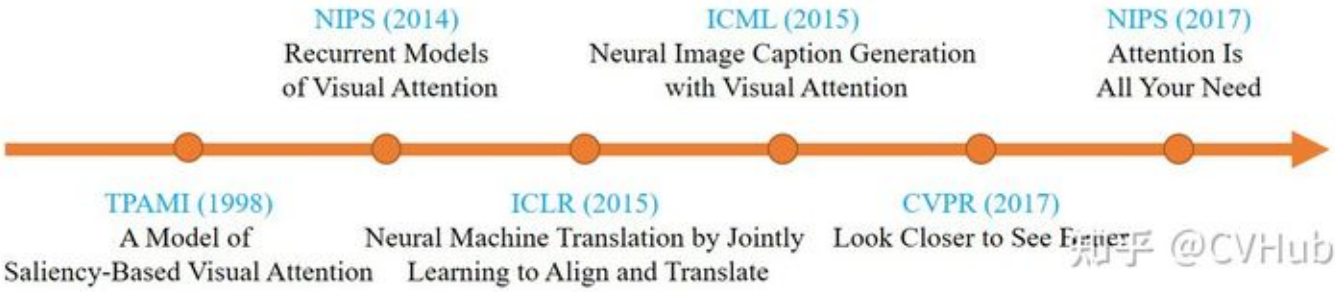
一文看尽深度学习中的各种注意力机制

导读

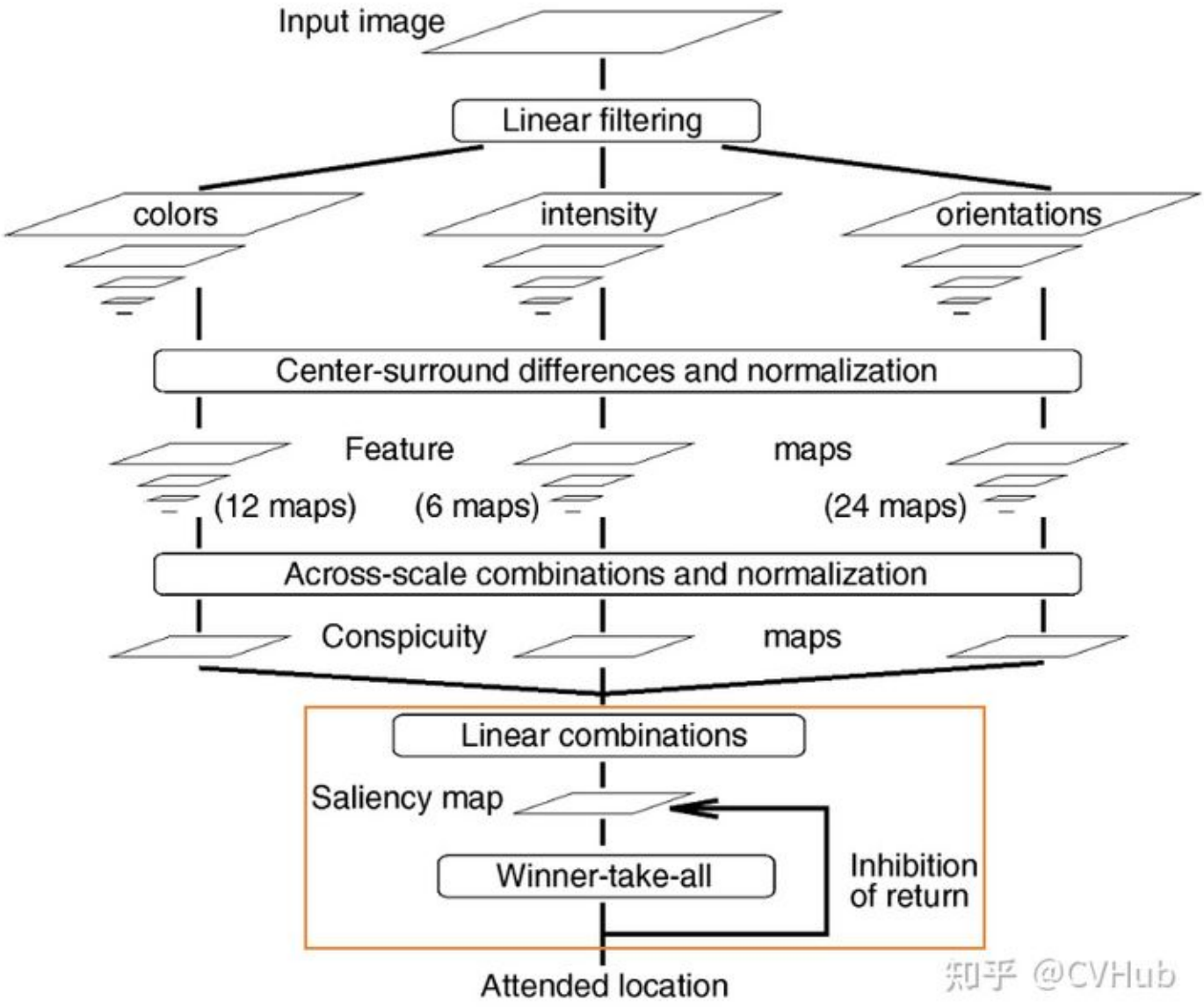
视觉注意力机制是人类视觉所特有的一种大脑信号处理机制，而深度学习中的注意力机制正是借鉴了人类视觉的注意力思维方式。一般来说，人类在观察外界环境时会迅速的扫描全景，然后根据大脑信号的处理快速的锁定重点关注的目标区域，最终形成**注意力焦点[1]**。该机制可以帮助人类在有限的资源下，从大量无关背景区域中筛选出具有重要价值信息的目标区域，帮助人类更加高效的处理视觉信息。

起源

注意力机制在计算机视觉领域的应用主要使用于捕捉图像上的respective field，而在自然语言处理领域中的应用主要使用于定位关键的token。下面简单介绍下注意力机制在早期的几个经典应用。

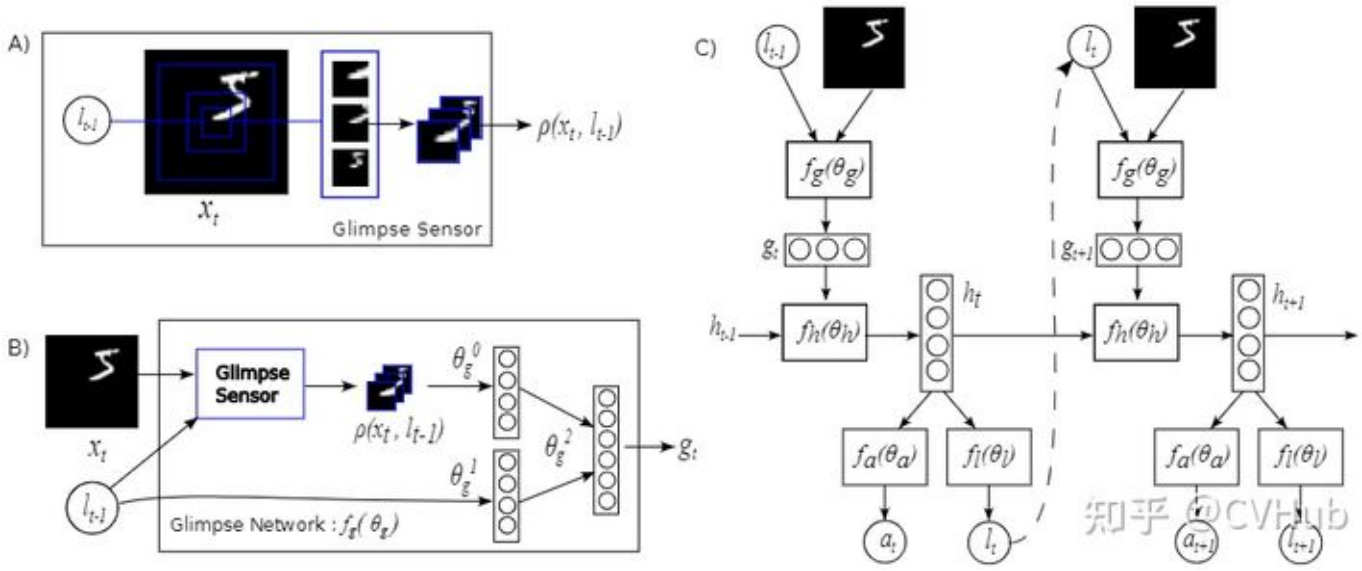


《A Model of Saliency-Based Visual Attention for Rapid Scene Analysis》[2]



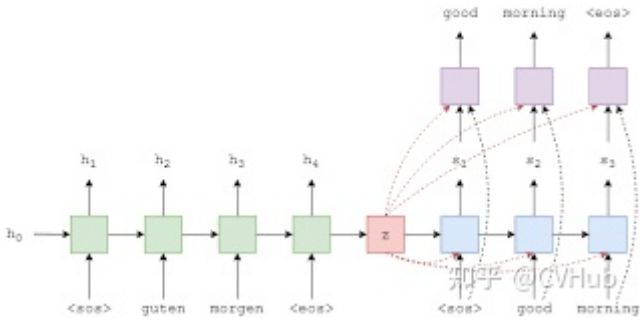
这是早期将注意力机制应用于计算机视觉领域的一篇代表作，文章于1998年发表于TAPMI。作者受早期灵长目视觉系统的神经元结构启发，提出了一种视觉注意力系统，可以将多尺度的图像特征组合成单一的显著性图。最后，利用一个动态神经网络，并按照显著性的顺序来高效的选择重点区域。

《Recurrent Models of Visual Attention》[3]



使注意力机制真正火起来的当属于谷歌DeepMind于2014年所提出的这篇文章，该论文首次在RNN模型上应用了注意力机制的方法进行图像分类。

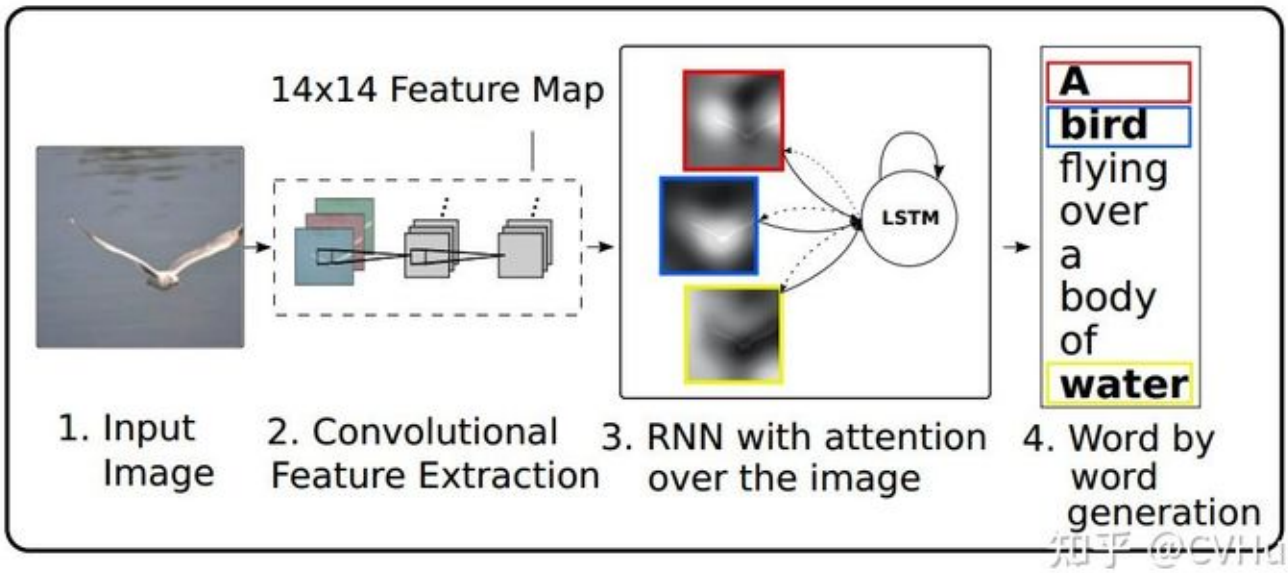
《Neural Machine Translation by Jointly Learning to Align and Translate》[4]



这是由深度学习三巨头之一Yoshua Bengio等人于2015年发表于ICLR上的一篇论文，该论文的最大贡献是将注意力机制首次应用到NLP领域，实现了同步的对齐和翻译，解决以往神经机器翻译(NMT)领域使用Encoder-Decoder架构的一个潜在问题，即将信息都压缩在固定长度的向量，无法对应长句子。

