摘要

燃气管道作为城市基础设施的重要组成部分，其焊接的安全性、稳定性和可靠性日益受到重视。然而对于焊接缺陷的检测却面临许多挑战：焊接接头的缺陷一般出现在管道的内侧，必须要通过超声波或 X 射线的方式进行探测，且需要具有专业知识与技能的工作人员对图像进行人工识别分类并分级，不仅效率低下还十分容易出现误判漏判。本文希望能够搭建一个系统，其不仅能基于超声相控阵系统完成缺陷的探测，还能够适应数据分布的不断变化，持续地优化已有模型。为了实现以上所提到的功能，笔者将对广泛应用于其他深度学习领域的持续学习方法在焊接缺陷检测的具体任务中进行评估，并寻找到能够适应燃气管道焊接缺陷检测场景且可以一定程度减少持续学习灾难性遗忘的算法。最后运用合适的算法搭建一个持续学习平台实现对模型的持续优化与训练。

所有的实验与软件平台代码可以在 https://github.com/KirigiriSuzumiya/CL_for_ImageSegmentation 获取。

关键词：持续学习，缺陷检测，机器视觉，深度学习

Abstract

As an important part of urban infrastructure, the safety, stability and reliability of gas pipeline welding have been paid more and more attention. However, the detection of welding defects faces many challenges: the defects of welded joints generally appear on the inside of the pipeline, which must be detected by ultrasonic or X-ray, and the staff with professional knowledge and skills are required to manually identify, classify and grade the images, which is not only inefficient but also prone to misjudgment and missing judgment. In this paper, we hope to build a system that can not only detect defects based on ultrasonic phased array system, but also adapt to the constant change of data distribution and continuously optimize the existing model. In order to realize the above mentioned functions, the author will evaluate the continuous learning methods widely used in other deep learning fields in the specific task of welding defect detection, and find an algorithm that can adapt to the welding defect detection scenario of gas pipeline and reduce the catastrophic forgetting of continuous learning to a certain extent. Finally, a continuous learning platform is built with appropriate algorithms to achieve continuous optimization and training of the model.

All the experiments and software platform code is available at https://github.com/KirigiriSuzumiya/CL_for_ImageSegmentation.

Keywords: Continual Learning, Defect Detection, Machine Vision, Deep Learning

目录

1 研究背景

随着城市化和工业化的快速发展，燃气管道作为城市基础设施的重要组成部分，其安全性、稳定性和可靠性日益受到重视。在燃气管道的施工过程中，焊接是一种常见的连接方式，然而由于各种因素的影响，焊接过程中可能会出现各种缺陷，如未熔合、裂纹、气孔等。这些缺陷可能导致燃气泄漏、管道破裂等安全事故，严重威胁人民生命财产安全。因此，对燃气管道焊接缺陷的检测和预防至关重要。

然而，焊接接头的缺陷一般出现在管道的内侧，无法通过肉眼观测。为了解决这一问题，目前大多使用 X 射线与超声波来进行无损检测。其中 X 射线多用于钢制管道，而超声波多用于 PE 管道的检测。虽然 X 射线或超声相控阵系统能够清晰展示管道内壁的情况，但是仍然需要具有专业知识与技能的工作人员对图像进行人工识别分类并分级，不仅效率低下还十分容易出现误判漏判。

近年来，随着人工智能和机器学习技术的快速发展，机器视觉和持续学习（Continuous Learning）作为重要的机器学习方法，逐渐在各个领域得到应用。在燃气管道焊接缺陷检测领域中，机器视觉技术可以代替人力完成对无损检测图像的识别，提高效率与准确度。然而，燃气管道焊接过程中产生的缺陷数据是动态变化的，随着时间的推移和技术的进步，缺陷的类型和分布可能发生变化，导致原有的机器视觉模型无法适应不断变化的场景。而从头重新训练一个能够适应当前分布的模型不仅会耗费大量时间与算力资源，还可能会使模型对之前缺陷的检测性能发生下降。

持续学习的应用旨在使机器视觉模型能够在不断接收新数据的情况下，持续更新和优化自身的性能，以适应环境的变化。在燃气管道焊接缺陷检测领域，持续学习方法可以在不重新训练整个模型的情况下，仅对新增数据进行学习，从而大大提高学习效率。这对于燃气管道焊接缺陷检测来说非常重要，因为在实际应用中，往往需要对大量的数据进行处理和分析，而持续学习可以在保证准确性的同时，大大提高处理速度，满足实际应用的需求。同时企业可以在不增加额外硬件和软件成本的情况下，实现对燃气管道焊接缺陷检测的不断优化。这不仅可以降低企业的运营成本，还可以减少因管道事故带来的经济损失和社会风险。

本文希望通过使用持续学习技术来扩展机器视觉模型在燃气管道焊接缺陷检测领域的实际应用场景，最后搭建一个能够提供自动化整理在线与离线数据并为视觉模型提供持续优化训练与评估服务的后端平台。为保障城市基础设施的安全稳定运行提供有力支持。

2 文献综述

2.1 图像缺陷检测

缺陷检测是一个重要的机器学习问题。与大多数现有机器学习方法所基于的静态和封闭系统的假设不同，它研究机器学习模型如何在开放和动态系统环境下处理未知和不确定的信息。在开放环境的假设下，为异常检测开发的学习系统通常期望利用已知（正常数据和模式）的知识来推断未知（与正常不同的异常或新模式）。异常检测方法通常使用可用的正常数据提取、表征和建模模式，然后开发合理的异常检测器来发现新观察到的数据中的新颖或异常模式。当异常检测的目标是图像数据时，就产生了视觉缺陷检测或图像异常检测的任务。

在视觉异常检测中，异常样本或模式可能以多种形式出现，如形状、颜色、纹理或运动模式的异常。这些异常可能由多种因素引起，如摄像头故障、物体异常行为或场景中的罕见事件。因此，视觉异常检测需要能够识别与正常模式显著不同的模式，即使它们可能非常复杂且难以预定义。

为了实现这一点，视觉异常检测方法通常依赖于特征提取和建模技术，以从正常数据中捕获关键信息，并构建能够区分正常和异常模式的模型。这些技术可能包括传统的图像处理技术（如滤波、边缘检测等）、深度学习技术（如卷积神经网络、自编码器等）或统计学习方法（如概率模型、密度估计等）。

此外，视觉异常检测还面临一些挑战，如异常模式的多样性和不可预测性、正常数据和异常数据之间的不平衡、以及缺乏足够的标记数据来训练模型。因此，研究人员一直在努力开发更强大和适应性更强的方法，以在各种场景中实现准确的视觉异常检测。视觉异常检测是一个活跃的研究领域，具有广泛的应用前景，包括安全监控、医疗诊断、工业质量控制等。随着技术的不断进步，笔者相信在未来能够看到更多创新和突破性的成果。

2.1.1 图像缺陷检测的分类与发展

从是否有监督信息（是否有异常样本或异常模式）的角度来看，视觉异常检测可以分为两个不同的研究方向：有监督和无监督的视觉异常检测。在本小节中，笔者将主要回顾解决监督视觉异常检测问题的方法。一方面，有监督的视觉异常检测推动了所提出的方法的发展，以开发与人类视觉能力相似的计算机视觉模型。另一方面，对于大多数实际应用场景来说，异常样本或异常模式通常在形状、颜色和大小上都是可变的，并且它们没有稳定的统计规律，并在发展的过程中不断变化。所有这些都会使模型难以在变化的数据分布中捕获足够的统计信息或关于异常图像模式的显著特征。

根据不同的视觉检测精确颗粒度，视觉异常检测可以分为两类：图像级和像素级视觉异常检测。其中，图像级检测通常只关注整个图像是否正常或异常，而像素级异常检

测则需要进一步检测或定位图像中的异常区域。

此外，根据视觉异常检测研究的历史发展，（包括图像级和像素级）异常检测的文献大致可以分为两个阶段：深度学习之前和深度学习之后。在深度学习提出之前，视觉异常检测的研究重点在于开发异常检测策略或机制。主要的研究问题是：在通过手工获得图像的浅层特征（如灰度值、SIFT[1]和 HOG[2]）后，尝试基于统计或传统机器学习方法（如密度估计、单类分类和图像重建）开发不同的检测机制。它首先估计正常图像或图像特征的分布模型。然后，如果图像或其特征不符合相应的分布模型，它们将被识别为异常。

随着深度学习技术的发展，特别是在低级和高级计算机视觉任务中深度卷积神经网络取得巨大成功后[3]，相关研究人员逐渐将注意力转移到如何将深度卷积网络的强大表示能力与视觉异常检测问题相结合，并致力于开发端到端的检测方法。

其中，FCDD[10]是一种无监督方法，用于合成异常样本以训练一类分类（One-Class Classification，OCC）模型。这个方式也可以应用于其他一类分类方法。Venkataramanan 等人[11]提出了一种带有引导注意力的卷积对抗变分自编码器（CAVGA），该编码器可以同等地应用于有异常图像和无异常图像的情况。在无监督设置中，CAVGA 通过注意力扩展损失的引导使模型专注于图像的正常区域。在弱监督设置中，CAVGA 使用互补的引导注意力损失来最小化与图像异常区域相对应的注意力图，同时专注于正常区域。Božič 等人[12]研究了图像级监督信息、混合监督信息和像素级监督信息对同一深度学习框架内表面缺陷检测任务的影响。Božič 等人[12]发现，少量的像素级注释可以帮助模型实现与完全监督相当的性能。而 DevNet[13]则尝试使用少量的异常样本来实现细粒度的端到端可微分学习。Wan 等人[14]提出了一种用于训练具有不平衡数据分布的逻辑斯蒂诱导损失（LIS）和用于表征异常特征的异常捕获模块（ACM），以有效利用少量异常信息。

2.2 图像分割

图像分割是计算机视觉领域中最受欢迎的研究方向之一，并且是模式识别和图像理解的基础。图像分割技术的发展与许多学科和领域密切相关，例如自动驾驶[15]，智能医疗技术[16]，图像搜索引擎[18]，工业检测和增强现实。

图像分割将图像划分为具有不同特征的区域，并提取感兴趣的区域（ROI）。根据人类的视觉感知，这些区域是有意义的且不重叠的。图像分割有两个难点：（1）如何定义“有意义的区域”，由于视觉感知的不确定性和人类理解的多样性导致对象缺乏明确的定义，这使得图像分割成为一个不适定的问题；（2）如何有效地表示图像中的对象。数字图像由像素组成，这些像素可以根据它们的颜色，纹理和其他信息组合在一起来构成更大的集合。这些被称为“像素集”或“超像素”。这些低级特征反映了图像的局部属性，但很难通过这些局部属性获得全局信息（例如，形状和位置）。

自 20 世纪 70 年代以来，图像分割一直受到计算机视觉研究人员的持续关注。经典

的分割方法主要侧重于突出和获取单个图像中包含的信息，这通常需要专业知识和人为干预。然而，从图像中获得高级语义信息是很困难的。协同分割方法涉及从一组图像中识别出共同的对象，这需要获取一定的先验知识。随着大规模精细标注图像数据集的丰富，基于深度神经网络的图像分割方法逐渐成为研究热点。这些方法通常不需要详细的图像标注，因此被归类为半监督或弱监督方法。

虽然图像分割研究已经取得了许多成果，但仍存在许多挑战，例如特征表示、模型设计和优化。特别是，由于标注数据有限或稀疏、类别不平衡、过拟合、训练时间长和梯度消失等问题，语义分割仍然面临诸多难题。

