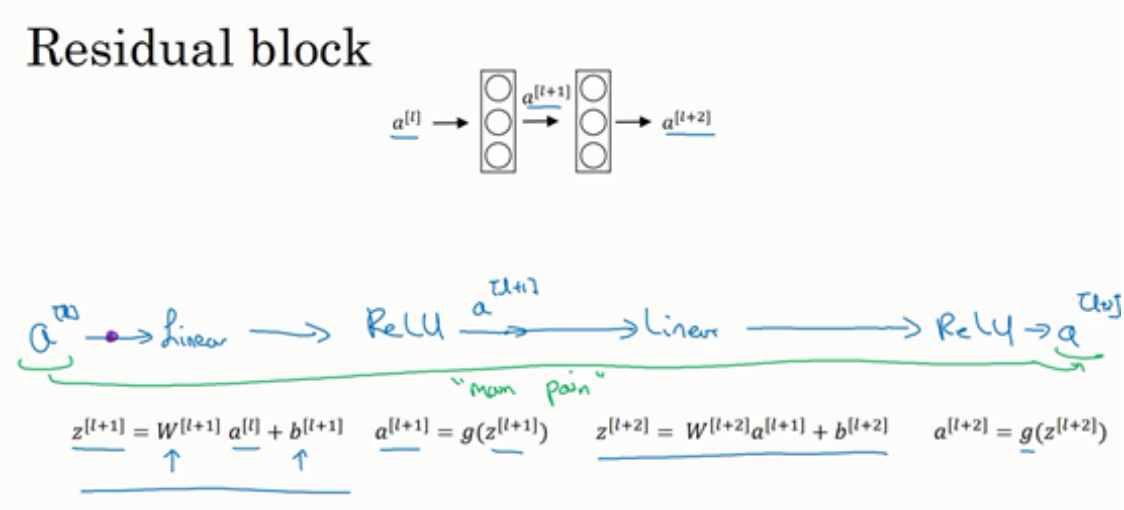
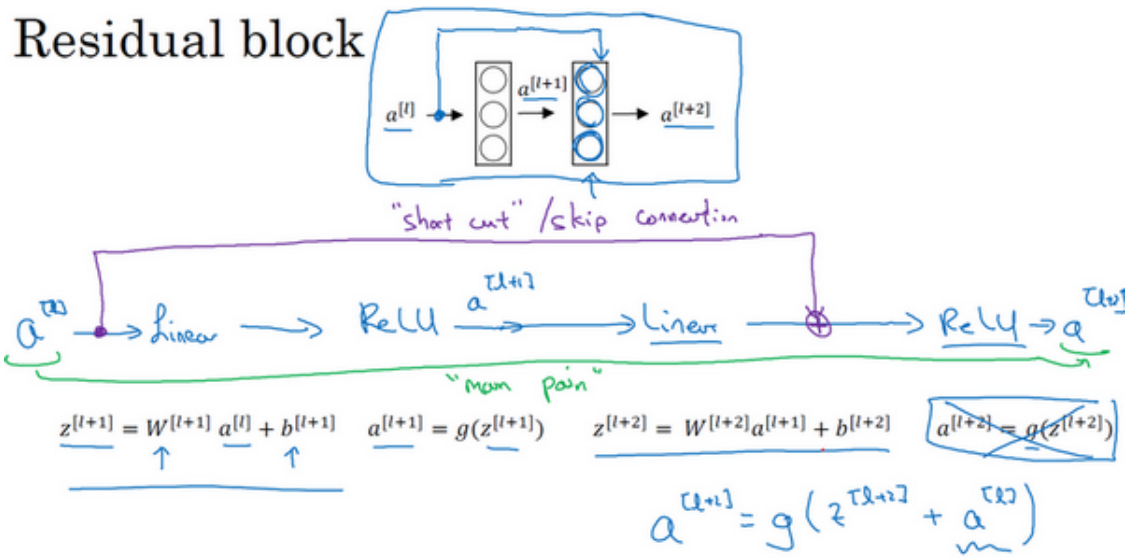
2.3 残差网络(ResNets)(Residual Networks (ResNets))

非常非常深的神经网络是很难训练的，因为存在梯度消失和梯度爆炸问题。这节课我们学习跳跃连接（Skip connection），它可以从某一层网络层获取激活，然后迅速反馈给另外一层，甚至是神经网络的更深层。我们可以利用跳跃连接构建能够训练深度网络的 **ResNets**，有时深度能够超过 100 层，让我们开始吧。

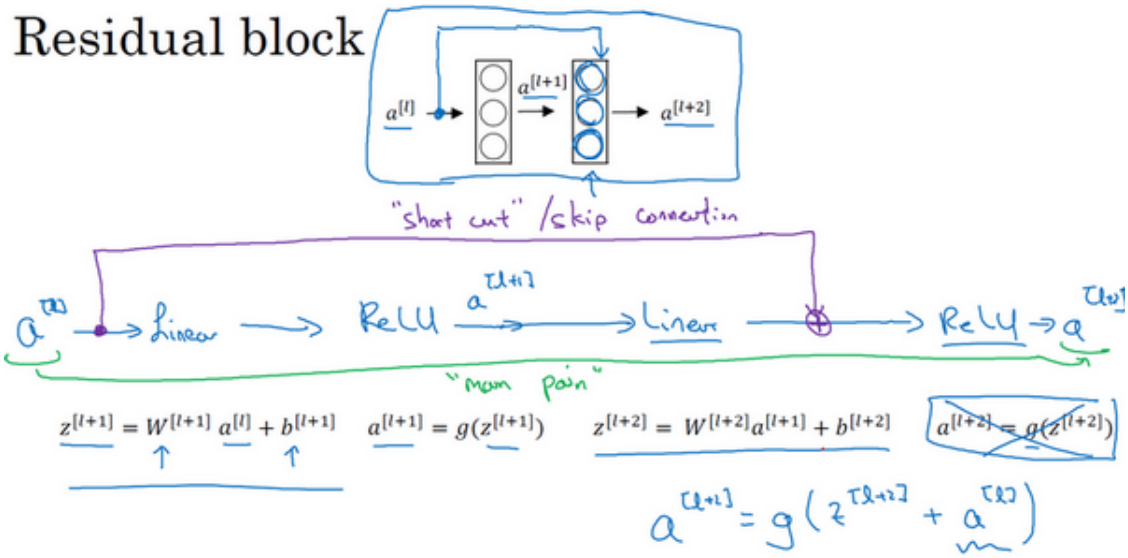
ResNets 是由残差块（Residual block）构建的，首先我解释一下什么是残差块。



这是一个两层神经网络，在L层进行激活，得到$a^{[l+1]}$，再次进行激活，两层之后得到$a^{[l+2]}$。计算过程是从$a^{[l]}$开始，首先进行线性激活，根据这个公式：$z^{[l+1]} = W^{[l+1]}a^{[l]} + b^{[l+1]}$，通过$a^{[l]}$算出$z^{[l+1]}$，即$a^{[l]}$乘以权重矩阵，再加上偏差因子。然后通过 **ReLU** 非线性激活函数得到 $a^{[l+1]}$，$a^{[l+1]} = g(z^{[l+1]})$ 计算得出。接着再次进行线性激活，依据等式 $z^{[l+2]} = W^{[2+1]}a^{[l+1]} + b^{[l+2]}$，最后根据这个等式再次进行 **ReLu** 非线性激活，即$a^{[l+2]} = g(z^{[l+2]})$，这里的g是指 **ReLU** 非线性函数，得到的结果就是$a^{[l+2]}$。换句话说，信息流从$a^{[l]}$到$a^{[l+2]}$需要经过以上所有步骤，即这组网络层的主路径。



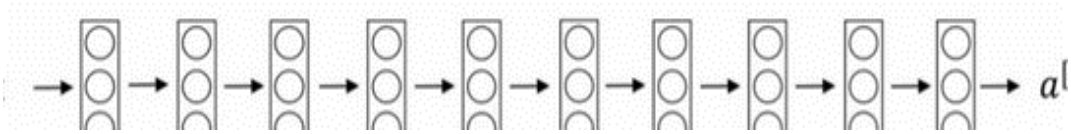
在残差网络中有一点变化，我们将$a^{[l]}$直接向后，拷贝到神经网络的深层，在 ReLU 非线性激活函数前加上$a^{[l]}$，这是一条捷径。$a^{[l]}$的信息直接到达神经网络的深层，不再沿着主路径传递，这就意味着最后这个等式($a^{[l+2]} = g(z^{[l+2]})$)去掉了，取而代之的是另一个 ReLU 非线性函数，仍然对$z^{[l+2]}$进行 g函数处理，但这次要加上$a^{[l]}$，即：$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$，也就是加上的这个$a^{[l]}$产生了一个残差块。



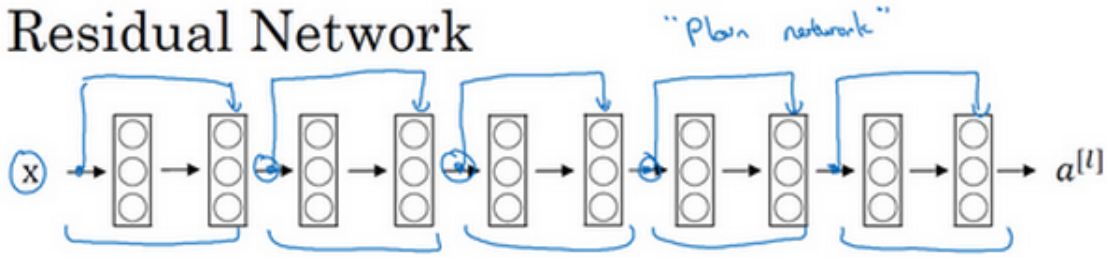
在上面这个图中，我们也可以画一条捷径，直达第二层。实际上这条捷径是在进行 **ReLU** 非线性激活函数之前加上的，而这里的每一个节点都执行了线性函数和 **ReLU** 激活函数。所以$a^{[l]}$插入的时机是在线性激活之后，**ReLU** 激活之前。除了捷径，你还会听到另一个术语“跳跃连接”，就是指$a^{[l]}$跳过一层或者好几层，从而将信息传递到神经网络的更深层。

ResNet 的发明者是**何恺明**（**Kaiming He**）、**张翔宇**（**Xiangyu Zhang**）、**任少卿**（**Shaoqing Ren**）和**孙剑**（**Jiangxi Sun**），他们发现使用残差块能够训练更深的神经网络。所以构建一个

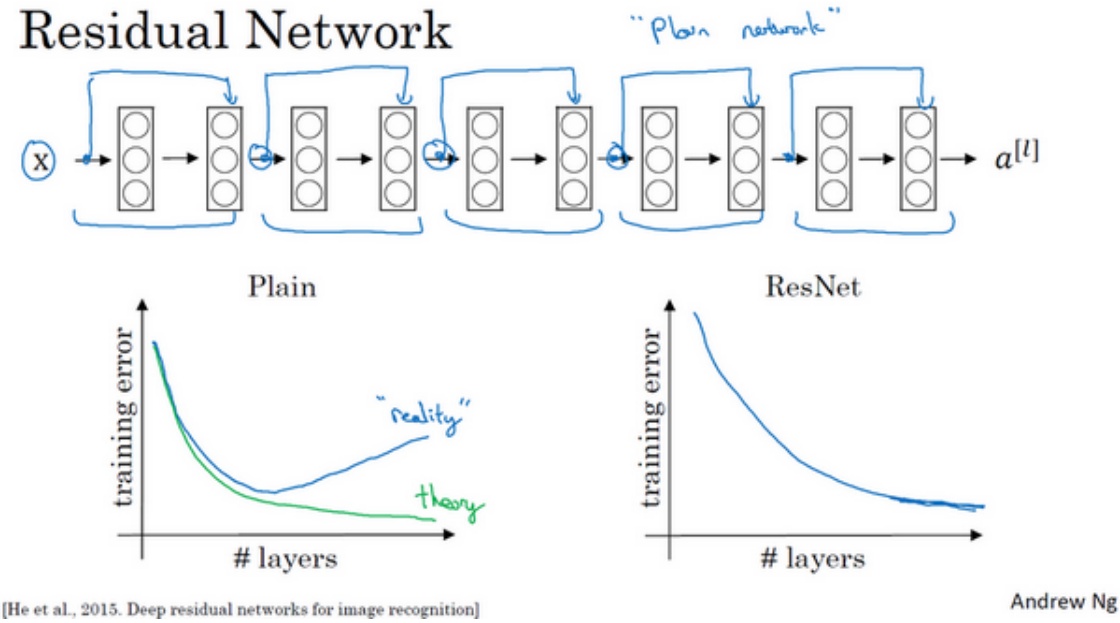
ResNet 网络就是通过将很多这样的残差块堆积在一起，形成一个很深神经网络，我们来看看这个网络。