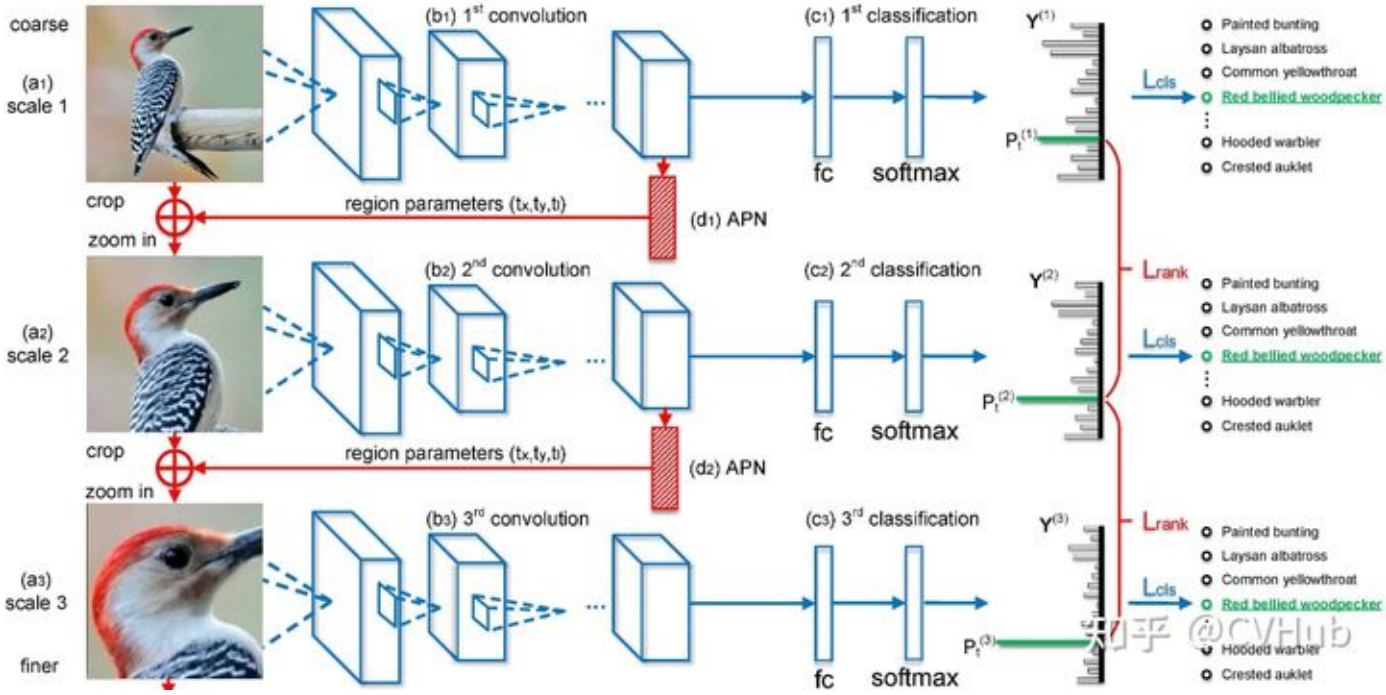
《Show, Attend and Tell: Neural Image Caption Generation with Visual Attention》[5]

Figure 1. Our model learns a words/image alignment. The visualized attentional maps (3) are explained in Sections 3.1 & 5.4



这篇文章由Yoshua Bengio等人于2015年在ICML上所发表的，该论文将注意力机制引入到图像领域，作者提出了两种基于注意力机制的图像描述生成模型: 使用基本反向传播训练的Soft Attetnion方法和使用强化学习训练的Hard Attention方法。

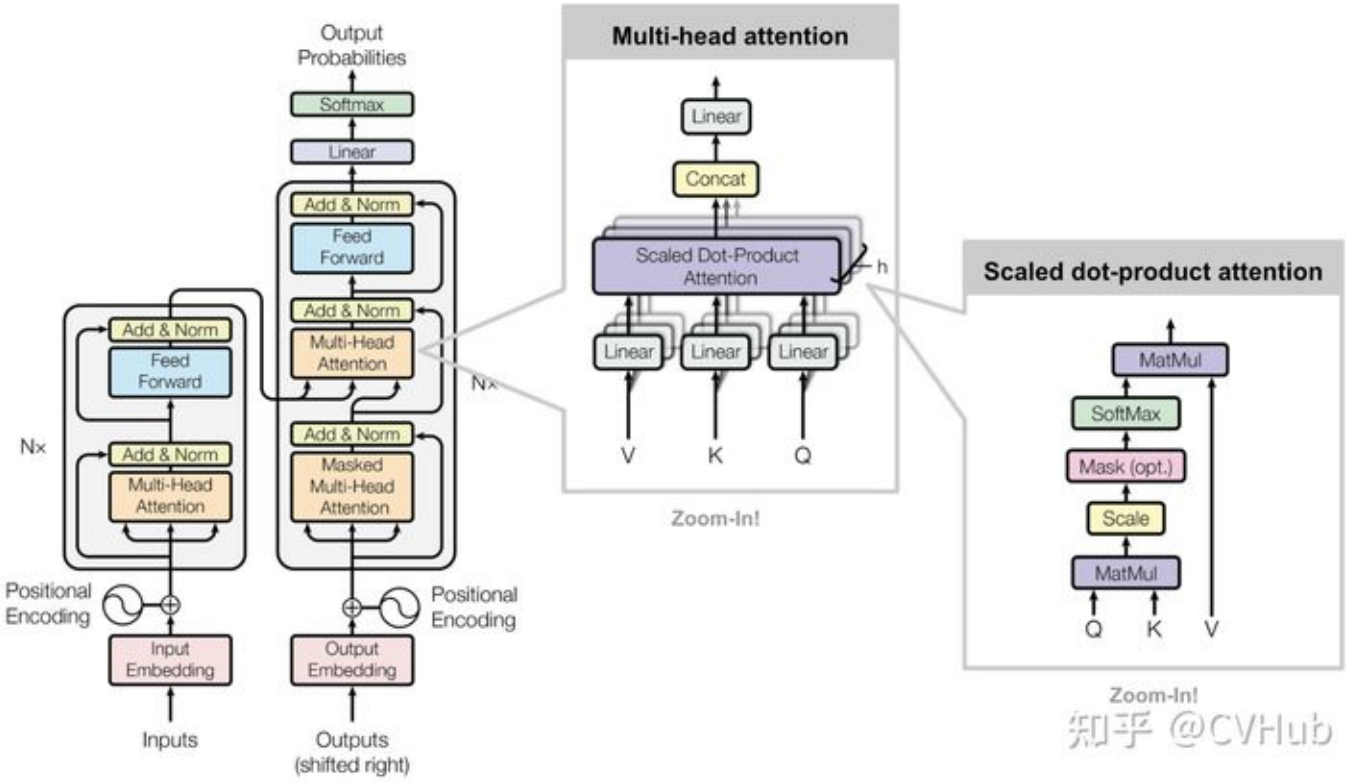
《Look Closer to See Better: Recurrent Attention Convolutional Neural Network for Fine-grained Image Recognition》[6]



这是发表于CVPR 2017年的一篇文章，作者提出了一种基于CNN的注意力机制，叫做循环注意力卷积神经网络（Recurrent Attention Convolutional Neural Network, RA-CANN），该网络可以递归地分析局部信息，并从所获取的局部区域中提取细粒度信息。此外，作者还引入了一个注意

力生成子网络（Attenion Proposal Sub-Network, APN），迭代的对整图操作以生成对应的子区域，最后再将各个子区域的预测记过整合起来，从而后的整张图片最终的分类预测结果。

《Attention is All Your Need》[7]



这是由谷歌机器翻译团队于2017年发表于NIPS上的一篇文章，该论文最大的贡献便是抛弃了以往机器翻译基本都会应用的RNN或CNN等传统架构，以编码器-解码器为基础，创新性的提出了一种Transformer架构。该架构可以有效的解决RNN无法并行处理以及CNN无法高效的捕捉长距离依赖的问题，近期更是被进一步地应用到了计算机视觉领域，同时在多个CV任务上取得了SOTA性能，挑战CNN在CV领域多年的霸主地位。

发展

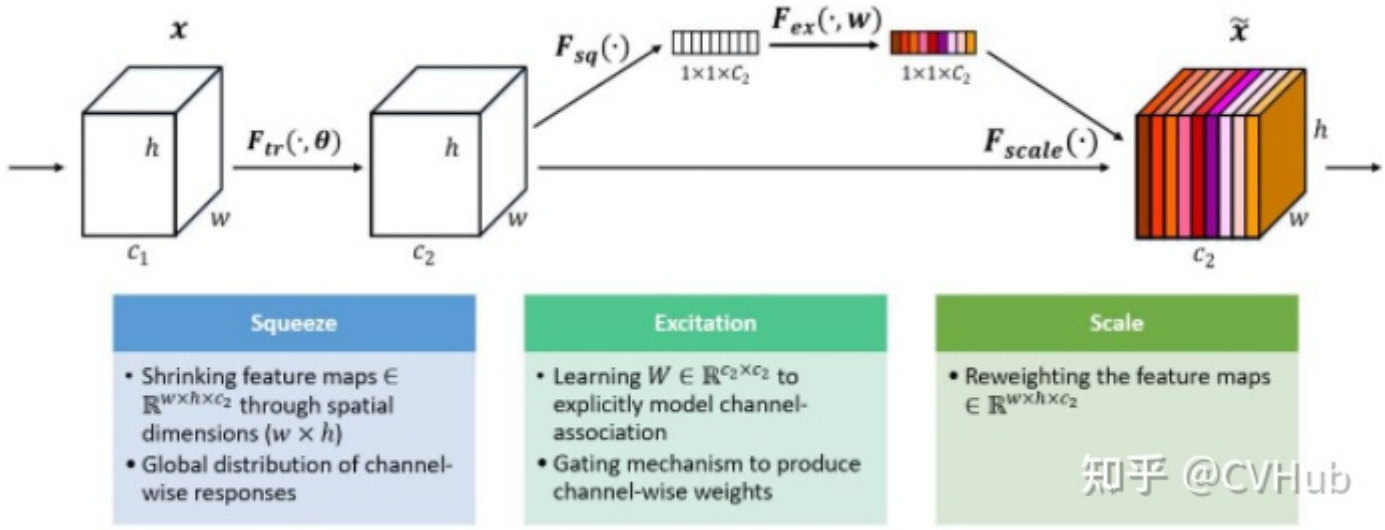
本文将重点围绕**通道、空间、自注意力、类别**等多个**维度[8]**介绍计算机视觉领域中较为出名的注意力机制方法,力争用最简短的语言解释得更加通俗易懂。

通道&空间注意力

> **通道注意力**旨在显示的建模出不同通道之间的相关性，通过网络学习的方式来自动获取到每个特征通道的重要程度，最后再为每个通道赋予不同的权重系数，从而来强化重要的特征抑制非重要的特征。
> **空间注意力**旨在提升关键区域的特征表达，本质上是将原始图片中的空间信息通过空间转换模块，变换到另一个空间中并保留关键信息，为每个位置生成权重掩膜（mask）并加权输出，从而增强感兴趣的特定目标区域同时弱化不相关的背景区域。

SE-Net[9]

《Squeeze-and-Excitation Networks》发表于CVPR 2018，是CV领域将注意力机制应用到通道维度的代表作，后续大量基于通道域的工作均是基于此进行润(魔)色(改)。SE-Net是ImageNet 2017大规模图像分类任务的冠军，结构简单且效果显著，可以通过特征重标定的方式来自适应地调整通道之间的特征响应。

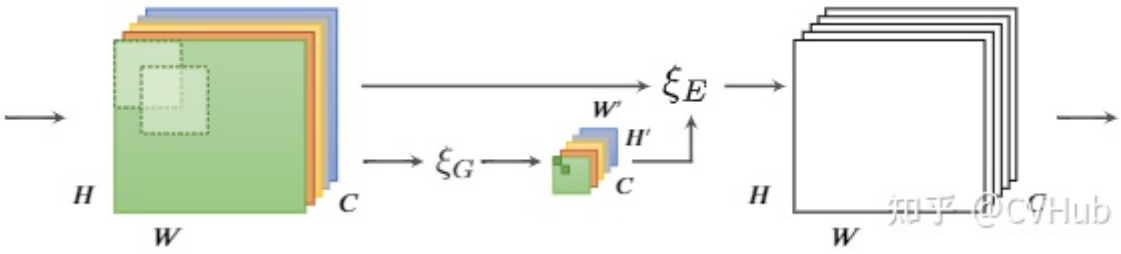


Squeeze 利用全局平均池化(Global Average Pooling, GAP) 操作来提取全局感受野，将所有特征通道都抽象为一个点；
Excitation 利用两层的多层感知机(Multi-Layer Perceptron, MLP) 网络来进行非线性的特征变换，显示地构建特征图之间的相关性；
Transform 利用Sigmoid激活函数实现特征重标定，强化重要特征图，弱化非重要特征图。

```
class SELayer(nn.Module):
    def __init__(self, channel, reduction=16):
        super(SELayer, self).__init__()
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        self.fc = nn.Sequential(
            nn.Linear(channel, channel // reduction, bias=False),
            nn.ReLU(inplace=True),
            nn.Linear(channel // reduction, channel, bias=False),
            nn.Sigmoid()
        )

    def forward(self, x):
        b, c, _, _ = x.size()
        y = self.avg_pool(x).view(b, c)
        y = self.fc(y).view(b, c, 1, 1)
        return x * y.expand_as(x)
```

GE-Net[10]

