南开大学

Nankai University

南　　开　　大　　学

计　算　机　学　院

深度学习期末实验报告

深度学习

管昀玫 2013750

宋佳蓁 2013904

康佳玉 2011605

石家琪 2011739

李玥婵 2013177

年级：2020 级

专业：计算机科学与技术

指导教师：候淇彬

2023 年 6 月 25 日

摘要

在本次作业中，我们总共复现了 8 种计算机视觉相关的深度学习模型，分别为：ResNet, VAN, RepLKNet, Coordinate Attention, Res2Net, Triplet Attention, SKNet, GCNet；同时，分别针对 VAN, Res2Net, Triplet Attention 这三个网络提出改进，并尝试进行模型的融合，获得了良好的效果。我们组最佳改进模型为 Triplet-CA，Top-1 Acc 为 65.76%，Top-5 Acc 为 88.91 %，改进最大提升模型为 Res2Net-cCD-ST，Top-1 Acc 提升了 32.25% (60.90 vs. 46.05)，Top-5 Acc 提升了 10.44% (84.85% vs. 74.41%) ，其余的改进模型也有或多或少的提升效果。

关键字：深度学习；图像处理；注意力机制

目录

一、 组内分工

(一) 论文复现

- 管昀玫：复现 RepLKNet
- 李玥婵：复现 ResNet, VAN
- 康家玉：复现 Res2Net，Triple Attention, SKNet, GCNet
- 石家琪：复现 Coordinate Attention

(二) 创新与实施

- 石家琪：负责 Triplet Attention 相关的创新，详见一；负责相关消融实验
- 管昀玫：负责 VAN 相关的创新，详见二；负责相关消融实验
- 宋佳蓁：负责 Res2Net 相关的创新，详见三；负责相关消融实验

(三) 报告撰写

- 管昀玫：负责模型改进的 VAN 部分，消融实验的 VAN 部分，MACs 与 Params 计算，摘要部分，组内分工部分
- 宋佳蓁：负责模型改进的 Res2Net 部分，消融实验的 Res2Net 部分，实验设置部分，总结部分
- 石家琪：负责模型改进的 Triplet Attention 部分，消融实验的 Triplet Attention 部分，总体模型表现分析
- 李玥婵：负责论文复现的 ResNet, VAN, RepLKNet, Coordinate Attention
- 康家玉：负责论文复现的 Res2Net, Triplet Attention, SKNet, GCNet

二、 论文复现

(一) ResNet

当深度神经网络的层数增加时，传统的网络往往会出现梯度消失和模型退化的问题。这是因为在反向传播过程中，梯度信号需要逐层传递，经过多次连续的激活函数和权重更新，梯度会逐渐变小，导致难以训练深层网络。

为了解决深度学习的退化问题，微软研究院的何恺明、张祥雨等人提出了残差学习 [5] 的概念。残差学习的核心思想是引入跳跃连接，允许网络学习残差映射。在传统的网络中，每一层的输出是通过非线性变换得到的，而残差学习将输出分为两部分：一部分是经过非线性变换得到的新特征，另一部分是直接连接上一层的输入特征，即残差。这样，网络可以通过学习残差来逐渐逼近目标映射。

为了适应网络深度和特征图的维度变化,ResNet 中存在着两种映射方式，Identity Mapping（恒等映射）和 Residual Mapping（残差映射），它们分别用于描述残差单元中的跳跃连接。

- Identity Mapping（恒等映射）：在某些残差单元中，输入特征图的维度与输出特征图的维度相同，此时可以直接将输入特征图传递到后续层中，从而保持特征的完整性。在这种情况下，恒等映射通过跳跃连接实现，可以避免信息的丢失和额外的变换。它能够有效地传递梯度，防止梯度消失和梯度爆炸问题，使得网络更容易优化。恒等映射适用于层数较少的网络层，其中特征图的尺寸和通道数变化较小，因此可以直接保留输入特征图的信息。

- Residual Mapping（残差映射）：在其他残差单元中，输入特征图的维度与输出特征图的维度不同，此时需要使用一个额外的 1x1 卷积层来调整输入特征图的通道数和维度，使其与输出特征图的维度相匹配。然后将调整后的输入特征图与输出特征图相加，得到残差（residual）。这个残差通过卷积层的变换，捕捉了输入特征图与期望输出之间的差异，称为残差映射。残差映射适用于层数较多的网络层，其中特征图的尺寸和通道数变化较大，因此需要通过调整输入特征图的维度和使用残差映射来更好地学习和优化网络。

通过使用不同的映射方式，ResNet 能够灵活地适应网络的深度和特征图的维度变化，同时解决梯度消失和模型退化问题。

具体来说，对于 ResNet-18 和 ResNet-34，它们的残差结构如下所示：

残差块（Residual Block）：

第一个卷积层：3x3 的卷积核，步长为 1，用于进行特征提取。

第二个卷积层：3x3 的卷积核，步长为 1，用于进一步提取特征。

跳跃连接：恒等映射，将输入特征图直接与输出特征图相加。

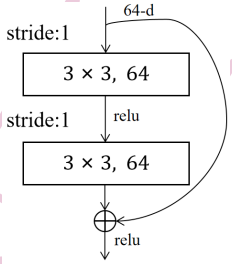


图 1: resnet 低层数结构

这个简单的残差结构在层数较少的网络中重复使用，构建了整个网络。

而对于层数较多的网络（如 ResNet-50、ResNet-101 和 ResNet-152），采用的是 BottleNeck 结构的残差块。它们的残差结构如下所示：

残差块（Residual Block）：第一个卷积层：1x1 的卷积核，用于降低输入特征图的维度和通道数。第二个卷积层：3x3 的卷积核，用于进行特征提取。第三个卷积层：1x1 的卷积核，用于恢复特征图的维度和通道数。跳跃连接：残差映射，将经过第三个卷积层的输出特征图与输入特征图相加。

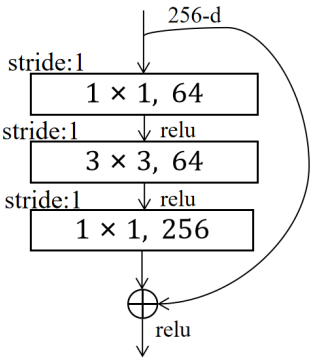


图 2: resnet 高层数结构

这种 BottleNeck 结构在层数较多的网络中使用，它使用了更少的参数，同时提升了网络的表达能力。

总结起来，层数较少的 ResNet 网络（如 ResNet-18 和 ResNet-34）使用简单的残差结构，每个残差块由两个卷积层组成，采用恒等映射。而层数较多的 ResNet 网络（如 ResNet-50、ResNet-101 和 ResNet-152）使用 BottleNeck 结构的残差块，每个残差块由三个卷积层组成，采用残差映射。这种设计可以满足不同深度网络的需求，并在训练深层网络时解决梯度消失和模型退化的问题。

在 ResNet 中，通常使用一个初始的卷积层进行降采样，以减小输入图像的尺寸。在每个残差块之间，特征图的尺寸减半，通道数加倍。这种策略增加了网络的感受野和特征提取能力。最后，通过全局平均池化将特征图转化为固定长度的向量，并连接一个全连接层用于分类任务。

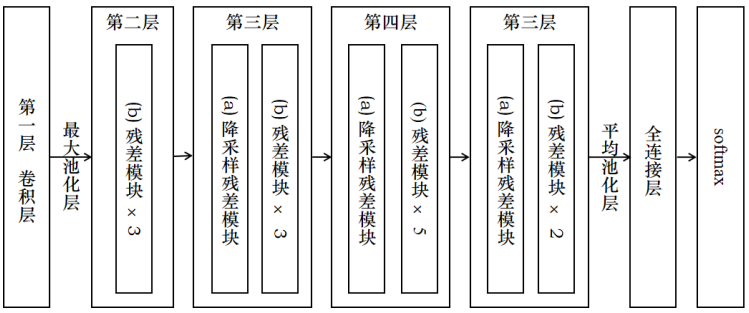


图 3: resnet 模型架构

通过引入残差连接，ResNet 使得梯度可以更容易地在网络中传播，解决了深度网络训练过程中的梯度消失和模型退化问题。这使得 ResNet 可以构建非常深的网络，如 ResNet-50、ResNet-101 和 ResNet-152 等，在图像分类、目标检测和语义分割等计算机视觉任务中取得了显著的性能提升。ResNet 的设计思想也对后续网络架构的发展产生了重要影响。

(二) VAN

Visual Attention Network（视觉注意力网络）是一种专为计算机视觉任务设计的注意力机制，结合了自注意力和卷积操作的优点，克服了它们各自的不足。自注意力首先成功应用于自然语言处理领域，但直接将其用于图像处理存在一些问题。

首先，自注意力在处理一维序列结构方面表现出色，但忽略了图像自身的二维结构信息。其次，由于自注意力本身的复杂度问题，难以处理高分辨率图像。此外，自注意力只考虑了空间上的自适应性，而忽略了通道维度上的自适应性，而在一些网络中已经证明了通道注意力的重要性。

因此，为了克服这些问题并更好地适应计算机视觉任务，需要设计一种特定的注意力机制。为了克服上述的缺点，清华南开团队提出对一个大核卷积运算进行分解从而获得远程关系 [4]。

LKA(Large Kernel Attention：利用自注意和大核卷积的优点，提出分解一个大核卷积运算来获取远程关系。如图所示，一个大的核卷积可以分为三部分

- 一个空间局部卷积 (depth-wise 的卷积)
- 一个空间远程卷积 (depth-wise 的空洞卷积)
- 一个通道卷积 (1×1 卷积)

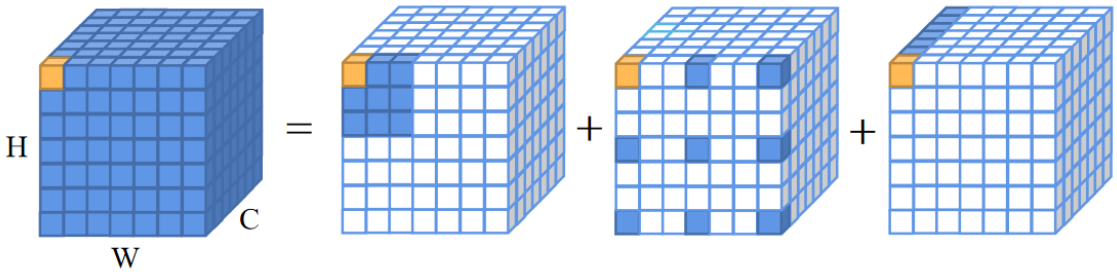


图 4: LKA 大核注意力机制

通过上述分解，可以捕捉到计算量和参数较小的远程关系。在获得远程关系之后，就可以估计一个点的重要性并生成注意图。其组成结构如下图所示。

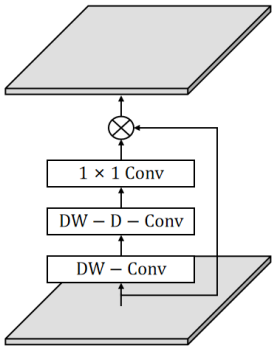


图 5: LKA 结构图

基于上述的 LKA 使用 Attention - FFN 结构搭建一个新的基于注意力机制的视觉主干网络 VAN，网络结构如下:

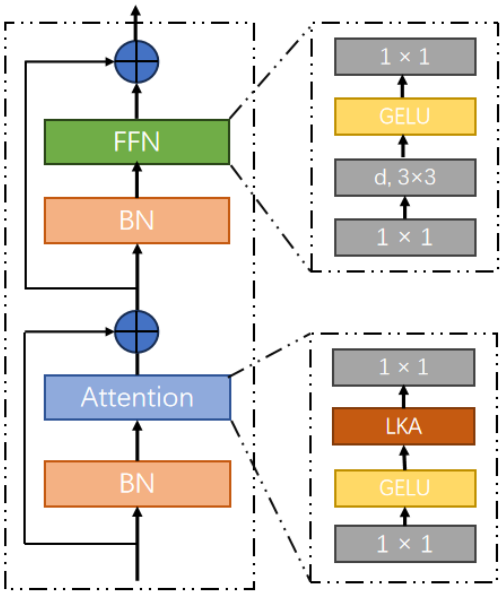


图 6: VAN 网络结构

通过使用 LKA 方法，Visual Attention Network 能够更好地适应图像处理任务。它综合了自注意力和卷积操作的优点，充分利用了图像的二维结构信息，并考虑了通道维度上的自适应性。这种注意力机制能够捕捉长距离依赖关系，并具有空间维度上的自适应性。相比于简单地沿用自然语言处理中的自注意力机制，Visual Attention Network 通过 LKA 方法专门为计算机视觉任务设计了一种更为有效的注意力机制。

(三) replknet

ReplkNet（Replicated Large-Kernel Network）是一种使用超大卷积核的模型，通过结构重参数化和 depthwise 卷积等设计要素，充分利用大核卷积的优势，在目标检测和语义分割等任务上超越了传统小卷积模型，并且在性能和速度方面超过了 Swin Transformer。同时，ReplkNet 也通过结构重参数化和 depthwise 卷积等技术来解决大核卷积计算量过大的问题.

