SE_ResNet的pytorch实现

残差块：

```
class Resiual_block(nn.Module):
    def __init__(self, in, middle_out, out, kernel_size=3, padding=1):
        self.out_channel = middle_out
        super(Resiual_block, self).__init__()

        self.shortcut = nn.Sequential(
            nn.Conv2d(nin, nout, kernel_size=1),
            nn.BatchNorm2d(nout)
        )
        self.block = nn.Sequential(
                                nn.Conv2d(in, middle_out,
kernel_size=kernel_size, padding=padding),
                                nn.BatchNorm2d(middle_out),
                                nn.ReLU(inplace=True),
                                nn.Conv2d(middle_out, out,
kernel_size=kernel_size, padding=padding),
                                nn.BatchNorm2d(nout))


    def forward(self, input):
        x = self.block(input)
        return nn.ReLU(inplace=True)(x + self.shortcut(input))
```

注意 在input与输出相加前，这里需要一个shortcut层来**调整input的通道大小**

SE-Net：

```
class SE(nn.Module):
    def __init__(self, in, middle_out, out, reduce=16):
        super(SE, self).__init__()
        self.block = Resiual_block(in, middle_out, out)

        self.shortcut = nn.Sequential(
            nn.Conv2d(nin, nout, kernel_size=1),
            nn.BatchNorm2d(nout)
        )
        self.se = nn.Sequential(
                                nn.Linear(out, out // reduce),
                                nn.ReLU(inplace=True),
                                nn.Linear(out // reduce, out),
                                nn.Sigmoid())


    def forward(self, input):
        x = self.block(input)
        batch_size, channel, _, _ = x.size()
        y = nn.AvgPool2d(x.size()[2])(x)
        y = y.view(y.shape[0],-1)
        y = self.se(y).view(batch_size, channel, 1, 1)
        y = x * y.expand_as(x)
```

```
        out = y + self.shortcut(input)
        return out
```

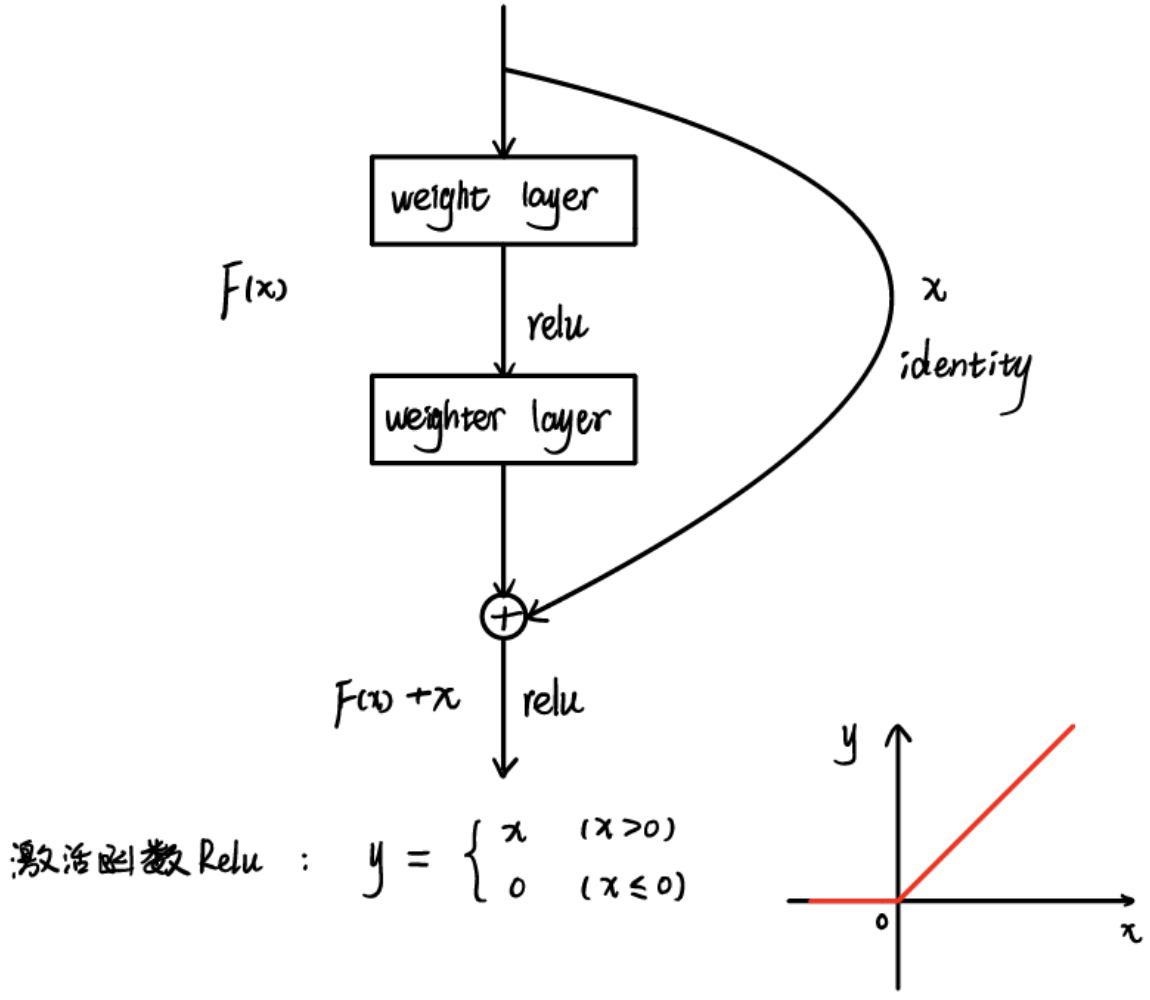
注意 我们使用平均池化层进行下采样，**这里的kernel_size是动态的**

