

“博采众长”——基于原图 RGB 增强、卷积特征提取和多模型集成三种 ViT 与 CNN 结合思路研究

王跃林, 3020001251

摘要

本研究致力于深入探索计算机视觉领域中的深度学习模型的潜力，特别强调视觉变换器（ViT）和卷积神经网络（CNN）的融合应用。我们的研究目标是全方位的：首先，通过原图 RGB 增强技术，我们旨在提升模型对原始图像的识别和处理效率；接着，利用高效的卷积特征提取方法，我们加强模型对图像中局部细节与全局信息的捕捉及其综合处理能力；最终，采用多模型集成学习策略，我们致力于提高整体模型的准确度和鲁棒性。在我们最新推出的图像增强 V2 版本中，通过优化算法和高效设计，我们显著加快了模型的收敛速度；利用先进的卷积神经网络特征提取技术，不仅在加速收敛的基础上进一步提高了模型的准确性，还大幅度提升了处理效率；我们采用了创新的集成学习方法，这一方法巧妙地融合了多种算法的优势，从而实现了自动的最优模型选择。经过严格的测试和验证，我们的方法在知名的 CIFAR10 数据集上展示了卓越的性能表现和显著的成功。这不仅证明了我们技术的先进性，也为未来在图像处理领域的应用开辟了新的可能性。

关键词— ViT, CNN, 图像增强, 集成学习

1. 引言

课题意义

在计算机视觉领域，卷积神经网络（CNN）与 VisionTransformer（ViT）的融合代表了一种创新的模型发展方向。CNN 以其在图像处理中的局部感知能力和平移不变性而著称，这使其能够高效地捕捉图像中的细节和局部特征。相比之下，ViT 通过其先进的自注意力机制，能够理解和利用图像的全局上下文信息。将这两种技术结合起来，我们可以构建出一个更为强大的模型，它不仅综合了局部特征的细致识别能力和全局信息的广泛捕捉能力，还能在各种数据规模上表现出色。具体来说，虽然 ViT 在处理大规模数据集时展现出卓越的性能，但其在小型数据集上的效果往往不如 CNN 类模型。这种结合方式能显著提升模型在各种数据集上的泛化能力，从而使模型在不同的应用环境下都能保持一致

的高性能。此外，这种结合还有助于我们更深入地理解 ViT 和 CNN 各自的优势和局限性，以及它们如何相互补充。这不仅提高了模型的鲁棒性，也增强了模型的可解释性，这对于需要高度安全性和可靠性的应用场景尤为重要。

最新研究进展

在计算机视觉领域的最新研究中，卷积神经网络（CNN）与视觉变换器（VisionTransformer, ViT）的融合策略受到了广泛关注。研究表明，结合 CNN 和 ViT 可以有效利用它们各自的优势，从而实现更优越的性能表现。具体来说，一些创新的研究设计了新型的模型架构，在这些架构中，CNN 负责提取图像的非连续像素局部特征，而这些局部特征随后被传递到 ViT 模块中，以便捕捉和整合全局信息。这种整合方法在不同规模的数据集上均展现出卓越的性能[1]。

此外，还有研究专注于使 ViT 在大规模数据集上高效工作，运用了多种图像增强技术来实现这一目标[2]。另一些研究则采用模型蒸馏方法，首先在广受认可的大型数据集上进行训练，随后通过微调技术使模型适应现实世界中普遍存在的小规模数据集[3]。

同时，也有研究者关注 ViT 在实际应用中的性能，提出了面向移动设备的 ViT 模型——MobileViT，并围绕 MobileViT 进行了一系列的优化和改进，旨在提高其在实际移动环境中的适用性和效率。这些研究不仅推动了计算机视觉技术的进步，也为该领域提供了新的研究方向和实用策略[4]。

当前存在的问题

专家们正集中力量探索提升视觉变换器（VisionTransformer, ViT）在小型数据集上的性能。这涉及到多种手段，如对原始图像进行增强处理，以及对网络架构进行细致的调参等，但现有研究在同时融合这两种方法方面尚有待完善。此外，面对这类深度学习问题时，大多研究集中在网络参数和架构的优化上，而较少考虑到应用机器学习算法思想的优化策略。

同时，当前研究的主要趋势是集中在通过串行改进手段提升模型效能，以及探索用于统一模型的并行

策略。然而，目前尚缺乏将串行和并行方法有效结合的研究方法。这表明在计算机视觉领域，尤其是 ViT 的应用和优化上，仍有巨大的探索空间和提升潜力。

课题挑战的贡献

我们的研究致力于通过高效的训练策略和先进技术，全方位提升视觉变换器（VisionTransformer，ViT）在小型数据集上的数据处理效率。我们的工作重点在于多角度支持 ViT 克服性能限制，这一方面填补了同时运用图像增强和网络架构优化的研究空白，另一方面也强调了机器学习算法思想在优化深度学习问题中的重要作用。总体而言，我们的研究目标是应对当前在整合图像增强和网络架构优化方面的挑战，并推动更全面的深度学习优化策略。

在最后，我们的项目引入了新的图像增强技术，并与卷积神经网络（CNN）的特征提取技术相结合，使用集成学习方法进一步优化 ViT 模型。最终，通过多个分类器的投票系统，我们成功自动进行了最优模型的选择，这些成果为当前领域提供了创新性和综合性的解决方案。

主要技巧

我们的核心技术涵盖了原始图像增强、基于不同卷积特征提取的 ViT 训练辅助，以及采用多模型组合进行集成学习的方法。在图像增强技术方面，我们采用了一种创新方法：将原始的输入图像传递到每一个层中；此外，每次卷积的结果不仅输入到下一个核，还输入到与其非直接相连的核，以此增强层与层之间的联系。我们还对 ViT 拆解，拆解后包括 embedding 层、transform 层和 fc 层。在生成 patch 的 embedding 层，我们引入了一些经典的卷积网络，通过卷积提取原始图像的特征，以更好地帮助 ViT 掌握图像信息。此外，我们还融合了刚才提到的基于输入图像的图像增强技术，以统一局部信息和全局信息的特征提取，使得模型能够更全面地理解图像内容，从而实现更准确的图像分类。