为了使用好超大核卷积，清华旷世团队通过进行了一系列探索实验，得出了一些高效使用的指导原则 [2]。

- 大深度卷积核在实际使用中是高效的。大卷积核被诟病的原因在于其参数量和计算量随着核大小平方级增长，而深度卷积核可以解决这一问题。
- 有无显式的残差连接非常重要，在大卷积核情形下更加明显。
- 用小卷积核重参数化有助于解决优化问题。
- 大卷积核在下游任务上的性能提升多于在 ImageNet 上的提升幅度。
- 大卷积核在小卷积核的特征图上也很有用。

基于上述指导思想，论文提出网络结构如下:

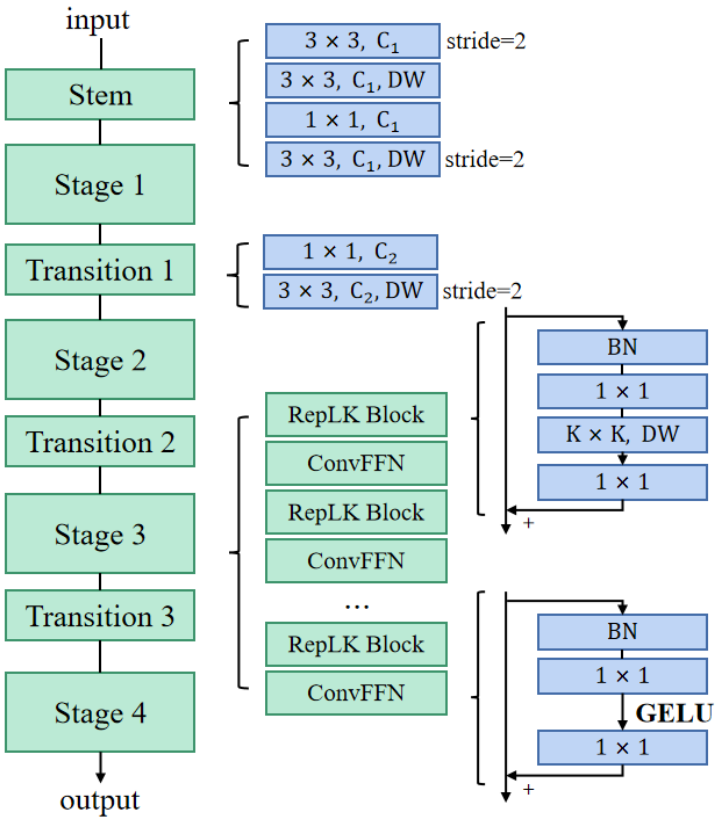


图 7: RelpkNet 网络结构

大核卷积对于下游任务的提升更为明显,通过增加卷积核尺寸可以大幅提升感受野(Effective Receptive Fields, ERFs),进而提升下游任务的性能。此外,大核卷积对于小图仍然有效,即使在特征尺寸相当大甚至更大的情况下,进一步增加感受野仍然可以提升下游任务的性能。ReplkNet 通过使用大核卷积和相关优化技术,在计算机视觉任务中取得了优秀的性能,并具备快速推理的能力。它的设计思想和技术手段为利用大核卷积在图像处理中的潜力提供了重要参考。

(四) coordinate attention

Coordinate Attention 是一种新的注意力机制,旨在解决在移动网络中应用传统注意力机制时遇到的计算开销和信息损失的问题。传统的注意力机制如 SE、BAM 和 CBAM 在大型网络上表现良好,但在移动网络上表现相对较差,因为它们无法承受计算开销,并且无法捕获长范围依赖的信息。

Coordinate Attention 通过将位置信息嵌入通道注意力中,使得移动网络能够获取更大范围的信息而不引入过大的计算开销。为了避免 2D 全局池化导致位置信息丢失,Coordinate Attention 使用两个并行的 1D 特征编码来高效地整合空间坐标信息到生成的注意力图中 [7]。

该种注意力机制通过在水平方向和垂直方向上进行平均池化,再进行 transform 对空间信息编码,最后把空间信息通过在通道上加权的方式融合。具体而言,通过两个 1D 全局池化操作,将输入特征沿垂直和水平方向分别聚合为两个具有方向感知的特征图。然后,将这两个具有嵌入方向信息的特征图编码为两个注意力图,每个注意力图都捕捉输入特征图在一个空间方向上的远距离依赖关系,从而保存了位置信息。最后,将这两个注意力图与输入特征图进行逐元素乘法,以强调感兴趣区域的表示。

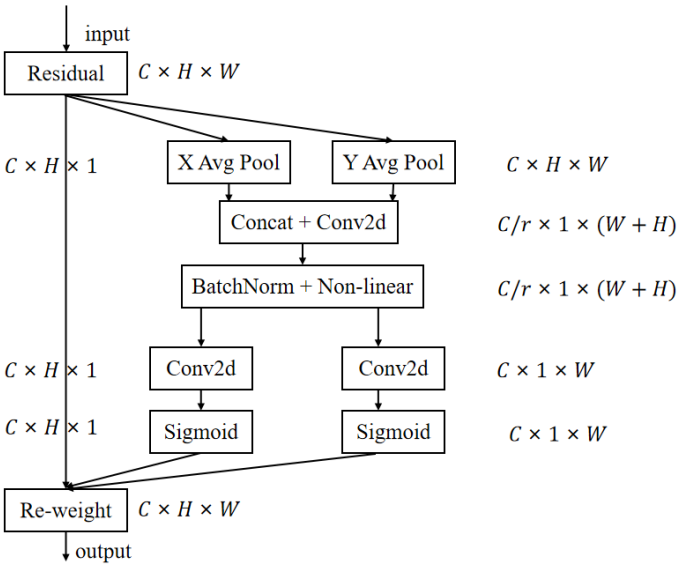


图 8: coordinate attention 网络结构

Coordinate Attention 具有以下三个优点:

- 不仅捕捉了跨通道的信息，也包含了方向感知和位置敏感的信息，从而使得模型能够更准确地定位和识别目标区域。
- 这种方法具有灵活性和轻量性，可以轻松地插入到现有的经典移动网络中，例如在 MobileNet_v2 的倒残差块和 MobileNeXt 的沙漏块中，以提升特征表示能力。
- 对于预训练模型，使用 Coordinate Attention 可以显著提升使用移动网络处理的下游任务的性能，尤其是密集预测任务，如语义分割。

通过引入位置信息并结合方向感知，Coordinate Attention 提供了一种有效的注意力机制，使得移动网络能够更好地处理图像任务，并在性能和计算效率上取得显著改进。

(五) GCNet

捕获长距离依赖关系的目标是对视觉场景进行全局理解，对很多计算机视觉任务都有效，比如图片分类、目标检测、语义分割等。为了捕获长距离依赖关系，NLNet 采用自注意力机制来建模像素对关系。然而 NLNet 对于每一个位置学习不受位置依赖的注意力图，造成了大量的计算浪费。而 SENet 用全局上下文对不同通道进行权值重标定，来调整通道依赖。然而，采用权值重标定的特征融合，不能充分利用全局上下文。

GCNet 是于 2019 年提出的一个基于全局上下文的注意力机制。在 [1]，作者首先实现了 Simplified NL Block，计算一个全局的注意力图来简化 NL block，并且对所有位置共享该全局注意力图。通过对比 NL block 和 Simplified NL Block，发现全局上下文是不受位置依赖的。之后，作者在 Simplified NL Block 的基础上，结合 SE block 搭建了 GCNet，使得网络计算量相对较小，又能很好地融合全局信息。

GCNet 在 context modeling 阶段使用了 Simplified NL block 中的机制，可以充分利用全局上下文信息；同时在 Transform 阶段借鉴了 SE block。其网络结构如下:

- **context modeling**: 采用 1x1 卷积和 softmax 函数来获取 attention 权值，然后执行 attention pooling 来获得全局上下文特征。

- **transform**: 为了获得 SE block 轻量的优点，将 Simplified NL block 中的 1x1 卷积用 bottleneck transform 模块来取代，可以显著的降低参数量。由于两层 bottleneck transform 增加了优化难度，所以在 ReLU 前面增加一个 layer normalization 层来降低优化难度且提高了泛化性。

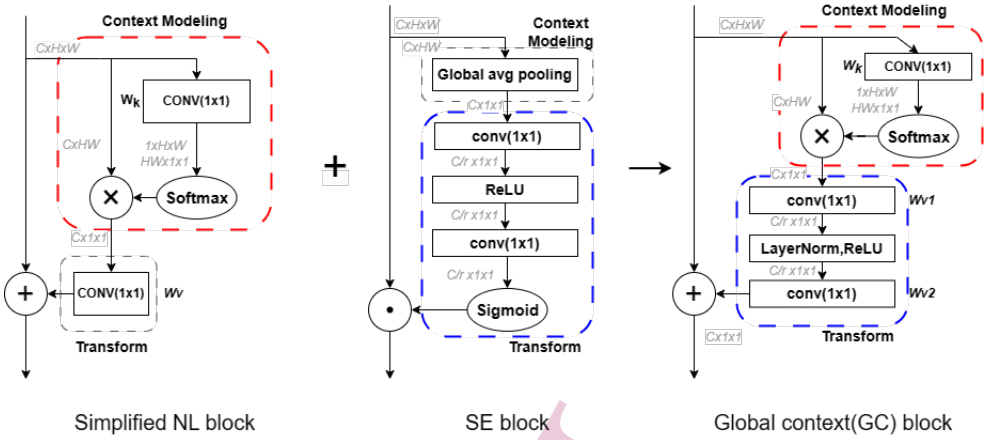


图 9: GCNet 结构图

GCNet 利用一种全局平均池化的方法来捕获全局上下文信息，并在融合多个特征图时将全局上下文信息与局部特征进行融合，在图像分割和目标检测任务中可以取得较好的效果。GC block 在 ResNet 中的使用位置是每两个 stage 之间的连接部分。在 [1] 中，作者将 GC Block 添加到 Resnet50 的 c3、c4、c5 残差结构中。本文中后续也将 GC Block 添加到 Resnet50 的 c3、c4、c5 残差块中进行实验。

(六) SKNet

传统的卷积网络在每层都是相同的感受野大小，由于卷积核的固定尺寸，模型对不同尺度的特征表达能力有限。较小的卷积核可以更好地捕捉细节特征，但在较大的感受野范围内缺乏上下文信息；而较大的卷积核可以获取更多的上下文信息，但会忽略一些细节特征。而在神经科学中，视觉皮层单元中的感受野大小应该是受外部刺激变化而自适应调整。因此，在 [9] 中提出了 SKNet。SKNet 是一个自适应性的注意力编码机制，对不同输入使用的卷积核感受野不同, 参数权重也不同, 可以自适应的对输出进行处理。

SKNet 对多个不同感受野大小的分支进行融合，融合中使用的是类 SENet 结构，最后使用类似门机制的 softmax attention 来更新分支的权重。SKNet 主要由以下三部分构成:

- **Split**: 即进行不同感受野卷积操作。图中上面的分支为 3x3 kernel, dilate size=1 的卷积; 而下面的分支为 5x5 的 dilate size=2 的卷积。
- **Fuse**: 即进行特征融合。首先，通过逐元素进行相加对不同分支的输出进行融合；再进行全局平均池化，获得每一个通道上的全局信息; 之后，对输出做全连接找到每一个通道占的比重大小。
- **Select**: 即对 Fuse 得到的两个分支的通道权重矩阵进行每个位置各自的 Softmax 操作，类似门机制，可得到两个分支中每个通道各自的权值。最后，与原始特征相乘，并进行特征叠加。

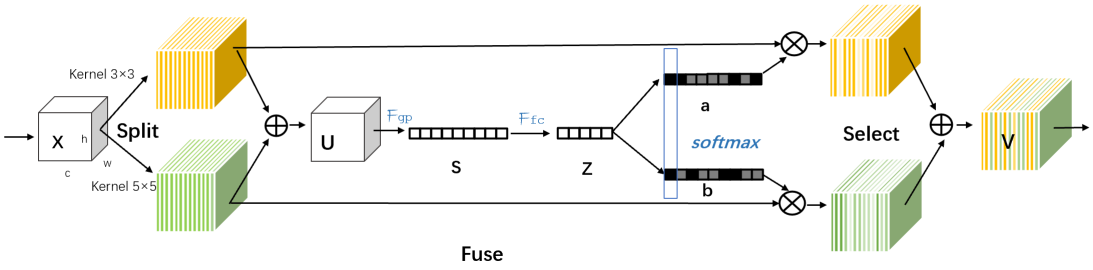


图 10: SKnet 结构

SKNet 能够自适应地选择性地融合不同尺度的特征，增强了模型对不同尺度特征的建模能力, 从而更好地表达图像中的细节和上下文信息。在 Select 部分结合了 soft attention 以及 SENet 的思想，考虑了不同通道和不同块的卷积权重，在多个基准数据集上取得了优异的性能。使用 SK 注意力机制构建 ResNet，也就是将残差块中的 3*3 的普通卷积核，替换成 SK 卷积核。本文通过 SK 注意力机制构建了 ResNet50 网络。

(七) Triplet Attention

Triplet Attetion 是在 2020 年提出的一个轻量且有效的注意力机制。不同于 CBAM 和 SENet, Triplet Attetion 是一个几乎无参数的注意力机制, 通过使用三分支结构捕获跨维度交互来计算注意力权重。对于输入张量，Triplet Attetion 通过旋转操作，然后使用残差变换建立维度间的依存关系，并以可忽略的计算开销对通道间和空间信息进行编码。

利用空间注意力和通道注意力可以提高神经网络的性能, 让网络知道去哪里关注并进一步关注目标对象。在 CBAM 和 SEnet 中，通道注意力的获取是简单的将输入减少到单通道输出获得注意力权重，并且中间使用了降维操作。[10] 中认为 CBAM 计算通道注意力时结合了降维功能，在捕获通道之间的非线性局部依赖关系方面是多余的。因此，[10] 提出了可以有效解决跨维度交互的 Triplet Attetion，强调了多维交互而不降低维度的重要性，消除了通道和权重之间的间接对应。

Triplet Attetion 的网络结构如下图11所示，其主要包含 3 个分支，其中两个分支分别用来捕获通道 C 维度和空间维度 W/H 之间的跨通道交互，剩下的一个分支就是传统的空间注意力权重的计算。

