络模型时，其主要通信开销为从数据中心传输过滤的普通样本的开销和将故障预测模块传输回数据中心时的开销 $\sum_i |W_i^E|$. 其计算开销主要为利用式(13)训练补丁以及利用式(15)训练故障预测模块中的回归模型时的开销. 其时间开销主要用于相应模块在边缘设备上进行训练，取决于边缘设备的算力.

4.2.2　数据中心端修复

当在数据中心端对神经网络进行修复时，由于补丁结构本就在数据中心端搜索获得，因此只需要将错误样本从边缘端传输回数据中心，即可对补丁和故障预测模块进行训练. 在训练时，可利用预先收集好的普通样本与传输至数据中心的错误样本结合来训练补丁和故障预测模块. 在数据中心端修复的主要通信开销为发送错误样本至数据中心的开销以及训练好后，将补丁（$|W|, W \in \mathcal{G}'$）和故障预测模块中的回归模型（$\sum_i |W_i^E|$）发送至边缘端的开销. 在数据中心端，模型训练的计算开销可以忽略不计. 修复过程的时间开销主要为模型训练的开销，在本文实验部分第 5.5 节中有进一步的对比分析. 然而，在数据中心端修复可能造成模型更新的延迟，且当边缘端地理分布较广时，多任务处理可能造成数据中心任务分配和维护的困难.

深度神经网络模型在边缘端部署时，从训练到修复的全部流程如算法 5 所示.

算法 5.　深度边缘计算修复部署算法

输入：神经网络模型 θ，有标签数据集 D_t，无标签环境数据集 D_u，错误样例数据集 D_e，各模块的训练周期

输出：依据部署环境修复后的神经网络及其推理过程

//训练阶段（数据中心端）

1. 利用有标签数据 D_t 训练神经网络
2. 利用算法 1 的 1 至 8 行进行神经网络插桩
//部署阶段(数据中心端)
3. 利用算法 2 搜索补丁结构
//修复阶段(数据中心端、边缘端皆可)
4. 利用算法 3 训练故障预测模块
5. 利用故障预测模块过滤 D_t,筛选被误判的数据组成 D_t'
6. FOR $e=1,2,\cdots,E$ DO //训练补丁
7. FOR $(x,y)\in D_e+D_t'$ DO
8. 根据式(12)计算目标函数
9. 反向传播更新补丁参数
10. END FOR
11. END FOR
//修复后的部署阶段(边缘端)
12. 利用算法 4 计算模型推理的结果

5 实验分析

本节将利用真实世界的图像数据集进行实验验证,这些图像包含了各种现实生活中可能存在的自然噪声.这些实验的目的是评估神经网络在深度边缘计算中的应用,如自动驾驶和安防监控;以及在实际部署中可能面临的挑战.我们希望通过这些实验,验证本文提出的修复算法的有效性和可行性.

5.1 实验数据

我们首先利用公开数据集 CIFAR-10 和 Tiny-ImageNet 的训练集作为预先收集的数据集 D_t.具体来说,我们在 CIFAR-10 上训练了:AllConvNet[39]、Wide-ResNet(WRN-40-2)[40]和DenseNet[41]三个神经网络;在Tiny-ImageNet上训练了MobileNet-V2[42]网络.本文首先采取通用的方法[22],向原数据集的测试集注入了 15 种在现实生活中存在的自然噪声,模拟了复杂的真实世界环境.这些噪声分别为:高斯噪声(GN)、散粒噪声(SN)、脉冲噪声(IN)、离焦模糊(DN)、玻璃模糊(GN)、运动模糊(MB)、缩放模糊(ZB)、雪花噪声(SO)、霜冻噪声(FR)、雾气噪声(FOG)、亮度噪声(BR)、对比度噪声(CO)、弹性变形(ET)、像素化(PL)和 JPEG 压缩噪声(JC).每种注入的噪声都包含 5 个严重等级,等级越严重,注入的噪声越大.这些噪声模拟了计算机视觉领域中边缘智能任务在真实世界部署时可能面临的挑战.

实验中,为了模拟真实部署环境中的无标签输入,我们随机抽样了一半的有噪声图像作为 D_u.另一半图像被用于测试已训练好的神经网络模型.基于测试结果,我们进一步从测试过的图像中抽取一小部分错误样例,用以模拟采集到的训练样例 D_t(其中,CIFAR-10 选取 100 个,Tiny-ImageNet 选取 2000 个).其他的错误样例则被保留作为测试集,用于评估模型修复后的效果.测试集的详细情况见表 1.

表 1 测试数据集中的错误样例数

噪声种类	CIFAR-10			Tiny-ImageNet
	AllConvNet	WRN-40-2	DenseNet	MobileNet-V2
GN	6960	4894	5609	17065
SN	4859	3557	3978	15738
IN	5381	2935	2477	16686
DB	1762	1252	1093	15765
GB	6761	4319	4896	16400
MB	2443	1704	1514	13815
ZB	1975	1516	1409	15095
SO	3255	2230	2153	13595
FR	3647	2326	2285	13068
FOG	2732	1750	1708	13301
BR	1613	1177	1110	12474
CO	5403	1874	1909	17441
ET	2523	2215	2128	13162
PL	3312	2854	3177	11935
JC	3386	2904	2966	11876

同时,为了进一步验证本文所提方法在实际应用场景中的效果,我们搜集并构建了一个包含噪声的真实图像数据集,并进行了相应的实验.该数据集中的图像来源于 LAION-400M 数据集[43],图像的类别和 CIFAR-10 数据集中的十个类别相同.LAION-400M 是一个大型的图文多模态数据集,包含了超过 4 亿个真实图像及对应的描述文本对.我们通过特定的筛选方法从 LAION-400M 中获取了这些图像:利用 Clip Retrieval 模型[44],以关键字“cls with (snow/rain/fog…)”在 LAION-400M 数据集进行检索,其中 cls 替换为 CIFAR-10 中的类别名称.从检索结果中,我们筛选出共 4664 个含有噪声的真实图像,以此构成数据集.如图 4 所示,与之前加入模拟噪声的合成图像不同,这个数据集中的图像是真实场景下的有噪声图像,用以检验本文所

