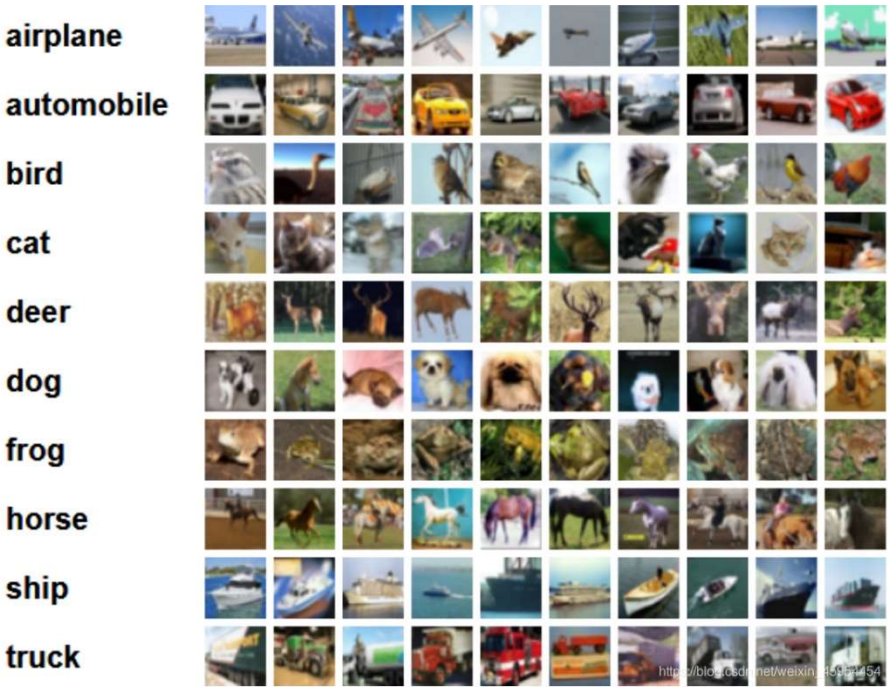
图像分类

一．数据集介绍

CIFAR10 数据集共有 60000 个样本，每个样本都是一张 32*32 像素的 RGB 图像（彩色图像），每个 RGB 图像又分为 3 个通道（R 通道、G 通道、B 通道）。这 60000 个样本被分成了 50000 个训练样本和 10000 个测试样本。

CIFAR10 数据集是用来监督学习训练的，那么每个样本就一定都配备了一个标签值，不同类别的物体用不同的标签值，CIFAR10 中有 10 类物体，标签值分别按照 0~9 来区分,他们分别是飞机（ airplane ）、汽车（ automobile ）、鸟（ bird ）、猫（ cat ）、鹿（ deer ）、狗（ dog ）、青蛙（ frog ）、马（ horse ）、船（ ship ）和卡车（ truck ），如图所示。



二．数据预处理

1. 数据下载

使用 torchvision.datasets 直接下载使用。

```
datasets.CIFAR10('cifar', True,  transform=myTransforms,  download=True)
```

其中第一个参数表示保存路径，第二个参数表示是否为训练集，默认为 True，第三个参数表示进行是否进行图像变换，第四个参数表示是否下载。

2. 数据增强

2.1 原图像大小为 32X32 大小则将其 resize 为 224X224 方便进行提取特征。

2.2 对所有原始图像进行概率为 0.5 的水平翻转，增加泛化能力。

2.3 对所有原始图像进行随机裁剪，首先裁剪下至少覆盖面积为 0.8 的部分然后对该部

分进行放缩到 224X224，减少过拟合。

2.4 对图像进行归一化处理。

```
myTransforms = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(p=0.5),
    transforms.RandomResizedCrop((224,224),scale=(0.8,1.0),ratio=(1.0,1.0)),
    transforms.ToTensor(),
    transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225))])
```

