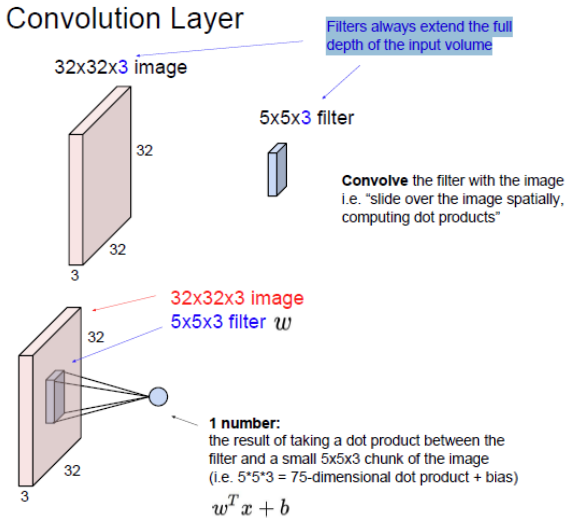
CNN-目标检测、定位、分割

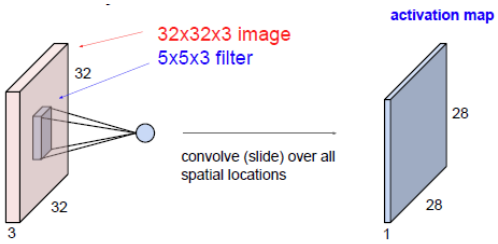
1. 基本概念

1）CNN：Convolutional Neural Networks
2）FC：Fully Connected
3）IoU：Intersection over Union（IoU的值定义：Region Proposal与Ground Truth的窗口的交集比并集的比值，如果IoU低于0.5，那么相当于目标还是没有检测到）
4）ICCV：International Conference on Computer Vision
5）R-CNN：Region-based Convolutional Neural Networks
6）AR：Average Recall
7）mAP：mean Average Precision
8）RPN：Region Proposal Networks
9）FAIR：Facebook AI Research
10）w.r.t.：with respect to
11）Image Classification（what?）：图像分类
12）Object Detection（what+where?）、Localization、Segmentation：对角检测、定位、分割

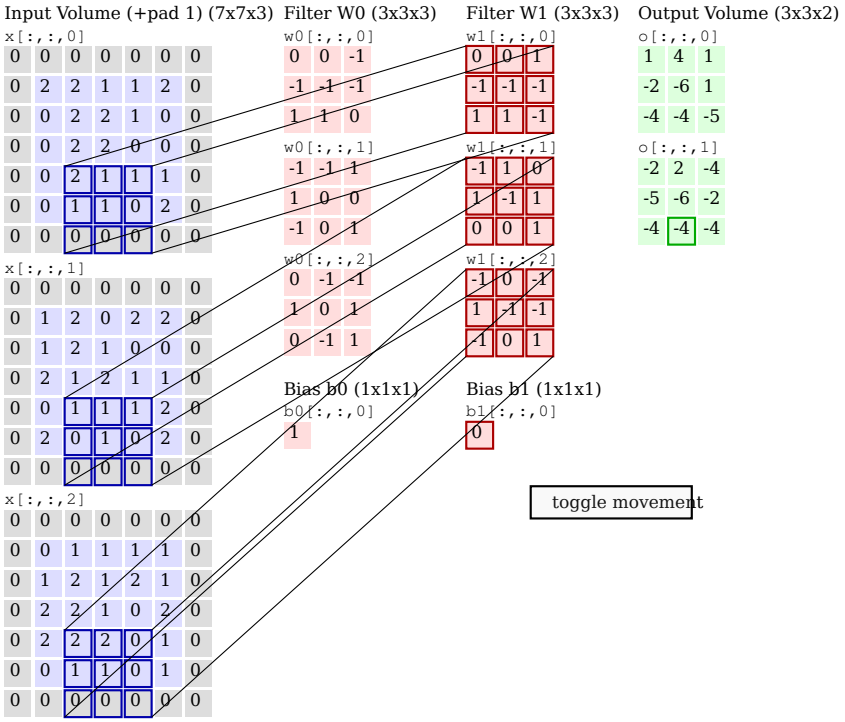
2. CNN基本知识

2.1 CNN的卷积流程

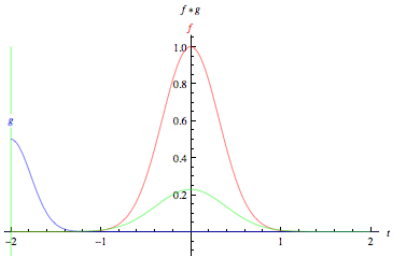




卷积计算过程如下图所示：



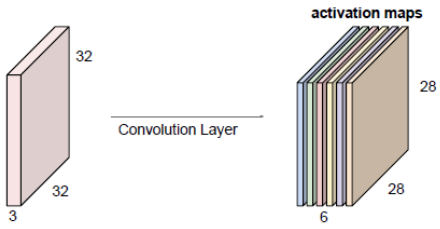
我们刚才描述的即是卷积。可以把卷积想象为信号处理中的一种奇特乘法。也可将两个矩阵生成点积想象为两个函数。图像就是底层函数，而过滤器就是在其上“卷过”的函数。



图像的主要问题在于其高维度，原因是对高维度的处理时间和运算能力成本很高。卷积网络就是为了通过各种方式降低图像的维度而设计的。过滤器步幅即是减少维度的一种方法，另一种方法是降采样。

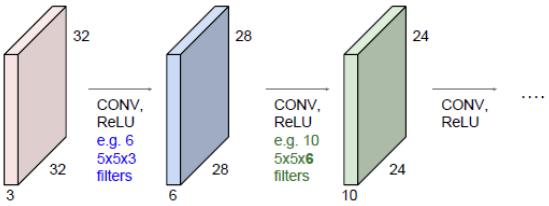
2.2 Activations maps的个数与Filter的个数一致

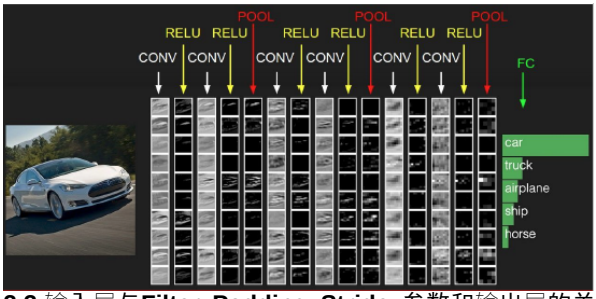
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions





2.3 输入层与Filter、Padding、Stride、参数和输出层的关系

Common settings:

K = (powers of 2, e.g. 32, 64, 128, 512)
- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = ? (whatever fits)
- F = 1, S = 1, P = 0

