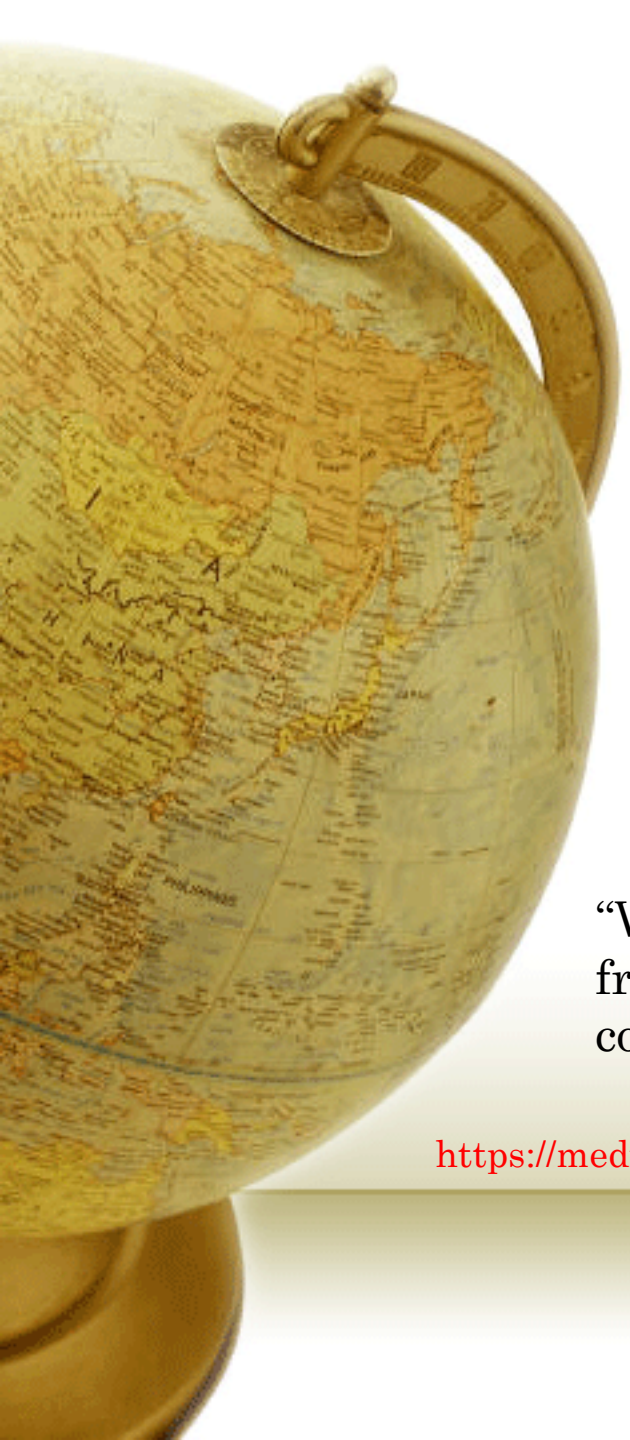
Νευρο-Ασαφής Υπολογιστική
Neuro-Fuzzy Computing

Διδάσκων –
Δημήτριος Κατσαρός

@ Τμ. ΗΜΜΥ
Πανεπιστήμιο Θεσσαλίας

Διάλεξη 13η



BackProp for CNNs: Do I need to understand it?

"Why do we have to write the backward pass when frameworks in the real world, such as TensorFlow, compute them for you automatically?"

https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b

Backpropagation for Convolutional Neural Networks

Notation

1. l is the l-th layer where l=1 is the first layer and l=L is the last layer
2. Input x is of dimension $H \times W$ and has i by j as the iterators
3. Filter or kernel w is of dimension $k_1 \times k_2$ has m by n as the iterators
4. $w^l_{m,n}$ is the weight matrix connecting neurons of layer l with neurons of layer $l-1$
5. b^l is the bias unit at layer l
6. $x^l_{i,j}$ is the convolved input vector at layer l plus the bias:

$$x^l_{i,j} = \sum_m \sum_n w^l_{m,n} o^{l-1}_{i+m,j+n} + b^l$$

7. $o^l_{i,j}$ is the output vector at layer l given by:

$$o^l_{i,j} = f(\mathrm{x}^l_{i,j})$$

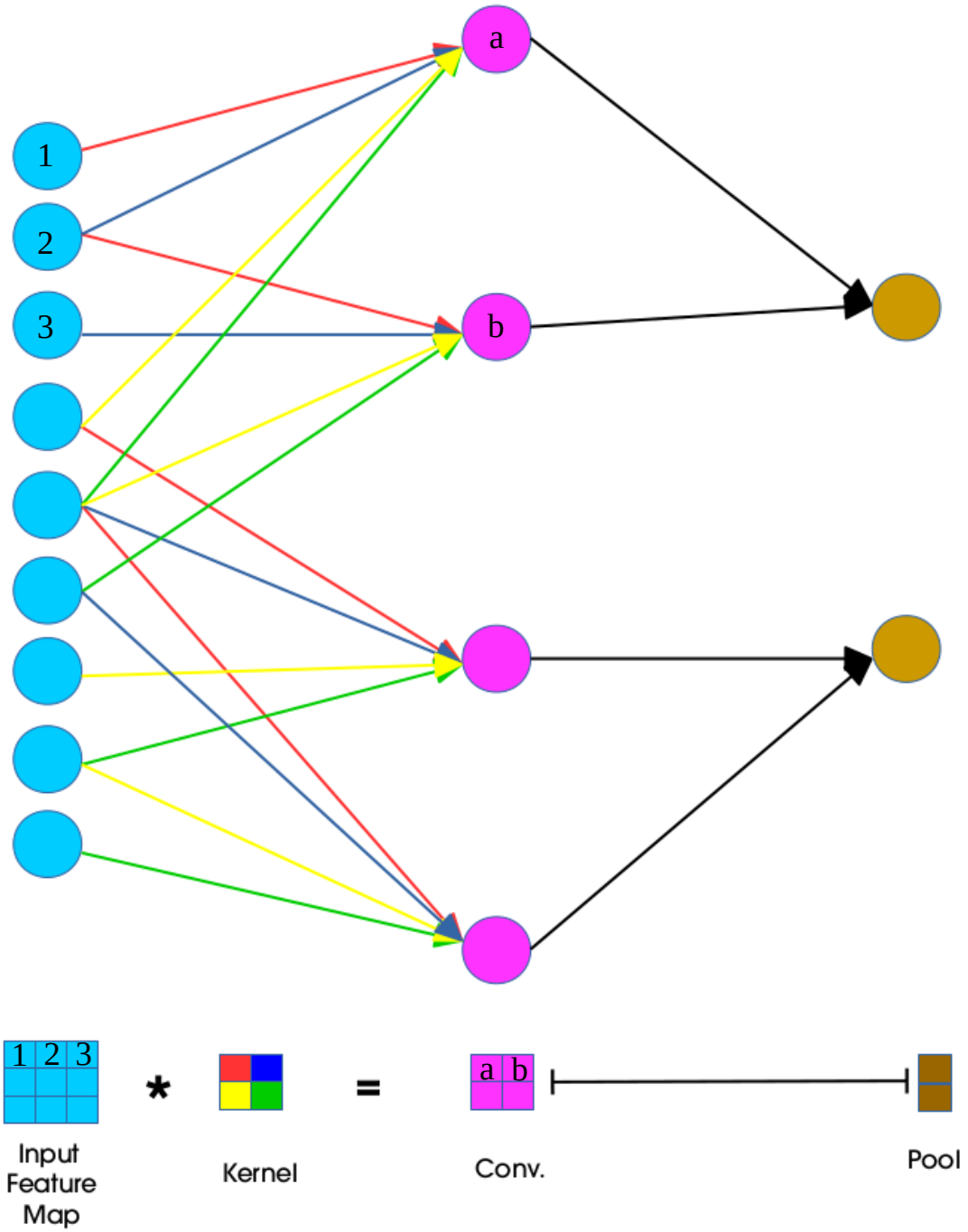
8. $f(\ldots)$ is the activation function. Application of the activation layer to the convolved input vector at layer l is given by $f(\mathrm{x}^l_{i,j})$