三．模型介绍

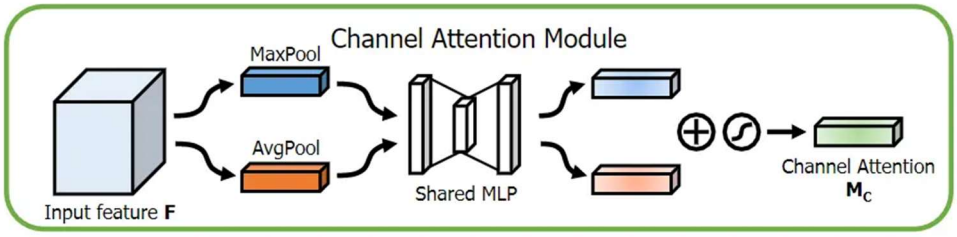
1. Sample ResNet

我们将通道空间注意模块、卷积块注意模块(CBAM)引入到我们的模型中，以增强特征表示，以关注图像中的重要信息。为了实现这一点，我们依次使用通道和空间注意模块。通道注意力是通过使用特征通道之间的关系生成的。该模块压缩输入特征图的空间维度，同时进行平均池化层和最大池化层，然后使用共享 MLP 层来计算输出并添加它们以获得最终的通道映射。

该部分的工作与 SENet 很相似，都是首先将 feature map 在 spatial 维度上进行压缩，得到一个一维矢量以后再进行操作。与 SENet 不同之处在于，对输入 feature map 进行 spatial 维度压缩时，作者不单单考虑了 average pooling，额外引入 max pooling 作为补充，通过两个 pooling 函数以后总共可以得到两个一维矢量。global average pooling 对 feature map 上的每一个像素点都有反馈，而 global max pooling 在进行梯度反向传播计算只有 feature map 中响应最大的地方有梯度的反馈，能作为 GAP 的一个补充。公式:

$$\begin{aligned}\mathbf{M_c}(\mathbf{F}) &= \sigma(MLP(AvgPool(\mathbf{F})) + MLP(MaxPool(\mathbf{F}))) \\ &= \sigma(\mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{avg}})) + \mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{max}}))),\end{aligned} \tag{2}$$

where σ denotes the sigmoid function, $\mathbf{W_0} \in \mathbb{R}^{C/r \times C}$, and $\mathbf{W_1} \in \mathbb{R}^{C \times C/r}$. Note that the MLP weights, $\mathbf{W_0}$ and $\mathbf{W_1}$, are shared for both inputs and the ReLU activation function is followed by $\mathbf{W_0}$.



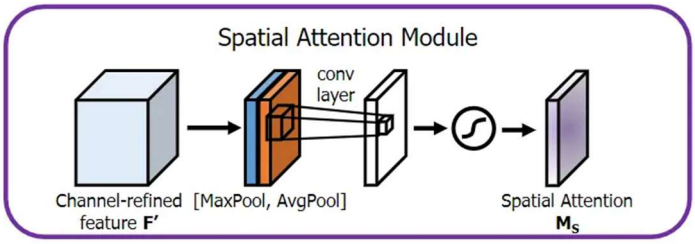
空间注意力是通过使用特征内部空间之间的相互关系生成的。为了计算空间注意力，该模块首先沿通道轴应用平均池化和最大池化操作。然后结合它们产生有效的特征表示。

除了在 channel 上生成了 attention 模型，作者表示在 spatial 层面上也需要网络能明白 feature map 中哪些部分应该有更高的响应。首先，还是使用 average pooling 和 max pooling 对输入 feature map 进行压缩操作，只不过这里的压缩变成了通道层面上的压缩，对输入特征分别在通道维度上做了 mean 和 max 操作。最后得到了两个二维的 feature，将其按通道维度拼接在一起得到一个通道数为 2 的 feature map，之后使用一个包含单个卷积

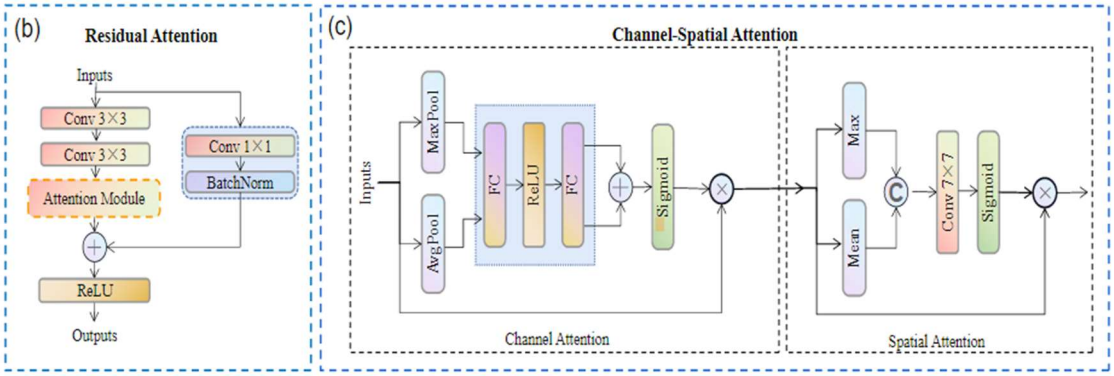
核的隐藏层对其进行卷积操作，要保证最后得到的 feature 在 spatial 维度上与输入的 feature map 一致。公式：

