華東理工大學

毕业设计(论文)
开题报告

题　　目	基于图像语义分割焊接缺陷检测的持续学习方法研究
学　　院	信息科学与工程学院
系　　别	自动化系
专　　业	智能科学与技术
年　　级	2020 级
学　　号	20002351
姓　　名	卜宜凡
导　　师	周家乐

定稿日期：　**2024** 年 **3** 月 **22** 日

基于图像语义分割焊接缺陷检测的持续学习方法研究

智能 202 卜宜凡（20002351）

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，我们希望对已有的持续学习方法进行评估和优化使其能够适应燃气管道焊接缺陷检测的场景。

关键词：持续学习，缺陷检测，机器视觉，深度学习

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，我们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，我们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。

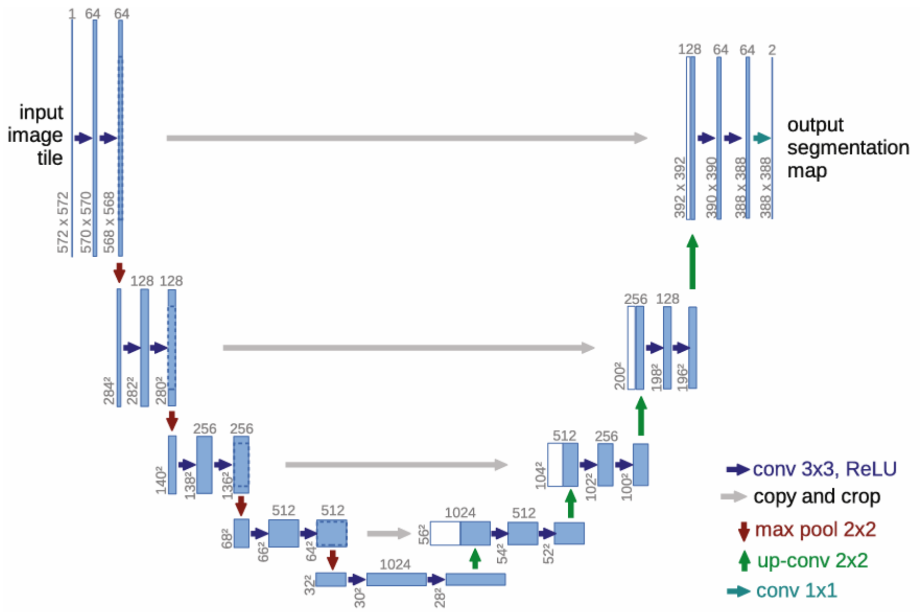

图 2-1 U-Net 结构图

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。

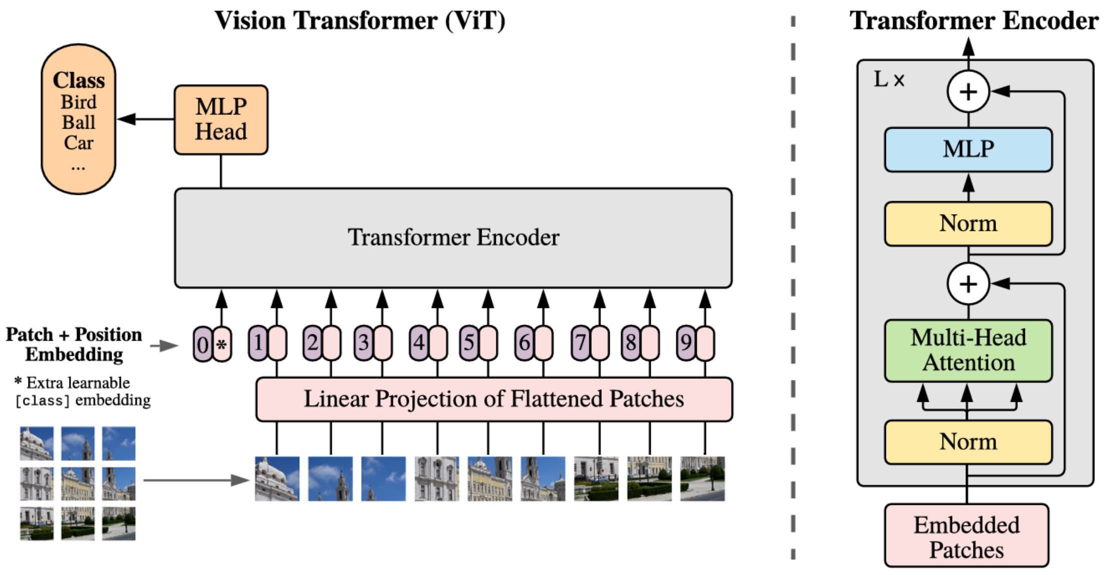

图 2-2 ViT 结构图

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。

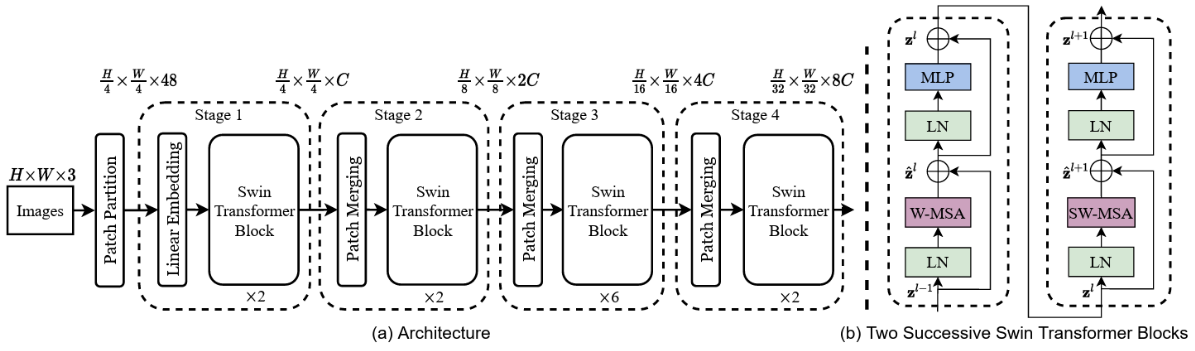

图 2-3 Swin Transformer 结构图

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) 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：

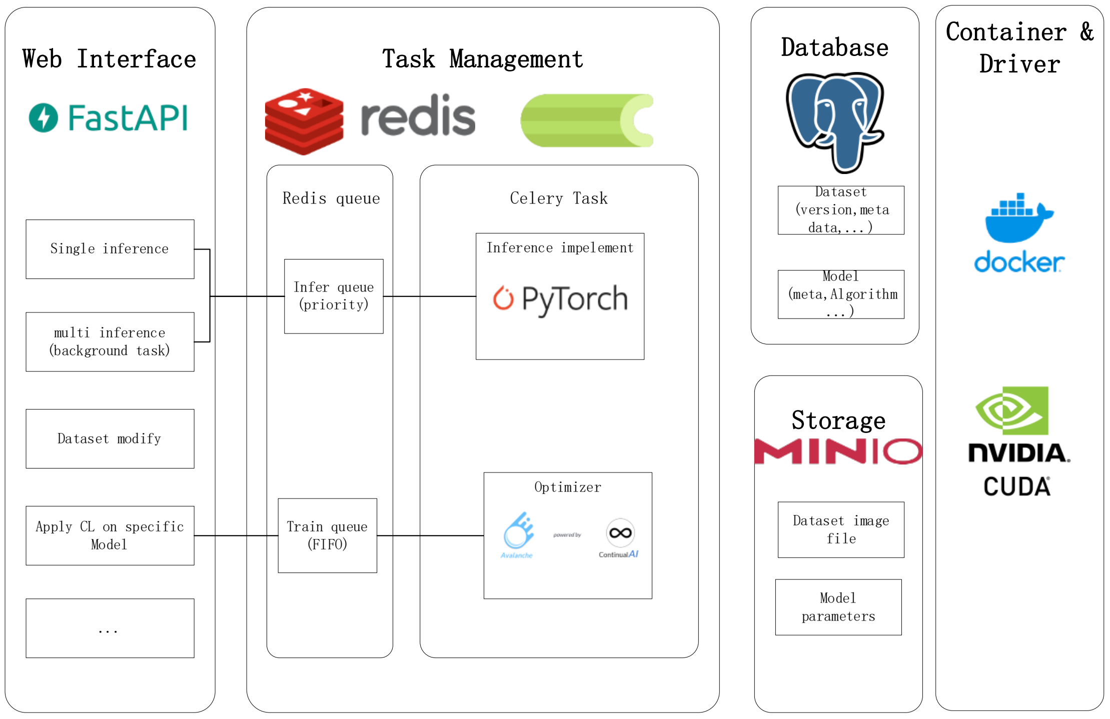

图 3-1 检测模型推理与训练平台架构图

4 进度安排

第 1~2 周	熟悉环境，对论文有初步的了解。
第 3~5 周	查阅文献，完成文献翻译及开题报告。
第 6~10 周	对持续学习方法进行实验比较。
第 11~14 周	完成软件架构的搭建并实现业务逻辑
第 15~18 周	撰写论文。

5 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

基于智能突触的持续学习方法

Friedemann Zenke, Ben Poole, Surya Ganguli

摘要

虽然深度学习在许多应用场景中取得了显著的进步，但当学习过程中数据分布发生变化时，深度学习仍然面临重大挑战。与之形成鲜明对比的是，生物神经网络通过利用复杂的分子机制能够同时解决许多任务，并不断适应变化的环境。在这项研究中，我们引入了智能突触，将这种生物复杂性的一部分引入到人工神经网络中。每个突触将随时间的推移逐渐积累与任务相关的信息，并利用这些信息在不遗忘旧记忆的情况下快速存储新记忆。我们对分类任务的持续学习方法进行了评估，结果表明，该方法在保持计算效率的同时，显著减少了信息遗忘。

1. 介绍

人工神经网络（ANNs）已经成为应用机器学习领域不可或缺的资产，在各种特定领域的任务中，其表现已经可以与人类相媲美（LeCun 等人，2015 年）。尽管人工神经网络最初是受生物学启发的（Rosenblatt，1958 年；Fukushima & Miyake，1982 年），但其底层的设计原则和学习方法与生物神经网络存在很大的不同。例如，人工神经网络的参数是在训练阶段的数据集上学习的，然后在部署或推理阶段，这些参数会被冻结并静态地应用于新数据。为了适应数据分布的变化，人工神经网络通常需要在整个数据集上重新训练，以避免过拟合和灾难性遗忘（Choy 等人，2006 年；Goodfellow 等人，2013 年）。

另一方面，生物神经网络却能够实现持续学习，它们在一生中都能不断获取新知识。因此，很难在学习和推理阶段之间划清界限。在某种程度上，我们的大脑已经进化到可以从非静态数据中学习，并即时更新内部记忆或信念。虽然目前还不知道大脑是如何完成这一壮举的，但生物所展现出的在持续学习中无与伦比的性能似乎依赖于底层生物湿件所实现的特定功能，而这些功能目前在人工神经网络中尚未被实现。

现代人工神经网络与生物神经网络在设计上最大的差距之一可能在于突触的复杂性。在人工神经网络中，单个突触（权重）通常由单个标量来描述。另一方面，单个生物突触则利用复杂的分子机制，这些机制可以在不同的空间和时间尺度上影响可塑性（Redondo & Morris，2011）。虽然这种复杂性被认为有助于记忆巩固（Fusi 等人，2005 年；Lahiri & Ganguli，2013 年；Zenke 等人，2015 年；Ziegler 等人，2015 年；Benna & Fusi，2016 年），但很少有研究说明它是如何有利于人工神经网络的学习的。