ResNet

1、为什么网络层数越深，误差越大？在ResNet中作者提出了两个问题：

- 随着网络层数的加深，梯度消失和梯度爆炸以及梯度爆炸这个问题会越来越明显（假设在反向传播过程中，每一层的误差梯度是一个小于一的数，在反向传播过程中，每向前传播一次都乘以一个小于一的系数，那么当网络越来越深的时候所乘的小于一的系数就越来越多，梯度就会越来越趋近于零，造成梯度消失。相反，再反向传播过程中，如果每一层的梯度都是一个大于一的数，则在反向传播过程中梯度会越来越大，即梯度爆炸）。梯度消失和梯度爆炸问题的解决通常是通过以下三种方式解决：
    1. 对数据进行标准化处理
    2. 权重初始化
    3. Batch Normalization（放弃dropout）
- Degredation problem：即在解决了梯度消失和爆炸的问题之后，依然会存在网络层数加深，但是效果依然不好的问题。ResNet提出了残差结构，通过残差结构，解决退化问题。

2、ResNet网络中残差块的结构：

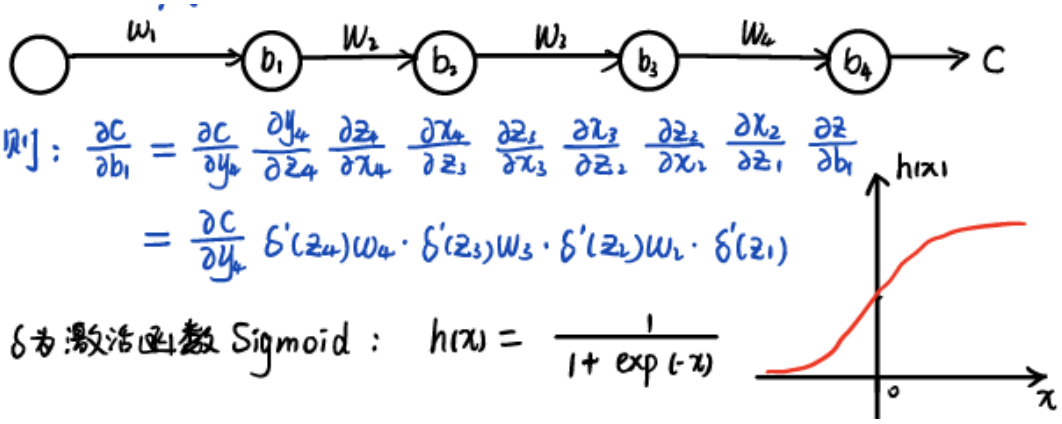


3、ResNet原论文中作者遇到的问题：

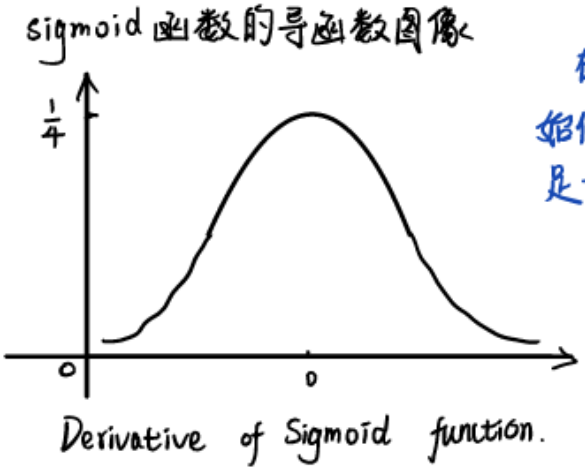
1、通过简单堆叠卷积层与池化层而形成的深层网络存在：层数越深，网络效果越差的问题。作者提出了1处的两个问题。关于梯度消失或者梯度爆炸的本质原因：网络太深，网络权值更新不稳定，本质是因为梯度反向传播过程中的连乘效应（链式法则）。关于梯度消失和梯度爆炸的直观解释：

- 以三个隐藏层为例：

  $y_i = \delta(z_i) = \delta(w_i x_i + b_i)$,$C$为代价函数，根据链式法则，可以得到图中表达式



- Sigmoid函数的导函数图像为：导函数最大值为$1/4$



在使用标准方法对网络进行初始化时，通常将权重初始化为满足均值为0，方差为1的高斯分布.

这种衰减是指数级的衰减.

可见，δ'(x) 的最大值为 1/4，而我们初始化网络权值 |w| 通常都小于1.

