炎炎课堂
Yan yan classroom

深度学习之

图像处理

ibaotu.com

深度学习之
图像处理

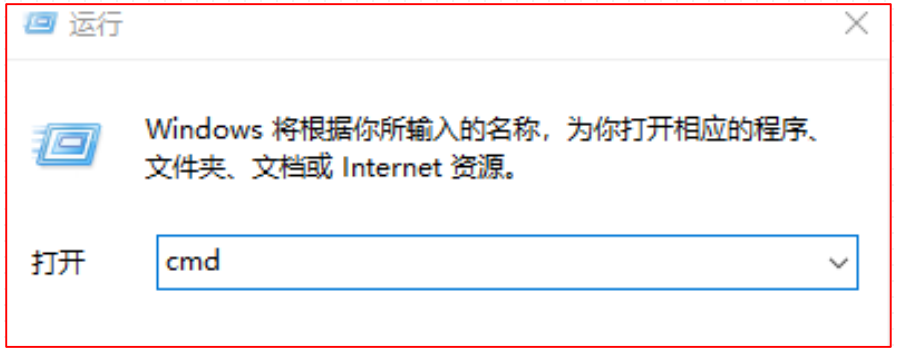
1. Anaconda 安装
Anaconda3-5.2.0下载地址
https://repo.continuum.io/archive/Anaconda2-5.2.0-Windows-x86_64.exe
注:安装路径不要有中文
配置环境变量，按下面博客的教程
https://www.jianshu.com/p/cdcd1c6d5a19

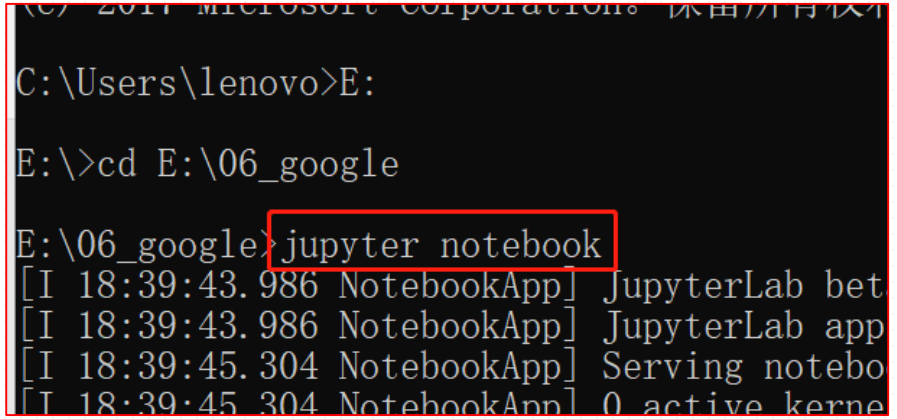
安装graphviz
https://www.cnblogs.com/shuodehaoa/p/8667045.html

3.安装tensorflow&keras
pip install tensorflow==1.5.0
pip install keras==2.1.4

4.安装其他软件
pip install jupyter
pip install opencv-python
pip install dlib
pip install sklearn
pip install pydot_ng

5.资源下载地址：
http://mooc.yanyantech.com/06_google.rar

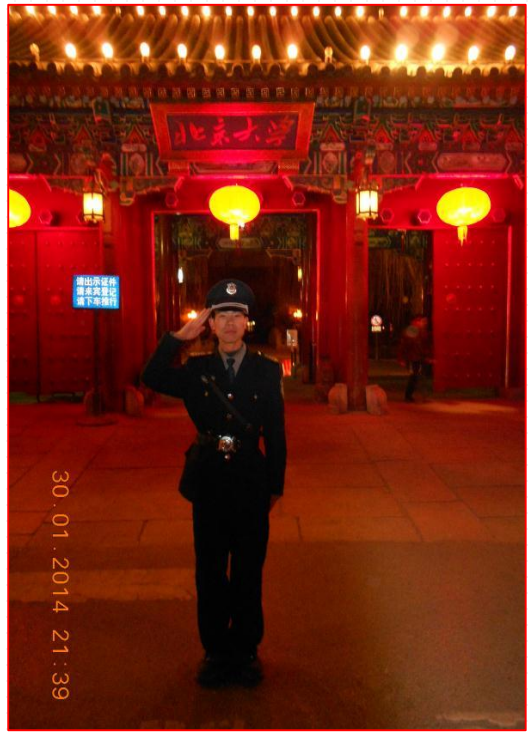




切换到解压的数据目录,解压的文件目录不要有中文
输入：jupyter notebook

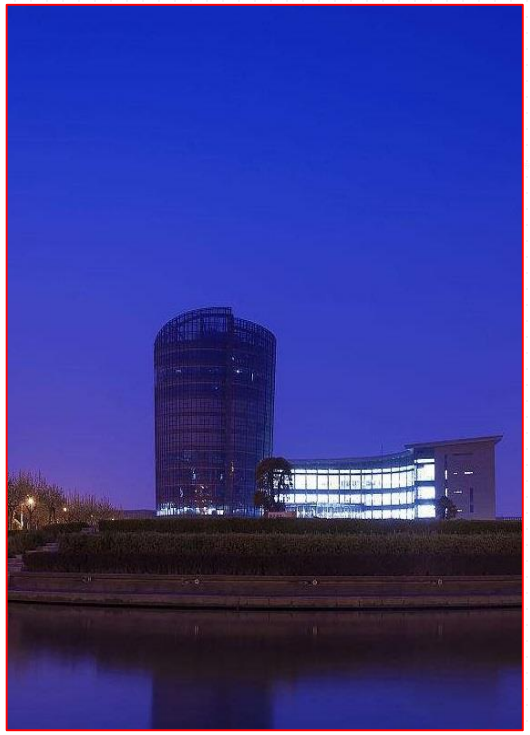
自我介绍

闻过则喜 祛衣受业
伯乐一顾 不胜感激



北大保安

爱国，进步，民主，科学



华东师范大学

生物医学工程
新中国成立后的第一所社会主义高等师范院校

翼支付

风险管理部建模中心
最受信赖的金融科技企业

炎炎课堂

负责人
交流学习，知识共享

目录 / CONTENTS

导数&偏导数

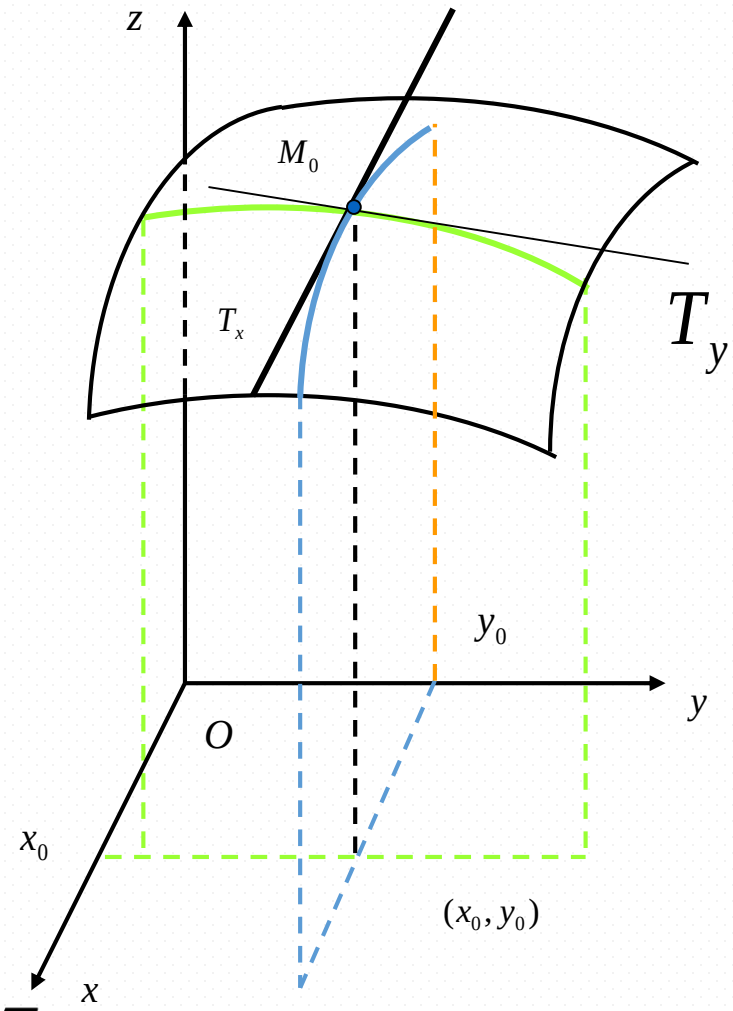
$$\left.\frac{\partial f}{\partial x}\right|_{\substack{x=x_0\\y=y_0}} = \left.\frac{\mathrm{d}}{\mathrm{d}x} f(x, y_0)\right|_{x=x_0}$$

$$\left.\frac{\partial f}{\partial y}\right|_{\substack{x=x_0\\y=y_0}} = \left.\frac{\mathrm{d}}{\mathrm{d}y} f(x_0, y)\right|_{y=y_0}$$

$z = x^2 + 3xy + y^2$ 在点(1 , 2) 处的偏导数.

$$\frac{\partial z}{\partial x} = 2x + 3y, \qquad \frac{\partial z}{\partial y} = 3x + 2y$$

$$\therefore \left.\frac{\partial z}{\partial x}\right|_{(1,2)} = 2\cdot 1 + 3\cdot 2 = 8, \qquad \left.\frac{\partial z}{\partial y}\right|_{(1,2)} = 3\cdot 1 + 2\cdot 2 = 7$$



梯度

$$\boldsymbol{gradf(x,y)=\frac{\partial f}{\partial x}\vec{i}+\frac{\partial f}{\partial y}\vec{j}}$$

$z = x^2 + 3xy + y^2$ 在点(1，2)处的梯度.

$$gradu(x,y) = \frac{\partial u}{\partial x}\vec{i} + \frac{\partial u}{\partial y}\vec{j}$$

$$= (2x + 3y)\vec{i} + (3x + 2y)\vec{j}$$

$$gradu(1,2) = 8\vec{i} + 7\vec{j}$$

$$\boldsymbol{gradf(x,y,z)=\frac{\partial f}{\partial x}\vec{i}+\frac{\partial f}{\partial y}\vec{j}+\frac{\partial f}{\partial z}\vec{k}.}$$

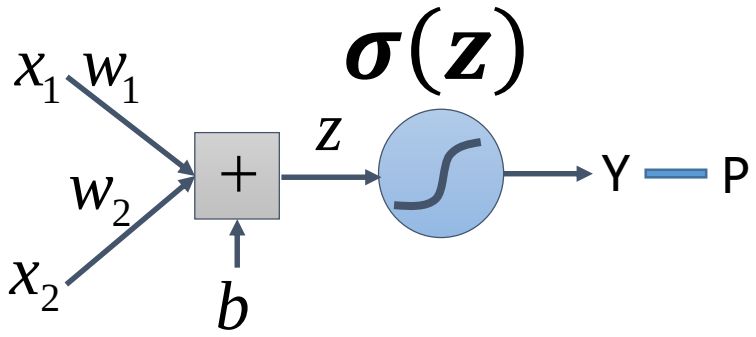
基础知识点回顾

损失函数

Y	期望(真实)类标签分布		
	class#0	class#1	class#2
sample#1	0	1	0
sample#2	1	0	0
sample#3	0	1	0
sample#4	0	0	1
sample#5	1	0	0

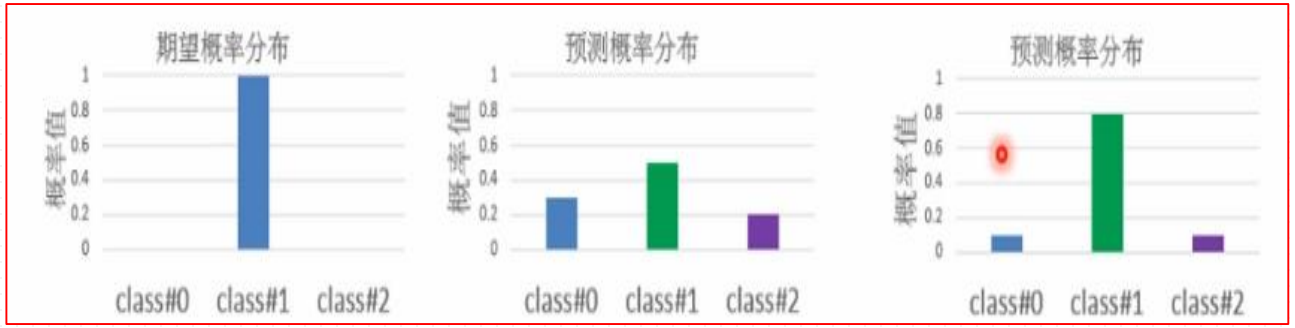
P	预测类标签分布		
	class#0	class#1	class#2
sample#1	0.2	0.7	0.1
sample#2	0.5	0.2	0.3
sample#3	0.3	0.4	0.3
sample#4	0.2	0.3	0.5
sample#5	0.3	0.3	0.4