在这里，我们研究了内部突触动力学在使用人工神经网络学习分类任务序列方面的作用。在突触（权重）中使用简单的一维标量会导致灾难性遗忘问题，使网络在学习新任务时忘记先前学习的任务。然而，通过使用具有更复杂的三维状态空间的突触能够在很大程度上缓解这个问题。在我们的模型中，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。我们的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，我们通过防止重要突触在未来任务中发生变化来巩固它们。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，从而避免了对这些过去任务的灾难性遗忘。

2. 先前的工作

减轻灾难性遗忘的问题已在许多先前的研究中得到解决。这些研究可以大致划分为（1）架构性、（2）功能性和（3）结构性方法。

应对灾难性遗忘的**架构性**方法通过改变网络的架构来减少任务之间的干扰，而无需改变目标函数。架构性正则化的最简单形式是冻结网络中的某些权重，使它们保持不变（Razavian 等人，2014 年）。一种稍微宽松的方法是在对原始任务共享的层进行微调时降低学习率，以避免参数发生剧烈变化（Donahue 等人，2014 年；Yosinski 等人，2014 年）。使用不同的非线性方法，如 ReLU、MaxOut 和局部胜者优先，已被证明可以提高在 permuted MNIST 和情感分析任务上的性能（Srivastava 等人，2013 年；Goodfellow 等人，2013 年）。此外，使用 dropout 注入噪声以稀疏化梯度也可以提高性能（Goodfellow 等人，2013 年）。Rusu 等人（2016 年）最近的工作提出了更为剧烈的架构变化，其中在解决新任务时，会复制先前任务的整个网络并增加新特征。这完全防止了对早期任务的遗忘，但会导致架构复杂性随着任务数量的增加而增长。

功能性方法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（2016 年）的研究中，他们使用一种知识蒸馏的形式（Hinton 等人，2014 年），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（2016 年）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。

结构性正则化方法则涉及对参数的惩罚，鼓励它们保持接近旧任务的参数。最近，Kirkpatrick 等人（2017 年）提出了弹性权重巩固（EWC），这是一种对新旧任务参数差异的二次惩罚。他们使用了与旧任务上旧参数的 Fisher 信息度量的对角线成比例的权重。精确计算 Fisher 的对角线需要对所有可能的输出标签进行求和，因此其复杂性与输

出数量成线性关系。这限制了该方法在低维输出空间中的应用。

3. 突触架构

为了解决神经网络中的持续学习问题，我们试图构建一个简单的结构性正则化模块，可以在线计算并在每个突触处局部实现。具体而言，我们的目标是为每个突触赋予一个局部的“重要性”度量，以解决网络过去已经训练过的任务。在训练新任务时，我们会惩罚对重要参数的更改，以避免旧记忆被覆盖。为此，我们开发了一类算法，跟踪一个重要性度量ω_k^μ，它反映了过去对任务目标L_μ的改进对单个突触θ_k的贡献。为简洁起见，我们认为“突触”一词与“参数”一词同义，这包括层之间的权重以及偏置。

训练神经网络的过程在参数空间中表现为轨迹$\theta(t)$（图 1）。成功训练的特性在于找到学习轨迹，其终点应接近所有任务上损失函数 L 的最小值。让我们首先考虑在时间t处对无穷小的参数更新$\delta(t)$的损失变化。在这种情况下，损失的变化被很好地近似为梯度$g = \frac{\partial L}{\partial \theta}$，因此我们可以写出

$$L\big(\theta(t)+\delta(t)\big)-L\big(\theta(t)\big) \approx \sum_k g_k(t)\delta_k(t), \tag{1}$$

这表明了对于每个参数变化$\delta_k(t) = \theta_k^{'}(t)$对总损失变化的贡献值为$g_k(t)\delta_k(t)$。为了计算整个轨迹在参数空间上的损失变化，我们需要对所有无穷小的变化进行求和。这相当于计算从初始点（在时间t_0处）到最终点（在时间t_1处）的参数轨迹上的梯度向量场的路径积分：

$$\int_C g\big(\theta(t)\big)d\theta = \int_{t_0}^{t_1} g\big(\theta(t)\big)\cdot\theta'(t)dt. \tag{2}$$

由于梯度是一个保守场，积分值等于终点与起点之间损失的差值：$L\big(\theta(t_1)\big) - L\big(\theta(t_0)\big)$。对于我们的方法，关键在于可以将等式 2 分解为各个参数的总和。

$$\begin{aligned}\int_{t^{\mu-1}}^{t^\mu} g\big(\theta(t)\big)\cdot\theta'(t)dt &= \sum_k \int_{t^{\mu-1}}^{t^\mu} g_k\big(\theta(t)\big)\theta_k'(t)dt \\ &\equiv -\sum_k \omega_k^\mu. \end{aligned}\tag{3}$$

现在，我们可以直观地理解w_k^u为损失中特定参数的贡献。请注意，我们在第二行引入了负号，因为我们通常关心的是如何降低损失。

在实践中，我们可以通过计算梯度总和$g_k(t) = \frac{\partial L}{\partial \theta_k}$和参数更新量$\theta_k^{'}(t) = \frac{\partial \theta_k}{\partial t}$的乘积来在线得计算$w_k^u$的近似值。对于具有无穷小学习率的批量梯度下降情况，w_k^u可以直接解释为总损失中每个参数的变化量。在大多数情况下，真实梯度由随机梯度下降（SGD）近似，导致对g_k的估计中包含了噪声。这样的直接后果是，近似后参数重要

性通常会高估的w_k^u真实值。

如何利用w_k^u中的知识来改善持续学习的效果呢？我们需要解决的核心问题是：在无法访问过去训练任务的损失函数的情况下，如何最小化所有任务的总损失函数$\mathcal{L} = \sum_{\mu} L_{\mu}$。相反，在任何事后，我们只能访问单个任务$\mu$的损失函数$\mathcal{L}_{\mu}$。在最小化$\mathcal{L}_{\mu}$的过程中可能会无意地导致先前任务的代价$\mathcal{L}_{v}(v < \mu)$大幅增加，这时灾难性遗忘就会出现（图 1）。

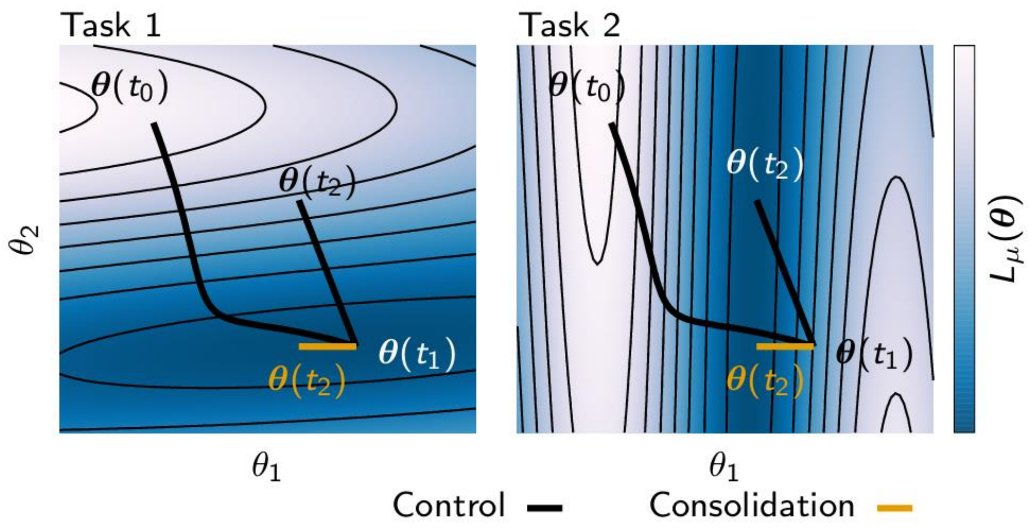

图 1 参数空间轨迹和灾难性遗忘的示意图。实线对应训练期间的参数轨迹。左右图对应不同任务(Task 1 和 Task 2)定义的不同损失函数。每个损失函数的值L_{μ}显示为热图。任务 1 上的梯度下降在参数空间中诱导了一个从$\theta(t_0)$到$\theta(t_1)$的运动轨迹。任务 2 上的梯度下降产生了参数空间中从$\theta(t_1)$到$\theta(t_2)$的运动。最后点以显著增加任务 1 的损失为代价，将任务 2 的损失最小化，从而导致对任务 1 的灾难性遗忘。然而，确实存在一个另一个替代点$\theta(t_2)$，用橙色标记，它在两个任务中都实现了小的损失。在下面的文章中，我们将展示如何通过确定比任务 1 中参数θ_1更重要的参数θ_2，并防止θ_2发生太大变化，来找到这个替代点。这产生了一种通过保持对解决过去任务很重要的参数并允许不重要的参数学习解决未来的任务来避免灾难性遗忘的在线方法。

为了避免所有先前任务$(v < \mu)$的灾难性遗忘，我们希望在训练新任务时避免对过去具有重要影响的权重发生剧烈变化。单个任务的参数θ_k的重要性由两个量决定：

1）单个参数在训练轨迹的整个过程中对损失的下降贡献程度w_k^v(公式 3)；

2）该参数的移动距离$\Delta_k^{\nu} \equiv \theta_k(t^{\nu}) - \theta_k(t^{\nu-1})$。

为了避免对重要参数产生大的变化，我们使用修改后的代价函数$\mathcal{L}$，其中引入了一个修改后的损失$\tilde{L}_{\mu}$，该损失近似于先前任务的累积损失函数$\mathcal{L}_{v}(v < \mu)$。具体来说，我们使用了一个二次代理损失，其最小值与前一个任务的代价函数相同，并且在参数距离Δ_k上产生的w_k^u相同。换句话说，如果在学习过程中使用我们修改后的损失而不是实际的损失函数，那么在训练期间将导致相同的最终参数和损失变化（图 2）。

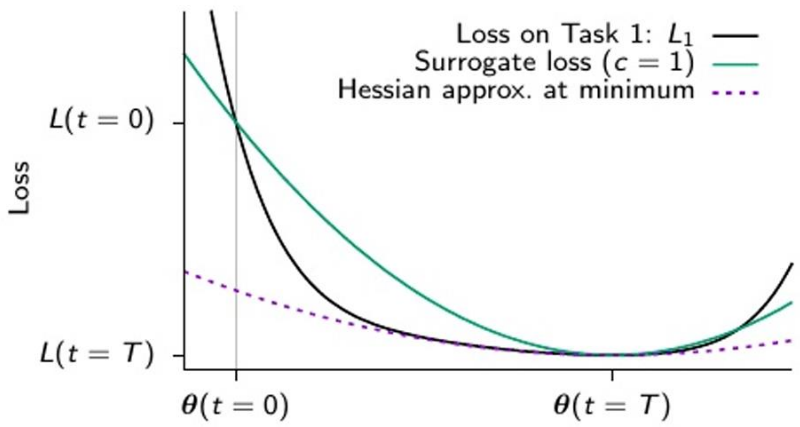

图 2 学习一项任务后代理损失的示意图。考虑任务 1(黑色)定义的一些损失函数。选择二次代理损失(绿色)是为了精确匹配原始损失函数上梯度下降的 3 个方面:损失函数$L(\theta(0)) - L(\theta(T))$中的总梯度，参数空间$\theta(0) - \theta(T)$中的净运动，以及在端点$\theta(T)$实现最小值。这 3 个条件唯一地决定了替代二次损失，它总结了原始损失上的下降轨迹。请注意，这个代理损失不同于 Hessian 在最小值处定义的二次近似(紫色虚线)。

对于两个不同任务，这正是通过以下二次代理损失实现的：

$$\tilde{L}_\mu = L_\mu + c \sum_k \Omega_k^\mu (\tilde{\theta}_k - \theta_k)^2 \tag{4}$$