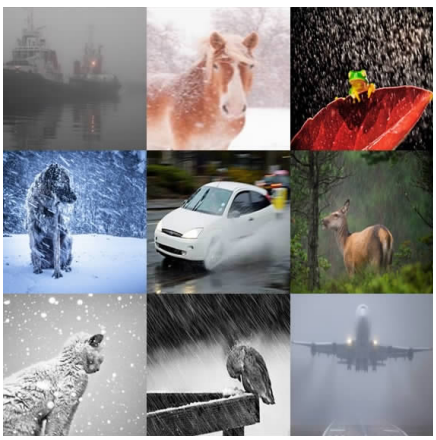


图 4 含有噪声的真实图像

提方法在实际应用场景中的效果.

5.2　算法实现

基于算法 5，我们实现了在数据中心端对神经网络进行修复的算法. 表 2 展示了实验中所用的数据、超参及训练设置. 我们设置补丁的基础单元的节点数为 4. 搜索结构时，本文采用三个最常见的无监督任务来对初始补丁进行训练，分别为：预测图像旋转角度、图像上色和求解图像分割后的拼图问题(jigsaw puzzle)[36].

表 2　训练数据、轮次及优化器设置

步骤	训练模块	所用数据	训练轮次(epoch)		优化器：SGD
			CIFAR-10	Tiny-ImageNet	
训练神经网络	神经网络	原始数据集	300	200	动量＝0.9 权重衰减＝5×10^{-4} 学习率＝0.01 余弦退火学习率
神经网络插桩	探针	原始数据集	30	10	
补丁结构搜索	初始补丁	边缘环境数据	20	10	
训练故障预测模块	回归模型	错误样例＋部分原始样例	800	20	
神经网络修复	搜索得到的补丁	错误样例＋过滤后的原始样例	50	20	

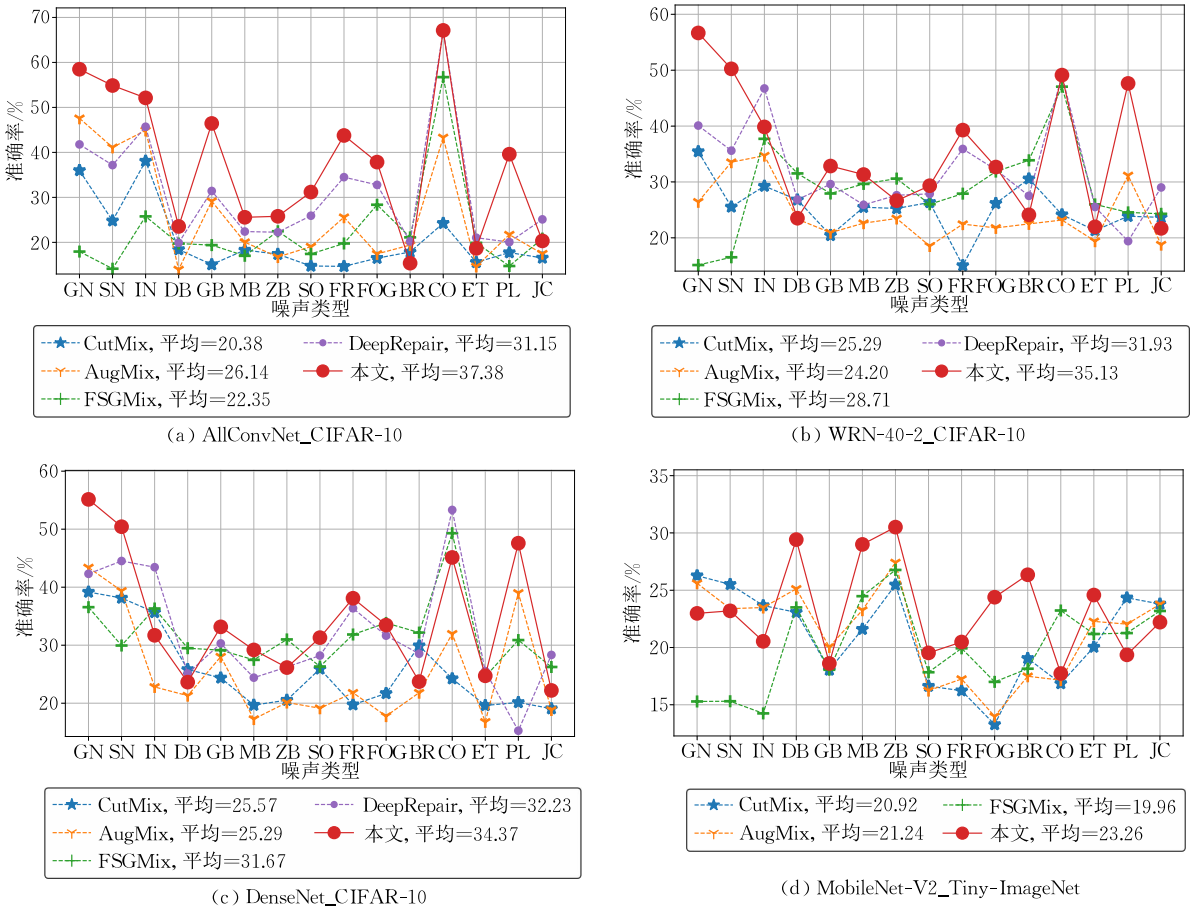
5.3　基准算法

本文选取了两类基准算法与本文所提的修复方法进行对比：一类为对错误样本进行数据增强后微调原网络的方法(CutMix[28] 和 AugMix[29])；另一类为现有的神经网络修复算法(FSGMix[8] 和 DeepRepair[6]). 在微调类方法中，收集到的错误样本和原数据集一齐用于微调神经网络. 对于已有的修复算法，错误样本和原数据集依照其原文中的具体方法来训练神经网络.