因此，|δ'(z)w| ≤ 1/4，故由公式知：

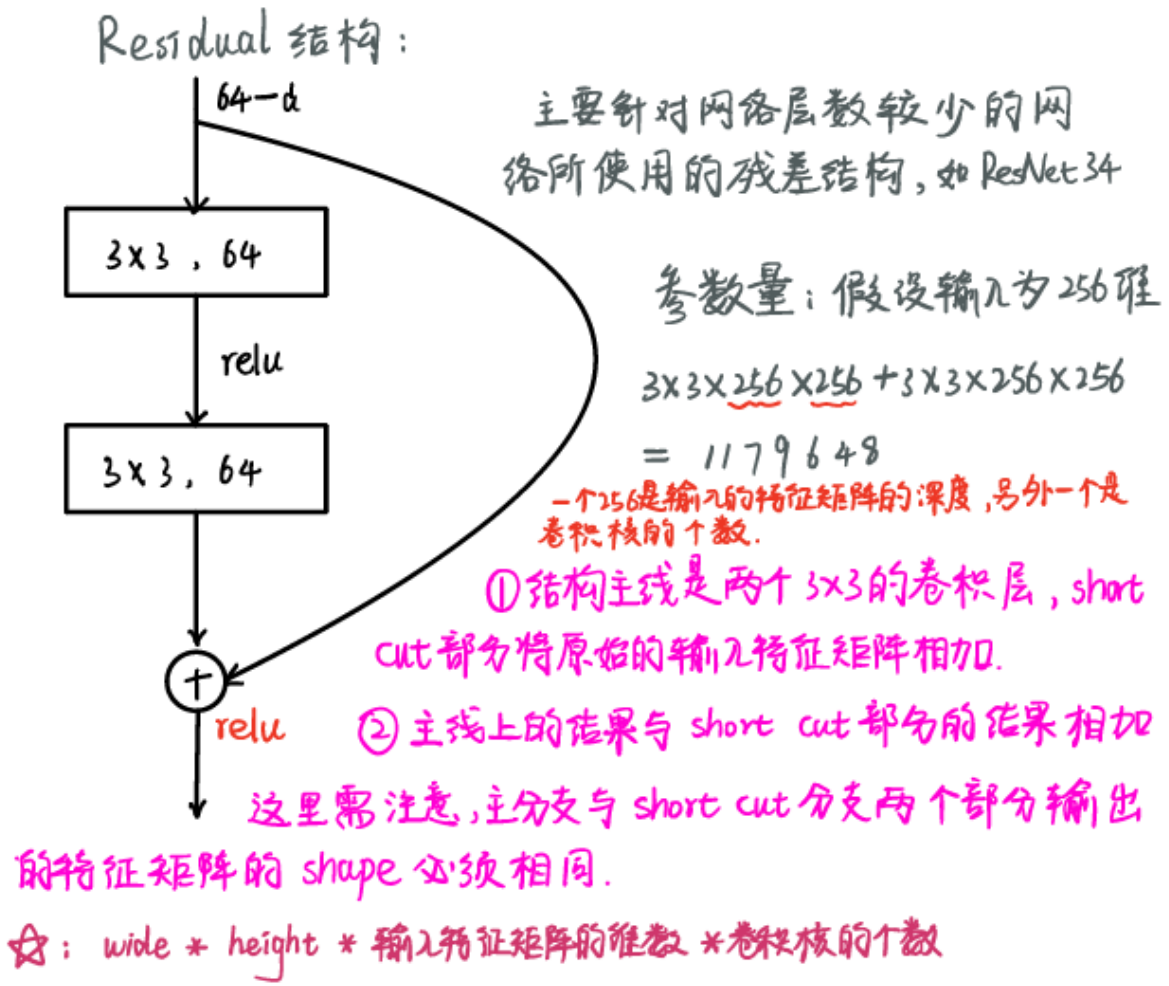
$$\frac{\partial C}{\partial b_1} = \frac{\partial C}{\partial y_4}\underbrace{\delta'(z_4)w_4}_{\le \frac{1}{4}}\cdot\underbrace{\delta'(z_3)w_3}_{\le \frac{1}{4}}\cdot\underbrace{\delta'(z_2)w_2}_{\le \frac{1}{4}}\cdot\underbrace{\delta'(z_1)}_{\le \frac{1}{4}}$$