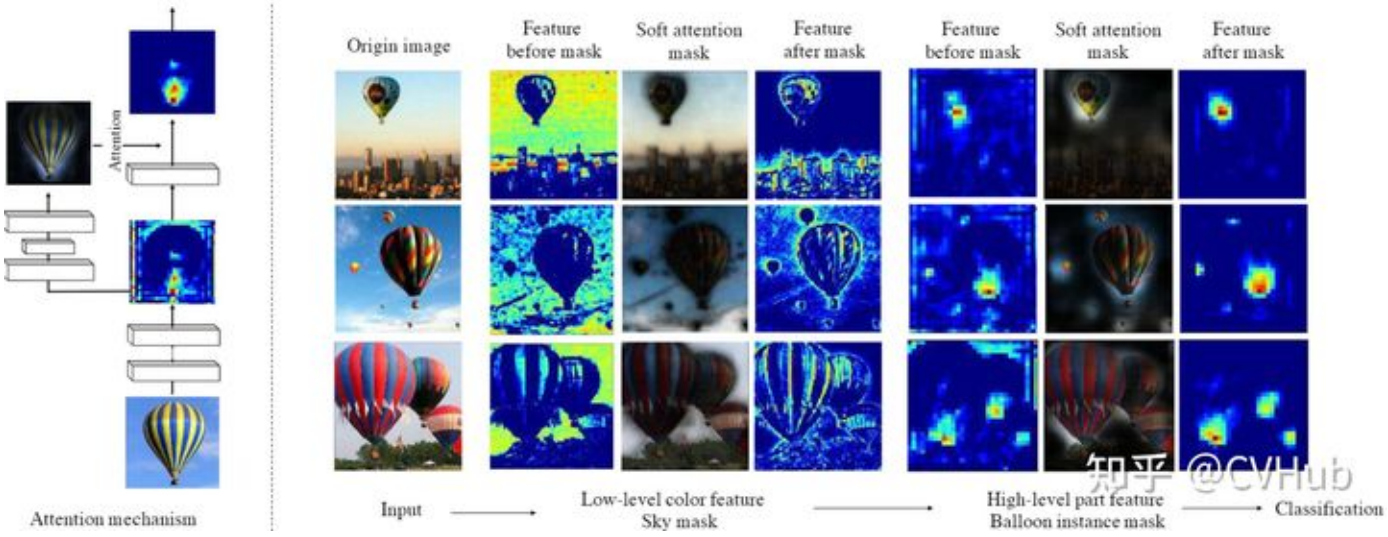
《Gather-Excite: Exploiting Feature Context in Convolutional Neural Networks》发表于NIPS 2018，从上下文建模的角度出发，提出了一种比SE-Net更一般的形式。GE-Net充分利用空间注意力来更好的挖掘特征之间的上下文信息。【代码链接可访问**github[11]**】



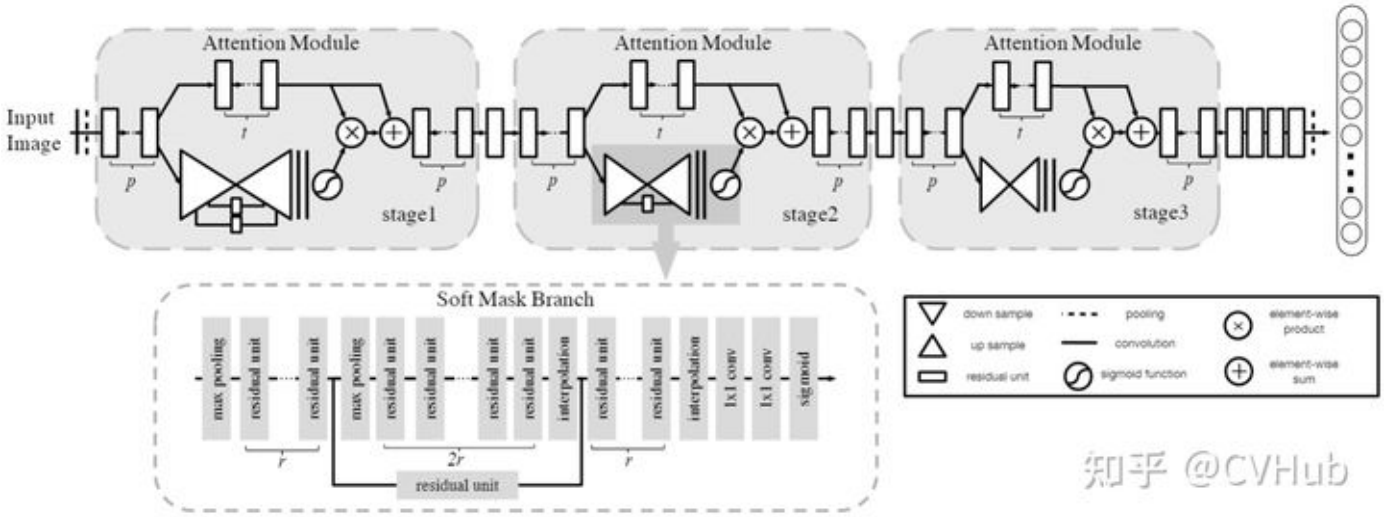
Gather 用于从局部的空间位置上提取特征；
Excite 用于将特征缩放至原始尺寸。

RA-Net[12]

《Residual attention network for image classification》发表于CVPR 2017，利用下采样和上采样操作提出了一种基于空间注意力机制的残差注意力网络。



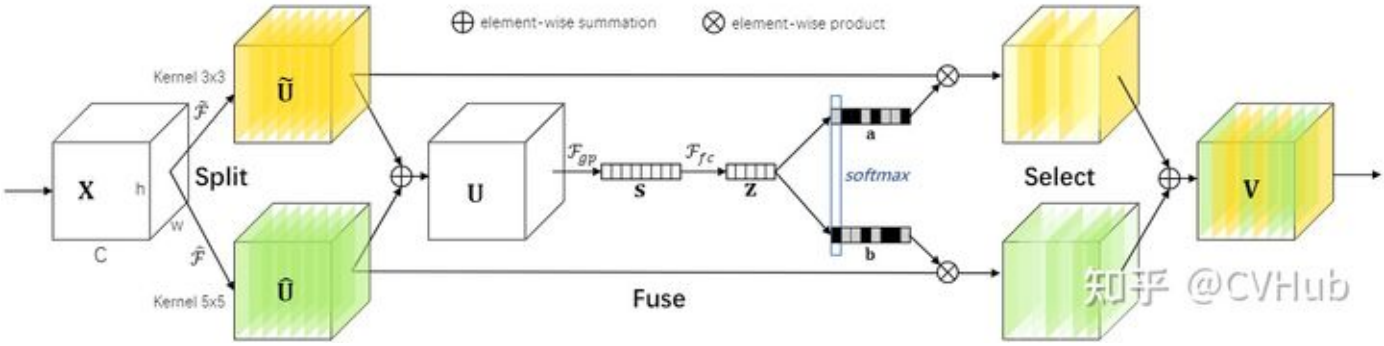
以往的Attention模型大多应用于图像分割和显著性检测任务，出发点在于将注意力集中在部分感兴趣区域或显著区域上。本文尝试在常规的分类网络中引入侧边分支，该分支同样是由一系列卷积和池化操作来逐渐地提取高级语义特征并增大网络的感受野，最后再将该分支直接上采样为原始分辨率尺寸作为特征激活图叠加回原始输入。



该方法提升效果好像并不明显，而且由于引入大量额外的参数，导致计算开销非常大。

SK-Net[13]

《Selective Kernel Networks》发表于CVPR 2019，原SE-Net的作者Momenta也参与到这篇文章中。SK-Net主要灵感来源于Inception-Net的多分支结构以及SE-Net的特征重标定策略，研究的是卷积核之间的相关性，并进一步地提出了一种选择性卷积核模块。SK-Net从多尺度特征表征的角度出发，引入多个带有不同感受野的并行卷积核分支来学习不同尺度下的特征图权重，使网络能够挑选出更加合适的多尺度特征表示，不仅解决了SE-Net中单一尺度的问题，而且也结合了多分枝结构的思想从丰富的语义信息中筛选出重要的特征。



Split 采用不同感受野大小的卷积核捕获多尺度的语义信息；
Fuse 融合多尺度语义信息，增强特征多样性；
Select 在不同向量空间（代表不同尺度的特征信息）中进行Softmax操作，为合适的尺度通道赋予更高的权重。

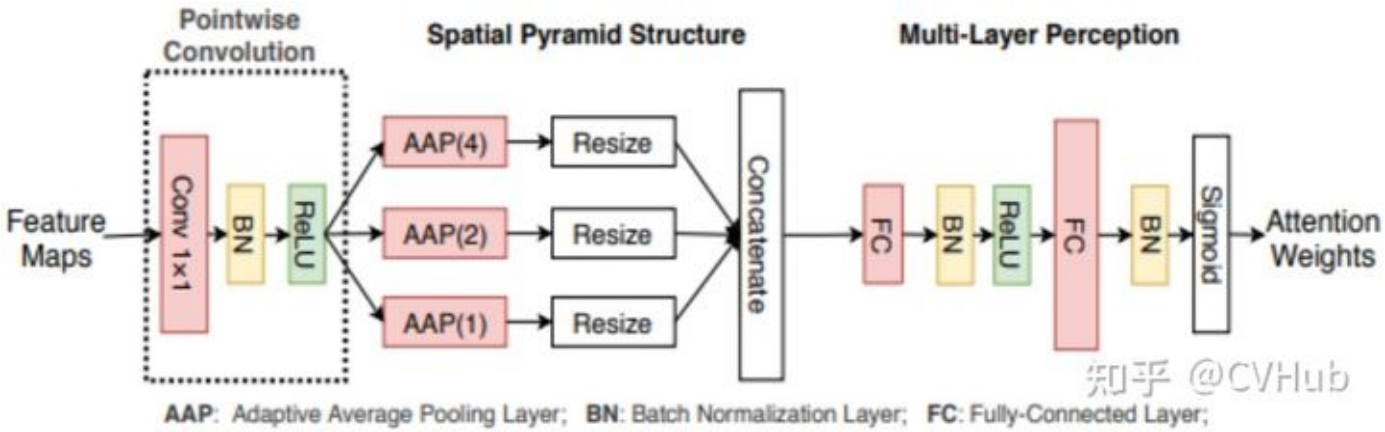
```
class SKConv(nn.Module):
    def __init__(self, features, M=2, G=32, r=16, stride=1, L=32):
        """ Constructor
        Args:
            features: input channel dimensionality.
```

```
            M: the number of branchs.
            G: num of convolution groups.
            r: the ratio for compute d, the length of z.
            stride: stride, default 1.
            L: the minimum dim of the vector z in paper, default 32.
        """
        super(SKConv, self).__init__()
        d = max(int(features/r), L)
        self.M = M
        self.features = features
        self.convs = nn.ModuleList([])
        for i in range(M):
            self.convs.append(nn.Sequential(
                nn.Conv2d(features, features, kernel_size=3,
                stride=stride,
                padding=1+i,
                dilation=1+i,
                groups=G, bias=False),
                nn.BatchNorm2d(features),
                nn.ReLU(inplace=False)
            ))
        self.gap = nn.AdaptiveAvgPool2d((1,1))
        self.fc = nn.Sequential(nn.Conv2d(features, d,
                kernel_size=1,
                stride=1, bias=False),
                                nn.BatchNorm2d(d),
                                nn.ReLU(inplace=False))
        self.fcs = nn.ModuleList([])
        for i in range(M):
            self.fcs.append(
                 nn.Conv2d(d, features, kernel_size=1, stride=1)
            )
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):

        batch_size = x.shape[0]

        feats = [conv(x) for conv in self.convs]
        feats = torch.cat(feats, dim=1)
        feats = feats.view(batch_size, self.M,
                self.features,
                feats.shape[2],
                feats.shape[3])
```

```
        feats_U = torch.sum(feats, dim=1)
        feats_S = self.gap(feats_U)
        feats_Z = self.fc(feats_S)

        attention_vectors = [fc(feats_Z) for fc in self.fcs]
        attention_vectors = torch.cat(attention_vectors, dim=1)
        attention_vectors = attention_vectors.view(batch_size,
                        self.M, self.features, 1, 1)
        attention_vectors = self.softmax(attention_vectors)

        feats_V = torch.sum(feats*attention_vectors, dim=1)

        return feats_V
```

SPA-Net[14]

《Spatial Pyramid Attention Network for Enhanced Image Recognition》 发表于ICME 2020 并获得了最佳学生论文。考虑到 SE-Net 这种利用 GAP 去建模全局上下文的方式会导致空间信息的损失，SPA-Net另辟蹊径，利用多个自适应平均池化(Adaptive Averatge Pooling, APP) 组成的空间金字塔结构来建模局部和全局的上下文语义信息，使得空间语义信息被更加充分的利用到。