- 第一个分支: 通道注意力计算分支, 输入特征经过 Z-Pool, 再接着 7 x 7 卷积, 最后 Sigmoid 激活函数生成空间注意力权重
- 第二个分支：通道 C 和空间 W 维度交互捕获分支，输入特征先经过 permute，变为 H X C X W 维度特征，接着在 H 维度上进行 Z-Pool，后面操作类似。最后需要经过 permuter 变为 C X H X W 维度特征，方便进行 element-wise 相加
- 第三个分支：通道 C 和空间 H 维度交互捕获分支，输入特征先经过 permute，变为 W X H X C 维度特征，接着在 W 维度上进行 Z-Pool，后面操作类似。最后需要经过 permuter 变为 C X H X W 维度特征，方便进行 element-wise 相加
- 最后对 3 个分支输出特征进行相加求 Avg

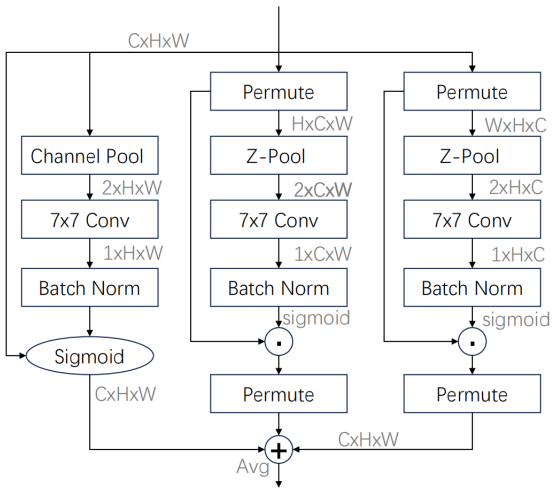


图 11: Triplet Attetion 结构图

(八) Res2net

传统的 CNN 主要通过分层的方式表示多尺度特征。后续，出现 InceptionNets [11] 系列在卷积层使用不同大小的卷积核、ResNet [5] 使用残差结构、DenseNet [8] 使用密集连接等方式来提高网络的多尺度信息提取能力。Res2Net [3] 由南开大学程明明组在 2019 年提出，主要贡献是对 ResNet 模型中的残差块进行了改进，在计算负载不增加的情况下，特征提取能力更强。Res2Net 不同于以往分层的方式，而是通过在一个残差块内构造分层的类残差连接，以细粒度级别表示多尺度特性，并增加了每个网络层的感受野范围。

Res2Net 的主要思想是将残差结构中的 3×3 卷积层接收到的来自输入层 1×1 卷积后的特征图分解为四部分，第一部分不进行操作，第二部分通过一个 3×3 卷积层，第三部分在通过一个 3×3 卷积层前与第二部分卷积后的特征图相加，第四部分在通过一个 3×3 卷积层前与第三部分卷积后的特征图相加，最终将得到的四个部分的特征图进行拼接成与输入层输出同样层数的特征图再送到输出层做最后的 1×1 卷积。Res2Net 还提出了一个新的概念：尺度（scale），scale 越大，多尺度表征能力更强。将输入 X 拆分成四个部分，每个部分通过不同的卷积层之后再融合到一起，得到的输出会获得更大的感受野，而且一些额外的计算开销可以忽略。此外，在 Res2Net 中 3×3 卷积层的数量可以任意调整，在 1×1 网络的最后可以加上 SE 模块。SE 模块主要包括压缩和激励两个步骤，通过学习通道权重来增强有用的特征并抑制不重要的特征，提高了网络的表达能力和泛化能力。

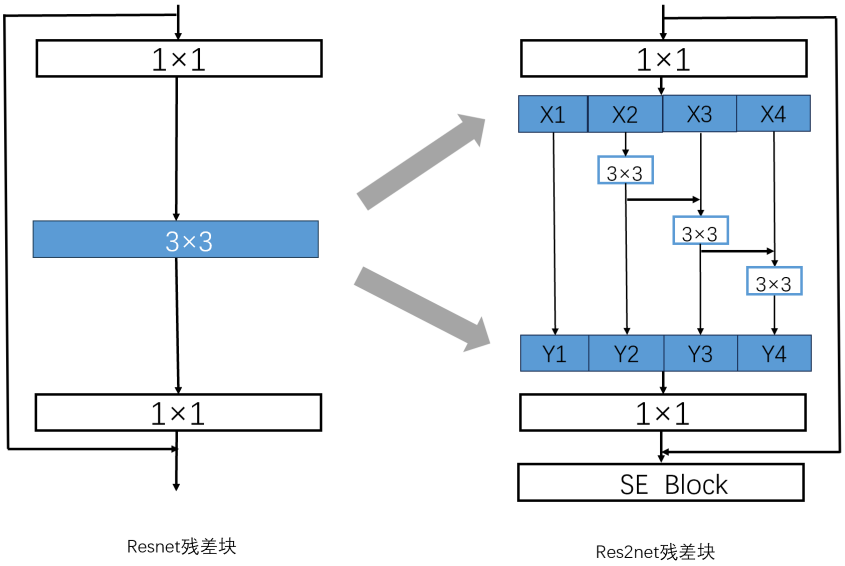


图 12: res2net 残差结构图

综上，Res2Net 是一种通过引入多尺度特征表示的残差连接结构来提高模型性能的深度卷积神经网络。它对于处理多尺度物体和提取细节信息非常有效，已经在计算机视觉领域得到广泛应用。Res2Net 模块可以很容易的集成到先进的模型中，如 ResNet，ResNeXt，和 DLA 网络，并取得了良好的效果。本文主要将 Res2Net 模块集成到 ResNet50 网络中并进行了后续改进实验。

三、 模型改进

(一) Triplet Attention

triplet 注意力机制相比于 SE 等其他注意力机制的独特之处在于交互的三维处理方式，该方式可以获取到除了通道维度和空间维度外隐藏的信息。为了实现这一机制，原作者首先利用旋转将维度间的顺序调整，然后经过 z-pool 池化层从三维数据中提取一个维度的信息，将其压缩成两个二维的数据，然后经过 7x7 的卷积层和 BatchNorm 层，最后通过激活函数输出三个通道的特征图，求平均后将其合成一个最终的特征图。

在这个过程中，我注意到以下几个细节处理:

1. z-pool: 原论文 [10] 中的 z-pool 使用的是 Avg,Max 提取其中一个通道的关键信息。
2. HxW 维度: 空间特征提取方式多样，有很多优秀的方法值得尝试。这里参考了 coordinate 注意力机制和 LKN 注意力机制的实现方式，提出两种新型注意力机制 TripletCA 和 TripletLKN。
3. 7x7conv: 该卷积层和 BatchNorm 层中间存在一个激活层，使用的是 relu 激活函数。
4. sigmoid 层：论文 [10] 中使用 sigmoid 作为激活函数。

基于上述观察，我提出修改 z-pool、尝试不同激活函数、修改空间特征提取通道三个主方向的改进，并最终在各个改进方向均获得优于原模型的结果。

1. z-pool

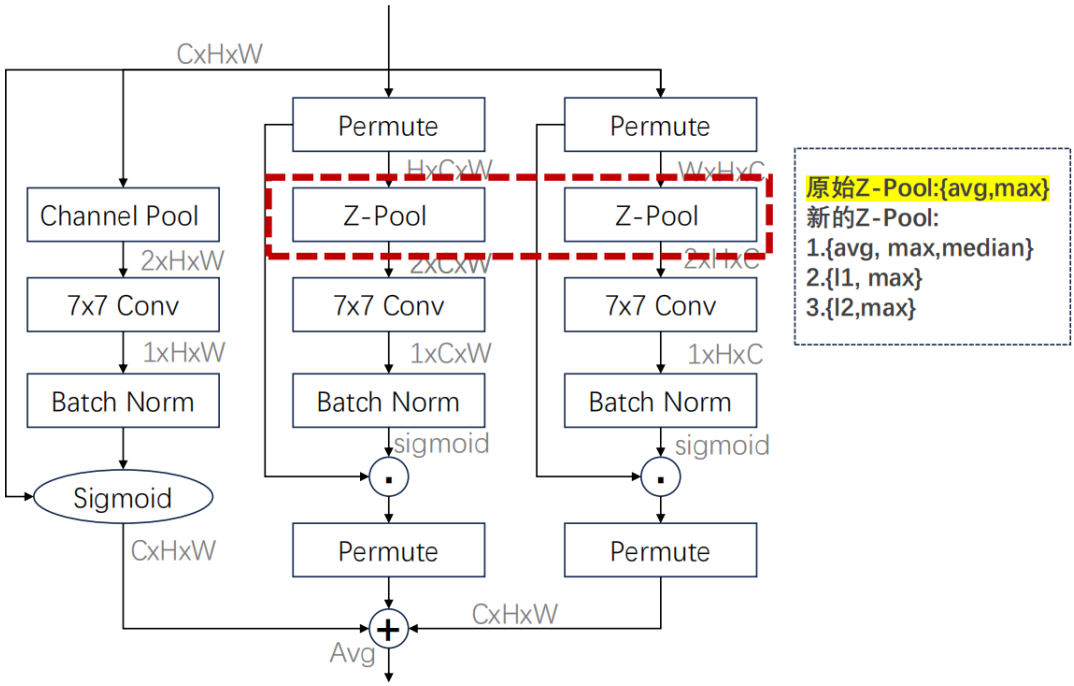


图 13: z-pool

原论文中使用的方法是获取通道的 Max 与 Avg，但是还可以获取通道的其他特征量，如中位数、L1 范数、L2 范数。Max 可获取到一维通道数值的最大值，Avg 可获取到一维通道数值的平均水平。仅利用这两个特征值并不能全面地获取该维度的全部特征。

当数据集的分布情况较为均匀时，平均数更有说服力。因为平均数能够反映出数据的总体水平，且能够很好地描述数据集中的大部分数据。但是，当数据集的分布情况较为不均匀时，中位数更有说服力。这是因为在这种情况下，平均数容易被极端值所影响，从而反映出的数据集的中心位置不够准确。而中位数则能够很好地描述数据集中的大部分数据，不会被极端值所影响。

L1 范数可以用作稀疏性的度量。通过计算向量的 L1 范数，可以确定向量中具有非零值的元素数量。较小的 L1 范数表示向量具有更多的零元素，即向量更加稀疏。与此同时，L1 范数还可以用于异常检测问题。通过计算数据点到中心点的 L1 范数，可以测量数据点与正常数据分布的偏离程度。较大的 L1 范数可能表示数据点较远离正常分布，因此可以被视为异常或离群点。

L2 范数可以用作向量的长度度量。通过计算向量的 L2 范数，可以得到向量的欧几里德长度或模长。因此 L2 范数可以获取数据的整体大小。

将上述描述数据特征的方式进行组合，可以生成不同的 z-pool 层，从不同角度提取数据的特征属性，更好地概括数据。组合方式如下：

- z-pool:[”avg”, ”max”]
- z-pool(median):[”avg”, ”max”,”median”]
- z-pool(l1):[”l1”, ”max”]
- z-pool(l2):[”l2”, ”max”]

2. TripletCA

coordinate 的意义在于：在获取通道特征的时候能够结合空间特征，即通过将空间拆分成两个方向的向量，分别获得该方向的通道特征 (CxWx1、Cx1xH)。

triplet 的意义在于：讨论了空间与通道之间的交互关系，分别获得了每个维度的特征图 (例：W 维-CxHx1)，最后能获得三个维度分别的特征图。

两者相同之处：其 Avg 池化均为 1 维池化，针对某一维度获取全局特征。

不同之处：最后特征图使用的位置不同。coordinate 在最后将特定通道、特定位置的三维权重相乘；triplet 先将每个权重图与原矩阵相乘，再求三者的平均。

个人认为，triplet 的优越点在于它将 C、W 和 C、H 这两个维度获取了交互信息，以往的模型均基于 W、H 的交互信息，再将其与 C 相乘。coordinate 的优势在于它符合直觉，在获取通道特征时能够有效结合空间特征。

我选择以 triplet 为骨干，使用 coordinate 的方法改进 C 维 (WxH) 部分的特征图提取，并尝试用这个方法提取其他交互维度的特征图。该注意力机制结构如图14所示。

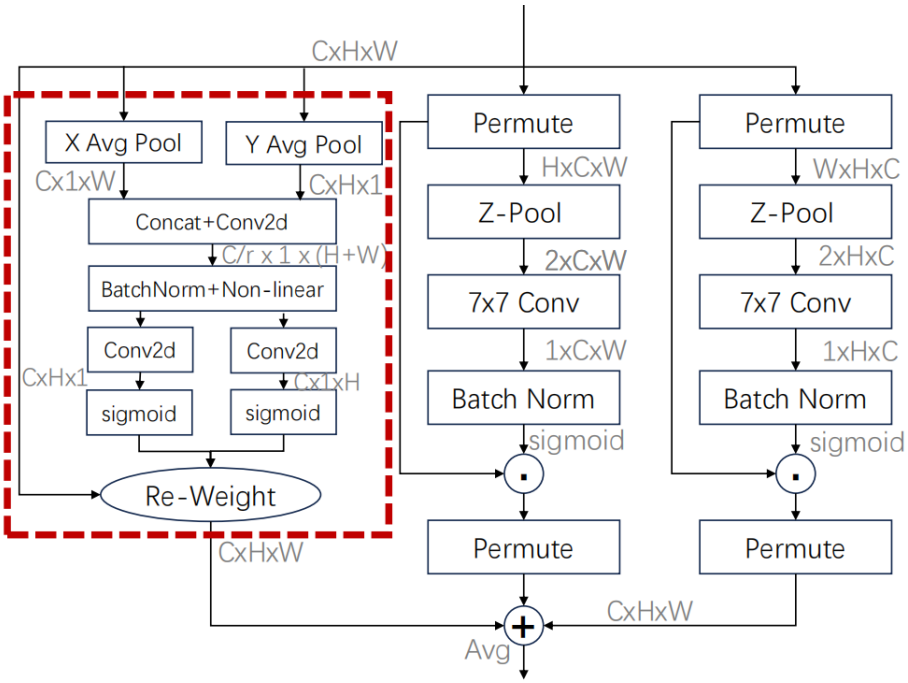


图 14: tripletCA

3. TripletLKN

VAN 中使用的 LKN 用于获取各点的 attention map，其思想是将大核拆成三个小卷积，分别是局部空间卷积（H,W），分布空间卷积 (H,W) 和通道卷积 (C)，用这三个卷积叠加后生成 attention map 再用于图片中。