这并不是一个残差网络，而是一个普通网络（**Plain network**），这个术语来自 **ResNet** 论文。



把它变成 **ResNet** 的方法是加上所有跳跃连接，正如前一张幻灯片中看到的，每两层增加一个捷径，构成一个残差块。如图所示，5 个残差块连接在一起构成一个残差网络。



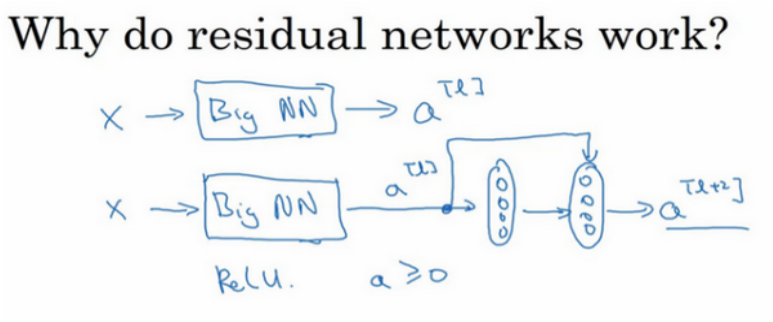
如果我们使用标准优化算法训练一个普通网络，比如说梯度下降法，或者其它热门的优化算法。如果没有残差，没有这些捷径或者跳跃连接，凭经验你会发现随着网络深度的加深，训练错误会先减少，然后增多。而理论上，随着网络深度的加深，应该训练得越来越好才对。也就是说，理论上网络深度越深越好。**但实际上，如果没有残差网络，对于一个普通网络来说，深度越深意味着用优化算法越难训练。实际上，随着网络深度的加深，训练错误会越来越多。**

但有了 **ResNets** 就不一样了，即使网络再深，训练的表现却不错，比如说训练误差减少，就算是训练深达 100 层的网络也不例外。有人甚至在 1000 多层的神经网络中做过实验，尽管目前我还没有看到太多实际应用。但是对x的激活，或者这些中间的激活能够到达网络的更深层。这种方式确实有助于解决梯度消失和梯度爆炸问题，让我们在训练更深网络的同时，又能保证良好的性能。也许从另外一个角度来看，随着网络越来深，网络连接会变得臃肿，但是 **ResNet** 确实在训练深度网络方面非常有效。