Forward propagation

- To perform a convolution operation, the kernel is flipped 180^{o} and slid across the input feature map in equal and finite strides
- At each location, the product between each element of the kernel and the input feature map element it overlaps is computed and the results summed up to obtain the output at that current location

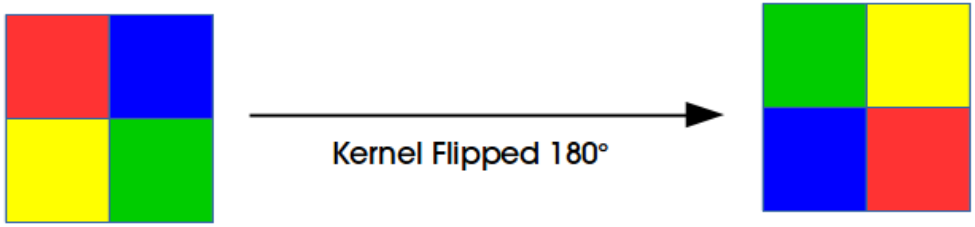
- This procedure is repeated using different kernels to form as many output feature maps as desired
- The concept of weight sharing is used as demonstrated in the next slide's diagram:

Weight Sharing in CNNs



Forward propagation

- Units in convolutional layer illustrated have receptive fields of size 4 in the input feature map and are thus only connected to 4 adjacent neurons in the input layer
- This is the idea of **sparse connectivity** in CNNs where there exists local connectivity pattern between neurons in adjacent layers
- The color codes of the weights joining the input layer to the convolutional layer show how the kernel weights are distributed (shared) amongst neurons in the adjacent layers. Weights of the same color are constrained to be identical
- The convolution process here is usually expressed as a cross-correlation but with a flipped kernel. In the diagram below we illustrate a kernel that has been flipped both horizontally and vertically:



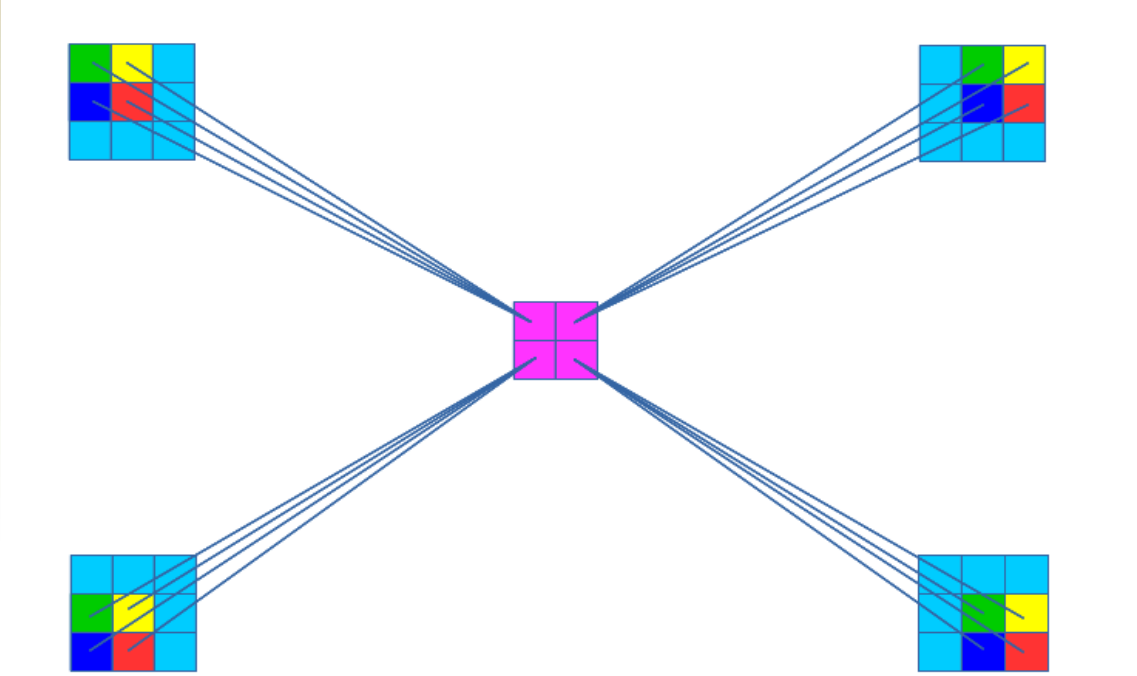
Forward propagation

- The convolution equation of the input at layer l is given by:

$$x_{i,j}^{l} = \mathrm{rot}_{180^o}\{w_{m,n}^{l}\} \circledast o_{i,j}^{l-1} + b_{i,j}^{l} = \sum_{m} \sum_{n} w_{m,n}^{l} o_{i+m,j+n}^{l-1} + b_{i,j}^{l}$$

$$o_{i,j}^{l} = f(x_{i,j}^{l})$$

- This is illustrated below:



Error

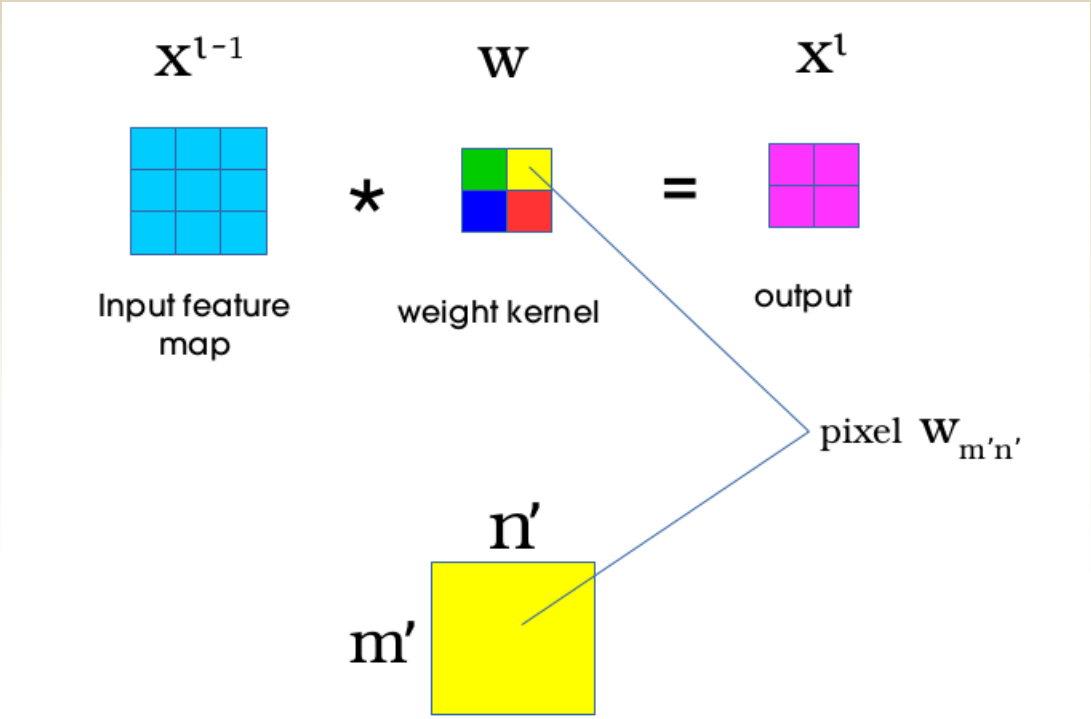
- For a total of P predictions, the predicted network outputs y_p and their corresponding targeted values t_p, the mean squared error is given by:

$$E = \frac{1}{2} \sum_{p} (t_p - y_p)^2$$

- Learning will be achieved by adjusting the weights such that y_p is as close as possible or equals to corresponding t_p
- In the classical backpropagation algorithm, the weights are changed according to the gradient descent direction of an error surface E

Backpropagation

- For backpropagation there are two updates performed, for the weights and the deltas
- Lets begin with the weight update.
- We are looking to compute $\partial E/\partial w^{l}_{m',n'}$ which can be interpreted as the measurement of how the change in a single pixel $w_{m',n'}$ in the weight kernel affects the loss function E



Backpropagation

- During forward propagation, the convolution operation ensures that the yellow pixel $w_{m',n'}$ in the weight kernel makes a contribution in all the products (between each element of the weight kernel and the input feature map element it overlaps)
- This means that pixel $w_{m',n'}$ will eventually affect all the elements in the output feature map
- Convolution (no padding, stride=1) between the input feature map of dimension $H \times W$ and the weight kernel of dimension $k_1 \times k_2$ produces an output feature map of size $(H-k_1+1)$ by $(W-k_2+1)$. The gradient component for the individual weights can be obtained by applying the chain rule in the following way:

$$\frac{\partial E}{\partial w^l_{m',n'}} = \sum_{i=0}^{H-k_1} \sum_{j=0}^{W-k_2} \frac{\partial E}{\partial x^l_{i,j}} \frac{\partial x^l_{i,j}}{\partial w^l_{m',n'}}$$

$$= \sum_{i=0}^{H-k_1} \sum_{j=0}^{W-k_2} \delta^l_{i,j} \frac{\partial x^l_{i,j}}{\partial w^l_{m',n'}} \qquad (0)$$

Backpropagation

- In Eq. 0, $x^l_{i,j}$ is equivalent to $\sum_m \sum_n w^l_{m,n} o^{l-1}_{i+m,j+n} + b^l$

  and expanding this part of the equation gives:

$$\frac{\partial x^l_{i,j}}{\partial w^l_{m',n'}} = \frac{\partial}{\partial w^l_{m',n'}} \left(\sum_m \sum_n w^l_{m,n} o^{l-1}_{i+m,j+n} + b^l \right) \tag{1}$$

- Further expanding the summations in Eq. 1, and taking the partial derivatives for all the components results in zero values for all except the components where $m=m'$ and $n=n'$ in $w^l{}_{m,n}\, o^{l-1}{}_{i+m,j+n}$ as follows:

$$\begin{aligned} \frac{\partial x^l_{i,j}}{\partial w^l_{m',n'}} &= \frac{\partial}{\partial w^l_{m',n'}} \left(w^l_{0,0} o^{l-1}_{i+0,j+0} + \cdots + w^l_{m',n'} o^{l-1}_{i+m',j+n'} + \cdots + b^l \right) \\ &= \frac{\partial}{\partial w^l_{m',n'}} \left(w^l_{m',n'} o^{l-1}_{i+m',j+n'} \right) \\ &= o^{l-1}_{i+m',j+n'} \end{aligned} \tag{2}$$

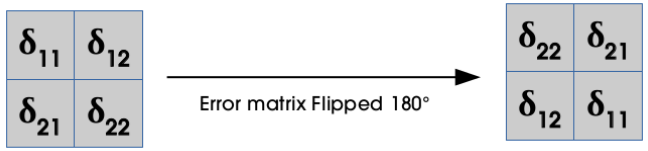
Backpropagation

- Substituting Eq. 2 in Eq. 0 gives us the following results:

$$\frac{\partial E}{\partial w^l_{m',n'}} = \sum_{i=0}^{H-k_1} \sum_{j=0}^{W-k_2} \delta^l_{i,j} o^{l-1}_{i+m',j+n'} \tag{3}$$