其中我们引入了无量纲强度参数 c，上一个任务结束时参数对应的参考权重$\tilde{\theta}_k = \theta_k(t^{u-1})$以及每个参数的正则化强度：

$$\Omega_k^\mu = \sum_{\nu<\mu} \frac{\omega_k^\nu}{(\Delta_k^\nu)^2 + \xi} \tag{5}$$

请注意，分母中的项$(\Delta_k^\nu)^2$确保正则化项与损失L具有相同的单位。在实践中，我们还引入了一个额外的阻尼参数ξ，以确保在$(\Delta_k^\nu)^2 = 0$时表达式有意义。最后，c是强度参数，用于平衡旧记忆和新记忆。如果路径积分（公式 3）的估计是精确的，那么此时旧记忆和新记忆的权重应该相等（$c = 1$）。然而，由于路径积分（公式 3）估计中噪声的存在，c通常都会选择选择小于 1 的值。除非另有说明，ω_k在训练过程中不断更新，而累积重要性测量Ω_k^μ和参考权重$\tilde{\theta}$仅在每个任务结束时更新。更新Ω_k^μ后，ω_k被设为零。尽管我们作为代理损失（公式 4）仅在两个任务的情况下有意义，但我们将通过经验证明我们的方法在学会更多额外任务时表现出良好的性能。

为了理解（公式 4）和（公式 5）如何影响学习过程，让我们考虑图 1 中所示学习两个任务的示例，其中我们。我们首先在 $Task$ 1 上进行训练。在时间t_1时，参数已接近 $Task$ 1 损失L_1的局部最小值。但是，相同的参数配置对于 $Task$ 2 并不接近最小值。因此，当在 $Task$ 2 上进行训练而没有任何其他策略时，L_1损失可能会无意中增加（图 1，黑色轨迹）。然而，当θ_2“记住”降低L_1是重要的要素时，它可以在 $Task$ 2 的训练

过程中利用这一知识，并使参数保持接近其当前值（图 1，橙色轨迹）。虽然这几乎不可避免地会导致$Task$ 2 的性能下降，但这种下降相较于两个任务组合的实质性能提升是可以忽略不计的。

这里提出的方法与 EWC (Kirkpatrick 等人，2017) 类似，即更有影响力的参数被更强烈地拉回到参考权重上，这使得以前的良好性能得以实现。然而，与 EWC 不同，我们在这里提出一种在线计算重要性度量的方法，该方法沿整个学习轨迹进行计算，而 EWC 则依赖于 Fisher 信息度量对角线的点估计值，该度量值必须在每个任务的末尾单独计算。

4. 特定情景下的理论分析

在以下部分，我们将说明我们的通用方法包括一个在简单且可分析的训练场景中合理且有意义的 Ω^{μ}_{k}。为此，我们分析了参数路径积分ω_u^k和其归一化版本Ω_k^μ（等式5）在简单二次误差函数的几何意义上所对应的内容：

$$E(\theta)=\frac{1}{2}(\theta-\theta^*)^T H(\theta-\theta^*), \tag{6}$$

该误差函数在θ^*处取得最小值，并且具有 Hessian 矩阵$\boldsymbol{H}$。进一步考虑在这个误差函数上的批量梯度下降，在离散时间学习率很小的极限情况下，这种梯度下降能够由连续时间微分方程描述：

$$\tau\frac{d\theta}{dt}=-\frac{\partial E}{\partial\theta}=-H(\theta-\theta^*), \tag{7}$$

其中τ与学习率有关，如果我们从初始条件$\theta(0)$，时间 $t = 0$处开始，下降路径的精确解由以下方程给出：

$$\theta(t)=\theta^*+e^{-H\frac{t}{\tau}}(\theta(0)-\theta^*), \tag{8}$$

这给出了时间依赖的更新方向:

$$\theta'(t)=\frac{d\theta}{dt}=-\frac{1}{\tau}He^{-H\frac{t}{\tau}}(\theta(0)-\theta^*). \tag{9}$$

现在，在梯度下降动力学中，梯度满足$g = \tau\frac{d\boldsymbol{\theta}}{dt}$，因此 (3) 中的$\omega_k^\mu$是通过矩阵的对角线元素来计算的

$$Q=\tau\int_0^\infty dt\frac{d\theta}{dt}\frac{d\theta}{dt}^T. \tag{10}$$

$\boldsymbol{Q}$ 的显式公式可以通过 Hessian 矩阵H的特征基来给出。具体来说，λ_α和u_α分别表示 $\boldsymbol{H}$ 的特征值和特征向量，使$d^\alpha=u^\alpha\cdot(\theta(0)-\theta^*)$表示初始参数和最终参数之间的差异在第$\alpha$个特征向量上的投影。然后，将 (公式 9) 代入 (公式 10)，进行基变换得到 $\boldsymbol{H}$ 的特征模态，并执行积分，得到：

$$Q_{ij}=\sum_{\alpha\beta}\mathrm{m}u_i^{\alpha}\,d^{\alpha}\frac{\lambda^{\alpha}\lambda^{\beta}}{\lambda^{\alpha}+\lambda^{\beta}}d^{\beta}u_j^{\beta}. \tag{11}$$

需要注意的是，作为时间积分的稳态量，$\boldsymbol{Q}$不再依赖于决定下降路径速度的时间常数 τ 。

乍一看，$\boldsymbol{Q}$矩阵的元素以复杂的方式依赖于 Hessian 的特征向量和特征值，以及初始条件$\theta(0)$。为了理解这种依赖性，让我们首先考虑对随机初始条件$\theta(0)$的Q进行平均，这样偏差d^{α}的集合就构成了一组均值为 0，方差为σ^2随机变量。因此，我们有平均值$\langle d^{\alpha}d^{\beta}\rangle=\sigma^2\delta_{\alpha\beta}$。然后，对 Q 进行这种平均计算。

$$\langle Q_{ij}\rangle=\frac{1}{2}\sigma^2\sum_{\alpha}u_i^{\alpha}\lambda^{\alpha}u_j^{\beta}=\frac{1}{2}\sigma^2H_{ij}. \tag{12}$$

因此，值得注意的是，在对初始条件进行平均后，$\boldsymbol{Q}$矩阵只需通过在突触对之间关联参数更新并随时间进行积分即可获得。并可以化简为 Hessian 矩阵，直到指示初始和最终条件之间差异的标度因子。事实上，这个标度因子在理论上激发了(公式 5)中的归一化；在零阻尼时，(公式 5)中的分母ξ平均为σ^2，从而消除了(公式 12)中的标度因子σ^2。

然而，我们更关注Q_{ij}在单个初始条件下的计算。在两种情况下，$\boldsymbol{Q}$和 Hessian $\boldsymbol{H}$之间的简单关系在不平均初始条件的情况下得以保持。首先，考虑 Hessian 为对角矩阵的情况，即$u_i^{\alpha}=\delta_{\alpha i}e_i$，其中$e_i$是第$i$个坐标向量。然后$\alpha$和$i$的索引可以互换，Hessian 的特征值是 Hessian 的对角线元素：$\lambda^i=H_{ii}$。那么(公式 11)简化为：

$$Q_{ij}=\delta_{ij}\left(d^i\right)^2H_{ii}. \tag{13}$$

再次考虑(公式 5)中的归一化，在零阻尼时，消除了在参数空间中的运动范围$\left(d^i\right)^2$，因此归一化的Q矩阵与对角 Hessian 相同。在第二种情况下，考虑 Hessian 是秩为 1 的极端情况，使得λ^1是唯一的非零特征值。然后(11)简化为

$$Q_{ij}=\frac{1}{2}(d^1)^2u_i^1\lambda_1u_j^1=\frac{1}{2}(d^1)^2H_{ij}. \tag{14}$$

因此，Q 矩阵再次简化为一个有比例因子的 Hessian 矩阵。归一化后的重要性度量变成了非对角但低秩 Hessian 矩阵的对角元素。我们注意到，对于连续学习而言，低秩 Hessian 十分有意义；误差函数中的低秩结构使得许多突触权重空间的方向不受给定任务的约束，从而为突触参数修改留下了解决未来任务而不会干扰旧任务性能的额外容量。

重要的是要强调，重要性路径积分是通过在整个学习轨迹上沿信息进行积分来计算的（参见图 2)。对于二次损失函数，Hessian 在整个轨迹上是常数，因此我们找到了重要性Ω_k^{μ}和 Hessian 在学习的终点之间精确的关系。但对于更一般的损失函数(Hessian 沿着轨迹发生变化)我们不能期望在学习的终点处的重要性Ω_k^{μ}和 Hessian 或其他相关参数的敏感性度量（Pascanu & Bengio，2013；Martens，2016；Kirkpatrick et al.，2017）

之间存在简单的数学对应关系。然而，在实践中，我们发现我们的重要性度量与基于这种端点估计的度量相关，这可能解释了它们在下一节中我们将看到的有竞争力的有效性。

5. 实验

我们在分割和打乱版本的 MNIST（LeCun 等，1998；Good- fellow 等，2013）以及 CIFAR-10 和 CIFAR-100（Krizhevsky & Hinton，2009）的分割版本上评估了我们的连续学习方法。

5.1. 分割 MNIST

我们首先在分割的 MNIST 基准上评估了我们的算法。在这个基准中，我们将完整的 MNIST 训练数据集分为 5 个连续数字的子集。这 5 个任务对应于学习区分从 0 到 10 的两个连续数字。我们使用了一个小的多层感知器（MLP），它只有两个隐藏层，每个隐藏层由 256 个单元组成，使用 ReLU 非线性激活函数，以及一个标准的类别交叉熵。为了避免数字之间在输出层上的串扰，我们使用了多头方法，其中在输出层上的类别交叉熵损失仅计算当前任务中存在的数字。最后，我们使用 64 的最小批次大小来优化我们的网络，并训练了 10 个 epoch。为了在较少的 epoch 数下获得良好的绝对性能，我们使用了自适应优化器 Adam（Kingma & Ba，2014）（$\eta = 1 \times 10^{-3}, \beta 1 = 0.9, \beta 2 = 0.999$）。在这个基准测试中，每个任务训练后都会重置优化器状态。

为了评估性能，我们计算了之前所有任务的上平均分类准确率与训练任务数量的函数关系。现在，我们将比较我们开启（$c = 1$）和关闭（$c = 0$）巩固惩罚网络之间的性能。在训练第一个任务时，由于没有可用于正则化的过去经验，巩固惩罚对于两种情况都为零。当训练数字“2”和“3”（任务 2）时，具有巩固和无巩固惩罚的网络在任务 2 上都表现出接近于 1 的准确性。然而，无巩固网络的准确性在任务 1 上出现了大幅下降（图 3）。

相比之下，具有整合惩罚的网络在任务 1 上的准确性仅受到轻微损害，并且两个任务的平均准确性都接近于 1。同样，当网络看到所有 MNIST 数字时，没有整合的网络在任务 1 和前两个任务上的准确性平均值已降至随机水平，而具有整合的网络在这些任务上的性能仅出现轻微下降（图 3）。

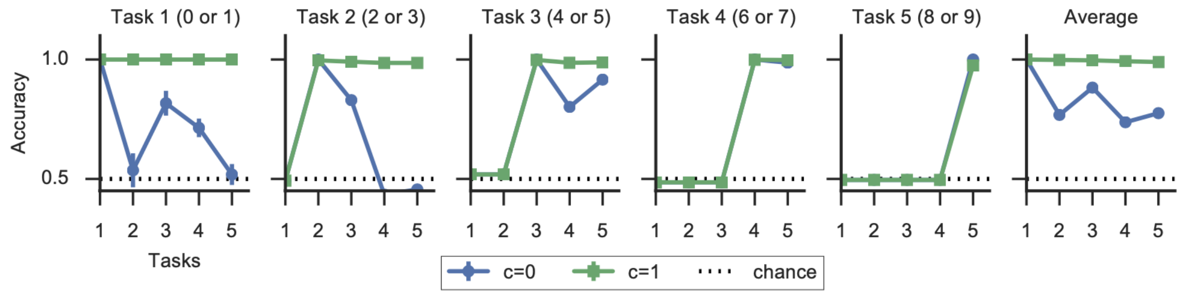

图 3 分割 MNIST 基准测试中平均分类准确率随任务数量的变化。前五个图表显示由两个 MNIST 数字组成的五个任务的分类准确率，随连续任务数量的变化而变化。最右边的图表显示平均准确率，这是通过对过去任务的准确率进行平均计算得出的，其中$\nu < \mu$，μ由x轴上的任务数量给出。请注意，在实验中有多个二进制输出头，准确率为 0.5 对应于偶然水平。误差条对应于 SEM（n=10）。