而 triplet 也是一种获取 attention map 的结构，该结构主要解决的问题是将空间和通道特征进行交互，获取到多维度的 attention map。注意到该结构使用的是 7x7 卷积核来获取局部特征，这里可以对应到 LKN 的局部空间这一维度，而没有考虑 long range 的元素之间的关联。并且 triplet 选择首先将其中一维进行压缩后再获取另外两个维度的关联信息，这个压缩过程仅仅是将某一维的元素进行求平均和最大值，可表征的信息过于局限。这里可以考虑使用 1x1 的卷积核获取被压缩维度的特征值。

改进后的 triplet 结构如图15所示。

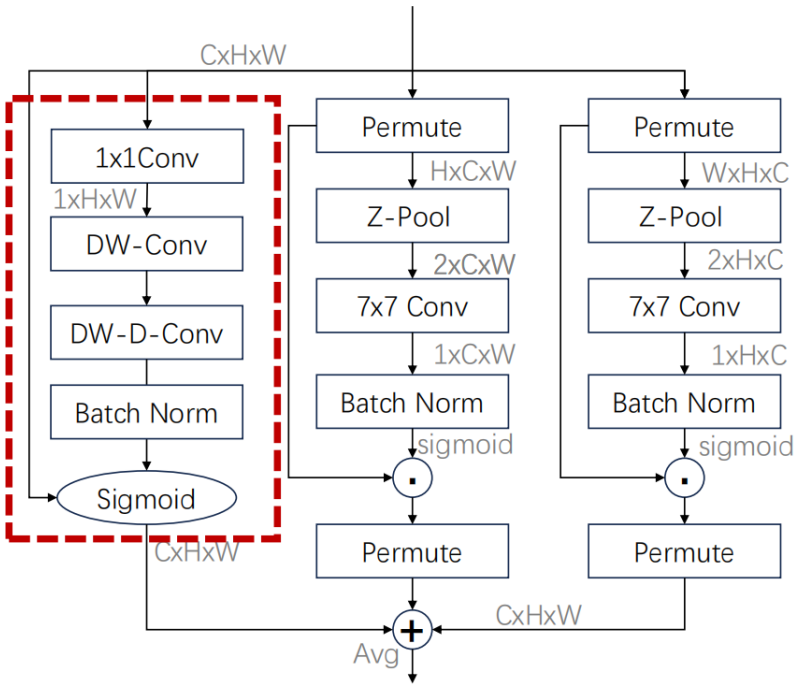


图 15: tripletLKN

4. 激活函数

该部分主要针对 7x7Conv 和 BatchNorm 后的激活函数进行多种尝试。

从图16可以看到，位于 7x7Conv 和 BatchNorm 层之间存在一个激活函数，我尝试使用以下激活函数替代原激活函数，测试这一改动对最终结果的影响。替换激活函数的代码如下，结果详见 Triplet 消融实验部分：

替换激活函数

```
        if activation=="relu":
            self.relu=nn.ReLU()
        elif activation=="sigmoid":
            self.relu=nn.Sigmoid()
        elif activation=="gelu":
            self.relu=nn.GELU()
        elif activation=="leaky-relu":
            self.relu=nn.LeakyReLU()
        elif activation=="celu":
            self.relu=nn.CELU()
        elif activation=="elu":
            self.relu=nn.ELU()
        else:
            self.relu=None
```

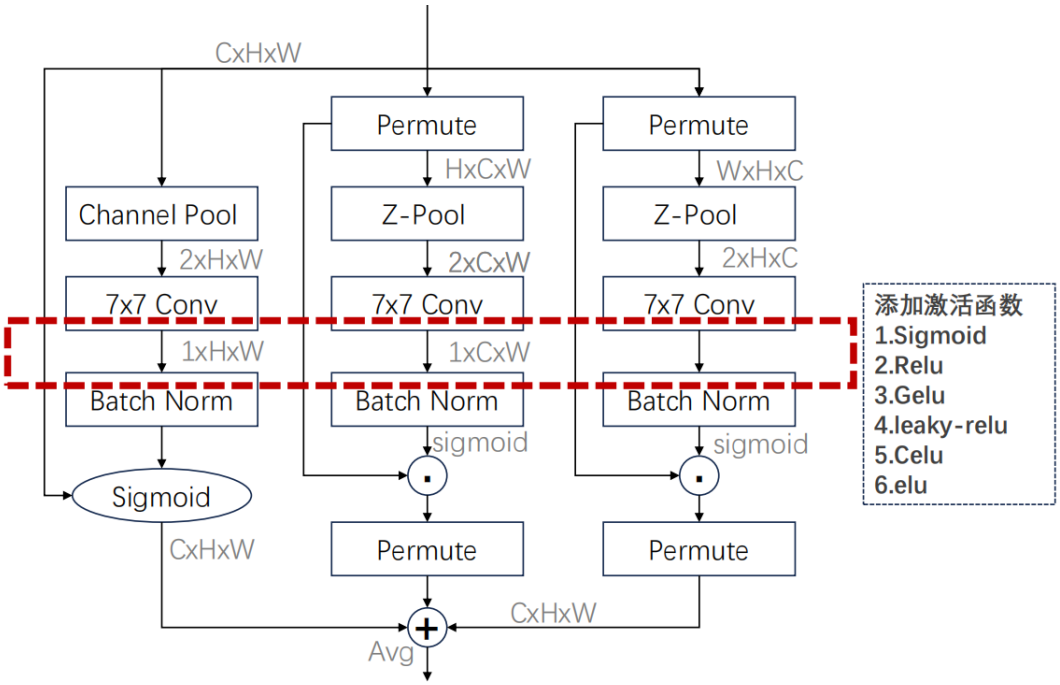


图 16: 7x7Conv→ BatchNorm

还可以通过修改图17中的 sigmoid 激活层，测试不同激活函数对最终实验结果的影响。使用的激活函数如下：

- tanh 函数
- softmax 函数

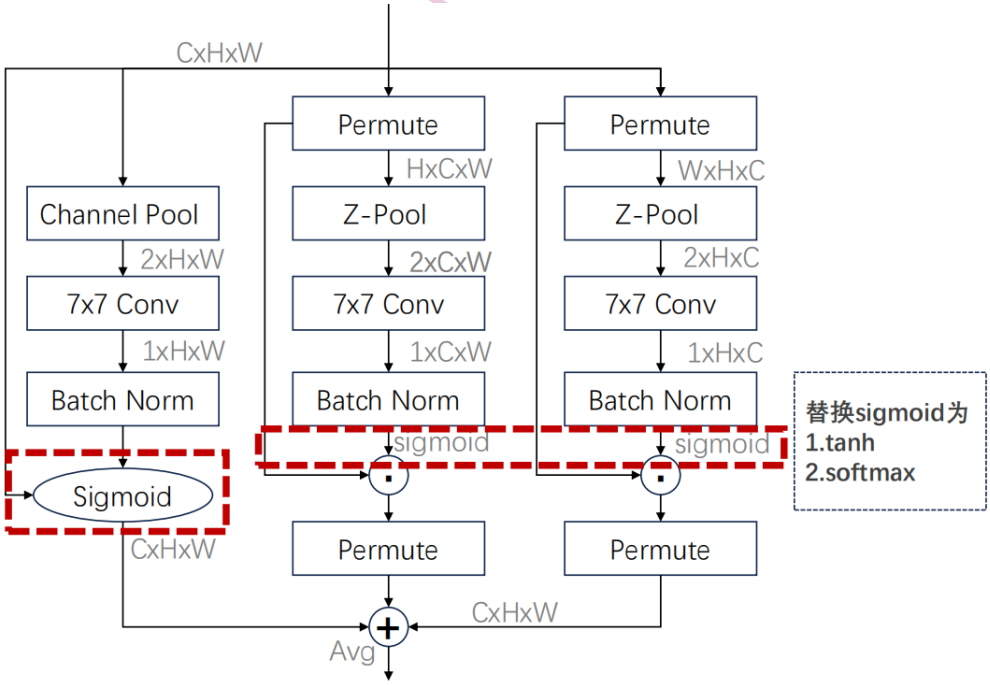


图 17: sigmoid 激活函数修改方案

(二) VAN

对于 VAN，主要有以下几种改进思路：

1. 大核卷积

作者在 future work 中提出，应用大核是一个潜在的改进点。大核卷积（或称为宽卷积）有以下的优势/特点：

1. 有效感受野更广：大核卷积的核尺寸较大，可以覆盖更多的输入区域。这使得网络可以更好地捕捉到输入数据的全局特征和上下文信息。相比之下，小核卷积只能捕捉局部特征。
2. 参数共享效果更好：卷积神经网络中的参数共享是一种重要的特性，可以显著减少网络的参数量，降低过拟合的风险。在大核卷积中，由于核尺寸较大，可以在更大的感受野范围内共享参数，进一步增强了参数共享的效果。
3. 强调全局特征：大核卷积通过其更大的感受野，更好地捕捉到了输入数据的全局特征。这对于一些任务，如图像分类和目标检测中，对整体结构和布局的理解非常重要。
4. 避免网络结构过深以至于难以优化。

默认情况下，LKA 采用 5×5 深度方向的卷积、膨胀率为 3 的 7×7 深度方向的卷积和 1×1 卷积来近似 21×21 卷积。我们尝试将其改为更大的卷积核。受 RepLKNet 的启发，论文作者使用 31×31 的卷积达到了良好的效果，因此我将 LKA 也改为 31×31 的卷积，即 7×7 深度方向的卷积、膨胀率为 3 的 11×11 深度方向的卷积和 1×1 卷积来近似。代码如下所示：

大核卷积

```python
class LKA(nn.Module):
    def __init__(self, dim):
        super().__init__()
        self.conv0 = nn.Conv2d(dim, dim, 7, padding=3, groups=dim)
        self.conv_spatial = nn.Conv2d(dim, dim, 11, stride=1, padding=25,
            groups=dim, dilation=5)
        self.conv1 = nn.Conv2d(dim, dim, 1)

    def forward(self, x):
        u = x.clone()
        attn = self.conv0(x)
        attn = self.conv_spatial(attn)
        attn = self.conv1(attn)
        return u * attn
```

2. 多尺度分支

多尺度分支是一个可能的改进点，它拥有以下若干优势：

1. 多尺度信息融合：通过使用多尺度分支，可以将来自不同尺度的特征进行融合，从而获得更丰富和全面的特征表示。不同尺度的特征包含了不同层次的信息，低层次特征包含细节和纹理等局部信息，而高层次特征则包含更抽象和语义的全局信息。通过融合这些不同尺度的信息，可以提升模型对图像的理解能力，使其更好地捕捉物体的结构、上下文和关系。

2. 尺度不变性和鲁棒性：多尺度分支可以提高模型对于尺度变化的鲁棒性。在图像处理任务中，物体的尺度可能会因为距离、角度或其他因素的变化而发生变化。通过使用多尺度分支，模型可以在不同尺度下进行特征提取和处理，从而提高模型对尺度变化的适应性。这使得模型能够更好地处理不同尺度的物体，提高目标检测、分割等任务的性能。

3. 细粒度特征捕捉：多尺度分支可以捕捉到不同尺度的细粒度特征。低层次的特征图通常能够更好地捕捉到图像的细节和局部特征，而高层次的特征图则更关注整体结构和语义信息。通过使用多尺度分支，可以同时利用这些细粒度特征和高层次语义特征，提高模型对图像的理解能力和分类准确性。

4. 语义分割和图像分割：多尺度分支在语义分割和图像分割等任务中特别有用。这些任务需要对图像中的不同区域进行像素级别的分类或分割，而不同区域的尺度可能存在差异。通过使用多尺度分支，可以提供更全面的上下文信息和物体形状信息，从而提高分割的准确性和鲁棒性。

原始的 LKA 的 kernel_size 设置为 5，而我将一个 LKA 改为 kernel_size 分别为 5/7/9 的三个 LKA，最后将他们按照第一维连接并返回，如图18所示；

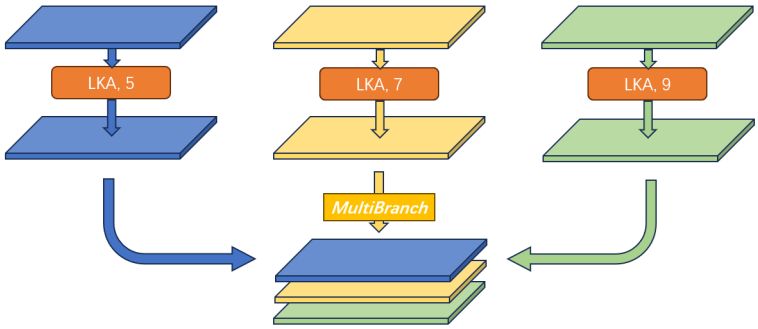


图 18: VAN MultiBranch

多尺度分支

```python
class MultiScaleBranch(nn.Module):
    def __init__(self, dim):
        super().__init__()
        self.LKA5 = LKA(dim, kernel_size=5)
        self.LKA7 = LKA(dim, kernel_size=7)
        self.LKA9 = LKA(dim, kernel_size=9)

    def forward(self, x):
        # u = x.clone()
        out_5 = self.LKA5(x)
        out_7 = self.LKA7(x)
        out_9 = self.LKA9(x)
        return torch.cat([out_5, out_7, out_9], dim=1)
```

多尺度分支将直接代替原来在 Attention 模块中的 LKA，即 self.spatial_gating_unit = LKA(d_model) 将改为 self.spatial_gating_unit = MultiScaleBranch(d_model)。