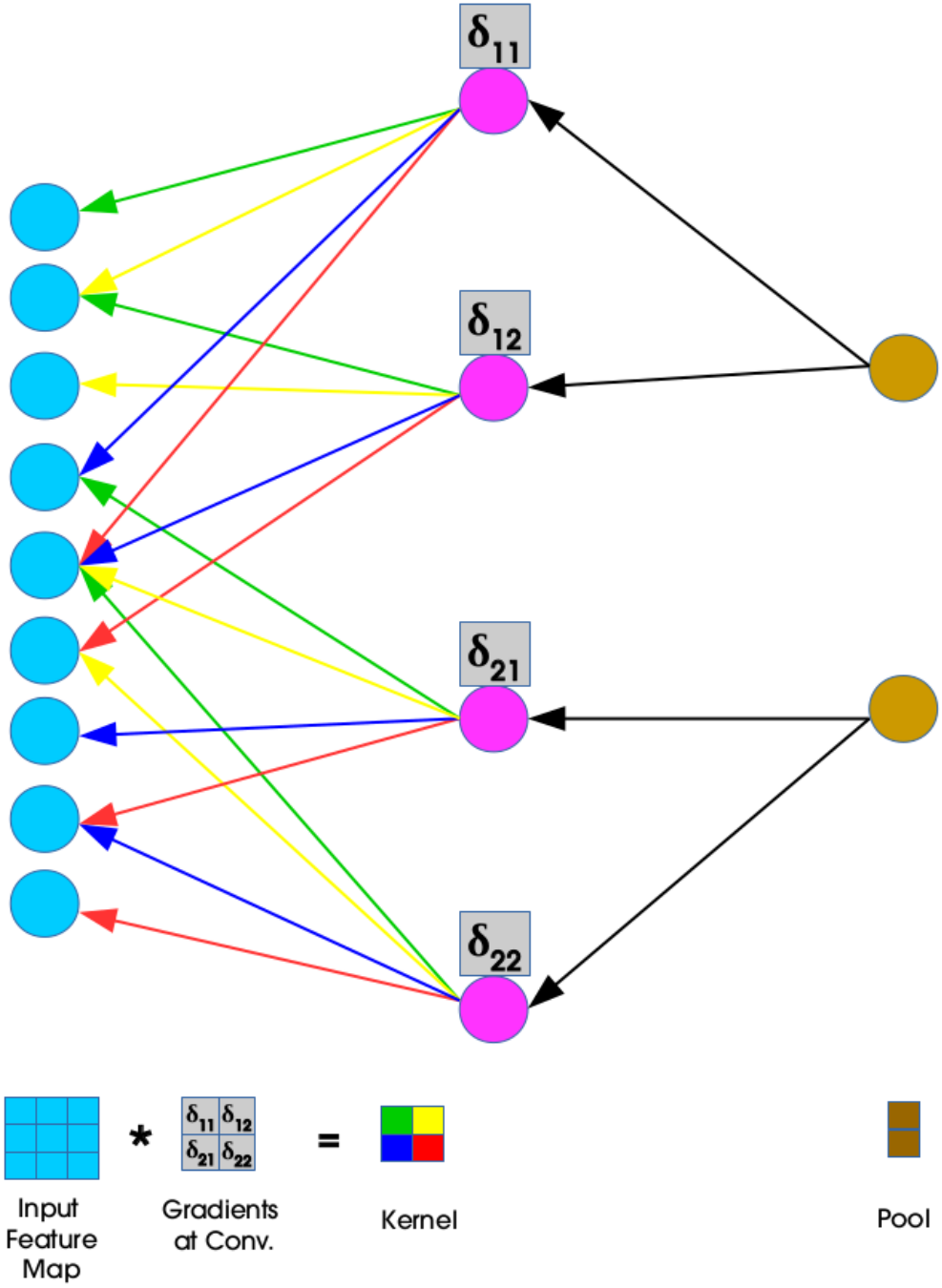
$$= \text{rot}_{180^\circ} \left\{ \delta^l_{i,j} \right\} * o^{l-1}_{m',n'} \tag{4}$$

- The dual summation in Eq. 3 is as a result of weight sharing in the network (same weight kernel is slid over all of the input feature map during convolution). The summations represent a collection of all the gradients $\delta^l_{i,j}$ coming from all the outputs in layer l
- Obtaining gradients w.r.t. the filter maps, we have a cross-correlation which is transformed to a convolution by "flipping" the delta matrix $\delta^l_{i,j}$ (horizontally and vertically) the same way we flipped the filters during the forward propagation. An illustration of the flipped delta matrix is shown:



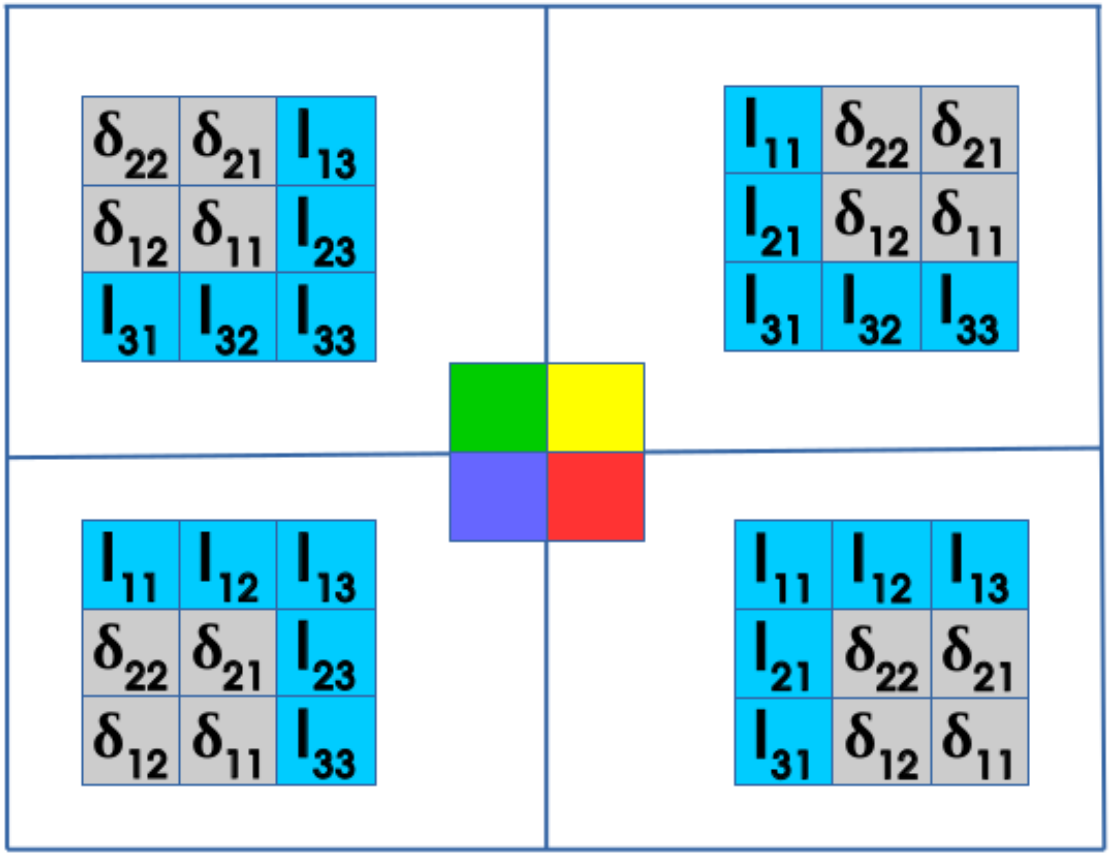
Backpropagation

- The diagram shows gradients (δ_{11},δ_{12},δ_{21},δ_{22}) generated during backpropagation



Backpropagation

- The convolution operation used to obtain the new set of weights as is shown below:

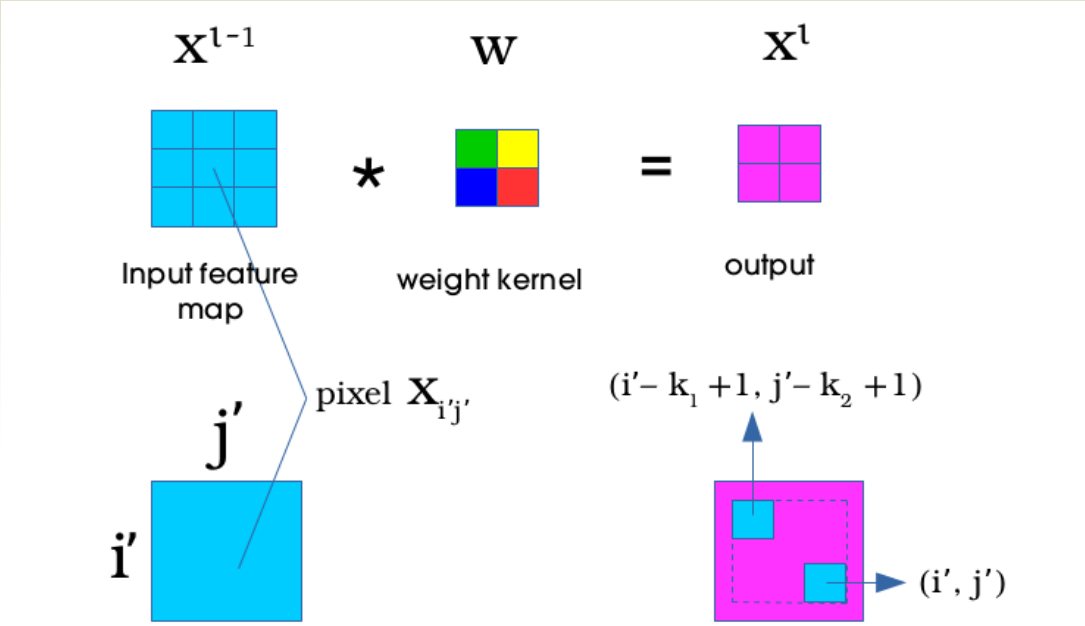


Backpropagation

- During the reconstruction process, the deltas (δ_{11},δ_{12},δ_{21},δ_{22}) are used. These deltas are provided by an equation of the form:

$$\delta_{i,j}^{l} = \frac{\partial E}{\partial x_{i,j}^{l}} \qquad (5)$$

- At this point we are looking to compute $\partial E/\partial x^{l}_{i',j'}$ which can be interpreted as the measurement of how the change in a single pixel $x_{i',j'}$ in the input feature map affects the loss function E



Backpropagation

- From the previous diagram, we can see that region in the output affected by pixel $x_{i',j'}$ from the input is the region in the output bounded by the dashed lines where the top left corner pixel is given by $(i'-k_1+1, j'-k_2+1)$ and the bottom right corner pixel is given by (i',j')
- Using chain rule and introducing sums gives us the following equation:

$$\frac{\partial E}{\partial x_{i',j'}^{l}} = \sum_{i,j \in Q} \frac{\partial E}{\partial x_{Q}^{l+1}} \frac{\partial x_{Q}^{l+1}}{\partial x_{i',j'}^{l}}$$

$$= \sum_{i,j \in Q} \delta_{Q}^{l+1} \frac{\partial x_{Q}^{l+1}}{\partial x_{i',j'}^{l}} \qquad (6)$$

where Q in the summation above represents the output region bounded by dashed lines and is composed of pixels in the output that are affected by the single pixel $x_{i',j'}$ in the input feature map

- A more formal way of representing Eq. 6 is the following:

Backpropagation

$$\frac{\partial E}{\partial x^{l}_{i',j'}} = \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} \frac{\partial E}{\partial x^{l+1}_{i'-m,j'-n}} \frac{\partial x^{l+1}_{i'-m,j'-n}}{\partial x^{l}_{i',j'}}$$

$$= \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} \delta^{l+1}_{i'-m,j'-n} \frac{\partial x^{l+1}_{i'-m,j'-n}}{\partial x^{l}_{i',j'}} \qquad (7)$$

- In the region Q, the height ranges from $i'-0$ to $i'-(k_1-1)$ and width $j'-0$ to $j'-(k_2-1)$
- These two can simply be represented by $i'-m$ and $j'-n$ in the summation, since the iterators m and n exist in the following similar ranges from $0 \le m \le k_1-1$ and $0 \le n \le k_2-1$

Backpropagation

- In Eq. 7, $x^{l+1}{}_{i'-m,j'-n}$ is equivalent to $\sum_{m'}\sum_{n'} w^{l+1}_{m',n'} o^{l}_{i'-m+m',j'-n+n'} + b^{l+1}$ and expanding this part of the equation gives:

$$\frac{\partial x^{l+1}_{i'-m,j'-n}}{\partial x^{l}_{i',j'}} = \frac{\partial}{\partial x^{l}_{i',j'}}\left(\sum_{m'}\sum_{n'} w^{l+1}_{m',n'} o^{l}_{i'-m+m',j'-n+n'} + b^{l+1}\right)$$

$$= \frac{\partial}{\partial x^{l}_{i',j'}}\left(\sum_{m'}\sum_{n'} w^{l+1}_{m',n'} f\left(x^{l}_{i'-m+m',j'-n+n'}\right) + b^{l+1}\right) \qquad (8)$$

- Further expanding the summation in Eq. 7 and taking the partial derivatives for all the components results in zero values for all except the components where $m'=m$ and $n'=n$ so that $f(x^{l}{}_{i'-m+m',j'-n+n'})$ becomes $f(x^{l}{}_{i',j'})$ and $w^{l+1}{}_{m',n'}$ becomes $w^{l+1}{}_{m,n}$ in the relevant part of the expanded summation as follows:

Backpropagation

$$\begin{aligned}
\frac{\partial x^{l+1}_{i'-m,j'-n}}{\partial x^{l}_{i',j'}} &= \frac{\partial}{\partial x^{l}_{i',j'}}\left(w^{l+1}_{m',n'} f\left(x^{l}_{0-m+m',0-n+n'}\right) + \cdots + w^{l+1}_{m,n} f\left(x^{l}_{i',j'}\right) + \cdots + b^{l+1}\right) \\
&= \frac{\partial}{\partial x^{l}_{i',j'}}\left(w^{l+1}_{m,n} f\left(x^{l}_{i',j'}\right)\right) \\
&= w^{l+1}_{m,n} \frac{\partial}{\partial x^{l}_{i',j'}}\left(f\left(x^{l}_{i',j'}\right)\right) \\
&= w^{l+1}_{m,n} f'\left(x^{l}_{i',j'}\right) \qquad (9)
\end{aligned}$$

- Substituting Eq. 9 in Eq. 7 gives us the following results:

$$\frac{\partial E}{\partial x^{l}_{i',j'}} = \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} \delta^{l+1}_{i'-m,j'-n} w^{l+1}_{m,n} f'\left(x^{l}_{i',j'}\right) \qquad (10)$$