随着图像采集设备的不断发展，图像细节的复杂性和物体间的差异（例如尺度、姿态）大大增加。低级特征（例如颜色、亮度和纹理）很难获得良好的分割结果，而基于手动或启发式规则的特征提取方法无法满足当前图像分割的复杂需求，这对图像分割模型的泛化能力提出了更高的要求。相应的，深度学习算法被越来越多地应用于分割任务，分割效果和性能得到了显著提升。原始的方法是将图像分成小块来训练神经网络，然后对像素进行分类。由于神经网络的全连接层需要固定大小的图像，因此采用了这种块分类算法[19]。

2015 年，Long 等人[20]提出了全卷积网络（FCN），用卷积代替全连接，使得可以输入任意大小的图像，FCN 架构证明了神经网络可以进行端到端的语义分割训练，为深度学习在语义分割中的应用奠定了基础。后续神经网络的发展大多基于 FCN 模型进行改进。下一节将介绍使用深度学习进行图像分割的主要技术和代表性模型。

2.2.1 编码器-解码器架构

在 FCN 之前，卷积神经网络（CNN）在图像分类方面取得了良好效果，例如 LeNet-5[21]、AlexNet[22]和 VGG[23]，其输出层是图像的类别。然而，语义分割需要在获得高级语义信息之后将高级特征映射回原始图像大小。对于这样的任务基于 FCN 的编码器-解码器架构十分有效。

在编码器阶段，主要进行卷积和池化操作以提取包含语义信息的高维特征。卷积操作涉及将图像特定区域与不同的卷积核进行逐像素的乘法和求和，然后通过激活函数变换获得特征图。池化操作涉及在特定区域（池化窗口）内进行采样，然后使用某种采样统计量作为该区域的代表特征。在分割网络编码器中常用的骨干块是 VGG、Inception[24]和 ResNet[26]。

在解码器阶段，通过高维特征向量生成语义分割掩码。将编码器提取的多级特征映射回原始图像的过程称为上采样。上采样的插值方法使用指定的插值策略在原始图像的像素之间插入新元素，从而扩展图像的大小并实现上采样的效果。早期的上采样任务插值并不需要不需要训练参数。而 FCN 采用反卷积进行上采样，反卷积将原始卷积核的参数上下颠倒并水平翻转，并在原始图像的元素之间及其周围填充空格。SegNet[27]则采用反池化的上采样方法，反池化是 CNN 中最大池化的逆操作。在最大池

化过程中，不仅要记录池化窗口的最大值，还要记录最大值的坐标位置；在反池化过程中，激活该位置的最大值，并将其他位置的值都设置为 0。Wang 等人[28]提出了一种密集上采样卷积（DUC），其核心思想是将特征图中的标签映射转换为具有多个通道的更小的标签映射。这种转换可以通过直接在输入特征图和输出标签图之间进行卷积来实现，而不需要在上采样过程中插值额外的值。

2.2.2 跳跃连接

跳跃连接或短路连接是为了改进粗糙的像素级定位而开发的。随着深度神经网络的训练，性能会随着深度的增加而降低，这是一个退化问题。为了缓解这个问题，ResNet 和 DenseNet[29]中提出了不同的跳跃连接结构。相比之下，U-Net[30]提出了一种新的长跳跃连接，如图 2-1 所示。

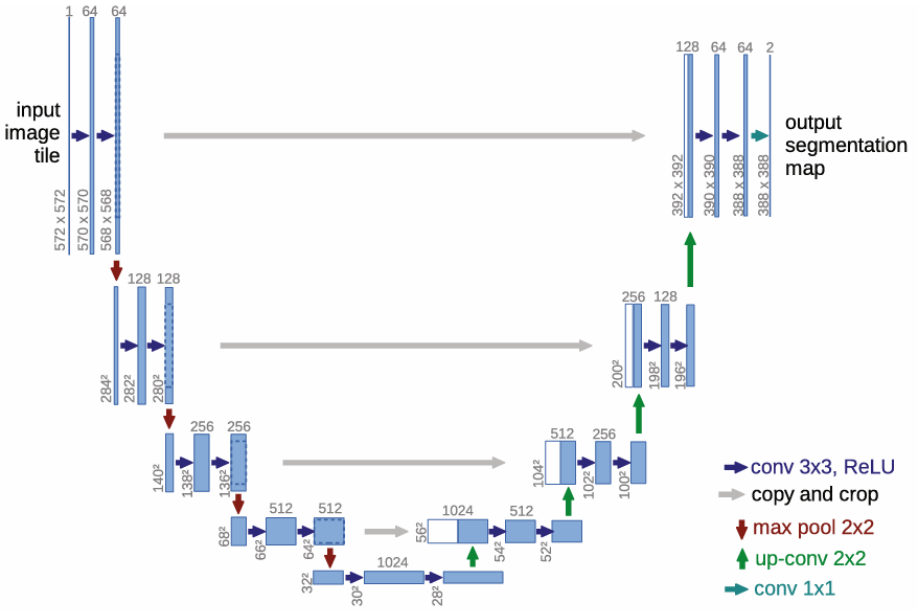

图 2-1 U-Net 结构图

U-Net 通过从编码器的层到解码器中相应层的跳跃连接和特征级联，获得图像的细粒度细节。它最初是为了解决生物显微图像分割问题而提出的，并自此在医学图像分割研究中得到广泛应用。U-Net 架构的设计灵感来自于全卷积网络（FCN），但它通过引入跳跃连接克服了 FCN 的一些限制。在 FCN 中，由于多次下采样操作，空间信息在网络的深层部分逐渐丢失，这可能导致分割结果中的细节信息不足。而 U-Net 通过跳跃连接将低层的空间信息与高层的语义信息结合起来，有效地解决了这个问题。

跳跃连接不仅有助于保持空间信息，还可以缓解梯度消失问题，使网络更容易训练。在反向传播过程中，梯度可以通过跳跃连接直接传递到较早的层，从而减少了梯度在传递过程中的消失。这使得 U-Net 等使用跳跃连接的网络能够在深度较大的情况下仍然保持良好的训练效果。

2.2.3 注意力机制

为了解决图像中不同区域之间的依赖关系，尤其是远距离区域，并获得它们的语义相关性，一些在自然语言处理（NLP）领域常用的方法已经被应用于计算机视觉，并在语义分割方面取得了良好的成果。注意力机制于 2014 年首次在计算机视觉领域提出。Google Mind 团队[31]采用循环神经网络（RNN）模型将注意力机制应用于图像分类，使注意力机制在图像处理任务中逐渐流行起来。

RNN 可以建立像素之间的短期依赖关系，连接像素，并按顺序处理它们，从而建立全局上下文关系。Visin 等人[32]基于 ReNet[33]提出了一种语义分割网络，每个 ReNet 层由四个 RNN 组成，这些 RNN 在图像的水平和垂直方向上扫描，以获得全局信息。

LSTM（长短期记忆）添加了一个新的功能来记录长期记忆，可以表示长距离依赖。Byeon 等人[34]使用 LSTM 实现了场景图像的像素级分割，证明了可以在二维 LSTM 模型中学习图像的纹理信息和空间模型参数。Liang 等人[35]基于图 LSTM 模型提出了一种语义分割模型，该模型将 LSTM 从序列数据或多维数据扩展到一般的图结构，进一步增强了全局上下文视觉特征。

自注意力机制则主要在编码器网络中使用，用于表示不同区域（像素）或特征图的不同通道之间的相关性。它计算单个样本所有位置之间键值对的加权和，以更新每个位置的特征。自注意力机制在图像分割方面取得了许多有影响力的成果，例如 PSANet[36]、DANet[37]、APCNet [38]、CARAFE[39]和 CARAFE++[40]。

2017 年，Vaswani 等人[41]提出了一种完全基于自注意力机制的深度神经网络 Transformer，完全摒弃了卷积和递归。此后，Transformer 及其变体（即 X-transformer）被应用于计算机视觉领域。利用 Transformer 的自注意力机制和 CNN 预训练模型，改进后的网络[42]也取得了一些突破。Dosovitskiy 等人[44]提出了一种视觉转换器（ViT），证明了 Transformer 可以替代 CNN 用于图像块序列的分类和预测。如图 2-2 所示，他们将图像划分为固定大小的块，排列图像块，并将块序列向量输入到由多头注意力层和多层感知器（MLP）组成的转换器编码器。

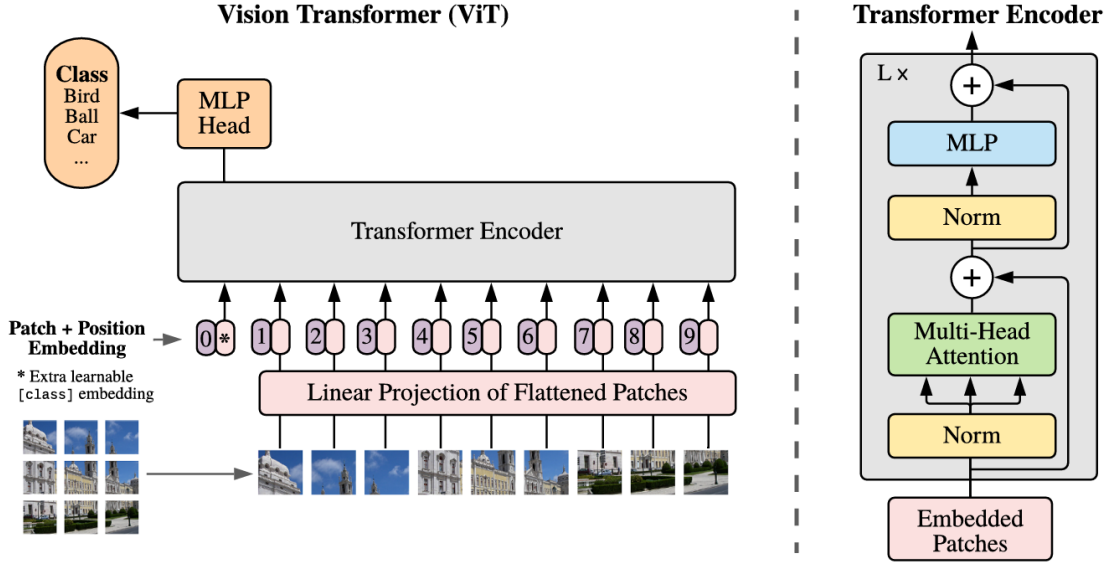

图 2-2 ViT 结构图

Liu 等人[45]开发了 Swin Transformer，它在图像语义分割和实例分割方面取得了令人印象深刻的性能。Swin Transformer 采用了滑动窗口方法，通过合并更深层次的图像块来构建不同层次的特征图，在每个局部窗口中计算自注意力，并在连续的 Swin Transformer 块中交替使用循环移位窗口来引入相邻非重叠窗口之间的跨窗口连接。如图 2-3 所示，Swin 转换器网络用移位窗口方法替换了转换器块中的标准多头自注意力（MSA）模块，其他层保持不变。

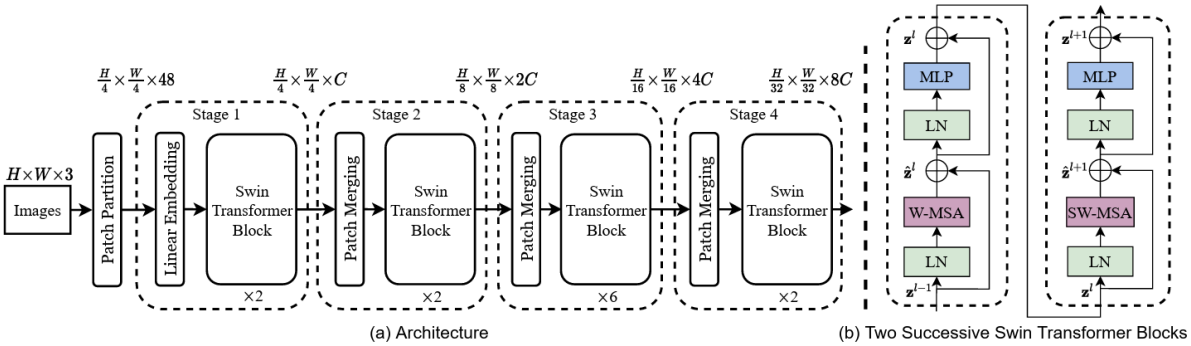

图 2-3 Swin Transformer 结构图

2.3 持续学习

深度学习中的一个重要问题是使神经网络能够增量地学习来自非平稳数据流的数据[46]。例如，当深度神经网络在新任务或数据分布的样本上进行训练时，它们往往会迅速失去先前获得的能力，这种现象被称为灾难性遗忘[47]。与此形成鲜明对比的是，人类和其他动物能够在不损害已经学到的技能的情况下增量地学习新技能[48]。持续学习，也称为终身学习，希望缩小生物智能和人工智能之间在增量学习能力方面的差距。近年来，由于持续学习算法在医学诊断[49]、自动驾驶[50]或预测金融市场[51]等应用中的潜在实用性，这一机器学习研究领域正在迅速扩大。