3. VAN 与 Res2Net 结合

Res2Net [5] 通过增加块内的感受野，而不是一层一层地捕获图像中更细粒度的不同级别的尺度，从而提高 CNN 检测和管理图像中物体的能力。在 Res2Net 块中，一个单独残差块中的分层的残差连接使感受野在更细粒度级别上的变化能够捕获细节和全局特性。实验结果表明，Res2Net 模块可以与网络设计相结合，进一步提高网络性能。因此，我希望能够尝试将 VAN 与 Res2Net 进行结合。

VAN 的整体结构中有一部分是 MLP，而在 MLP 中有一个 DWConv；我则将 DWConv 换成了 Res2Net 中的残差结构，如图19所示：

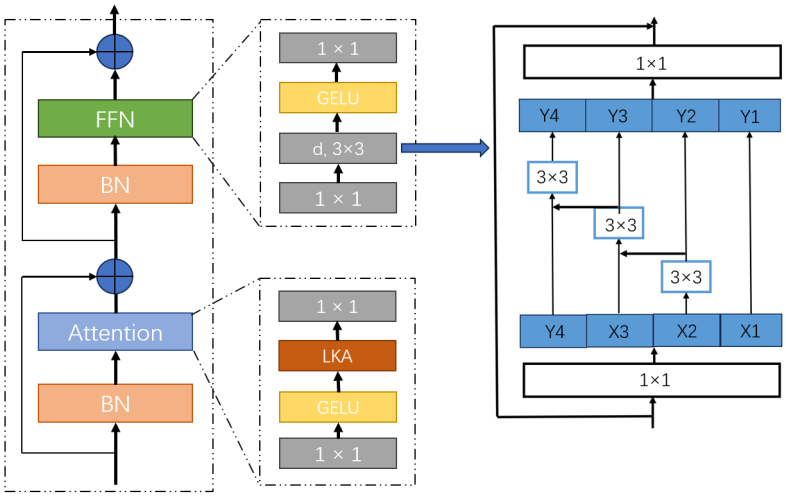


图 19: VAN 与 Res2Net 结合

需要注意的是，Bottleneck 类中的 out += identity 时，out 的特征映射数量是 planes * 4，而 identity 的特征映射数量是原始输入 x 的特征映射数量，即 inplanes。因此我们需要加入一个下采样操作：

downsample

```
        if inplanes != planes * 4:
            self.downsample = nn.Sequential(
                nn.Conv2d(inplanes, planes * 4, kernel_size=1, stride=1, bias
                    =False),
                nn.BatchNorm2d(planes * 4),
            )
        else:
            self.downsample = downsample
```

而在 Mlp 模块中，有一个 fc1 卷积层，它把输入特征从 in_features 映射到 hidden_features。然后紧跟着一个 Bottleneck 模块，它将特征数量扩大到 hidden_features*4。最后接一个 fc2 卷积层，试图将特征映射回 out_features，但是这里的问题在于，fc2 的输入特征数量并非是 hidden_features，而是 hidden_features*4，即 Bottleneck 模块的输出特征数量。

因此我们需要给予修正，即在创建 fc2 时，将输入特征数量设置为 hidden_features*4：self.fc2 = nn.Conv2d(hidden_features * 4, out_features, 1)，即可解决问题。

4. VAN 和 RepLKNet 结合

RepLKNet [2] 是大核卷积的又一范例。作者使用 depth-wise 超大卷积加 shortcut，并用小卷积核做重参数化（即结构重参数化方法论），在 ImageNet 上达到了 87.8%，超越了 VAN。同时作者认为，大核能克服传统的深层小核的 CNN 的固有缺陷。作者丁霄汉博士在知乎中这样写道：

“我们曾经相信大 kernel 可以用若干小 kernel 来替换，比如一个 7x7 可以换成三个 3x3，这样速度更快（3x3x3< 1x7x7），效果更好（更深，非线性更多）。有的同学会想到，虽然深层小 kernel 的堆叠容易产生优化问题，但这个问题已经被 ResNet 解决了（ResNet-152 有 50 层 3x3 卷积），那么这种做法还有什么缺陷呢？——ResNet 解决这个问题的代价是，模型即便理论上的最大感受野很大，实质上的有效深度其实并不深，所以有效感受野并不大。这也可能是传统 CNN 虽然在 ImageNet 上跟 Transformer 差不多，但在下游任务上普遍不如 Transformer 的原因。也就是说，ResNet 实质上帮助我们回避了「深层模型难以优化」的问题，而并没有真正解决它。既然深而 kernel 小的模型有这样的本质问题，浅而 kernel 大的设计范式效果会如何呢?”

作者认为，大卷积核架构有以下的优势：

- 相比直接加深，大卷积核可以更高效的提升有效感受野:

  根据有效感受野理论，有效感受野的尺寸正比于卷积核尺寸，跟卷积核的层数开根号成正比，就是说通过添加深度的方式提高感受野，不如直接把卷积核拉大对感受野的提升更为有效。

- 卷积核可以部分回避模型深度增加带来的优化难题:

  ResNet 看起来可以做的很深，或者可以达到成百上千层，但是他的有效深度并不深，它的很多信号是 shortcut 层过去的，它并没有增加它的有效深度，如图，把 ResNet 从 101 层提高到 152 层，虽然深度提升了很多，但是他的有效感受野大小其实基本不变，但是如果把卷积核尺寸增大，从 13 增加到 31，有效感受野扩大的趋势就非常显著。

- 大卷积核对 FCN-based（全卷积）的视觉下游任务提升更明显。

秉持这一想法，我们尝试将 RepLKNet 与 VAN 结合起来，使用 RepLKBlock 来直接替换 LKA，并查看它的性能。实验证明，未使用重参数化的 RepLKBlock 相较于原始的 LKA，Top-1 Acc 和 Top-5 Acc 分别提升了 2.87% 和 2.46%。而 RepLK 使用的是 31*31 的卷积核，其实验性能稍强于 31*31 的 LKA，但准确率相差不到 1%，这也验证了大核卷积的功效。

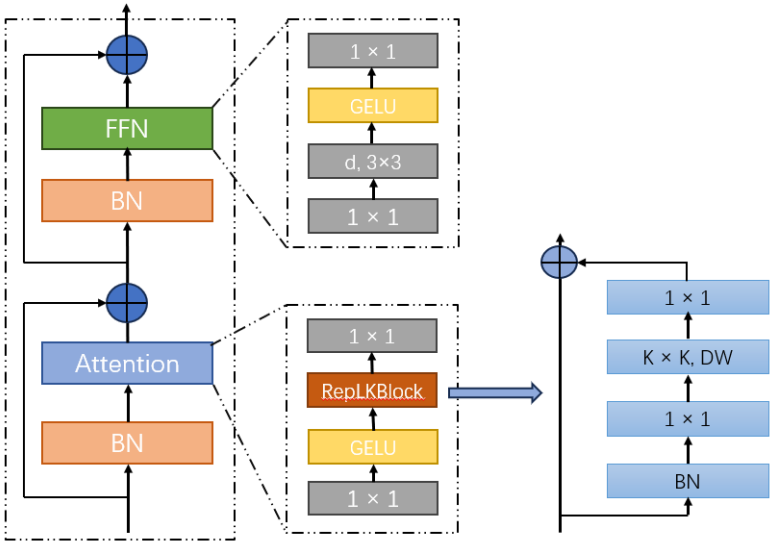


图 20: VAN 与 RepLKBlock 结合

5. 增加批归一化

在原始代码的 LKA 模块中，并没有使用批归一化。但是批归一化有以下优点：

1. 加速训练收敛：批归一化可以加速神经网络的训练收敛过程。在训练过程中，神经网络的每一层的输入分布会随着参数的更新而发生变化，这被称为“Internal Covariate Shift”。批归一化通过将每个批次的输入进行归一化，使得网络中每一层的输入分布保持稳定。这样可以减少梯度消失和梯度爆炸的问题，使得网络更容易训练。

2. 改善激活函数的表达能力：批归一化对于激活函数的表达能力有所改善。由于归一化操作使得输入的分布在 0 附近，激活函数的输入更容易落在激活函数的敏感区域，增强了激活函数的非线性特性，使得网络能够更好地拟合复杂的函数。

3. 提高模型的泛化能力：批归一化有助于减少模型的过拟合风险。通过对每个批次的输入进行归一化，批归一化引入了一些随机性，类似于一种正则化。这样可以降低模型对特定批次的依赖性，增加模型的泛化能力，提高模型在未见过的数据上的表现。

4. 具有正则化效果：批归一化层在一定程度上具有正则化的效果，可以减少模型的参数量和复杂度。通过归一化操作，批归一化减少了模型中每个参数的依赖性，降低了模型的自由度，有助于控制过拟合。

因此，我改造了 LKA 的结构，在三次卷积之后分别增加了批归一化：

增加 BN

```
class LKA(nn.Module):
    def __init__(self, dim):
        super().__init__()
        self.conv0 = nn.Conv2d(dim, dim, 5, padding=2, groups=dim)
        self.bn0 = nn.BatchNorm2d(dim)
        self.conv_spatial = nn.Conv2d(dim, dim, 7, stride=1, padding=9,
            groups=dim, dilation=3)
```

```
        self.bn_spatial = nn.BatchNorm2d(dim)
        self.conv1 = nn.Conv2d(dim, dim, 1)
        self.bn1 = nn.BatchNorm2d(dim)

    def forward(self, x):
        u = x.clone()
        attn = self.conv0(x)
        attn = self.bn0(attn)
        attn = self.conv_spatial(attn)
        attn = self.bn_spatial(attn)
        attn = self.conv1(attn)
        attn = self.bn1(attn)

        return u * attn
```

6. 增加非线性激活函数

在原始的 LKA 代码中，也没有使用非线性激活函数。增加激活函数可能有以下潜在的优势：

1. 提高模型的非线性逼近能力：深度神经网络具有多层堆叠的结构，通过使用非线性激活函数，每一层可以对输入数据进行非线性变换。这使得网络能够逼近更加复杂的函数，通过多个非线性变换的组合，可以构建出更复杂和抽象的特征表示，提高模型的拟合能力和表达能力。
2. 增加模型的表达能力：非线性激活函数可以引入更复杂的非线性变换，从而增加模型的表达能力。通过适当选择非线性激活函数，可以使网络具有更强的非线性逼近能力，能够处理更复杂的输入数据和学习更复杂的函数映射关系。

因此我在 LKA 的前两个卷积之后分别增加了非线性激活函数，代码如下所示。具体不同的激活函数有什么不同的激活效果，这一部分见消融实验。

增加激活函数

```
class LKA(nn.Module):
    def __init__(self, dim):
        super().__init__()
        self.conv0 = nn.Conv2d(dim, dim, 5, padding=2, groups=dim)
        self.conv_spatial = nn.Conv2d(dim, dim, 7, stride=1, padding=9,
            groups=dim, dilation=3)
        self.conv1 = nn.Conv2d(dim, dim, 1)

    def forward(self, x):
        u = x.clone()
        attn = self.conv0(x)
        attn = F.celu(attn)
        attn = self.conv_spatial(attn)
        attn = F.celu(attn)
        attn = self.conv1(attn)

        return u * attn
```

(三) Res2net

首先明确，本节中所有提到的 Res2Net 网络均指 Res2Net50 网络。

1. 激活函数的替换

原始 Res2Net Module 中使用的激活函数为 ReLU 函数，其优点在于运算速度快，可以有效缓解过拟合，其缺点主要在于 Dead ReLU 问题，即如果某个神经元的输入小于零，那么该神经元的输出将永远为零，这称为死亡神经元，在训练过程中，神经元的权重可能调整到非常小的值，导致这些神经元几乎不再被激活，从而丧失了学习能力，可以使用以下代码计算经过激活函数 Relu 后死亡的神经元个数：

Listing:

```
self.relucounternone += (out == 0).sum().item()
```

经过统计，仅计算运行 1 个 epoch，测试集第 30-40 个批次，Res2Net50 网络结构经过最后一个 Res2Net Module 中经过残差连接的激活函数后死亡的神经元个数之和，就达到了惊人的 4138107!