$$\begin{aligned}\mathbf{M_s}(\mathbf{F}) &= \sigma(f^{7\times7}([AvgPool(\mathbf{F}); MaxPool(\mathbf{F})])) \\ &= \sigma(f^{7\times7}([\mathbf{F}^{\mathbf{s}}_{\mathbf{avg}}; \mathbf{F}^{\mathbf{s}}_{\mathbf{max}}])),\end{aligned} \tag{3}$$

where σ denotes the sigmoid function and $f^{7\times7}$ represents a convolution operation with the filter size of 7×7.



我们设计了一个简单的网络，对于图像我们依次使其经过 Conv7X7，BatchNorm，ReLU，MaxPool，ResAttention，AttentionModule，ResAttention，AvgPool，FC，Softmax 最后得到分类结果，其中 ResAttention 和 AttentionModule 如下图所示。

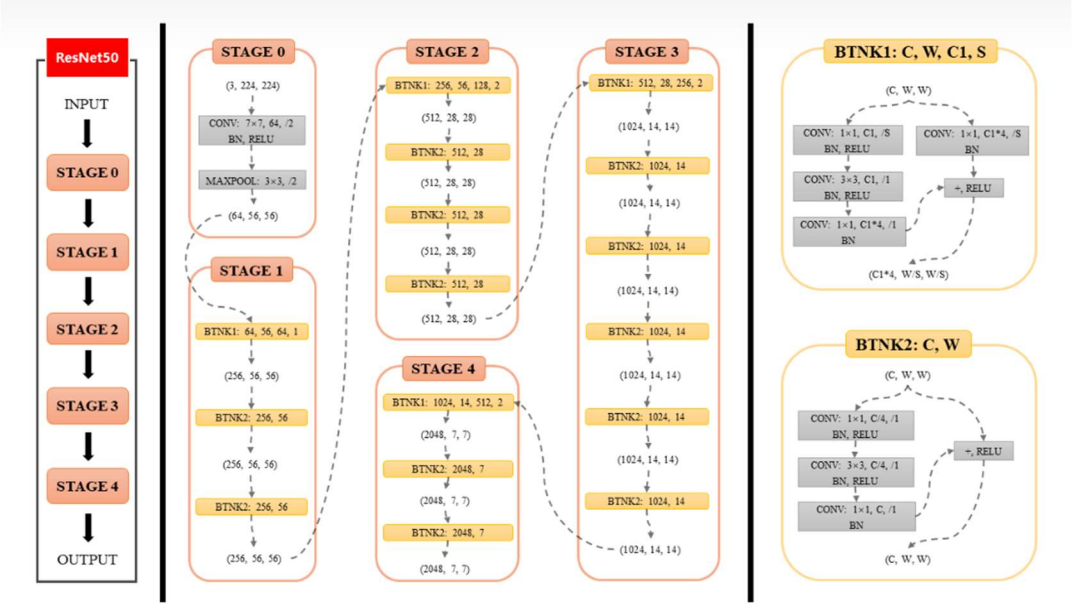


2. ResNet50

在传统的深度神经网络中，随着网络层数增加，模型的复杂度也会增加。然而，随着网络层数的增加，也会遇到一些问题，比如梯度消失和梯度爆炸问题。这些问题会导致模型难以训练，限制了网络的深度。

为了解决这一问题，Resnet（Residual Network）提出了一个创新的设计思想：残差连接（residual connections）。在传统的神经网络中，每一层的输出都是由该层前一层的输出计算得到的。而在 Resnet 中，每一层的输出不仅由该层前一层的输出计算得到，还通过残差连接与该层前一层的输入相加。这个残差连接的操作可以理解为对于网络的改进部分，可以通过学习残差而不是直接学习前一层的输出。这种设计帮助网络学习更加容易，解决了深度神经网络中的梯度问题，使得网络能够更深地进行训练。