5.2. 打乱 MNIST 基准测试

在这个基准测试中，我们对每个任务的对所有 MNIST 像素进行了不同的随机排列。我们使用具有两个隐藏层、每个隐藏层有 2000 个 ReLUs 激活和 softmax 损失的 MLP 进行训练。我们使用了与之前相同的 Adam 参数。然而，在这里我们使用了$\xi = 0.1$，并且通过在保留验证集上进行粗网格搜索确定了$c = 0.1$的值。我们将 mini 批次大小设置为 256，并训练了 20 个 epoch。与分割 MNIST 基准测试相比，我们在任务之间保持了 Adam 优化器的状态，从而获得了更好的结果。最终测试误差是在 MNIST 测试集上计算得出的。性能是通过网络解决所有任务的能力来衡量的。

为了建立比较的基准，我们首先对所有任务按顺序训练了一个没有突触巩固的网络（c = 0）。在这种情况下，系统表现出灾难性遗忘，即它学会了解决最近的任务，但严重地遗忘了之前的任务（图 4，蓝色线）。相比之下，当启用突触巩固时（采用 c>0 的取值），相同网络在 9 个附加任务上训练时仍然能够保持对任务 1 的高分类准确度（图 4）。

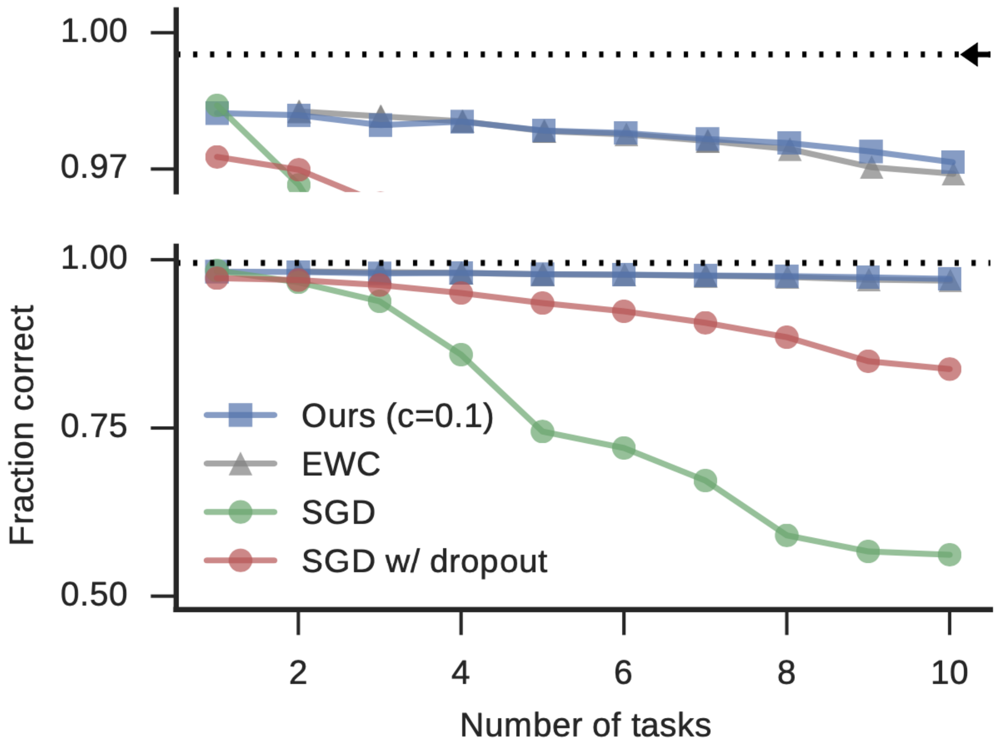

图 4. 在打乱的 MNIST 基准上，随着任务数量的增加，所有学习任务的平均分类准确度变化。我们的方法（蓝色）和 EWC（灰色，从 Kirkpatrick 等人的研究中提取并重新绘制(2017)）随着任务数量的增加，保持了高准确度。而随机梯度下降（绿色）和在隐藏层上使用 0.5 的 dropout 的随

机梯度下降（红色）的表现要差得多。顶部的面板是对图的上半部分的放大，其中虚线表示单个任务的初始训练准确度，黑色箭头表示同时训练所有任务时的相同网络的训练准确度。

此外，该网络学会了以高准确度解决所有其他任务，并且仅比同时训练所有数据的网络稍差一些（图 4）。最后，这些结果在训练和验证误差上是一致的，并且与使用 EWC（Kirkpatrick 等，2017）报告的结果相当。

为了更好地理解训练过程中的突触动态，我们可视化了不同任务μ之间的ω_k^μ的两两相关性（图 5b）。我们发现，没有巩固的情况下，第二隐藏层的ω_k^μ在任务之间是相关的，这可能是灾难性遗忘的原因。然而，在进行巩固时，这些减少损失的突触集在任务之间大多是不相关的，从而在更新权重以解决新任务时避免了干扰。

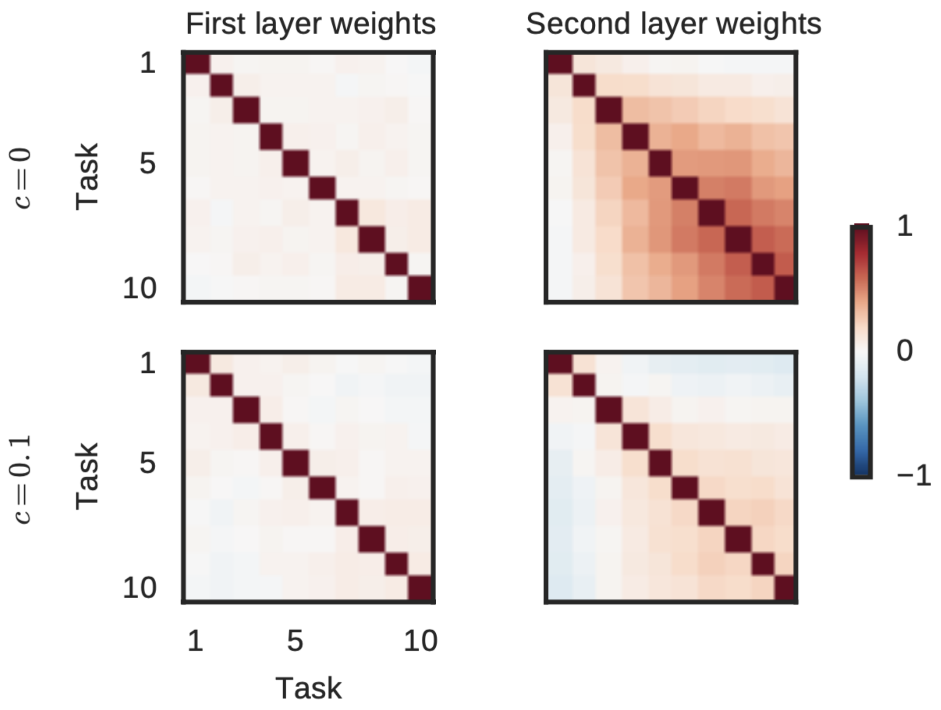

图 5. 打乱的 MNIST 上每个任务μ的权重重要性ω_k^μ的相关矩阵。比较正常的微调（$c = 0$，顶部）和巩固（$c = 0.1$，底部），由于打乱的 MNIST 数据集在输入层是不相关的，第一层权重重要性（左侧）在任务之间是不相关的。然而，随着更多的任务通过微调被学习，第二层的权重重要性（右侧）变得更加相关。相反，巩固可以防止ω_k^μ之间的强相关性，这与使用不同权重解决新任务的概念是一致的。

5.3. CIFAR-10/CIFAR-100 数据集的基准测试

为了评估突触巩固是否也能在更复杂的数据集和更大的模型中防止灾难性遗忘，我们进行了一项基于 CIFAR-10 和 CIFAR-100 的持续学习任务实验。具体来说，我们训练了一个卷积神经网络（包含 4 个卷积层，后跟 2 个带有 dropout 的全连接层；详见附录）。我们使用了与 split MNIST 相同的多元设置，使用 Adam 优化器（$\eta = 1 \times 10^{-3}$，$\beta 1 = 0.9$，$\beta 2 = 0.999$，$minibatch$大小为 256。首先，我们在整个 CIFAR-10 数据集上训练网络 60 个周期（任务 1），然后依次在 5 个附加任务上训练，每个任务

对应 CIFAR-100 数据集中的 10 个连续类别（图 6)。为了确定最佳c值，我们在参数范围$1\times10^{-3}<c<0.1$内进行了此实验。任务之间，优化器的状态会被重置。此外，我们还获得了两个特定控制案例的值。一方面，我们连续在所有任务上训练相同的网络（$c=0$）。另一方面，我们从每个任务上单独训练相同的网络，以评估跨任务的泛化性。最后，为了评估准确性的统计波动幅度，所有运行都重复了$n=5$次。

我们发现，在所有任务上训练后，具有巩固的网络在所有任务上的验证准确度相似，而没有巩固的网络准确度会随着任务量的增长而明显下降，其中旧任务的准确度较低（图 6)。

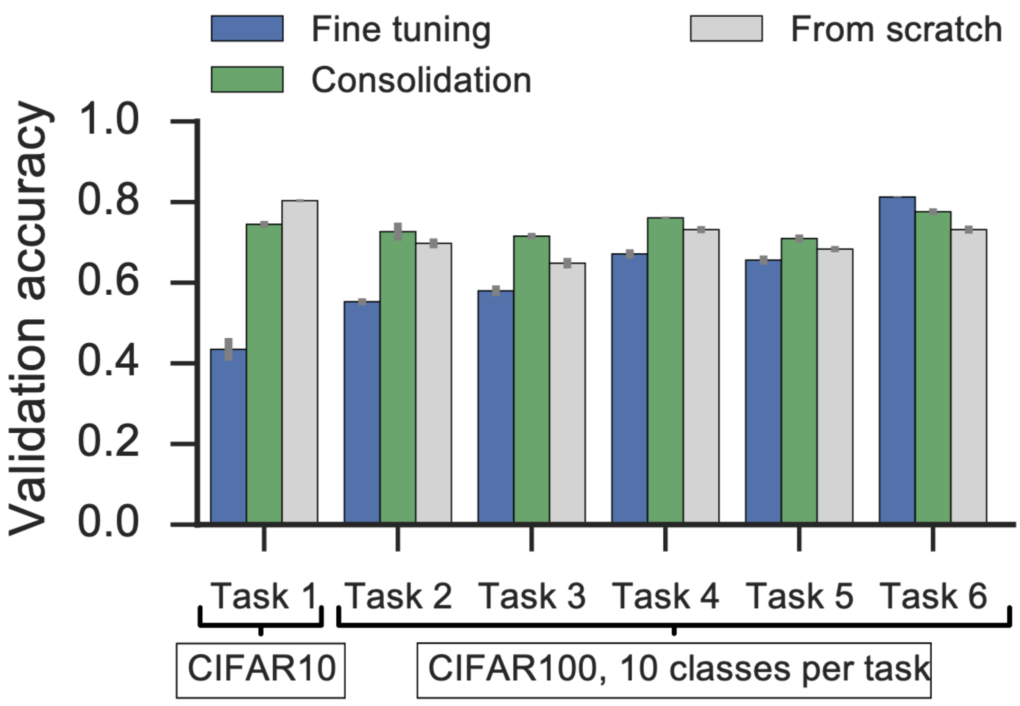

图 6. 分割 CIFAR-10/100 基准上的验证准确度。蓝色：没有巩固（c=0）的验证误差。绿色：有巩固（c=0.1）的验证误差。灰色：没有巩固的网络仅在单个任务上从头开始训练。该基准中的机会水平为 0.1。误差条对应标准差（n=5)。