5.4　性能对比

5.4.1　边缘环境中包含单一自然噪声的修复

在神经网络部署的边缘端存在某一种自然噪声时，我们用基准算法及本文提出的修复算法分别对神经网络进行了修复，并利用原始神经网络的错误样例(排除了训练样例)来测试修复后神经网络的准确率，所有实验均重复进行了 3 次，平均结果如图 5 所示. 在两个数据集上的四个神经网络模型中，本文提出的方法均取得了更高的准确率. 特别值得注意的是，在



(a) AllConvNet_CIFAR-10　(b) WRN-40-2_CIFAR-10

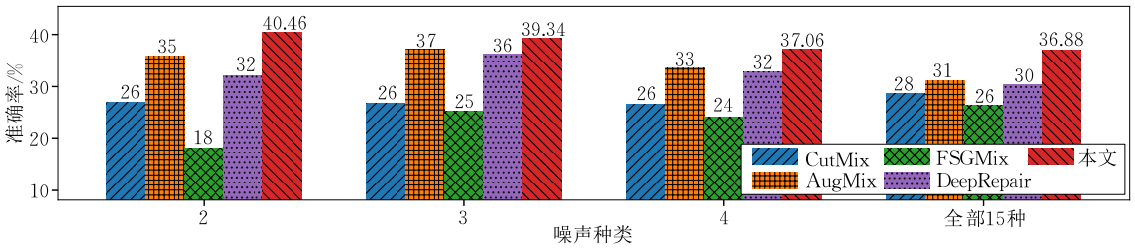
(c) DenseNet_CIFAR-10　(d) MobileNet-V2_Tiny-ImageNet

图 5　环境中存在单一噪声时，神经网络修复后，在错误样例测试集上的准确率

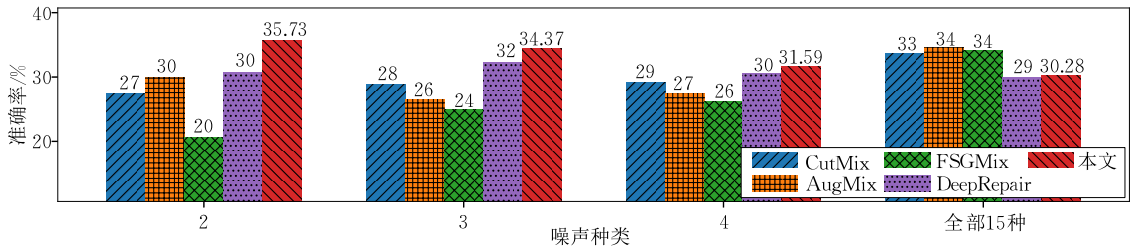
那些更易导致原始神经网络出错的自然噪声场景下，本文方法的修复效果更加显著. 例如，在高斯噪声和玻璃模糊等噪声类型上，本文方法的修复后准确率表现非常出色. 某些自然噪声可以从数据的角度，通过数据增强的方式来进行有效地修复(如亮度噪声)，这些噪声本身对于轻量级的补丁模型结构的敏感性相对较低. 相比于在 CIFAR-10 数据集上表现最好的基准算法(DeepRepair)，本文所提方法的平均准确率提高了 6.64%至 20.00%；在 Tiny-ImageNet 上(AugMix)，提升幅度达到了 9.51%. 这些实验结果表明，本文提出的方法在修复边缘端神经网络模型时具有显著的优势，能够提高模型的分类准确性.

5.4.2 边缘环境中包含多种自然噪声的修复

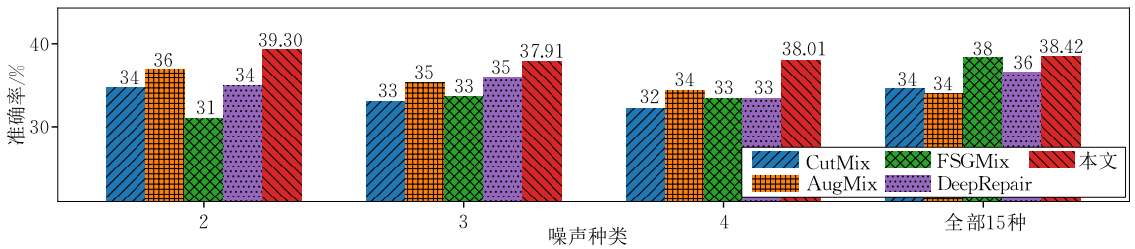
我们测试了当部署环境中含有多种自然噪声时，各种神经网络修复算法的性能，结果如图 6 所示. 实验结果显示，本文提出的方法在大部分情况下取得了最好的结果. 与其他基准算法相比，本文的方法的准确率提升幅度约为 25%. 这表明，当部署环境极其复杂，含有的自然噪声种类较多时，用单一结构的补丁进行修复的效果会有所下降. 但本文提出的方法在噪声种类数量适中时的结果均高于基准算法. 这些实验结果表明，本文的方法在复杂的噪声环境中能够更好地修复边缘端的神经网络模型，提高其分类准确性.