损失函数(Loss function)：是定义在单个训练样本的损失或者误差，也就是就算一个样本的误差。比如我们想要分类，就是预测的类别和实际类别的区别。



常见损失函数

y 是离散型变量	
0-1 损失函数	$L(y,h(x)) = \begin{cases} 1, & y \cdot h(x) < 0 \\ 0, & y \cdot h(x) > 0 \end{cases} = I\{y \cdot h(x) < 0\}$
对数 损失函数	$L(y,h(x)) = \log\left(1 + \exp(-2y \cdot h(x))\right)$
指数 损失函数	$L(y,h(x)) = \exp(-y \cdot h(x))$
支持向量 损失函数	$L(y,h(x)) = (1 - y \cdot h(x))^+$
y 是连续型变量	
L_2 损失函数	$L(y,h(x)) = \frac{1}{2}[y - h(x)]^2$
L_1 损失函数	$L(y,h(x)) = \lvert y - h(x) \rvert$
Huber 损失函数	$L(y,h(x))_\delta = \begin{cases} [y - h(x)]^2/2, & \lvert y - h(x) \rvert \le \delta \\ \delta(\lvert y - h(x) \rvert - \delta/2), & \lvert y - h(x) \rvert > \delta \end{cases}$



Y　　P1　　P2

代价函数

Y	期望(真实)类标签分布		
	class#0	class#1	class#2
sample#1	0	1	0
sample#2	1	0	0
sample#3	0	1	0
sample#4	0	0	1
sample#5	1	0	0

P	预测类标签分布		
	class#0	class#1	class#2
sample#1	0.2	0.7	0.1
sample#2	0.5	0.2	0.3
sample#3	0.3	0.4	0.3
sample#4	0.2	0.3	0.5
sample#5	0.3	0.3	0.4

代价函数(Cost function)：是定义在整个训练集整体的误差描述，也就是所有样本的误差的总和的平均，也就是损失函数的总和的平均，有没有这个平均其实不会影响最后的参数的求解结果。

常见代价函数

平方代价函数

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\tilde{Y}_i - Y_i)^2$$

指数代价函数

$$C(y, f(x)) = \frac{1}{n}\sum_{i=1}^{n} exp[-y_i f(x_i)]$$

log对数代价函数

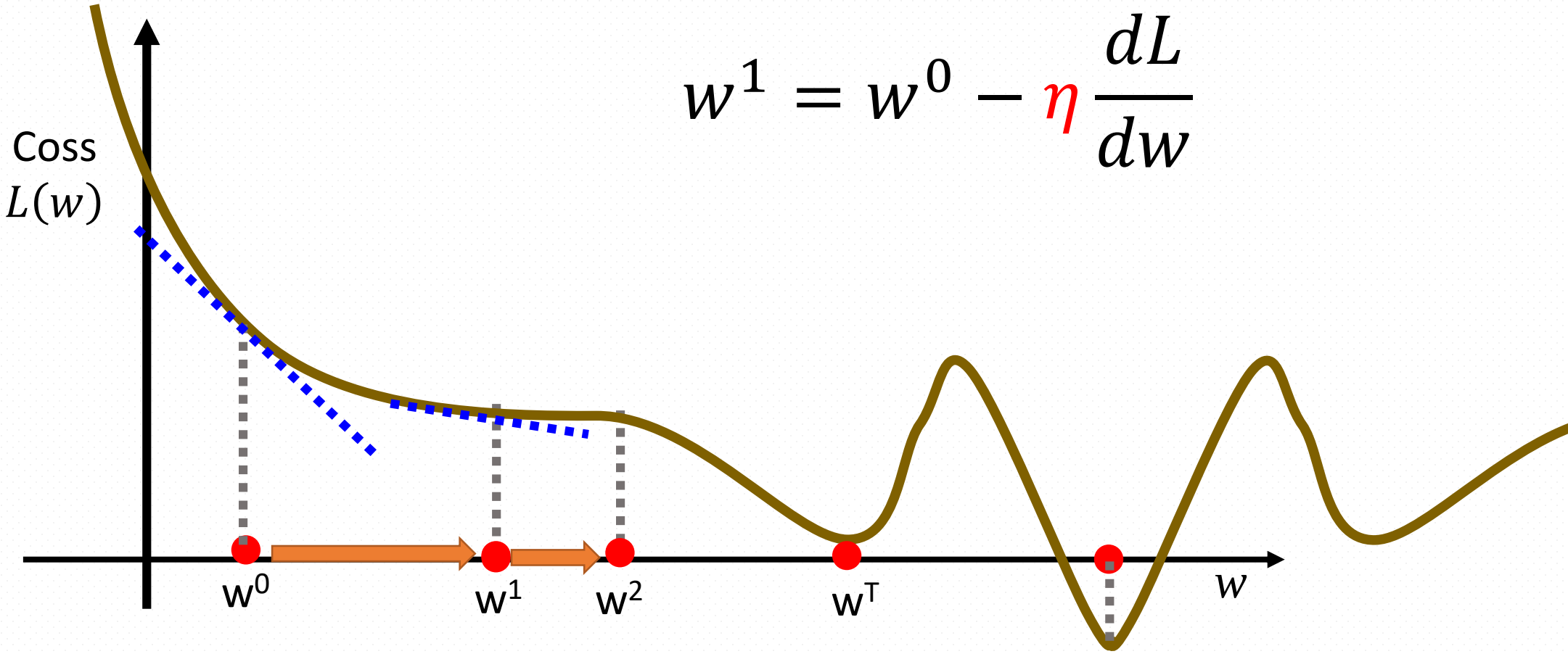
$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

铰链代价函数

$$\min_{w,b}\sum_{i}^{N}[1 - y_i(w\cdot x_i + b)]$$

Gradient Descent 梯度下降

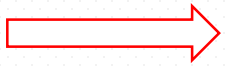
$$w^1 = w^0 - \eta \frac{dL}{dw}$$



基础知识点回顾

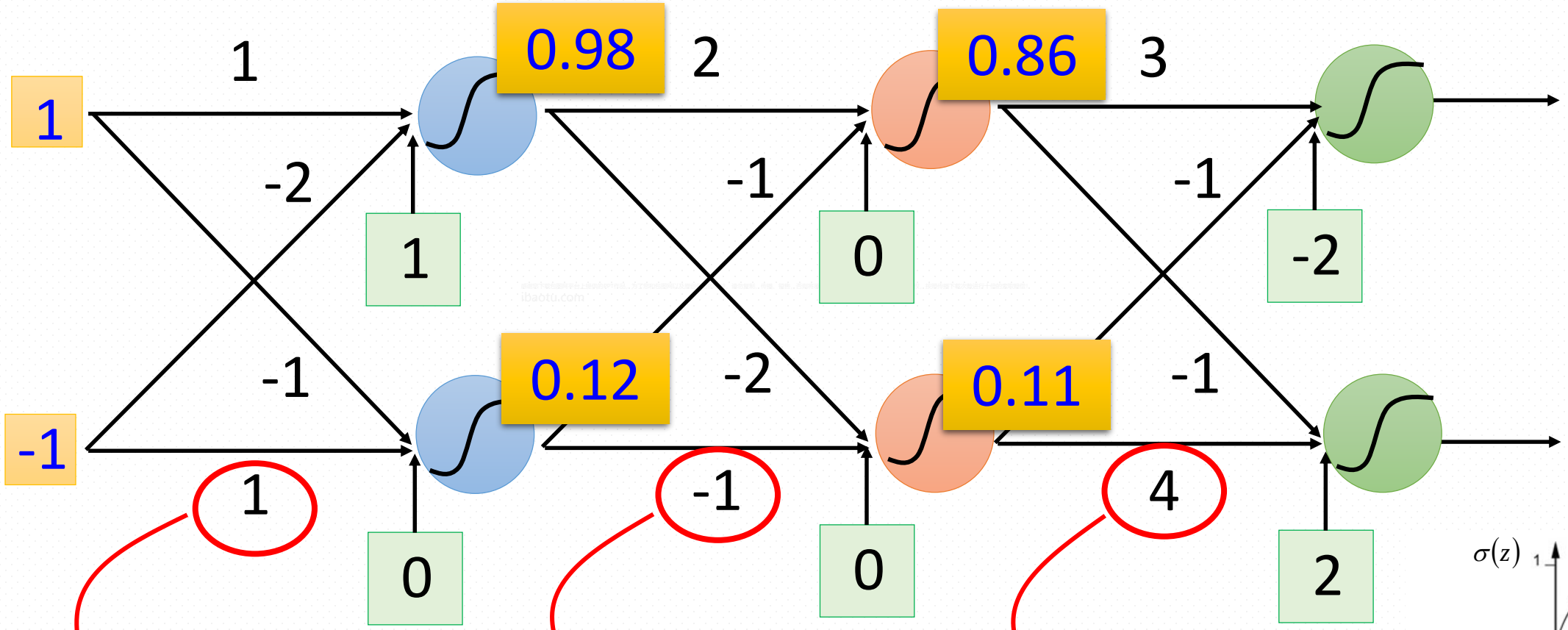
建模基本步骤

1 导入或者生成样本数据集
2 转换和归一化数据
3 特征选择
4 划分样本数据集为训练样本集、测试样本集和验证样本集
5 设置机器学习参数（超参数）
6 初始化变量
7 定义模型结构
8 声明代价函数
9 初始化模型和训练模型
10 评估机器学习模型
11 调优超参数
13 模型上线
14 发布和预测结果