重要的是，除了最后一个任务以外，具有巩固的网络的表现始终优于没有巩固的网络。最后，当比较在所有任务上训练的有巩固网络与仅在单个任务上从头开始训练的网络的表现时（图 6，绿色与灰色），前者要么显著优于后者，要么获得了相同的验证准确度，而这一趋势在训练准确度上发生了逆转。这表明没有巩固的网络更容易出现过拟合。唯一的例外是 CIFAR-10 的第一项任务，这可能是由于每个类别的样本数量增加了 10 倍。总之，我们发现巩固不仅能够保护旧的记忆免于随着时间的推移被慢慢遗忘，而且还能使网络在有限数据的新任务上更好地泛化。

6. 讨论

我们已经表明，在持续学习场景中经常遇到的灾难性遗忘问题可以通过采用单个

突触估计其对解决过去任务的重要性来缓解。然后，通过对最重要的突触的更改进行惩罚，可以最小程度地干扰以前学过的任务，同时学习新的任务。

正则化惩罚大多与 Kirkpatrick 等人最近引入的 EWC（2017）类似。然而，我们的方法是在线计算每个突触的巩固强度，并在整个参数空间的学习轨迹上计算，而 EWC 的突触重要性是在指定任务的损失最小值处的 Fisher 信息离线计算。尽管存在这种差异，但在这两种方法在打乱的 MNIST 基准上取得了相似的性能，这可能是由于两种不同重要性度量之间的相关性。

我们的方法要求单个突触不仅仅是对应于单个标量突触权重，而是作为各自独立的高维动态系统。这种高维状态使我们的每个突触能够在训练过程中智能地积累任务相关信息，并保留先前参数值的记忆。尽管我们不声称生物突触的行为与我们模型的智能突触相似，但神经生物学中的大量实验数据表明，生物突触的行为要比当前机器学习模型中占主导地位的人工标量突触复杂得多。本质上，突触是否发生变化，以及它们是否变得持久或最终衰退，可以由许多不同的生物学因素控制。例如，突触可塑性的诱导可能取决于单个突触的历史和状态（Montgomery & Madison，2002）。此外，最近的突触变化可能在几小时内消退，除非释放与可塑性相关的特定化学因素。这些化学因素被认为编码了最近变化的价值或新颖性（Redondo & Morris，2011）。最后，最近的突触变化可以通过刻板的神经活动来重置，而较旧的突触记忆变得越来越不受改变的影响（Zhou et al.，2003）。

在这里，我们介绍了一种特定的高维突触模型来解决一个特定问题：持续学习中的灾难性遗忘。这为这方面研究指出了新的方向，其中我们可以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动态。本质上，在机器学习中，除了增加网络的深度外，我们可能还需要给我们的突触增加智能。

致谢

作者感谢 Subhaneil Lahiri 的指导。FZ 得到瑞士国家科学基金会(SNSF)和威康信托基金会(Wellcome Trust)的支持。BP 获得斯坦福 MBC IGERT 奖学金和斯坦福跨学科研究生奖学金(Stanford Interdisciplinary Graduate Fellowship)的支持。SG 获得 Burroughs Wellcome、McKnight、Simons 和 James S. McDonnell 基金会以及海军研究办公室(Office of Naval Research)的支持。

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

Continual Learning Through Synaptic Intelligence

Friedemann Zenke [*] [1] **Ben Poole** [*] [1] **Surya Ganguli** [1]

Abstract

While deep learning has led to remarkable advances across diverse applications, it struggles in domains where the data distribution changes over the course of learning. In stark contrast, biological neural networks continually adapt to changing domains, possibly by leveraging complex molecular machinery to solve many tasks simultaneously. In this study, we introduce *intelligent synapses* that bring some of this biological complexity into artificial neural networks. Each synapse accumulates task relevant information over time, and exploits this information to rapidly store new memories without forgetting old ones. We evaluate our approach on continual learning of classification tasks, and show that it dramatically reduces forgetting while maintaining computational efficiency.

1. Introduction

Artificial neural networks (ANNs) have become an indispensable asset for applied machine learning, rivaling human performance in a variety of domain-specific tasks (LeCun et al., 2015). Although originally inspired by biology (Rosenblatt, 1958; Fukushima & Miyake, 1982), the underlying design principles and learning methods differ substantially from biological neural networks. For instance, parameters of ANNs are learned on a dataset in the training phase, and then frozen and used statically on new data in the deployment or recall phase. To accommodate changes in the data distribution, ANNs typically have to be retrained on the entire dataset to avoid overfitting and catastrophic forgetting (Choy et al., 2006; Goodfellow et al., 2013).

On the other hand, biological neural networks exhibit *continual learning* in which they acquire new knowledge over a lifetime. It is therefore difficult to draw a clear line between a learning and recall phase. Somehow, our brains have evolved to learn from non-stationary data and to update internal memories or beliefs on-the-fly. While it is unknown how this feat is accomplished in the brain, it seems possible that the unparalleled biological performance in continual learning could rely on specific features implemented by the underlying biological wetware that are not currently implemented in ANNs.

Perhaps one of the greatest gaps in the design of modern ANNs versus biological neural networks lies in the complexity of synapses. In ANNs, individual synapses (weights) are typically described by a single scalar quantity. On the other hand, individual biological synapses make use of complex molecular machinery that can affect plasticity at different spatial and temporal scales (Redondo & Morris, 2011). While this complexity has been surmised to serve memory consolidation (Fusi et al., 2005; Lahiri & Ganguli, 2013; Zenke et al., 2015; Ziegler et al., 2015; Benna & Fusi, 2016), few studies have illustrated how it benefits learning in ANNs.

Here we study the role of internal synaptic dynamics to enable ANNs to learn sequences of classification tasks. While simple, scalar one-dimensional synapses suffer from catastrophic forgetting, in which the network forgets previously learned tasks when trained on a novel task, this problem can be largely alleviated by synapses with a more complex three-dimensional state space. In our model, the synaptic state tracks the past and current parameter value, and maintains an online estimate of the synapse's "importance" toward solving problems encountered in the past. Our importance measure can be computed efficiently and locally at each synapse during training, and represents the local contribution of each synapse to the change in the global loss. When the task changes, we consolidate the important synapses by preventing them from changing in future tasks. Thus learning in future tasks is mediated primarily by synapses that were unimportant for past tasks, thereby avoiding catastrophic forgetting of these past tasks.

2. Prior work

The problem of alleviating catastrophic forgetting has been addressed in many previous studies. These studies can be

[*]Equal contribution [1]Stanford University. Correspondence to: Friedemann Zenke <fzenke@stanford.edu>, Ben Poole <poole@cs.stanford.edu>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

broadly partitioned into (1) architectural, (2) functional, and (3) structural approaches.

Architectural approaches to catastrophic forgetting alter the architecture of the network to reduce interference between tasks without altering the objective function. The simplest form of architectural regularization is freezing certain weights in the network so that they stay exactly the same (Razavian et al., 2014). A slightly more relaxed approach reduces the learning rate for layers shared with the original task while fine-tuning to avoid dramatic changes in the parameters (Donahue et al., 2014; Yosinski et al., 2014). Approaches using different nonlinearities like ReLU, MaxOut, and local winner-take-all have been shown to improve performance on permuted MNIST and sentiment analysis tasks (Srivastava et al., 2013; Goodfellow et al., 2013). Moreover, injecting noise to sparsify gradients using dropout also improves performance (Goodfellow et al., 2013). Recent work from Rusu et al. (2016) proposed more dramatic architectural changes where the entire network for the previous task is copied and augmented with new features while solving a new task. This entirely prevents forgetting on earlier tasks, but causes the architectural complexity to grow with the number of tasks.

Functional approaches to catastrophic forgetting add a regularization term to the objective that penalizes changes in the input-output function of the neural network. In Li & Hoiem (2016), the predictions of the previous task's network and the current network are encouraged to be similar when applied to data from the new task by using a form of knowledge distillation (Hinton et al., 2014). Similarly, Jung et al. (2016) regularize the ℓ_2 distance between the final hidden activations instead of the knowledge distillation penalty. Both of these approaches to regularization aim to preserve aspects of the input-output mapping for the old task by storing or computing additional activations using the old task's parameters. This makes the functional approach to catastrophic forgetting computationally expensive as it requires computing a forward pass through the old task's network for every new data point.

The third technique, **structural** regularization, involves penalties on the parameters that encourage them to stay close to the parameters for the old task. Recently, Kirkpatrick et al. (2017) proposed elastic weight consolidation (EWC), a quadratic penalty on the difference between the parameters for the new and the old task. They used a diagonal weighting proportional to the diagonal of the Fisher information metric over the old parameters on the old task. Exactly computing the diagonal of the Fisher requires summing over all possible output labels and thus has complexity linear in the number of outputs. This limits the application of this approach to low-dimensional output spaces.

3. Synaptic framework

To tackle the problem of continual learning in neural networks, we sought to build a simple structural regularizer that could be computed online and implemented locally at each synapse. Specifically, we aim to endow each individual synapse with a local measure of "importance" in solving tasks the network has been trained on in the past. When training on a new task we penalize changes to important parameters to avoid old memories from being overwritten. To that end, we developed a class of algorithms which keep track of an *importance measure* ω_k^μ which reflects past credit for improvements of the task objective L_μ for task μ to individual synapses θ_k. For brevity we use the term "synapse" synonymously with the term "parameter", which includes weights between layers as well as biases.

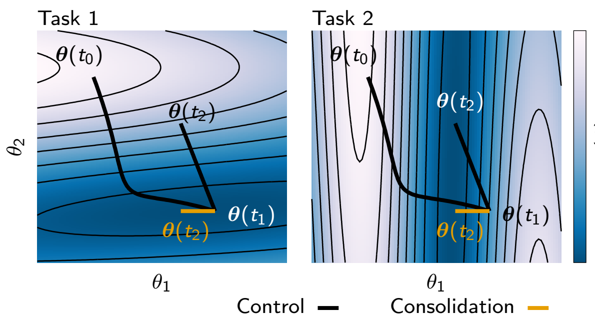

Figure 1. Schematic illustration of parameter space trajectories and catastrophic forgetting. Solid lines correspond to parameter trajectories during training. Left and right panels correspond to the different loss functions defined by different tasks (Task 1 and Task 2). The value of each loss function L_μ is shown as a heat map. Gradient descent learning on Task 1 induces a motion in parameter space from from $\boldsymbol{\theta}(t_0)$ to $\boldsymbol{\theta}(t_1)$. Subsequent gradient descent dynamics on Task 2 yields a motion in parameter space from $\boldsymbol{\theta}(t_1)$ to $\boldsymbol{\theta}(t_2)$. This final point minimizes the loss on Task 2 at the expense of significantly increasing the loss on Task 1, thereby leading to catastrophic forgetting of Task 1. However, there does exist an alternate point $\boldsymbol{\theta}(t_2)$, labelled in orange, that achieves a small loss for *both* tasks. In the following we show how to find this alternate point by determining that the component θ_2 was more important for solving Task 1 than θ_1 and then preventing θ_2 from changing much while solving Task 2. This leads to an online approach to avoiding catastrophic forgetting by consolidating changes in parameters that were important for solving past tasks, while allowing only the unimportant parameters to learn to solve future tasks.