(a) AllConvNet_CIFAR-10



(b) WRN-40-2_CIFAR-10



(c) DenseNet_CIFAR-10

图 6 环境中同时存在多种噪声时，神经网络修复后，在错误样例测试集上的准确率

5.4.3 在真实图像中的修复

本文在有真实噪声图像的 CIFAR-10 的数据集上预训练的三个模型：AllConvNet、WRN-40-2 和 DenseNet 的准确率分别为 54.44%、59.35% 和 58.85%. 针对这一数据集，我们对模型进行了修复，并得到了相应的实验结果，结果如图 7 所示. 由于真实图像在原始尺寸等数据格式上与模型预训练时的训练数据有一定差异，且这些噪声多为雪、雾、光照等第 5.4.1 节中较难修复的天气类噪声，各修复算法在真实图像上的修复后准确率较模拟图像上的均有一定下降. 但本文所提的修复算法在三个网络上均取得了更好的修复效果. 这进一步验证了本文所

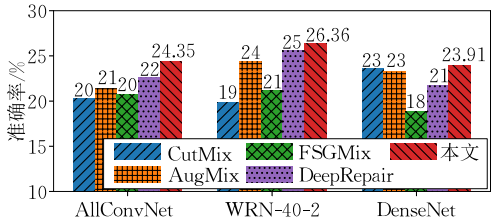


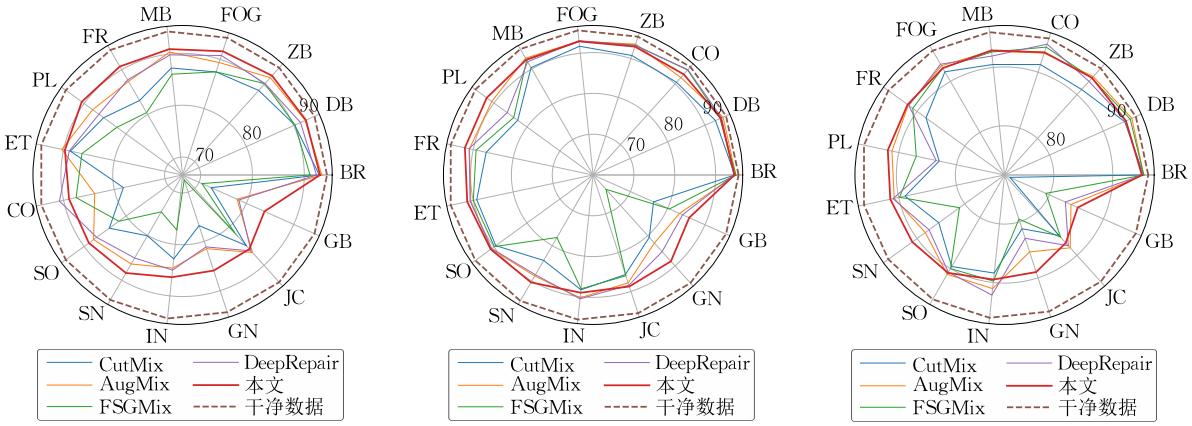
图 7 在含有噪声的真实图像中，神经网络修复后，在错误样例测试集上的准确率

提算法在实际应用场景中的有效性.

5.4.4　修复后模型的鲁棒性

为了测试修复后的神经网络模型的鲁棒性，我们利用干净测试数据和有噪声测试数据（排除用于修复模型时使用的数据）对各种修复算法修复后的模型进行了测试. 实验结果如图 8 所示，虽然在测试图像中可能含有噪声时，修复后的神经网络的准确率仍有一定程度的下降，但本文提出的方法所修复的模型总体上表现最好.

值得注意的是，随着神经网络准确率的下降，本文方法与其他方法之间的性能差距反而逐渐增大. 这证明通过设计一个更适合部署环境的补丁结构，可以很好地弥补原神经网络模型结构在特定部署环境中的不足之处. 实验结果显示，本文的方法在修复后的模型上展现出更好的鲁棒性，能够更好地应对带有噪声的测试数据. 这意味着修复后的模型在真实环境中可能会更加稳定和可靠，能够更好地处理噪声干扰，从而提高边缘计算系统的整体性能和可靠性.



(a) AllConvNet_CIFAR-10　(b) WRN-40-2_CIFAR-10　(c) DenseNet_CIFAR-10

图 8　修复后神经网络模型的鲁棒性评估

5.5　修复与部署开销分析

5.5.1　修复开销

如表 2 所示，本文提出的方法中一共有 5 个模块需要进行训练. 其中，神经网络、探针和初始补丁的训练不需要错误样例，这些模块可在部署前及部署后立即进行训练. 而当错误样例在边缘设备的运行过程中被发现并收集后，本文提出的方法只需训练故障预测模块中的回归模型，并利用补丁修复原始神经网络. 我们比较了基准算法及本文提出的神经网络修复算法在模型出现错误需要修复时，平均所需要的修复时间，结果如图 9 所示. 与多个需要数小时训练的基准修复算法相比，本文提出的算法在获取错误样例后，只需约 10 min 的时间即可对模型进行修复. 这表明本文的方法具有更快的修复速度，适用于边缘环境下快速响应模型修复需求的场景.

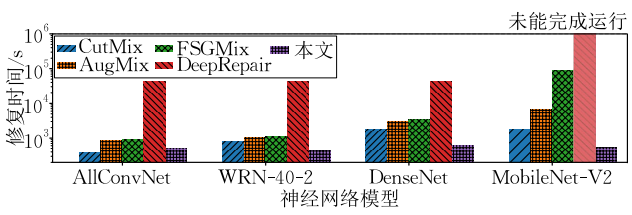


图 9　修复算法时间对比

同时，我们比较了各修复算法在修复神经网络时，所使用的训练样本数量，结果如表 3 所示. 基准算法在修复网络时，需要将错误样例与原始数据集一起用于微调神经网络，而本文所提的方法只需要使用错误样例和少量原始数据集中的样本. 这不仅大大减少了本文方法的修复时间，还降低了在边缘设备上修复模型时的存储和计算开销. 本文的方法通过精确选择训练样本，有效地减少了修复过程中所需的数据量，同时保持了修复的准确性. 这为在资源受限的边缘环境下进行高效的模型修复提供了重要的优势.