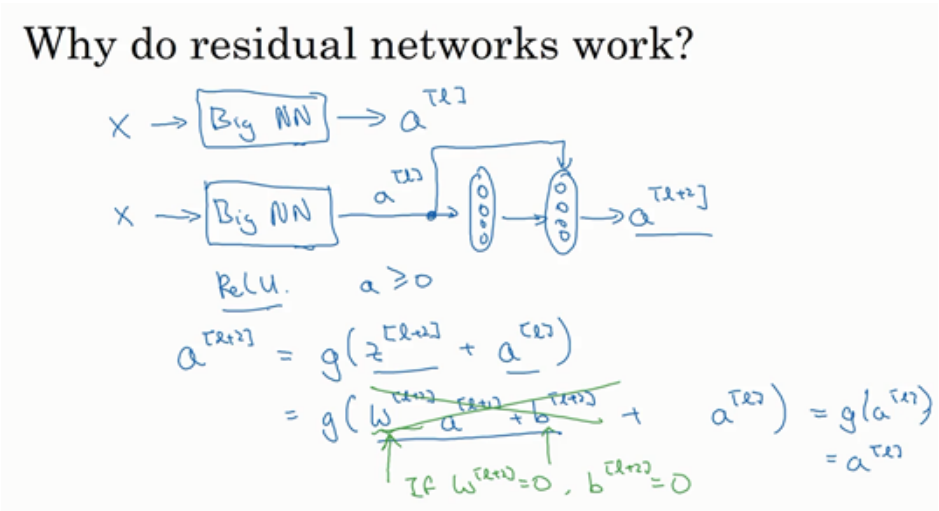
2.4 残差网络为什么有用？（Why ResNets work?）

为什么 **ResNets** 能有如此好的表现，我们来看个例子，它解释了其中的原因，**至少可以说明，如何构建更深层次的 ResNets 网络的同时还不降低它们在训练集上的效率。**希望你已经通过第三门课了解到，通常来讲，网络在训练集上表现好，才能在 **Hold-Out** 交叉验证集或 **dev** 集和测试集上有好的表现，所以至少在训练集上训练好 **ResNets** 是第一步。

先来看个例子，上节课我们了解到，**一个网络深度越深，它在训练集上训练的效率就会有所减弱，这也是有时候我们不希望加深网络的原因。**而事实并非如此，至少在训练 **ResNets** 网络时，并非完全如此，举个例子。

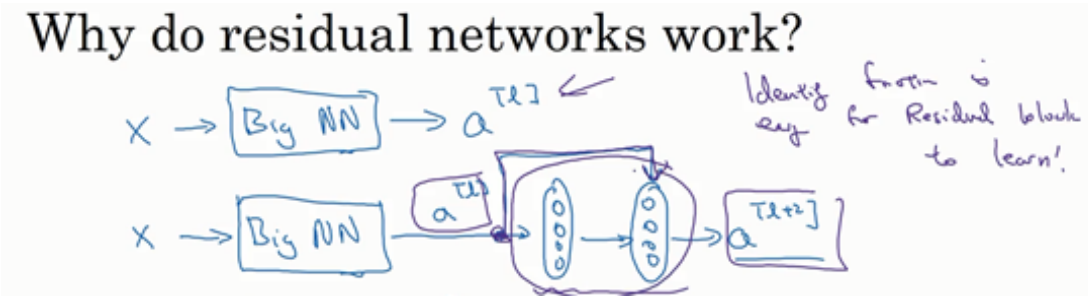


假设有一个大型神经网络，其输入为X，输出激活值$a^{[l]}$。假如你想增加这个神经网络的深度，那么用 **Big NN** 表示，输出为$a^{[l]}$。再给这个网络额外添加两层，依次添加两层，最后输出为$a^{[l+2]}$，可以把这两层看作一个 **ResNets** 块，即具有捷径连接的残差块。为了方便说明，假设我们在整个网络中使用 **ReLU** 激活函数，所以激活值都大于等于 0，包括输入X的非零异常值。因为 **ReLU** 激活函数输出的数字要么是 0，要么是正数。



我们看一下$a^{[l+2]}$的值，也就是上节课讲过的表达式，即$a^{[l+2]}=g(z^{[l+2]}+a^{[l]})$，添加项$a^{[l]}$是刚添加的跳跃连接的输入。展开这个表达式$a^{[l+2]}=g(W^{[l+2]}a^{[l+1]}+b^{[l+2]}+a^{[l]})$，其