持续学习存在三种基本的有监督学习场景：

（a）在任务增量学习中，算法必须增量地学习一组可以清晰区分的任务；

（b）在领域增量学习中，算法必须在不同的上下文中学习相同类型的问题；

（c）在类别增量学习中，算法必须增量地学习区分越来越多的对象或类别。

在下一小节，本文将更详细地介绍这三种场景，并指出了与每种场景相关的不同挑战。同时也会回顾使用持续学习来优化深度神经网络的现有策略，并简单比较这些不同的策略对于每个场景的适用性。

2.3.1 持续学习的三个场景

在经典的机器学习中场景，算法可以同时访问所有训练数据。而在持续学习中，数据则是按顺序或分步骤到达的，并且数据的基础分布在随时间变化。

第一个持续学习场景是“任务增量学习”(Task-incremental Learning，简称 Task-IL)。

任务增量学习的定义是，在测试时始终清楚需要执行哪个任务。在实践中，这可能意味着任务身份是明确提供的，或者任务之间是可以明确区分的。在这种场景中，可以使用具有特定任务模组的模型进行训练（例如，每个任务都有一个单独的输出层），甚至可以为每个要学习的任务分配一个完全独立的网络，在后一种情况下，根本不存在灾难性遗忘的问题。因此，任务递增学习的挑战不在于简单地防止灾难性遗忘，而是找到有效的方式来共享跨任务的知识，以优化性能和计算复杂度之间的关系。并希望能够进一步地利用从不同任务中学习到的信息来提高其他任务的性能[52]。任务递增学习在现实中的例子是学习不同的运动或演奏不同的乐器，通常在学习过程中学习者总是很清楚应该学习哪种运动或演奏哪种乐器。

第二种场景称为“领域增量学习”（或 Domain-Incremental Learning，简称 Domain-IL）。在这个场景中，问题的结构总是相同的，但上下文或输入分布会发生变化。与任务增量学习类似，这个场景也可以描述为算法必须增量地学习一组不同领域的相同任务，但关键的区别在于算法不知道样本属于哪个领域。然而，模型并不一定需要识别当前任务所在的领域，因为每个领域上的任务都会使用相同的类别输出。然而，在这个场景中，算法只有在识别出任务所在的领域之后，才有可能使用特定任务的组件[53]，这相较于 Task-IL 所在的场景而言明显不那么合适。因此，在领域增量学习中，不可能通过为每一个任务设计不同的模组来防止遗忘。因此，在 Domain-IL 中，缓解灾难性遗忘仍然是一个重要的未解决问题。这个场景在现实中的例子包括在可变光照条件下逐步学习识别物体[56]或在不同天气条件下学习自动驾驶[57]。

第三种持续学习场景是“类别增量学习”（或 Class-Incremental Learning，简称 Class-IL）。在这个场景中算法必须增量地学习区分越来越多的对象或类别。这个场景常用的设置是遇到一系列基于分类的任务，其中每个任务包含不同的类别，算法必须学习区分所有类别[58]。在这种情况下，任务识别对于解决问题是必要的，因为它决定了当前样本可能属于哪些可能的类别。换句话说，算法应该既能解决每个单独的任务，又能识别样本属于哪个任务。例如，模型可能首先学习区分猫和狗，然后学习区分牛和马；而在任务增量学习中，模型不需要区分不同领域中遇到的动物（例如猫和牛），但在类别递增学习中这是必需的。这个场景中的一个重要挑战是学习区分没有一起观察到的类别，尤其是当我们只有有限的资源来存储先前任务的信息时，这对深度神经网络来说是十分难以解决的问题。

2.3.2 持续学习的三种方案

我们将现有的持续学习工作大致分为三类。在第一类中，我们讨论基于重要性加权参数正则化的方法，该方法使神经网络的学习过程减少遗忘。在第二类中，我们讨论基于数据重放的方法，这些方法要么存储以前任务中的示例，要么学习从先前观察到的数据分布生成新示例。在第三类中，我们讨论随时间增长改变神经网络架构的方法。

基于正则化的方法。这类方法的共同关键思想是识别在学习过去经验中起重要作用的参数。然后，这些参数在未来更新中受到保护，而不重要的参数则进一步训练以学习新任务。该类方法通常通过向目标添加一个正则化项来解决灾难性遗忘问题，该正则化项会惩罚神经网络输入输出函数的变化。在 Li 和 Hoiem 的研究中[59]，他们使用一种知识蒸馏的形式。而 Hinton[60]，鼓励先前任务网络和当前网络在新任务数据上的预测结果相似。类似地，Jung 等人[61]对最终隐藏激活之间的 L2 距离进行了正则化，而不是使用知识蒸馏惩罚。这两种正则化方法都旨在通过使用旧任务的参数存储或计算额外的激活函数来保留旧任务的输入输出的一部分映射。这使得功能性方法在计算上变得昂贵，因为它需要对每个新数据点通过旧任务的网络进行一次前向传播。

基于数据重放的方法。这类方法假设我们只有有限的内存空间来存储以前任务的示例，而模型将使用这些有限示例来防止在学习新任务时出现灾难性遗忘。在这样的场景下，生成模型显得十分有益，因为生成模型能够从学习的数据分布中抽取伪示例，为基于记忆的模型提供了一种替代的采样方法。例如 Robins[62]发现将新的信息与先前内部生成模式交织在一起，有助于巩固现有知识，而无需显式存储训练样本。Draelos 等人[63]也使用伪排练方法对自编码器进行增量训练，在数据重放过程中利用编码器的输出统计信息生成解码器的输入。然而，与上述大多数方法类似，伪排练方法的使用仅在两个相对低复杂度的数据集上进行了严格评估，例如 MNIST 和 Street View House Number。因此，这种生成方法是否能扩展到更复杂的领域，这是一个值得探讨的问题。

基于动态神经网络的方法。到目前为止所讨论的方法都假设有一个无限容量的网络来学习新任务。然而在实际场景中，神经网络的有限容量会限制其随时间学习新任务的能力。关于动态神经网络的各种工作[64]旨在解决这个问题。这些方法从一个简化的架构开始，并在需要时逐步增加新组件以扩展网络，从而在当前任务上获得满意的性能。在[66]中，神经网络模型通过增加一个称为列的单独子网络来进行扩展，该子网络负责学习新任务。先前学习任务中开发的列表示保持固定，以避免灾难性干扰。然后，这些表示作为新添加的列的额外输入进行馈送。然而，这种方法存在扩展性问题，因为每次需要学习新任务时都会添加一个新列。由于需要对先前添加的列进行计算以将信号传递到新列，因此此类网络中的推理变得越来越困难。最近，Jaehong[66]提出了一种创新的方法，该方法随时间仅亚线性地增长网络的规模。

3 技术路线

整个研究可以分为两个部分的主要工作，分别是持续学习方法的静态环境数据实验与燃气管道缺陷检测与训练的后端软件开发。

3.1 持续学习方法实验

根据燃气管道焊接缺陷检测的实际场景，系统所需检测的类别大致不变，而数据分布可能根据时间的推移发生变化，这是一个类似领域增量（Domain-IL）学习的任务。接下来将在领域增量学习的环境下使用实际场景数据对现有的各种领域增量学习算法进行测试。

3.1.1 持续学习框架

Avalanche [67]是一个基于 PyTorch 的端到端持续学习库，诞生于 ContinualAI，其提供了一个共享和协作的开源代码库，用于快速原型设计、训练和可重复评估持续学习算法。Avalanche 可以以多种方式帮助持续学习的研究者和实践者减少代码编写量，更快地原型设计，并减少错误；提高可重复性、模块化和可重用性；增加代码效率、可伸缩性和可移植性；增强研究报告的影响力和可用性。

该库分为五个主要模块：Benchmarks（基准测试）：此模块维护一个统一的数据处理API，主要是从一个或多个数据集中生成数据流。它包含所有主要的持续学习基准测试；Training（训练）：此模块提供有关模型训练的所有必要实用程序。这包括实现新的持续学习策略的简单而高效的方法，以及一组预实现的持续学习基准和最新算法，可以方便的使用他们完成比较测试；Evaluation（评估）：此模块提供所有可以帮助评估持续学习算法相对于我们认为对持续学习系统重要的所有因素的实用程序和指标。Models（模型）：在此模块中，您将能够找到可用于持续学习实验的多个模型架构和预训练模型；Logging（日志记录）：它包括先进的日志记录和绘图功能，包括本机 stdout、文件和 TensorBoard 支持。

后续的实验将依托 Avalanche 平台完成持续学习的基准测试、算法、评估指标等等的开发与实验。我们希望基于在线与离线的两种不同持续学习策略进行实验与比较。

3.1.2 在线持续学习

能够进行模型参数的优化同时不中断推理服务的进行，能够实现这样的在线场景总是最理想的。我们最佳的期望是能够在燃气管道焊接检测系统上实现这一点。为此，我们需要对已有的在线持续学习方法进行评估实验。Gradient Episodic Memory (GEM) [70]是一个持续学习的模型， GEM 的主要特点是它有一个情景记忆M_t，它存储了从任务 t 中观察到的示例的一个子集。为了简化，GEM 假设任务描述符是整数，并使用它们来索引情景记忆。当使用整数任务描述符时，不能期望显著的正向梯度，因为可能会有零

样本学习的情况。相反，GEM 专注于通过有效利用情景记忆来最小化负向转移，也就是灾难性遗忘的情况。

GEM 有三个可以改进的方向。首先，GEM 没有利用结构化的任务描述符，这可能被利用以获得积极的正向转移。其次，GEM 没有研究先进的内存管理（例如构建任务的核心集）。第三，每个 GEM 迭代需要对每个任务进行一次反向传递，从而增加了计算时间。我们希望在经过评估后 GEM 可以被有效的迁移到焊接缺陷检测的任务当中。

3.1.3 离线持续学习

离线的持续学习方法更易于维护和扩展，在系统设计中也更容易实现。所以我们也希望可以评估与修改更多的离线持续学习方法并应用到我们的场景中，如：

1）EWC（Elastic Weight Consolidation）[68]算法是一种在持续学习中用于解决遗忘问题的方法。该算法基于贝叶斯学习理论，主要思想是保持对之前学习任务的权重分布的不确定性估计，以此在面临新任务时防止对之前知识的过度遗忘。具体来说，EWC 算法通过计算每个参数的 Fisher Information Matrix（FIM）来估计参数的重要性。FIM 是概率分布梯度的协方差，表示参数的变化对模型输出的影响程度。然后，算法会对每个参数赋予一个与其 FIM 值成比例的惩罚项，这样在更新参数时，对重要的参数变化会施加更大的阻力，从而保持对之前任务的记忆。

2）LFL[61]（Less-Forgetful Learning）对最终隐藏激活之间的 L2 距离进行了正则化，而不是使用知识蒸馏惩罚，这种正则化方法旨在通过使用旧任务的参数存储或计算额外的激活函数来保留旧任务的输入输出的一部分映射。

3）SI[69]（Synaptic Intelligence）介绍了一种特定的高维突触模型来解决持续学习中的灾难性遗忘问题，其模仿神经生物学赋予单个突触潜在的复杂动态属性，以智能地控制神经网络的学习过程。

3.2 软件架构

软件后端的开发总体可以四个层级：

1）WEB 接口层。在最外层的 Web 接口将负责接收来自外部用户的推理或训练请求，并将上传的数据集整理到对象存储桶中。Web 接口的实现将基于 fastapi 实现。

2）任务管理层。任务管理层负责任务的状态整理和执行，当用户请求经过 fastapi 后将被整理为特定的 celery 任务，celery 将会自行维护一个 redis queue，将要执行的任务信息会被压入 redis 队列中，在 worker 空闲时再从 redis 中取出。

3）数据存储层。对于结构化的关系型数据（例如持续学习模型的版本与学习轮次等元数据）我们将把他们存入 postgresql 数据库中。而对于模型参数、数据集等大容量的数据我们将把他们存入 MinIO 存储桶中。

4）容器与驱动层。对于不同整个系统所需要的不同服务与硬件我们将不同的模组封装在不同的 docker 容器中方便系统的迁移与资源调动。例如 redis、postgresql、minio

等将单独封装在容器中，方便以后部署在读写能力强的主机上；而需要使用 GPU 的 celery 模块这需要部署在有 cuda 环境支持的主机上。

系统的详细架构可以参考下图 3-1：

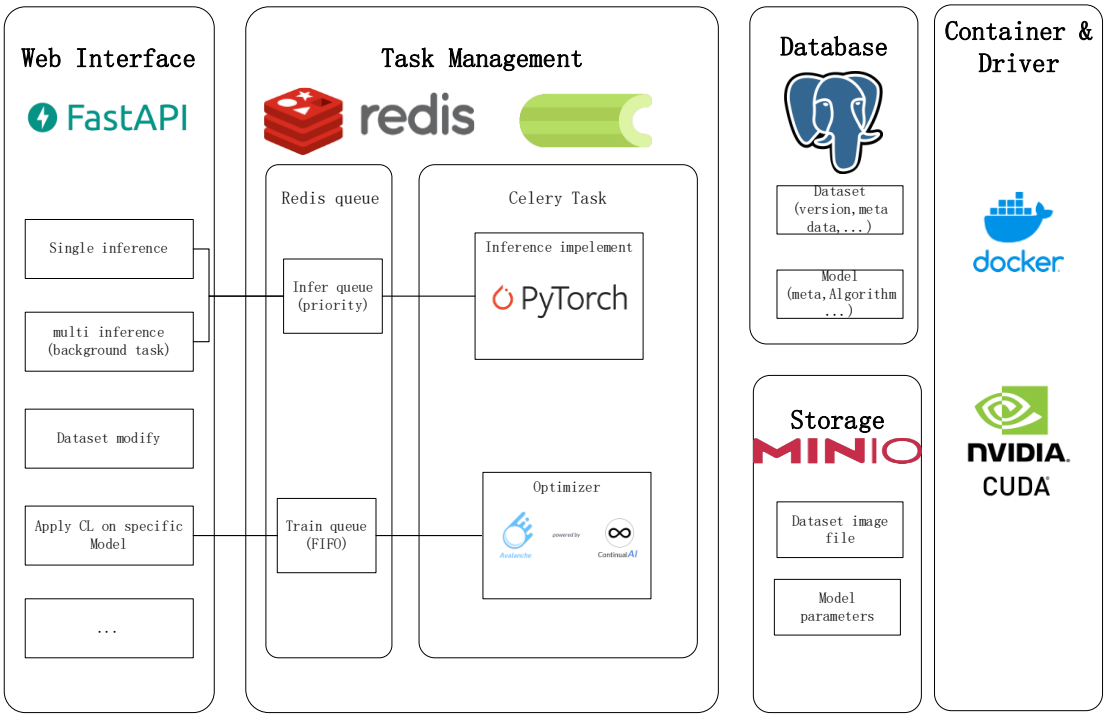

图 3-1 检测模型推理与训练平台架构图

4 算法评估实验

为了验证各持续学习算法在实际焊接缺陷检测环境下的表现，笔者将基于生产环境中的数据与场景设计实验，对 NAIVE，EWC[68]，LFL[61]，SI[69]和 GEM[70]五种持续学习策略进行对比评估。其中 NAIVE 作为对照组，是为朴素的持续学习方法，即没有应用任何特殊的算法来避免持续学习的灾难性遗忘问题。算法评估实验的结果可以在 https://api.wandb.ai/links/kirigiri_suzumiya/nrn4i0as 获取。

4.1 实验场景设计

与传统机器学习不同，在持续学习场景中数据集是随时间有序输入的。如图 4-1 所示，整个数据流（Benchmark）由多个按顺序排列的子任务组成（$e_1, e_2, \ldots, e_n$）。在训练过程中，子数据集逐个参与到模型（M）的训练中，而为了避免由于各子任务中数据分布的不同而可能导致的灾难性遗忘，在持续学习场景中还需要应用额外的训练策略（A_{CL}）来优化训练过程。同样在评估流程中，持续学习场景除了要关注模型在当前任务下的表现，还要及时察觉到模型在之前任务中性能的变化（$p_1, \ldots, p_m$）。

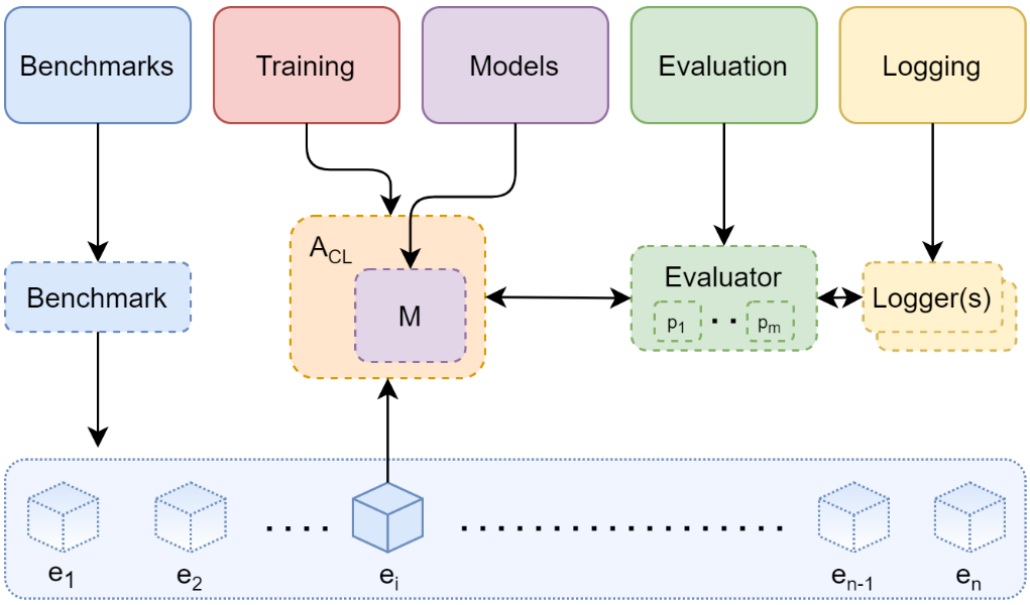

图 4-1 持续学习场景

由于 Avalanche 原来设计的主要场景面向简单多分类任务，与本文场景中的图像实例分割不同，故笔者在整个训练过程中使用了 Avalanche 的众多底层方法以在实例分割的任务上实现其原有的持续学习功能。具体实践如下：

4.1.1 数据流搭建

根据以上提到的种种不同，笔者结合实际焊接缺陷场景设计乐模拟的持续学习数据流。原始数据集总共 253 个样本，输入为[513,447,3]的 RGB 图像，输出为[513,447，1]的类别标签。在数据流搭建中，将总共 253 个训练样本首先经过随机采样分为 5 个子数

据集使每个子数据集中的样本数为[51,51,51,50,50]。

然而，在数据集处理的过程中，Avalanche 原生的 Dataset Loader 面向多分类任务与类别增量持续学习，其构建方法中对模型最终输出执行了哈希编码，并使用哈希编码来匹配输出。但在本文的场景中，由于实例分割的输出与图像像素类别相关，对最终输出层进行哈希后便无法与 label 数据进行逐像素的比对，这导致 Avalanche 的绝大多数 Dataset 高层方法与本文的实例分割场景无法兼容，所以笔者选择从 torch dataset 出发，手动搭建 Avalanche 的 Benchmark 数据流。

首先继承 Torch Dataset 并重写初始化方法（__init__）、数据预处理方法(preprocess)以及迭代器方法（__getitem__）以从文件系统中读取数据并满足 Torch Dataset 的类接口。将各个子任务分别加载为不同的 Torch Dataset

之后将 Torch Dataset 转换为 Avalanche Dataset 并手动添加 task_labels。此处的 task_labels 是在持续学习中用于标注不同任务的数据参数，由于本文的场景是 Domain-CL，故所有子任务都应该拥有相同的 task_labels。

此时已经拥有的 5 个 Avalanche Dataset，可以在上一节的定义中表示为（$e_1, e_2, \dots, e_5$）。由这 5 个子任务组成了 Benchmark 的训练数据流。除此之外还需要一个等长的测试数据流，根据文本的环境，笔者希望测试数据流可以用于评估灾难性遗忘的程度。故这里将第一个子任务的数据集在测试数据流中重复 5 次以记录在子任务推进的过程中模型对第一个任务的性能变化。综上，测试数据流可以表示为（$e_1, e_1, \dots, e_1$）。

通过 Torch Dataset 来生成 Avalanche Dataset 的过程绕过了 Avalanche 的原生分类数据集构建过程，使本文的实例分割任务数据集能够在 Avalanche 框架下被正确加载。并且提供了训练与测试两个数据流以满足后续的实验需求。

4.1.2 持续学习策略

如本节开头所说本文需要分别对 5 种持续学习方法进行评估，在场景设计层面笔者将不涉及算法的具体表现与优化，将更多的关注在如何将上述算法执行与应用到本文的场景中。

在上一节说到为了绕过原生分类数据集的构建，使用 Torch Dataset 来构建 Avalanche 数据流。这解决了数据集的加载问题，但在 Avalanche 的训练模组中又引发了其他的问题。由于加载方式的不同，后续数据集的 Dataloader 中部分接口也与 Avalanche 中的格式有一定差别，导致训练过程中数据无法以正常的 mini-batch 形式被加载。

为了解决这个问题，笔者尝试在学习策略模块对其 mini-batch 解析的部分代码进行重构，以实现本文所需的功能。具体的，针对所有 5 个持续学习策略类需要重载 4 个方法：mb_x, mb_y, mb_task_label, _unpack_minibatch。分别对应从 mini-batch 中取出 x，y, task_label 以及从 Dataloader 中取出 minibatch 的功能。

额外的，由于 LFL 策略的执行中要求冻结当前模型的最后一层全连接，这在普通的

分类模型中很普遍，分类模型的最后通常都是几层全连接。但在本文的场景中，任务是像素级的实例分割，而使用的 UNet 模型最后也没有全连接层。由此笔者重写了 SGLFL 方法，相较与 LFL 的区别在于其冻结了 UNet 下采样最底层的参数而不是最后的全连接层。此外，LFL 还需要模型拥有特定方法以取得特征，这部分修改将在下一节的模型中介绍

通过重写各训练策略的 mini-batch 相关方法使图像实例分割在 Avalanche Training 模块上的执行能够实现。

4.1.3 模型与优化器

由于本文的研究重点在于持续学习方法的应用，在基础模型方面笔者希望能够使用尽量简单且被证明通用的模型以更直观准确地展示不同持续学习算法在特定场景中克服灾难性遗忘的能力。各个持续学习算法在进行比较时需要保证模型和优化器超参数的一致性。

在上述的背景下，笔者选择 UNet 作为实例分割的基础模型，输入通道数为 3，输出特征图为 4。优化器选择 RMSprop, 参数设置为$lr = 0.00001, weight_decay = 0.00000001, momentum = 0.999$。损失函数选择在类别通道上进行计算的交叉熵损失。同时在每一个子任务的训练循环中，设置 epoch=1, batch_size=8。

4.1.4 评估插件与日志

在深度学习的训练中，使用评估插件和日志记录器是非常必要的，在本文的场景中，我们希望比较不同持续学习方法在实际数据集中的表现，这需要能够在训练过程中实时地监控损失（loss）、准确率（accuracy）等关键指标。这有助于你了解模型的学习情况，并判断是否需要调整超参数或采取其他优化措施。

在实验设计中，笔者通过不同的评估插件实时记录了模型在训练过程中的各种指标：

- Loss Metrics:该评估插件用于计算损失（loss）的度量。损失度量将在所有的时间尺度（mini-batch、epoch、replay 等）和不同的数据流上进行计算和记录。
- Timing Metrics:该评估插件用于计算时间的度量。时间度量将在 epoch 和 mini-batch 这两个时间尺度上进行计算，以比较不同持续学习算法的运算耗时。该度量只会在训练数据流上计算。
- CPU & GPU Usage Metrics:该评估插件用于计算 CPU 和 GPU 使用率的度量。和损失度量类似，这些度量将在所有不同的时间尺度和数据流上进行计算和记录。

此外，笔者还使用了数个日志记录器（logger）用于在训练的同时同步上述度量到本地文件或者是线上数据库以支持后续的数据分析与可视化

- Interactive Logger：用于在交互式环境中（如 Jupyter notebook）显示日志信息，方便笔者实时查看和监控训练过程。

- Text Logger：用于将日志信息记录到文本文件中，主要用于在长时间运行后出现的错误定位。
- TensorBoard Logger：用于将日志信息记录到 TensorBoard 格式的数据文件中，以便使用本地的 TensorBoard 服务及其可视化功能来监控和分析训练过程。
- Wandb Logger：用于将日志与度量信息上传到 WandB。其后端的数据库 Weights & Biases 是一个用于机器学习实验的日志记录和可视化工具，可以非常方便的生成图表与报告，本文的分析图表大多基于他们的服务绘制。

4.2 NAIVE（灾难性遗忘的出现）

灾难性遗忘在持续学习的环境下经常出现，由于模型在学习后续任务的过程中为了适应新任务的数据分布，对模型参数进行的更新破坏了旧任务的识别模式。根据 4.1 节的实验设计，首先需要观测理论中的灾难性遗忘在实际焊接缺陷检测的场景设计中是否真实存在。

在我们的实验中，我们通过使用不同持续学习策略对 5 个不同的不重复随机采样子任务进行训练，并在过程中观察模型在第一个子任务上的性能变化来评估模型发生咋男性遗忘的程度。实验结果如图 4-2 所示。

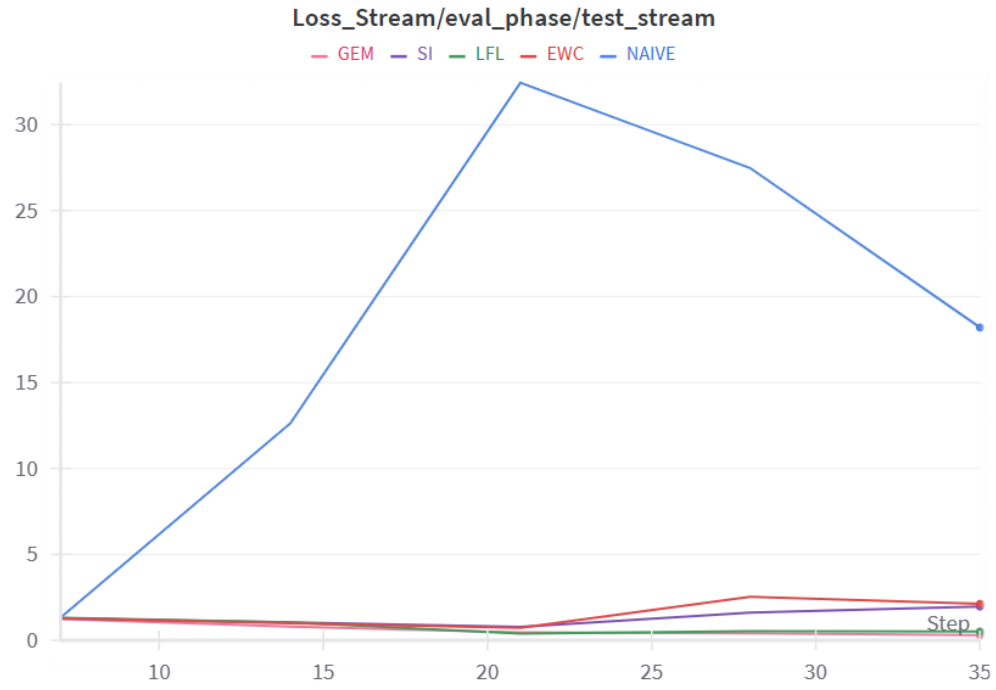

图 4-2 子任务训练过程中首个任务的损失变化

可以看到作为对照组的 NAIVE 训练策略（即简单的直接将模型参数在下一个子任务上继续训练）随着子任务训练的推进在第一个任务的性能上发生了明显且灾难性的倒退。当训练到第三个子任务时模型在第一个任务上的性能最差，相较于刚完成第一个任务训练时性能倒退了 23.47 倍。

相较于其他的 4 个持续学习策略而言，NAIVE 方法证实了在焊接缺陷检测的模拟持续学习环境下，如果不对模型训练过程做任何有效的约束，模型在持续学习新任务的过程中对旧任务的识别模型将可能发生灾难性遗忘。

4.3 EWC

EWC（Elastic Weight Consolidation）在当前任务训练结束时计算每个权重的重要性。在对每个小批量数据进行训练时，损失会增加一个惩罚项，该惩罚项会使当前权重的值与其在先前经验中的值保持接近，这种接近程度与该权重在该经验上的重要性成正比。具体地，EWC 在当前任务的损失后添加一个正则项来避免灾难性遗忘，如公式 1 所示：

$$\mathcal{L}(\theta) = \mathcal{L}_{\mathcal{B}}(\theta) + \sum_{i} \frac{\lambda}{2} F_i \left(\theta_i - \theta^*_{A,i}\right)^2 \tag{1}$$

其中$\mathcal{L}_{\mathcal{B}}(\theta)$表示当前任务的损失，$\lambda$表示过去任务对当前的重要程度，$F_i$为 fisher 矩阵反应对参数估计的不确定性，$\theta_i$表示模型的各个参数。

在本文的场景中，EWC 在训练耗时方面表现优秀，但根据其计算原理，其复杂度会跟随任务数线性增长，随着任务数的提升，其速度优势逐渐消失，在进行到第四个任务时耗时已经超过了 SI 算法，如图 4-3 所示

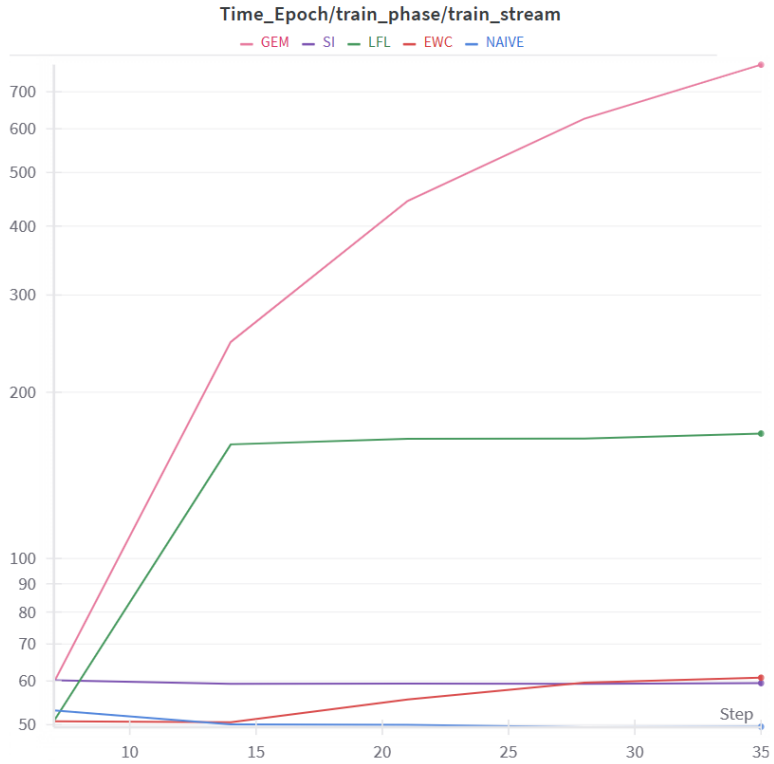

图 4-3 各持续学习策略耗时（对数尺度）

在克服灾难性遗忘方面，EWC 在所有参与比较的 4 个持续学习策略中表现得最差。任务数较少时，EWC 在旧任务上的性能保持稳定；然而随着任务数的增加，EWC 的正

则化方法难以维持住多个任务的性能，最终在进行到第 4 个任务时产生了明显的遗忘。跟耗时相近且不会随时间增长的 SI 算法相比，他们都在第四个任务时出现了一定程度的遗忘，但 EWC 明显更甚。如图 4-4 所示

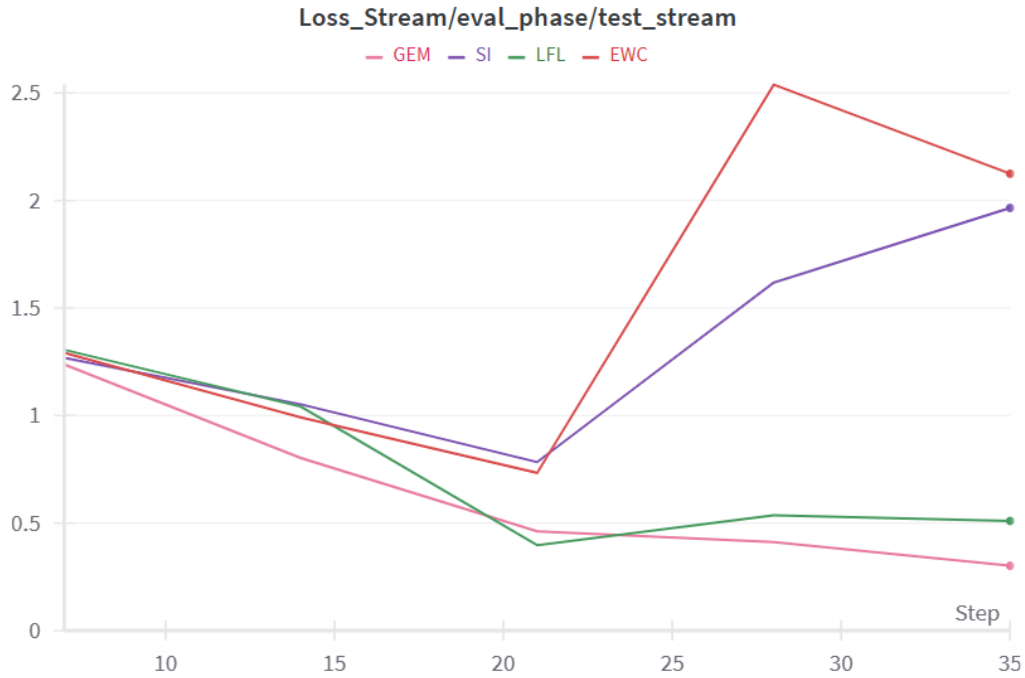

图 4-4 各持续学习策略训练过程中首个任务的损失变化

综合考虑在时间以及性能上的表现，EWC 适合于子任务较少的场景（n<4）并在该情况下有出色的速度和防止灾难性遗忘的性能。

4.4 LFL

LFL（Less-Forgetful Learning）应用两种策略以减轻灾难性遗忘：

1）保持决策边界不变：这意味着模型在学习新任务时，应尽量不改变对旧任务实例的分类。

2）目标（新）数据上的特征空间变化不大：即学习新任务时，不应大幅修改表示旧任务数据的特征表示。

为了实现这两个策略，LFL 采用了一种正则化手段，即计算当前模型版本与先前模型版本所提取特征之间的欧氏距离损失（Euclidean loss）。这种方法有助于维护特征空间的稳定性，从而防止灾难性遗忘的发生。通过将这种损失作为正则项加入到训练过程中，可以促使模型在学习新知识的同时，保留对已有任务的泛化能力。

然而，这样的策略在本文的场景中遇到了一定的问题。如 4.1.2 节所述，在本文的实例分割场景中的模型 UNet 主要分为上采样与下采样两个过程，而同颗粒度的上下采样层数之间还有直接的跳跃连接，这使得 LFL 在确定特征提取层时遇到了困难。将特征欧氏距离作为正则项的方法实际上影响了模型在各个任务上的有效收敛。如图 4-5 所示

图 4-5 各持续学习策略训练过程中损失函数的变化

当模型每次切换到下一个任务时都会导致模型性能的整体下降，并导致整个优化地形逼得崎岖且难以优化。当损失函数整体下降到 0.6 左右时，模型损失函数开始发生震荡难以继续优化。而使用其他策略的模型则并未在损失函数到达 0.6 时发生震荡，仍然可以继续优化。由此可以确定 LFL 的决策边界与特征表示策略在实例分割的场景中表现不稳定，在一定程度上阻碍了模型在当前任务的收敛过程。

但是，在非线性增长的时间复杂度下，LFL 展现出了最优秀的克服遗忘的能力，其在整个 5 个子任务的训练过程中几乎没有出现明显的遗忘趋势（见图 4-4），这也许在侧面佐证了其阻碍当前任务优化的过程实则有效的避免了对先前任务的遗忘。

4.5 SI

SI(Synaptic Intelligence) 使用具有更复杂的三维状态空间的突触（通常我们称为模型参数）来缓解灾难性遗忘问题。在应用了 SI 的模型中，突触状态会跟踪过去和当前的参数值，并计算突触在解决过去遇到的问题中的“重要性”的在线估计。该重要性度量可以在训练期间在每个突触处高效且局部地计算，并能够代表每个突触对全局损失变化的局部贡献。当任务发生变化时，通过防止重要突触在未来任务中发生变化来避免遗忘。因此，未来任务的学习主要由对过去任务不重要的突触学习，从而避免了对过去任务的遗忘。

由于其将原有的一维参数替换成了三维，故在 SI 算法训练过程中占用了最多的系统内存来存储这些参数。SI 拥有所有 4 个持续学习方法中最高的空间复杂度，如图 4-6 所示

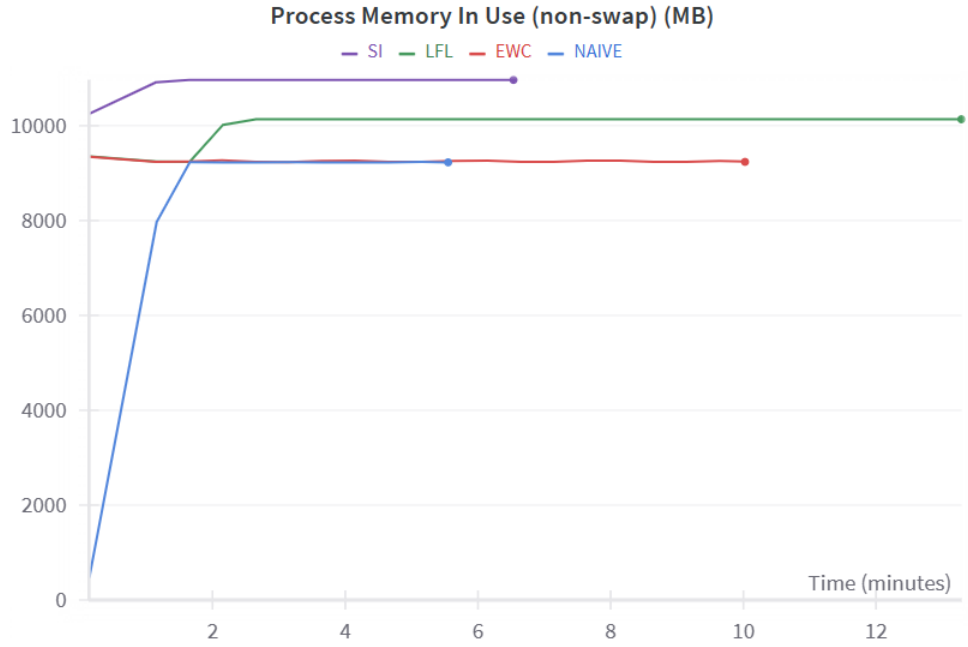

图 4-6 各持续学习策略训练过程中使用的内存

在训练耗时上，由于其专注于维护三维的参数，故复杂度不会随着任务的推进而增加，训练耗时维持不动。然而在防止灾难性遗忘的性能上，其与 EWC 相同，随着任务数的增加 SI 算法逐渐难以控制旧任务的模式，最终逐渐发生了对旧任务的遗忘。

4.6 GEM

GEM（Gradient Episodic Memory，GEM）来自先前任务的外部情景记忆，来投影当前小批量数据上的梯度。具体来说，它使得当前小批量数据的梯度投影后，与所有先前任务的参考梯度的点积保持为正。这使得在更新模型参数以适应新任务的过程中，GEM 确保这个更新的方向不会损害模型在之前任务上学到的知识。通过这种方式，GEM 能够解决在连续学习场景下的灾难性遗忘问题，通过维护一个包含过去经验样本的情景记忆，并在每次更新时考虑这些记忆，从而促进对历史任务的持续表现。

然而，由于每次更新都需要包含之前的样本，这使得 GEM 的训练耗时随着任务数的增加而不断增长。在本文的场景中，GEM 的耗时相较所有其他 4 个持续学习方法都更加长，且随着任务数的增长持续地线性增加。在进行到第 5 个子任务时，其耗时达到了惊人的 784 秒，而没有应用任何策略的 NAIVE 仅耗时 49 秒，GEM 在进行 5 个任务后，单个任务耗时增长了 15 倍。（见图 4-3）这使得 GEM 在实际场景中应用的可能大打折扣，其实质上是一种经验重现策略，这意味着如果我们希望在实际环境中从一个中断点恢复该策略的模型，那么我们需要能够获取之前所有任务的训练样本。而前面的 4 种策略都基于不能够访问先前样本的情况。

而更长的耗时也带来了更好的性能表现，GEM 在所有 5 种策略中拥有不容置疑的收敛稳定性与克服灾难性以往的性能（见图 4-5&图 4-4）。在不考虑训练成本的情况下 GEM 无疑是最佳的选择。

4.7 总结

本章探讨了在持续学习背景下灾难性遗忘的问题及其多种缓解策略在焊接缺陷检测任务中的应用与对比分析。研究共分为 5 个关键部分：

灾难性遗忘的验证：实验通过比较使用不同持续学习策略的模型在连续学习五个子任务时的表现，证实了 NAIVE（无特殊策略）方法确实导致了在首个任务上性能的显著衰退，最高达到 23.47 倍的性能倒退，确立了灾难性遗忘现象的存在。

Elastic Weight Consolidation (EWC)：作为一种持续学习策略，EWC 通过在损失函数中加入正则项来限制关键权重的变动，以维护过往任务的知识。实验显示，EWC 在少量任务时能有效减缓遗忘，但随着任务增多，其时间和性能优势减弱，特别是在处理第四个任务后，遗忘现象变得明显，且训练时间超过其他某些策略。

Less-Forgetful Learning (LFL)：LFL 策略旨在保持决策边界稳定并减少特征空间的变化，利用特征欧氏距离作为正则项。然而，在实例分割任务中，LFL 因模型结构（如 UNet 中的跳跃连接）遇到实施难题，导致模型收敛受阻。尽管如此，LFL 仍表现出较强的抗遗忘能力。

Synaptic Intelligence (SI)：SI 通过跟踪突触（模型参数）状态和其解决问题的重要性来防止遗忘。尽管 SI 在内存占用上最高，且最终也会出现遗忘现象，但它在时间复杂度上相对稳定，不随任务增加而显著增长。

Gradient Episodic Memory (GEM)：GEM 通过维护一个情景记忆来指导梯度更新，确保新任务的学习不会破坏旧任务的性能。GEM 在防止遗忘方面效果显著，但其训练时间随任务数线性增长，成为实际应用的一大障碍，尤其是在处理多个子任务时。

综上，各策略在本文的场景中各有优劣：EWC 和 SI 在任务较少时表现良好但扩展性有限；LFL 虽然在特定场景下优化遇到挑战，但在抗遗忘方面表现出潜力；GEM 提供了强大的遗忘缓解能力，但伴随高昂的计算成本。

具体策略的选择仍需依据实际应用场景的时间与资源成本进行权衡。

5 后端应用开发

在完成持续学习方法的性能评估后，本文将继续介绍开发一个使用上述持续学习方法的后端应用的开发流程。具体架构在 3.2 节已经介绍，这里将严格根据图 3-1 中展示的模型推理与训练平台进行设计。

5.1 数据库、对象存储与中间件

一个可靠的后端服务必然需要存储与中间件的支持。本小节将着重介绍所使用的各种开源数据库、对象存储与中间件组件。

Postgresql 数据库：由于后端应用需要的模型推理与训练功能，需要架设一个数据库服务，提供结构化数据的存储服务。具体地，需要两张表来分别存储模型与数据集的相关元数据。其中模型表包括模型的训练策略，所依赖的数据集，继承的基础模型以及在 MinIO 中存储的路径等；数据集表包括上传的时间，数据集中样本的数量，以及在 MinIO 中存储的路径。

MinIO 对象存储：由于模型与数据集都会占用比较大的存储，为保证可用性与可能的分布式计算场景，我们需要使用 MinIO 作为后端存储架构来保存模型与数据集。具体的，在 MinIO 初始化后需要建立两个存储桶，分别存储数据集与 Avalanche checkpoint 文件，这里简单的将两个存储桶命名为 dataset 和 model。

Redis 缓存数据库：由于模型训练是一个相当耗时的任务，数个任务请求就可能占用后端的所有 worker，为了避免可能的性能我们希望使用 Redis 来缓存任务信息，实现消息队列的缓存。

此外，为了方便应用的数据迁移与服务启停，这里将使用 docker 部署必要的框架。笔者使用 Docker Compose 文件定义了一个包含上述三个服务的应用栈：PostgreSQL 数据库、MinIO 对象存储服务和 Redis 缓存服务器。下面是每个服务的具体配置和作用：

PostgreSQL 服务

- 使用 postgres:13.2 作为基础镜像。
- 重启策略: 如果服务因故障退出，它将自动重启(restart: on-failure)。
- 环境变量:
  - POSTGRES_USER: 设置数据库用户名为 root；
  - POSTGRES_PASSWORD: 设置数据库密码为 password；
  - POSTGRES_DB: 初始化时创建的数据库名为 cldb。
- 数据持久化: 通过挂载.volume/db 到容器内的/var/lib/postgresql/data，实现数据持久化。
- 端口映射: 容器内的 5432 端口映射到宿主机的 5432 端口，允许外部访问数据库。

MinIO 服务

- 使用官方 quay.io/minio/minio 镜像。
- 重启策略: 同样设为在失败后自动重启。
- 环境变量:
  - MINIO_ROOT_USER: MinIO 访问的用户名设置为 root。
  - MINIO_ROOT_PASSWORD: 密码设置为 password。
- 数据持久化: 通过挂载.volume/minio 到容器的/data 目录，确保存储的数据可以在容器重启后保留。
- 端口映射:
  - 9000 端口映射使得可以通过宿主机的 9000 端口访问 MinIO 服务。
  - 9001 端口映射用于访问 MinIO 的控制台界面。
  - 启动命令: server /data --console-address ":9001" 指令启动 MinIO 服务，并且在 9001 端口上启用控制台访问。

Redis 服务

- 镜像: 使用了最新版本的 Redis 镜像。
- 端口映射: 将容器的 6379 端口映射到宿主机的 6379 端口，以便外部应用可以访问 Redis 服务。
- 数据持久化: 通过挂载.volume/redis 到容器的/data 目录，保持 Redis 的数据持久性。

使用一个 Docker Compose 配置文件构建了一个包含数据库(PostgreSQL)、对象存储服务(MinIO)和缓存服务器(Redis)的基础环境，为后续的具体应用开发快速架设了所需的服务。所有服务的数据都通过挂载宿主机目录的方式进行了持久化处理，保证了服务重启后数据不丢失。

5.2 FastAPI 接口开发

因为本文场景中对于数据库的增删改查需求并不复杂，因此笔者希望尽量使用轻量化的 Web 框架，所用框架只需提供简单的 RestfulAPI 处理能力，其他与数据库模型或者存储的交互将在 5.4 节其他工具模块进行开发；而具体的训练与推理任务的执行将在 5.3 节 Celery 任务队列中实现。以此保证接口部分代码的简介。同时将网络通信、运算与存储解耦的设计模式也更符合后续可能扩展到微服务架构的设计理念。

在 FastAPI app 的全局应用中我们实现了以下功能：

- 认证: 使用了自定义的依赖方法函数来实现 HTTPBasic 用户认证。
- 数据库操作: 通过 query_data 和 update_data 函数与数据库交互，管理数据集和模型的元数据。这部分将在 5.4 节具体介绍
- MinIO 交互: 应用程序直接与 MinIO 进行交互，上传和下载文件，在 FastAPI

全局中实例化了 MinIO Client 连接，实现在 FastAPI 应用中集成对象存储服务。

- 异步任务处理: 利用了 Celery 进行异步任务调度，特别是在模型训练和图像推理这样的耗时操作上，提高了应用响应性。
- 数据处理: 使用 Pandas 进行数据处理，便于将数据结构化并写入数据库。

这是一个涉及文件上传、数据验证、模型管理和异步任务处理的复杂应用，适合于本文需要进行的模型训练和推理的服务场景。

具体地，笔者使用 FastAPI 应用程序开发了以下后端系统所需的接口：

5.2.1 健康检查 /ping

/ping 路由返回一个简单的欢迎信息，用于确认服务运行正常。

5.2.2 上传 ZIP 文件数据集 /upload-zip/

/upload-zip/ 路由接收一个在 form-data 中的 ZIP 文件上传以及一个 annotation 注释。之后改处理函数将解压 zip 文件并使用本文 4.1.1 节中的 BasicDataset 类尝试加载该数据集以验证该数据集的合法性。如果通过验证则将原文件上传至 MinIO 对象存储，同时更新数据库记录文件信息。其中数据库更新的工具将在 5.4 节进行介绍。完成所有操作后，函数将清理之前解压的文件以保持文件存储的空闲。

此功能还包括了基于 Basic Auth 的认证逻辑，确保操作由已认证的用户执行。

5.2.3 模型训练任务 /train_model/

/train_model/ 路由接受一个 JSON 体形式的配置字典，用于定制模型训练参数。之后它会根据配置从数据库和 MinIO 中下载必要的数据集和模型（如果指定的话），最后启动一个 Celery 异步任务（cl_training）进行持续学习模型训练，并记录训练任务的状态到数据库中。此路由同样需要用户认证。此处的持续学习模型训练 Celery 任务将在 5.3 节详细介绍。

5.2.4 图像推理 /infer_image/{model_id}

/infer_image/{model_id} 路由接收单个图像文件，并指定模型 ID 进行推理。它首先验证模型是否存在并从 MinIO 下载模型，然后使用 Celery 异步任务（infer）执行推理，并直接返回推理结果作为一个文件响应。推理过程同样遵循特定的配置参数。推理完成后，临时文件会被清理。此功能也实现了认证逻辑。

5.3 Celery 任务队列开发

在 celery_worker.py 中，笔者实现了一个 Celery 应用以执行由异步的 FastAPI 申明的模型训练与模型推理任务：cl_training 和 infer。

这个应用利用了 Celery 进行分布式任务调度；使用 Avalanche 库进行持续学习（Continual Learning）；使用 PyTorch 进行深度学习模型训练与推理；使用 MinIO 进行

模型 checkpoint 的云存储上传；使用 Wandb 进行实验日志记录。以下是这两个任务实现的主要功能概述：

5.3.1 模型训练任务 cl_training

首先需要初始化随机种子以确保模型训练的可复现性，即使是从 checkpoint 中加载之前的模型也需要确保随机种子不变。这个功能由 Avalanche 内置的 RNGManager 实现，他不仅固定了 Avalanche 框架中的随机种子甚至还一同固定了 numpy, pandas 等常用 python 第三库的随机种子值，十分可靠。

之后的流程与 4.1 节中的大致相同：构建 U-Net 模型与日志记录器（Wandb，logger）；根据 Fastapi 接收到的配置参数创建一个持续学习策略（包括在第 4 章提到的所有五种学习策略）；配置评估插件以监控损失和时间指标。

接着执行检查点加载：如果 FastAPI 提供了模型 checkpoint 的信息，那么 Celery 将从 MinIO 中下载 checkpoint 并以继续训练。

接下来进行数据的准备：使用 4.1.1 节中提到的 BasicDataset 类加载图像数据，并将其包装为 AvalancheDataset 以支持连续学习场景。

所有前置准备完成，可以开始训练循环了：遍历 Benchmark 中的训练流，对每个子任务（experience）执行训练。训练完成后，保存当前策略状态到本地文件，并通过 MinIO 客户端上传到云存储。

最后更新数据库中的记录，标记任务完成状态及存储的 MinIO 路径。并删除已上传的本地检查点文件以节省空间。

5.3.2 模型推理任务 infer

在初始化与模型加载中类似于 cl_training，但目的是进行推理。加载模型、设置日志器，并根据需要加载训练好的策略检查点。随后将 FastAPI 接收的二进制图像数据，转换为 Torch Tensor，关闭模型的梯度计算并在模型上执行前向传播（forward）。

不同于 cl_training，推理任务需要返回人类可读的输出，所以需要更多的结果后处理。具体地，模型输出的维数为[batch_size,class_number,w,h]而笔者在第二维度上使用 argmax 将多维特征图转换为单通道的类别标签图，并将其输出维数调整为[w,h,1]。最后将输出保存为 PNG 图像文件，并向 FastAPI 返回该文件的路径。之后由 FastAPI 完成文件的网络传输任务。

5.4 其他工具开发

此模块包含了各种实用的函数方法。

其中 db_utils.py 完成了与 postgresql 数据库的交互。由于笔者希望保持后端的轻量化与可维护性，在后端代码中没有声明具体地数据模型，这使得维护人员可以使用他们熟悉的数据库连接软件动态修改数据表的结构，而无需关注此处的模型声明。在日常的

数据表增删改查中可以简单的使用 pandas DataFrame 的 sql 相关方法完成。该脚本中的数据库相关函数由 Celery 和 FastAPI 应用共享。

在 train_policy.py 文件中，实现了在 4.1.2 节中提到的客制化 SGLFL 持续学习策略，以及一个 get_strategy 函数。此函数根据配置字典中的 cl_strategy 选择并返回一个特定的持续学习策略实例。支持的策略包括：

- NAIVE：普通的多任务学习，不特别处理遗忘问题；
- EWC：Elastic Weight Consolidation，通过在线维护任务特定的重要性权重来减轻遗忘；
- LFL：使用前面定义的 SGLFL 类，结合了 Split Gradient Learning 和 Learning Without Forgetting。
- SI：Synaptic Intelligence，通过跟踪权重更新的重要性来调整优化过程。
- GEM：Gradient Episodic Memory，存储过往任务的梯度并约束当前梯度更新，以避免与过去知识冲突。

函数内部根据所选策略，初始化相应的实例，如 Naive, EWC, SGLFL, Synaptic Intelligence, 或 GEM，并传入必要的配置参数（如模型、优化器、损失函数等），以及一个评估插件（eval_plugin）。所有策略都配置了在"CUDA:0"设备上运行，意味着它们默认使用 GPU 进行训练和评估。

5.5 Postman 测试工具

Postman 是一款功能强大的 API 开发和测试工具，它允许用户轻松地构建、测试、文档化和共享 RESTful API。Postman 可以模拟各种 HTTP 请求（GET、POST、PUT、DELETE 等），帮助开发者和测试人员调试后端 API 接口。用户可以直接在界面中输入请求 URL、设置 HTTP 头部、添加请求体内容（如 JSON、XML、表单数据等），并发送请求查看响应结果。由于 Postman 支持从简单的手动测试到复杂的自动化测试场景，它已成为 API 开发和测试流程中不可或缺的工具。

在本文的系统开发过程中，使用了 Postman 来测试 FastAPI 的接口可用性。并将相关 collection 导出为 JSON 文件，方便运维与其他开发人员进行测试。

6 总结与展望

本文从焊接缺陷检测的场景出发，根据其实际面临的挑战希望能够搭建一个系统，不仅能基于超声相控阵系统完成缺陷的探测，还能够适应数据分布的不断变化，持续地优化已有模型。

为了实现以上所提到的功能，笔者对广泛应用于其他深度学习领域的持续学习方法在焊接缺陷检测的具体任务中进行评估：首先验证了灾难性遗忘的出现：实验通过比较使用不同持续学习策略的模型在连续学习五个子任务时的表现，证实了 NAIVE（无特殊策略）方法确实导致了在首个任务上性能的显著衰退，最高达到 23.47 倍的性能倒退，确立了灾难性遗忘现象的存在。之后对 4 种不同的持续学习策略进行比较，发现各策略在本文的场景中各有优劣：EWC 和 SI 在任务较少时表现良好但扩展性有限；LFL 虽然在特定场景下优化遇到挑战，但在抗遗忘方面表现出潜力；GEM 提供了强大的遗忘缓解能力，但伴随高昂的计算成本。

最后基于实际场景开发了一个持续学习模型的推理与训练平台。在存储与中间件方面使用 Postgresql 数据库存储结构化的数据集与模型元数据；使用 MinIO 存储二进制的数据集与模型检查点文件；使用 Redis 缓存任务队列，确保 Web 的可达性。使用 FastAPI 开发了健康检查、数据集上传、模型训练、模型推理四个 endpoint；使用 Celery 实现了模型训练与模型推理两个异步任务的执行。

目前的实验与系统只包含了在域持续学习中常用的 5 中持续学习方法，希望在未来可以扩展到更多的先进持续学习算法；同时目前的实验与系统也只包含 UNet 一个基础模型，希望在之后的研究中能够使用其他的先进模型进行实验与部署。

参考文献

[1] Lowe D G. Object recognition from local scale-invariant features[C]//Proceedings of the seventh IEEE international conference on computer vision. Ieee, 1999, 2: 1150-1157.

[2] Dalal N, Triggs B. Histograms of oriented gradients for human detection[C]//2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05). Ieee, 2005, 1: 886-893.

[3] Zhang Y, Tian Y, Kong Y, et al. Residual dense network for image restoration[J]. IEEE transactions on pattern analysis and machine intelligence, 2020, 43(7): 2480-2495.

[4] Yang J, Qi Z, Shi Y. Learning to incorporate structure knowledge for image inpainting[C]//Proceedings of the AAAI conference on artificial intelligence. 2020, 34(07): 12605-12612.

[5] Xie S, Tu Z. Holistically-nested edge detection[C]//Proceedings of the IEEE international conference on computer vision. 2015: 1395-1403.

[6] Li K, Tian Y, Wang B, et al. Bi-directional pyramid network for edge detection[J]. Electronics, 2021, 10(3): 329.

[7] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[8] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[9] Sun R, Zhu X, Wu C, et al. Not all areas are equal: Transfer learning for semantic segmentation via hierarchical region selection[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 4360-4369.

[10] Liznerski P, Ruff L, Vandermeulen R A, et al. Explainable deep one-class classification[J]. arXiv preprint arXiv:2007.01760, 2020.

[11] Venkataramanan S, Peng K C, Singh R V, et al. Attention guided anomaly localization in images[C]//European Conference on Computer Vision. Cham: Springer International Publishing, 2020: 485-503.

[12] Božič J, Tabernik D, Skočaj D. Mixed supervision for surface-defect detection: From weakly to fully supervised learning[J]. Computers in Industry, 2021, 129: 103459.

[13] Pang G, Ding C, Shen C, et al. Explainable deep few-shot anomaly detection with deviation networks[J]. arXiv preprint arXiv:2108.00462, 2021.

[14] Wan Q, Gao L, Li X. Logit inducing with abnormality capturing for semi-supervised

image anomaly detection[J]. IEEE Transactions on Instrumentation and Measurement, 2022, 71: 1-12.

[15] Kabiraj A, Pal D, Ganguly D, et al. Number plate recognition from enhanced super-resolution using generative adversarial network[J]. Multimedia Tools and Applications, 2023, 82(9): 13837-13853.

[16] Jin B, Cruz L, Gonçalves N. Deep facial diagnosis: deep transfer learning from face recognition to facial diagnosis[J]. IEEE Access, 2020, 8: 123649-123661.

[17] Zhao M, Liu Q, Jha A, et al. VoxelEmbed: 3D instance segmentation and tracking with voxel embedding based deep learning[C]//Machine Learning in Medical Imaging: 12th International Workshop, MLMI 2021, Held in Conjunction with MICCAI 2021, Strasbourg, France, September 27, 2021, Proceedings 12. Springer International Publishing, 2021: 437-446.

[18] Yao T, Qu C, Liu Q, et al. Compound figure separation of biomedical images with side loss[C]//Deep Generative Models, and Data Augmentation, Labelling, and Imperfections: First Workshop, DGM4MICCAI 2021, and First Workshop, DALI 2021, Held in Conjunction with MICCAI 2021, Strasbourg, France, October 1, 2021, Proceedings 1. Springer International Publishing, 2021: 173-183.

[19] Li H, Zhao R, Wang X. Highly efficient forward and backward propagation of convolutional neural networks for pixelwise classification[J]. arXiv preprint arXiv:1412.4526, 2014.

[20] Shelhamer E, Long J, Darrell T. Fully convolutional networks for semantic segmentation[J]. IEEE Trans. Pattern Anal. Mach. Intell., 2017, 39(4): 640-651.

[21] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

[22] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[J]. Communications of the ACM, 2017, 60(6): 84-90.

[23] Karen S, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv 2014[J]. arXiv preprint arXiv:1409.1556, 2014.

[24] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.

[25] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2818-2826.

[26] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern

recognition. 2016: 770-778.

[27] Badrinarayanan V, Kendall A, Cipolla R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation[J]. IEEE transactions on pattern analysis and machine intelligence, 2017, 39(12): 2481-2495.

[28] Wang P, Chen P, Yuan Y, et al. Understanding convolution for semantic segmentation[C]//2018 IEEE winter conference on applications of computer vision (WACV). Ieee, 2018: 1451-1460.

[29] Maaten L, Huang G, Liu Z, et al. Densely connected convolutional networks[C]. CVPR, 2017.

[30] Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation[C]//Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. Springer International Publishing, 2015: 234-241.

[31] Mnih V, Heess N, Graves A. Recurrent models of visual attention[J]. Advances in neural information processing systems, 2014, 27.

[32] Visin F, Ciccone M, Romero A, et al. Reseg: A recurrent neural network-based model for semantic segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2016: 41-48.

[33] Visin F, Kastner K, Cho K, et al. Renet: A recurrent neural network based alternative to convolutional networks[J]. arXiv preprint arXiv:1505.00393, 2015.

[34] Byeon W, Breuel T M, Raue F, et al. Scene labeling with lstm recurrent neural networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 3547-3555.

[35] Liang X, Shen X, Feng J, et al. Semantic object parsing with graph lstm[C]//Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14. Springer International Publishing, 2016: 125-143.

[36] Zhao H, Zhang Y, Liu S, et al. Psanet: Point-wise spatial attention network for scene parsing[C]//Proceedings of the European conference on computer vision (ECCV). 2018: 267-283.

[37] Fu J, Liu J, Tian H, et al. Dual attention network for scene segmentation[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 3146-3154.

[38] He J, Deng Z, Zhou L, et al. Adaptive pyramid context network for semantic

segmentation[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 7519-7528.

[39] Wang J, Chen K, Xu R, et al. Carafe: Content-aware reassembly of features[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2019: 3007-3016.

[40] Wang J, Chen K, Xu R, et al. Carafe++: Unified content-aware reassembly of features[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021, 44(9): 4674-4687.

[41] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

[42] Weissenborn D, Täckström O, Uszkoreit J. Scaling autoregressive video models[J]. arXiv preprint arXiv:1906.02634, 2019.

[43] Cordonnier J B, Loukas A, Jaggi M. On the relationship between self-attention and convolutional layers[J]. arXiv preprint arXiv:1911.03584, 2019.

[44] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale[J]. arXiv preprint arXiv:2010.11929, 2020.

[45] Liu Z, Lin Y, Cao Y, et al. Swin transformer: Hierarchical vision transformer using shifted windows[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2021: 10012-10022.

[46] Chen Z, Liu B. Lifelong machine learning[M]. San Rafael: Morgan & Claypool Publishers, 2018.

[47] French R M. Catastrophic forgetting in connectionist networks[J]. Trends in cognitive sciences, 1999, 3(4): 128-135.

[48] Kudithipudi D, Aguilar-Simon M, Babb J, et al. Biological underpinnings for lifelong learning machines[J]. Nature Machine Intelligence, 2022, 4(3): 196-210.

[49] Lee C S, Lee A Y. Clinical applications of continual learning machine learning[J]. The Lancet Digital Health, 2020, 2(6): e279-e281.

[50] Shaheen K, Hanif M A, Hasan O, et al. Continual learning for real-world autonomous systems: Algorithms, challenges and frameworks[J]. Journal of Intelligent & Robotic Systems, 2022, 105(1): 9.

[51] Philps D, Weyde T, Garcez A A, et al. Continual learning augmented investment decisions[J]. arXiv preprint arXiv:1812.02340, 2018.

[52] Lopez-Paz D, Ranzato M A. Gradient episodic memory for continual learning[J]. Advances in neural information processing systems, 2017, 30.

[53] Aljundi R, Chakravarty P, Tuytelaars T. Expert gate: Lifelong learning with a network of experts[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 3366-3375.

[54] Von Oswald J, Henning C, Grewe B F, et al. Continual learning with hypernetworks[J]. arXiv preprint arXiv:1906.00695, 2019.

[55] Wortsman M, Ramanujan V, Liu R, et al. Supermasks in superposition[J]. Advances in Neural Information Processing Systems, 2020, 33: 15173-15184.

[56] Lomonaco V, Maltoni D. Core50: a new dataset and benchmark for continuous object recognition[C]//Conference on robot learning. PMLR, 2017: 17-26.

[57] Mirza M J, Masana M, Possegger H, et al. An efficient domain-incremental learning approach to drive in all weather conditions[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022: 3001-3011.

[58] Shin H, Lee J K, Kim J, et al. Continual learning with deep generative replay[J]. Advances in neural information processing systems, 2017, 30.

[59] Li Z, Hoiem D. Learning without forgetting[J]. IEEE transactions on pattern analysis and machine intelligence, 2017, 40(12): 2935-2947.

[60] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network[J]. arXiv preprint arXiv:1503.02531, 2015.

[61] Jung H, Ju J, Jung M, et al. Less-forgetting learning in deep neural networks[J]. arXiv preprint arXiv:1607.00122, 2016.

[62] Robins A. Catastrophic forgetting, rehearsal and pseudorehearsal[J]. Connection Science, 1995, 7(2): 123-146.

[63] Draelos T J, Miner N E, Lamb C C, et al. Neurogenesis deep learning: Extending deep networks to accommodate new classes[C]//2017 international joint conference on neural networks (IJCNN). IEEE, 2017: 526-533.

[64] Cortes C, Gonzalvo X, Kuznetsov V, et al. Adanet: Adaptive structural learning of artificial neural networks[C]//International conference on machine learning. PMLR, 2017: 874-883.

[65] Yoon J, Yang E, Lee J, et al. Lifelong learning with dynamically expandable networks[J]. arXiv preprint arXiv:1708.01547, 2017.

[66] Rusu A A, Rabinowitz N C, Desjardins G, et al. Progressive neural networks[J]. arXiv preprint arXiv:1606.04671, 2016.

[67] Carta A, Pellegrini L, Cossu A, et al. Avalanche: A pytorch library for deep continual learning[J]. Journal of Machine Learning Research, 2023, 24(363): 1-6.

[68] Kirkpatrick J, Pascanu R, Rabinowitz N, et al. Overcoming catastrophic forgetting in

neural networks[J]. Proceedings of the national academy of sciences, 2017, 114(13): 3521-3526.

[69] Zenke F, Poole B, Ganguli S. Continual learning through synaptic intelligence[C]//International conference on machine learning. PMLR, 2017: 3987-3995

[70] Lopez-Paz D, Ranzato M A. Gradient episodic memory for continual learning[J]. Advances in neural information processing systems, 2017: 30.

致谢

在我完成这项基于深度学习管道的多场景行人检测与分析系统的过程中，得到了众多的支持与帮助，现在我要向他们表达我最真挚的感谢。

首先，我要感谢我的导师。在整个研究过程中，导师给予了我耐心的指导和宝贵的建议，使我能够顺利地完成研究任务。导师严谨的治学态度和深厚的学术造诣深深地影响了我，成为我今后学习和工作的楷模。

其次，我要感谢我的实习导师和其他团队成员。在研究过程中，他们与我共同探讨问题，互相学习，共同进步。他们的支持和帮助使我能够克服研究中的困难和挑战， 取得更好的成果。

此外，我还要感谢开源数据集和模型的开发者们。没有他们提供的框架与模型，我无法完成这项研究。他们的无私奉献和开源精神激励着我，让我更加坚定共同维护，共同发展的开源信念。

最后，我要感谢我的家人和朋友。他们一直支持我，鼓励我，给予我无尽的爱和关怀。没有他们的支持，我无法走到今天。我将永远珍惜他们的付出和关爱，为共同的美好未来而努力奋斗。

再次感谢所有支持和帮助过我的人，谢谢你们！

卜宜凡

2024 年 05 月 04 日