6 初始化变量
7 定义模型结构
8 声明代价函数
9 初始化模型和训练模型

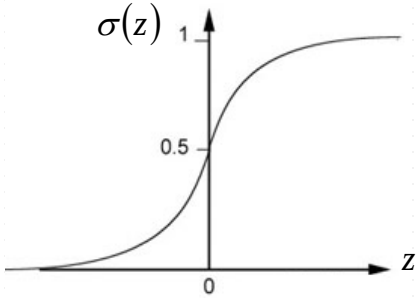
BP神经网络案例



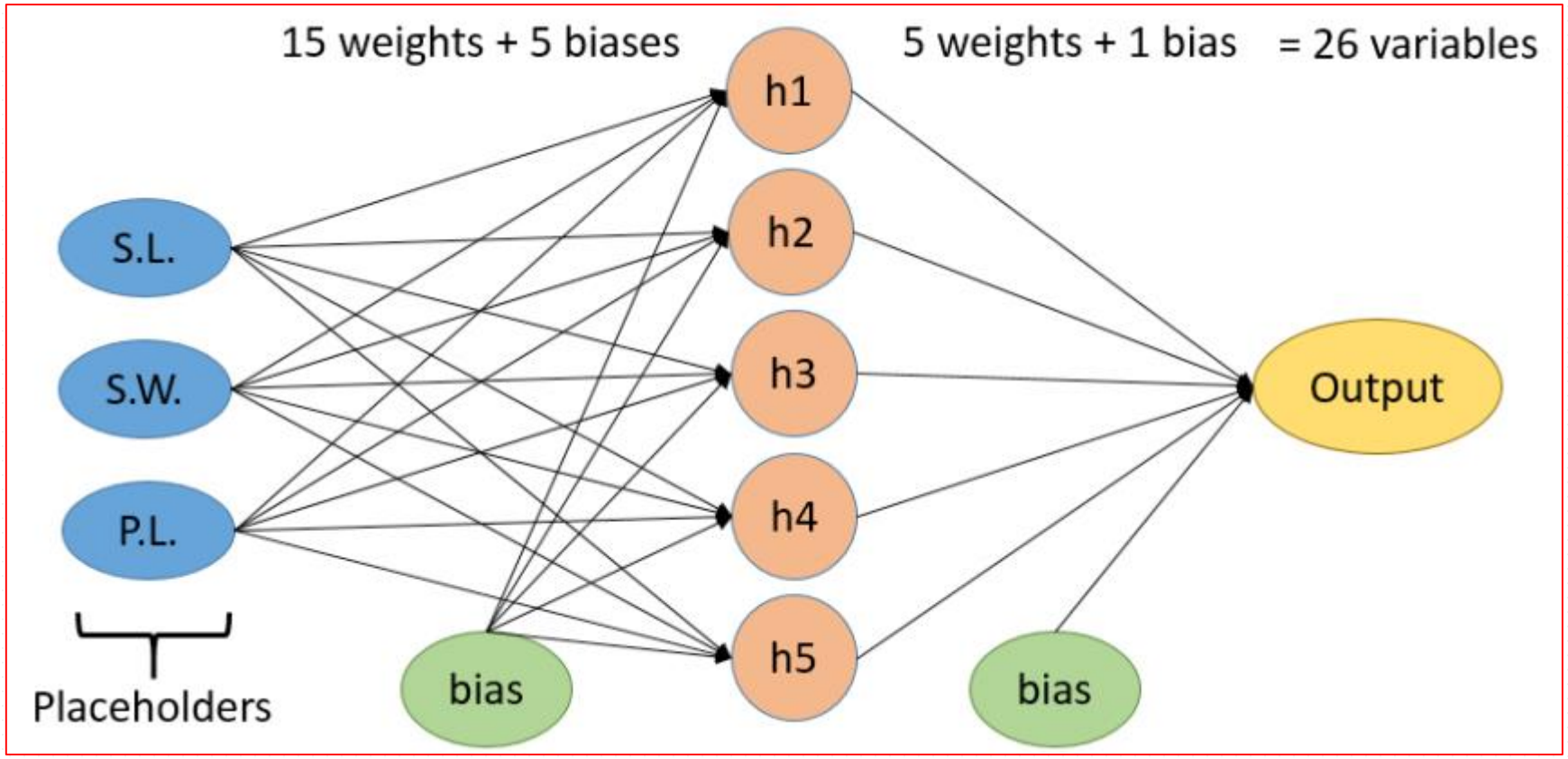
$$\frac{\partial z}{\partial w} = -1$$

$$\frac{\partial z}{\partial w} = 0.12$$

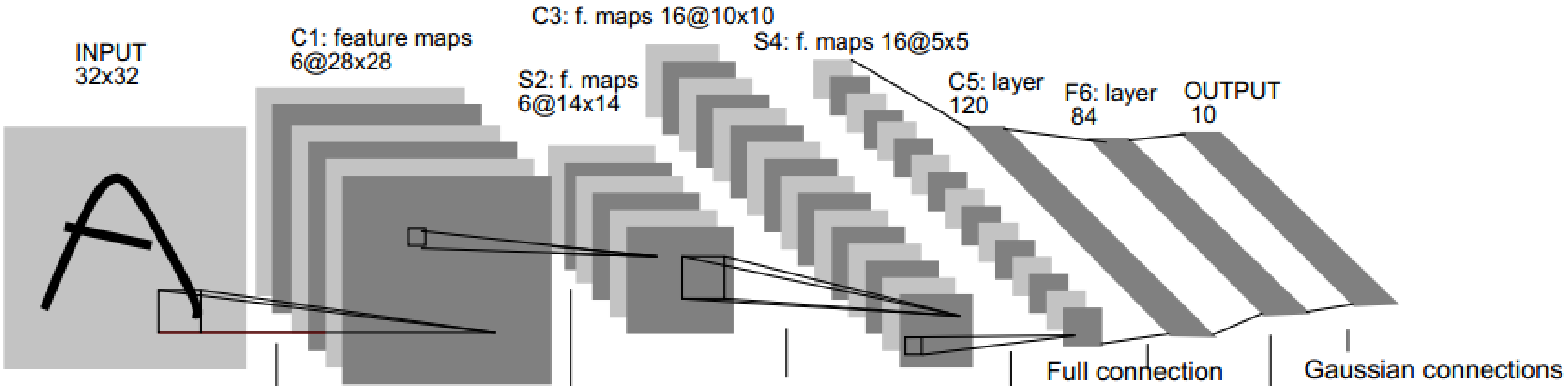
$$\frac{\partial z}{\partial w} = 0.11$$



BP神经网络案例


15 weights + 5 biases
5 weights + 1 bias
= 26 variables
S.L.
S.W.
P.L.
Placeholders
bias
h1
h2
h3
h4
h5
Output
bias

卷积神经网络结构讲解& 图像分类



卷积

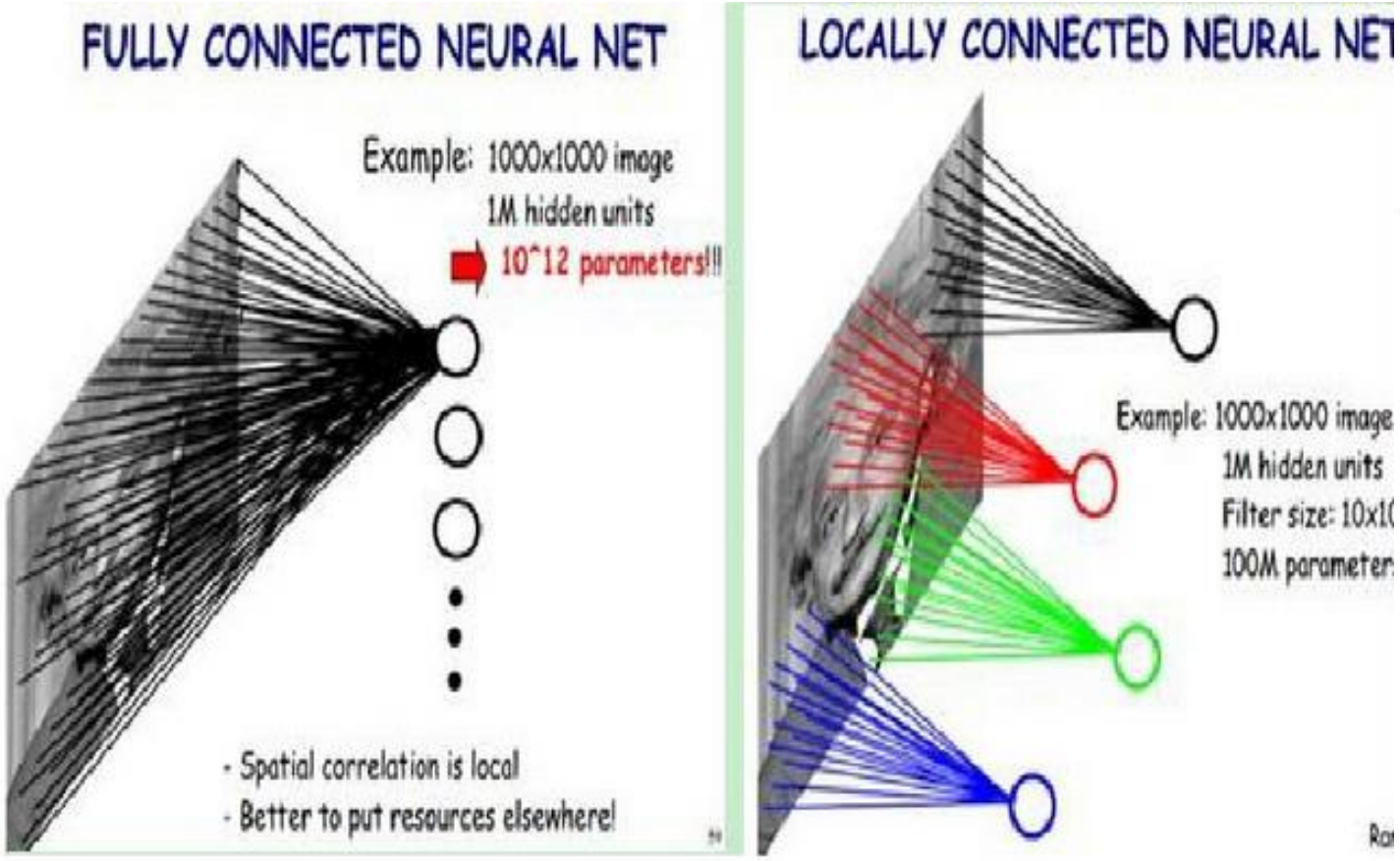
池化

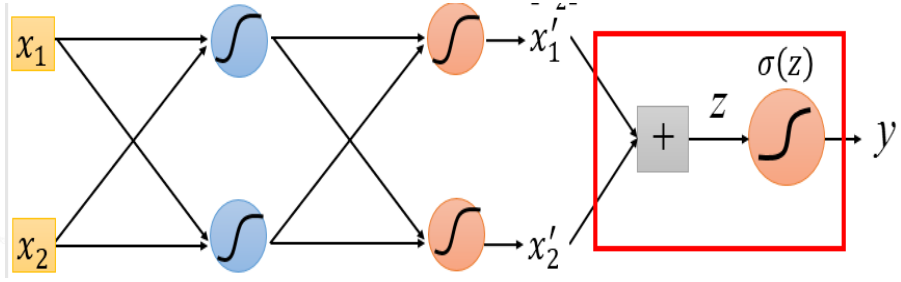
卷积

池化

BP神经网络

局部连接





1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

1	-1	-1
-1	1	-1
-1	-1	1

权值共享

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

权值共享

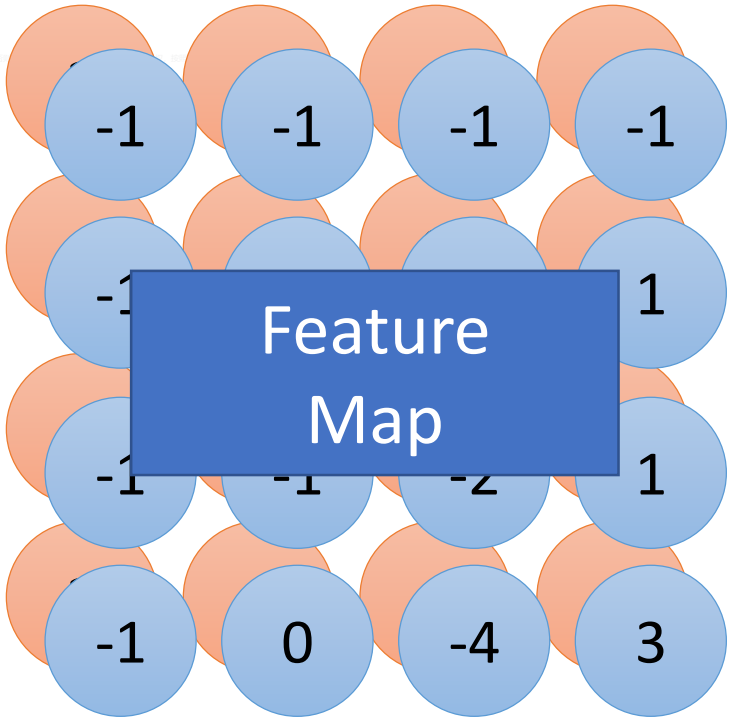
stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2



4 x 4 image

http://cs231n.github.io/convolutional-networks/

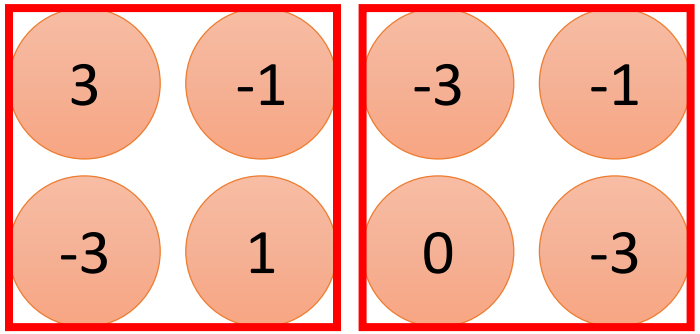
池化

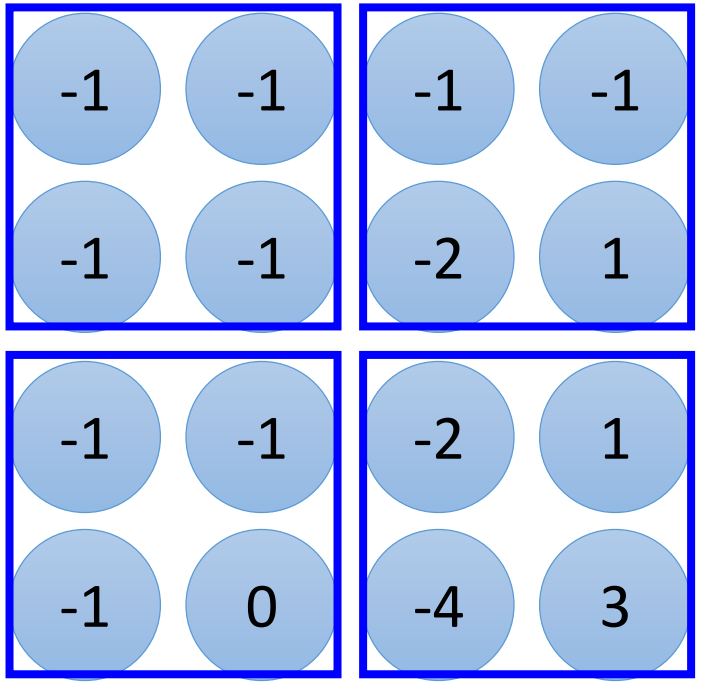
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