因此 Res2Net Module 中存在严重的 Dead ReLU 问题，考虑替换激活函数，CELu、ELU、GELU、Leaky ReLU 都可以在一定程度上缓解 Dead ReLU 问题：

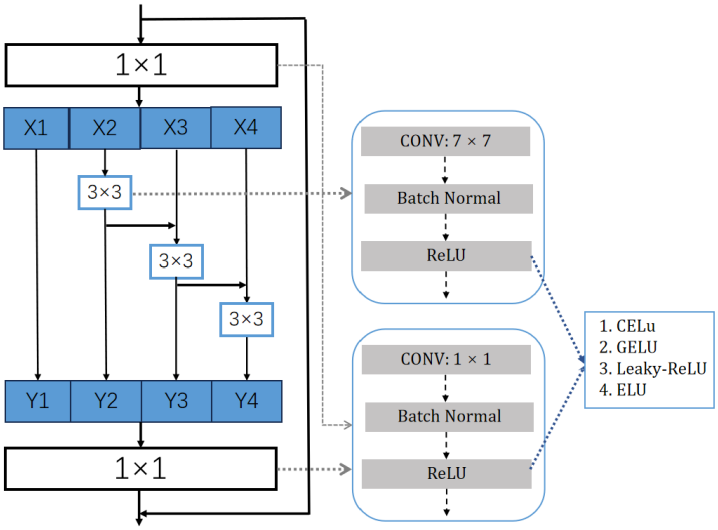


图 21: Res2Net Module 激活函数替换

替换激活函数的代码如下，结果详见 Res2Net 消融实验部分：

替换激活函数

```
        #激活函数
        if activation == 'relu':
            self.activation = nn.ReLU(inplace=True)
        elif activation == 'celu':
            self.activation = nn.CELU(inplace=True)
        elif activation == 'gelu':
            self.activation = nn.GELU()
        elif activation == 'elu':
```

```
            self.activation = nn.ELU(inplace=True)
        elif activation == 'leaky-relu':
            self.activation = nn.LeakyReLU(inplace=True)
        else:
            raise NotImplementedError(f"Activation not implemented!")
```

2. 增加注意力机制

在 Res2Net 网络中增加注意力机制此处主要有三种思路:

1. **方案一**的想法来源是参考原始 Res2Net 论文 [3] 中增加注意力机制的方法，在 Res2Net Module 结构中第二个 1x1 卷积层后引入注意力机制 SE，其网络结构如下所示:

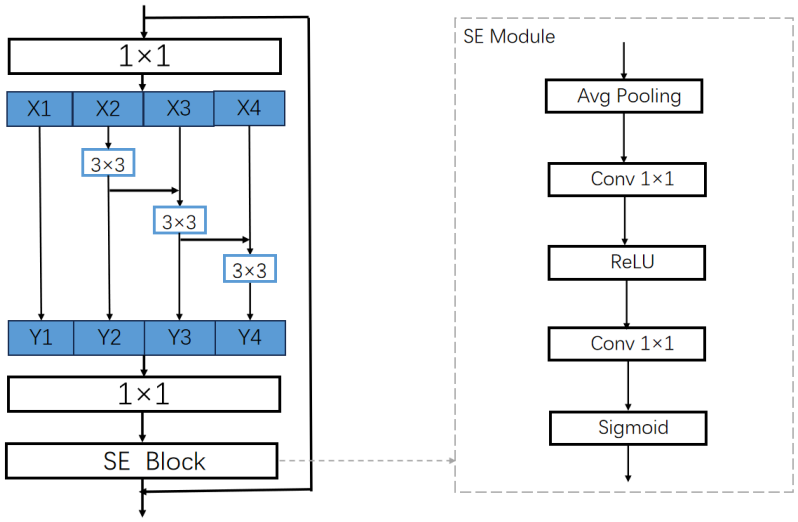


图 22: 在 1x1 卷积层后添加 SE 机制

通俗的讲这样可以让网络知道去哪里关注并进一步关注目标对象，提高神经网络的性能。因此考虑到之前论文复现时使用的 Triplet Attention 机制，一个简单的想法是可以在第二个 1x1 卷积层后引入注意力机制 Triplet Attention，其网络结构图如下:

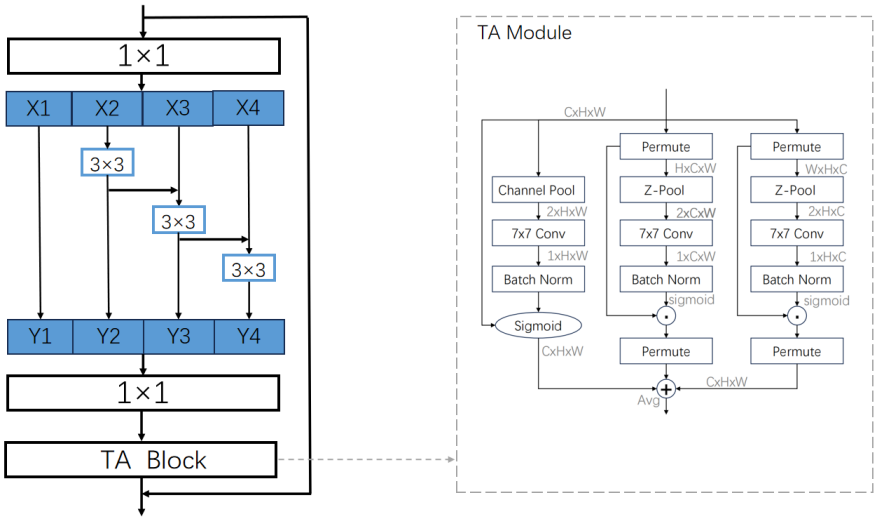


图 23: 在 1x1 卷积层后添加 TA 机制

2. **方案**二的想法来源是 Res2Net Module 结构已经将原始网络分成了 4 个多尺度子网络，那么对于每一个子网络，我们都可以引入 SE 或 Triplet Attention 注意力机制，其网络结构图如下:

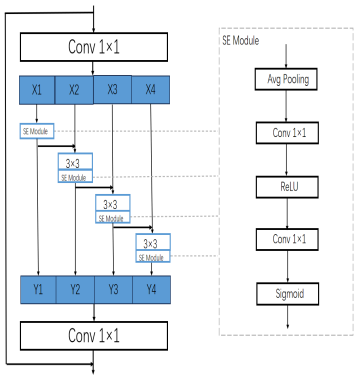


图 24: 子网络内引入 SE 机制

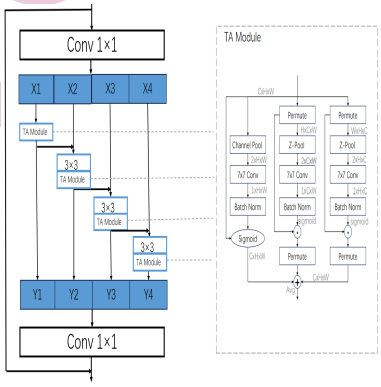


图 25: 子网络内引入 TA 机制

图 26: 子网络内增加注意力机制

3. **方案**三的思路是将方案一与方案二融合，同时在第二个 1x1 卷积层后和子网络内引入注意力机制，由于之前提到了 SE 或 Triplet Attention 两种注意力机制，因此此处有四个网络结构图，分别记为 Res2Net_SS,Res2Net_ST,Res2Net_TS,Res2Net_TT, 以 Res2Net_ST 为例，其名称中的 S 代表子网络内引入 SE 注意力机制，T 代表子网络内引入 Triplet Attention 注意力机制, 四个网络结构图如下所示:

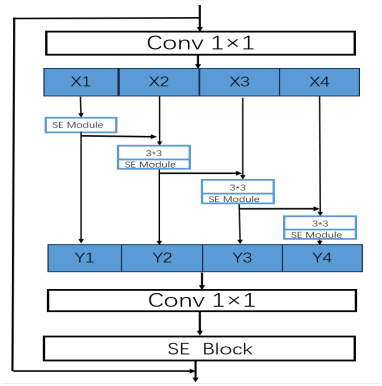


图 27: Res2Net_SS

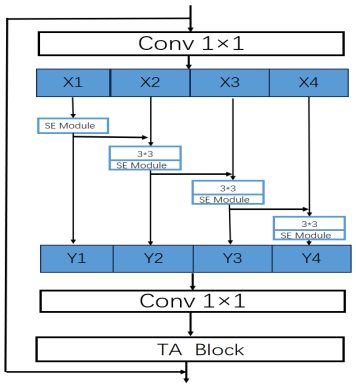


图 28: Res2Net_ST

图 29: 同时在 1x1 卷积层后及子网络内增加注意力机制

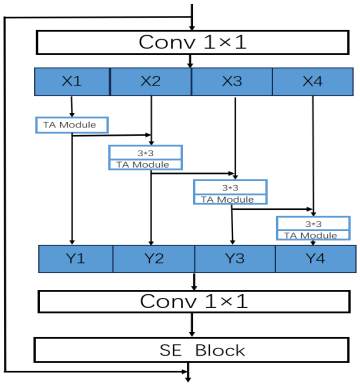


图 30: Res2Net_TS

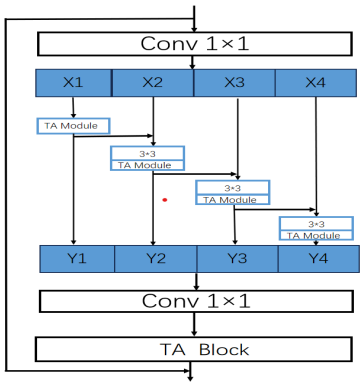


图 31: Res2Net_TT

图 32: 同时在 1x1 卷积层后及子网络内增加注意力机制

3. 大核卷积的替换

Res2Net 网络的实质是使用 Res2Net Module 结构的残差块替换原始 ResNet50 中的残差块，因此可以考虑对于 Res2Net 网络应用 ResNet 中的改进，本小节的优化以及下一小节中的修改下采样方式的想法均来自于此。参考 ResNet-C 网络结构 [6]，将 Res2Net50 中第一个 7x7 的大核卷积使用 3 个 3x3 的小卷积核替代实现，大核卷积替换的网络结构图如下：

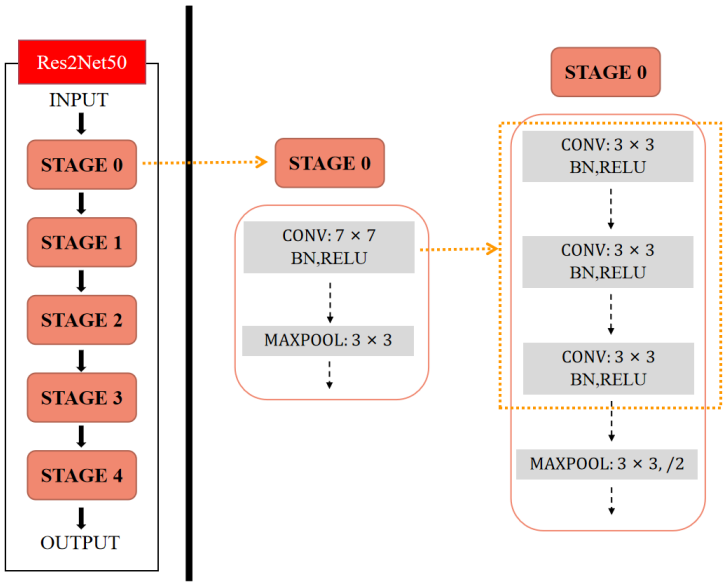


图 33: 大核卷积的替换

这么做的好处主要有以下几点:

1. 使用 3 个 3x3 的小卷积核替代之后的感受野和一个 7x7 的大卷积核卷积的感受野是相同的
2. 使用 3 个 3x3 的小卷积核替代之后的参数计算量要小于一个 7x7 的大卷积核卷积的参数计算量，可以假设我们输入的图像为 HxWxC，我们使用 M 个卷积核，步长为 1，则参数的个数为:

   7x7filter: M·(7x7·C) = 49M·C

   3 个 3x3filter: 3·M·(3x3·C) = 27M·C
3. 网络层数加深，增加了网络的表达能力，可以防止过拟合的情况。同时多层的 relu 激活，增加了卷积过程中的非线性。

大核卷积替换的代码实现如下:

替换大核卷积

```
self.conv1 = nn.Sequential(
            nn.Conv2d(3, self.inplanes, kernel_size=3, stride=2, padding=1,
                bias=False),
            nn.BatchNorm2d(self.inplanes),
            nn.ReLU(inplace=True),
            nn.Conv2d(self.inplanes, self.inplanes, kernel_size=3, stride=1,
                padding=1, bias=False),
            nn.BatchNorm2d(self.inplanes),
            nn.ReLU(inplace=True),
            nn.Conv2d(self.inplanes, self.inplanes, kernel_size=3, stride=1,
                padding=1, bias=False),
            nn.BatchNorm2d(self.inplanes),
            nn.ReLU(inplace=True)
        )
```

4. 修改下采样方式

在原先的 ResNet50 网络结构中, downsample 操作是由一个 conv_1x1, stride=2; conv_3x3; conv_1x1 构成的 pathA 和 conv_1x1，Stride=2 构成的 pathB 组成双路，然后 Add 操作而成。这么做的问题在于 pathA 和 pathB 各有 3/4 的信息被忽略掉了。因此修改下采样方式，参考 ResNet-D 网络结构 [6]

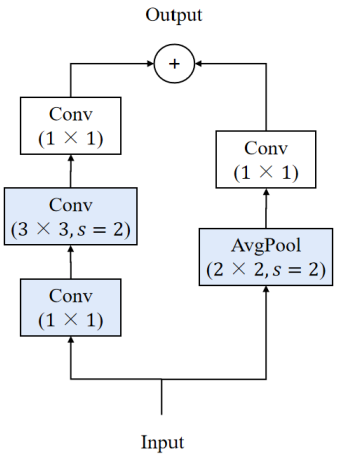


图 34: 修改下采样方式

对于 pathA 修改前两个卷积块的步长，将 stride=2 移到 conv_3x3 卷积之中，从而减少信息的丢失。同样的对于 pathB，在卷积之前添加一个 2×2 的平均池化层，步长为 2 并将卷积的步长改为 1，从而也减少了信息的丢失。

四、 实验

我们的实验在 CIFAR100 数据集上进行。

(一) 实验设置

本节描述了在进行实验时使用的硬件、软件、数据集、学习率设置等详细信息。

1. 硬件环境

   GPU: NVIDIA GeForce RTX 3060Ti

2. 软件环境

   深度学习框架: PyTorch 2.0.0

   CUDA 版本: NVIDIA CUDA 12.1.98 driver

   Python 版本: 3.10.0

3. 数据集

   实验使用的数据集为 CIFAR-100，它是一个广泛用于图像分类任务的数据。CIFAR-100 数据集包含了 100 个不同的类别，每个类别包含 600 张大小为 32x32 的彩色图像。数据集中的图像被分为训练集和测试集，其中训练集包含了 50,000 张图像，测试集包含了 10,000 张图像。

4. 学习率

实验中按照如下代码设置自适应的学习率:

设置学习率

```python
def adjust_learning_rate(optimizer, epoch):
    """Sets the learning rate to the initial LR decayed by 10 every 30 epochs
    """
    lr = args.lr * (0.1 ** (epoch // 30))
    for param_group in optimizer.param_groups:
        param_group["lr"] = lr
    # wandb.log({"lr": lr})
```

将学习率设置为每 30 个 epoch 衰减 10 倍的初始学习率，图像如下所示，注意此时 1 epoch = 2 steps:

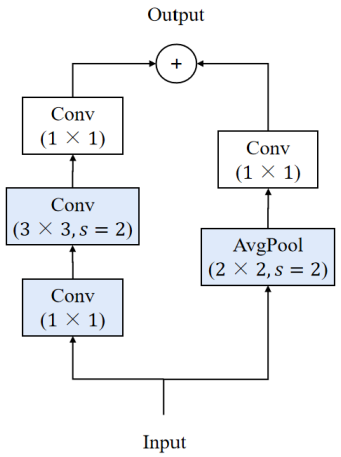
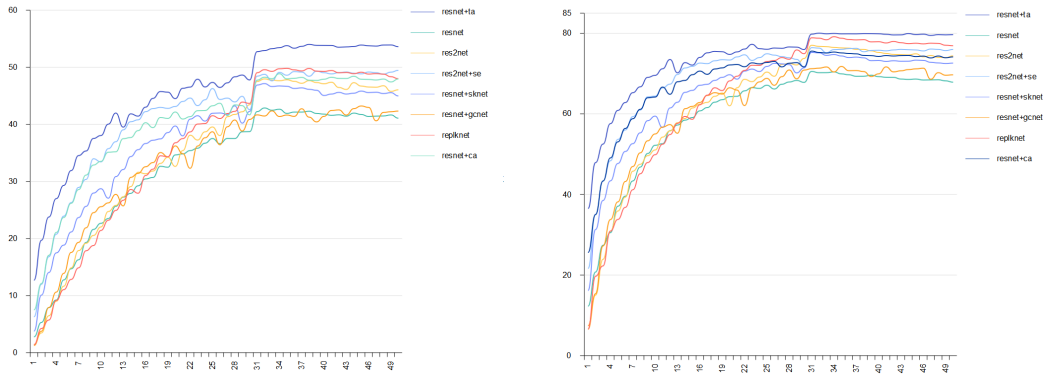


图 35: 学习率图像

(二) 实验结果

1. 基础实验

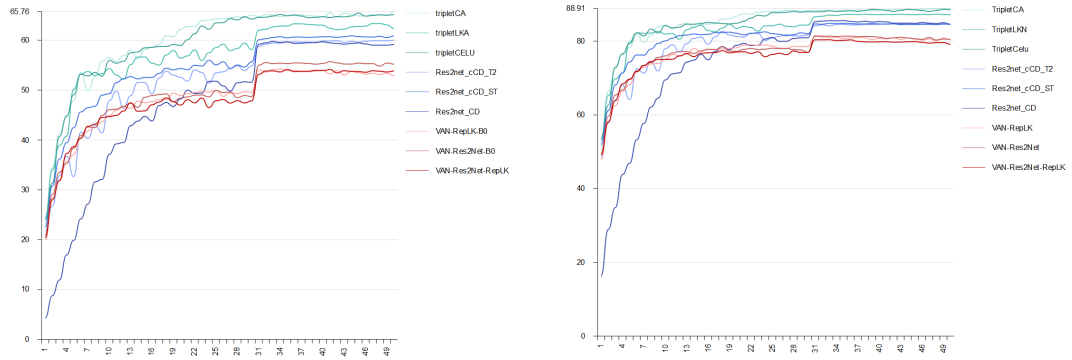


(a) Top-1 Acc 比较 (b) Top-5 Acc 比较

图 36: 基础模型的 Acc 比较

基于 baseline 曲线图36，可以看到 resnet 的 Top-1 Acc 处于 42% 范围，Top-5 Acc 处于 75% 范围。通过在 resnet 架构中加入不同的注意力机制，可以获得 Top-1 和 Top5 不同幅度的提升。其中添加 Triplet 注意力机制获得的提升最高，Top-1 Acc 可达到 54%，Top-5 可达到 80%。

2. 改进结果



(a) Top-1 Acc 比较 (b) Top-5 Acc 比较

图 37: 改进模型的 Acc 比较

在基础网络模型上进行的改进结果如图37所示。针对 Triplet 部分的改进结果，TripletCA 获得了最高的 Top-1 Acc 和最高的 Top-5 Acc 结果，其中 Top-1 Acc 可达到 65.76%，Top-5 Acc 可达到 88.91%。针对 VAN 部分的改进结果，VAN-RepLK 的提升最为显著，直接证明了大核卷积能够更高效的提升有效感受野，从而提升分类模型的准确度。针对 res2net 部分，效果最好的模型是 Res2Net_cC，它的 Top-5 的 Accuracy 为 84.85%，相比原始 rest2net 模型有较高的性能提升。

(三) 消融实验

1. Triplet Attention

Triplet 注意力机制的改进方案已在改进部分给出，该部分旨在对比不同改进方向的消融实验结果，获得性能良好的改进模型。接下来，将分别对 z-pool 的改进部分、TripletCA 和 TRipletLKN 的改进部分以及激活函数的消融实验结果进行分析。

首先是 z-pool 部分的改进实验，在这部分使用 {avg,max,median}、{l1,max}、{l2,max} 三组改进方式，对 z-pool 进行改进尝试并测试改进后的模型。以下数据均基于 resnet50 的骨干架构，将 triplet 改进前后的注意力机制用于该网络架构。z-pool 实验数据如表1所示。

Method	MACs (G)	Params. (M)	Top-1 Acc (%)	Top-5 Acc (%)
avg,max	4.17	22.52	64.2	87.23
avg,max,median	4.17	22.521	64.02	**87.54**
l1,max	4.18	22.52	64.29	87.36
l2,max	4.19	22.52	**65.24**	87.26

表 1: 针对 z-pool 部分的改进模型运行结果

在基于 z-pool 的改进部分，l2,max 的改进方案获得最高的 Top-1 Acc，为 65.24%，相比原始 z-pool 方案上涨 1.04%；Top-5 Acc 部分的改进效果并不明显，均维持在 87.2%-87.6% 的区间。

然后，为了测试基于 Triplet 的 TripletCA 和 TripletLKN 注意力机制的模型性能，基于 resnet50 架构，将 Triplet、CA、TripletCA 和 TripletLKN 注意力机制插入相同的位置作为消融实验的一部分，测试不同注意力机制 Top-1 和 Top-5 的 Acc。实验结果如表2所示。

Method	MACs (G)	Params. (M)	Top-1 Acc (%)	Top-5 Acc (%)
Triplet	4.17	22.52	64.6	87.23
CA	4.90	22.94	63.09	87.21
TripletCA	4.83	22.53	**65.71**	**88.55**
TripletLKN	5.01	23.49	63.02	87.19

表 2: 基于 Triplet 提出的新模型运行结果

该测试结果显示，TripletCA 注意力机制相比 Triplet、CA 注意力机制，在 Top-1 和 Top-5 的 Acc 中均有较好的提升。TripletCA 的 Top-1 Acc 可达到 65.71%，同比增长 1.11 个百分点；Top-5 Acc 可达到 88.55%，同比增长 1.32 个百分点。而与此相对的是，TripletLKN 注意力机制不仅没有较 Triplet 和 CA 有一定的提高，反而 Top-1 和 Top-5 均处于最低。因此，TripletCA 注意力机制能够取得更好实验结果，参数量仅增加 1.16%。

针对 7x7Conv 之间的 relu 激活函数的消融实验与 batchNorm 后的 sigmoid 激活函数的消融实验结果在表3中显示。

Method	MACs (G)	Params. (M)	Top-1 Acc (%)	Top-5 Acc (%)
7x7Conv-sigmoid	4.17	22.52	64.16	87.23
7x7Conv-relu	4.17	22.52	64.36	87.44
7x7Conv-gelu	4.17	22.52	64.11	87.51
7x7Conv-celu	4.17	22.52	64.86	**88.35**
7x7Conv-elu	4.17	22.52	**65.72**	87.89
BatchNorm-sigmoid	4.17	22.52	**63.02**	**87.44**
BatchNorm-tanh	4.17	22.52	61.32	85.83
BatchNorm-softmax	4.17	22.52	44.86	73.47

表 3: Triplet 激活函数替换的运行结果

通过对比可以发现，将 Conv7x7 部分的激活函数替换为 celu 时模型能获得最高的 Top-5 Acc:88.35%；将 Conv7x7 部分的激活函数替换为 elu 时，模型能获得最高的 Top-1 Acc:65.72%。并且 celu 和 elu 激活函数的 Top-1 和 Top-5 精确度均位于较高的水平。相反，将激活函数替换为 gelu、sigmoid 并不能提升模型的性能，模型性能反而相对下降。

将 BatchNorm 部分的激活函数替换后，获得的训练结果均相比原始的 sigmoid 激活函数有所下降。其中，softmax 下降幅度最大，如果将 BatchNorm 后的激活函数替换成 softmax 将使 Top-1 准确度下降 18.16%，使 Top-5 准确度下降 13.97%。

综上所述，在 Triplet 部分我们进行了三个方向的融合实验，分别用于测试更改 z-pool 策略、提出新注意力机制和更改不同位置的激活函数后的实验结果。实验结果表明，将 z-pool 替

换为 { l2,max} 会使 Top-1 的 Acc 有 1.04% 的提升；使用 TripletCA 新注意力机制可以同时提升 Top-1 和 Top-5；使用 celu 和 elu 替换 conv7x7 中的激活函数，可以在不额外增加参数的前提下，提高模型的准确度。

2. VAN

VAN 的总体实验结果如表4所示。由于算力资源不足，我们只运行了基于 VAN-B0 的改进模型和部分基于 VAN-B1 的改进模型。剩下的一些基于 VAN-B1 的改进模型由于显存不足等问题运行失败。

Method	MACs (G)	Params. (M)	Top-1 Acc (%)	Top-5 Acc (%)
VAN-B0	0.87	4.103	51.79	78.68
VAN-LK-B0	0.931	4.256	52.8	80.3
VAN-MB-B0	1.173	5.46	52.49	79.55
VAN-BN-B0	0.878	13.873	47.45	74.07
VAN-Act-B0	0.87	4.103	53.15	79.95
VAN-RepLK-B0	1.491	5.803	54.25	**81.55**
VAN-Res2Net-B0	20.409	62.459	**55.47**	81.47
VAN-Res2Net-RepLK-B0	21.05	64.204	54.13	80.36
VAN-B1	2.506	13.856	52.49	79.87
VAN-LK-B1	2.592	14.114	53.28	80.13
VAN-MB-B1	3.276	18.191	50.98	78.12
VAN-BN-B1	2.516	13.873	48.78	75.11
VAN-Act-B1	2.506	13.856	52.96	80.19
VAN-RepLK-B1	58.456	213.874	**56.11**	**82.23**

表 4: 基于 VAN 的改进模型运行结果

在基于 VAN 的改进中，对于 B0 分支，VAN-Res2Net 拥有最高的 Top-1 Acc，为 55.47%；VAN-RepLK 拥有最高的 Top-5 Acc，为 81.55%；对于 B1 分支，VAN-RepLK 同时拥有最高的 Top-1 和 Top-5 Acc；其他的改进也具有良好的表现。

从 VAN-B0 和 VAN-LK-B0 的对比可以看出，大核明显具有一些优势，Top-1 Acc 提升了 1.49 (53.28% vs. 51.79%)，Top-5 Acc 提升了 1.45 (80.13% vs. 78.68%)，提升较为明显。但是，大核也会带来参数量与运算量的上升，因此需要进行权衡。

尝试在多尺度分支内设置不同的 kernel_size，结果表5所示：

Method	Kernel Size	MACs (G)	Params. (M)	Top-1 Acc (%)	Top-5 Acc (%)
VAN-MB-B0	3, 5, 7	1.127	5.344	49.94	76.99
VAN-MB-B0	5, 7, 9	1.173	5.46	**52.49**	**79.55**
VAN-MB-B0	7, 9, 11	1.234	5.613	51.64	78.74
VAN-B0	7	0.87	4.103	51.79	78.68

表 5: VAN-MB 消融实验

从表5可以看出，相对较大的卷积核能够捕捉更广泛的图像上下文信息。这样的卷积核可以

更好地捕获边缘、纹理等细节信息，并具有较大的感受野。这有助于模型理解图像中的结构和形状，并提供更丰富的特征表示。但是，如果卷积核过大，同时也可能引入更多的模糊性，从而引起性能的降低。在不同的 Kernel Size 的多尺度分支对比中，Kernel Size 为 5, 7, 9 的模型获得了最高的准确率，但 Kernel Size 为 7, 9, 11 的模型的准确率缺不升反降。在某些任务中，过大的卷积核可能导致细节信息的丢失，从而降低了模型的准确性。

尝试更换不同的激活函数，结果如表6所示：

Method	Act. Layer	MACs (G)	Params. (M)	Top-1 Acc (%)	Top-5 Acc (%)
VAN-Act-B0	ReLU	0.87	4.103	53.02	79.78
VAN-Act-B0	CeLU	0.87	4.103	49.05	77.62
VAN-Act-B0	GeLU	0.87	4.103	**52.98**	**79.82**
VAN-Act-B0	ELU	0.87	4.103	49.05	77.62
VAN-B0	GeLU	0.87	4.103	51.79	78.68

表 6: VAN-Act 消融实验

通过对比可以发现，当激活函数为 GeLU 是模型能获得最高的 Top-1 Acc: 52.98%, 和最高的 Top-5 Acc: 79.62%；但是更换激活函数为 CeLU 的时候，模型的准确率并没有像 Res2Net 那样有明显的上升，反而下降了大约 2%。这说明不同的激活函数在不同的深度学习模型上有不同的适配性，选择激活函数时需经过精巧的设计与实验证明。

下面进行 VAN-Res2Net-RepLK-B0 的消融实验，如表7所示：

Method	MACs (G)	Params. (M)	Top-1 Acc (%)	Top-5 Acc (%)
VAN-Res2Net-RepLK-B0	21.05	64.204	54.13	80.36
w/o RepLK	20.409	62.459	**55.47**	81.47
w/o Res2Net	1.491	5.803	54.25	**81.55**
VAN-B0	0.87	4.103	51.79	78.68

表 7: VAN-Res2Net-RepLK 消融实验

我们可以发现，去掉 RepLK 模块或去掉 Res2Net 模块能获得比 VAN-Res2Net-RepLK-B0 更高的准确度。去掉 RepLK 模块时，Top-1 Acc 提高了 1.34% (55.47% vs. 54.13%)，Top-5 Acc 提高了 1.11% (81.47% vs. 80.36%)；去掉 Res2Net 模块时，Top-1 Acc 提高了 0.12% (54.24% vs. 54.13%)，Top-5 Acc 提高了 1.19% (81.55% vs. 80.36%)；可能导致这个问题的原因如下：

