深度学习与神经网络

1. 基本原理

> 问题：以图像识别为例，给定 N 个样本 $\{(x_1, y_1), \cdots, (x_N, y_N)\}$，其中$x_i$ 表示图像，y_i 表示图像的类别标签（猫？狗？），现在给定一张新图 x, 识别出图像的标签

一个前馈神经网络，描述一个条件概率函数$p(y|x, \theta)$, 其中 x 表述输入，

$$\theta^* = \operatorname*{argmax}_\theta S(\theta)$$
$$S(\theta) \equiv \frac{1}{n}\sum_{i=1}^{n} \log p(y_i|x_i, \theta)$$

- logistic regression

$$\begin{aligned} z &= \mathbf{x} \cdot \mathbf{w} + b \\ a &= \sigma(z) \\ \theta^* &= \operatorname*{argmin}_\theta \sum_i \|y_i - a_i\|^2 \end{aligned} \qquad \text{(非概率模型)}$$

- percepton

$$\begin{aligned} z &= \mathbf{x} \cdot \mathbf{w} + b \\ a &= \sigma(z) \\ \theta^* &= \operatorname*{argmin}_\theta \sum_i [-y_i \ln a_i - (1 - y_i) \ln(1 - a_i)] \end{aligned} \qquad \text{(概率模型)}$$

- neural network with 1-hidden layer (binary classification)

$$\begin{aligned} \mathbf{h} &= f(\mathbf{x}\mathbf{W_h} + \mathbf{b_h}) \\ z &= \mathbf{h} \cdot \mathbf{w_a} + b_a \\ a &= \sigma(z) \\ \theta^* &= \operatorname*{argmin}_\theta \sum_i [-y_i \ln a_i - (1 - y_i) \ln(1 - a_i)] \end{aligned}$$

- neural network with 1-hidden layer (multiple classification)

$$\begin{aligned} \mathbf{h} &= f(\mathbf{x}\mathbf{W_h} + \mathbf{b_h}) \\ \mathbf{z} &= \mathbf{h} \cdot \mathbf{W_a} + \mathbf{b_a}) \\ a_k &= \frac{e^{zk}}{\sum_j e^{zj}} \\ \theta^* &= \operatorname*{argmin}_\theta \sum_i \left[-\mathbf{y}_i \cdot \ln \mathbf{a}_i\right] \end{aligned}$$

2. SGD

怎样求解如下形式的最优化问题

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, S_\theta, \qquad S_\theta \equiv \frac{1}{N}\sum_{i=1}^{N} L_\theta(\mathbf{x}_i, \mathbf{y}_i)$$

梯度算法的思路：

$$\theta \leftarrow \theta - \eta \nabla_\theta S_\theta$$

但 S_θ 由很多项求和构成，计算成本很高，于是引入 Stochastic 思想：在每步迭代中引入近似：

$$S_\theta = \frac{1}{N}\sum_{i=1}^{N} L_\theta(\mathbf{x}_i, \mathbf{y}_i) \approx \frac{1}{M}\sum_{i=1}^{M} L_\theta(\mathbf{x}_i, \mathbf{y}_i)$$

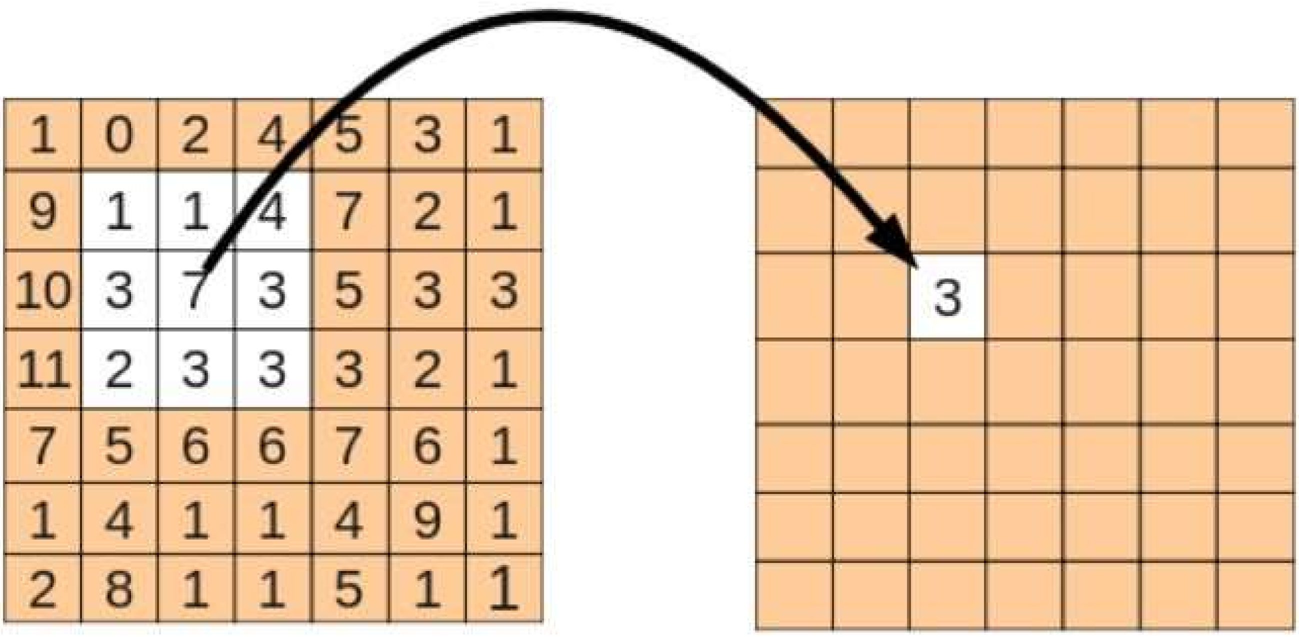
momentum, anealing learing rate and adaptive learning rate etc.

[文章推荐 (../papers/SGD_review.pdf)](../papers/SGD_review.pdf)

3. 卷积神经网络

卷积操作

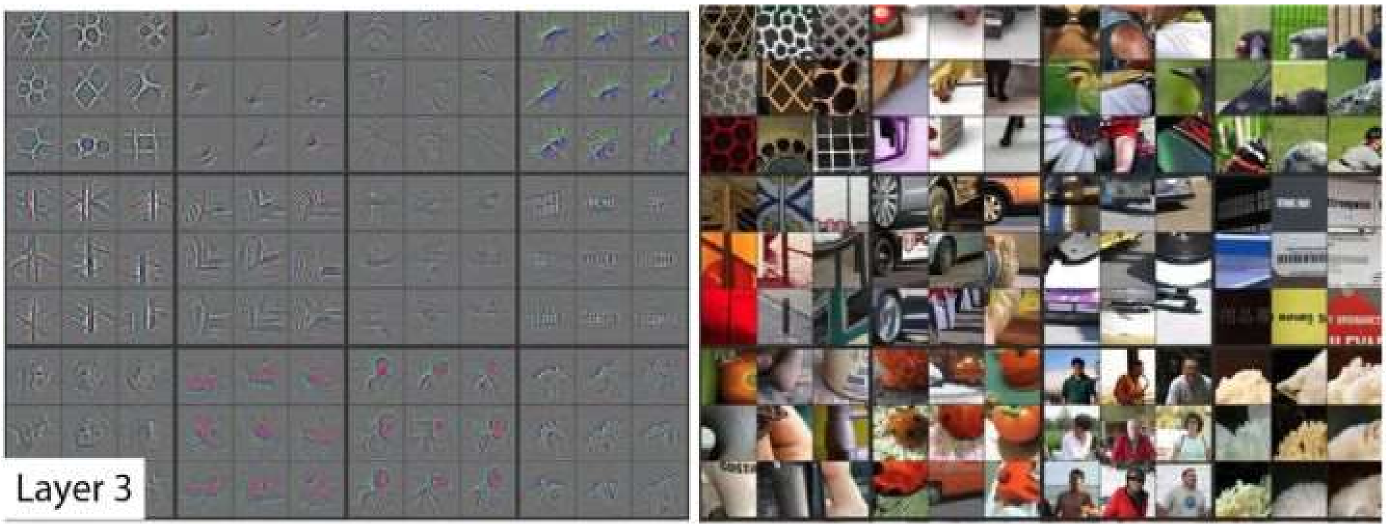
$$W = \frac{1}{9} * \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

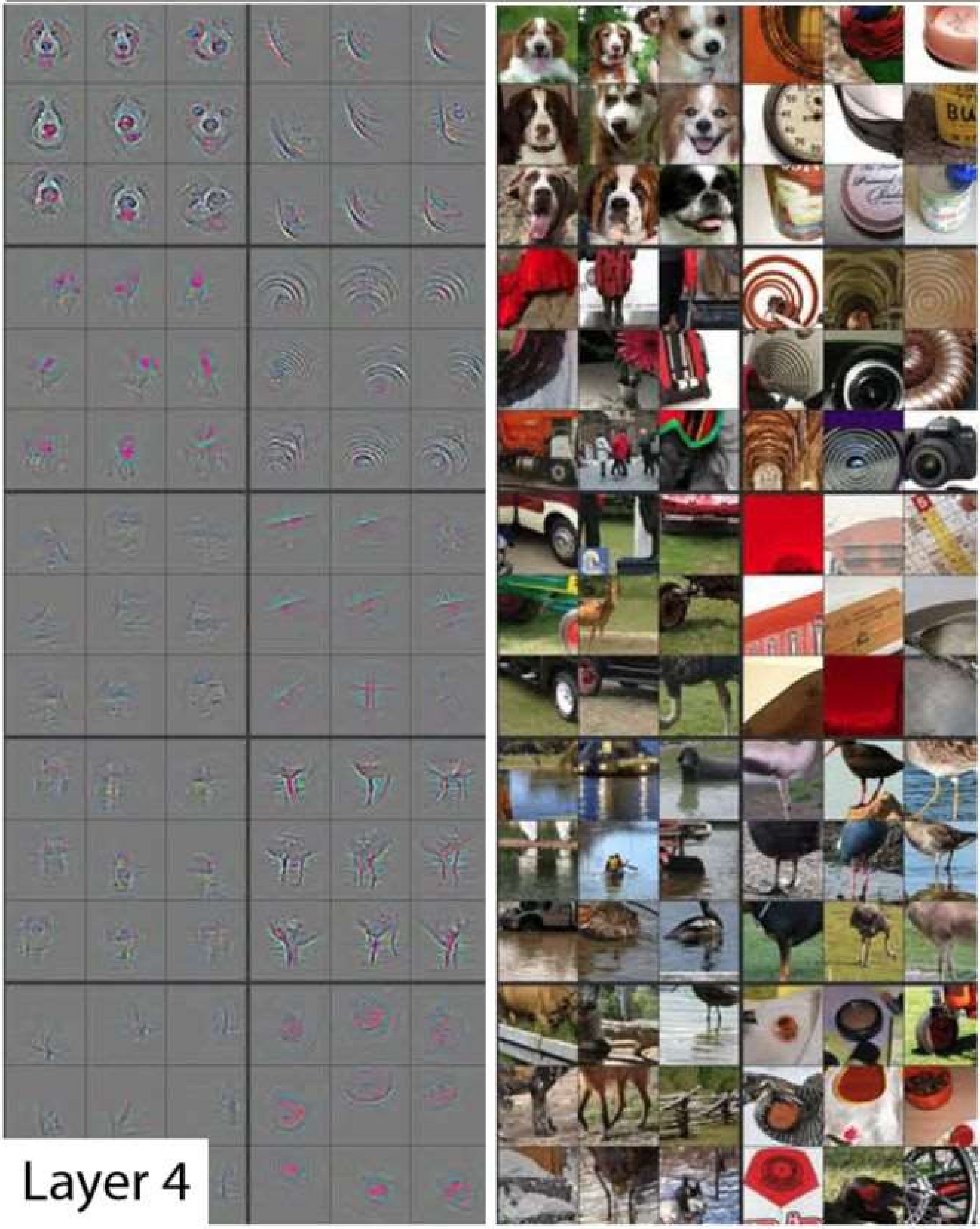


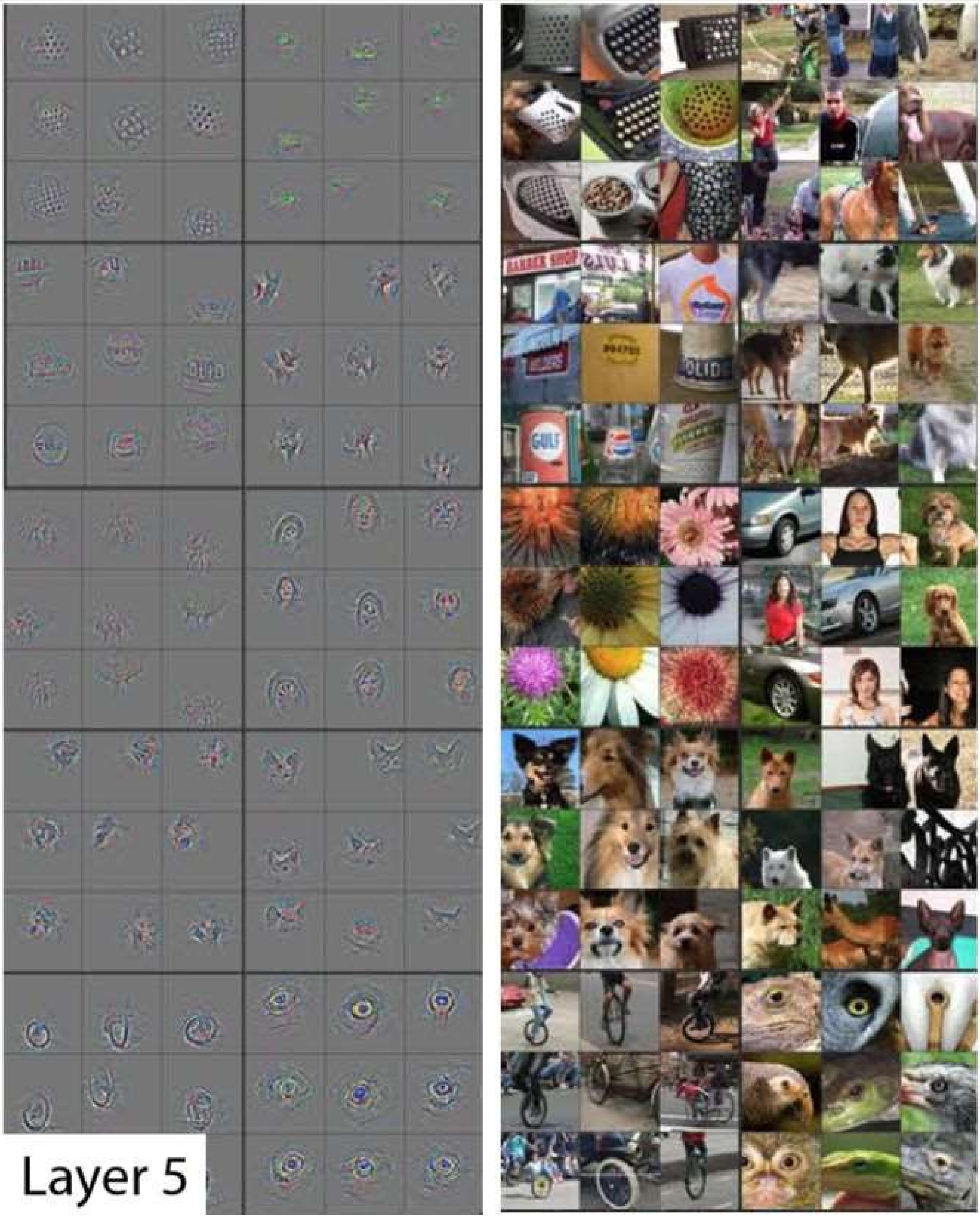
卷积作用


Layer 1


Layer 2


Layer 3


Layer 4



其他变种

transpose cnn(fractional cnn) / dilated cnn

[http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html (http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html)](http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html)

4. Back propagation

考虑如下过程：一个batch的samples （个数为M）输入一个前馈神经网络，并最终获得 cost function（“标量”函数）$\mathscr{C}$ 的值，

除输入层外，假设共有L层神经元，每层神经元的个数分别记为 $N_1, N_2, \ldots, N_L$；

对于batch 中的第m个 sample （$m = 1, 2, \ldots, M$）,其对应第l层的第i个神经元输出（activation）记为 $a_{mi}^{(l)}$（$l = 1, 2, \ldots, L; \quad i = 1, 2, \ldots, N_l$）

将第m 个sample 在第l 层的所有输出记为$\mathcal{A}_m^{(l)}$，即

$$\mathcal{A}_m^{(l)} \equiv \left\{a_{m1}^{(l)}, a_{m2}^{(l)}, \ldots, a_{mN_l}^{(l)}\right\}, \qquad \text{size}\left(\mathcal{A}_m^{(l)}\right) = N_l$$

将batch中所有sample 在第l 层的所有输出记为$\mathcal{A}^{(l)}$，即

$$\mathcal{A}^{(l)} \equiv \mathcal{A}_1^{(l)} \cup \mathcal{A}_2^{(l)} \cup \cdots \cup \mathcal{A}_M^{(l)}, \qquad \text{size}\left(\mathcal{A}^{(l)}\right) = M \times N_l$$

由于前馈神经网络中，每层输出值仅“直接”依赖于其上一层神经元的输出值，即，

$$a_{mi}^{(l)} = f_i^{(l)}\left(\mathcal{A}_m^{(l-1)}, \vec{\theta}_i^{(l)}\right), \qquad \text{（其中}m = 1, 2, \ldots, M\text{分别对应M个sample）}$$

其中$f_i^{(l)}$为，第l层第i个神经元的输出函数 (以$l-1$ 层神经元的输出作为其输入量)，$\vec{\theta}_i^{(l)}$ 是第l层第i个神经元的参数。

cost function 仅“直接”依赖于最后一层输出值，即 $\mathscr{C} = \mathscr{C}\left(\mathcal{A}^{(L)}\right)$，由于上述输出量的层层依赖关系，因此，总可以认为 $\mathscr{C} = \mathscr{C}\left(\mathcal{A}^{(l)}\right)$，$\forall l \in \{1, 2, \ldots, L\}$。记$l$层所有参数的集合为$\Theta^{(l)}$， 即
$\Theta^{(l)} \equiv \left\{\vec{\theta}_1^{(l)}, \vec{\theta}_2^{(l)}, \ldots, \vec{\theta}_{N_l}^{(l)}\right\}$；记神经网络中的所有参数为$\Theta$，即 $\Theta \equiv \Theta^{(1)} \cup \Theta^{(2)} \cup \cdots \cup \Theta^{(L)}$，则也可以认为 $\mathscr{C} = \mathscr{C}(\Theta)$

定义

$$\delta_{mi}^{(l)} \equiv \frac{\partial \mathscr{C}\left(\mathcal{A}^{(l)}\right)}{\partial a_{mi}^{(l)}}$$

由于 $\mathscr{C}$ 可以按batch中的各个sample拆分为：$\mathscr{C}\left(\mathcal{A}^{(l)}\right) = \sum_{m=1}^{M} \mathscr{C}_m\left(\mathcal{A}_m^{(l)}\right)$，于是

$$\delta_{mi}^{(l)} = \frac{\partial \mathscr{C}_m\left(\mathcal{A}_m^{(l)}\right)}{\partial a_{mi}^{(l)}}$$

我们有：

$$\delta_{mi}^{(L)} = \frac{\partial \mathscr{C}\left(\mathcal{A}^{(L)}\right)}{\partial a_{mi}^{(L)}} = \frac{\partial \mathscr{C}_m\left(\mathcal{A}_m^{(L)}\right)}{\partial a_{mi}^{(L)}}$$

$$\delta_{mi}^{(l-1)} = \frac{\partial \mathscr{C}\left(\mathcal{A}^{(l-1)}\right)}{\partial a_{mi}^{(l-1)}} = \sum_{n=1}^{M}\sum_{j=1}^{N_l} \frac{\partial \mathscr{C}\left(\mathcal{A}^{(l)}\right)}{\partial a_{nj}^{(l)}} \frac{\partial a_{nj}^{(l)}}{\partial a_{mi}^{(l-1)}} = \sum_{j=1}^{N_l} \delta_{mj}^{(l)} \frac{\partial f_j^{(l)}\left(\mathcal{A}_m^{(l-1)}, \vec{\theta}_j^{(l)}\right)}{\partial a_{mi}^{(l-1)}}$$

$$\frac{\partial \mathscr{C}(\Theta)}{\partial \vec{\theta}_i^{(l)}} = \sum_{m=1}^{M}\sum_{k=1}^{L}\sum_{j=1}^{N_k} \frac{\partial \mathscr{C}\left(\mathcal{A}^{(k)}\right)}{\partial a_{mj}^{(k)}} \frac{\partial a_{mj}^{(k)}}{\partial \vec{\theta}_i^{(l)}} = \sum_{m=1}^{M} \frac{\partial \mathscr{C}\left(\mathcal{A}^{(l)}\right)}{\partial a_{mi}^{(l)}} \frac{\partial a_{mi}^{(l)}}{\partial \vec{\theta}_i^{(l)}} = \sum_{m=1}^{M} \delta_{mi}^{(l)} \frac{\partial f_i^{(l)}\left(\mathcal{A}_m^{(l-1)}, \vec{\theta}_i^{(l)}\right)}{\partial \vec{\theta}_i^{(l)}}$$

max pool layer

设由$l-1$层向l层的运算是一个 max pool 操作：对于batch 中的第$\forall m$个 sample，第l层的第$\forall i$个神经元，在第$l-1$ 层对应的“采样池”记为 $\text{pool}\left(\mathcal{A}_m^{(l-1)}, i\right)$，举个具体例子，

假设l层的第7个神经元的输出值要求是$l-1$ 层的第2、3、5、8神经元输出值中的最大值，则 $\text{pool}\left(\mathcal{A}_m^{(l-1)}, 7\right) \equiv \left\{a_{m2}^{(l-1)}, a_{m3}^{(l-1)}, a_{m5}^{(l-1)}, a_{m8}^{(l-1)}\right\}$，$a_{m7}^{(l)} = \max\left(\left\{a_{m2}^{(l-1)}, a_{m3}^{(l-1)}, a_{m5}^{(l-1)}, a_{m8}^{(l-1)}\right\}\right) \equiv \max\text{pool}\left(\right.$

此时，第$l-1$层第$\forall i$个神经元的导数为：

$$\delta_{mi}^{(l-1)} = \sum_{j=1}^{N_l} \delta_{mj}^{(l)} \frac{\partial \max\text{pool}\left(\mathcal{A}_m^{(l-1)}, j\right)}{\partial a_{mi}^{(l-1)}} = \sum_{[j]} \delta_{mj}^{(l)}$$

其中$\sum_{[j]}$ 表示在N_l个“采样池”中仅对满足条件 $\max\text{pool}\left(\mathcal{A}_m^{(l-1)}, j\right) = a_{mi}^{(l)}$ 所对应的角标j 求和

（注意：max pool 层只需要传递导数，无参数需要拟合）

linear kernel layer

对于大多数前馈神经网络的中间层 $a_{mi}^{(l)} = f_i^{(l)}\left(\mathcal{A}_m^{(l-1)}, \vec{\theta}_i^{(l)}\right)$ 中的$f_i^{(l)}$ 函数一致，且取如下linear kernel 形式：

$$a_{mi}^{(l)} = F^{(l)}\left(\sum_{j=1}^{N_{l-1}} a_{mj}^{(l-1)} W_{ji}^{(l)} + b_i^{(l)}\right)$$

其中 $N_{l-1}+1$ 个参数 $\left\{W_{1i}^{(l)}, W_{2i}^{(l)}, \ldots, W_{N_{l-1}i}^{(l)}, b_i^{(l)}\right\} \equiv \vec{\theta}_i^{(l)}$，令$z_{mi}^{(l)} = \sum_j a_{mj}^{(l-1)} W_{ji}^{(l)} + b_i^{(l)}$，并将$l$层所有参数的集合$\Theta^{(l)}$ 在linear kernel 背景下改用$\left(\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\right)$ 标记，并将Θ 中 除l层以外的所有参数集记为Θ'，我们有：

$$\delta_{mi}^{(l-1)} = \sum_{j=1}^{N_l} \delta_{mj}^{(l)} \left(\frac{\partial F^{(l)}(z)}{\partial z}\right)_{z=z_{mj}^{(l)}} W_{ij}^{(l)}$$

$$\frac{\partial \mathscr{C}\left(\Theta', \mathbf{W}^{(l)}, \mathbf{b}^{(l)}\right)}{\partial W_{ji}^{(l)}} = \sum_{m=1}^{M} \delta_{mi}^{(l)} \left(\frac{\partial F^{(l)}(z)}{\partial z}\right)_{z=z_{mi}^{(l)}} a_{mj}^{(l-1)}$$

$$\frac{\partial \mathscr{C}\left(\Theta', \mathbf{W}^{(l)}, \mathbf{b}^{(l)}\right)}{\partial b_i^{(l)}} = \sum_{m=1}^{M} \delta_{mi}^{(l)} \left(\frac{\partial F^{(l)}(z)}{\partial z}\right)_{z=z_{mi}^{(l)}}$$

注意：$\langle ji \rangle$ 可遍历网络Graph中$l-1$和l层间的所有连线；（每一根连线，对应一个weight参数$W_{ji}^{(l)}$。对于全连通网络，i遍历所有l层的N_l个神经元，j 遍历$l-1$层的N_{l-1}个神经元；）

特别地：

一、 当$F^{(l)}$ 取 ReLU neuron形式时，则

$$\left(\frac{\partial F^{(l)}(z)}{\partial z}\right)_{z=z_{mi}^{(l)}} = \begin{cases} 1, & \text{if } z_{mi}^{(l)} \geq 0 \\ 0, & \text{if } z_{mi}^{(l)} < 0 \end{cases}$$

$$\frac{\partial \mathscr{C}(\Theta)}{\partial \omega_{\alpha s}^{(l)}} = \sum_{\langle ij \rangle_{\alpha s}} \frac{\partial \mathscr{C}(\Theta)}{\partial W_{ji}^{(l)}}$$

二、对于卷积神经网络，由于weights share，独立的参数比Graph中的连线（对应weights）和节点（对应bias）更少。设 $l-1$层向l层的运算是一个“卷积”操作，并且我们将独立的“卷积”weight 和 bias参数分别记为$\omega_{\mu\nu}^{(l)}$ 和 $\beta_\nu^{(l)}$ (其中ν 标记feature map，μ 标记该feature map上的各weight分量，例如：若l-th layer有10个feature map，每个feature map有25个独立weight，则独立的参数共有$10\times 25+10=260$个)，将这些独立的weight和bias参数集合记为，$\left(\Omega^{(l)}, B^{(l)}\right)$,则 cost function的依赖参数可以看做：$\mathscr{C}=\mathscr{C}\left(\Theta', \Omega^{(l)}, B^{(l)}\right)$，且我们有，

$$\delta_{mi}^{(l-1)}=\sum_{j=1}^{N_l}\delta_{mj}^{(l)}\left(\frac{\partial F^{(l)}(z)}{\partial z}\right)_{z=z_{mj}^{(l)}}W_{ij}^{(l)}$$

$$\frac{\partial\mathscr{C}\left(\Theta', \Omega^{(l)}, B^{(l)}\right)}{\partial\omega_{\mu\nu}^{(l)}}=\sum_{ij}\frac{\partial\mathscr{C}\left(\Theta', \mathbf{W}^{(l)}, \mathbf{b}^{(l)}\right)}{\partial W_{ji}^{(l)}}\frac{\partial W_{ji}^{(l)}}{\partial\omega_{\mu\nu}^{(l)}}=\sum_{\langle ji\rangle_{\mu\nu}}\sum_{m=1}^{M}\delta_{mi}^{(l)}\left(\frac{\partial F^{(l)}(z)}{\partial z}\right)_{z=z_{mi}^{(l)}}a_{mj}^{(l-1)}$$

$$\frac{\partial\mathscr{C}\left(\Theta', \Omega^{(l)}, B^{(l)}\right)}{\partial\beta_{\nu}^{(l)}}=\sum_{i}\frac{\partial\mathscr{C}\left(\Theta', \mathbf{W}^{(l)}, \mathbf{b}^{(l)}\right)}{\partial b_{i}^{(l)}}\frac{\partial b_{i}^{(l)}}{\partial\beta_{\nu}^{(l)}}=\sum_{\langle i\rangle_{\nu}}\sum_{m=1}^{M}\delta_{mi}^{(l)}\left(\frac{\partial F^{(l)}(z)}{\partial z}\right)_{z=z_{mi}^{(l)}}$$

其中= $\sum\limits_{\langle ji\rangle_{\mu\nu}}$ 表示对Graph中$l-1$层与l层之间所有“卷积”连线$\langle ji\rangle$求和（连线对应的卷积weight 值要求是 $\omega_{\mu\nu}^{(l)}$）；
$\sum\limits_{\langle i\rangle_{\nu}}$ 表示对第l层上第μ个feature map上的所有节点求和

我们再以tensoflow中的卷积函数来做比较：

output = tf.nn.conv2d(**input**, **filter**, **strides**, **padding**, use_cudnn_on_gpu=None, **data_format**=None, name=None)

- **input**： 对应输入层$a_{mj}^{(l-1)}$，对于输入层的节点index j 可以用其在图像中的行列位置及所属channel来表征（ $l-1$层是原始图像输入时，有rgb 三个 channel；$l-1$层是中间层时，每个channel对应一个feature map）；因此 input 可以看做是一个由 $a_{mj}^{(l-1)}$ reshape 而成的4阶tensor：

shape(input) = [batch_size, input_height, input_width, input_channels(depth)]

（注意：**data_format** 字符串指定 input 的格式，默认为"NHWC", 表示input的脚标依次表示batch number，height，width，channels；指定为"NWHC" 时，则input的脚标依次表示batch number，width，height，channels；此时：

shape(input) = [batch_size, input_width, input_height, input_channels(depth)]）

- **filter**：对应独立参数 $\omega_{\mu\nu}^{(l)}$，ν 表示输出到哪个feature map（又称为output channel）；在卷积操作中，“输入图像”被划分为各个小的patch，这些patch上每个位置对应一个weight参数，因此μ 可以用patch上对应的行列位置及所在的channel来表征；因此，filter 可以看作是一个由 $\omega_{\mu\nu}^{(l)}$ reshape 而成的四阶tensor：

shape(filter) = [filter_height, filter_width, input_channels(depth), output_channels]

由于一般图像在xy方向上应该同等对待，因此一般取 filter_height = filter_width

- **strides**：描述怎样在“输入图像”上选取参与卷积的各个patch,即描述patch平移的步长:

strides = [batch_step, height_step, width_step, channel_step]

由于我们需要遍历所有batch和所有channel，所以一般**选定 batch_step=1，channel_step=1**；另外由于一般图像在xy方向上应该同等对待，因此一般取 height_step = width_step

- **padding**：同样描述怎样在“输入图像”上选取参与卷积的各个patch，当padding 取“SAME”字符串时，将“输入图像”的四周padding 0 元素（延拓图像的高宽），以确保patch 在xy方向上可平移的步数，与原图像的pixel数一致，即 output_height = input_height, output_width = input_width；padding 取“VALID”字符串时，不用在“输入图像”的四周padding 0 元素，此时

$$\text{output\_width} = \frac{\text{input\_width} - \text{filter\_width}}{\text{width\_step}} + 1$$
$$\text{output\_height} = \frac{\text{input\_height} - \text{filter\_height}}{\text{height\_step}} + 1$$

结合上面的公式，可以认为**strides 和 padding 参数是用来描述 $W_{ji}^{(l)}$ 和 $\omega_{\mu\nu}^{(l)}$ 之间对应关系的**

- **output**： 对应输出层$a_{mi}^{(l)}$，和input参数类似，output可以看做是一个由 $a_{mi}^{(l)}$ reshape 而成的4阶tensor：

shape(output) = [batch_size, output_height, output_width, output_channels(depth)]