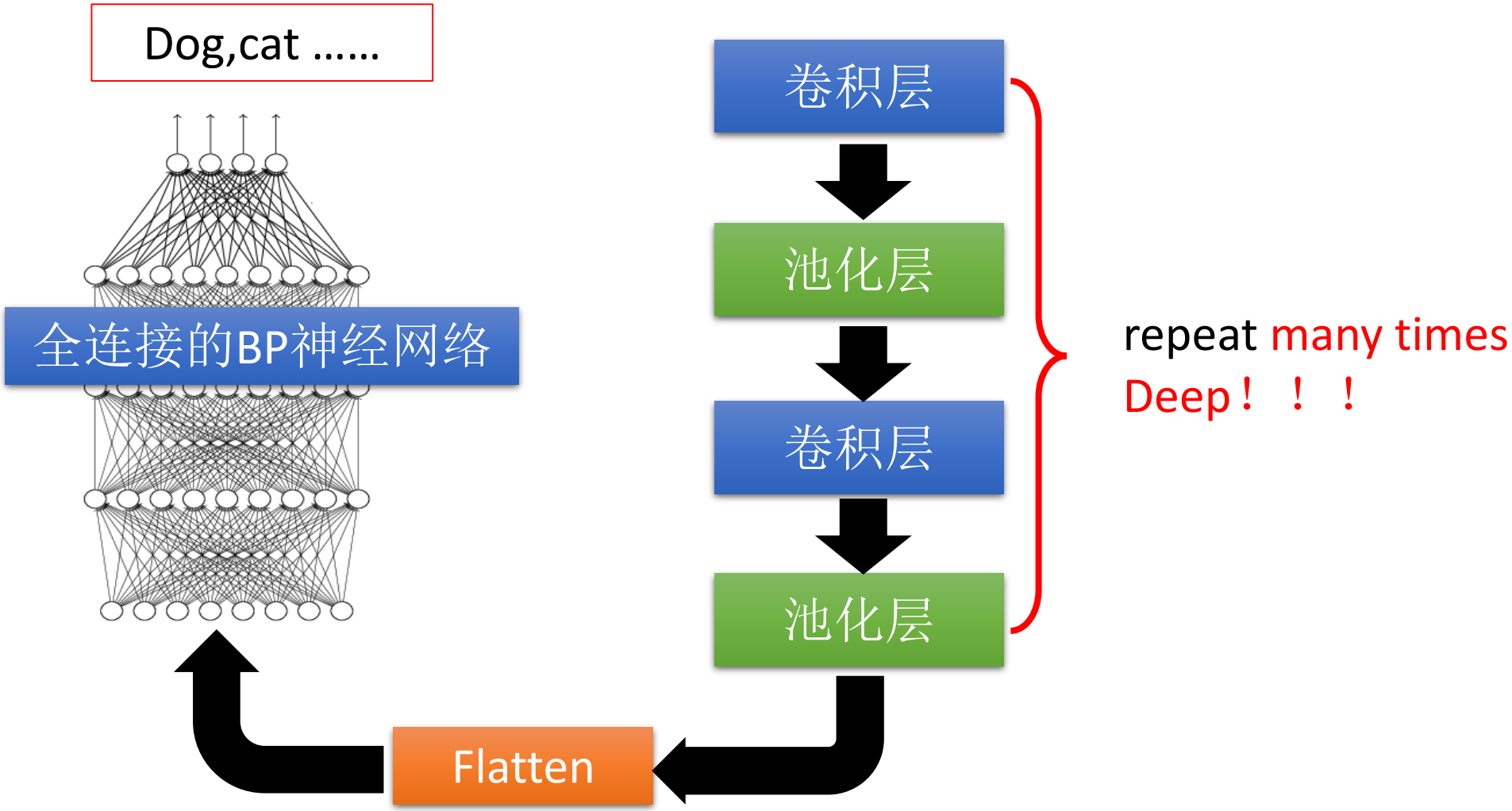
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

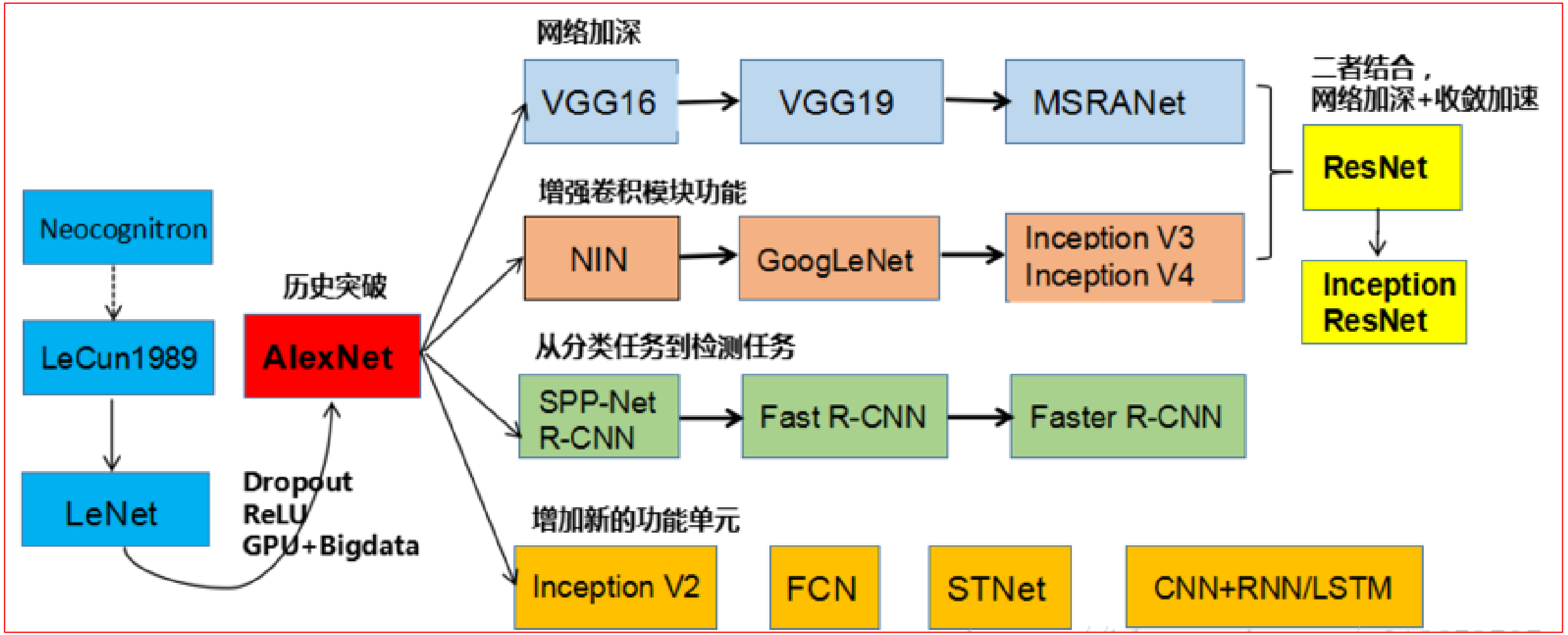




卷积网络结构


Dog,cat ……
全连接的BP神经网络
卷积层
池化层
卷积层
池化层
Flatten
repeat many times
Deep！ ！ ！

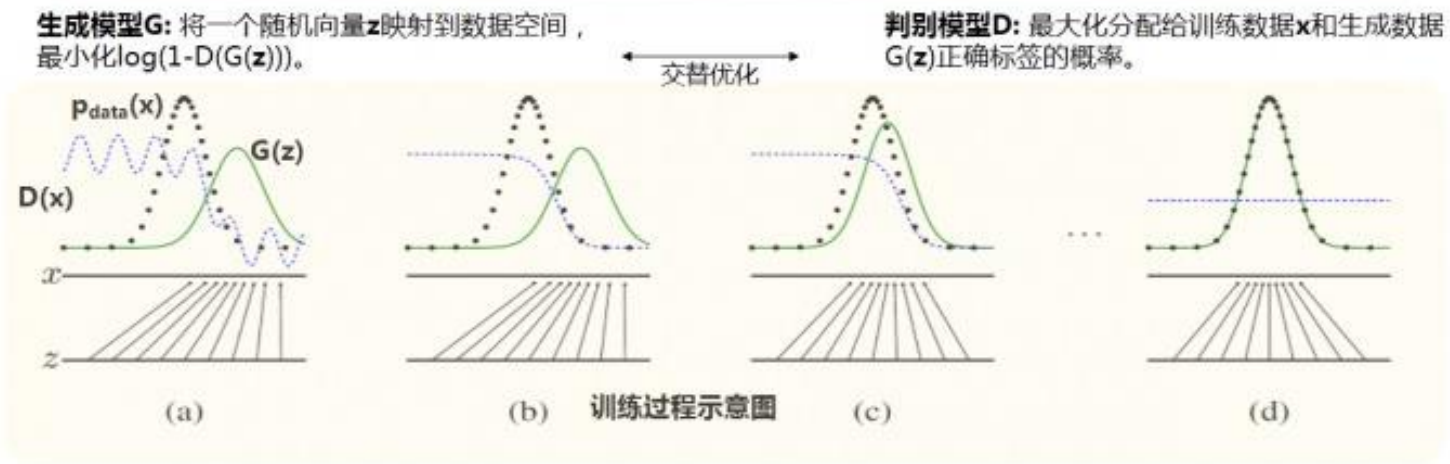
卷积网络发展

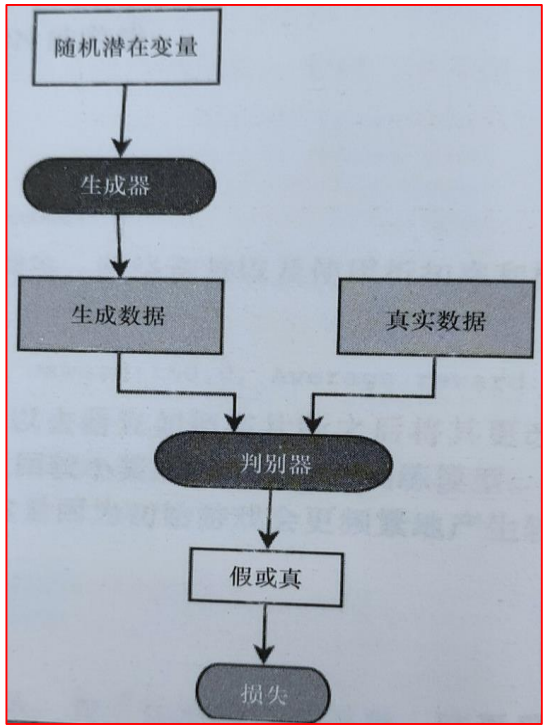

网络加深
VGG16
VGG19
MSRANet
二者结合，
网络加深+收敛加速
ResNet
Inception
ResNet
增强卷积模块功能
NIN
GoogLeNet
Inception V3
Inception V4
Neocognitron
历史突破
LeCun1989
AlexNet
从分类任务到检测任务
SPP-Net
R-CNN
Fast R-CNN
Faster R-CNN
LeNet
Dropout
ReLU
GPU+Bigdata
增加新的功能单元
Inception V2
FCN
STNet
CNN+RNN/LSTM

对抗生成网络结构讲解& 图像生成

对抗生成网络结构讲解& 图像生成

原理





图(a)中黑色大点虚线P(x)是真实的数据分布，绿线G(z)是通过生成模型产生的数据分布（输入是均匀分布变量z，输出是绿色的曲线）。蓝色的小点虚线D(x)代表判别函数。

在图(a)中，我们可以看到，绿线G(z)分布和黑色P(x)真实分布，还有比较大的差异。这点也反映在蓝色的判别函数上，判别函数能够准确的对左面的真实数据输入，输出比较大的值。对右面虚假数据，产生比较小的值。但是随着训练次数的增加，图（b）和图（c）反映出，绿色的分布在逐渐靠近黑色的分布。到图（d），产生的绿色分布和真实数据分布已经完全重合。这时，判别函数对所有的数据（无论真实的还是生成的数据），输出都是一样的值，已经不能正确进行分类。G成功学习到了数据分布，这样就达到了GAN的训练和学习目的。