表 3　修复算法所用训练样本数量对比

模型	数据集	修复所需样本数量		样本量减少比例/%
		基准算法	本文	
AllConvNet	CIFAR-10	50 100	2847	94.32
WRN-40-2			2675	94.66
DenseNet			1682	96.64
MobileNet-V2	Tiny-ImageNet	102 000	5495	94.61

5.5.2　部署开销

由于基准算法均需要直接更新原始神经网络模型的参数，在数据中心进行神经网络的修复时，需要传输修复后的模型至边缘端. 相比之下，本文提出

的方法无需更新原始神经网络模型，只需要将用于故障预测的探针、回归模型和补丁传输至边缘端. 表 4 展示了这两种更新方式所需要传输的模型的参数量的对比.

表 4 修复算法所用训练样本数量对比

模型	基准算法参数量/M	本文参数量/M	参数量减少比例/%
AllConvNet	1.41	0.19	86.52
WRN-40-2	2.24	0.17	92.41
DenseNet	0.77	0.18	76.62
MobileNet-V2	2.48	0.20	91.94

与基准算法相比，本文的修复算法所需更新的参数量最高减少了 91.94%. 这种轻量级的更新方式不仅有利于修复模型的快速部署，减少了维护成本，同时也提高了神经网络服务的可用性. 通过仅更新修复模块的参数，本文方法降低了修复过程中所需的通信开销，同时保持了原始神经网络模型的稳定性和性能.

5.6 模块分析

5.6.1 补丁无监督结构搜索有效性分析

由于数据标记的成本高昂，在部署环境中，通常无法获取足够的标记数据来搜索合适的补丁结构. 因此，本文采用无监督任务来利用环境中的输入数据搜索补丁结构. 为了证明搜索得到的补丁在无监督任务中取得良好性能时，其抽取的特征同样也有助于最终的有监督分类任务，我们进行了如下的实验. 我们从初始补丁中随机抽取了 100 个子结构，并分别对这些补丁进行无监督任务和有监督分类任务的训练. 具体而言，表 2 中的边缘环境数据被用于对补丁进行无监督训练. 然后，这些数据的标签将被添加回去，重新训练了补丁. 训练轮次均为 5. 每个补丁在两类任务上的准确率绘制在图 10 中，其中直线为使用鲁棒线性回归拟合数据点后的结果.

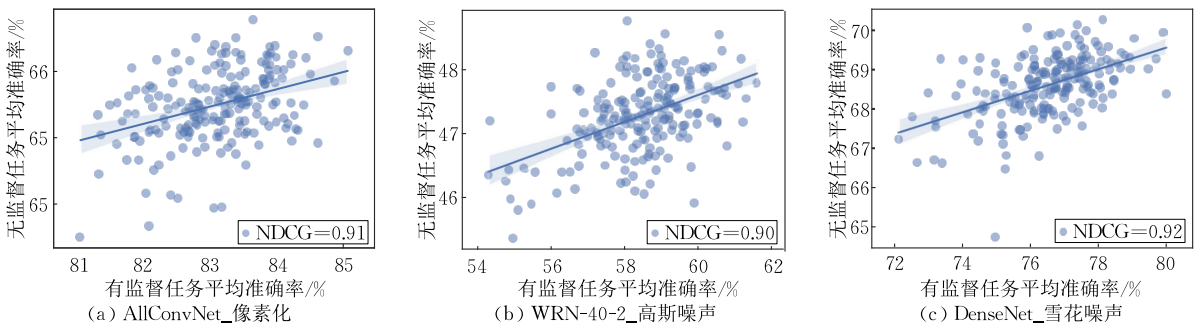


图 10 补丁结构分别经有监督和无监督任务训练后的准确率

从图中可以观察到，补丁在两个任务上的准确率成正相关趋势，线性回归拟合的曲线斜率均为正. 我们还计算了两个任务的准确率之间的标准化折扣累计增益 *NDCG*(Normalize Discounted Cumulative Gain)[45]. 该指标用于衡量排序结果(ranking)之间的相关性，其值越接近于 1，则两个任务的准确率排序越一致. 可以看出，所有实验的 *NDCG* 值都不低于 0.9. 这可以证明在无监督任务上表现出色的补丁结构往往也能在最终的分类任务中取得好的效果，即它能够从相应环境输入中提取出有效特征.

5.6.2 故障预测模块有效性分析

为了探究故障预测模块在不同阈值 σ 下对错误和正确样例的区分能力，图 11 展示了故障预测模块的 ROC 曲线以及对应的 *AUC* 值. 本文所用故障预测的方法可以较精准地预测可能的错误样例，大多数情况下，其 *AUC* 值均高于 0.9. 即使在最不利的情况下，例如在 AllConvNet 模型遭遇亮度噪声时，其 *AUC* 值仍然达到了 0.8. 这证明了故障预测模块的有效性和稳健性.

5.6.3 神经网络修复后的混淆矩阵对比

针对在部署了 AllConvNet 网络的环境中存在高斯噪声的情况，图 12 展示了图 5 中准确率最高的基准算法 AugMix 修复后的神经网络模型和用本文所提的方法修复后的神经网络模型的混淆矩阵. 如图 12 所示，在存在高斯噪声的情况下，用基准算法修复后的神经网络仍容易将不同种类的动物(如猫和狗)混淆. 尽管本文所提的算法也有相同的趋势，但是其修复后的神经网络在大多数类别(特别是飞机、狗和马)上的性能显著提高. 这说明本文所提的算法能够改善神经网络的分类准确性.