最后，我们基于不同模型的独特学习策略，建立了集成学习算法，进一步提高了最终决策的正确性。该算法整合了多个模型的预测结果，并通过多分类器投票评选的方式，为最终决策提供了更为可靠的依据。这项研究不仅引入了新型图像增强技术，还结合了 CNN 卷积特征提取和集成学习方法，为该领域的发展贡献了创新和综合性的解决方案。总体而言，我们致力于解决当前研究中整合图像增强和网络架构优化方面的挑战，以推动更全面的深度学习问题优化方法。

在本节的最后一段中，我将对本技术报告的主要贡献总结如下：

- 1) 本报告提出了一种类残差连接的原始数据增强的方法，同时通过卷积加强了神经网络层与层之间的联系；
- 2) 将多种 CNN 架构作为特征提取器和 ViT 结合起来，得到了 ViT 和 CNN 相结合的模型；
- 3) 考虑了基于集成学习思想的多模型共同参与投票决策的新类 inception 模型架构。

2. 研究方法

我们的研究方法分为两个主要部分：原始图像增强 CNN 模型，串行和并行的模型集成。

在原始图像增强部分，传统的卷积网络通常将输入图像仅传递到第一层，然后逐层向前传递每层的输出。在这种方法中，原始输入图像在每一层都重新引入。这意味着不仅每个卷积层的输出传递到下一个层，原始图像也会被直接传递给每个后续层。通过在每一层中重新引入原始图像，网络能够更好地保持和利用原始数据中的信息。通过将层的输出传递给非直接相连的层，增强了不同层间的联系，可能有助于捕捉更复杂的特征。此网络不仅将每层的输出传递到下一个直接相连的卷积层，还传递给其他非直接相连的层。通过在每一层中重新引入原始图像，网络能够更好地保持和利用原始数据中的信息；通过将层的输出传递给非直接相连的层，增强了不同层间的联系，可能有助于捕捉更复杂的特征。

在串行部分，我们首先关注于如何通过不同深度和尺寸的卷积核，对图像进行有效的 patch 特征提取。通过使用不同深度和尺寸的卷积核，CNN 部分可以高效地提取图像的局部特征。这在处理图像时非常重要，因为局部特征（如边缘、角点和纹理）对于理解图像内容至关重要。这些特征随后被输入到 ViT 模型中，ViT 模型擅长处理图像的全局信息，可以捕捉图像不同区域之间的关系。通过将 CNN 提取的特征输入到 ViT，模型能够综合利用局部特征和全局上下文，从而更全面地理解图像。此外，我们还通过叠加 transformer 层来加强模型对图像的局部细节和全局信息的综合理解。这种串行结合的方法旨在发挥 CNN 在局部特征提取方面的优势，以及 ViT 在处理全局信息方面的高效性。叠加的 transformer 层增强了模型对图像的局部细节和全局信息的综合理解。这意味着模型不仅能识别图像中的细节特征，还能理解这些特征在整个图像中的作用和意义。

在执行并行处理的部分，鉴于卷积神经网络（CNN）系列和视觉变换器（ViT）系列在图像分类的基本原理上存在明显差异，我们采纳了一种基于集成学习的策略。具体而言，我们将多个模型以并行方式组合

使用，使得每个模型都可以独立地处理输入数据。随后，这些模型的输出被集成和综合，以形成最终的分类决策。这种方法的显著优点在于，它能够最大限度地利用不同模型的独特特性，从而显著增强整个系统的适应性和鲁棒性。这样的设计不仅提高了分类的正确性，还增加了模型对新颖或复杂数据的处理能力，确保在各种情况下都能保持高效和可靠的性能。

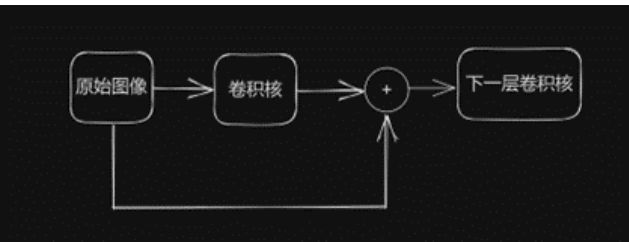
通过这种串行和并行的混合方法，我们旨在创造一个更加精准、高效且适应性强的计算机视觉系统。这不仅可以提高模型对复杂图像数据的处理能力，还能作为计算机视觉领域提供新的研究视角和解决方案。

研究思路

原图 RGB 增强

在传统的前馈神经网络设计中，网络的每一层主要依赖于其前一层传递过来的特征。这种结构虽然简单，但却限制了网络对信息的处理能力，因为每一层只能接触到其前一层提取和传递的特征。这种方式未能充分挖掘和利用每个卷积层的潜在优势，导致了信息处理的局限性。

为了解决这个问题，可以借鉴 ResNet-（残差网络）的设计理念。ResNet-通过引入所谓的“残差连接”，允许网络的较深层直接访问之前层的原始数据，从而避免了信息在传递过程中的丢失。受此启发，我们可以进一步发展这一概念：在保持每层卷积结果传递给下一层的同时，还将原始图像直接输入到每个卷积层中。这样，每一层不仅能够处理由前一层提供的特征，还能够直接访问原始数据，增强了网络对信息的综合处理能力，使得每个卷积层的潜力得到了更加充分的发挥。通过这种方式，网络的每一层都能对原始图像和传递特征进行综合分析，从而提高了整个网络的性能和效率。