使用链式法则求导，层数越深，求导结果越小，不断趋近于零，造成梯度消失

梯度爆炸：当权重参数 w 被初始化为一个较大值时，当 |δ'(z)w| > 1 时，由链式法则，代价函数的梯度会随网络层数的加深呈指数级增长

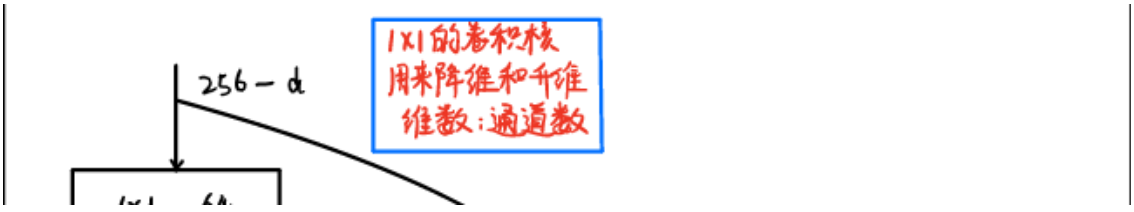
4、ResNet残差结构：

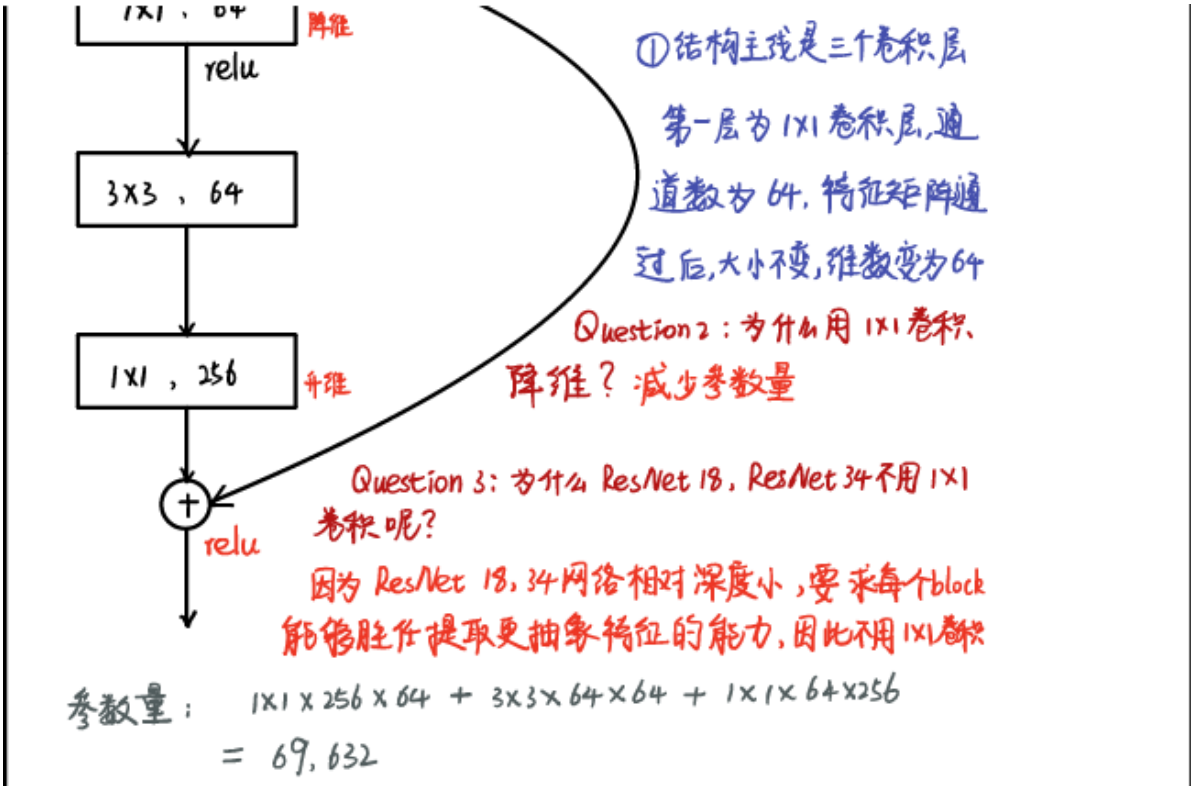
- 主要分为两种：针对相对浅层的网络如ResNet18，ResNet34和针对相对深层的网络ResNet50、ResNet101以及ResNet152等
- 残差结构①：



此处参数量计算的解释：两个256是指，假设输入特征矩阵的维度是256，主线上卷积核的个数也得对应是256.