损失函数

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

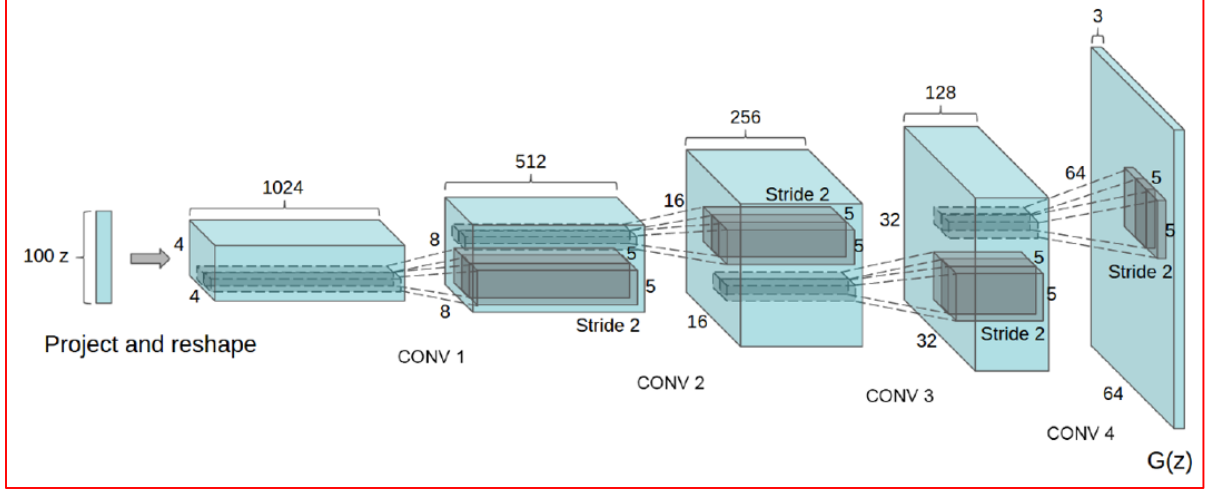
它做的是去最大化D的区分度，最小化G和real数据集的数据分布

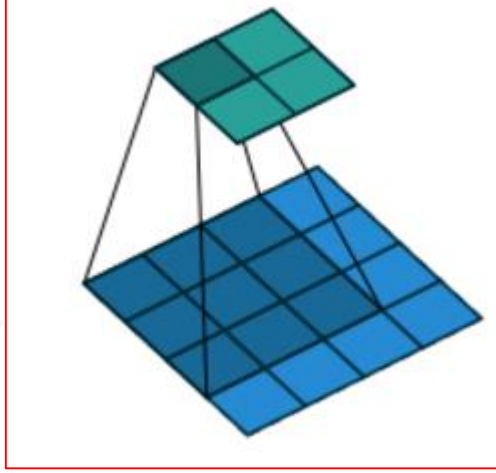
判别模型：$\log(D_1(x)) + \log(1 - D_2(G(z)))$

生成模型：$\log(D_2(G(z)))$

对抗生成网络结构讲解& 图像生成

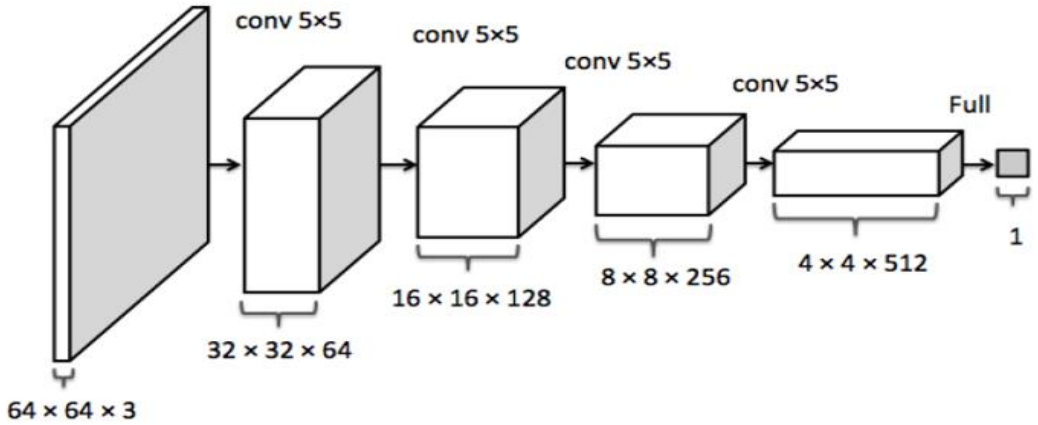
DCGAN





https://github.com/vdumoulin/conv_arithmetic

反卷积



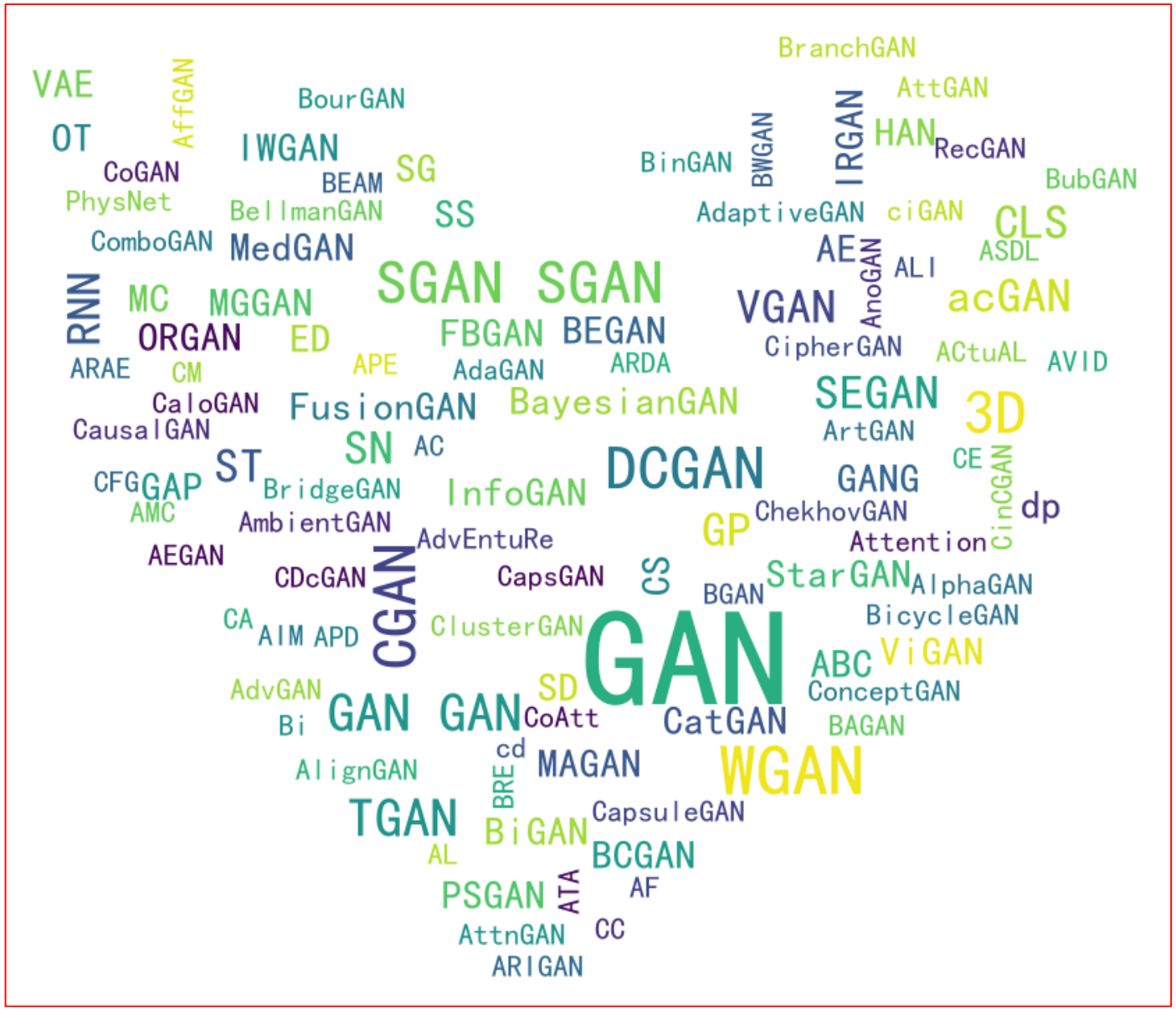
判别模型：使用带步长的卷积（strided convolutions）取代了的空间池化（spatial pooling），容许网络学习自己的空间下采样（spatial downsampling）。

生成模型：使用微步幅卷积（fractional strided），容许它学习自己的空间上采样（spatial upsampling）。
激活函数： LeakyReLU

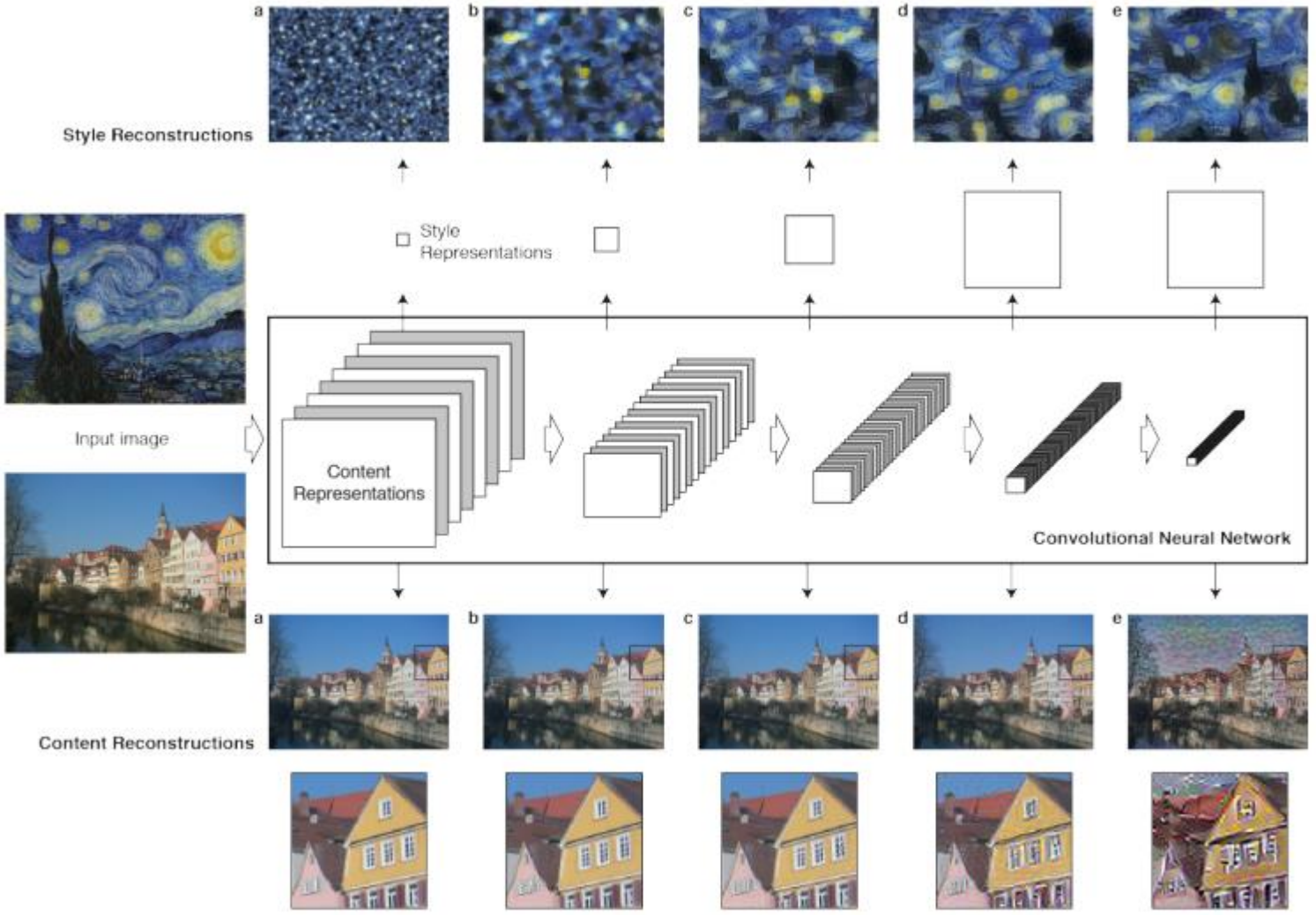
Batch Normalization 批标准化：解决因糟糕的初始化引起的训练问题，使得梯度能传播更深层次。 Batch Normalization证明了生成模型初始化的重要性，避免生成模型崩溃：生成的所有样本都在一个点上（样本相同），这是训练GANs经常遇到的失败现象。

对抗生成网络结构讲解& 图像生成

GAN发展



风格迁移


a
b
c
d
e
Style Reconstructions
Style Representations
Input image
Content Representations
Convolutional Neural Network
Content Reconstructions

格拉姆矩阵

$$G = A^T A = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \vdots \\ \mathbf{a}_n^T \end{bmatrix} \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1^T\mathbf{a}_1 & \mathbf{a}_1^T\mathbf{a}_2 & \cdots & \mathbf{a}_1^T\mathbf{a}_n \\ \mathbf{a}_2^T\mathbf{a}_1 & \mathbf{a}_2^T\mathbf{a}_2 & \cdots & \mathbf{a}_2^T\mathbf{a}_n \\ & & & \\ \mathbf{a}_n^T\mathbf{a}_1 & \mathbf{a}_n^T\mathbf{a}_2 & \cdots & \mathbf{a}_n^T\mathbf{a}_n \end{bmatrix}$$

Gram矩阵和协方差矩阵的差别在于，Gram矩阵没有白化，也就是没有减去均值，直接使用两向量做内积。
Gram矩阵和相关系数矩阵的差别在于，Gram矩阵既没有白化，也没有标准化（也就是除以两个向量的标准差）。
这样，Gram所表达的意义和协方差矩阵相差不大，只是显得比较粗糙。两个向量的协方差表示两向量之间的相似程度，协方差越大越相似。对角线的元素的值越大，表示其所代表的向量或者说特征越重要。

风格生成网络结构讲解& 图像风格迁移

损失函数

假设一个卷积层包含 N_l 个过滤器 `filters`，则可以得到 N_l 个 `feature maps`，假设 `feature map` 的大小是 M_l (长乘宽)，则可以通过一个矩阵来存储 `l` 层的数据

$$F^l \in R^{N_l \times M_l}$$

- $F^l_{i,j}$ 表示第 `l` 层的第 `i` 个 `filter` 在 `j` 位置上的激活值

所以现在一张内容图片 $\vec{p}$，一张生成图片 $\vec{x}$ (初始值为高斯分布), 经过一层**卷积层l**可以得到其对应的特征表示：P^l 和 F_l, 则对应的损失采用**均方误差**:

$$L_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2}\sum_{ij}(F^l_{ij} - P^l_{ij})^2$$

- F 和 P是两个矩阵，大小是$N_l \times M_l$，即 `l` 层过滤器的个数 和 `feature map` 的长乘宽的值

内容损失

风格的表示这里采用**格拉姆矩阵**(`Gram Matrix`): $G^l \in R^{N_l \times N_l}$

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}$$

- 格拉姆矩阵计算的是**两两特征的相关性**，即哪两个特征是同时出现的，哪两个特征是此消彼长的等，能够**保留图像的风格**
- (比如一幅画中有人和树，它们可以出现在任意位置，格拉姆矩阵可以衡量它们之间的关系，可以认为是这幅画的风格信息)

假设 $\vec{a}$ 是风格图像，$\vec{x}$ 是生成图像，A^l 和 G^l 表示在 l 层的格拉姆矩阵，则这一层的损失为：

$$E_l = \frac{1}{4N_l^2 M_l^2}\sum_{i,j}(G^l_{ij} - A^l_{ij})^2$$

提取风格信息是我们会使用多个卷积层的输出，所以总损失为：

$$L_{style}(\vec{a}, \vec{x}) = \sum_l^L w_l E_l$$

风格损失

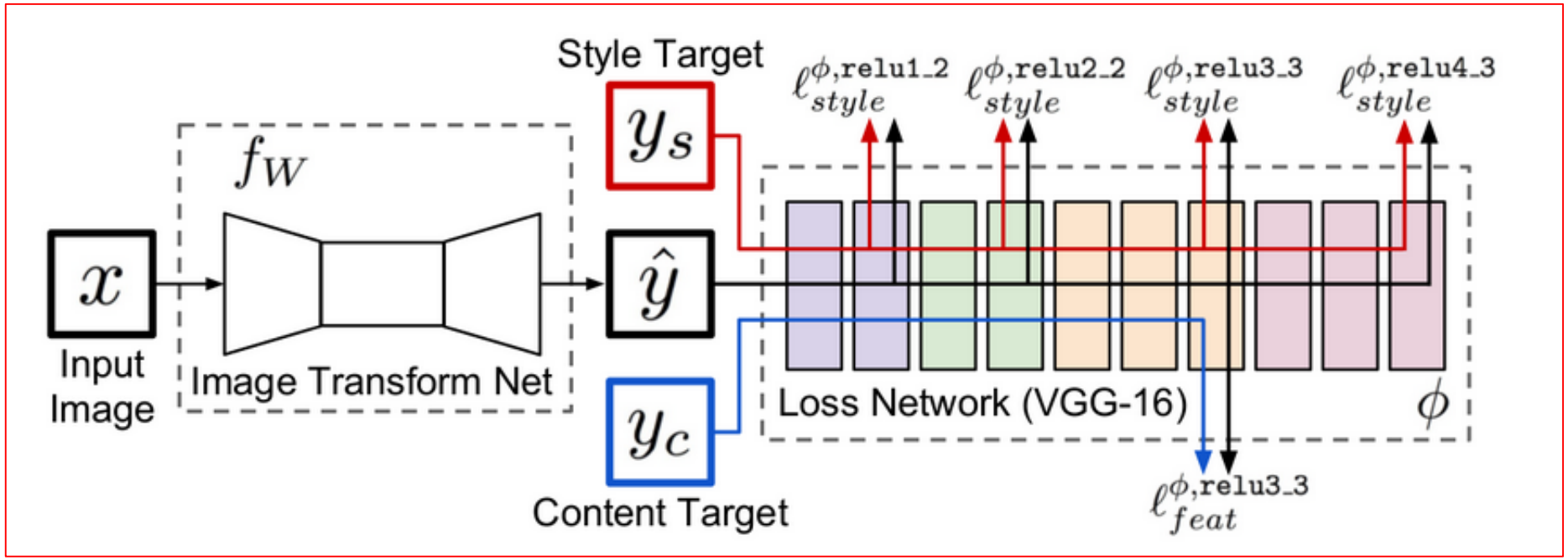
- 通过**白噪声初始化**(就是高斯分布)一个输出的结果，然后通过网络对这个结果进行风格和内容两方面的约束进行修正

$$L_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha L_{content}(\vec{p}, \vec{x}) + \beta L_{style}(\vec{a}, \vec{x})$$

总损失

风格生成网络结构讲解& 图像风格迁移

快速图像风格迁移



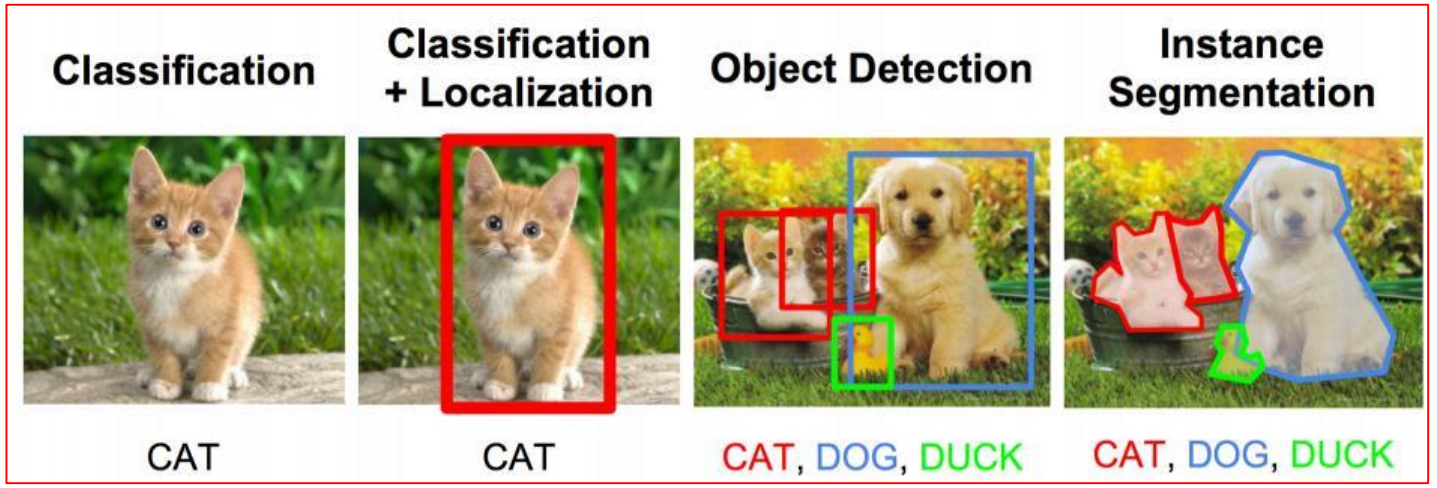
转换网络

损失网络

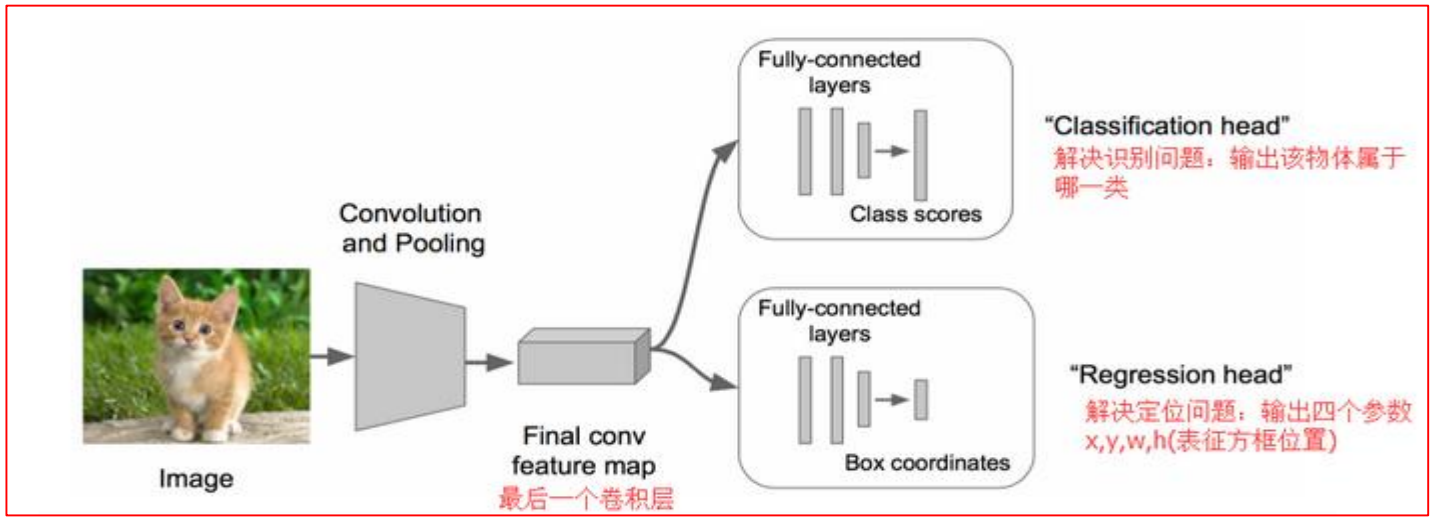
转换网络：参数需要训练，将内容图片转换成迁移图片
损失网络：计算迁移图片和风格图片之间的风格损失，以及迁移图片和原始内容图片之间的内容损失

R-CNN、Fast R-CNN、Faster R-CNN 网络结构讲解& 图像检测

问题描述



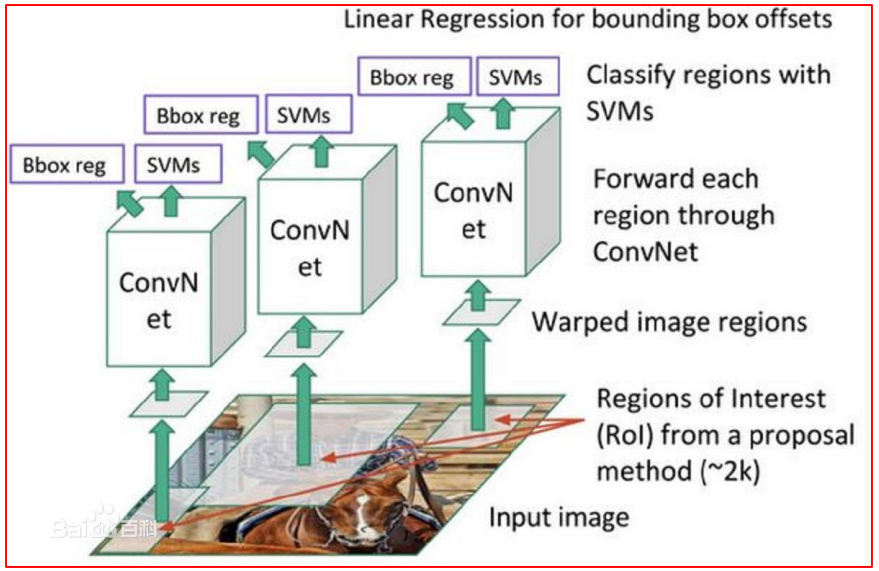
识别定位

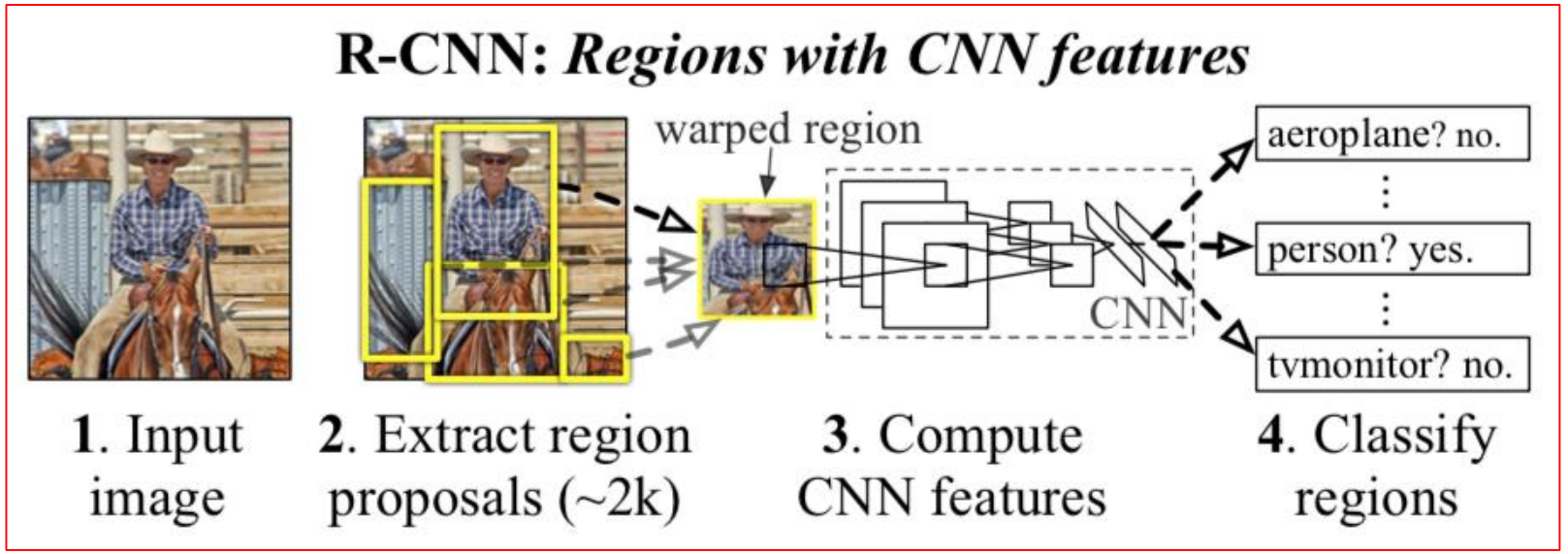


分类回归

R-CNN、Fast R-CNN、Faster R-CNN 网络结构讲解& 图像目标

R-CNN



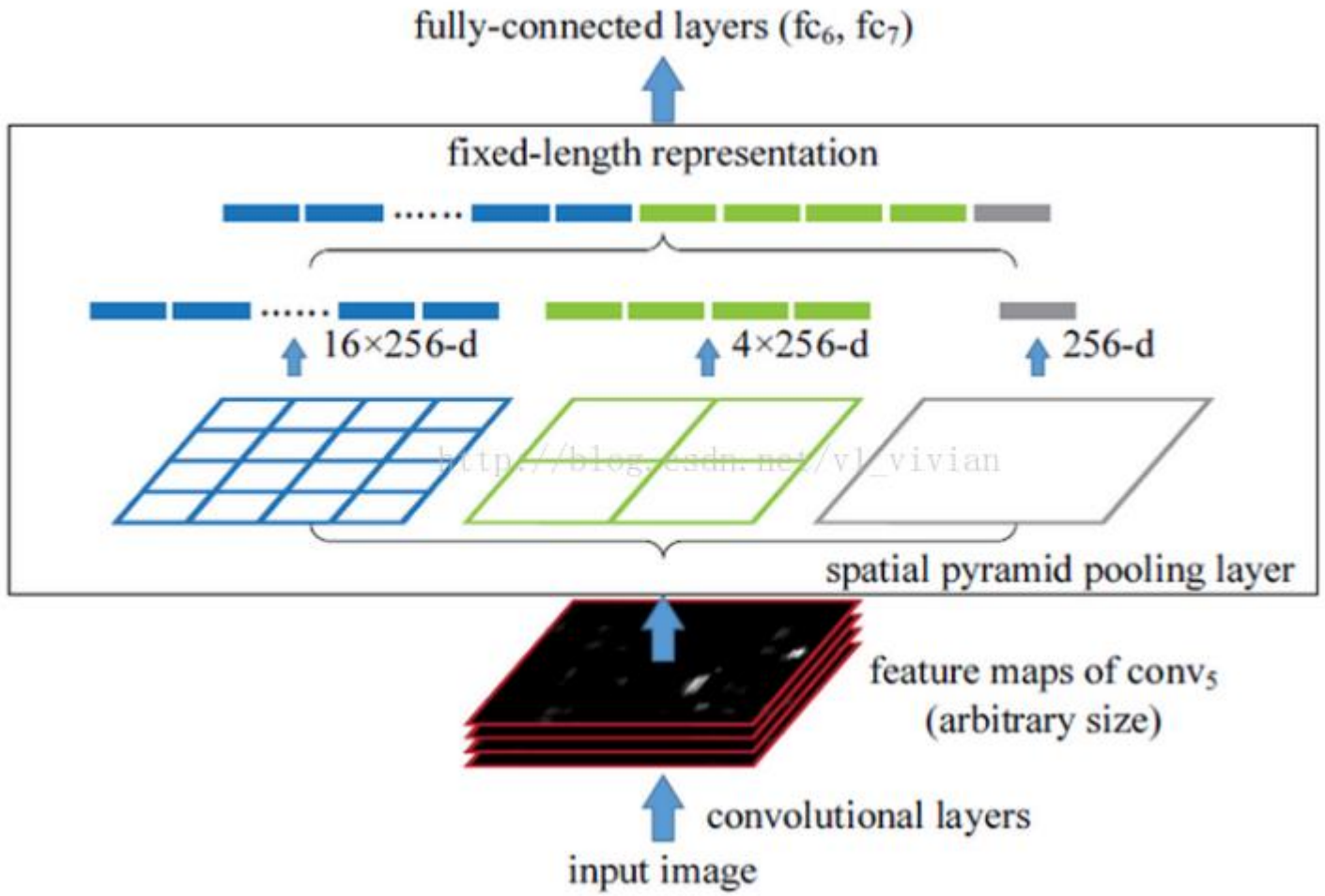


1.获取输入图像
2.提取约2000个候选区域(selective search)
3.缩放候选区域
4.将候选区域分别输入CNN网络
5.将CNN的输出输入SVM中进行类别的判定

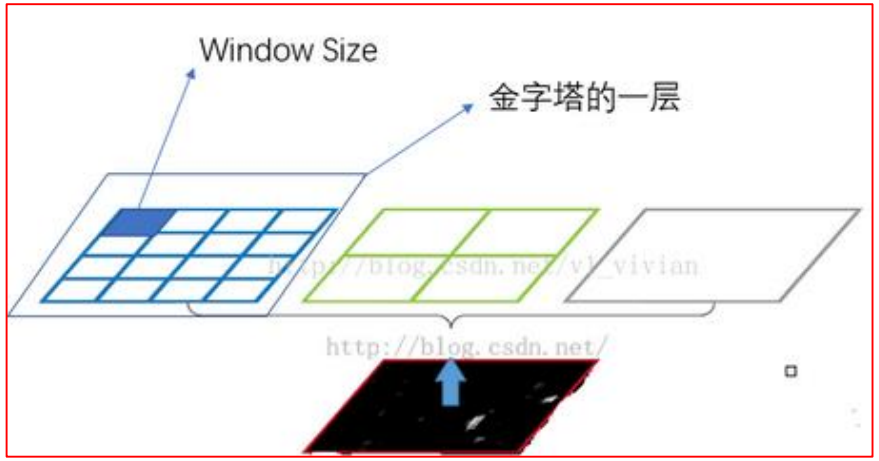
1. 训练步骤繁琐（微调网络+训练SVM+训练bbox）；
2.训练、测试均速度慢；
3.训练占空间
4.推理比较慢

R-CNN、Fast R-CNN、Faster R-CNN 网络结构讲解& 图像目标

SPPNet



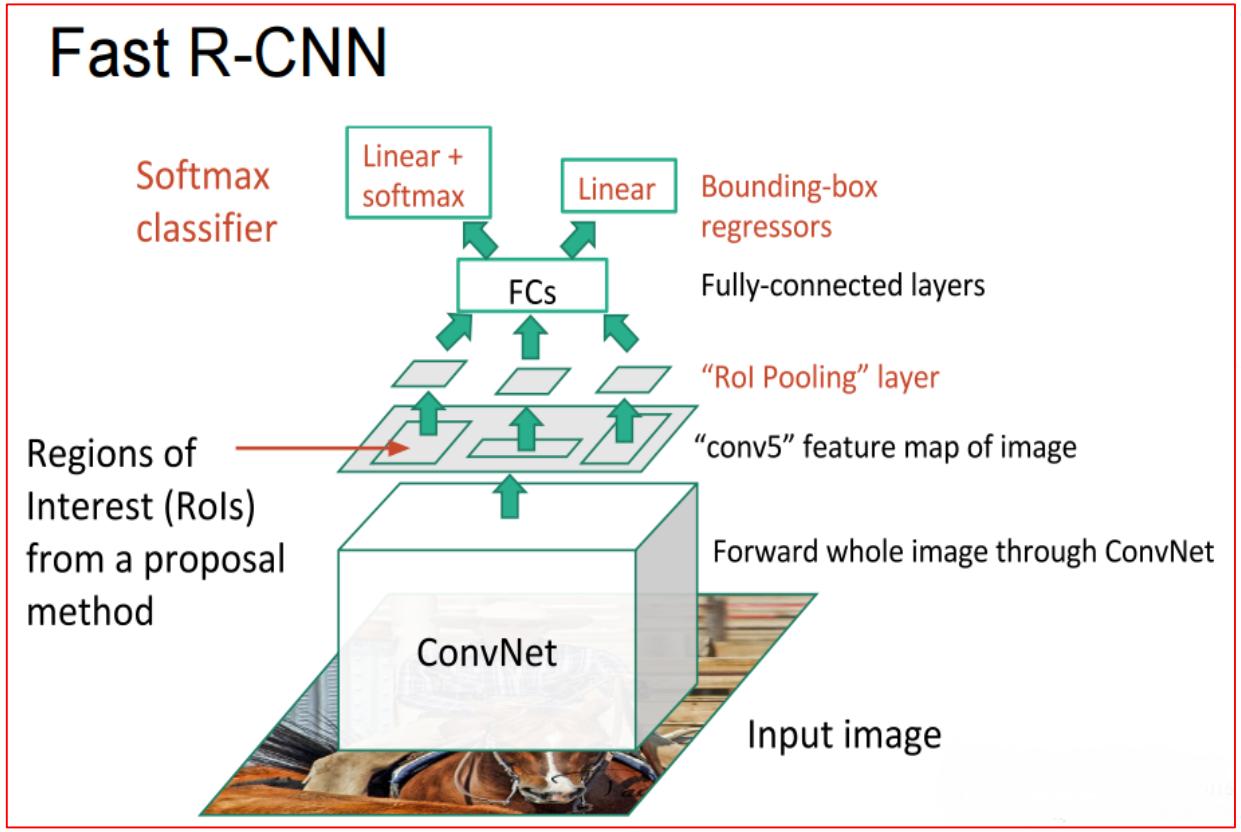
如图计算出的feature map也是任意大小的，现在经过SPP之后，就可以变成固定大小的输出了，以上图为例，一共可以输出（16+4+1）*256的特征。

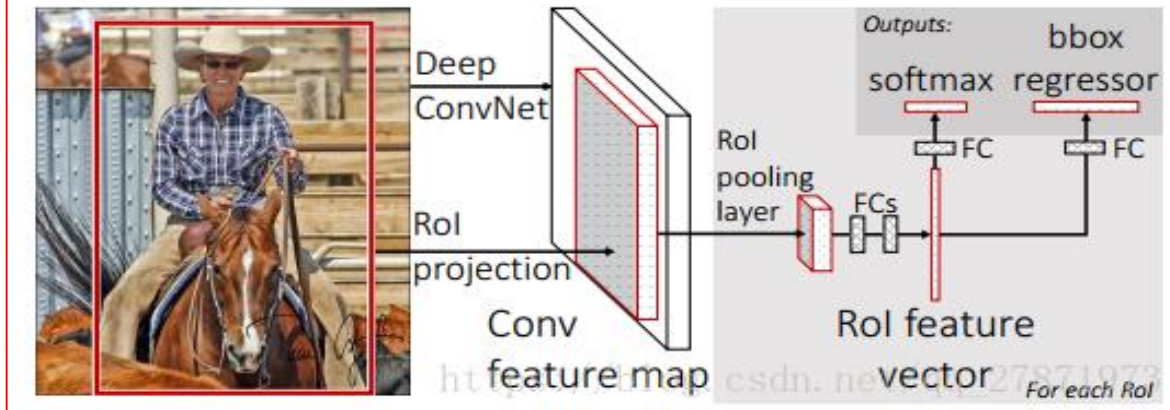


黑色图片代表卷积之后的特征图，接着我们以不同大小的块来提取特征，分别是4*4，2*2，1*1，将这三张网格放到下面这张特征图上，就可以得到16+4+1=21种不同的块(Spatial bins)，我们从这21个块中，每个块提取出一个特征，这样刚好就是我们要提取的21维特征向量。这种以不同的大小格子的组合方式来池化的过程就是空间金字塔池化（SPP）。比如，要进行空间金字塔最大池化，其实就是从这21个图片块中，分别计算每个块的最大值，从而得到一个输出单元，最终得到一个21维特征的输出。

R-CNN、Fast R-CNN、Faster R-CNN 网络结构讲解& 图像目标

Fast R-CNN





输入图片size不同导致得到的feature map尺寸也不同，不能直接接到一个全连接层进行分类，但是可以加入这个神奇的ROI Pooling层，对每个region都提取一个固定维度的特征表示，再通过正常的softmax进行类型识别。
之前RCNN的处理流程是先提proposal，然后CNN提取特征，之后用SVM分类器，最后再做bbox regression，而在Fast-RCNN中，作者巧妙的把bbox regression放进了神经网络内部，与region分类和并成为了一个multi-task模型

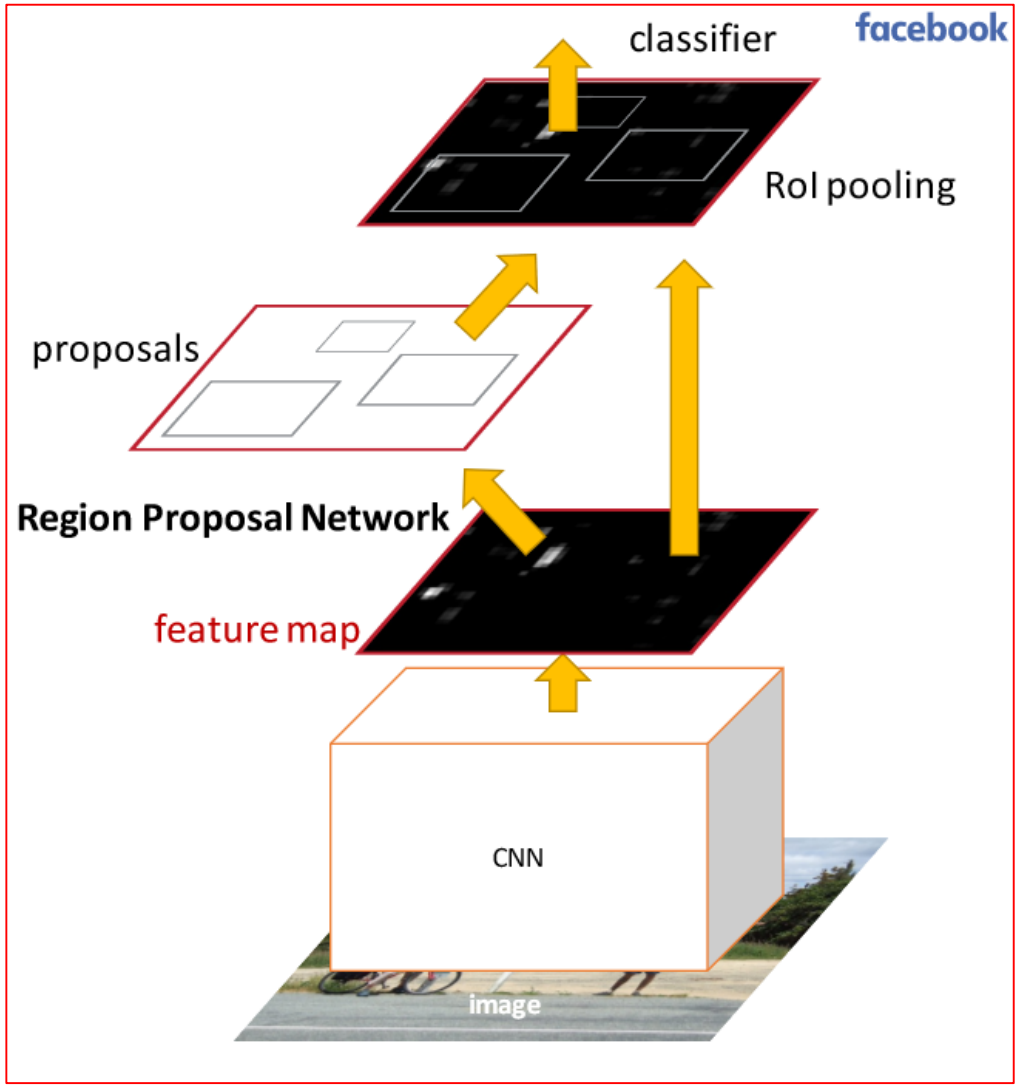
1. 在图像中确定约1000-2000个候选框 (使用选择性搜索)
2. 对整张图片输进CNN，得到feature map
3. 找到每个候选框在feature map上的映射patch，将此patch作为每个候选框的卷积特征输入到SPP layer和之后的层
4. 对候选框中提取出的特征，使用分类器判别是否属于一个特定类
5. 对于属于某一特征的候选框，用回归器进一步调整其位置

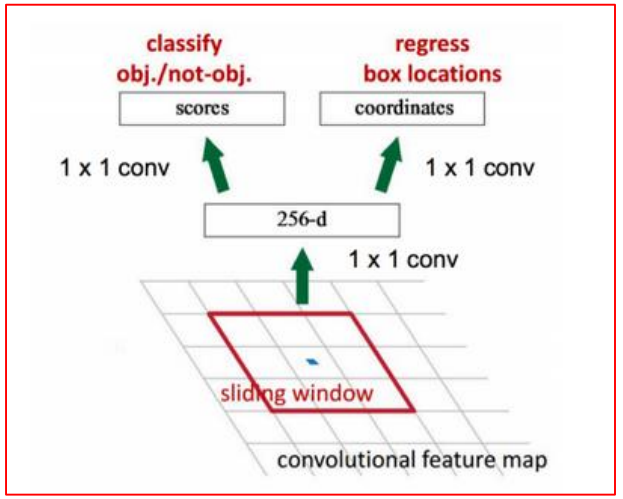
R-CNN、Fast R-CNN、Faster R-CNN 网络结构讲解& 图像目标

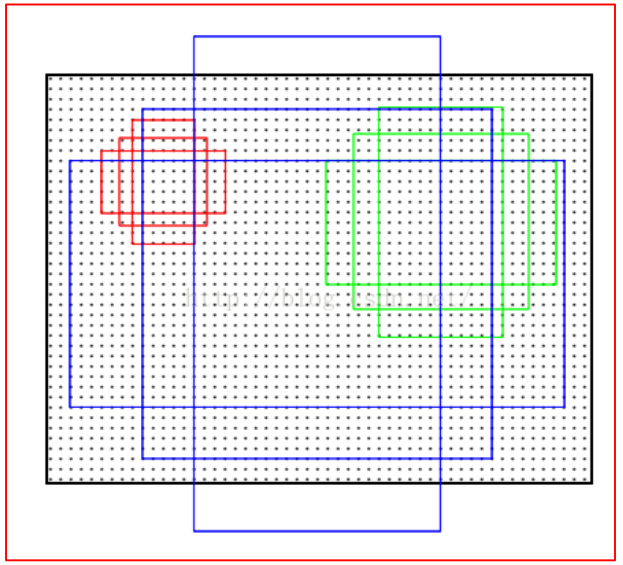
Fast R-CNN存在的问题：存在瓶颈：选择性搜索，找出所有的候选框，这个也非常耗时。那我们能不能找出一个更加高效的方法来求出这些候选框呢？解决：加入一个提取边缘的神经网络，也就说找到候选框的工作也交给神经网络来做了。做这样的任务的神经网络叫做Region Proposal Network(RPN)。

1. 对整张图片输进CNN，得到feature map
2. 卷积特征输入到RPN，得到候选框的特征信息
3. 对候选框中提取出的特征，使用分类器判别是否属于一个特定类
4. 对于属于某一特征的候选框，用回归器进一步调整其位置

Faster R-CNN







RPN

R-CNN、Fast R-CNN、Faster R-CNN 网络结构讲解& 图像目标

目标检测展望

R-CNN 2013.11 → **OverFeat** ICLR' 14 → MultiBox CVPR' 14 → SPP-Net ECCV' 14 → MR-CNN ICCV' 15 → DeepBox ICCV' 15 → AttentionNet ICCV' 15

Fast R-CNN ICCV' 15 → DeepProposal ICCV' 15 → **Faster R-CNN** NIPS' 15 → **OHEM** CVPR' 16 → **YOLO v1** CVPR' 16 → G-CNN CVPR' 16 → AZNet CVPR' 16

Inside-OutsideNet(ION) CVPR' 16 → HyperNet CVPR' 16 → CRAFT CVPR' 16 → MultiPathNet(MPN) BMVC' 16 → **SSD** ECCV' 16 → GBDNet ECCV' 16

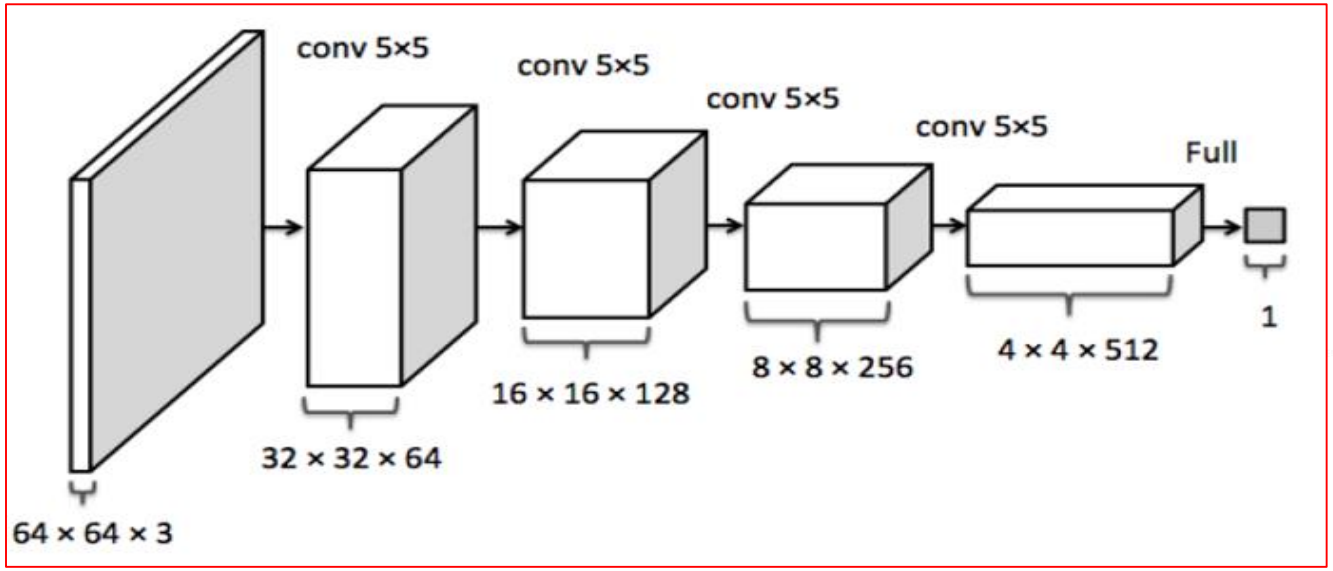
CPF ECCV' 16 → MS-CNN ECCV' 16 → **R-FCN** NIPS' 16 → PVANET NIPSW' 16 → DeepID-Net PAMI' 16 → NoC TPAMI' 16 → DSSD arXiv' 17 → TDM CVPR' 17 → **YOLO v2** CVPR' 17

Feature Pyramid Net(**FPN**) CVPR' 17 → RON CVPR' 17 → DCN ICCV' 17 → DeNet ICCV' 17 → CoupleNet ICCV' 17 → **RetinaNet** ICCV' 17 → DSOD ICCV' 17

Mask R-CNN ICCV' 17 → SMN ICCV' 17 → **YOLO v3** arXiv' 18 → SIN CVPR' 18 → STDN CVPR' 18 → **RefineDet** CVPR' 18 → MLKP CVPR' 18 → Relation-Net CVPR' 18

Cascade R-CNN CVPR' 18 → RFBNet ECCV' 18 → CornerNet ECCV' 18 → PFPNet ECCV' 18 → Pelee NIPS' 18 → HKRM NIPS' 18 → R-DAD AAAI' 19 → **M2Det** AAAI' 19 …

作业



```
#######作业#######补充下面的代码#######作业########
### https://keras.io/zh/ 参考网站
for i in range(0,d_n_layers):
    #1.添加的卷层 Conv2D (参数 filters=d_n_filters[i], kernel_size=(5,5), padding='same')
    #2.添加激活函数 Activation  (参数 'tanh')
    #3.添加池化层  MaxPooling (参数pool_size=(2,2), strides=(2,2))
```

02_2_dcgan.py 补充完代码

THANK YOU

我就是我不一样的两个火—施炎