1. 冗余特征：Res2Net 模块和 RepLK 模块可能提取了与目标任务不相关或冗余的特征。在消融实验中，移除这些模块可能消除了冗余特征的影响，反而使得模型更专注于更有助于任务的特征，从而提高了正确率。

2. 过拟合：VAN-Res2Net-RepLK-B0 可能在训练过程中过度拟合了训练数据，而 RepLK 模块和 Res2Net 模块提供的额外复杂性增加了过拟合的风险。通过移除这些模块，模型的复杂性降低，减少了过拟合的可能性，从而提高了在测试数据上的正确率。

综上所示，我们在 VAN 的基础上提出了模型 VAN-LK, VAN-MB, VAN-Act, VAN-RepLK, VAN-Res2Net, VAN-Res2Net-RepLK，进一步发展了大核卷积，并探索了不同激活函数带来的

效果，使模型的准确率有所提升。其中，VAN-RepLK 的提升最为显著，直接证明了大核卷积能够更高效的提升有效感受野，从而提升分类模型的准确率。

3. Res2Net

在之前的模型改进小节中，已经介绍了 Res2Net 的四种改进方式，包括激活后函数的替换、增加注意力机制、大核卷积的替换、修改下采样方式，在增加注意力机制的改进方法中又分为了改进方案一即在 Res2Net Module 的第二个 1x1 卷积层后增加注意力机制；改进方案二即在 Res2Net Module 中被划分的四个多尺度子网络内分别增加注意力机制；以及改进方案三即同时在方案一方案二指定的位置添加注意力机制，综上所述，在本节中，分别进行了以下消融实验：

1. 激活函数替换的消融实验

   替换 Res2Net 中的激活函数 ReLU，希望缓解 Dead ReLU 问题，消融实验结果如下表所示，其中 Res2Net 表示使用 ReLU 激活函数的原始 Res2Net，Res2Net_c、Res2Net_g、Res2Net_l、Res2Net_e 分别代表 Res2Net Module 中的激活函数被替换成 CELu、ELU、GELU、Leaky-ReLU 之后的模型：

Method	MACs (G)	Params. (M)	Top-1 Acc (%)	Top-5 Acc (%)
Res2Net	2.630	6.360	46.05	74.41
Res2Net_c	2.630	16.360	**57.23**	**83.72**
Res2Net_g	2.630	16.360	50.19	78.35
Res2Net_l	2.630	16.360	43.72	73.25
Res2Net_e	2.630	16.360	57.15	83.07

表 8: 激活函数替换的消融实验

   通过对比分析上表结果，可以发现使用 CELu 激活函数替换原始的 ReLU 激活函数效果最为显著。使用 ELU 和 GELU 替换原始也会使模型的准确率提升，其中 CELu、ELU 激活函数 Top-1 的 Accuracy 提升率分别达到了 24.28% 和 24.1%，说明替换激活函数是非常高效的优化策略，可以大幅度提升模型的分类效果！

   同时我们还注意到使用 Leaky-ReLU 激活函数替换原始的 ReLU 激活函数后不仅没有提升，反而 Top-1 的 Accuracy 由 46.05 降低到了 43.72，这可能原因是因为 CELu、ELU 和 GELU 激活函数在负值区间引入了平滑的非线性变化，使得神经网络可以更好地拟合数据。而 Leaky-ReLU 和 ReLU 在负值区间上都是线性的，容易因为负值的消失而导致神经元的死亡或激活函数的失活，这限制了网络的非线性能力，无法很好地拟合复杂的数据模式。此外 CELu、ELU 和 GELU 允许网络中的神经元自动选择激活或者失活状态，支持稀疏激活，因此使用 CELu、ELU 和 GELU 激活函数的模型比使用 Leaky-ReLU 和 ReLU 的模型在非线性性质、鲁棒性、抑制梯度消失和稀疏激活等方面的表现更好。

   此外依旧使用模型改进部分所述的计算经过激活函数 CELu 后死亡神经元个数的代码，统计仅运行 1 个 epoch，测试集第 30-40 个批次，Res2Net50 网络结构经过最后一个 Res2Net Module 中经过残差连接的激活函数后死亡的神经元个数之和，发现此时仅为 2 个!! 与之前惊人的 4138107 个死亡神经元个数相比有着非常良好的改进，这也进一步印证了我们改进之后模型效果的正确性，和改进的高效性！

2. 增加注意力机制的消融实验

分别、同时在 Res2Net Module 最后一个 1x1 卷积层后以及分成四个自网络 x1、x2、x3、x4 内部增加注意力机制，希望网络可以进一步关注目标对象，提高神经网络的性能。消融实验结果如下表所示，其中：

(a) Res2Net 表示原始没有注意力机制的 Res2Net 网络

(b) Res2Net_S1、Res2Net_T1 表示在 Res2Net Module 分成四个子网络 x1、x2、x3、x4 内部分别增加注意力机制 SE、Triplet Attention 的网络模型

(c) Res2Net_S2、Res2Net_T2 分别代表 Res2Net Module 最后一个 1x1 卷积层后分别增加注意力机制 SE、Triplet Attention 的网络模型

(d) Res2Net_ST 代表在 Res2Net Module 分成四个子网络 x1、x2、x3、x4 内部增加注意力机制 SE 同时在最后一个 1x1 卷积层后增加注意力机制 Triplet Attention 的网路模型，Res2Net_SS、Res2Net_TS、Res2Net_TT 代表的模型同理

Method	MACs (G)	Params. (M)	Top-1 Acc (%)	Top-5 Acc (%)
Res2Net	2.630	6.360	46.05	74.41
Res2Net_S1	2.631	16.371	44.19	72.56
Res2Net_T1	2.643	16.365	40.78	69.62
Res2Net_S2	2.633	18.885	49.70	76.81
Res2Net_T2	2.674	16.359	**52.49**	**79.09**
Res2Net_SS	2.635	18.896	50.37	77.39
Res2Net_ST	2.675	16.370	**53.51**	**80.55**
Res2Net_TS	2.631	16.371	49.22	76.88
Res2Net_TT	2.687	16.359	49.41	76.65

表 9: 增加注意力机制的消融实验

对比上表所示的结果，对于方案二，发现在子网络内部添加注意力机制的效果并不好，模型效果不升反降，分析原因可能是因为在子网络内增加注意力机制使得多尺度特征中的某些重要特征被忽视或被错误地加权，同时引入额外的计算和参数量，增加模型的计算复杂性，而现有训练的轮次不够满足复杂模型需求，从而影响模型性能。相比较而言此时 SE 机制实现的效果更好一些，可能是因为 SE 机制的相对参数和计算量较少。

对比上表所示的结果，对于方案一，发现在最后的 1x1 卷积之后添加注意力机制可以使得模型性能在一定程度上实现提升，而相比于 SE 机制添加 TA 机制的效果会更好。这可能是因为 SE 机制主要应用于通道注意力，它主要关注不同通道之间的相关性，通过调整通道权重来增强重要特征的表达能力，而 TA 机制同时应用于通道注意力和空间注意力，它不仅关注通道之间的相关性，还考虑了特征图中不同位置之间的关系，在局部和全局范围内对特征进行加权和集成，所以 TA 机制的提升效果更为明显。

对比上表所示的结果，对于方案三，发现比起分别添加注意力机制，同时在子网络内部和最后的 1x1 卷积之后同时添加注意力机制可以使得模型性能有着更好的提升！这可能是因为子网络内部的注意力机制可以强化了模型的关注对象，有利于 1x1 卷积之后对于重点特征的提取，此时在子网络内部引入 SE 机制、在 1x1 卷积后引入 TA 机制的效果最好，这与之前方案一二的实验效果相吻合，进一步验证了正确性！

3. 大核卷积替换与修改下采样方式的消融实验

将 Res2Net 原始结构网络记为 Res2Net，替换 Res2Net 原始结构中的第一个 7x7 卷积，改进后的模型记为 Res2Net_C，替换 Res2Net 原始的下采样方式记为 Res2Net_D，同时进行大核卷积替换与修改下采样方式改进的网络记为 Res2Net_CD。

消融实验结果如下表所示，据此可以看出这两种改进思路分开、结合都会对模型的性能有所提升，符合我们的预期。同时相对而言大核卷积替换的提升效果更加明显一些，三个改进模型的 Top-1 的 Accuracy 提升率分别为 1.5%、0.4% 和 2.08%：

Method	MACs (G)	Params. (M)	Top-1 Acc (%)	Top-5 Acc (%)
Res2Net	2.630	6.360	46.05	74.41
Res2Net_C	3.464	16.421	46.75	75.86
Res2Net_D	2.625	16.354	46.26	74.83
Res2Net_CD	3.464	16.421	**47.01**	**76.81**

表 10: 大核卷积替换与修改下采样方式的消融实验

4. 四种改进整体的消融实验

对于我们之前提出的四种改进思路进行整体的消融实验，根据之前的消融实验结果我们选择其中最优的结果进行组合，使用 CELu 激活函数替换 ReLU 激活函数（改进模型以 c 表示），同时替换大核卷积、修改下采样方式（改进模型以 CD 表示），在 Res2Net Module 最后一个 1x1 卷积层后引入 TA 注意力机制（改进模型以 T2 表示），同时在 Res2Net Module 子网络中引入 SE 机制和最后一个 1x1 卷积层后引入 TA 注意力机制（改进模型以 ST 表示），消融实验结果如下所示：

Method	MACs (G)	Params. (M)	Top-1 Acc (%)	Top-5 Acc (%)
Res2Net	2.630	6.360	46.05	74.41
Res2Net_c	2.630	16.360	57.23	83.72
Res2Net_cCD	3.464	16.421	57.99	83.82
Res2Net_cCD_T2	3.512	16.425	58.47	**84.89**
Res2Net_cCD_ST	3.513	16.436	**60.90**	84.85

表 11: 四种改进整体的消融实验

根据上表中的数据可以看出，替换激活函数为模型效果提升贡献最大。而引入注意力机制、替换大核卷积、修改下采样方式也会在此基础上对模型进行一定的优化。效果最好的模型是 Res2Net_cCD_ST，即使用了 CELu 激活函数替换 ReLU 激活函数、替换大核卷积、修改下采样方式、同时在 Res2Net Module 子网络中引入 SE 机制和最后一个 1x1 卷积层后引入 TA 注意力机制，虽然它的 Top-5 的 Accuracy 只有 84.85%，提升了 14.03%，不如 Res2Net_cCD_T2 的 84.89%，不过这一部分差距并不大；但是 Res2Net_cCD_ST 的 Top-1 的 Accuracy 可以到达 60.90%，远高于其他改进模型，提升率达到了 32.25%，实验提升明显，改进卓有成效！

五、 Git 记录

我们的 GitHub 仓库为：https://github.com/Myrrolinz/Deep-Learning-Final

五位同学的 Git commit 均超过 10 次以上；三位负责改进模型的同学分别在自己的 branch 内进行操作，branch 名为名字简写。

Git 用户名对应：

- 管昀玫：Myrrolinz
- 宋佳蓁：Jiazhen-Song
- 石家琪：civilizwa
- 康家玉：biukjy
- 李玥婵：qudn

六、 总结

本次深度大作业主要可以分为模型复现与模型改进两个阶段，选取 resnet 作为 baseline。在第一阶段模型复现部分，我们总共复现了 8 种计算机视觉相关的深度学习模型，分别为：ResNet, VAN, RepLKNet, Coordinate Attention, Res2Net, Triplet Attention, SKNet, GCNet，其中 Triplet Attention 的复现效果最好，Top-1 Acc 可达到 54%，Top-5 可达到 80%。

在第二阶段，即模型改进部分，我们选择了 VAN, Res2Net, Triplet Attention 三个网络，针对性的提出改进，尝试进行模型融合，获得了良好的效果。最终实现效果最好的改进模型为 Triplet-CA，Top-1 Acc 为 65.76%，Top-5 Acc 为 88.91 %，改进最大提升模型为 Res2Net_cCD_ST，Top-1 Acc 提升了 32.25% (60.90 vs. 46.05)，Top-5 Acc 提升了 10.44% (84.85% vs. 74.41%)，证明我们的实验改进效果显著。

在本次深度学习大作业的实践过程中，每位同学都经历了仔细研读原始论文、查阅改进方向资料、学习模型结构和算法原理的过程。在这其中当然也遇到了一些困难和问题，比如引入注意力机制时通道的计算和其他的一些数学推导方面的问题，但是通过向老师请教、组内组间之间讨论最终成功克服了困难。不仅深入了解了深度学习模型的原理和实践，提高了个人的技术水平，也培养了团队协作的能力、思考与解决问题的能力，收获了知识、友谊和成就感！

此处声明：本报告中出现的所有图均为小组成员手绘。

参考文献

[1] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Proceedings of the IEEE/CVF international conference on computer vision workshops*, pages 0–0, 2019.

[2] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11963–11975, 2022.

[3] Shang-Hua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr. Res2net: A new multi-scale backbone architecture. *IEEE transactions on pattern analysis and machine intelligence*, 43(2):652–662, 2019.

[4] Meng-Hao Guo, Cheng-Ze Lu, Zheng-Ning Liu, Ming-Ming Cheng, and Shi-Min Hu. Visual attention network. *arXiv preprint arXiv:2202.09741*, 2022.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 558–567, 2019.

[7] Qibin Hou, Daquan Zhou, and Jiashi Feng. Coordinate attention for efficient mobile network design. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13713–13722, 2021.

[8] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[9] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 510–519, 2019.

[10] Diganta Misra, Trikay Nalamada, Ajay Uppili Arasanipalai, and Qibin Hou. Rotate to attend: Convolutional triplet attention module. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3139–3148, 2021.

[11] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.