- 残差结构②：此种结构主要是针对ResNet50、Resnet101、ResNet152等深度网络使用的。





此处参数计算量解释：输入特征图的维度是256，但第一个卷积层的卷积核的数量是64

5、ResNet论文中的结构图：

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

此表中的 conv2_x 、conv3_x 、conv4_x 以及 conv5_x 表示一系列的残差结构，以 ResNet 34 为例.

Conv2_x：包含 3个残差层

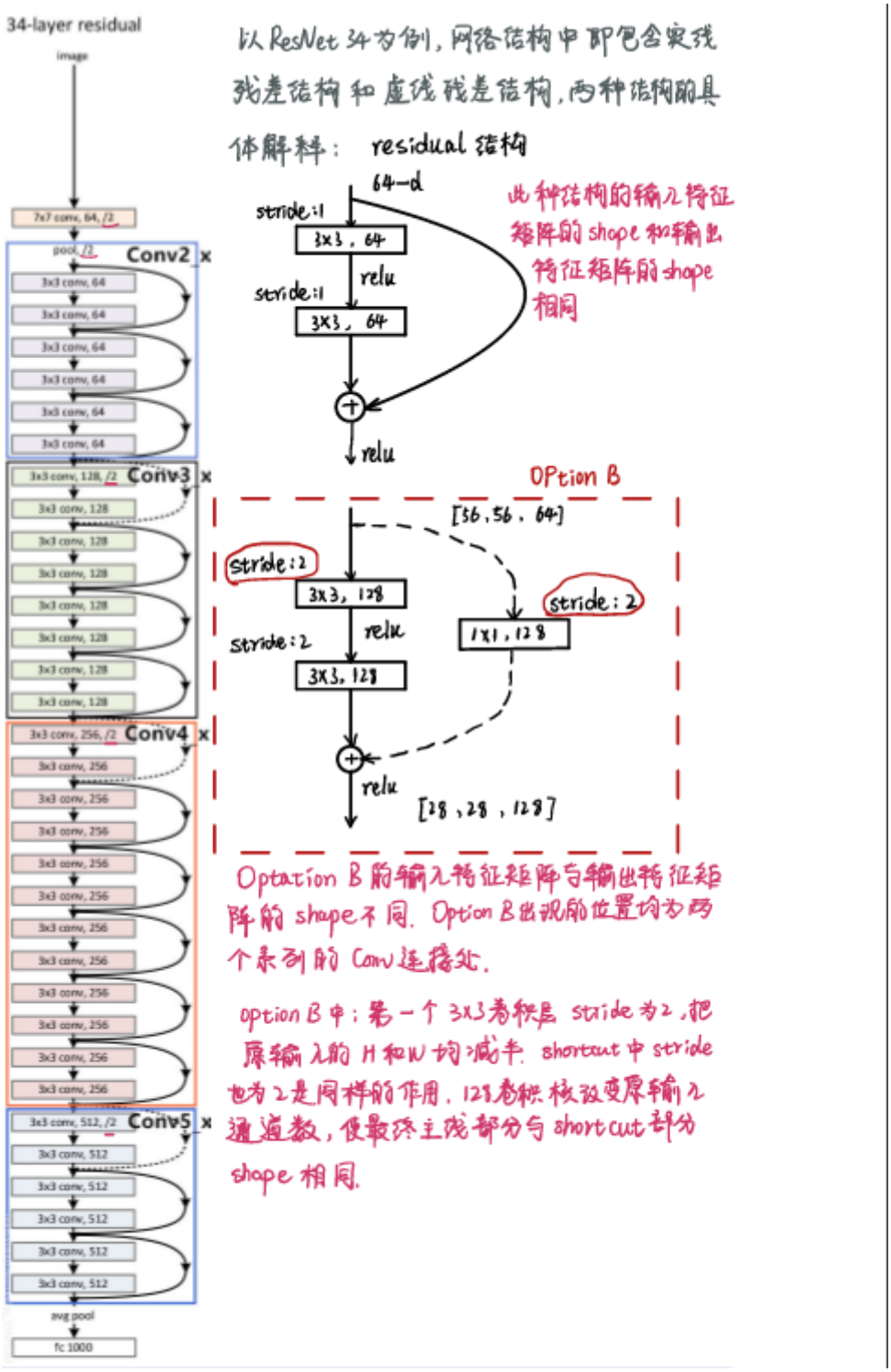
Conv3_x：包含 4个残差层

Conv4_x：包含 6个残差层

Conv5_x：包含 3个残差层

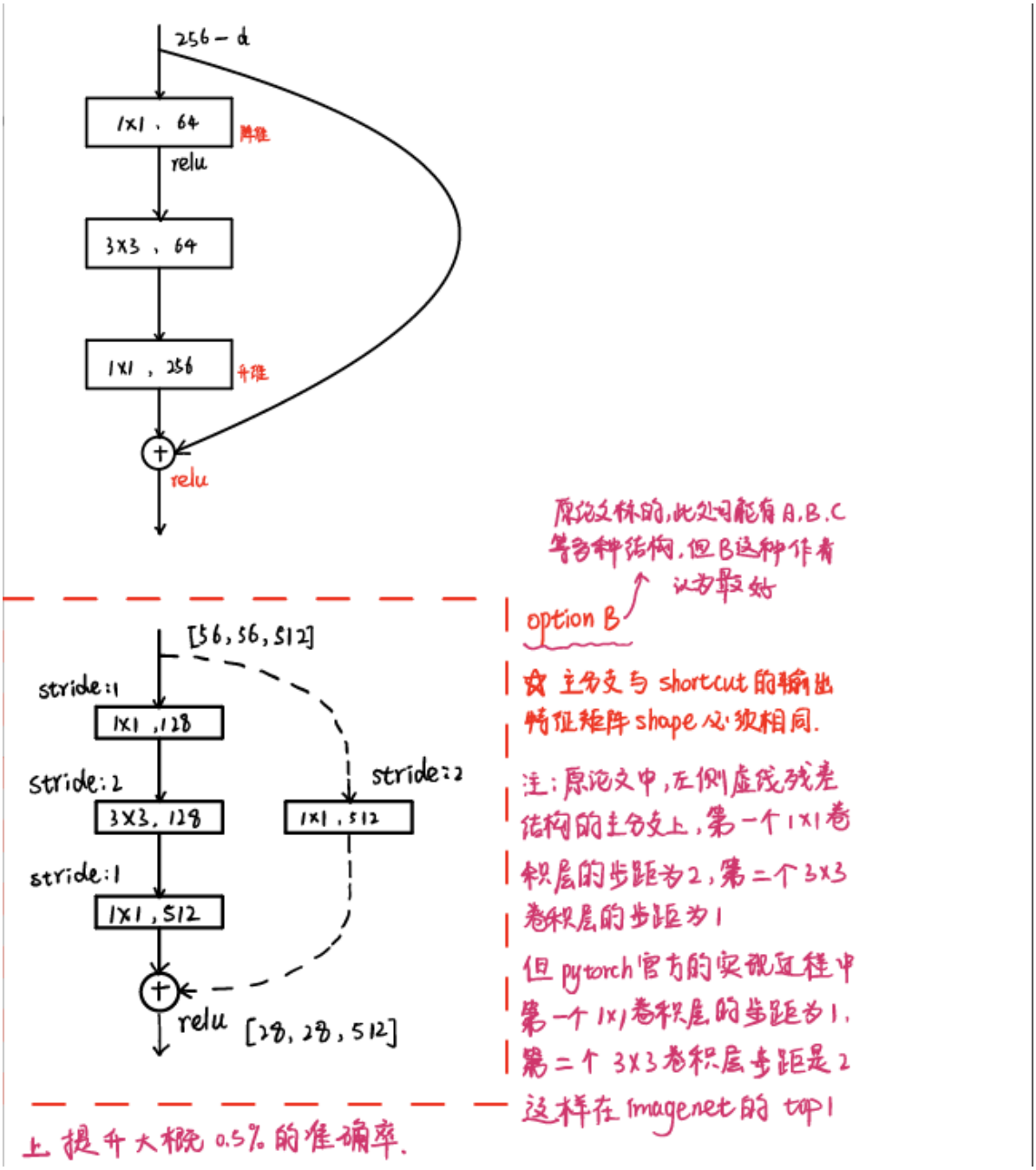
	layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
	conv1	112×112	7×7, 64, stride 2				
			3×3 max pool, stride 2				
layer1	conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
layer2	conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
layer3	conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
layer4	conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
		1×1	average pool, 1000-d fc, softmax				
	FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

6、以ResNet34为例：



注意：Conv3_x、Conv4_x、Conv5_x各个部分之间相连接的第一个部分均为optionB这种残差结构，因为通道数不匹配，但Conv2_x处直接使用第一种结构，因为在这一层输入与输出的通道数是相同。

7、ResNet50、ResNet101、ResNet152基础结构：



- 虚线的残差结构键输入图片的高和宽以及深度等都做变化，而实线部分的输入特征矩阵的和输出特征矩阵是一样的。
- Conv3_x、Conv4_x、Conv5_x对应的一系列残差结构的第一层均为虚线结构，因为第一层必须将上一层的特征矩阵的高和宽以及深度调整为当前层特征矩阵所需的高和宽以及深度（原文中：Downsampling is performed by Conv3_1，Conv4_1，Conv5_1 with a stride of 2）.

Batch Normalization

1、Batch Normalization的目的是将一批（Batch）feature map满足均值为0，方差为1的分布规律，即将一批数据对应的feature map（即特征矩阵）的每一个维度也就是每一个channel对应的维度满足均值为0，方差为一的分布规律。

2、通常情况下，在将图像输入到网络之前，会对图像进行预处理。即将输入图像数据调整到满足某一分布规律，这样可以加速网络训练。但原始图像在通过卷积层之后，得到的feature map却不一定满足某一分布规律，因此考虑使用Batch Normalization调整feature map，使每一层的feature map都能满足均值为0和方差为1的分布。

3、

3. 原论文 " For a layer with d-dimensional input $x=(x^{(1)},\cdots,x^{(d)})$, we will normalize each dimension

$$\hat{x}^{(k)}=\frac{x^{(k)}-E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}}$$

即对于一个拥有d维的输入x，我们将对它的每一个维度进行处理. 假设我们输入的x是RGB三通道的彩色图像，那么d就是输入图像的channel即 $d=3$，$x=(x^{(1)},x^{(2)},x^{(3)})$，其中$x^{(1)}$就代表R通道对应的特征矩阵，依此类推，标准化处理也就是分别对我们的R通道，G通道，B通道进行处理.

计算详解：

Input : Values of x over a mini-batch : $B=\{x_1,\cdots,m\}$;
Parameters to be learned : γ,β

Output: $\{y_i=BN_{\gamma,\beta}(x_i)\}$

$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m}x_i \quad // \text{mini-batch mean}$$

$$\delta_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i-\mu_B)^2 \quad // \text{mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i-\mu_B}{\sqrt{\delta_B^2+\varepsilon}} \quad // \text{normalize}$$

$$y_i \leftarrow \gamma\hat{x}_i+\beta \equiv BN_{\gamma,\beta}(x_i) \quad // \text{scale and shift}$$

γ：调方差　β：调均值

满足0-1分布的效果不一定是最好的，因此设置两个参数γ和β进行调整，γ和β是通过反向传播学习得到的. 而均值和方差是通过一批批数据计算统计得到的.

- μ δ^2 在正向传播过程中统计得到
- γ β 在反向传播过程中训练得到

注：此处 β 和 γ 的初始值分别为0和1.