The process of training a neural network is characterized by a trajectory $\boldsymbol{\theta}(t)$ in parameter space (Fig. 1). The feat of successful training lies in finding learning trajectories for which the endpoint lies close to a minimum of the loss function L on all tasks. Let us first consider the change in loss for an infinitesimal parameter update $\boldsymbol{\delta}(t)$ at time t.

In this case the change in loss is well approximated by the gradient $\boldsymbol{g} = \frac{\partial L}{\partial \boldsymbol{\theta}}$ and we can write

$$L(\boldsymbol{\theta}(t) + \boldsymbol{\delta}(t)) - L(\boldsymbol{\theta}(t)) \approx \sum_k g_k(t)\delta_k(t) \;, \qquad (1)$$

which illustrates that each parameter change $\delta_k(t) = \theta'_k(t)$ contributes the amount $g_k(t)\delta_k(t)$ to the change in total loss.

To compute the change in loss over an entire trajectory through parameter space we have to sum over all infinitesimal changes. This amounts to computing the path integral of the gradient vector field along the parameter trajectory from the initial point (at time t_0) to the final point (at time t_1):

$$\int_C \boldsymbol{g}(\boldsymbol{\theta}(t))d\boldsymbol{\theta} = \int_{t_0}^{t_1} \boldsymbol{g}(\boldsymbol{\theta}(t)) \cdot \boldsymbol{\theta}'(t)dt. \qquad (2)$$

As the gradient is a conservative field, the value of the integral is equal to the difference in loss between the end point and start point: $L(\boldsymbol{\theta}(t_1)) - L(\boldsymbol{\theta}(t_0))$. Crucial to our approach, we can decompose Eq. 2 as a sum over the individual parameters

$$\begin{aligned}\int_{t^{\mu-1}}^{t^\mu} \boldsymbol{g}(\boldsymbol{\theta}(t)) \cdot \boldsymbol{\theta}'(t)dt &= \sum_k \int_{t^{\mu-1}}^{t^\mu} g_k(\theta(t))\theta'_k(t)dt \\ &\equiv -\sum_k \omega_k^\mu. \qquad (3)\end{aligned}$$

The ω_k^μ now have an intuitive interpretation as the parameter specific contribution to changes in the total loss. Note that we have introduced the minus sign in the second line, because we are typically interested in decreasing the loss.

In practice, we can approximate ω_k^μ online as the running sum of the product of the gradient $g_k(t) = \frac{\partial L}{\partial \theta_k}$ with the parameter update $\theta'_k(t) = \frac{\partial \theta_k}{\partial t}$. For batch gradient descent with an infinitesimal learning rate, ω_k^μ can be directly interpreted as the per-parameter contribution to changes in the total loss. In most cases the true gradient is approximated by stochastic gradient descent (SGD), resulting in an approximation that introduces noise into the estimate of g_k. As a direct consequence, the approximated per-parameter importances will typically overestimate the true value of ω_k^μ.

How can the knowledge of ω_k^μ be exploited to improve continual learning? The problem we are trying to solve is to minimize the total loss function summed over all tasks, $\mathcal{L} = \sum_\mu L_\mu$, with the limitation that we do not have access to loss functions of tasks we were training on in the past. Instead, we only have access to the loss function L_μ for a single task μ at any given time. Catastrophic forgetting arises when minimizing L_μ inadvertently leads to substantial increases of the cost on previous tasks L_ν with $\nu < \mu$ (Fig. 1). To avoid catastrophic forgetting of all previous tasks ($\nu < \mu$) while training task μ, we want to avoid drastic changes to weights which were particularly influential in the past. The importance of a parameter θ_k for a single task is determined by two quantities: 1) how much an individual parameter contributed to a drop in the loss ω_k^ν over the entire trajectory of training (cf. Eq. 3) and 2) how far it moved $\Delta_k^\nu \equiv \theta_k(t^\nu) - \theta_k(t^{\nu-1})$. To avoid large changes to important parameters, we use a modified cost function $\tilde{L}_\mu$ in which we introduced a surrogate loss which approximates the summed loss functions of previous tasks L_ν ($\nu < \mu$). Specifically, we use a quadratic surrogate loss that has the same minimum as the cost function of the previous tasks and yields the same ω_k^ν over the parameter distance Δ_k. In other words, if learning were to be performed on the surrogate loss instead of the actual loss function, it would result in the same final parameters and change in loss during training (Fig. 2). For two tasks this is achieved exactly by the following quadratic surrogate loss

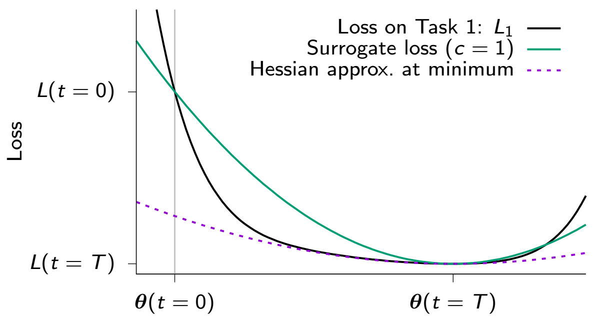

Figure 2. Schematic illustration of surrogate loss after learning one task. Consider some loss function defined by Task 1 (black). The quadratic surrogate loss (green) is chosen to precisely match 3 aspects of the descent dynamics on the original loss function: the total drop in the loss function $L(\boldsymbol{\theta}(0)) - L(\boldsymbol{\theta}(T))$, the total net motion in parameter space $\boldsymbol{\theta}(0) - \boldsymbol{\theta}(T)$, and achieving a minimum at the endpoint $\boldsymbol{\theta}(T)$. These 3 conditions uniquely determine the surrogate quadratic loss that summarizes the descent trajectory on the original loss. Note that this surrogate loss is different from a quadratic approximation defined by the Hessian at the minimum (purple dashed line).

$$\tilde{L}_\mu = L_\mu + \underbrace{c \sum_k \Omega_k^\mu \left(\tilde{\theta}_k - \theta_k\right)^2}_{\text{surrogate loss}} \qquad (4)$$

where we have introduced the dimensionless strength parameter c, the reference weight corresponding to the parameters at the end of the previous task $\tilde{\theta}_k = \theta_k(t^{\mu-1})$, and the per-parameter regularization strength:

$$\Omega_k^\mu = \sum_{\nu<\mu} \frac{\omega_k^\nu}{(\Delta_k^\nu)^2 + \xi} \quad . \tag{5}$$

Note that the term in the denominator $(\Delta_k^\nu)^2$ ensures that the regularization term carries the same units as the loss L. For practical reasons we also introduce an additional damping parameter, ξ, to bound the expression in cases where $\Delta_k^\nu \to 0$. Finally, c is a strength parameter which trades off old versus new memories. If the path integral (Eq. 3) is evaluated precisely, $c = 1$ would correspond to an equal weighting of old and new memories. However, due to noise in the evaluation of the path integral (Eq. 3), c typically has to be chosen smaller than one to compensate. Unless otherwise stated, the ω_k are updated continuously during training, whereas the cumulative importance measures, Ω_k^μ, and the reference weights, $\tilde{\boldsymbol{\theta}}$, are only updated at the end of each task. After updating the Ω_k^μ, the ω_k are set to zero. Although our motivation for Eq. 4 as a surrogate loss only holds in the case of two tasks, we will show empirically that our approach leads to good performance when learning additional tasks.

To understand how the particular choices of Eqs. 4 and 5 affect learning, let us consider the example illustrated in Figure 1 in which we learn two tasks. We first train on Task 1. At time t_1 the parameters have approached a local minimum of the Task 1 loss L_1. But, the same parameter configuration is not close to a minimum for Task 2. Consequently, when training on Task 2 without any additional precautions, the L_1 loss may inadvertently increase (Fig. 1, black trajectory). However, when θ_2 "remembers" that it was important to decreasing L_1, it can exploit this knowledge during training on Task 2 by staying close to its current value (Fig. 1, orange trajectory). While this will almost inevitably result in a decreased performance on Task 2, this decrease could be negligible, whereas the gain in performance on both tasks combined can be substantial.

The approach presented here is similar to EWC (Kirkpatrick et al., 2017) in that more influential parameters are pulled back more strongly towards a reference weight with which good performance was achieved on previous tasks. However, in contrast to EWC, here we are putting forward a method which computes an importance measure online and along the *entire learning trajectory*, whereas EWC relies on a point estimate of the diagonal of the Fisher information metric at the final parameter values, which has to be computed during a separate phase at the end of each task.

4. Theoretical analysis of special cases

In the following we illustrate that our general approach recovers sensible Ω_k^μ in the case of a simple and analytically tractable training scenario. To that end, we analyze what the parameter specific path integral ω_k^u and its normalized version Ω_k^μ (Eq. (5)), correspond to in terms of the geometry of a simple quadratic error function

$$E(\boldsymbol{\theta}) = \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T \boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{\theta}^*), \tag{6}$$

with a minimum at $\boldsymbol{\theta}^*$ and a Hessian matrix $\boldsymbol{H}$. Further consider batch gradient descent dynamics on this error function. In the limit of small discrete time learning rates, this descent dynamics is described by the continuous time differential equation

$$\tau \frac{d\boldsymbol{\theta}}{dt} = -\frac{\partial E}{\partial \boldsymbol{\theta}} = -\boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{\theta}^*), \tag{7}$$

where τ is related to the learning rate. If we start from an initial condition $\boldsymbol{\theta}(0)$ at time $t = 0$, an exact solution to the descent path is given by

$$\boldsymbol{\theta}(t) = \boldsymbol{\theta}^* + e^{-\boldsymbol{H}\frac{t}{\tau}}(\boldsymbol{\theta}(0) - \boldsymbol{\theta}^*), \tag{8}$$

yielding the time dependent update direction

$$\boldsymbol{\theta}'(t) = \frac{d\boldsymbol{\theta}}{dt} = -\frac{1}{\tau}\boldsymbol{H}e^{-\boldsymbol{H}\frac{t}{\tau}}(\boldsymbol{\theta}(0) - \boldsymbol{\theta}^*). \tag{9}$$

Now, under gradient descent dynamics, the gradient obeys $\boldsymbol{g} = \tau\frac{d\boldsymbol{\theta}}{dt}$, so the ω_k^μ in (3) are computed as the diagonal elements of the matrix

$$\boldsymbol{Q} = \tau \int_0^\infty dt \, \frac{d\boldsymbol{\theta}}{dt}\frac{d\boldsymbol{\theta}}{dt}^T \quad . \tag{10}$$

An explicit formula for $\boldsymbol{Q}$ can be given in terms of the eigenbasis of the Hessian $\boldsymbol{H}$. In particular, let λ^α and $\boldsymbol{u}^\alpha$ denote the eigenvalues and eigenvectors of $\boldsymbol{H}$, and let $d^\alpha = \boldsymbol{u}^\alpha \cdot (\boldsymbol{\theta}(0) - \boldsymbol{\theta}^*)$ be the projection of the discrepancy between initial and final parameters onto the α'th eigenvector. Then inserting (9) into (10), performing the change of basis to the eigenmodes of $\boldsymbol{H}$, and doing the integral yields

$$\boldsymbol{Q}_{ij} = \sum_{\alpha\beta} \boldsymbol{u}_i^\alpha d^\alpha \frac{\lambda^\alpha \lambda^\beta}{\lambda^\alpha + \lambda^\beta} d^\beta \boldsymbol{u}_j^\beta. \tag{11}$$

Note that as a time-integrated steady state quantity, $\boldsymbol{Q}$ no longer depends on the time constant τ governing the speed of the descent path.

At first glance, the $\boldsymbol{Q}$ matrix elements depend in a complex manner on both the eigenvectors and eigenvalues of the Hessian, as well as the initial condition $\boldsymbol{\theta}(0)$. To understand this dependence, let's first consider averaging $\boldsymbol{Q}$ over random initial conditions $\boldsymbol{\theta}(0)$, such that the collection of discrepancies d^α constitute a set of zero mean iid random variables with variance σ^2. Thus we have the average $\langle d^\alpha d^\beta \rangle = \sigma^2 \delta_{\alpha\beta}$. Performing this average over $\boldsymbol{Q}$ then yields

$$\langle \boldsymbol{Q}_{ij} \rangle = \frac{1}{2}\sigma^2 \sum_\alpha \boldsymbol{u}_i^\alpha \lambda^\alpha \boldsymbol{u}_j^\beta = \frac{1}{2}\sigma^2 \boldsymbol{H}_{ij}. \tag{12}$$

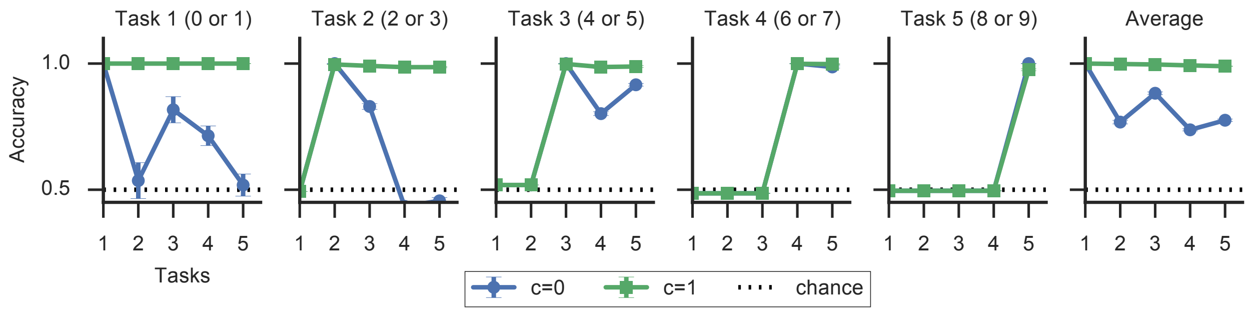

Figure 3. Mean classification accuracy for the split MNIST benchmark as a function of the number of tasks. The first five panels show classification accuracy on the five tasks consisting of two MNIST digits each as a function of number of consecutive tasks. The rightmost panel shows the average accuracy, which is computed as the average over task accuracies for past tasks ν with $\nu < \mu$ where μ is given by the number of tasks on the x-axis. Note that in this setup with multiple binary readout heads, an accuracy of 0.5 corresponds to chance level. Error bars correspond to SEM (n=10).

Thus remarkably, after averaging over initial conditions, the $\boldsymbol{Q}$ matrix, which is available simply by correlating parameter updates across pairs of synapses and integrating over time, reduces to the Hessian, up to a scale factor dictating the discrepancy between initial and final conditions. Indeed, this scale factor theoretically motivates the normalization in (5); the denominator in (5), at zero damping, ξ averages to σ^2, thereby removing the scale factor σ^2 in (12)

However, we are interested in what $\boldsymbol{Q}_{ij}$ computes for a *single* initial condition. There are two scenarios in which the simple relationship between $\boldsymbol{Q}$ and the Hessian $\boldsymbol{H}$ is preserved *without* averaging over initial conditions. First, consider the case when the Hessian is diagonal, so that $\boldsymbol{u}_i^\alpha = \delta_{\alpha i}\boldsymbol{e}_i$ where $\boldsymbol{e}_i$ is the i'th coordinate vector. Then α and i indices are interchangeable and the eigenvalues of the Hessian are the diagonal elements of the Hessian: $\lambda^i = \boldsymbol{H}_{ii}$. Then (11) reduces to

$$\boldsymbol{Q}_{ij} = \delta_{ij}(d^i)^2\boldsymbol{H}_{ii}. \tag{13}$$

Again the normalization in (5), at zero damping, removes the scale of movement in parameter space $(d^i)^2$, and so the normalized $\boldsymbol{Q}$ matrix becomes identical to the diagonal Hessian. In the second scenario, consider the extreme limit where the Hessian is rank 1 so that λ^1 is the only nonzero eigenvalue. Then (11) reduces to

$$\boldsymbol{Q}_{ij} = \frac{1}{2}(d^1)^2\boldsymbol{u}_i^1\lambda_1\boldsymbol{u}_j^1 = \frac{1}{2}(d^1)^2\boldsymbol{H}_{ij}. \tag{14}$$

Thus again, the $\boldsymbol{Q}$ matrix reduces to the Hessian, up to a scale factor. The normalized importances then become the diagonal elements of the non-diagonal but low rank Hessian. We note that the low rank Hessian is the interesting case for continual learning; low rank structure in the error function leaves many directions in synaptic weight space unconstrained by a given task, leaving open excess capacity for synaptic modification to solve future tasks without interfering with performance on an old task.

It is important to stress that the path integral for importance is computed by integrating information along the *entire* learning trajectory (cf. Fig. 2). For a quadratic loss function, the Hessian is constant along this trajectory, and so we find a precise relationship between the importance and the Hessian. But for more general loss functions, where the Hessian varies along the trajectory, we cannot expect any simple mathematical correspondence between the importance Ω_k^μ and the Hessian at the endpoint of learning, or related measures of parameter sensitivity (Pascanu & Bengio, 2013; Martens, 2016; Kirkpatrick et al., 2017) at the endpoint. In practice, however, we find that our importance measure is correlated to measures based on such endpoint estimates, which may explain their comparable effectiveness as we will see in the next section.

5. Experiments

We evaluated our approach for continual learning on the split and permuted MNIST (LeCun et al., 1998; Goodfellow et al., 2013), and split versions of CIFAR-10 and CIFAR-100 (Krizhevsky & Hinton, 2009).

5.1. Split MNIST

We first evaluated our algorithm on a split MNIST benchmark. For this benchmark we split the full MNIST training data set into 5 subsets of consecutive digits. The 5 tasks correspond to learning to distinguish between two consecutive digits from 0 to 10. We used a small multi-layer perceptron (MLP) with only two hidden layers consisting of 256 units each with ReLU nonlinearities, and a standard categorical cross-entropy loss function plus our consolidation cost term (with damping parameter $\xi = 1 \times 10^{-3}$). To avoid the complication of crosstalk between digits at the readout layer due to changes in the label distribution during training, we used a multi-head approach in which the categorical cross entropy loss at the readout layer was computed only for the digits present in the current task. Finally, we optimized our network using a minibatch size of 64 and trained for 10 epochs. To achieve good absolute performance with a smaller number of epochs we used the adaptive optimizer Adam (Kingma & Ba, 2014) ($\eta = 1 \times 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$). In this benchmark the optimizer state was reset after training each task.

To evaluate the performance, we computed the average classification accuracy on all previous tasks as a function of number of tasks trained. We now compare this performance between networks in which we turn consolidation dynamics on ($c = 1$) against cases in which consolidation was off ($c = 0$). During training of the first task the consolidation penalty is zero for both cases because there is no past experience that synapses could be regularized against. When trained on the digits “2” and “3” (Task 2), both the model with and without consolidation show accuracies close to 1 on Task 2. However, on average the networks without synaptic consolidation show substantial loss in accuracy on Task 1 (Fig. 3). In contrast to that, networks with consolidation only undergo minor impairment with respect to accuracy on Task 1 and the average accuracy for both tasks stays close to 1. Similarly, when the network has seen all MNIST digits, on average, the accuracy on the first two tasks, corresponding to the first four digits, has dropped back to chance levels in the cases without consolidation whereas the model with consolidation only shows minor degradation in performance on these tasks (Fig. 3).

5.2. Permuted MNIST benchmark

In this benchmark, we randomly permute all MNIST pixels differently for each task. We trained a MLP with two hidden layers with 2000 ReLUs each and softmax loss. We used Adam with the same parameters as before. However, here we used $\xi = 0.1$ and the value for $c = 0.1$ was determined via a coarse grid search on a heldout validation set. The mini batch size was set to 256 and we trained for 20 epochs. In contrast to the split MNIST benchmark we obtained better results by maintaining the state of the Adam optimizer between tasks. The final test error was computed on data from the MNIST test set. Performance is measured by the ability of the network to solve all tasks.

To establish a baseline for comparison we first trained a network without synaptic consolidation ($c = 0$) on all tasks sequentially. In this scenario the system exhibits catastrophic forgetting, i.e. it learns to solve the most recent task, but

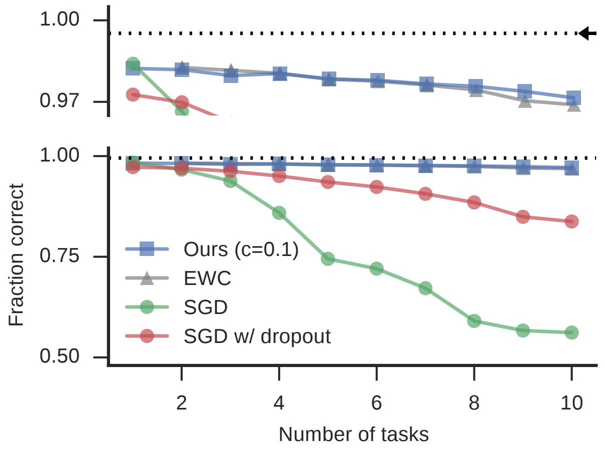

Figure 4. Average classification accuracy over all learned tasks from the permuted MNIST benchmark as a function of number of tasks. Our approach (blue) and EWC (gray, extracted and replotted from Kirkpatrick et al. (2017)) maintain high accuracy as the number of tasks increase. SGD (green) and SGD with dropout of 0.5 on the hidden layers (red) perform far worse. The top panel is a zoom-in on the upper part of the graph with the initial training accuracy on a single task (dotted line) and the training accuracy of the same network when trained on all tasks simultaneously (black arrow).

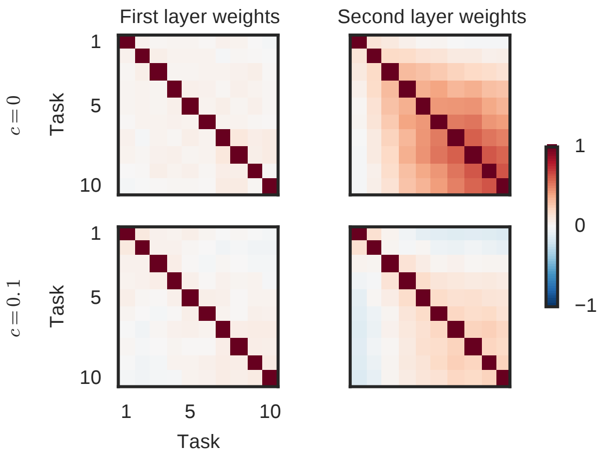

Figure 5. Correlation matrices of weight importances, ω_k^μ, for each task μ on permuted MNIST. For both normal fine-tuning ($c = 0$, top) and consolidation ($c = 0.1$, bottom), the first layer weight importances (left) are uncorrelated between tasks since the permuted MNIST datasets are uncorrelated at the input layer. However, the second layer importances (right) become more correlated as more tasks are learned with fine-tuning. In contrast, consolidation prevents strong correlations in the ω_k^μ, consistent with the notion of different weights being used to solve new tasks.

rapidly forgets about previous tasks (blue line, Fig. 4). In contrast to that, when enabling synaptic consolidation, with a sensible choice for $c > 0$, the same network retains high classification accuracy on Task 1 while being trained on 9 additional tasks (Fig. 4). Moreover, the network learns to solve all other tasks with high accuracy and performs only slightly worse than a network which had trained on all data simultaneously (Fig. 4). Finally, these results were consistent across training and validation error and comparable to the results reported with EWC (Kirkpatrick et al., 2017).