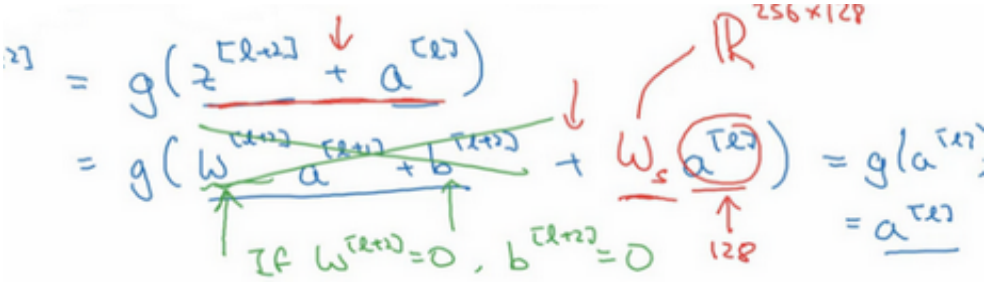
中$z^{[l+2]} = W^{[l+2]}a^{[l+1]} + b^{[l+2]}$。注意一点，如果使用 **L2** 正则化或权重衰减，它会压缩$W^{[l+2]}$的值。如果对b应用权重衰减也可达到同样的效果，尽管实际应用中，你有时会对b应用权重衰减，有时不会。这里的W是关键项，如果$W^{[l+2]} = 0$，为方便起见，假设$b^{[l+2]} = 0$，这几项就没有了，因为它们（$W^{[l+2]}a^{[l+1]} + b^{[l+2]}$）的值为 0。最后$a^{[l+2]} = g\left(a^{[l]}\right) = a^{[l]}$，因为我们假定使用 **ReLU** 激活函数，并且所有激活值都是非负的，$g\left(a^{[l]}\right)$是应用于非负数的 **ReLU** 函数，所以$a^{[l+2]} = a^{[l]}$。



结果表明，残差块学习这个恒等式函数并不难，跳跃连接使我们很容易得出$a^{[l+2]} = a^{[l]}$。这意味着，即使给神经网络增加了这两层，它的效率也并不逊色于更简单的神经网络，因为学习恒等函数对它来说很简单。尽管它多了两层，也只把$a^{[l]}$的值赋值给$a^{[l+2]}$。所以给大型神经网络增加两层，不论是把残差块添加到神经网络的中间还是末端位置，都不会影响网络的表现。

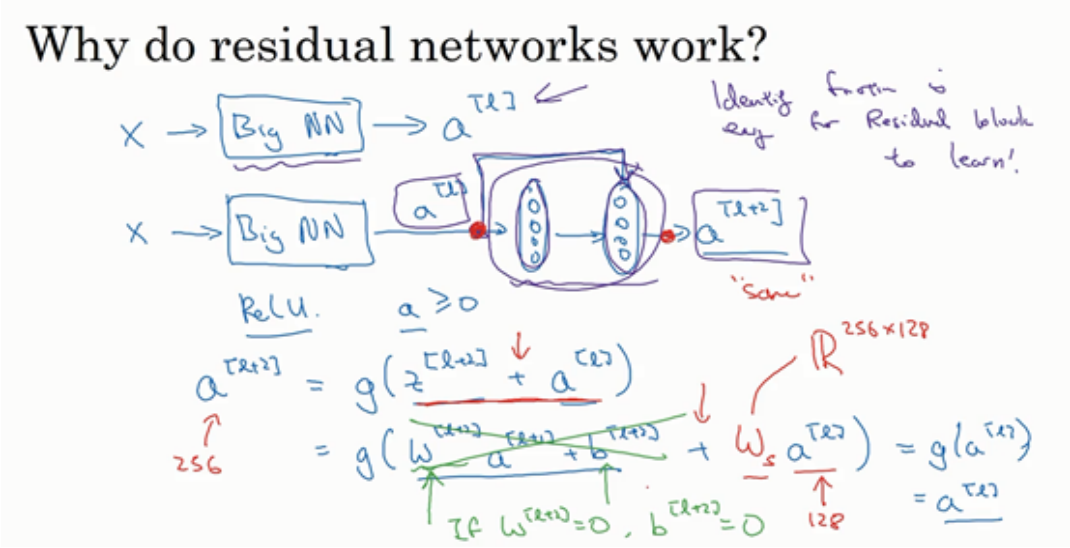
当然，我们的目标不仅仅是保持网络的效率，还要提升它的效率。想象一下，如果这些隐藏层单元学到一些有用信息，那么它可能比学习恒等函数表现得更好。而这些不含有残差块或跳跃连接的深度普通网络情况就不一样了，当网络不断加深时，就算是选用学习恒等函数的参数都很困难，所以很多层最后的表现不但没有更好，反而更糟。