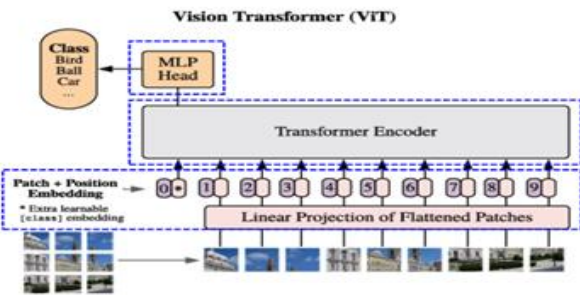


卷积特征提取

在改进视觉变换器（ViT）基础架构时，一个关键的考虑点是对第一步骤中的嵌入（embedding）方法进行优化。原有的 ViT 模型采用的是一个简单的单层卷积网络来处理图像块（patch），但这种方法在特征提取方面存在局限性。为了克服这一点，提出了采用更为复杂和先进的传统卷积模型，如 LeNet-5 和 ResNet-18，作

为图像块的特征提取器。这些模型因其深度和复杂性，能够更有效地捕捉和表示图像数据的细节和高级特征。

还可以考虑结合原图的 RGB 增强技术。这种方法涉及到在原始图像的基础上，应用一系列的预处理步骤，以增强图像的关键特征，从而提高模型对图像中对象的识别和分类能力。具体来说，可以将经过预处理增强的原始图像与通过卷积神经网络（CNN）提取的特征结合起来，共同作为 ViT 模型的输入。这种融合方法有望进一步提升 ViT 模型在图像分类任务中对目标对象特征的提取和识别能力，从而提高整体的分类准确率。通过这样的改进，ViT 模型可以更有效地处理和理解各种复杂的视觉场景，从而在图像分类等任务中取得更好的性能。

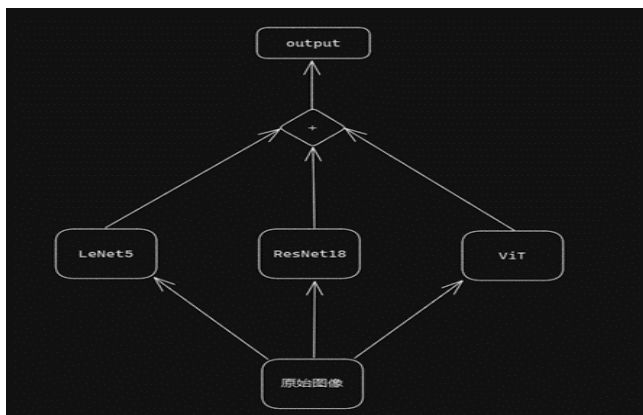


多模型集成学习

在进行了一系列实验后，我们观察到基于 ViT（VisionTransformer）模型和传统的 CNN（卷积神经网络）系列模型的串联组合并没有产生预期中显著的效果。这一发现引发了一个思考：鉴于这两种模型在图像分类的基本原理上存在差异，我们是否可以采用集成学习的策略，同时并行运行多个网络来提高分类的准确性呢？

受到 inception 架构的启发，我们决定采用一种类似的方法，即同时训练多个网络并通过一种类似于投票的机制来决定最终的分类结果。具体来说，我们选取了三种具有代表性的模型：LeNet-5、ResNet-18 和 ViT 作为我们的实验对象。这些模型将并行运行，各自独立进行学习和分类预测。在分类的最终阶段，我们将采用一种多数投票的策略：如果超过两个模型同意某一分类结果，则该结果被认定为最终的类别。

这种方法的优势在于它结合了不同模型的独特视角和学习能力，可能会在一定程度上克服单一模型的局限性，从而提供更为精确和鲁棒的分类结果。通过这种并行训练和多数投票的策略，我们期望能够更有效地利用各个模型的优势，以达到更高的分类准确率。



网络结构

具体内容见附录，建议直接看对应模型的代码即可，每一部分将会直接放上模型部分的 Python 代码。

优化原则

本课题的优化策略虽然不是重点内容，但仍然采用了一种高效且广泛应用的方法：交叉熵损失结合 Adam 优化器。这种组合是常见的优化方法，它有效地结合了两个重要部分：损失函数和优化算法。

交叉熵损失（Cross-Entropy Loss）：这是一种常用的损失函数，特别适用于分类问题。它量化了实际输出和预测输出之间差异，目的是最小化这种差异。在 multi-class 分类问题中，交叉熵损失可以有效地处理输出层的概率分布，使模型能够更准确地预测每个类别的概率。

Adam 优化器（Adam Optimizer）：Adam 是一种基于一阶梯度的优化算法，结合了动量和 RMSprop 的优点。它通过调整学习率来优化训练过程，使得参数更新更加有效和稳定。Adam 优化器因其高效的收敛速度和对超参数的低敏感度而受到广泛欢迎。

这种优化策略在本课题中虽然不是重点，但其稳定性和高效性使得可以专注于研究神经网络模型的其他关键方面，如网络结构设计、特征提取方法等。通过使用这种成熟的优化方法，可以确保模型训练过程的平稳进行，从而更专注于提高模型的性能和准确性。

3. 实验与结果分析

实验数据集

本次实验所选用的数据集为广泛知名的 CIFAR10 数据集。这一数据集包括了总计 60000 张尺寸为 32×32 像素的 RGB 格式图片，每张图片均具有三个颜色通道。在数据集的规模上，CIFAR10 属于较小范围的数据集类别。它的主要特色在于其多样性和适用性，尤其适合于进行计算机视觉领域的基础研究。在实际的实验编码过程中，我们通过使用 Python 编程语言中的 torchvision 库来调用和处理这些图片数据。Torchvision

库为处理此类图像数据集提供了极大的便利，它不仅简化了数据加载和预处理的步骤，还提供了一系列的工具和方法，以便更高效地进行图像处理和模型训练。

实验平台

实验平台采用了基于 Linux 的操作系统，具体为内核版本 3.10.0-1160.88.1.el7.x86_64，它运行在 x86_64 架构的硬件上。这个平台专为高效的性能和稳定性设计，确保了与 Linux 内核的兼容性和优化。为了实现最佳的实验效果，我们细心挑选了一系列必要的 Python pip 包，并将其详细列表整理在 requirements.txt 文件中。这些包涵盖了从数据处理到复杂算法实现的各种需求，保证了实验环境的完整性和实验过程的顺利进行。