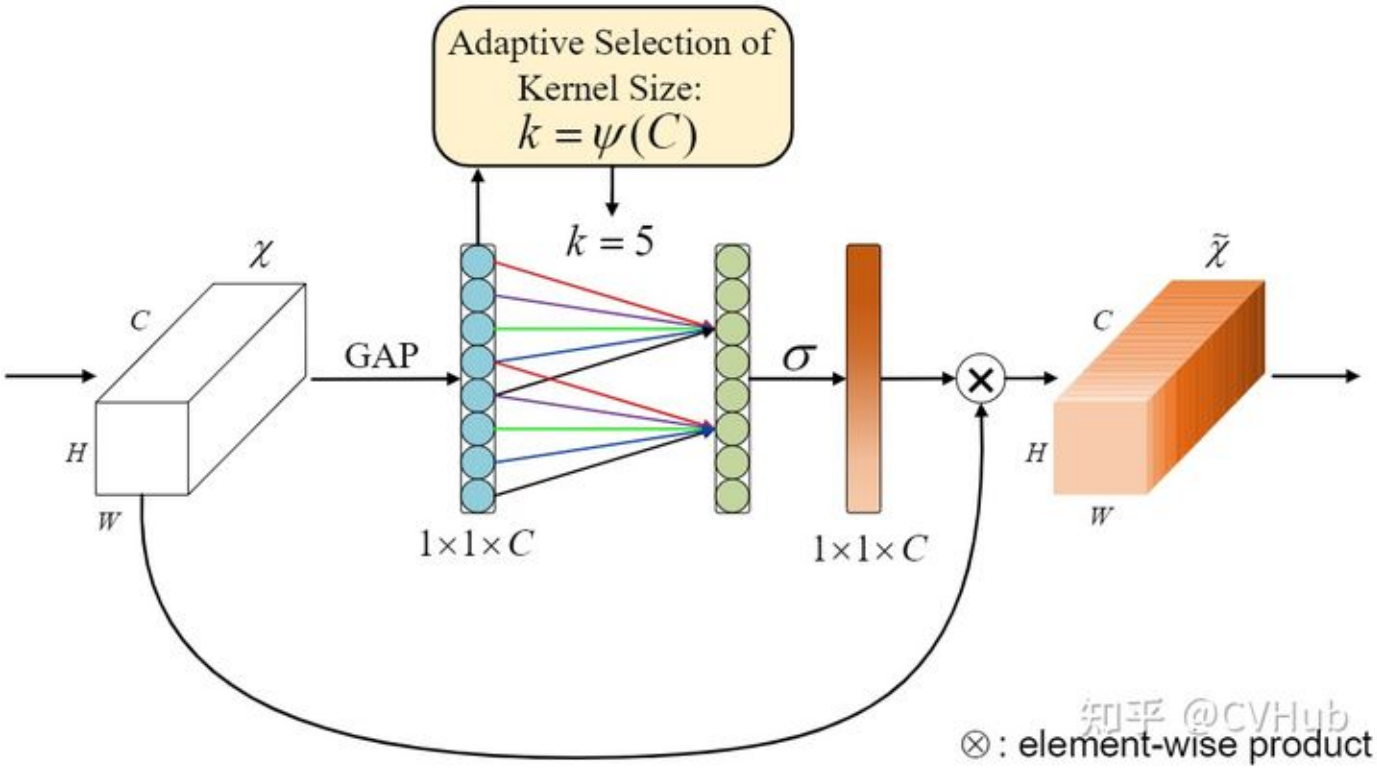



```
class CPSPPSELayer(nn.Module):
    def __init__(self,in_channel, channel, reduction=16):
        super(CPSPPSELayer, self).__init__()
        if in_channel != channel:
            self.conv1 = nn.Sequential(
                nn.Conv2d(in_channel, channel, kernel_size=1, stride=1, bias=False),
                nn.BatchNorm2d(channel),
                nn.ReLU(inplace=True)
            )
        self.avg_pool1 = nn.AdaptiveAvgPool2d(1)
        self.avg_pool2 = nn.AdaptiveAvgPool2d(2)
        self.avg_pool4 = nn.AdaptiveAvgPool2d(4)
```

```
        self.fc = nn.Sequential(
            nn.Linear(channel*21, channel*21 // reduction, bias=False),
            nn.ReLU(inplace=True),
            nn.Linear(channel*21 // reduction, channel, bias=False),
            nn.Sigmoid()
        )

    def forward(self, x):
        x = self.conv1(x) if hasattr(self, 'conv1') else x
        b, c, _, _ = x.size()
        y1 = self.avg_pool1(x).view(b, c)  # like resize() in numpy
        y2 = self.avg_pool2(x).view(b, 4 * c)
        y3 = self.avg_pool4(x).view(b, 16 * c)
        y = torch.cat((y1, y2, y3), 1)
        y = self.fc(y)
        b, out_channel = y.size()
        y = y.view(b, out_channel, 1, 1)
        return y
```

ECA-Net[15]

《ECANet：Efficient Channel Attention for Deep Convolutional Neural Networks》发表于CVPR 2020，是对SE-Net中特征变换部分进行了改进。SE-Net的通道信息交互方式是通过全连接实现的，在降维和升维的过程中会损害一部分的特征表达。ECA-Net则进一步地利用一维卷积来实现通道间的信息交互，相对于全连接实现的全局通道信息交互所带来的计算开销，ECA-Net提出了一种基于自适应选择卷积核大小的方法，以实现局部交互，从而显著地降低模型复杂度且保持性能。



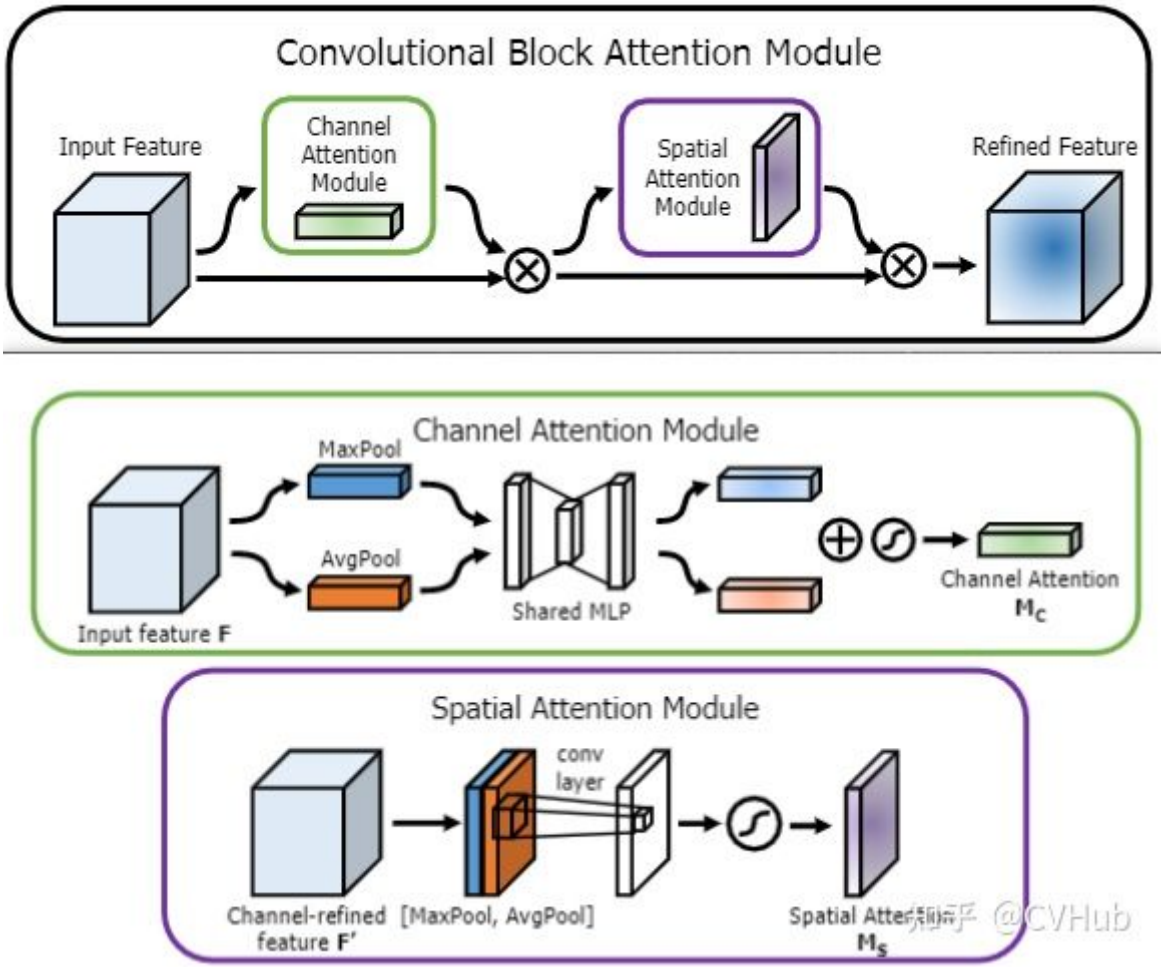

```
class ECALayer(nn.Module):
    """Constructs a ECA module.
    Args:
        channel: Number of channels of the input feature map
        k_size: Adaptive selection of kernel size
    """
    def __init__(self, channel, k_size=3):
        super(ECALayer, self).__init__()
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        self.conv = nn.Conv1d(1, 1, kernel_size=k_size,
                padding=(k_size - 1) // 2, bias=False)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        # feature descriptor on the global spatial information
        y = self.avg_pool(x)
        # Two different branches of ECA module
        y = self.conv(y.squeeze(-1).transpose(-1, -2))
                .transpose(-1, -2).unsqueeze(-1)
        # Multi-scale information fusion
        y = self.sigmoid(y)

        return x * y.expand_as(x)
```

混合注意力

> 空间注意力由于将每个通道中的特征都做同等处理，忽略了通道间的信息交互；
> 通道注意力则是将一个通道内的信息直接进行全局处理，容易忽略空间内的信息交互；
> **混合注意力**主要是共同结合了通道域、空间域等注意力的形式来形成一种更加综合的特征注意力方法。

CBAM[16]

《CBAM: Convolutional Block Attention Module》发表于CVPR 2018，在原有通道注意力的基础上，衔接了一个空间注意力模块(Spatial Attention Modul, SAM)。SAM是基于通道进行全局平均池化以及全局最大池化操作，产生两个代表不同信息的特征图，合并后再通过一个感受野较大的7×7卷积进行特征融合，最后再通过Sigmoid操作来生成权重图叠加回原始的输入特征图，从而使得目标区域得以增强。总的来说，对于空间注意力来说，由于将每个通道中的特征都做同等处理，忽略了通道间的信息交互；而通道注意力则是将一个通道内的信息直接进行全局处理，容易忽略空间内的信息交互。



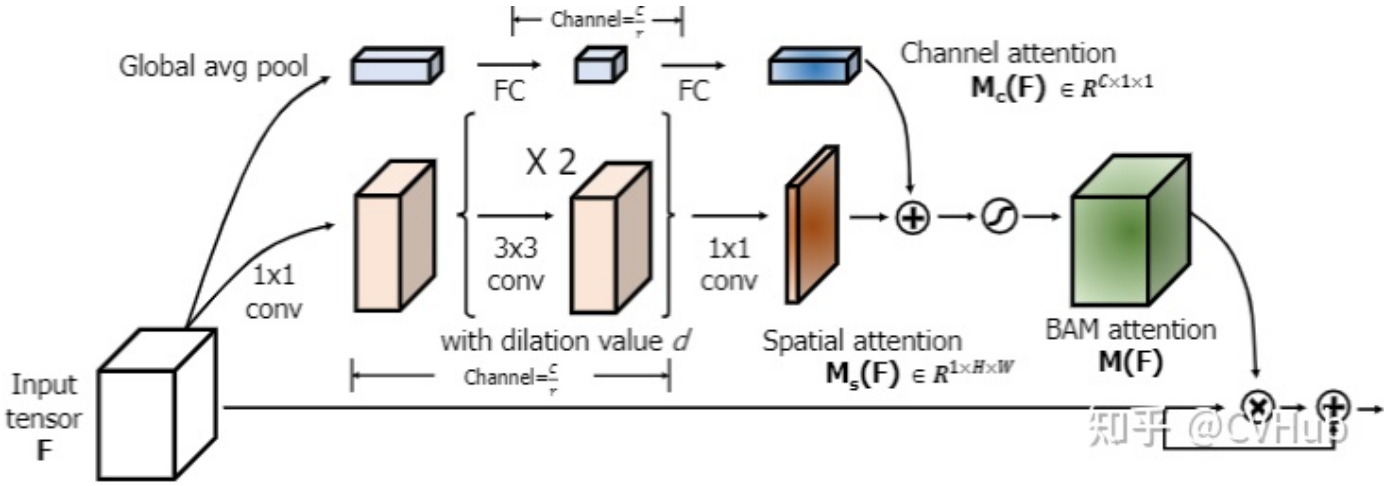
作者最终通过实验验证先通道后空间的方式比先空间后通道或者通道空间并行的方式效果更佳。

```
# https://github.com/Jongchan/attention-module/blob/master/MODELS/cbam.py
class CBAM(nn.Module):
    def __init__(self, gate_channels, reduction_ratio=16,
                 pool_types=['avg', 'max'], no_spatial=False):
```