我认为残差网络起作用的主要原因就是这些残差块学习恒等函数非常容易，你能确定网络性能不会受到影响，很多时候甚至可以提高效率，或者说至少不会降低网络的效率，因此创建类似残差网络可以提升网络性能。

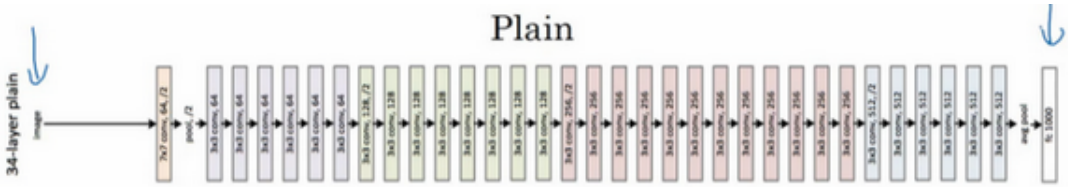


除此之外，关于残差网络，另一个值得探讨的细节是，假设$z^{[l+2]}$与$a^{[l]}$具有相同维度，所以 **ResNets** 使用了许多 **same** 卷积，所以这个$a^{[l]}$的维度等于这个输出层的维度。之所以能实现跳跃连接是因为 **same** 卷积保留了维度，所以很容易得出这个捷径连接，并输出这两个相同维度的向量。

如果输入和输出有不同维度，比如输入的维度是 128，$a^{[l+2]}$的维度是 256，再增加一个矩阵，这里标记为W_s，W_s是一个 256×128 维度的矩阵，所以$W_s a^{[l]}$的维度是 256，这个新增项是 256 维度的向量。你不需要对W_s做任何操作，它是网络通过学习得到的矩阵或参数，它是一个固定矩阵，**padding** 值为 0，用 0 填充$a^{[l]}$，其维度为 256，所以者几个表达式都可以。

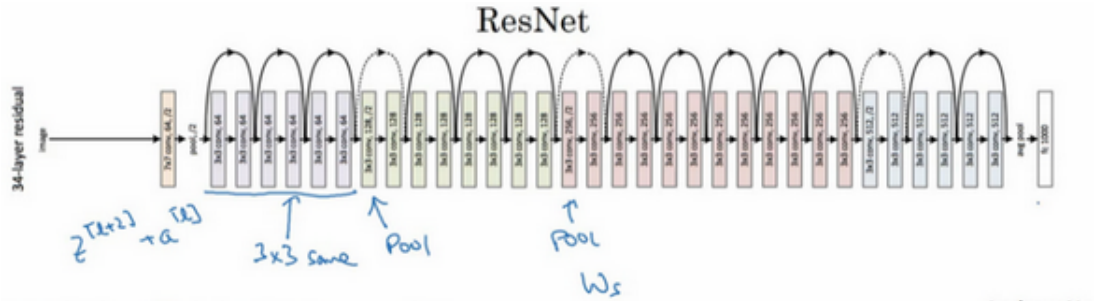


最后，我们来看看 **ResNets** 的图片识别。这些图片是我从何凯明等人论文中截取的，这是一个普通网络，我们给它输入一张图片，它有多个卷积层，最后输出了一个 **Softmax**。



如何把它转化为 **ResNets** 呢？只需要添加跳跃连接。这里我们只讨论几个细节，这个网

络有很多层 3×3 卷积，而且它们大多都是 **same** 卷积，这就是添加等维特征向量的原因。所以这些都是卷积层，而不是全连接层，因为它们是 **same** 卷积，维度得以保留，这也解释了添加项$z^{[l+2]}+a^{[l]}$（维度相同所以能够相加）。



ResNets 类似于其它很多网络，也会有很多卷积层，其中偶尔会有池化层或类池化层的层。不论这些层是什么类型，正如我们在上一张幻灯片看到的，你都需要调整矩阵W_s的维度。普通网络和 **ResNets** 网络常用的结构是：卷积层-卷积层-卷积层-池化层-卷积层-卷积层-卷积层-池化层……依此重复。直到最后，有一个通过 **softmax** 进行预测的全连接层。