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

1）参数个数由Filter定义及Filter个数决定，其公式为：

The number of parameters = (FxFxD + 1) * K

2）一个Activation Map共享一个Filter及其权重和偏差

3）Activation Map个数与Filter个数相同

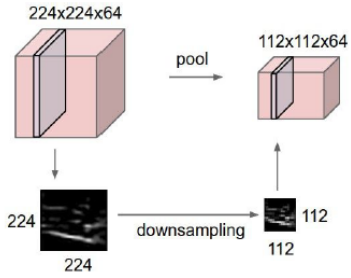
2.4 Pooling(池化/降采样)过程

1）Pooling在每个Activation Map上单独做，在Pooling之后，Activation Map数量不变

Pooling层一般用于降维，将一个kxk的区域内取平均或取最大值，作为这一个小区域内的特征，传递到下一层。传统的Pooling层是不重叠的，使Pooling层重叠可以降低错误率，而且对防止过拟合有一定的效果。

Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



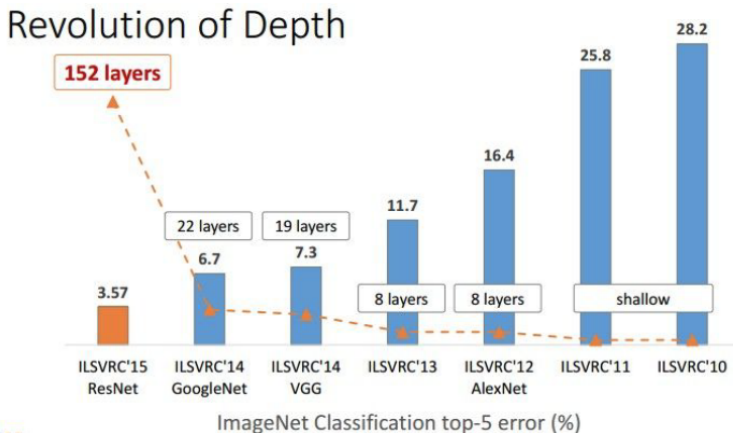
2）Pooling过程描述（Pooling过程不需要参数）

Common settings:

F = 2, S = 2
F = 3, S = 2

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

2.5 深度革命2015

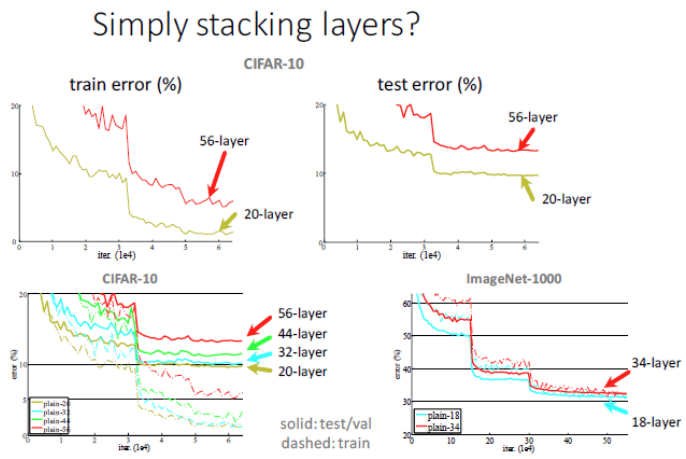


1）深度革命中遇到的问题：

随着CNN网络的发展，尤其的VGG网络的提出，大家发现网络的层数是一个关键因素，貌似越深的网络效果越好。但是随着网络层数的增加，问题也随之而来。

（1）第一个问题：vanishing/exploding gradients（即梯度消失或爆炸）：这就导致训练难以收敛。但是随着normalized initialization and BN(Batch Normalization)的提出，解决了梯度消失或爆炸问题。

（2）第二个问题：网络越深，训练误差和测试误差越大。在收敛问题解决后，又一个问题暴露出来：随着网络深度的增加，系统精度得到饱和之后，迅速的下滑。让人意外的是这个性能下降不是过拟合导致的。对一个合适深度的模型加入额外的层数导致训练误差变大。如下图所示，可通过Deep Residual Learning 框架来解决这种因为深度增加而导致准确性下降问题。

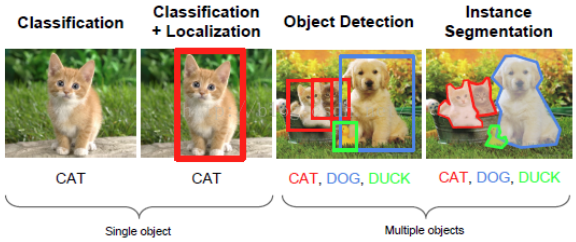


3. 空间定位与检测

参考信息《基于深度学习的目标检测研究进展》

3.1 计算机视觉任务

Computer Vision Tasks



3.2 传统目标检测方法

传统目标检测流程：

1）区域选择（穷举策略：采用滑动窗口，且设置不同的大小，不同的长宽比对图像进行遍历，时间复杂度高）

2）特征提取（SIFT、HOG等；形态多样性、光照变化多样性、背景多样性使得特征鲁棒性差）

3）分类器（主要有SVM、Adaboost等）

传统目标检测的主要问题：

1）基于滑动窗口的区域选择策略没有针对性，时间复杂度高，窗口冗余

2）手工设计的特征对于多样性的变化没有很好的鲁棒性

3.3 基于侯选区域(Region Proposal)的深度学习目标检测法

3.3.1 R-CNN (CVPR2014, TPAMI2015)

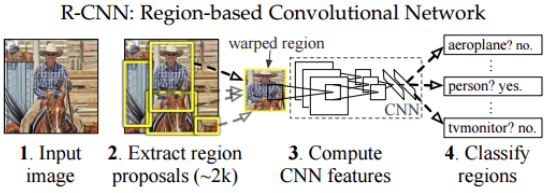
1）Region Proposal：可以解决滑动窗口的问题

候选区域（Region Proposal）：是预先找出图中目标可能出现的位置。它利用了图像中的纹理、边缘、颜色等信息，可以保证在选取较少窗口(几千甚至几百）的情况下保持较高的召回率（Recall）。

常用的Region Proposal有(详见"What makes for effective detection proposals?")：

- Selective Search
- Edge Boxes

2）R-CNN：可以解决特征鲁棒性的问题



参考信息

(1) 输入测试图像

(2) 利用selective search算法在图像中从下到上提取2000个左右的Region Proposal

(3) 将每个Region Proposal缩放（warp）成227x227的大小并输入到CNN，将CNN的fc7层的输出作为特征

(4) 将每个Region Proposal提取到的CNN特征输入到SVM进行分类

注：1）对每个Region Proposal缩放到同一尺度是因为CNN全连接层输入需要保证维度固定。

2）上图少画了一个过程——对于SVM分好类的Region Proposal做边框回归（bounding-box regression），边框回归是对region proposal进行纠正的线性回归算法，为了让region proposal提取到的窗口跟

目标真实窗口更吻合。因为region proposal提取到的窗口不可能跟人手工标记那么准，如果region proposal跟目标位置偏移较大，即便是分类正确了，但是由于IoU(region proposal与Ground Truth的窗口的交集比并集的比值)低于0.5，那么相当于目标还是没有检测到。

3）R-CNN缺点：

(1) 训练分为多个阶段，步骤繁琐: 微调网络+训练SVM+训练边框回归器

(2) 训练耗时，占用磁盘空间大：5000张图像产生几百G的特征文件

(3) 速度慢: 使用GPU, VGG16模型处理一张图像需要47s。

(4) 测试速度慢：每个候选区域需要运行整个前向CNN计算

(5) SVM和回归是事后操作：在SVM和回归过程中CNN特征没有被学习更新

针对速度慢的这个问题，SPP-NET给出了很好的解决方案。

3.3.2 SPP-NET (ECCV2014, TPAMI2015)

SSP-Net：Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition

先看一下R-CNN为什么检测速度这么慢，一张图都需要47s！仔细看下R-CNN框架发现，对图像提完Region Proposal（2000个左右）之后将每个Proposal当成一张图像进行后续处理(CNN提特征+SVM分类)，实际上对一张图像进行了2000次提特征和分类的过程！这2000个Region Proposal不都是图像的一部分吗，那么我们完全可以对图像提一次卷积层特征，然后只需要将Region Proposal在原图的位置映射到卷积层特征图上，这样对于一张图像我们只需要提一次卷积层特征，然后将每个Region Proposal的卷积层特征输入到全连接层做后续操作。（对于CNN来说，大部分运算都耗在卷积操作上，这样做可以节省大量时间）。

现在的问题是每个Region Proposal的尺度不一样，直接这样输入全连接层肯定是不行的，因为全连接层输入必须是固定的长度。SPP-NET恰好可以解决这个问题。

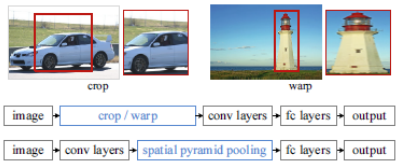


Figure 1: Top: cropping or warping to fit a fixed size. Middle: a conventional CNN. Bottom: our spatial pyramid pooling network structure.

由于传统的CNN限制了输入必须固定大小（比如AlexNet是224x224），所以在实际使用中往往需要对原图片进行crop或者warp的操作：

- crop：截取原图片的一个固定大小的patch
- warp：将原图片的ROI缩放到一个固定大小的patch

无论是crop还是warp，都无法保证在不失真的情况下将图片传入到CNN当中：

- crop：物体可能会产生截断，尤其是长宽比大的图片。
- warp：物体被拉伸，失去“原形”，尤其是长宽比大的图片

SPP为的就是解决上述的问题，做到的效果为：不管输入的图片是什么尺度，都能够正确的传入网络。

具体思路为：CNN的卷积层是可以处理任意尺度的输入的，只是在全连接层处有限制尺度——换句话说，如果找到一个方法，在全连接层之前将其输入限制到等长，那么就解决了这个问题。

具体方案如下图所示：

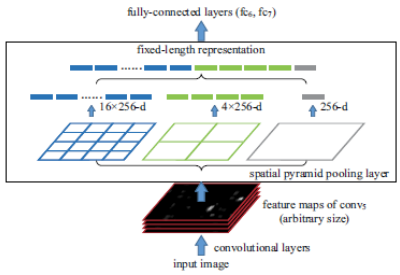


Figure 3: A network structure with a spatial pyramid pooling layer. Here 256 is the filter number of the $conv_5$ layer, and $conv_5$ is the last convolutional layer.

如果原图输入是224x224，对于conv5出来后的输出，是13x13x256的，可以理解成有256个这样的filter，每个filter对应一张13x13的activation map。如果像上图那样将activation map pooling成4x4 2x2 1x1三张子图，做max pooling后，出来的特征就是固定长度的(16+4+1)x256那么多的维度了。如果原图的输入不是224x224，出来的特征依然是(16+4+1)x256；直觉地说，可以理解成将原来固定大小为(3x3)窗口的pool5改成了自适应窗口大小，窗口的大小和activation map成比例，保证了经过pooling后出来的feature的长度是一致的。

使用SPP-NET相比于R-CNN可以大大加快目标检测的速度，但是依然存在着很多问题：

(1) 训练分为多个阶段，步骤繁琐: 微调网络+训练SVM+训练训练边框回归器

(2) SPP-NET在微调网络的时候固定了卷积层，只对全连接层进行微调，而对于一个新的任务，有必要对卷积层也进行微调。（分类的模型提取的特征更注重高层语义，而目标检测任务除了语义信息还需要目标的位置信息）

针对这两个问题，RBG又提出Fast R-CNN，一个精简而快速的目标检测框架。

3.3.3 Fast R-CNN(ICCV2015)

有了前边R-CNN和SPP-NET的介绍，我们直接看Fast R-CNN的框架图：

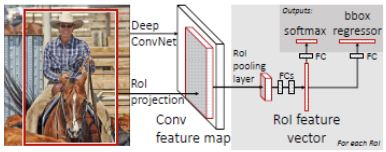


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

与R-CNN框架图对比，可以发现主要有两处不同：一是最后一个卷积层后加了一个ROI pooling layer，二是损失函数使用了多任务损失函数(multi-task loss)，将边框回归直接加入到CNN网络中训练。

(1) ROI pooling layer实际上是SPP-NET的一个精简版，SPP-NET对每个proposal使用了不同大小的金字塔映射，而ROI pooling layer只需要下采样到一个7x7的特征图。对于VGG16网络conv5_3有512个特征图，这样所有region proposal对应了一个7*7*512维度的特征向量作为全连接层的输入。

(2) R-CNN训练过程分为了三个阶段，而Fast R-CNN直接使用softmax替代SVM分类，同时利用多任务损失函数边框回归也加入到了网络中，这样整个的训练过程是端到端的(除去region proposal提取阶段)。

(3) Fast R-CNN在网络微调的过程中，将部分卷积层也进行了微调，取得了更好的检测效果。

性能对比数据：

		R-CNN	Fast R-CNN
Faster!	Training Time:	84 hours	**9.5 hours**
	(Speedup)	1x	**8.8x**
FASTER!	Test time per image	47 seconds	**0.32 seconds**
	(Speedup)	1x	**146x**
Better!	mAP (VOC 2007)	66.0	**66.9**

Using VGG-16 CNN on Pascal VOC 2007 dataset

Test-time speeds don't include region proposals

	R-CNN	Fast R-CNN
Test time per image	47 seconds	**0.32 seconds**
(Speedup)	1x	**146x**
Test time per image with Selective Search	50 seconds	**2 seconds**
(Speedup)	1x	**25x**

1）Fast R-CNN优点：

Fast R-CNN融合了R-CNN和SPP-NET的精髓，并且引入多任务损失函数，使整个网络的训练和测试变得十分方便。在Pascal VOC2007训练集上训练，在VOC2007测试的结果为66.9%(mAP)，如果使用VOC2007+2012训练集训练，在VOC2007上测试结果为70%（数据集的扩充能大幅提高目标检测性能）。使用VGG16每张图像总共需要3s左右。

2）Fast R-CNN 缺点：

Region Proposal的提取使用selective search，目标检测时间大多消耗在这上面（提Region Proposal 2~3s，而提特征分类只需0.32s），无法满足实时应用，而且并没有实现真正意义上的端到端训练测试（region proposal使用selective search先提取处来）。那么有没有可能直接使用CNN直接产生Region Proposal并对其分类？Faster R-CNN框架就是符合这样需要的目标检测框架。

3.3.4 Faster R-CNN(NIPS2015)

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

在Region Proposal + CNN分类的这种目标检测框架中，Region Proposal质量好坏直接影响到目标检测任务的精度。如果找到一种方法只提取几百个或者更少的高质量的假选窗口，而且召回率很高，这不但能加快目标检测速度，还能提高目标检测的性能（假阳例少）。RPN(Region Proposal Networks)网络应运而生。

1）RPN的核心思想

是使用卷积神经网络直接产生Region Proposal，使用的方法本质上就是滑动窗口。RPN的设计比较巧妙，RPN只需在最后的卷积层上滑动一遍，因为Anchor机制和边框回归可以得到多尺度多长宽比的Region Proposal。

2）Faster R-CNN架构

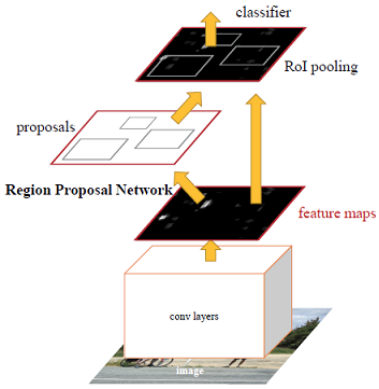


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

3）RPN架构

RPN采用任意大小的的图像作为输入，并输出一组候选的矩形，每个矩形都有一个对象分数。

RPN被用于训练直接产生候选区域，不需要外部的候选区域。

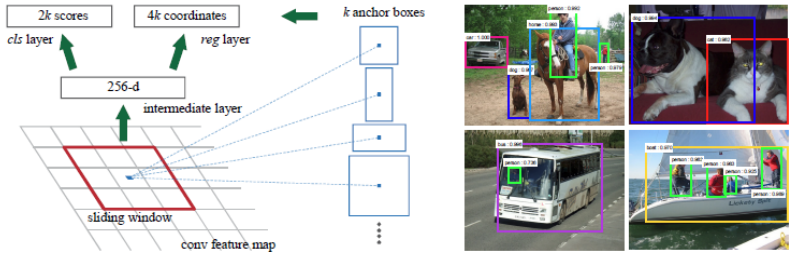
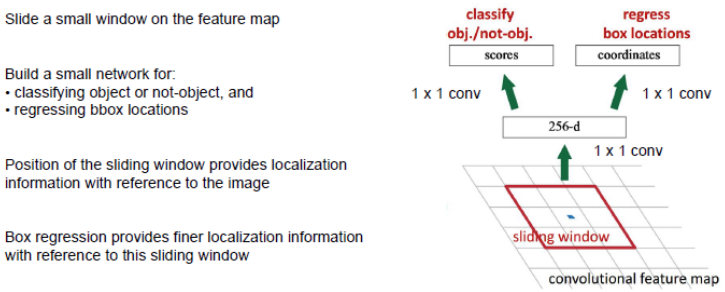


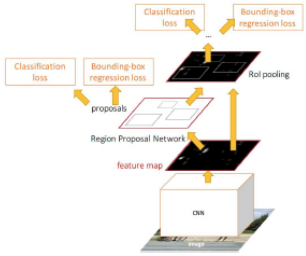
Figure 3: **Left**: Region Proposal Network (RPN). **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.



Faster R-CNN: Training

In the paper: Ugly pipeline
- Use alternating optimization to train RPN, then Fast R-CNN with RPN proposals, etc.
- More complex than it has to be

Since publication: Joint training!
One network, four losses
- RPN classification (anchor good / bad)
- RPN regression (anchor -> proposal)
- Fast R-CNN classification (over classes)
- Fast R-CNN regression (proposal -> box)



	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image (with proposals)	50 seconds	2 seconds	**0.2 seconds**
(Speedup)	1x	25x	**250x**
mAP (VOC 2007)	66.0	**66.9**	**66.9**

Anchor是滑动窗口的中心，它与尺度和长宽比相关，默认采3种尺度（128,256,512），3种长宽比（1:1,1:2,2:1），则在每一个滑动位置k=9 anchors。

我们直接看上边的RPN网络结构图(使用了ZF<Zeiler and Fergus model>模型),给定输入图像(假设分辨率为600*1000),经过卷积操作得到最后一层的卷积特征图(大小约为40*60)。在这个特征图上使用3*3的卷积核(滑动窗口)与特征图进行卷积,最后一层卷积层共有256个feature map,那么这个3*3的区域卷积后可以获得一个256维的特征向量,后边接cls layer(box-classification layer)和reg layer(box-regression layer)分别用于分类和边框回归(跟Fast R-CNN类似,只不过这里的类别只有目标和背景两个类别)。3*3滑窗对应的每个特征区域同时预测输入图像3种尺度(128,256,512),3种长宽比(1:1,1:2,2:1)的region proposal,这种映射的机制称为anchor。所以对于这个40*60的feature map,总共有约20000(40*60*9)个anchor,也就是预测20000个region proposal。

这样设计的好处是什么呢?虽然现在也是用的滑动窗口策略,但是:滑动窗口操作是在卷积层特征图上进行的,维度较原始图像降低了16*16倍(中间经过了4次2*2的pooling操作);多尺度采用了9种anchor,对应了三种尺度和三种长宽比,加上后边接了边框回归,所以即便是这9种anchor外的窗口也能得到一个跟目标比较接近的region proposal。

4)总结

Faster R-CNN将一直以来分离的region proposal和CNN分类融合到了一起,使用端到端的网络进行目标检测,无论在速度上还是精度上都得到了不错的提高。然而Faster R-CNN还是达不到实时的目标检测,预先获取Region Proposal,然后在对每个Proposal分类计算量还是比较大。比较幸运的是YOLO这类目标检测方法的出现让实时性也变的成为可能。

总的来说,从R-CNN, SPP-NET, Fast R-CNN, Faster R-CNN一路走来,基于深度学习目标检测的流程变得越来越精简,精度越来越高,速度也越来越快。可以说基于Region Proposal的R-CNN系列目标检测方法是当前目标最主要的一个分支。

3.3.5 R-FCN(2016.5)

《R-FCN: Object Detection via Region-based Fully Convolutional Networks》

顾名思议:全卷积网络,就是全部是卷积层,而没有全连接层(fc)。

R-FCN(基于区域的检测器)的方法是:在整个图像上共享计算,通过移除最后的fc层实现(即删除[illegible]子网络)。使用"位置敏感的得分图"来解决了图像分类平移不变性与对象检测平移变化之间的矛盾。

此矛盾为:物体分类要求平移不变性越大越好 (图像中物体的移动不用区分),而物体检测要求有平移变化。所以,ImageNet 分类领先的结果证明尽可能有平移不变性的全卷积结构更受亲睐。另一方面,物体检测任务需要一些平移变化的定位表示。比如,物体的平移应该使网络产生响应,这些响应对描述候选框覆盖真实物体的好坏是有意义的。我们假设图像分类网络的卷积层越深,则该网络对平移越不敏感。

CNN随着网络深度的增加,网络对于位置(Position)的敏感度越来越低,也就是所谓的translation-invariance,但是在Detection的时候,需要对位置信息有很强的的敏感度。

那么**ResNet-101**的**detection**是怎么做的?

在R-FCN之前,很简单,把ROI-pooling层放到了前面的卷积层,然后后面的卷积层不共享计算,这样一可以避免过多的信息损失,二可以用后来的卷积层学习位置信息。

R-FCN:采用全卷积网络结构作为 FCN,为给 FCN 引入平移变化,用专门的卷积层构建位置敏感分数地图(position-sensitive score maps)。每个空间敏感地图编码感兴趣区域的相对空间位置信息。在FCN上面增加1个位置敏感 RoI 池化层来监管这些分数地图。

R-FCN思路就是利用最后一层网络通过FCN构成一个position-sensitive的feature map。具体而言,每一个proposal的位置信息都需要编码,那么先把proposal分成k*k个grid,然后对每一个grid进行编码。在最后一层map之后,再使用卷积计算产生一个k*k*(C+1)的map(k*k代表总共的grid数目,C代表class num,+1代表加入一个背景类)。

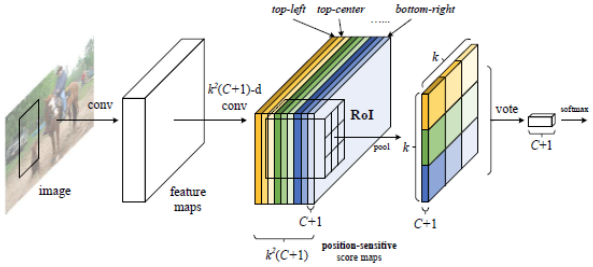


Figure 1: Key idea of **R-FCN** for object detection. In this illustration, there are $k \times k = 3 \times 3$ position-sensitive score maps generated by a fully convolutional network. For each of the $k \times k$ bins in an RoI, pooling is only performed on one of the k^2 maps (marked by different colors).

Table 1: Methodologies of *region-based* detectors using **ResNet-101** [9].

	R-CNN [7]	Faster R-CNN [19, 9]	R-FCN [ours]
depth of shared convolutional subnetwork	0	91	101
depth of RoI-wise subnetwork	101	10	**0**

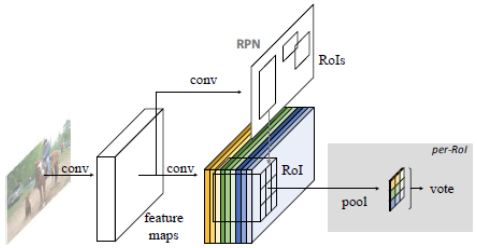


Figure 2: Overall architecture of R-FCN. A Region Proposal Network (RPN) [18] proposes candidate RoIs, which are then applied on the score maps. All learnable weight layers are convolutional and are computed on the entire image; the per-RoI computational cost is negligible.

RPN 给出感兴趣区域，R-FCN 对该感兴趣区域分类。R-FCN 在与 RPN 共享的卷积层后多加1个卷积层。所以，R-FCN 与 RPN 一样，输入为整幅图像。但 R-FCN 最后1个卷积层的输出从整幅图像的卷积响应图像中分割出感兴趣区域的卷积响应图像。

R-FCN 最后1个卷积层在整幅图像上为每类生成k*k个位置敏感分数图，有C类物体外加1个背景，因此有k*k(C+1)个通道的输出层。k*k个分数图对应描述位置的空间网格。比如，k×k=3×3，则9个分数图编码单个物体类的 {top−left,top−center,top−right,...,bottom−right}。

R-FCN 最后用位置敏感 RoI 池化层，给每个 RoI 1个分数。选择性池化图解：看上图的橙色响应图像 (top−left)，抠出橙色方块 RoI，池化橙色方块 RoI 得到橙色小方块 (分数)；其它颜色的响应图像同理。对所有颜色的小方块投票 (或池化) 得到1类的响应结果。

产生完了这张map之后，再根据proposal产生一个长宽各为k，channel数目为C+1的score map。具体产生score map的方法是，假如k=3，C=20，那么score map的20个类每个类都有3*3的feature，一共9个格子，9个格子都记录了空间信息。而这每一个类的每一个格子都对应前面那个channel数为3*3*21的大map个channel的map。现在把score map中的格子对应的区域的map中的信息取平均，然后这个平均值就是score map格子中的值。最后把score map的值进行vote（avg pooling）来形成一个21维的向量来做分类即可。

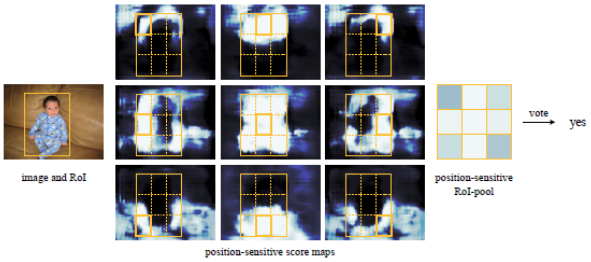


Figure 3: Visualization of R-FCN ($k \times k = 3 \times 3$) for the *person* category.

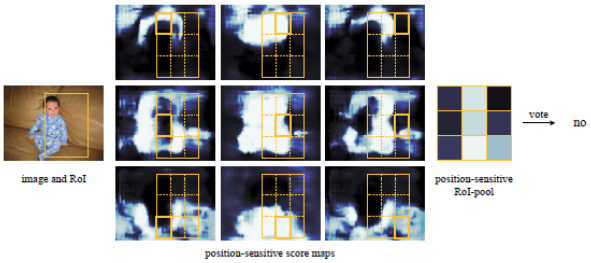


Figure 4: Visualization when an RoI does not correctly overlap the object.

当分类正确时，该类通道的位置敏感分数图 (中间) 的大多数橙色实线网格内的响应在整个 RoI 位置范围内最强。

对应的bbox regression只需要把C+1设成4就可以了。

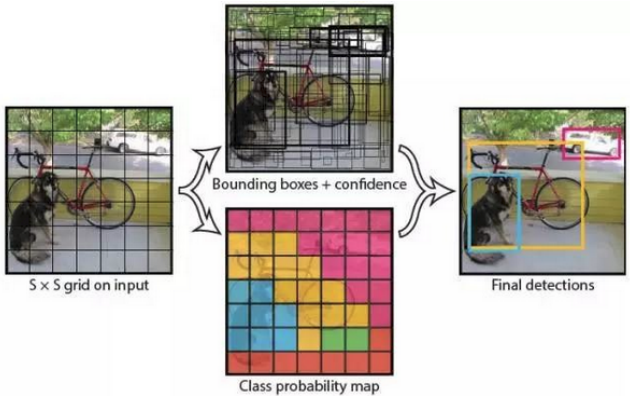
R-FCN采用的一些方法比Faster R-CNN的baseline提高了3个点，并且比原来Faster R-CNN更快（因为全部计算都共享了）。但是和改进过的Faster R-CNN相比（ROI Pooling提前那种）提高了0.2个点，速度快了2.5倍。所以目前为止这个方法的结果应该是所有方法中速度和Performance结合的最好的。

3.4 基于回归方法的深度学习目标检测算法

Faster R-CNN的方法目前是主流的目标检测方法，但是速度上并不能满足实时的要求。YOLO一类的方法慢慢显现出其重要性，这类方法使用了回归的思想，即给定输入图像，直接在图像的多个位置上回归出这个位置的目标边框以及目标类别。

3.4.1 YOLO (CVPR2016, oral)

YOLO：You Only Look Once: Unified, Real-Time Object Detection



我们直接看上面YOLO的目标检测的流程图：

(1) 给个一个输入图像，首先将图像划分成7*7(设S=7)的网格

(2) 对于每个网格，我们都预测2个边框（包括每个边框是目标的置信度以及每个边框区域在多个类别上的概率）

(3) 根据上一步可以预测出7*7*2个目标窗口，然后根据阈值去除可能性比较低的目标窗口，最后NMS去除冗余窗口即可。

可以看到整个过程非常简单，不需要中间的Region Proposal在找目标，直接回归便完成了位置和类别的判定。

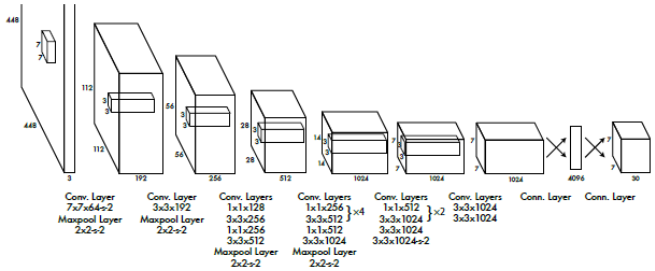


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

那么如何才能做到直接在不同位置的网格上回归出目标的位置和类别信息呢？上面是YOLO的网络结构图，前边的网络结构跟GoogLeNet的模型比较类似，主要的是最后两层的结构，卷积层之后接了一个4096维的全连接层，然后后边又全连接到一个7*7*30维的张量上。实际上这7*7就是划分的网格数，现在要在每个网格上预测目标两个可能的位置以及这个位置的目标置信度和类别，也就是每个网格预测两个目标，每个目标的信息有4维坐标信息(中心点坐标+长宽)，1个是目标的置信度，还有类别数20(VOC上20个类别)，总共就是(4+1)*2+20 = 30维的向量。这样可以利用前边4096维的全图特征直接在每个网格上回归出目标检测需要的信息（边框信息加类别）。

总结：

YOLO将目标检测任务转换成一个回归问题，大大加快了检测的速度，使得YOLO可以每秒处理45张图像。而且由于每个网络预测目标窗口时使用的是全图信息，使得false positive比例大幅降低（充分的上下文信息）。但是YOLO也存在问题：没有了Region Proposal机制，只使用7*7的网格回归会使得目标不能非常精准的定位，这也导致了YOLO的检测精度并不是很高。

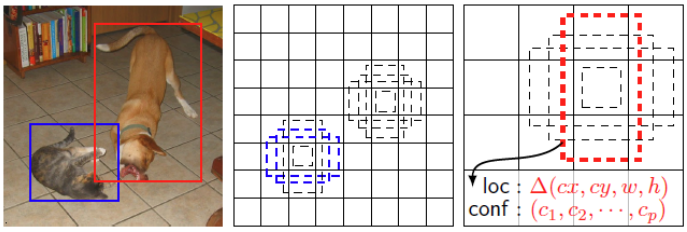
Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18

3.4.2 SSD(单次检测)

SSD: Single Shot MultiBox Detector

上面分析了YOLO存在的问题，使用整图特征在7*7的粗糙网格内回归对目标的定位并不是很精准。那是不是可以结合Region Proposal的思想实现精准一些的定位？SSD结合YOLO的回归思想以及Faster R-CNN的

anchor机制做到了这点。



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

上图是SSD的一个框架图，首先SSD获取目标位置和类别的方法跟YOLO一样，都是使用回归，但是YOLO预测某个位置使用的是全图的特征，SSD预测某个位置使用的是这个位置周围的特征（感觉更合理一些）。那么如何建立某个位置和其特征的对应关系呢？可能你已经想到了，使用Faster R-CNN的anchor机制。如SSD的框架图所示，假如某一层特征图(图b)大小是8*8，那么就使用3*3的滑窗提取每个位置的特征，然后这个特征回归得到目标的坐标信息和类别信息(图c)。

不同于Faster R-CNN，这个anchor是在多个feature map上，这样可以利用多层的特征并且自然的达到多尺度（不同层的feature map 3*3滑窗感受野不同）。

小结：

SSD结合了YOLO中的回归思想和Faster R-CNN中的anchor机制，使用全图各个位置的多尺度区域特征进行回归，既保持了YOLO速度快的特性，也保证了窗口预测的跟Faster R-CNN一样比较精准。SS VOC2007上mAP可以达到72.1%，速度在GPU上达到58帧每秒。

Method	*mAP*	FPS	# Boxes
Faster R-CNN [2](VGG16)	73.2	7	300
Faster R-CNN [2](ZF)	62.1	17	300
YOLO [5]	63.4	45	98
Fast YOLO [5]	52.7	155	98
SSD300	72.1	58	7308
SSD500	**75.1**	23	20097

总结：YOLO的提出给目标检测一个新的思路，SSD的性能则让我们看到了目标检测在实际应用中真正的可能性。

3.5 基于残差（Residual）方法的深度学习目标检测算法

3.5.1 深度残差网络（Deep Residual Networks）

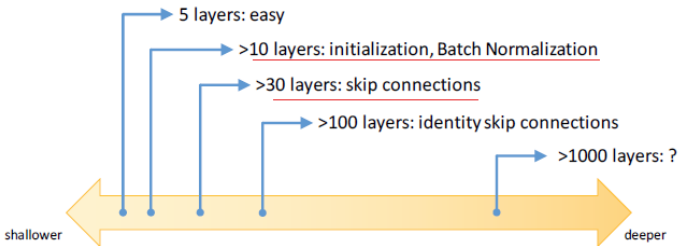
Deep Residual Networks

它使用残差学习的这一思想使得学习更深的网络成为可能，从而学习到更好的表达。

层数越多的神经网络越难以训练。当层数超过一定数量后，传统的深度网络就会因优化问题而出现欠拟合（underfitting）的情况。残差学习框架大幅降低训练更深层网络的难度，也使准确率得到显著提升。在 ImageNet 和 COCO 2015 竞赛中，共有 152 层的深度残差网络 ResNet 在图像分类、目标检测和语义分割各个分项都取得最好成绩，相关论文更是连续两次获得 CVPR 最佳论文。

最新研究发现，当残差网络将身份映射作为 skip connection 并实现 inter-block activation，正向和反向信号能够直接从一个区块传播到另一个区块，这样就达到了 1001 层的残差网络。由此可见，神经网络的深度这一非常重要的因素，还有很大的提升空间。

1）深度谱



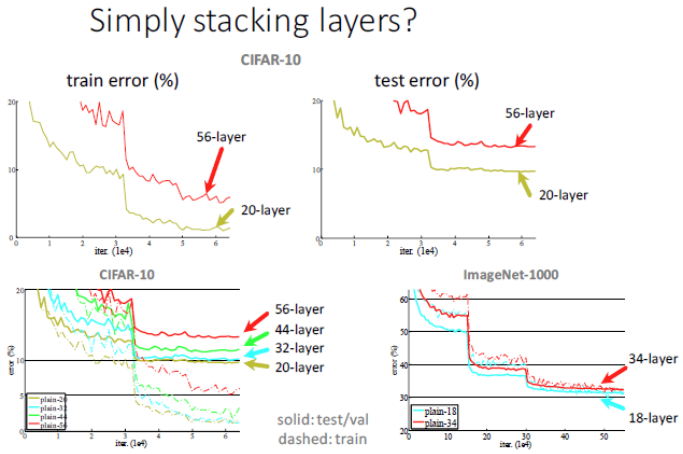
2）为使用网络层数更多，通常采用的方法有：初始化算法，BN方法

3）是否简单堆叠的网络层数越多，训练误差和测试误差就越小？答案正好相反

4）目前流行的深度神经网络结构大致可以分为三类：

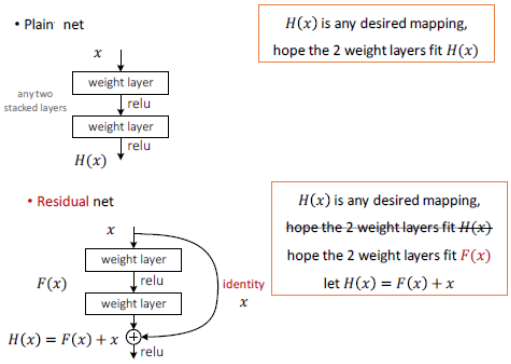
- 直线型（如AlexNet, VGGNet）
- 局部双分支型（ResNet）
- 局部多分支型（GoogleNet）

很久以前人们就已经认识到更深的网络能够产生更好的数据表达，但是如何训练一个很深的网络却一直是一个困扰人们的问题，这主要是由于梯度消失或爆炸以及尺度不均匀的初始化造成的。围绕这一问题，人们提出了ReLU、Xavier、pReLU、batch normalization和path-SGD等一系列方法，但是本文作者却发现即使有这些方法，神经网络的训练仍然呈现了degradation的现象。所谓degradation现象，就是随着网络深度的增加，网络的性能反而下降，而且这种性能的下降并不是由前面所说的问题造成的。



4）深度残差学习（Deep Residual Learning）的思想

假如目前有一个可以工作的很好的网络A，这时来了一个比它更深的网络B，只需要让B的前一部分与A完全相同，后一部分只实现一个恒等映射（identity mapping），这样B最起码能获得与A相同的性能，而不差。深度残差学习的思想也由此而产生，既然B后面的部分完成的是恒等映射，何不在训练网络的时候加上这一先验（在网络训练过程中，加入先验信息指导非常重要，合理的先验往往会取得非常好的效果），于是构造网络的时候加入了捷径（shortcut）连接，即每层的输出不是传统神经网络中的输入的映射，而是输入的映射和输入的叠加，如下图中的"Residual net"所示。



在Residual net中：

（1）identity：为恒等映射，此条路径一直存在

（2）F(x)：为需要学习的残差函数(residual function)：H(x)-x = F(x)

问题的重新表示或预处理会简化问题的优化!

假设我们期望的网络层关系映射为 H(x)，我们让 the stacked nonlinear layers 拟合另一个映射，F(x):= H(x)-x，那么原先的映射就是 F(x)+x。这里我们假设优化残差映射F(x) 比优化原来的映射 H(x)容易。

这里我们首先求取残差映射 F(x):= H(x)-x，那么原先的映射就是 F(x)+x。尽管这两个映射应该都可以近似理论真值映射 the desired functions (as hypothesized)，但是它俩的学习难度是不一样的。

这种改写启发于"网络层数越多，训练和测试误差越大"性能退化问题违反直觉的现象。如果增加的层数可以构建为一个恒等映射(identity mappings)，那么增加层数后的网络训练误差应该不会增加，与没增加之前相比较。性能退化问题暗示多个非线性网络层用于近似identity mappings 可能有困难。使用残差学习改写问题之后，如果identity mappings 是最优的，那么优化问题变得很简单，直接将多层非线性网络参数趋0。

实际中，identity mappings 不太可能是最优的，但是上述改写问题可能帮助预处理问题。如果最优函数接近identity mappings，那么优化将会变得容易些。实验证明该思路是对的。

F(x)+x 可以通过shortcut connections 来实现，如下图所示：

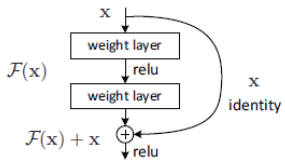


Figure 2. Residual learning: a building block.

上图中的shortcut connections执行一个简单的恒等映射；既没有参数，也没有计算复杂度。

公式分析如下：

(1)需要学习的残差映射

$$y = \mathcal{F}(x, \{W_i\}) + x$$

x, y：分别表示此块的输入和输出向量

$\mathcal{F}(x, \{W_i\})$：表示需要学习的残差映射

$\mathcal{F} = W_2\sigma(W_1 x)$ σ：表示ReLU

(2)x和F的维数必须相同

如果x和F的维数不相同，则对x进行线性投影（linear projection）使用其与F的维数一致，公式如下：

$$y = \mathcal{F}(x, \{W_i\}) + W_s x$$

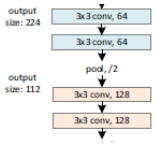
5)网络架构

5.1)普通网络(Plain Network)

设计原则：

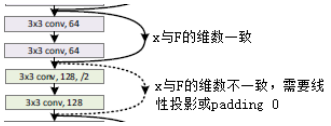
(1)对于输出特征图大小相同的层，它们的卷积拥有相同的filter个数

(2)如果输出的特征图大小减半，则filter个数乘以2，以确保每层的时间复杂度相同



5.2)残并网络(Residual Network)

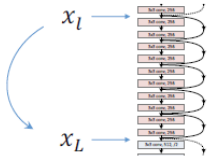
在遵循普通网络设计原则的基础上，增加了shortcut connections。



6)恒等映射的重要性

6.1)平滑的正向传播

$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$$



任意xl被直接正向传播到xL，xL是xl与残差相加的结果。

6.2)平滑的反向传播

$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$$

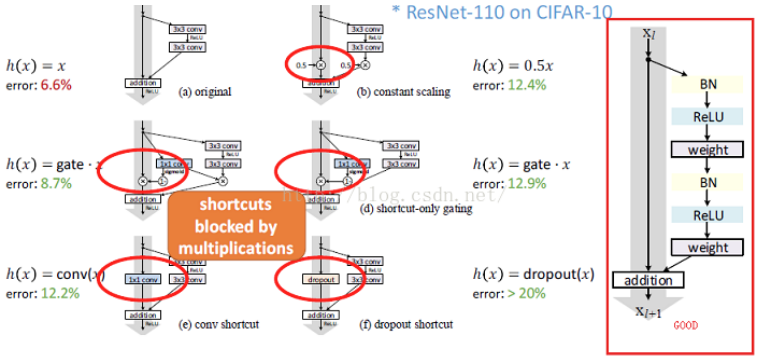
$$\frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L}\frac{\partial x_L}{\partial x_l} = \frac{\partial E}{\partial x_L}(1 + \frac{\partial}{\partial x_l}\sum_{i=l}^{L-1} F(x_i))$$

- Any $\frac{\partial E}{\partial x_L}$ is directly back-prop to any $\frac{\partial E}{\partial x_l}$, plus residual.
- Any $\frac{\partial E}{\partial x_l}$ is additive; unlikely to vanish

forward: $x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$

backward: $\frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L}(1 + \frac{\partial}{\partial x_l}\sum_{i=1}^{L-1} F(x_i))$

7)保持最短路径尽量平滑



- 如果h(x)不是identity mapping, 它将阻塞正向、反向传播, 从而导致误差增加

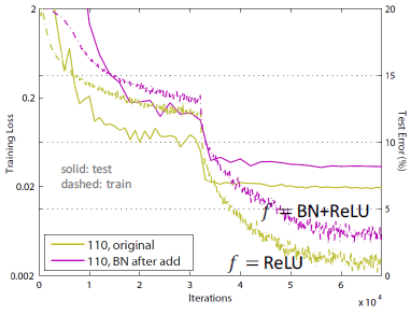
If h is multiplicative, e.g. $h(x) = \lambda x$

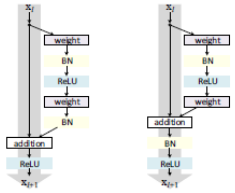
forward: $x_L = \lambda^{L-l} x_l + \sum_{i=l}^{L-1} \hat{F}(x_i)$

- if h is multiplicative, shortcuts are blocked
- direct propagation is decayed

backward: $\frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L}(\lambda^{L-l} + \frac{\partial}{\partial x_l}\sum_{i=1}^{L-1} \hat{F}(x_i))$

- BN可能阻塞传播

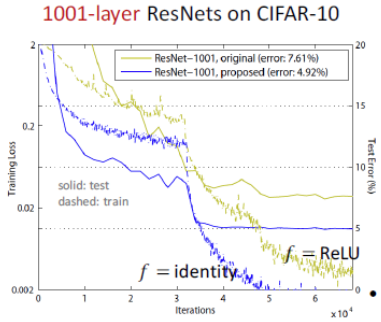


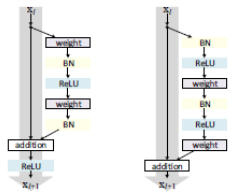


f = ReLU f = BN+ReLU

- BN could block prop
- Keep the shortest pass as smooth as possible

- ReLU可能阻塞传播





f = ReLU f = identity

- ReLU could block prop when there are 1000 layers
- pre-activation design eases optimization (and improves generalization; see paper)

3.5 提高目标检测方法

R-CNN系列目标检测框架和YOLO目标检测框架给了我们进行目标检测的两个基本框架。除此之外，研究人员基于这些框架从其他方面入手提出了一系列提高目标检测性能的方法。

(1) 难分样本挖掘（**hard negative mining**）

R-CNN在训练SVM分类器时使用了难分样本挖掘的思想，但Fast R-CNN和Faster R-CNN由于使用端到端的训练策略并没有使用难分样本挖掘（只是设置了正负样本的比例并随机抽取）。CVPR2016的Training Region-based Object Detectors with Online Hard Example Mining(oral)将难分样本挖掘(hard example mining)机制嵌入到SGD算法中，使得Fast R-CNN在训练的过程中根据region proposal的损失自动选取合适的

Region Proposal作为正负例训练。实验结果表明使用OHEM（Online Hard Example Mining）机制可以使得Fast R-CNN算法在VOC2007和VOC2012上mAP提高 4%左右。

(2) 多层特征融合

Fast R-CNN和Faster R-CNN都是利用了最后卷积层的特征进行目标检测，而由于高层的卷积层特征已经损失了很多细节信息（pooling操作），所以在定位时不是很精准。HyperNet等一些方法则利用了CNN的多层特征融合进行目标检测，这不仅利用了高层特征的语义信息，还考虑了低层特征的细节纹理信息，使得目标检测定位更精准。

(3) 使用上下文信息

在提取Region Proposal特征进行目标检测时，结合Region Proposal上下文信息，检测效果往往会更好一些。（Object detection via a multi-region & semantic segmentation-aware CNN model以及Inside-Outside Net等论文中都使用了上下文信息）

3.6 总结

Localization:

- Find a fixed number of objects (one or many)
- L2 regression from CNN features to box coordinates
- Much simpler than detection; consider it for your projects!
- Overfeat: Regression + efficient sliding window with FC -> conv conversion
- Deeper networks do better

Object Detection:

- Find a variable number of objects by classifying image regions
- Before CNNs: dense multiscale sliding window (HoG, DPM)
- Avoid dense sliding window with region proposals
- R-CNN: Selective Search + CNN classification / regression
- Fast R-CNN: Swap order of convolutions and region extraction
- Faster R-CNN: Compute region proposals within the network
- Deeper networks do better