实验内容与结果分析

本次实验的内容涵盖了三个关键部分，分别为训练集损失、测试集损失以及模型的准确度。我们详细记录了模型准确度如何随着训练过程中的 epoch 轮数的增加而变化。此外，本实验还包括了三个指标（训练集损失、测试集损失和模型准确度）随着 epoch 轮数变化的详尽图像表示，这些图像清晰地展示了模型在训练过程中的性能变化情况，为我们提供了深入理解模型训练动态的视觉资料，这些图片均在附录中，具体见附录。

原图 RGB 增强

结果分析：在进行卷积神经网络的实验中，我们注意到了一些有趣的现象和趋势。首先，当我们尝试通过增大卷积核的尺寸来提升网络的性能时，结果并不如预期那样理想。这种现象可能源于输入图像的分辨率较低，导致较大的卷积核在处理时反而容易忽略或丢失一些细小的纹理特征。这表明在处理低分辨率图像时，卷积核大小需要更加谨慎地选择，以避免过度平滑和特征丢失。

接着，我们观察到使用版本 v1 的增强模型时，模型的收敛速度和最终的准确率只有轻微的提升。这一结果可能说明在浅层神经网络中，每一层网络都在提取不同范围和类型的特征，而直接对原始图像进行输入可能不是最佳选择。尽管效果有限，但这也提示我们应该探索更为精细和多样化的原图像增强策略，以期达到更好的优化效果。

最后，使用经过 v2 版本加强的模型时，我们观察到了显著的收敛速度提升。特别地，在训练至第 25 个 epoch 时，模型出现了过拟合现象，但在过拟合发生之前，模型的准确率已经超过了未经 v2 版本加强的基线模型。这一发现证明了 V2 版本的有效性，并强调了对模型训练过程的细致监控的重要性，以及在达到最佳性能之前及时调整训练策略以避免过拟合的必要性。

卷积特征提取

结果分析：原始的 VisionTransformer(ViT)使用单层 Transformer 就能在性能上超越 LeNet-5，这表明 ViT 模型即使在简化的架构下也能超越一些基础的 CNN 模型。然而，与更复杂的模型如 ResNet-18 相比，它在准确率上还有较大差距。但有趣的是，通过使用 LeNet-5 模型进行 Embedding 处理，ViT 的准确率可以从 73% 提升至 78%，这与使用两层 Transformer 的 ViT 结果相似。这表明通过适当的 Embedding 方法可以在一定程度上弥补 Transformer 层数较少的不足，使得单层 Transformer 的效果接近于双层。

在使用 ResNet-18 进行 Embedding 的对比中，这一结论更加明显。Transformer 层数的增加在图像分类任务中似乎像是更细致地利用了图像信息，类似于增强图像细节特征提取的方法。因此，在进行大卷积核实验时，效果不如不使用大卷积核时好，可能是因为大卷积核未能有效捕捉到图片中的局部纹理信息，从而未能充分发挥其优势，也未能满足 Transformer 对局部信息的需求。使用两层 Transformer 时也是如此，如果没有恰当的 Embedding 方法来补充细节信息，可能会影响模型性能。

因此，无论是使用单层还是双层 Transformer，当结合 ResNet-18 进行特征提取时，结果都超过了 80%，说明在这种情况下，提取图像本身的信息起到了关键作用。而 Transformer 层数的增加对结果的影响并不大。这表明在设计模型时，合适的特征提取和有效的结构设计比单纯增加层数更为重要。

三层 transformer 及以上

相关研究主要集中在使用两层的 Transformer 结构，原因是在增加到三层或更多层时，遇到了显著的优化难题。这些挑战包括但不限于复杂的参数调整和增加的计算成本，这些因素共同导致效果提升并不明显。因此，尽管 Transformer 模型在深度学习领域具有广泛的应用前景，但目前的研究并没有对其结构进行更深层次的扩展。

多模型集成学习

通过并行的策略将几种方法均运用起来，通过投票的方式进行分类，具体每一层网络架构，框架网络架构同研究思路图。

结果分析

这里最初的设想是通过对 ResNet-18、ViT 和 LeNet-5 这三种不同模型进行集成学习，以提高最终结果的准确性。然而，实验结果表明，集成模型的输出总是稳定地表现出最好的那个模型的效果。我们推测，不论是 Transformer 还是 CNN 等不同模型，在进行特征学习和分类时可能存在相似的错误。或许，Transformer 方法提供了一种不同于 CNN 的思考角度。总而言之，通

过并行比较的结果显示，最终选用的模型在稳定输出方面表现最为出色，可以看作是一种类似 Inception 和继承集成学习思想的模型策略。

4. 结论

课题特点与方法

这个深度学习项目专注于图像处理和分类，涵盖了一系列技术和方法。核心技术包括原始图像增强，通过创新方法将输入图像传递到每个层级，从而提高图像的质量和明晰度。项目中还结合了经典的卷积网络特征提取技术与 VisionTransformer (ViT) 的优化，以进一步提升模型的性能。此外，该项目采用了集成学习的策略，通过多模型组合来增强决策的准确性。

在图像增强方面，该项目不仅在图像处理流程的每个阶段传递原始图像，还通过将卷积结果传递到非直接相连的核，加强了不同层之间的连接。而在 ViT 的优化上，则通过分解 ViT 结构，并引入卷积网络到 embedding 层，来帮助 ViT 更有效地理解图像内容。

集成学习算法方面，项目采用了基于多模型的学习策略，整合了多个模型的预测结果，提高了决策的准确性。此外，通过多分类器的投票系统，能够提供更可靠的决策依据。

综合来看，这个项目不仅融合了新型图像增强技术、CNN 卷积特征提取和集成学习方法，还旨在解决图像增强和网络架构优化的挑战，推动深度学习领域的发展。总体而言，该项目展示了对深度学习和图像处理领域的深入理解，并通过创新的技术和方法，旨在提高图像分类的准确性和效率。

结果：实验总结表，其中单层网络训练 epoch 为 25 轮，多层网络训练 epoch 为 50 轮，涉及到的准确度均为最高准确度，按准确度从小到大排序。

模型	准确度
图像加强 V2LeNet-5 大核	62.55%
图像加强 V1LeNet-5 大核	63.43%
大核 LeNet-5	63.52%
原始 LeNet-5	66.09%
原始图像加强 V1LeNet-5	68.34%
原始图像加强 V2LeNet-5	69.03%
LeNet-5 大核+单层 transformer	69.55%
LeNet-5 大核+双层 transformer	70.43%
原始单层 ViT	73.28%
LeNet-5+单层 transformer	77.43%
LeNet-5+双层 transformer	78.25%
原始双层 ViT	78.48%

• 模型	• 准确度
原始 ResNet-18	86.38%
ResNet-18+双层 transformer	87.47%
ResNet-18+单层 transformer	88.27%

其中集成学习方法是选择模型在对应数据集上的最好模型的结果。

不足及原因

在当前的图像增强部分，执行的工作还较为基础和粗糙。特别是，我们没有采用基于残差连接的先进图像增强技术，这在一定程度上影响了结果的细致度和效果。从最终准确度的角度来看，未加入图像增强的结果与加入后的相似，但这种相似性仅限于表面观察。更深入地分析，特别是观察损失函数随迭代轮数的变化，我们可以明显发现一个关键的差异：加入图像增强技术后，模型的收敛速度显著提升。这表明，尽管最终准确度看似未受显著影响，图像增强技术实际上在优化训练过程中起到了重要作用，提高了模型学习的效率。因此，进一步改进和细化图像增强策略，特别是探索和应用基于残差连接的高级技术，可能会对模型性能产生更加显著的正面影响。

在本次实验中，我们尝试通过增大卷积核尺寸来提升模型的性能。然而，结果表明这种方法并未能显著提高模型在最终测试中的准确率。这一发现提示我们，仅仅增大感知野的尺寸并不一定能带来性能的提升。推测原因可能与输入图像的尺寸有关。由于输入图像仅为 32*32 像素，使用较大的卷积核可能会导致在图像边缘进行过多的 padding，这种不必要的 padding 可能会对模型的学习和泛化能力产生不利影响。

此外，值得注意的是，在一些关于大核卷积的先进研究[5]中，研究者们发现，在某些条件下，大核卷积能够超越视觉转换器（VisionTransformer，简称 ViT）模型的性能。这些研究结果表明，如果选用更加适合的数据集，结合使用大核卷积和 ViT 技术可能会取得更优异的表现。因此，我们建议在未来的研究中，选取更加合适的数据集，并尝试将大核卷积与 ViT 结合应用，以探索模型性能的提升空间。

对未来工作进行探讨

未来的研究和开发工作将主要聚焦于结合大核卷积网络和视觉变换器（ViT）的先进技术。这一领域的关键任务是探索和优化 embedding 方法，特别是在串行和并行处理架构的整合上。此外，我们也将面临如何有效选择大核卷积网络的挑战。针对这一问题，我们目前正考虑采用 inception 架构的多路选择策略。这种方法有望实现自动化的寻找和调整卷积核大小的功能，从而为这一领域带来创新和突破。通过这样的技术

融合和方法创新，我们期望在图像识别、数据处理和深度学习等多个层面实现显著的进步。

5. 参考文献

- [1] Wei Z, Pan H, Li L, et al. DMFormer: Closing the gap Between CNN and Vision Transformers[C]//ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2023: 1-5.
- [2] Lee S H, Lee S, Song B C. Vision transformer for small-size datasets[J]. arxiv preprint arxiv:2112.13492, 2021.
- [3] Touvron H, Cord M, Douze M, et al. Training data-efficient image transformers & distillation through attention[C]//International conference on machine learning. PMLR, 2021: 10347-10357.
- [4] Mehta S, Rastegari M. MobileViT: light-weight, general-purpose, and mobile-friendly vision transformer[J]. arxiv preprint arxiv:2110.02178, 2021.
- [5] Ding X, Zhang X, Han J, et al. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022: 11963-11975.

本项目的 github 地址为：[ylwango613/Network_DLreport: This is the assignment of ylwang for deep learning and neural networks \(github.com\)](https://github.com/ylwango613/Network_DLreport)。

附录

网络结构

图像增强V1

```
class ConvNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(6, 9, 5, padding=2) # 28x28
        self.pool1 = nn.MaxPool2d(2, 2) # 14x14
        self.conv2 = nn.Conv2d(12, 21, 5) # 10x10
        self.pool2 = nn.MaxPool2d(2, 2) # 5x5
        self.conv3 = nn.Conv2d(24, 120, 5)
        self.fc1 = nn.Linear(864, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        in_size = x.size(0)
        old = x
        out = torch.cat([x, old], dim=1)

        out = self.conv1(out) # 24
        out = F.relu(out)
        out = self.pool1(out) # 12

        transform = transforms.Resize((out.shape[2], out.shape[2]))
        old = transform(x)
        out = torch.cat([out, old], dim=1)

        out = self.conv2(out) # 10
        out = F.relu(out)
        out = self.pool2(out)

        transform = transforms.Resize((out.shape[2], out.shape[2]))
        old = transform(x)
        out = torch.cat([out, old], dim=1)
        residual3 = self.conv3(out)

        out = out.view(in_size, -1)
        out = self.fc1(out)
        out = F.relu(out)
        out = self.fc2(out)
        out = F.log_softmax(out, dim=1)
        return out
```