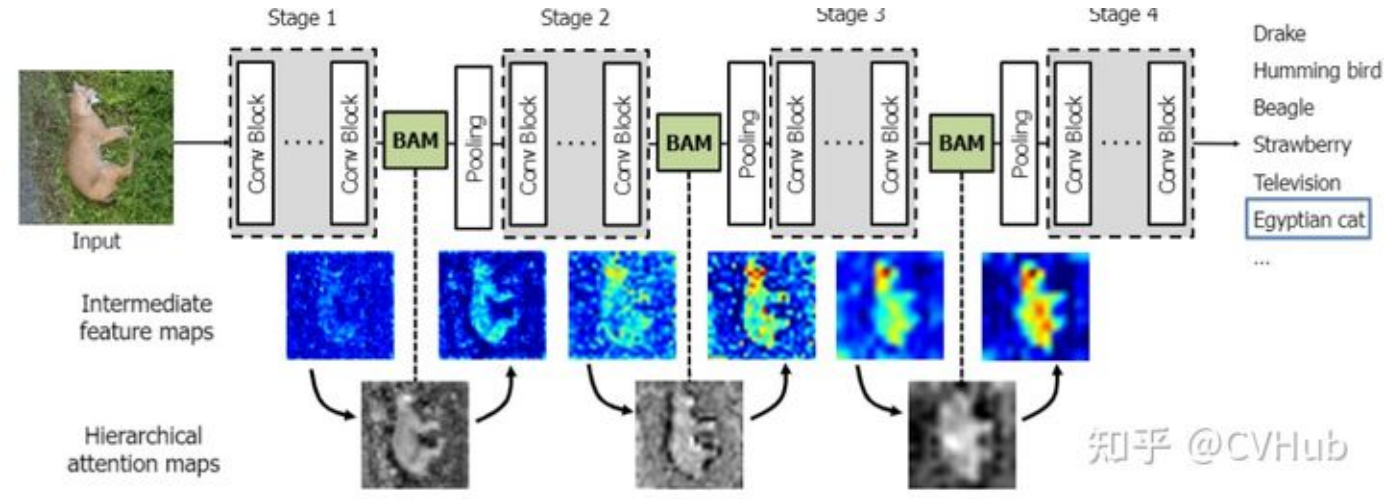
```
        super(CBAM, self).__init__()
        self.ChannelGate = ChannelGate(gate_channels, reduction_ratio, pool_types)
        self.no_spatial=no_spatial
        if not no_spatial:
            self.SpatialGate = SpatialGate()
    def forward(self, x):
        x_out = self.ChannelGate(x)
        if not self.no_spatial:
            x_out = self.SpatialGate(x_out)
        return x_out
```

BAM[17]

《BAM: Bottleneck Attention Module》发表于BMC 2018，提出了一个简单有效的注意力模型来获取空间和通道的注意力图。



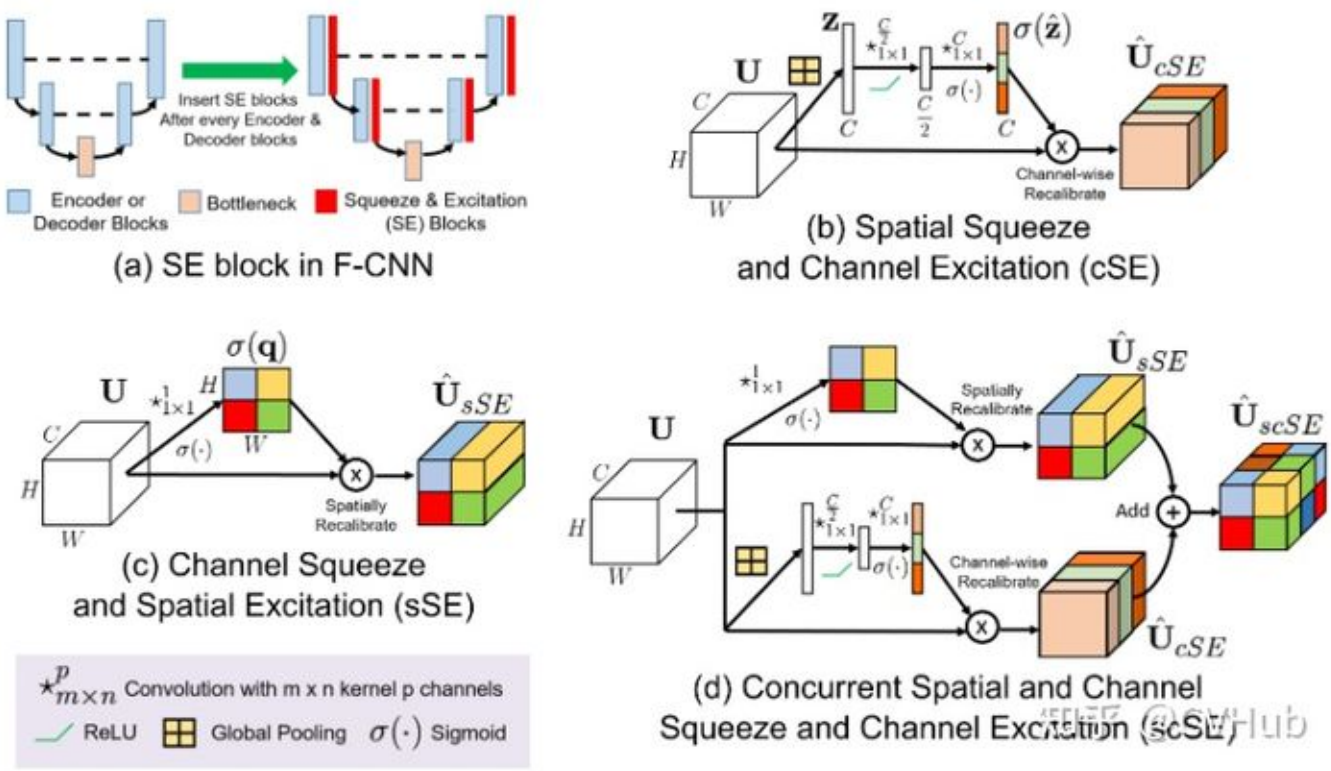
BAM形成了一种分层的注意力机制，可以有效地抑制背景特征，使模型更加聚焦于前景特征，从而加强高级语义，实现更高的性能。



不同于CBAM并联的方式，BAM以串联的方式来相继提取不同域的注意力图。

scSE[18]

《Concurrent Spatial and Channel Squeeze & Excitation in Fully Convolutional Networks》发表于MICCAI 2018，是一种更轻量化的SE-Net变体，在SE的基础上提出cSE、sSE、scSE这三个变种。cSE和sSE分别是根据通道和空间的重要性来校准采样。scSE则是同时进行两种不同采样校准，得到一个更优异的结果。

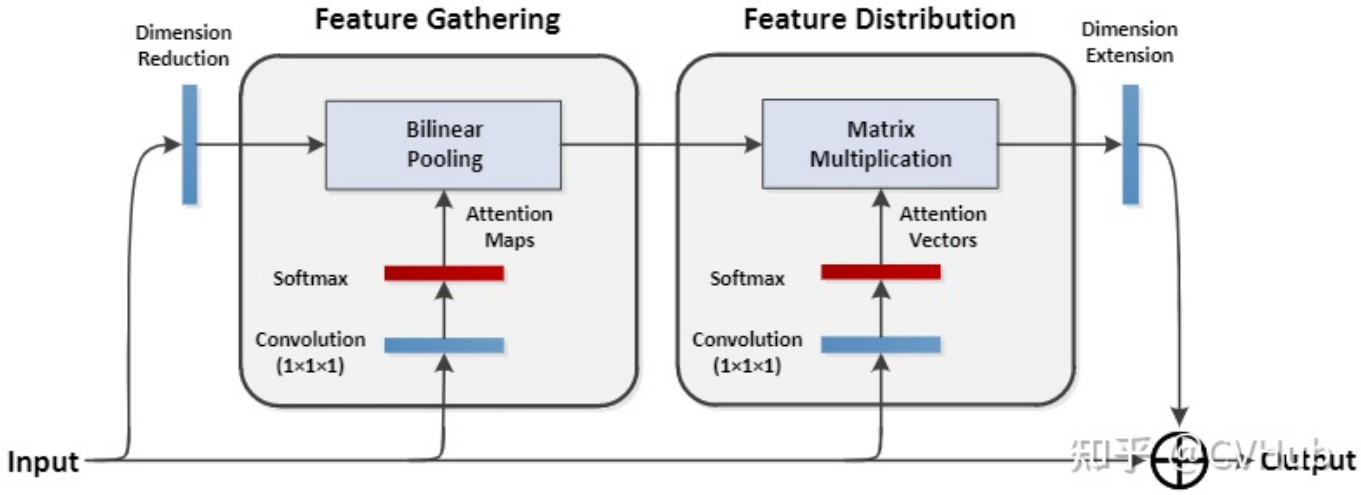



```
class SCSEModule(nn.Module):
    def __init__(self, ch, re=16):
        super().__init__()
        self.cSE = nn.Sequential(nn.AdaptiveAvgPool2d(1),
                                 nn.Conv2d(ch,ch//re,1),
                                 nn.ReLU(inplace=True),
                                 nn.Conv2d(ch//re,ch,1),
                                 nn.Sigmoid())
        self.sSE = nn.Sequential(nn.Conv2d(ch,ch,1),
                                 nn.Sigmoid())
    def forward(self, x):
        return x * self.cSE(x) + x * self.sSE(x)
```

A2-Nets[19]

《A2-Nets: Double Attention Networks》发表于NIPS 2018，提出了一种双重注意力网络。该网络首先使用二阶的注意力池化(Second-order Attention Pooling, SAP) 用于将整幅图的所有关

键特征归纳到一个集合当中，然后再利用另一种注意力机制将这些特征分别应用到图像中的每个区域。




```
class DoubleAtten(nn.Module):
    """
    A2-Nets: Double Attention Networks. NIPS 2018
    """
    def __init__(self,in_c):
        super(DoubleAtten,self).__init__()
        self.in_c = in_c
        """
        Convolve the same input feature map to produce
        three feature maps with the same scale, i.e., A, B, V (as shown in paper).
        """
        self.convA = nn.Conv2d(in_c,in_c,kernel_size=1)
        self.convB = nn.Conv2d(in_c,in_c,kernel_size=1)
        self.convV = nn.Conv2d(in_c,in_c,kernel_size=1)
    def forward(self,input):

        feature_maps = self.convA(input)
        atten_map = self.convB(input)
        b, _, h, w = feature_maps.shape

        feature_maps = feature_maps.view(b, 1, self.in_c, h*w) # reshape A
        # reshape B to generate attention map
        atten_map = atten_map.view(b, self.in_c, 1, h*w)
        # Multiply the feature map and the attention weight map
        # to generate a global feature descriptor
        global_descriptors = torch.mean(
                (feature_maps * F.softmax(atten_map, dim=-1)),dim=-1)
```

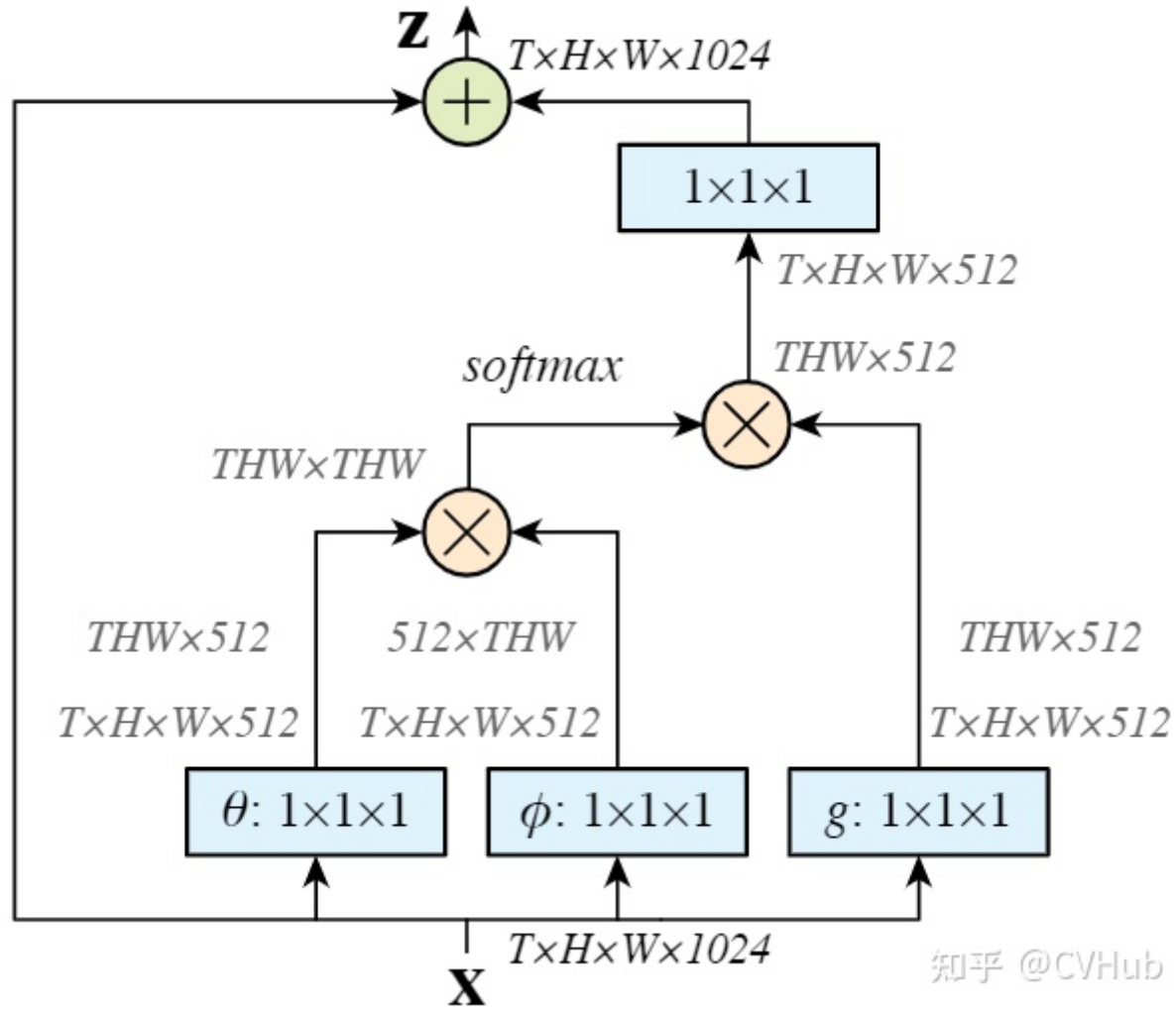
```
        v = self.convV(input)
        # 生成 attention_vectors
        atten_vectors = F.softmax(v.view(b, self.in_c, h*w), dim=-1)
        out = torch.bmm(atten_vectors.permute(0,2,1), global_descriptors)
                .permute(0,2,1)

        return out.view(b, _, h, w)
```

自注意力

> 自注意力是注意力机制的一种变体，其目的是为了减少对外部信息的依赖，尽可能地利用特征内部固有的信息进行注意力的交互。

Non-Local[20]

《Non-local Neural Networks》发表于CVPR 2018，是第一篇将自注意力机制引入图像领域的文章。文中提出了经典的Non-Local模块，通过Self-Attention机制对全局上下午进行建模，有效地捕获长距离的特征依赖。后续许多基于自注意力的方法都是根据Non-Local来改进的。【代码链接可访问**github[21]**】

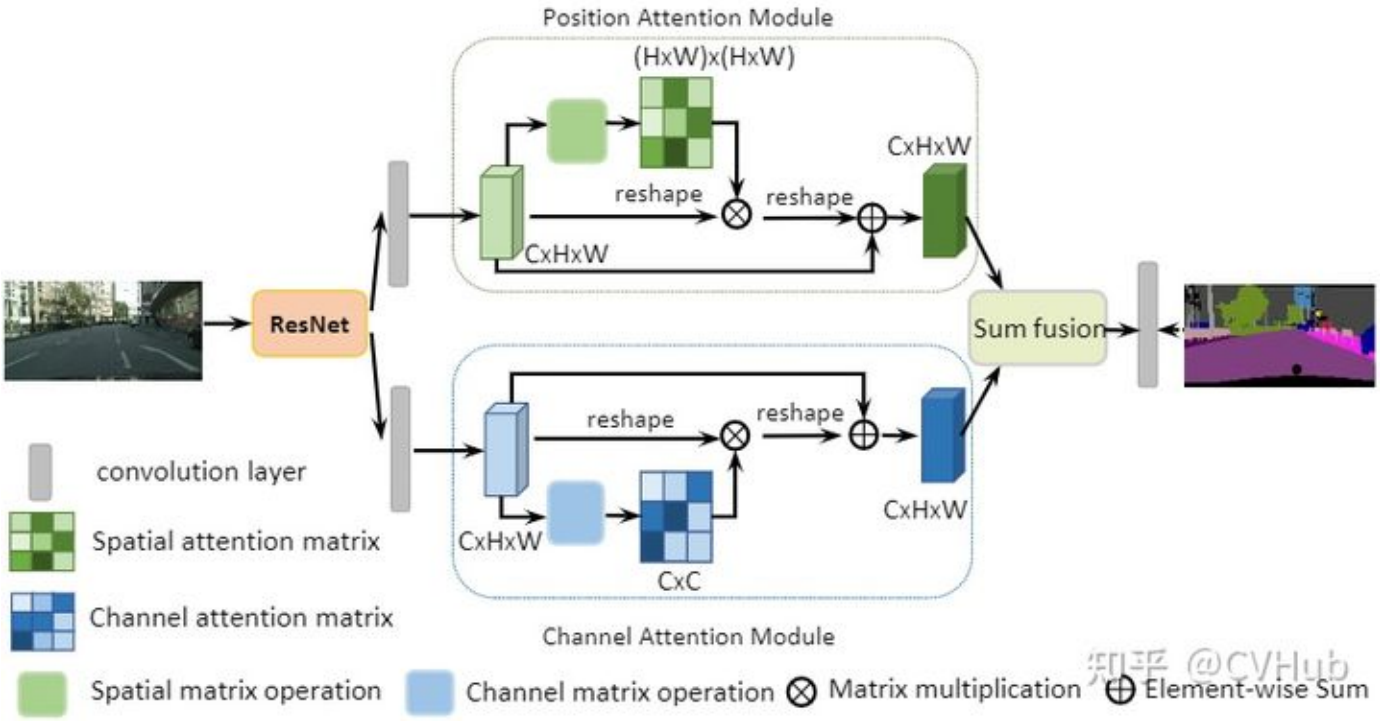


自注意力流程一般是通过将原始特征图映射为三个向量分支，即Query、Key和Value。

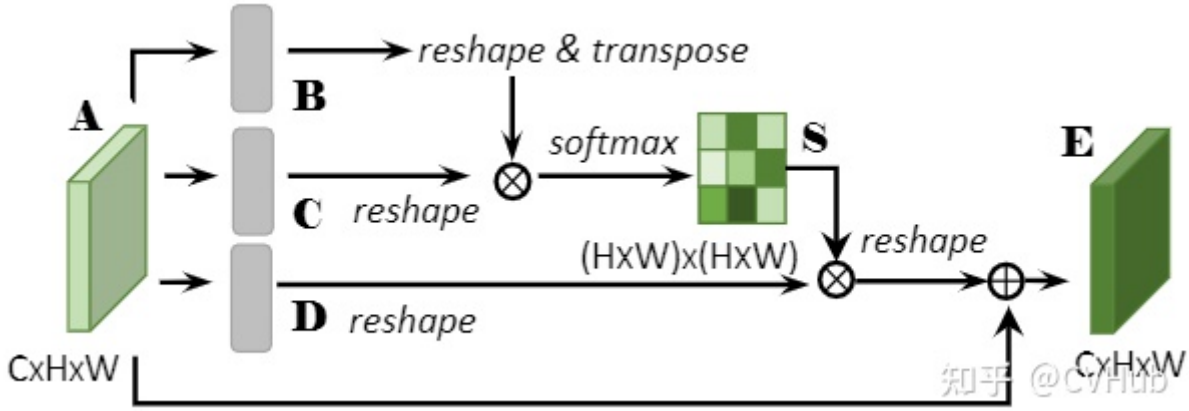
首先，计算Q和K的相关性权重矩阵系数；
其次，通过软操作对权重矩阵进行归一化；
最后，再将权重系数叠加到V上，以实现全局上下文信息的建模。

DA-Net[22]

《DA-Net：Dual Attention Network for Scene Segmentation》发表于CVPR 2019，该论文将Non-local的思想同时引入到了通道域和空间域，分别将空间像素点以及通道特征作为查询语句进行上下文建模，自适应地整合局部特征和全局依赖。【代码链接可访问**github[23]**】

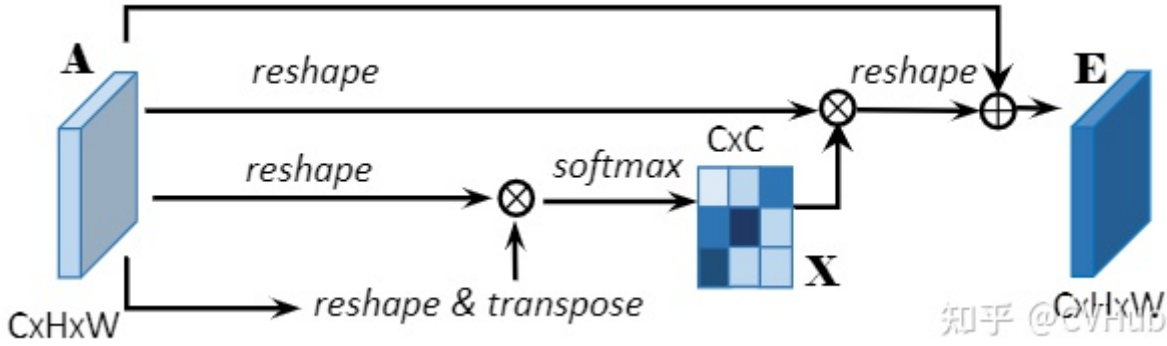


Position Attention Module



> PAM将更广泛的上下文信息编码为局部特征，从而提高了它们的代表性。

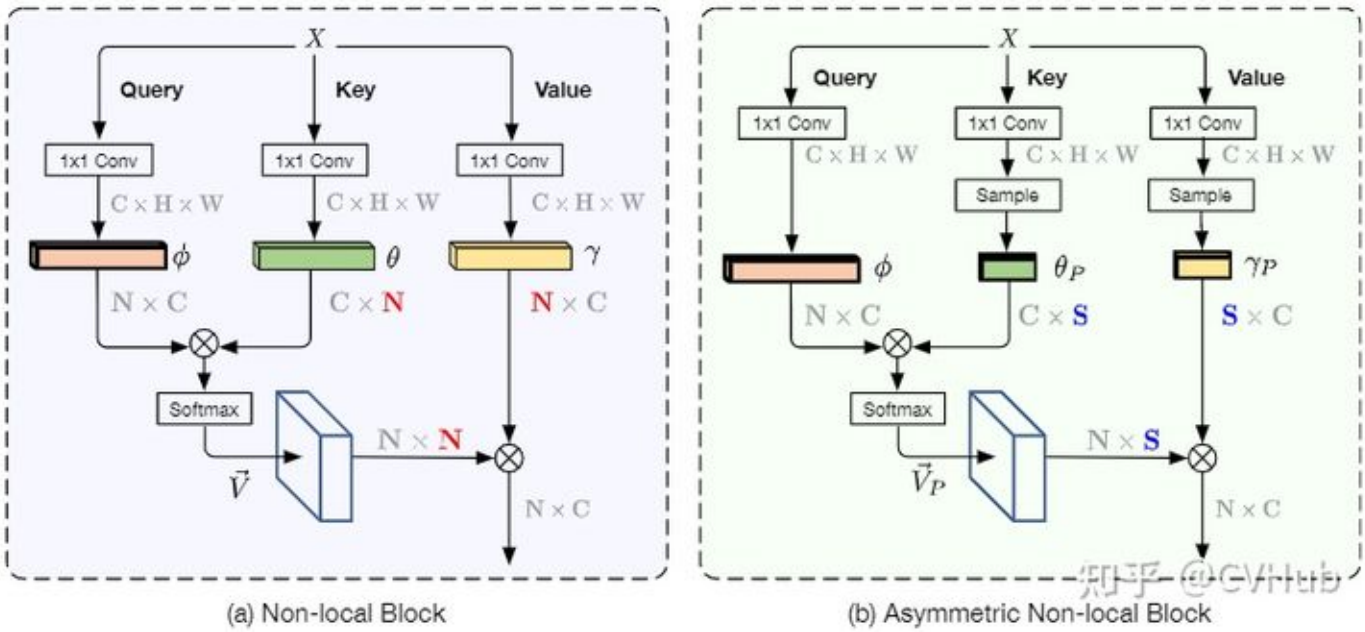
Channel Attention Module



> CAM通过挖掘通道图之间的相互依赖关系，可以强调相互依赖的特征图，改进特定语义的特征表示。

ANLNet[24]

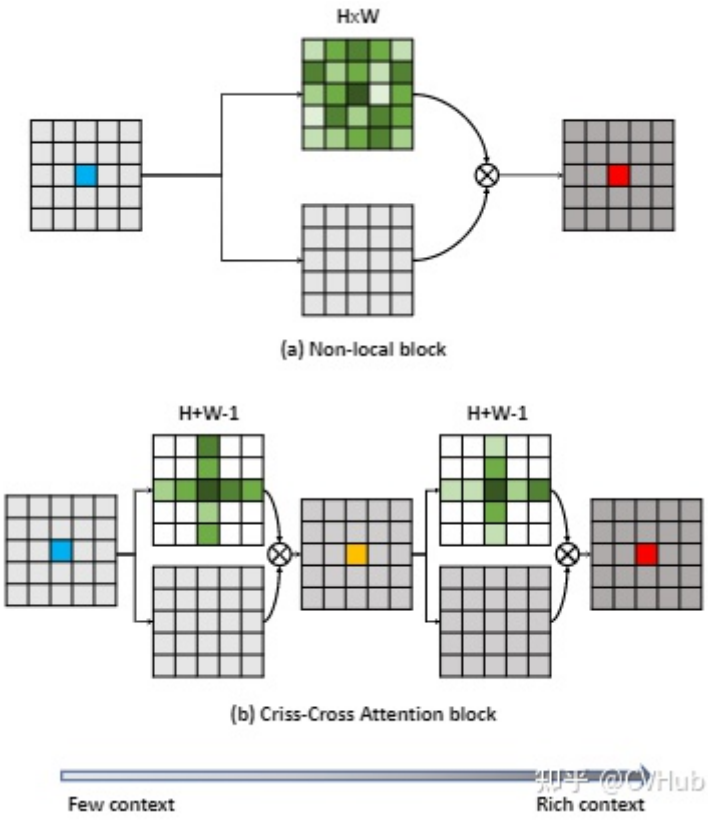
《ANLNet：Asymmetric Non-local Neural Networks for Semantic Segmentation》发表于ICCV 2019，是基于Non-Local的思路往轻量化方向做改进。Non-Local模块是一种效果显著的技术，但同时也受限于过大计算量而难以很好地嵌入网络中应用。为了解决以上问题，ANLNet基于Non-Local结构并融入了金字塔采样模块，在充分考虑了长距离依赖的前提下，融入了不同层次的特征，从而在保持性能的同时极大地减少计算量。【代码链接可访问**github[25]**】



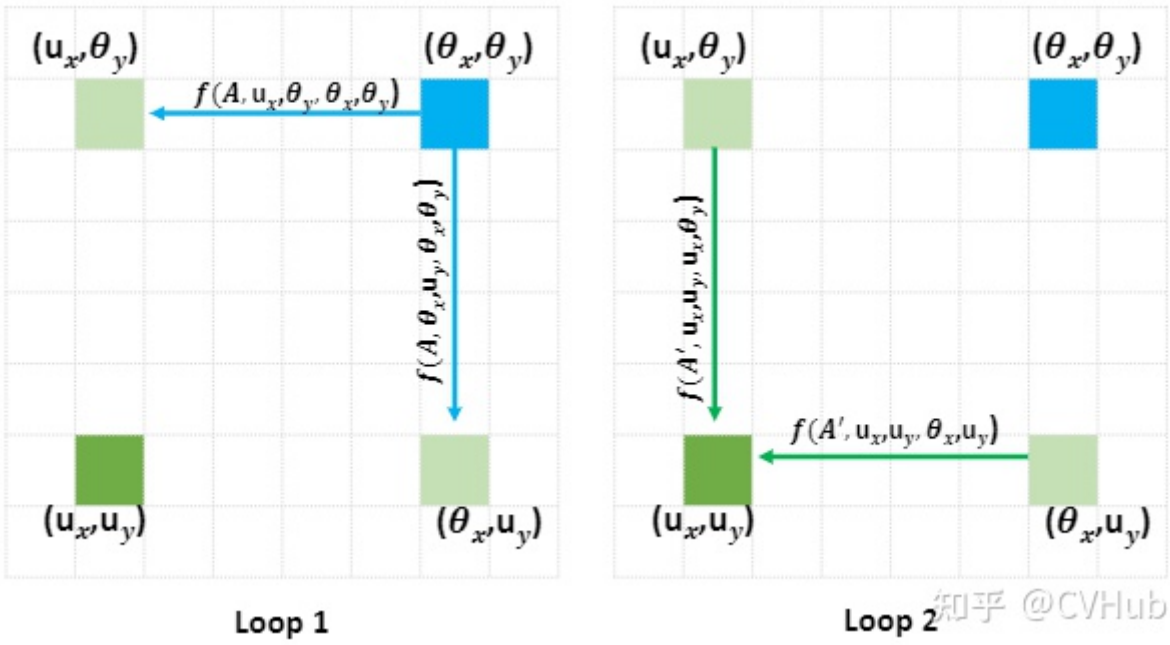
结合SPP和Non-Local，前者从不同大小区域内提取出关键点，后者对这些关键点建模长距离依赖；
Non-Local 需要计算每一个像素点，可以看作是点对点建模；ANL-Net只计算通过SPP提取出的关键点，可以看作是点对区域建模；
AFNB包含两个输入，一个高级特征图，一个低级特征图，分别对应图中的Stage5和Stage4。其中高级特征图作为Query，低级特征图作为Key和Value；
APNB的建模方式与AFNB类似。

CC-Net[26]

《CCNet：Criss-Cross Attention for Semantic Segmentation》发表于ICCV 2019，同样是同Non-Local轻量化的另一种尝试。【代码链接可访问**github[27]**】



与Non-Local的每个查询点需要与其它所有点计算相关性的建模方式不同，CC-Net仅计算该查询点与其在水平和垂直方向上的点相关性，并且重复计算两次就可以获取该点的全局依赖，极大减少计算量。



CC-Net基于Non-Local的思路并利用横纵交叉方式完成对全局上下文建模。时间复杂度从$O(HW*HW)$变为$O(HW*(H+W-1))$，降低了1~2个数量级;

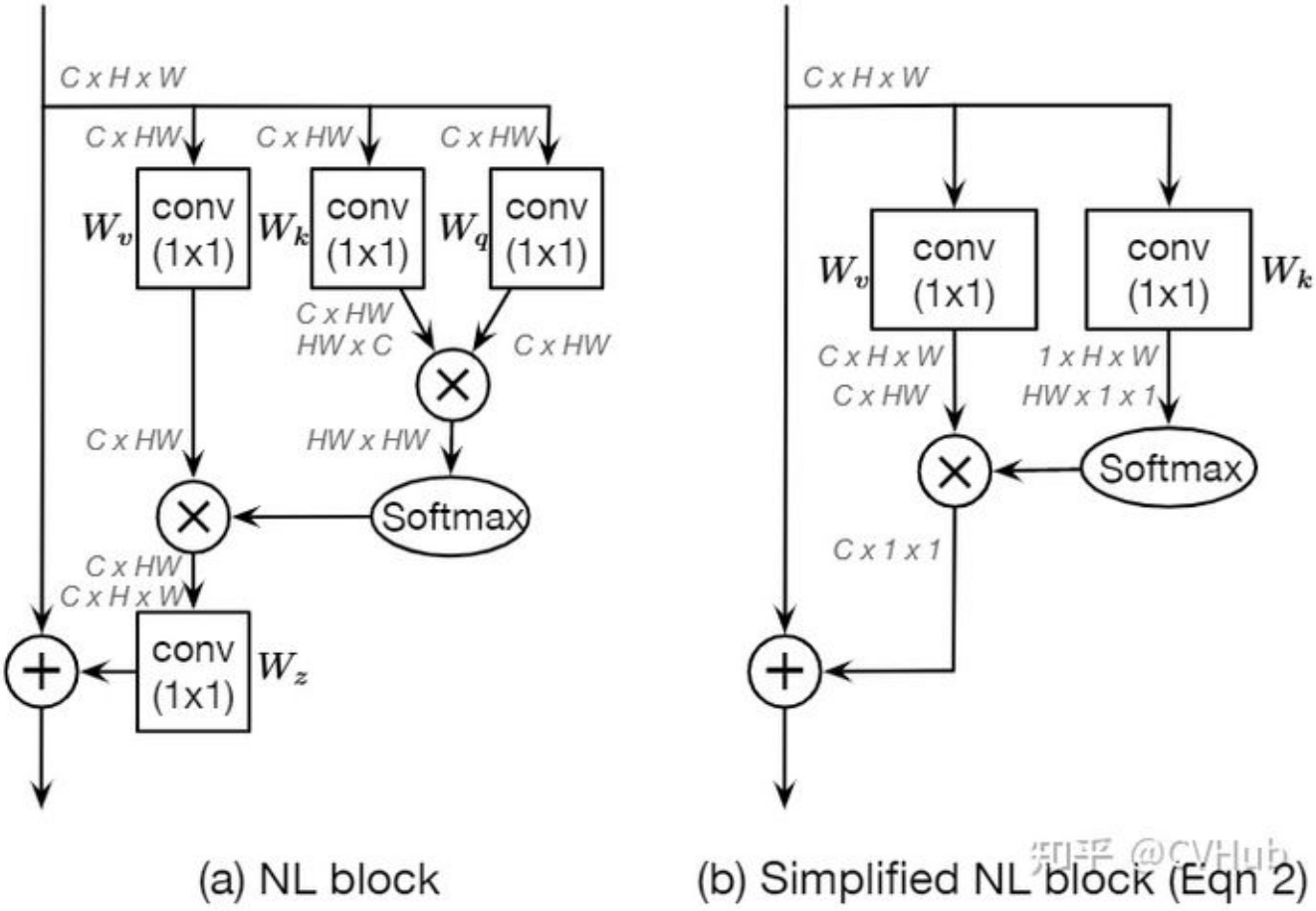
和ANL-Net一样，CC-Net也是点对区域建模一次建模只能获取当前查询点上同一行和一列的信息，二次建模就可以捕捉全局信息;

Loop1：右上方像素点（蓝色）只能捕捉水平（左上）和垂直（右下）方向的像素点信息，但无法捕捉左下角像素点（绿色）信息;

Loop2：右上方像素点（蓝色）再次捕捉左上和右下的像素点信息时，由于在Loop1左上和右下像素点已经捕捉了左下的像素点信息，此时右上像素点就可以间接捕捉到左下像素点信息。

GC-Net[28]

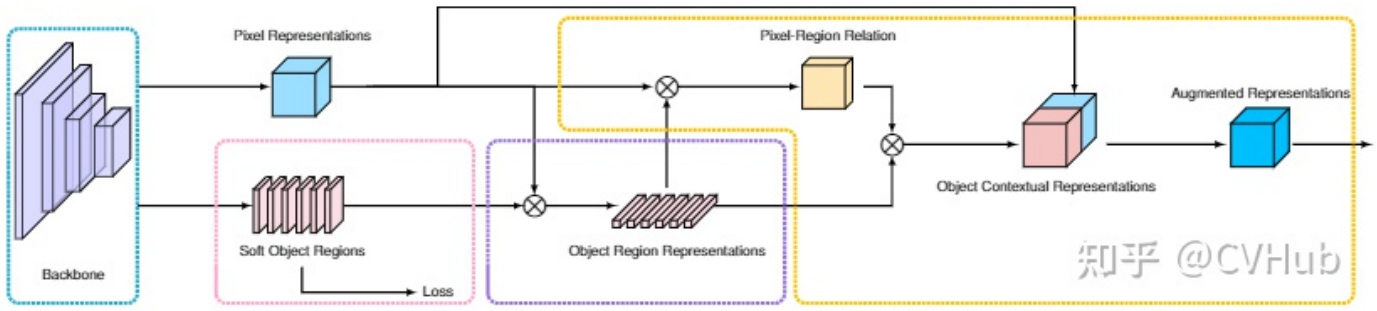
《GCNet: Non-local Networks Meet Squeeze-Excitation Networks and Beyond》发表于ICCV 2019，受SE-Net和Non-local思想的启发提出了一种更简化的空间自注意力模块。Non-local采用Self-attention机制来建模全局的像素对关系，建模长距离依赖，但这种基于全局像素点(pixel-to-pixel) 对的建模方式其计算量无疑是巨大的。SE-Net则利用GAP和MLP完成通道之间的特征重标定，虽然轻量，但未能充分利用到全局上下文信息。因此，作者提出了GC-Net可以高效的建模全局的上下文信息。【代码链接可访问**github[29]**】



类别注意力

OCR-Net[30]

《Object-Contextual Representations for SemanticSegmentation》发表于ECCV 2020，是一种基于自注意力对类别信息进行建模的方法。与先前的自注意力对全局上下文建模的角度（通道和空间）不同，OCR-Net是从类别的角度进行建模，其利用粗分割的结果作为建模的对象，最后加权到每一个查询点，这是一种轻量并有效的方法。【代码链接可访问**github[31]**】

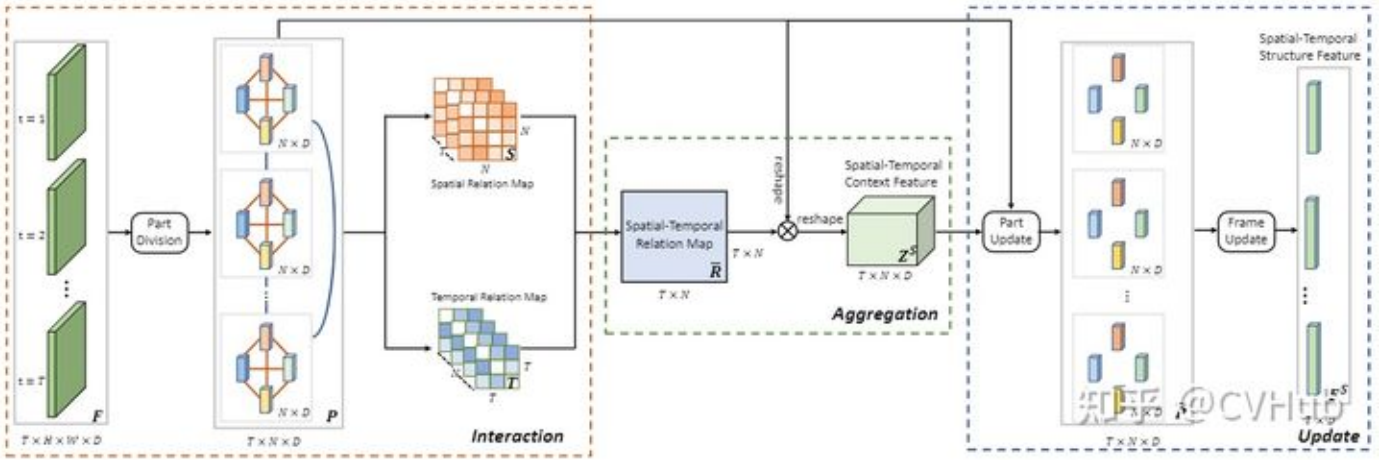


Soft Object Regions 对Backbone倒数第二层所输出的粗分割结果进行监督;
Object Region Representations 融合粗分割和Backbone网络最后一层所输出的高级语义特征图生成对象区域语义，每一条向量代表不同的类别信息;
Pixel-Region Relations 结合最后一层的高级语义特征图以及对象区域语义信息，建模像素与对象区域之间的相关性;
Object Contextual Representations 将像素-对象区域相关性加权到对象区域信息中，完成加权目标类别信息到每一个像素上; 不难发现，这种类别信息的建模方式是完全遵循自注意力机制（Q，K，V）的。

时间注意力

IAU-Net[32]

《IAUnet: Global Context-Aware Feature Learning for Person Re-Identification》发表于IEEE Trans. on Neural Networks and Learning Systems，将自注意力机制的方法扩展到时间维度并应用于行人充识别任务，有效的解决了大多数基于卷积神经网络的方法无法充分对空间-时间上下文进行建模的弊端。【代码链接可访问**github[33]**】

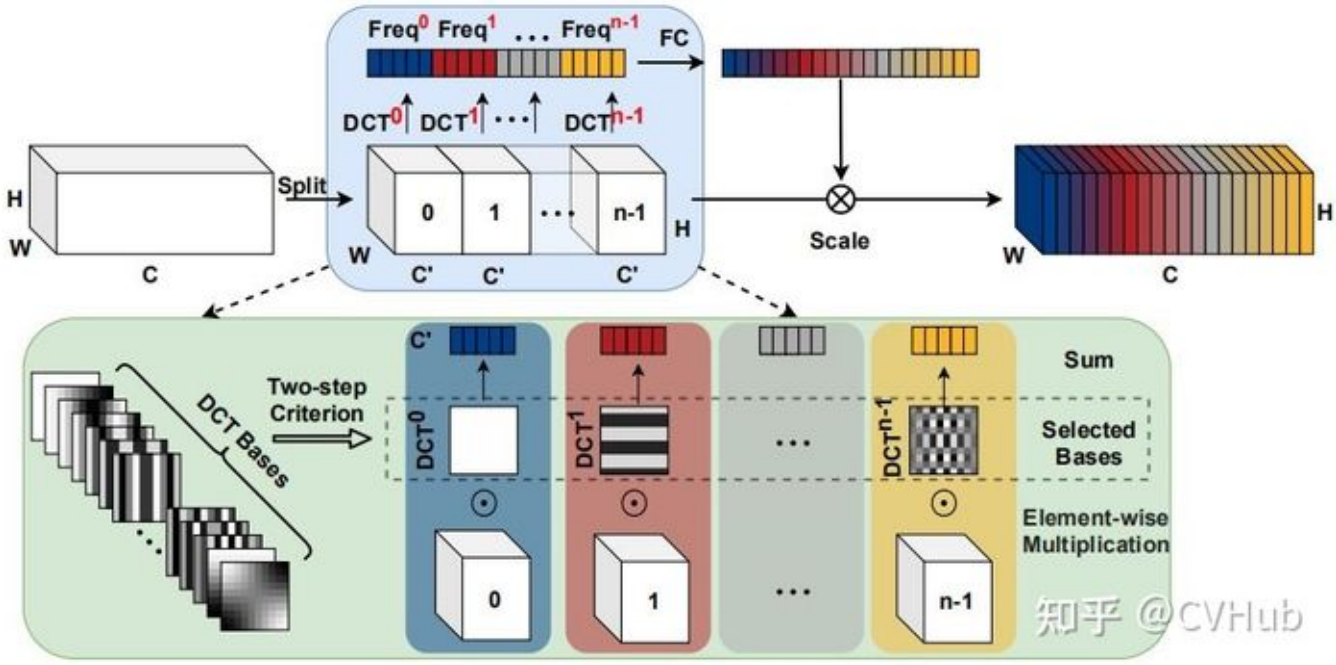


交互聚合模块(Interaction-Aggregation-Update, IAU)同时包含全局空间，时间和频道上下文信息，可用于高性能的reID；
空间-时间IAU(Spatial-Temporal IAU, STIAU)可有效地融合两种类型的上下文依赖；
通道IAU(Channel IAU, CIAU）模块旨在模拟信道特征之间的语义上下文交互，以增强特征表示，尤其是对于小型视觉线索和身体部位。

频率注意力

Fca-Net[34]

《FcaNet: Frequency Channel Attention Networks》截止至目前还未被接收，作者从频域角度切入，证明了GAP是DCT的特例，弥补了现有通道注意力方法中特征信息不足的缺点，将GAP推广到一种更为一般的表示形式，即二维的离散余弦变换(Discrete Cosine Transform, DCT)，并提出了多光谱通道注意力Fca-Net, 通过引入更多的频率分量来充分的利用信息。【代码链接可访问**github[35]**】



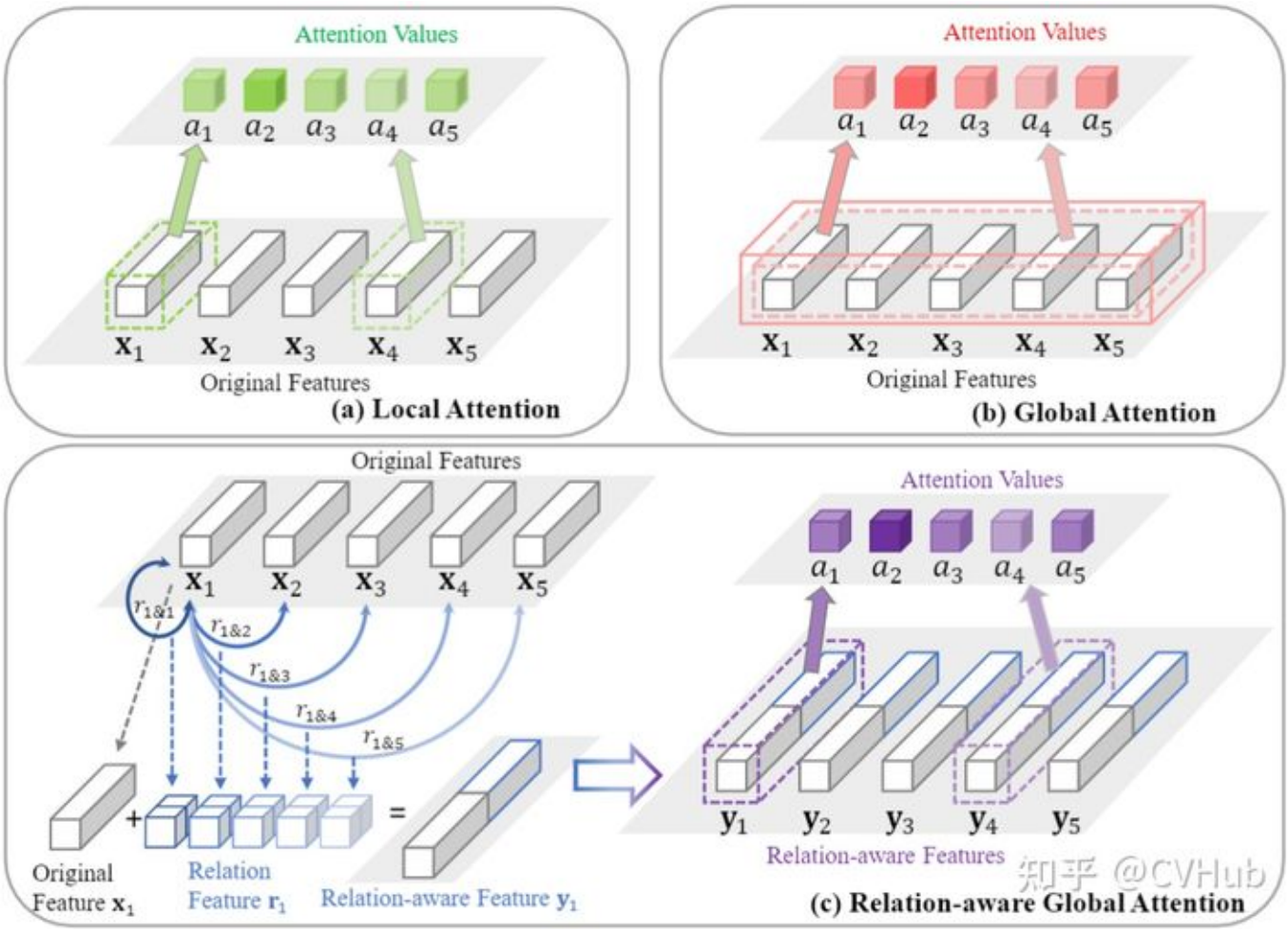
通过探讨使用不同数量的频率分量及其不同组合的影响，提出了选择频率分量的两步准则:

分别计算出通道注意力中每个频率分量的结果；
根据所得结果筛选出Top-k个性能最佳的频率分量。

全局注意力

RGA-Net[36]

《Relation-Aware Global Attention for Person Re-identification》发表于CVPR 2020，作者针对行人重识别任务提出了的一种基于关系感知的全局注意力方法。【代码链接可访问**github[37]**】

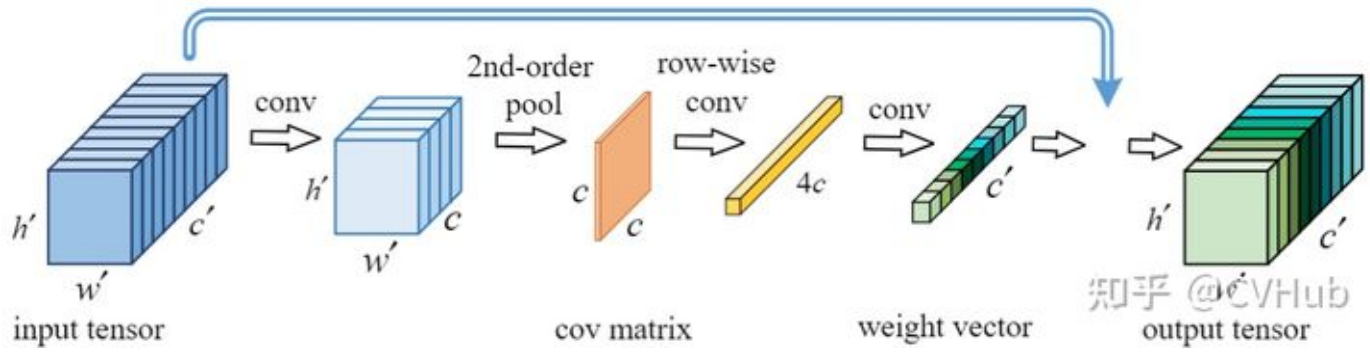


作者认为，要直观地判断一个特征节点是否重要，首先要知道其全局范围的特征信息，以便在决策的过程中更好地探索每个特征节点各自的全局关系，从而学习出更鲁棒的注意力特征。更详细的解读可参考**本文[38]**

高阶注意力

GSoP-Net[39]

《Global Second-order Pooling Convolutional Networks》发表于CVPR 2019，通过应用GSoP可以充分利用到图像中的二阶统计量，以高效的捕获全局的上下文信息。考虑到传统的一阶网络显然不能够有效的表征CNN，因为其目标是表征高维空间中数千个类别的复杂边界，因此学习高阶表示对于增强非线性建模能力至关重要。【代码链接可访问**github[40]**】



从底层到高层逐步引入全局的二阶池化模块，通过对整体图像信息的相关性建模，来捕获长距离的统计信息，充分利用到了图像的上下文信息。与SE等操作提倡的利用二维的GAP操作不同，GSoP通过引入协方差来计算通道之间的关系。具体来说，在利用卷积和池化进行非线性变换以后，该协方差矩阵不仅可以用于沿通道维度进行张量的缩放，也可以用于沿空间维度进行张量缩放。

总结

在计算机视觉领域中，注意力机制大致可分为强注意力和软注意力。由于强注意力是一种随机的预测，其强调的是动态变化，虽然效果不错，但由于不可微的性质导致其应用很受限制。与之相反的是，软注意力是处处可微的，即能够通过基于梯度下降法的神经网络训练所获得，因此其应用相对来说也比较广泛，本文所列举的注意力方法均为软注意力方式。总的来说，目前所有的注意力机制方法大都是基于各个不同的维度利用有限的资源进行信息的充分利用，本质作用是增强重要特征，抑制非重要特征。注意力机制的特点是参数少-速度快-效果好。