6 结论及展望

本文提出了一种基于补丁的神经网络修复方

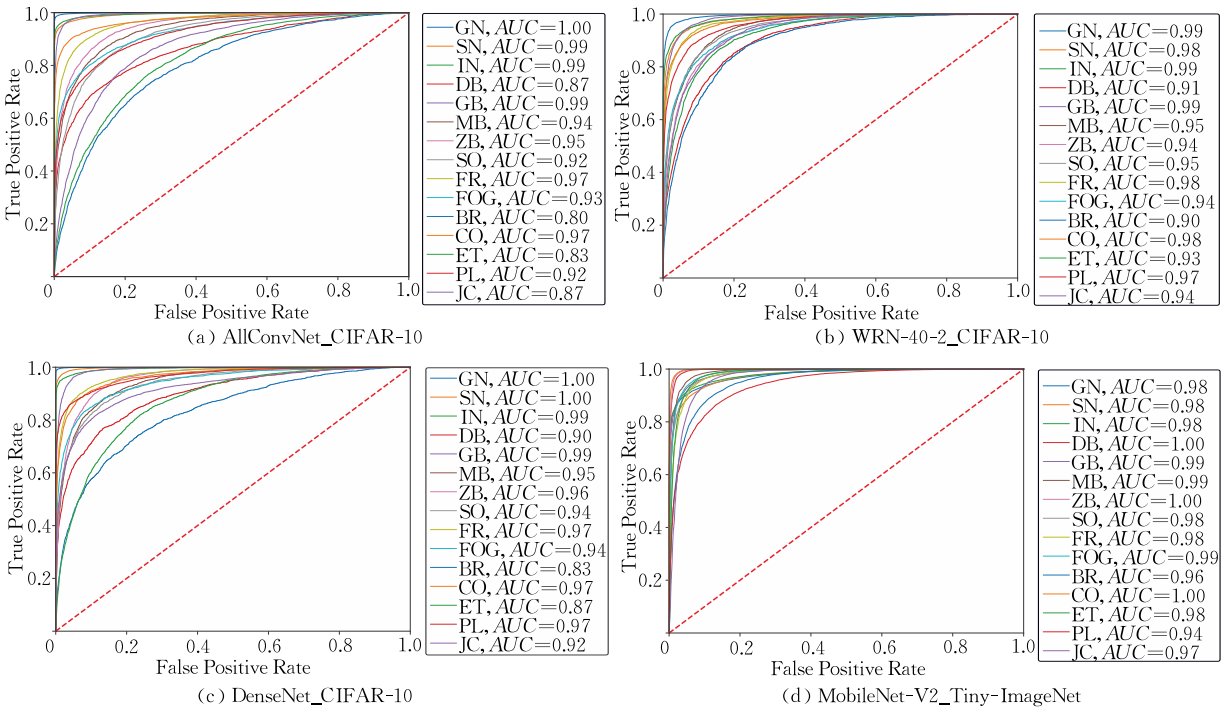


图 11　故障预测模块的 ROC 曲线

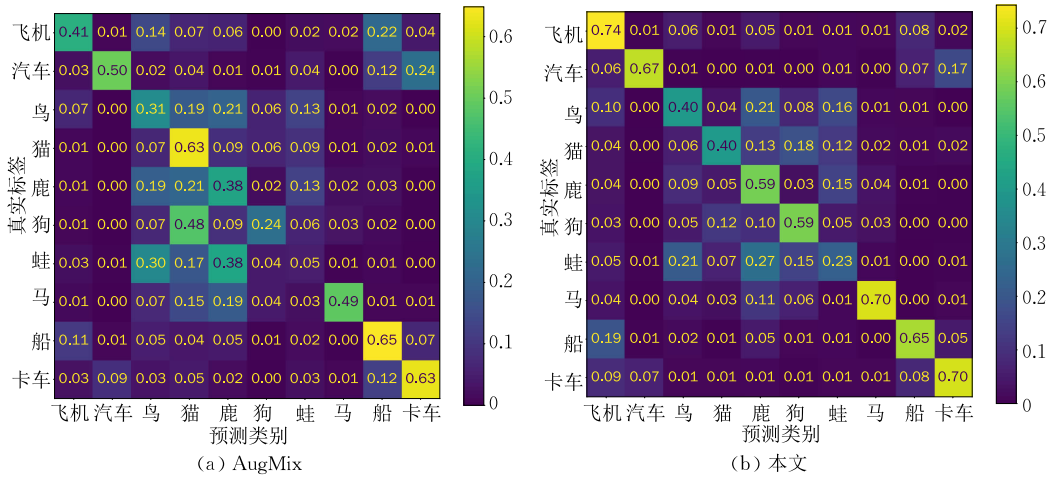


图 12　修复后的 AllConvNet 网络在高斯噪声输入上的混淆矩阵

法，旨在快速高效地修复部署在边缘节点上的模型. 与现有的修复算法需要直接训练和更新原始神经网络不同，本文的方法通过给模型打补丁的方式，能够快速高效地修复部署中的神经网络. 修复过程分为故障定位、故障修复和故障预测三个步骤. 该方法的优点在于可以为边缘节点提供个性化的修复服务，并避免了对神经网络模型进行更新造成的服务中断. 通过多组实验证明，相比现有的神经网络修复算法，本文提出的方法可以获得更好的修复效果，同时大大减少边缘场景下模型修复所需的训练开销和部署开销.

基于本文方法的特点，我们分析了两个潜在应用场景：(1) 安防监控摄像头场景. 该场景中，由于环境因素如天气、位移等可能导致输入图像含有潜在噪声，进而影响模型的输出[23]. 本文提出的修复方法可应用于此类场景，为快速变化的环境中的神经网络模型提供应急修复服务；(2) 光纤故障检测场景：卷积神经网络广泛应用于分布式光纤传感系统中，例如对电力、能源系统中的局部放电、管道泄漏等故障进行事件检测和分类[46-47]. 然而，传感器所处环境中的噪声(如光线、地质条件、动物活动等)可能影响成像[48]，导致模型分类结果出错. 本文提出的修复方法适用于此类场景，为复杂环境中的神经网络模型提供个性化修复服务. 未来，我们也将进一

步探讨其他潜在的应用领域,同时深入研究并在实际应用场景中进行进一步的验证.