图像增强V2

```
class ConvNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(6, 9, 5, padding=2) # 28x28
        self.pool1 = nn.MaxPool2d(2, 2) # 14x14
        self.conv2 = nn.Conv2d(12, 21, 5) # 10x10
        self.pool2 = nn.MaxPool2d(2, 2) # 5x5
        self.conv3 = nn.Conv2d(24, 120, 5)
        self.conv11 = nn.Conv2d(6, 24, 5)
        self.conv12 = nn.Conv2d(6, 120, 5)
        self.conv21 = nn.Conv2d(12, 120, 5)
        self.fc1 = nn.Linear(9504, 1024)
        self.fc2 = nn.Linear(1024, 10)

    def forward(self, x):
        in_size = x.size(0)
        old = x
        out = torch.cat([x, old], dim=1)
        out12 = self.conv12(out)
        out = self.conv1(out) # 24
        out = F.relu(out)
        out = self.pool1(out) # 12

        transform = transforms.Resize((out.shape[2], out.shape[2]))
        old = transform(x)
        out = torch.cat([out, old], dim=1)
        out21 = self.conv21(out)

        out = self.conv2(out) # 10
        out = F.relu(out)
        out = self.pool2(out)

        transform = transforms.Resize((out.shape[2], out.shape[2]))
        old = transform(x)
        out12 = transform(out12)
        out21 = transform(out21)
        out = torch.cat([out, old], dim=1)
        residual3 = self.conv3(out)

        out = torch.cat([out, out12, out21], dim=1)
        out = out.view(in_size, -1)
        out = self.fc1(out)
        out = F.relu(out)
        out = self.fc2(out)
        out = F.log_softmax(out, dim=1)
        return out
```


ViT+lenet-5

```
# 定义Vision Transformer模型
class visionTransformer(nn.Module):
    def __init__(self, num_classes=10, image_size=32, patch_size=4,
hidden_dim=128, num_heads=8, num_layers=1):
        super(visionTransformer, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5, padding=2) # 28x28
        self.pool1 = nn.MaxPool2d(2, 2) # 14x14
        self.conv2 = nn.Conv2d(6, 16, 5, padding=2) # 10x10
        self.pool2 = nn.MaxPool2d(2, 2) # 5x5
        self.conv3 = nn.Conv2d(16, 128, 5)

        # self.embedding = nn.Conv2d(3, hidden_dim, kernel_size=patch_size,
stride=patch_size)
        self.transformer =
nn.TransformerEncoder(nn.TransformerEncoderLayer(d_model=hidden_dim,
nhead=num_heads), num_layers)
        self.fc = nn.Linear(2048, 1024)
        self.fc1 = nn.Linear(1024,128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        out = self.conv1(x) #24
        out = F.relu(out)
        out = self.pool1(out) #12
        out = self.conv2(out) #10
        out = F.relu(out)
        out = self.pool2(out)
        out = self.conv3(out)

        # out = F.interpolate(out, size=(32, 32), mode='bilinear',
align_corners=False)
        # out = self.embedding(x)
        out = out.flatten(2).permute(2, 0, 1)
        out = self.transformer(out)
        out = out.permute(1, 0, 2).flatten(1)

        out = self.fc(out)
        out = self.fc1(out)
        out = self.fc2(out)
        return out
```

ViT+resnet18

具体见代码。

```
class ViT_ResNet18(nn.Module):
    def __init__(self, num_classes=10):
        super(ViT_ResNet18, self).__init__()

        # 创建ViT模型
        self.vit = visionTransformer(num_classes=num_classes)
```

```

# 创建resnet模型
self.resnet = ResNet18()

def forward(self, x):
    # 使用resnet模型处理ViT的输出
    x = self.resnet(x)
    # x = F.interpolate(x, size=(32, 32), mode='bilinear',
align_corners=False)
    # 使用ViT模型处理输入
    x = self.vit(x)
    return x

```

ViT+Ienet-5大核

```

# 定义Vision Transformer模型
class VisionTransformer(nn.Module):
    def __init__(self, num_classes=10, image_size=32, patch_size=4,
hidden_dim=128, num_heads=8, num_layers=1):
        super(VisionTransformer, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 11, padding=5) # 28x28
        self.pool1 = nn.MaxPool2d(2, 2) # 14x14
        self.conv2 = nn.Conv2d(6, 16, 7, padding=3) # 10x10
        self.pool2 = nn.MaxPool2d(2, 2) # 5x5
        self.conv3 = nn.Conv2d(16, 128, 7, padding=1)

        # self.embedding = nn.Conv2d(3, hidden_dim, kernel_size=patch_size,
stride=patch_size)
        self.transformer =
nn.TransformerEncoder(nn.TransformerEncoderLayer(d_model=hidden_dim,
nhead=num_heads), num_layers)
        self.fc = nn.Linear(2048, 1024)
        self.fc1 = nn.Linear(1024, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        out = self.conv1(x) #24
        out = F.relu(out)
        out = self.pool1(out) #12
        out = self.conv2(out) #10
        out = F.relu(out)
        out = self.pool2(out)
        out = self.conv3(out)

        # out = F.interpolate(out, size=(32, 32), mode='bilinear',
align_corners=False)
        # out = self.embedding(x)
        out = out.flatten(2).permute(2, 0, 1)
        out = self.transformer(out)
        out = out.permute(1, 0, 2).flatten(1)

        out = self.fc(out)
        out = self.fc1(out)
        out = self.fc2(out)
        return out

```

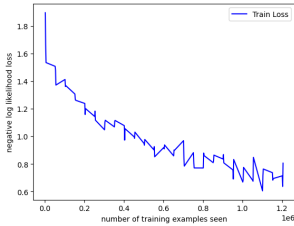
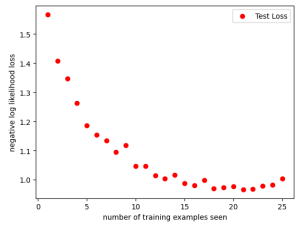
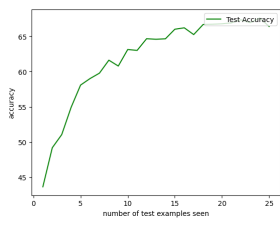
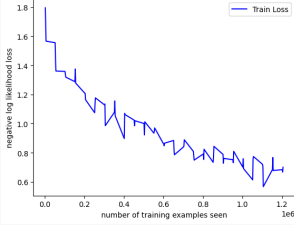
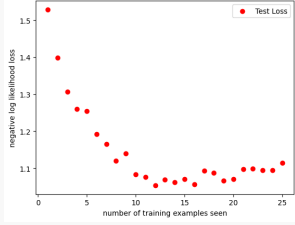
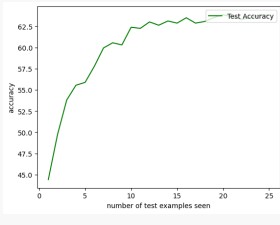
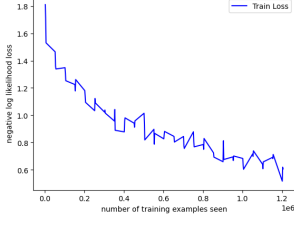
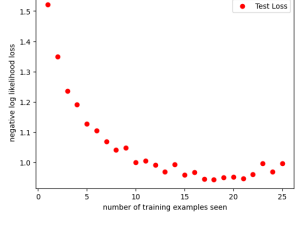
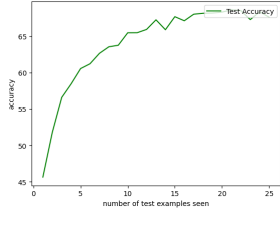
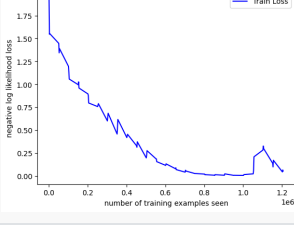
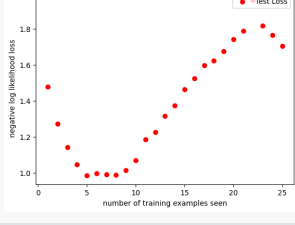
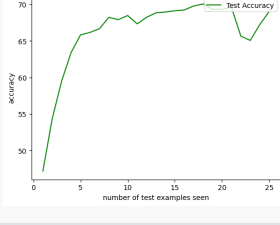
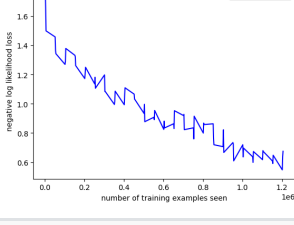
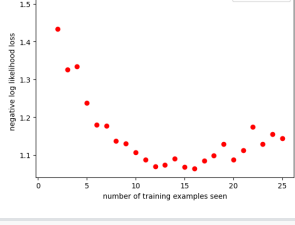
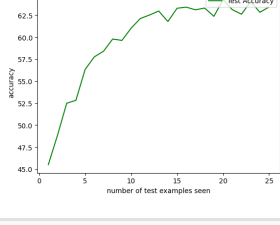
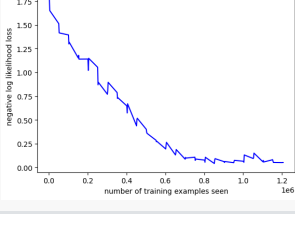
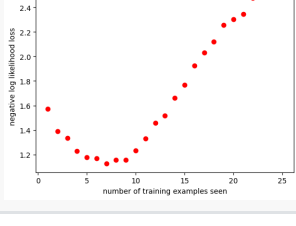
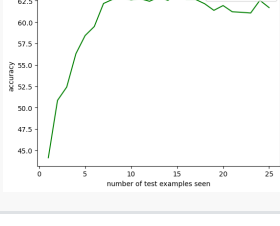
重要网络结构见上，其他具体网络结构见代码即可

代码链接：

```
https://github.com/ylwango613/Network_DLreport
```

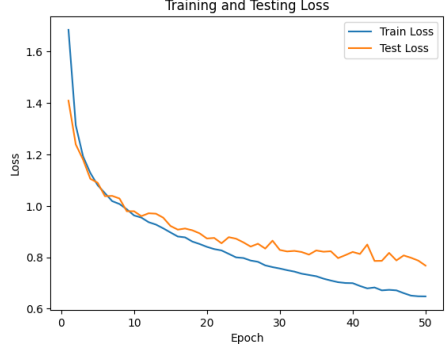
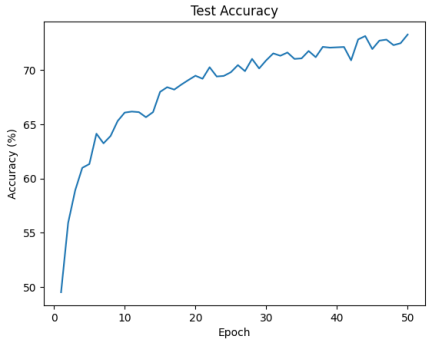