Backpropagation

- For backpropagation, we make use of the flipped kernel and as a result we will now have a convolution that is expressed as a cross-correlation with a flipped kernel:

$$\frac{\partial E}{\partial x^l_{i',j'}} = \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} \delta^{l+1}_{i'-m,j'-n} w^{l+1}_{m,n} f'\left(x^l_{i',j'}\right)$$

$$= \mathrm{rot}_{180^\circ}\left\{\sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} \delta^{l+1}_{i'+m,j'+n} w^{l+1}_{m,n}\right\} f'\left(x^l_{i',j'}\right) \tag{11}$$

$$= \delta^{l+1}_{i',j'} * \mathrm{rot}_{180^\circ}\left\{w^{l+1}_{m,n}\right\} f'\left(x^l_{i',j'}\right) \tag{12}$$

Backpropagation for the pooling layer

- The function of the pooling layer is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. No learning takes place on the pooling layers

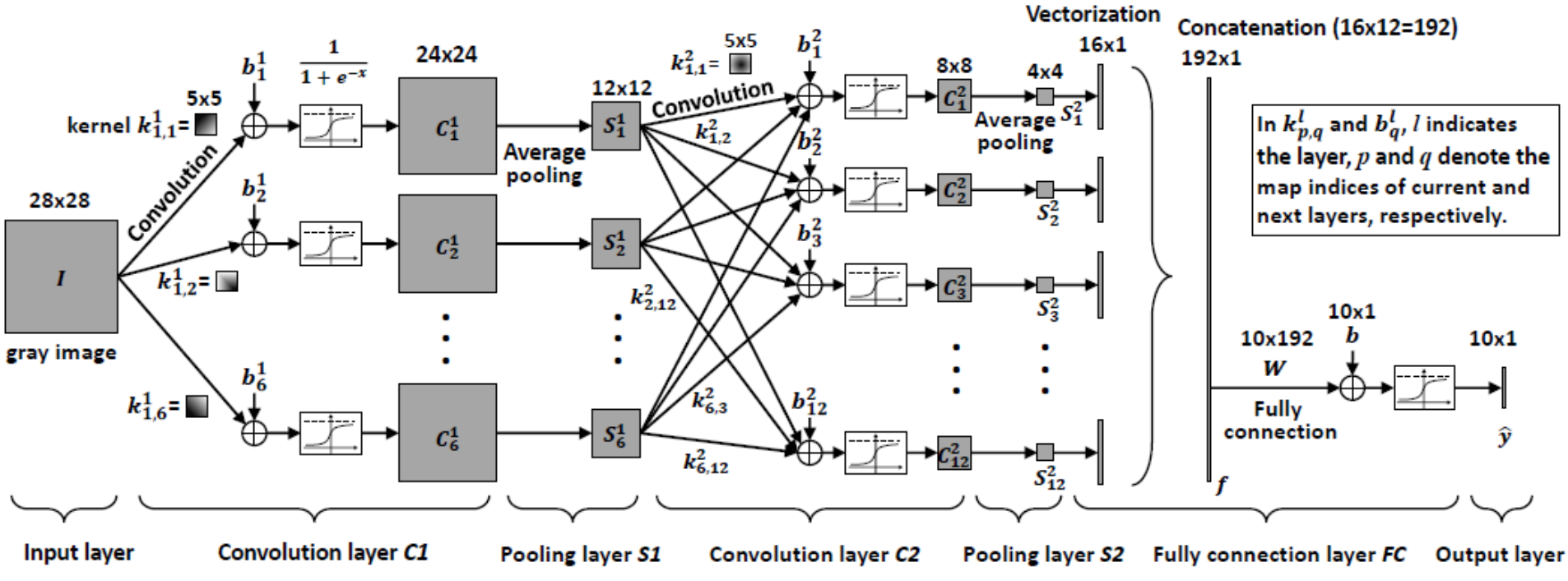
- Pooling units are obtained using functions like max-pooling, average pooling and even L2-norm pooling. At the pooling layer, forward propagation results in an N×N pooling block being reduced to a single value - value of the "winning unit"
- Backpropagation of the pooling layer then computes the error which is acquired by this single value "winning unit"

Backpropagation for the pooling layer

- To keep track of the "winning unit" its index noted during the forward pass and used for gradient routing during backpropagation.
- Gradient routing is done in the following ways:
  - Max-pooling - the error is just assigned to where it comes from - the "winning unit" because other units in the previous layer's pooling blocks did not contribute to it hence all the other assigned values of zero
  - Average pooling - the error is multiplied by $1/(N \times N)$ and assigned to the whole pooling block (all units get this same value)

Example of forward and backpropagation pass for a sample CNN

The sample CNN



Parameter initialization

- **C1 layer**: $k^1{}_{1,p}$ (size 5x5) and $b^1{}_p$ (size 1x1), p=1,2,…,6
- **C2 layer**: $k^2{}_{p,q}$ (size 5x5) and $b^2{}_q$ (size 1x1), q=1,2,…,12
- **FC layer**: W (size 10x192) and b (size 10x1)
- All bias $b^1{}_p$, $b^2{}_q$ and b are initialized to zero. The others are drawn randomly from a uniform distribution defined based on the kernel size and number of input and output maps on corresponding layers

$$k^1_{1,p} \sim U\left(\pm\sqrt{\frac{6}{(1+6)\times 5^2}}\right) \qquad k^2_{p,q} \sim U\left(\pm\sqrt{\frac{6}{(6+12)\times 5^2}}\right)$$

$$W \sim U\left(\pm\sqrt{\frac{6}{192+10}}\right)$$

where $U(\pm x)$ denotes a uniform distribution with upper and lower bounds of $\pm x$.

Totally, the number of parameters is:

$$(5 \times 5 + 1) \times 6 + (5 \times 5 \times 6 + 1) \times 12 + 10 \times 192 + 10 = 3898$$

Convolution layer C1

$$C_p^1 = \sigma(I \circledast k_{1,p}^1 + b_p^1), \text{ where } \sigma(x) = \frac{1}{1+e^{-x}}$$

$$C_p^1(i,j) = \sigma\left(\sum_{u=-2}^{2}\sum_{v=-2}^{2} I(i-u, j-v)k_{1,p}^1(u,v) + b_p^1)\right)$$

where p=1,2,…6 because there are 6 feature maps on C1 layer, and i, j are row and column indices of the feature map. Only keeping those parts of the convolution that are computed without the zero-padded edges, the size of C^1_p is 24×24, rather than 28 × 28 like I

Pooling layer S1

$$S_p^1(i,j) = \frac{1}{4} \sum_{u=0}^{1} \sum_{v=0}^{1} C_p^1(2i-u, 2j-v) \quad i,j = 1,2,\ldots,12$$

Convolution layer C2

$$C_q^2 = \sigma\left(\sum_{p=1}^{6} S_p^1 \circledast k_{p,q}^2 + b_q^2\right)$$

$$C_q^2(i,j) = \sigma\left(\sum_{p=1}^{6}\sum_{u=-2}^{2}\sum_{v=-2}^{2} S_p^1(i-u, j-v)k_{p,q}^2(u,v) + b_q^2\right)$$

where q=1,2,…12 because there are 12 feature maps on C1 layer. Only keeping those parts of the convolution that are computed without the zero-padded edges, the size of $C^2{}_q$ is 8×8, rather than 12 × 12 like $S^1{}_p$

Pooling layer S2

$$S_q^2(i,j) = \frac{1}{4}\sum_{u=0}^{1}\sum_{v=0}^{1} C_q^2(2i-u, 2j-v) \quad i,j = 1,2,\ldots,4$$

Vectorization and Concatenation

- Each S^2_q is a 4x4 matrix, and there are 12 such matrices on the S2 layer
- First, each S^2_q is vectorized by column scan, then all 12 vectors are concatenated to form a long vector with the length of 4x4x12 = 192
- We denote this process by:

$$f = F(\{S_q^2\}_{q=1,2,\ldots,12})$$

and the reverse process is:

$$\{S_q^2\}_{q=1,2,\ldots,12} = F^{-1}(f)$$

Fully Connected Layer FC

$$\hat{y} = \sigma(W \times f + b)$$

Loss function

- Assuming the true label is y, the loss function is express by:

$$L = \frac{1}{2}\sum_{i=1}^{10}(\hat{y}(i) - y(i))^2$$

Backpropagation

- In the backpropagation, we will update the parameters from the back to start, namely W and b, $k^2_{p,q}$ and b^2_q, $k^1_{1,p}$ and b^1_p

Backpropagation: ΔW (size 10x192)

$$\Delta W(i,j) = \frac{\partial L}{\partial W(i,j)} = \frac{\partial L}{\partial \hat{y}(i)} \frac{\partial \hat{y}(i)}{\partial W(i,j)}$$

$$= (\hat{y}(i) - y(i)) \frac{\partial}{\partial W(i,j)} \sigma \left(\sum_{j=1}^{192} W(i,j) \times f(j) + b(i) \right)$$

$$= (\hat{y}(i) - y(i))\ \hat{y}(i)(1 - \hat{y}(i))\ f(j)$$

- Let $\Delta\hat{y}(i) = (\hat{y}(i) - y(i))\ \hat{y}(i)(1 - \hat{y}(i))$, whose size is 10x1, then

$$\Delta W(i,j) = \Delta\hat{y}(i)\ f(j) \implies \Delta W = \Delta\hat{y} \times f^T$$

Backpropagation: Δb (size 10x1)

$$\Delta b(i) = \frac{\partial L}{\partial b(i)} = \frac{\partial L}{\partial \hat{y}(i)} \frac{\partial \hat{y}(i)}{\partial b(i)}$$

$$= (\hat{y}(i) - y(i)) \frac{\partial}{\partial b(i)} \sigma \left(\sum_{j=1}^{192} W(i,j) \times f(j) + b(i) \right)$$

$$= (\hat{y}(i) - y(i))\ \hat{y}(i)(1 - \hat{y}(i))$$

$$\Longrightarrow \ \Delta b = \Delta \hat{y}$$

Backpropagation: $\Delta k^2_{p,q}$ (size 5x5)

- Because of concatenation, vectorization, and pooling, we need to compute the backpropagation error ΔC^2_q on C2 layer before calculating $k^2_{p,q}$

$$\Delta f(j) = \frac{\partial L}{\partial f} = \sum_{i=1}^{10} \frac{\partial L}{\partial \hat{y}(i)} \frac{\partial \hat{y}(i)}{\partial f(j)}$$

$$= (\hat{y}(i) - y(i)) \frac{\partial}{\partial f(j)} \sigma \left(\sum_{j=1}^{192} W(i,j) \times f(j) + b(i) \right)$$

$$= \sum_{i=1}^{10} (\hat{y}(i) - y(i))\ \hat{y}(i)(1 - \hat{y}(i))\ W(i,j)$$

$$= \sum_{i=1}^{10} \Delta \hat{y}(i)\ W(i,j)$$

$$\Longrightarrow\ \Delta f = W^T \times \Delta \hat{y}$$

Backpropagation: $\Delta k^2_{p,q}$ (size 5x5) cont'd

- From Slide "Vectorization & concatentaion", we reshape the long error vector Δf (size 192×1) by

$$\{\Delta S_q^2\}_{q=1,2,\ldots,12} = F^{-1}(\Delta f)$$

which gets the error on S2 layer (twelve 4×4 error maps). Because there is no parameters on S2 layer, we do not need to do any derivative stuff. Then, upsampling is performed to obtain the error on C2 layer

$$\Delta C_q^2(i,j) = \frac{1}{4}\Delta S_q^2(\lceil i/2 \rceil, \lceil j/2 \rceil) \quad i,j = 1,2,\ldots,8$$

Note that the size of ΔS^2_q and ΔC^2_q are 4×4 and 8×8, respectively. Now, we are ready to derive $\Delta k^2_{p,q}$

Backpropagation: $\Delta k^2_{p,q}$ (size 5x5) cont'd

$$\Delta k^2_{p,q}(u,v) = \frac{\partial L}{\partial k^2_{p,q}(u,v)}$$

$$= \sum_{i=1}^{8}\sum_{j=1}^{8} \frac{\partial L}{\partial C^2_q(i,j)} \frac{\partial C^2_q(i,j)}{\partial k^2_{p,q}(u,v)}$$

$$= \sum_{i=1}^{8}\sum_{j=1}^{8} \Delta C^2_q(i,j) \frac{\partial}{\partial k^2_{p,q}(u,v)} \sigma\left(\sum_{p=1}^{6}\sum_{u=-2}^{2}\sum_{v=-2}^{2} S^1_p(i-u,j-v) k^2_{p,q}(u,v) + b^2_q\right)$$

$$= \sum_{i=1}^{8}\sum_{j=1}^{8} \Delta C^2_q(i,j)\ C^2_q(i,j)\left(1 - C^2_q(i,j)\right)\ S^1_p(i-u,j-v)$$

- Let $\Delta C^2_{q,\sigma}(i,j) = \Delta C^2_q(i,j)\ C^2_q(i,j)\left(1 - C^2_q(i,j)\right)$, which is actually the error before sigmoid function onC2 layer. Therefore,

$$\Delta C^2_{q,\sigma}(i,j) = \sum_{p=1}^{6}\sum_{u=-2}^{2}\sum_{v=-2}^{2} S^1_p(i-u,j-v) k^2_{p,q}(u,v) + b^2_q$$

Backpropagation: $\Delta k^2_{p,q}$ (size 5x5) cont'd

- Rotating S^1_p 180 degrees, we get $S^1_{p,rot180}$, thus $S^1_{p,rot180}(u\text{-}i,v\text{-}j)=S^1_p(i\text{-}u, j\text{-}v)$. Therefore, $\Delta k^2_{p,q}$ can be expressed:

$$\Delta k^2_{p,q}(u,v) = \sum_{i=1}^{8}\sum_{j=1}^{8} S^1_{p,rot180}(u-i, v-j)\ \Delta C^2_{q,\sigma}(i,j)$$

$$\Longrightarrow\ \Delta k^2_{p,q} = S^1_{p,rot180} \circledast \Delta C^2_{q,\sigma}$$

Backpropagation: Δb^2_q (size 1x1)

$$\Delta b^2_q = \frac{\partial L}{\partial b^2_q}$$

$$= \sum_{i=1}^{8} \sum_{j=1}^{8} \frac{\partial L}{\partial C^2_q(i,j)} \frac{\partial C^2_q(i,j)}{\partial b^2_q}$$

$$= \sum_{i=1}^{8} \sum_{j=1}^{8} \Delta C^2_q(i,j) \frac{\partial}{\partial b^2_q} \sigma \left(\sum_{p=1}^{6} \sum_{u=-2}^{2} \sum_{v=-2}^{2} S^1_p(i-u, j-v) k^2_{p,q}(u,v) + b^2_q \right)$$

$$= \sum_{i=1}^{8} \sum_{j=1}^{8} \Delta C^2_q(i,j)\ C^2_q(i,j) \left(1 - C^2_q(i,j)\right)$$

$$= \sum_{i=1}^{8} \sum_{j=1}^{8} \Delta C^2_{q,\sigma}(i,j)$$

Backpropagation: $\Delta k^1_{1,p}$ (size 5x5)

- Similar to the derivation of $\Delta k^2_{p,q}$, we should first obtain ΔS^1_p, the error on S1 layer. Then, upsampling will be performed to get ΔC^1_p, the error on C1 layer. Finally, following the same way, we can calculate $\Delta k^1_{1,p}$

$$\Delta S_p^1(i,j) = \frac{\partial L}{\partial S_p^1(i,j)}$$

$$= \sum_{q=1}^{12} \sum_{u=-2}^{2} \sum_{v=-2}^{2} \frac{\partial L}{\partial C_{q,\sigma}^2(i+u,j+v)} \frac{\partial C_{q,\sigma}^2(i+u,j+v)}{\partial S_p^1(i,j)}$$

$$= \sum_{q=1}^{12} \sum_{u=-2}^{2} \sum_{v=-2}^{2} \Delta C_{q,\sigma}^2(i+u,j+v) \frac{\partial}{\partial S_p^1(i,j)} \left(\sum_{p=1}^{6} \sum_{u=-2}^{2} \sum_{v=-2}^{2} S_p^1(i,j) k_{p,q}^2(u,v) + b_q^2 \right)$$

$$= \sum_{q=1}^{12} \sum_{u=-2}^{2} \sum_{v=-2}^{2} \Delta C_{q,\sigma}^2(i+u,j+v)\ k_{p,q}^2(u,v)$$

Backpropagation: $\Delta k^1_{1,p}$ (size 5x5) cont'd

- Rotating $k^2_{p,q}$ 180 degrees, we get $k^2_{p,q,rot180}(-u,-v)=k^2_{p,q}(u,v)$. Therefore:

$$\Delta S_p^1(i,j) = \sum_{q=1}^{12} \sum_{u=-2}^{2} \sum_{v=-2}^{2} \Delta C_{q,\sigma}^2(i-(-u), j-(-v))\ k_{p,q,rot180}^2(-u,-v)$$

$$\Longrightarrow\ \Delta S_p^1 = \sum_{q=1}^{12} \Delta C_{q,\sigma}^2 \circledast k_{p,q,rot180}^2$$

- By upsampling, we get the error on C1 layer:

$$\Delta C_p^1(i,j) = \frac{1}{4} \Delta S_p^1(\lceil i/2 \rceil, \lceil j/2 \rceil) \quad i,j = 1,2,\ldots,24$$

Backpropagation: $\Delta k^1_{1,p}$ (size 5x5) cont'd

- Now, we are ready to calculate $\Delta k^1_{1,p}$

$$\Delta k^1_{1,p}(u,v) = \frac{\partial L}{\partial k^1_{1,p}(u,v)}$$

$$= \sum_{i=1}^{24} \sum_{j=1}^{24} \frac{\partial L}{\partial C^1_p(i,j)} \frac{\partial C^1_p(i,j)}{\partial k^1_{1,p}(u,v)}$$

$$= \sum_{i=1}^{24} \sum_{j=1}^{24} \Delta C^1_p(i,j) \frac{\partial}{\partial k^1_{1,p}(u,v)} \sigma \left(\sum_{u=-2}^{2} \sum_{v=-2}^{2} I(i-u, j-v)\ k^1_{1,p}(u,v) + b^1_p \right)$$

$$= \sum_{i=1}^{24} \sum_{j=1}^{24} \Delta C^1_p(i,j)\ C^1_p(i,j) \left(1 - C^1_p(i,j)\right)\ I(i-u, j-v)$$

Backpropagation: $\Delta k^1{}_{1,p}$ (size 5x5) cont'd

- By the same token, rotate I 180 degrees, and let

$$\Delta C^1_{p,\sigma}(i,j) = \Delta C^1_p(i,j)\; C^1_p(i,j)\left(1 - C^1_p(i,j)\right)$$

- Finally

$$\Delta k^1_{1,p}(u,v) = \sum_{i=1}^{24}\sum_{j=1}^{24} I_{rot180}(u-i, v-j)\; \Delta C^1_{p,\sigma}(i,j)$$

$$\Longrightarrow\; \Delta k^1_{1,p} = I_{rot180} \circledast \Delta C^1_{p,\sigma}$$

Backpropagation: $\Delta b^1{}_p$ (size 1x1)

$$\Delta b_p^1 = \frac{\partial L}{\partial b_p^1}$$

$$= \sum_{i=1}^{24} \sum_{j=1}^{24} \frac{\partial L}{\partial C_p^1(i,j)} \frac{\partial C_p^1(i,j)}{\partial b_p^1}$$

$$= \sum_{i=1}^{24} \sum_{j=1}^{24} \Delta C_p^1(i,j) \frac{\partial}{\partial b_p^1} \sigma \left(\sum_{u=-2}^{2} \sum_{v=-2}^{2} I(i-u, j-v)\ k_{1,p}^1(u,v) + b_p^1 \right)$$

$$= \sum_{i=1}^{24} \sum_{j=1}^{24} \Delta C_p^1(i,j)\ C_p^1(i,j) \left(1 - C_p^1(i,j)\right)$$

$$= \sum_{i=1}^{24} \sum_{j=1}^{24} \Delta C_{p,\sigma}^1(i,j)$$

Backpropagation: Parameter update

- We need to set a learning rate $\alpha \in (0,1]$

$$k^1_{1,p} \leftarrow k^1_{1,p} - \alpha \Delta k^1_{1,p}$$
$$b^1_p \leftarrow b^1_p - \alpha \Delta b^1_p$$
$$k^2_{p,q} \leftarrow k^2_{p,q} - \alpha \Delta k^2_{p,q}$$
$$b^2_q \leftarrow b^2_q - \alpha \Delta b^2_q$$
$$W \leftarrow W - \alpha \Delta W$$
$$b \leftarrow b - \alpha \Delta b$$