参 考 文 献

[1] Eykholt K, Evtimov I, Fernandes E, et al. Robust physical-world attacks on deep learning visual classification//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Salt Lake City, USA, 2018: 1625-1634

[2] Schwinn L, Raab R, Nguyen A, et al. Exploring misclassifications of robust neural networks to enhance adversarial attacks. Applied Intelligence, 2023, 53(17): 19843-19859

[3] Zheng X, Fan Y, Wu B, et al. Robust physical-world attacks on face recognition. Pattern Recognition, 2023, 133: 109009

[4] Hua H, Li Y, Wang T, et al. Edge computing with artificial intelligence: A machine learning perspective. ACM Computing Surveys, 2023, 55(9): 1-35

[5] Laskaridis S, Venieris S I, Almeida M, et al. SPINN: Synergistic progressive inference of neural networks over device and cloud//Proceedings of the 26th Annual International Conference on Mobile Computing and Networking. London, UK, 2020: 1-15

[6] Yu B, Qi H, Guo Q, et al. DeepRepair: Style-guided repairing for deep neural networks in the real-world operational environment. IEEE Transactions on Reliability, 2021, 71(4): 1401-1416

[7] Hendrycks D, Dietterich T. Benchmarking neural network robustness to common corruptions and perturbations//Proceedings of the International Conference on Learning Representations. New Orleans, US, 2019: 1-11

[8] Ren X, Yu B, Qi H, et al. Few-shot guided mix for DNN repairing//Proceedings of the 2020 IEEE International Conference on Software Maintenance and Evolution. Adelaide, Australia, 2020: 717-721

[9] Zhang H, Chan W K. Apricot: A weight-adaptation approach to fixing deep learning models//Proceedings of the 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). San Diego, USA, 2019: 376-387

[10] Sohn J, Kang S, Yoo S. Search based repair of deep neural networks. arXiv preprint arXiv:1912.12463, 2019

[11] Dodge J, Prewitt T, Tachet des Combes R, et al. Measuring the carbon intensity of AI in cloud instances//Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency. Seoul, Republic of Korea, 2022: 1877-1894

[12] Kurakin A, Goodfellow I J, Bengio S. Adversarial examples in the physical world//Artificial Intelligence Safety and Security. Chapman and Hall/CRC, 2018: 99-112

[13] Huang J C. Program instrumentation and software testing. Computer, 1978, 11(4): 25-32

[14] Fang Y, Li W, Zeng Y, et al. FIREPruning: Learning-based filter pruning for convolutional neural network compression//Proceedings of the Asian Conference on Machine Learning. 2020: 385-400

[15] Wang Z, Li C, Wang X. Convolutional neural network pruning with structural redundancy reduction//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021: 14913-14922

[16] Wang L, Dong X, Wang Y, et al. Learnable lookup table for neural network quantization//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. New Orleans, USA, 2022: 12423-12433

[17] Zhang J, Zhou Y, Saab R. Post-training quantization for neural networks with provable guarantees. SIAM Journal on Mathematics of Data Science, 2023, 5(2): 373-399

[18] He Q, Dong Z, Chen F, et al. Pyramid: Enabling hierarchical neural networks with edge computing//Proceedings of the ACM Web Conference 2022. Lyon, France, 2022: 1860-1870

[19] Jia Z, Lin S, Qi C R, et al. Exploring hidden dimensions in parallelizing convolutional neural networks//Proceedings of the 35th International Conference on Machine Learning. Stockholm, Sweden, 2018: 2279-2288

[20] Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572, 2014

[21] Tu Y, Zhang B, Li Y, et al. Learning with noisy labels via self-supervised adversarial noisy masking//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Vancouver, Canada, 2023: 16186-16195

[22] Hendrycks D, Zhao K, Basart S, et al. Natural adversarial examples//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021: 15262-15271

[23] Zhong Y, Liu X, Zhai D, et al. Shadows can be dangerous: Stealthy and effective physical-world adversarial attack by natural phenomenon//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. New Orleans, USA, 2022: 15345-15354

[24] Tian Y, Pei K, Jana S, et al. DeepTest: Automated testing of deep-neural-network-driven autonomous cars//Proceedings of the 40th International Conference on Software Engineering. Gothenburg, Sweden, 2018: 303-314

[25] Tang S, Gong R, Wang Y, et al. RobustART: Benchmarking robustness on architecture design and training techniques. arXiv preprint arXiv:2109.05211, 2021

[26] Devaguptapu C, Agarwal D, Mittal G, et al. On adversarial robustness: A neural architecture search perspective//Proceedings of the IEEE/CVF International Conference on Computer Vision. Montreal, Canada, 2021: 152-161

[27] Yue Z, Lin B, Zhang Y, et al. Effective, efficient and robust neural architecture search//Proceedings of the 2022 International Joint Conference on Neural Networks. Padua, Italy, 2022: 1-8

[28] Yun S, Han D, Oh S J, et al. CutMix: Regularization strategy to train strong classifiers with localizable features//Proceedings of the IEEE/CVF International Conference on Computer Vision. Seoul, Korea, 2019: 6023-6032

[29] Hendrycks D, Mu N, Cubuk E D, et al. AugMix: A simple data processing method to improve robustness and uncertainty //Proceedings of the International Conference on Learning Representations. 2020: 1-11

[30] Fang Y, Li W, Zeng Y, et al. PatchNAS: Repairing DNNs in deployment with patched network architecture search// Proceedings of the AAAI Conference on Artificial Intelligence. Washington, USA, 2023, 37(12): 14811-14819

[31] Cramér H. Mathematical Methods of Statistics. Princeton, USA: Princeton University Press, 1999

[32] Xu R, Luo F, Zhang Z, et al. Raise a child in large language model: Towards effective and generalizable fine-tuning// Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2021: 9514-9528

[33] Liu H, Simonyan K, Yang Y. DARTS: Differentiable architecture search//Proceedings of the International Conference on Learning Representations. New Orleans, USA, 2019: 1-11

[34] Ren P, Xiao Y, Chang X, et al. A comprehensive survey of neural architecture search: Challenges and solutions. ACM Computing Surveys, 2021, 54(4): 1-34

[35] Chu X, Lu S, Li X, et al. MixPath: A unified approach for one-shot neural architecture search//Proceedings of the IEEE/CVF International Conference on Computer Vision. Paris, France, 2023: 5972-5981

[36] Liu C, Dollár P, He K, et al. Are labels necessary for neural architecture search?//Proceedings of the European Conference on Computer Vision. 2020: 798-813

[37] Zhang X, Hou P, Zhang X, et al. Neural architecture search with random labels//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021: 10907-10916

[38] Bender G, Kindermans P J, Zoph B, et al. Understanding and simplifying one-shot architecture search//Proceedings of the International Conference on Machine Learning. Stockholm, Sweden, 2018: 550-559

[39] Springenberg J T, Dosovitskiy A, Brox T, et al. Striving for simplicity: The all convolutional net. arXiv preprint arXiv: 1412.6806, 2014

[40] Zagoruyko S, Komodakis N. Wide residual networks//Proceedings of the British Machine Vision Conference. York, UK, 2016: 87.1-87.12

[41] Huang G, Liu Z, Van Der Maaten L, et al. Densely connected convolutional networks//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Honolulu, USA, 2017: 4700-4708

[42] Sandler M, Howard A, Zhu M, et al. MobileNetV2: Inverted residuals and linear bottlenecks//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Salt Lake City, USA, 2018: 4510-4520

[43] Schuhmann C, Vencu R, Beaumont R, et al. LAION-400M: Open dataset of CLIP-filtered 400 million image-text pairs. arXiv preprint arXiv:2111.02114, 2021

[44] Radford A, Kim J W, Hallacy C, et al. Learning transferable visual models from natural language supervision//Proceedings of the International Conference on Machine Learning. 2021: 8748-8763

[45] Buckley C, Voorhees E M. Retrieval evaluation with incomplete information//Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Sheffield, UK, 2004: 25-32

[46] Duan Y, Liang L, Tong X, et al. Application of pipeline leakage detection based on distributed optical fiber acoustic sensor system and convolutional neural network. Journal of Physics D: Applied Physics, 2023, 57(10): 105102

[47] Che Q, Wen H, Li X, et al. Partial discharge recognition based on optical fiber distributed acoustic sensing and a convolutional neural network. IEEE Access, 2019, 7: 101758-101764

[48] Wang Fang, Xing Ji-Chuan. The influence of soil temperature on vibration signal in the optical fiber warning system. Optical Technique, 2016, 42(5): 420-423(in Chinese)
(王放，邢冀川. 土壤温度对光纤预警系统中振动信号的影响. 光学技术，2016，42(5)：420-423)

FANG Yu-Chu, Ph. D. candidate. His current major research interests include model compression, neural network on the edge, neural network robustness.

LI Wen-Zhong, Ph. D., professor. His main research interests include distributed computing, data mining, mobile cloud computing, wireless networks, pervasive computing, and social networks.

ZENG Yao, M. S. His main research interests include neural network compression, edge computing, and neural network robustness.

ZHENG Yang, Ph. D. His main research interests include testing, monitoring and repairing AI systems in ICT and automotive industries.

HU Zheng, Ph. D. His main research interests include software reliability, reliability theory and ah-hoc networks.

LU Sang-Lu, Ph. D., professor, Ph. D. supervisor. Her main research interests include distributed computing, pervasive computing, and wireless network.

Background

In recent times, there has been a notable increase in deploying Deep Neural Networks (DNNs) in typical edge computing environments. Given their wide-ranging applications in safety-critical fields like intelligent transportation, it has become crucial to address the robustness and reliability of DNNs. Typically, DNNs are trained using pre-collected datasets before being deployed in real-world edge environments. However, the runtime environment often contains unforeseen noises and scenarios. In complex edge deployment environments, neural network models are susceptible to producing erroneous output results. Consequently, finding ways to use error samples collected at the edge to repair deployed neural networks has become a crucial research focus. Existing neural network repair algorithms typically require training and updating the entire model, leading to increased training and communication costs in edge computing scenarios. Furthermore, the difficulty of repairing a single model significantly rises when edge nodes are geographically distributed and exhibit diverse environmental conditions.

In this study, we propose an efficient repair algorithm for DNNs on the edge. We first identify the faulty stage of the neural network for collected failure samples by employing neural network instrumentation and finding the stage with the maximum Fisher information value. Next, we introduce a patch, whose structure is discovered using unsupervised environment inputs, to be added to the faulty stage of the DNN to correct its output. Additionally, we incorporate a fault prediction module to anticipate potential failure samples for future inputs, allowing the patch to intervene during runtime inference. We analyze the training and communication costs for the entire proposed repair algorithm. Extensive experiments using 2 datasets with 4 network models demonstrate that our proposed method significantly outperforms other repair methods in terms of effectiveness and achieves excellent repair efficiency.

This work was partially supported by the Natural Science Foundation of Jiangsu Province (Project "Research on Frontier Basic Theory and Method of Security Defense for Power Systems with High-dimensional Uncertain Factors", Grant No. BK20222003), the National Natural Science Foundation of China (Grant No. 61972196), the Key Program of National Natural Science Foundation of China (Grant Nos. 61832008, 61832005).