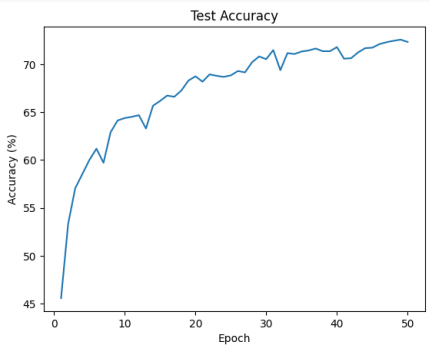
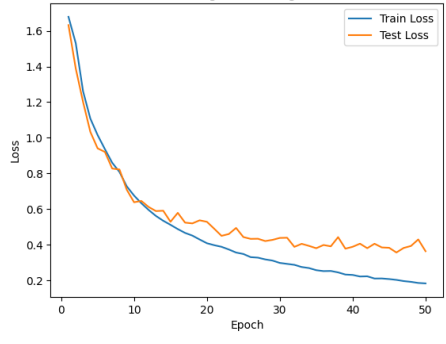
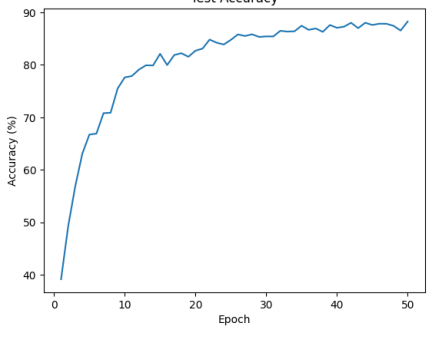
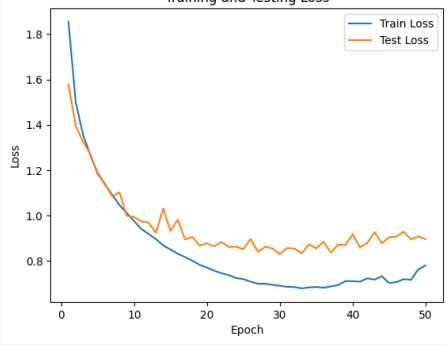
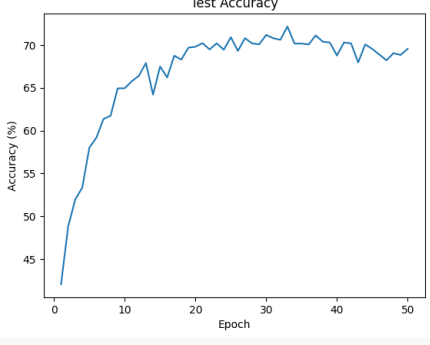
实验结果图

原图RGB增强

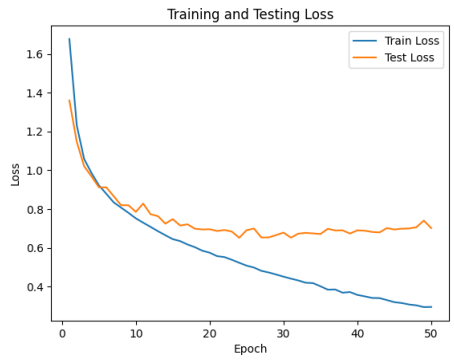
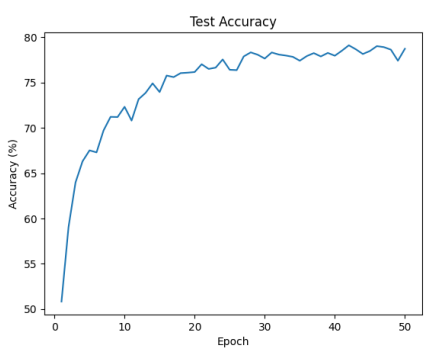
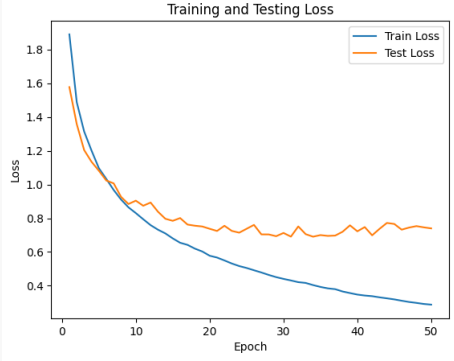
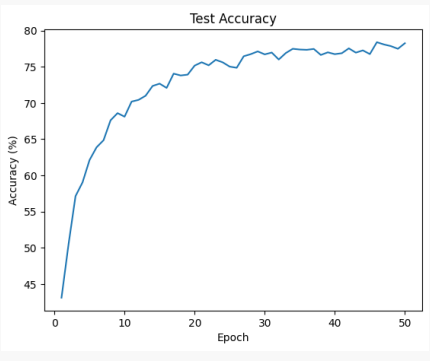
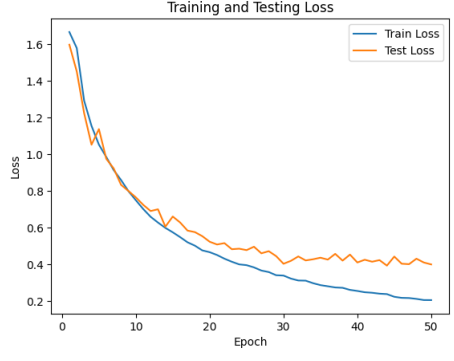
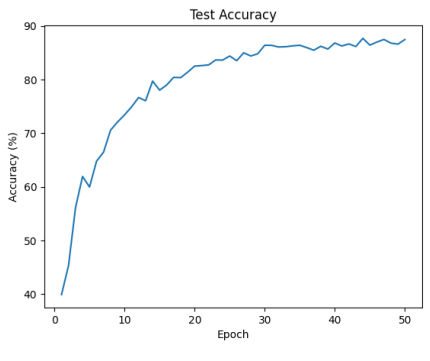
模型	训练集loss	loss变化	准确度变化
LeNet			
大核 LeNet			
V1增强 LeNet			
V2增强 LeNet			
V1增强 大核 LeNet			
V2增强 大核 LeNet			

卷积特征提取

一层transformer

模型	loss变化	准确度变化
无卷积		
LeNet5辅助卷积		
ResNet-18辅助卷积		
LeNet5大核辅助卷积		

两层transformer

模型	loss变化	准确度变化
无卷积		
LeNet5-5辅助卷积		
ResNet-18辅助卷积		
LeNet5大核辅助卷积	