To gain a better understanding of the synaptic dynamics during training, we visualized the pairwise correlations of the ω_k^μ across the different tasks μ (Fig. 5b). We found that without consolidation, the ω_k^μ in the second hidden layer are correlated across tasks which is likely to be the cause of catastrophic forgetting. With consolidation, however, these sets of synapses contributing to decreasing the loss are largely uncorrelated across tasks, thus avoiding interference when updating weights to solve new tasks.

5.3. Split CIFAR-10/CIFAR-100 benchmark

To evaluate whether synaptic consolidation dynamics would also prevent catastrophic forgetting in more complex datasets and larger models, we experimented with a continual learning task based on CIFAR-10 and CIFAR-100. Specifically, we trained a CNN (4 convolutional, followed by 2 dense layers with dropout; see Appendix for details). We used the same multi-head setup as in the case of split MNIST using Adam ($\eta = 1 \times 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, minibatch size 256). First, we trained the network for 60 epochs on the full CIFAR-10 dataset (Task 1) and sequentially on 5 additional tasks each corresponding to 10 consecutive classes from the CIFAR-100 dataset (Fig. 6). To determine the best c, we performed this experiment for different values in the parameter range $1\times10^{-3} < c < 0.1$. Between tasks the state of the optimizer was reset. Moreover, we obtained values for two specific control cases. On the one hand we trained the same network with $c = 0$ on all tasks consecutively. On the other hand we trained the same network from scratch on each task individually to assess generalization across tasks. Finally, to assess the magnitude of statistical fluctuations in accuracy, all runs were repeated $n = 5$ times.

We found that after training on all tasks, networks with consolidation showed similar validation accuracy across all tasks, whereas accuracy in the network without consolidation showed a clear age dependent decline in which old tasks were solved with lower accuracy (Fig. 6). Importantly, the performance of networks trained with consolidation was always better than without consolidation, except on the last task. Finally, when comparing the performance of networks trained with consolidation on all tasks with net-

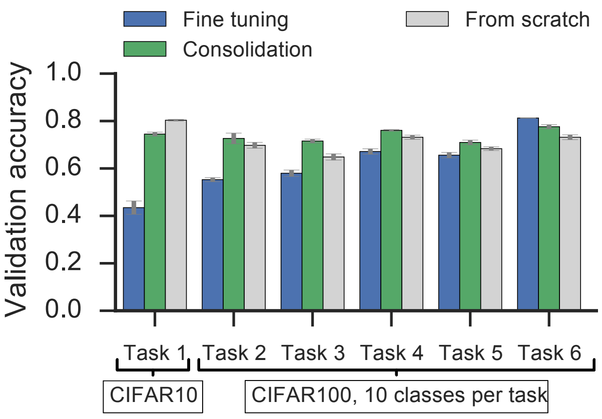

Figure 6. Validation accuracy on the split CIFAR-10/100 benchmark. Blue: Validation error, without consolidation ($c = 0$). Green: Validation error, with consolidation ($c = 0.1$). Gray: Network without consolidation trained from scratch on the single task only. Chance-level in this benchmark is 0.1. Error bars correspond to SD (n=5).

works trained from scratch only on a single task (Fig. 6; green vs gray), the former either significantly outperformed the latter or yielded the same validation accuracy, while this trend was reversed in training accuracy. This suggests that networks without consolidation are more prone to over fitting. The only exception to that rule was Task 1, CIFAR-10 which is presumably due to its $10\times$ larger number of examples per class. In summary, we found that consolidation not only protected old memories from being slowly forgotten over time, but also allowed networks to generalize better on new tasks with limited data.

6. Discussion

We have shown that the problem of catastrophic forgetting commonly encountered in continual learning scenarios can be alleviated by allowing individual synapses to estimate their importance for solving past tasks. Then by penalizing changes to the most important synapses, novel tasks can be learned with minimal interference to previously learned tasks.

The regularization penalty is similar to EWC as recently introduced by Kirkpatrick et al. (2017). However, our approach computes the per-synapse consolidation strength in an online fashion and over the entire learning trajectory in parameter space, whereas for EWC synaptic importance is computed offline as the Fisher information at the minimum of the loss for a designated task. Despite this difference, these two approaches yielded similar performance on the permuted MNIST benchmark which may be due to correlations between the two different importance measures.

Our approach requires individual synapses to not simply correspond to single scalar synaptic weights, but rather act as higher dimensional dynamical systems in their own right. Such higher dimensional state enables each of our synapses to intelligently accumulate task relevant information during training and retain a memory of previous parameter values. While we make no claim that biological synapses behave like the intelligent synapses of our model, a wealth of experimental data in neurobiology suggests that biological synapses act in much more complex ways than the artificial scalar synapses that dominate current machine learning models. In essence, whether synaptic changes occur, and whether they are made permanent, or left to ultimately decay, can be controlled by many different biological factors. For instance, the induction of synaptic plasticity may depend on the history and the synaptic state of individual synapses (Montgomery & Madison, 2002). Moreover, recent synaptic changes may decay on the timescale of hours unless specific plasticity related chemical factors are released. These chemical factors are thought to encode the valence or novelty of a recent change (Redondo & Morris, 2011). Finally, recent synaptic changes can be reset by stereotypical neural activity, whereas older synaptic memories become increasingly insensitive to reversal (Zhou et al., 2003).

Here, we introduced one specific higher dimensional synaptic model to tackle a specific problem: catastrophic forgetting in continual learning. However, this suggests new directions of research in which we mirror neurobiology to endow individual synapses with potentially complex dynamical properties, that can be exploited to intelligently control learning dynamics in neural networks. In essence, in machine learning, in addition to adding depth to our networks, we may need to add intelligence to our synapses.

Acknowledgements

The authors thank Subhaneil Lahiri for helpful discussions. FZ was supported by the SNSF (Swiss National Science Foundation) and the Wellcome Trust. BP was supported by a Stanford MBC IGERT Fellowship and Stanford Interdisciplinary Graduate Fellowship. SG was supported by the Burroughs Wellcome, McKnight, Simons and James S. McDonnell foundations and the Office of Naval Research.

References

Benna, Marcus K. and Fusi, Stefano. Computational principles of synaptic memory consolidation. *Nat Neurosci*, advance online publication, October 2016. ISSN 1097-6256. doi: 10.1038/nn.4401.

Choy, Min Chee, Srinivasan, Dipti, and Cheu, Ruey Long. Neural networks for continuous online learning and control. *IEEE Trans Neural Netw*, 17(6):1511–1531, November 2006. ISSN 1045-9227. doi: 10.1109/TNN.2006.881710.

Donahue, Jeff, Jia, Yangqing, Vinyals, Oriol, Hoffman, Judy, Zhang, Ning, Tzeng, Eric, and Darrell, Trevor. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference in Machine Learning (ICML)*, 2014.

Fukushima, Kunihiko and Miyake, Sei. Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition. In *Competition and Cooperation in Neural Nets*, pp. 267–285. Springer, Berlin, Heidelberg, 1982. DOI: 10.1007/978-3-642-46466-9_18.

Fusi, Stefano, Drew, Patrick J., and Abbott, Larry F. Cascade models of synaptically stored memories. *Neuron*, 45(4):599–611, February 2005. ISSN 0896-6273. doi: 10.1016/j.neuron.2005.02.001.

Goodfellow, Ian J., Mirza, Mehdi, Xiao, Da, Courville, Aaron, and Bengio, Yoshua. An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. *arXiv:1312.6211 [cs, stat]*, December 2013. arXiv: 1312.6211.

Hinton, Geoffrey, Vinyals, Oriol, and Dean, Jeff. Distilling the knowledge in a neural network. *NIPS Deep Learning and Representation Learning Workshop*, 2014.

Jung, Heechul, Ju, Jeongwoo, Jung, Minju, and Kim, Junmo. Less-forgetting Learning in Deep Neural Networks. *arXiv:1607.00122 [cs]*, July 2016. arXiv: 1607.00122.

Kingma, Diederik and Ba, Jimmy. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, December 2014. arXiv: 1412.6980.

Kirkpatrick, James, Pascanu, Razvan, Rabinowitz, Neil, Veness, Joel, Desjardins, Guillaume, Rusu, Andrei A., Milan, Kieran, Quan, John, Ramalho, Tiago, Grabska-Barwinska, Agnieszka, Hassabis, Demis, Clopath, Claudia, Kumaran, Dharshan, and Hadsell, Raia. Overcoming catastrophic forgetting in neural networks. *PNAS*, pp. 201611835, March 2017. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1611835114.

Krizhevsky, Alex and Hinton, Geoffrey. Learning multiple layers of features from tiny images. 2009.

Lahiri, Subhaneil and Ganguli, Surya. A memory frontier for complex synapses. In *Advances in Neural Information Processing Systems*, volume 26, pp. 1034–1042, Tahoe, USA, 2013. Curran Associates, Inc.

LeCun, Yann, Cortes, Corinna, and Burges, Christopher JC. *The MNIST database of handwritten digits*. 1998.

LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. Deep learning. *Nature*, 521(7553):436–444, May 2015. ISSN 0028-0836. doi: 10.1038/nature14539.

Li, Zhizhong and Hoiem, Derek. Learning without forgetting. In *European Conference on Computer Vision*, pp. 614–629. Springer, 2016.

Martens, James. *Second-order optimization for neural networks*. PhD thesis, University of Toronto, 2016.

Montgomery, Johanna M. and Madison, Daniel V. State-Dependent Heterogeneity in Synaptic Depression between Pyramidal Cell Pairs. *Neuron*, 33(5):765–777, February 2002. ISSN 0896-6273. doi: 10.1016/S0896-6273(02)00606-2.

Pascanu, Razvan and Bengio, Yoshua. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

Razavian, Ali Sharif, Azizpour, Hossein, Sullivan, Josephine, and Carlsson, Stefan. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 806–813, 2014.

Redondo, Roger L. and Morris, Richard G. M. Making memories last: the synaptic tagging and capture hypothesis. *Nat Rev Neurosci*, 12(1):17–30, January 2011. ISSN 1471-003X. doi: 10.1038/nrn2963.

Rosenblatt, Frank. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Rusu, Andrei A., Rabinowitz, Neil C., Desjardins, Guillaume, Soyer, Hubert, Kirkpatrick, James, Kavukcuoglu, Koray, Pascanu, Razvan, and Hadsell, Raia. Progressive Neural Networks. *arXiv:1606.04671 [cs]*, June 2016. arXiv: 1606.04671.

Srivastava, Rupesh K, Masci, Jonathan, Kazerounian, Sohrob, Gomez, Faustino, and Schmidhuber, Juergen. Compete to Compute. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 2310–2318. Curran Associates, Inc., 2013.

Yosinski, Jason, Clune, Jeff, Bengio, Yoshua, and Lipson, Hod. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pp. 3320–3328, 2014.

Zenke, Friedemann, Agnes, Everton J., and Gerstner, Wulfram. Diverse synaptic plasticity mechanisms orchestrated to form and retrieve memories in spiking neural networks. *Nat Commun*, 6, April 2015. doi: doi: 10.1038/ncomms7922.

Zhou, Qiang, Tao, Huizhong W., and Poo, Mu-Ming. Reversal and Stabilization of Synaptic Modifications in a Developing Visual System. *Science*, 300(5627):1953–1957, June 2003. doi: 10.1126/science.1082212.

Ziegler, Lorric, Zenke, Friedemann, Kastner, David B., and Gerstner, Wulfram. Synaptic Consolidation: From Synapses to Behavioral Modeling. *J Neurosci*, 35(3): 1319–1334, January 2015. ISSN 0270-6474, 1529-2401. doi: 10.1523/JNEUROSCI.3989-14.2015.