Resnet50 是 Resnet 的一个具体实现，它由 50 层堆叠而成，包括卷积层、池化层、全连接层和残差模块。Resnet50 在 ImageNet 数据集上进行了预训练，可以用来进行图像分类任务。同时，Resnet50 还可以用于目标检测和语义分割等计算机视觉任务。其主要特点是具有较高的准确度和良好的泛化能力。下图为其结构所示。



我们分别使用了只修改全连接层层数的预训练模型以及从头训练两种方式进行测试，并进行参数量和准确率对比。通过预训练，在大规模数据上学习到的通用特征表示可以作为初始化参数，加速模型在特定任务上的训练过程。这是因为预训练的参数已经接近最优，并且已经捕捉到了输入数据中的一些通用模式，这样在目标任务上的优化过程更容易收敛。以下是使用预训练模型优点：

提高性能：预训练的模型通常在具体任务上表现更好。这是因为在预训练阶段，模型学习到了大量的数据中的通用特征，这些特征对于许多任务都是有用的。在目标任务中，预训练的模型能够更好地利用这些通用特征，从而提高性能。

解决数据不足问题：在许多实际任务中，数据往往是有限的，特别是深度学习模型需要大量的数据进行训练。通过预训练，可以利用大规模数据集进行通用特征的学习，然后将这些学到的特征应用于目标任务，从而克服数据不足的问题。

迁移学习：预训练的模型可以作为迁移学习的基础。将预训练模型的参数应用于新的相关任务，可以利用预训练模型在大规模数据上学习到的通用特征，从而在新任务上提高性能。这对于目标任务数据较少的情况下特别有用。

提高泛化能力：预训练有助于提高模型的泛化能力，即在未见过的数据上表现良好。通过在大规模数据上学习通用特征，模型更能够从输入数据中捕捉普遍的模式，而不是过度拟合训练集。

3. ResNet Attention

我们在固定 ResNet50 参数的同时，在其 STAGE4（layer4）后添加了一个卷积层将其映射到低维同时添加了一层 ResAttention 对得到的特征图更好的提取信息同时增加了参数的规模防止欠拟合。同时使用了预训练模型以及从头训练两种方式。其结构如下所示：

```
ResAttention(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (layer1): ……
```

```
  (layer2): ……
  (layer3): ……
  (layer4): ……
  Conv2d(2048, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  Residual_attention(
      (conv1): Conv2d(256, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (conv2): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (conv3): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (relu): ReLU()
      (bn): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (attention): Channel_Spatial_Attention(
        (channel): ChannelAttention(
          (avg_pool): AdaptiveAvgPool2d(output_size=1)
          (max_pool): AdaptiveMaxPool2d(output_size=1)
          (fc): Sequential(
            (0): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): ReLU()
            (2): Conv2d(32, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
          )
          (sigmoid): Sigmoid()
        )
        (Spatial): SpatialAttention(
          (conv1): Conv2d(2, 1, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), bias=False)
          (sigmoid): Sigmoid()
        )
      )
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Sequential(
    (0): Linear(in_features=64, out_features=10, bias=True)
    (1): LogSoftmax(dim=1)
  )
)
```

四．训练设置

Epoch 为 200；学习率为 0.01，并设置衰减；batch_size 为 128，使用 GPU 训练。

使用交叉熵损失函数，使用 momentum=0.9 的 SGD 优化器，每 5 个 epoch 进行一次

训练集测试精度和损失，记录最优准确率并保存模型参数。

对于 ResNet50，使用预训练模型进行微调，冻结除了全连接层之外的所有参数，修改最后全连接层为 Linear(in_features=2048, out_features=10, bias=True)。

五 . 精度测试

Model	Train accuracy	Test accuracy	parameters
Sample ResNet	0.7605	0.6609	230000
ResNet50	0.8647	0.8443	20490
ResNet Attention	0.9457	0.8854	5108884
ResNet50 Scratch	0.9914	0.9613	23528522
ResNet Attention Scratch	0.9952	0.9631	28616876

六 . 程序运行截图: